DOCUMENT RESUME

ED 427 709 IR 019 260

AUTHOR Ip, Albert
TITLE Authoring Educational Courseware Using OXYGEN.
PUB DATE 1998-11-00
NOTE 7p.; In: WebNet 98 World Conference of the WWW, Internet and Intranet Proceedings (3rd, Orlando, FL, November 7-12, 1998); see IR 019 231. Figures may not reproduce clearly.
PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Authoring Aids (Programming); *Computer Software Development; Computer System Design; Computer Uses in Education; *Courseware; Educational Technology; Foreign Countries; Higher Education; Information Technology; *Instructional Design; Instructional Effectiveness; Interaction; Learning Processes; Models; Multimedia Materials; Teamwork; *World Wide Web
IDENTIFIERS University of Melbourne (Australia); Web Pages

ABSTRACT

Engaging learners on the World Wide Web is more than sending Web pages to the user. However, for many course delivery software programs, the smallest unit of delivery is a Web page. How content experts can create engaging Web pages has largely been ignored or taken for granted. This paper reports on an authoring model for creating pedagogically sound courseware components running on the Web server. The design is driven by: (1) the recognition of the team nature of multimedia production efforts; (2) the requirement of a scalable approach with an emphasis on educational quality; and (3) the need to reduce the requirement regarding the computing skills of the content author. The components reported here represent a wide scope of learning paradigms and are based on the author's OXYGEN (Object eXtensible analYsis and Generation of Education coNtent) engine as part of an open, component-based layered course delivery architecture. Current uses of OXYGEN at the University of Melbourne (Australia) are described. Three figures present the digital delivery architecture, conversion of a Web page into an OXYGEN template, and multiple views of data in the OXYGEN template. (Author/AEF)

Authoring Educational Courseware Using OXYGEN

Albert Ip
MEU, The University of Melbourne, Australia
a.ip@meu.unimelb.edu.au

ED 427 709

Abstract

Engaging learners on the web is more than sending web pages to the user. However, for many course delivery software, the smallest atomic unit of delivery is a web page. How content experts can create engaging web pages has largely been ignored or taken for granted. This paper reports an authoring model for creating pedagogically sound courseware components running on the web server. The design is driven by

- the recognition of the team nature of multimedia production efforts,
- the requirement of a scalable approach with an emphasis on the educational quality we endeavor to ensure, and
- the need to reduce the requirement in the computing skills of the content author.

The components reported here represent a wide scope of learning paradigm and are based on the author's Object Extensible Analysis and Generation of Education Content (OXYGEN) engine as part of an open, component-based layered course delivery architecture.

Introduction

From the earliest days of computing, experts have predicted that information technology holds the potential to make major transformations in how people learn [NSF 1995]. The advent of Internet, especially the proliferation of the Web, has led to new ways of doing things. Early attempts to use the web for education exposed many problems. Many researchers have argued that using the new media and technologies in traditional ways is not the answer to "the digital transformation of curriculum" in order to meet the current challenges. One of these challenges arises from the trend of mass education for the Higher Education [Ip 1997]. After years of research, educators are beginning to create a model of learning radically different from the dominant model of the last century. Effective learning is not a passive activity; it is not something just "delivered" to the student, as is assumed in the traditional lecture mode. Learning requires that students think, work with ideas and be actively engaged in their subject materials and the materials' processes [NSF 1995]. There are serious pedagogical issues to be resolved before we can clearly understand what is appropriate and effective for curriculum design in this digital era [Twigg 1996]. Technologically, there are significant issues too. [Roschelle and Kaput 1997] identified that the monolithic approach to educational software has a 'consistent pattern of failure' in providing a cost-effective and scalable solution. [Riesbeck et.el 1995] reported the current problems of tools for authoring education content as follows:

1. Authoring tools are generic programming environments such that an author has to be expert in programming, education, and subject content, all the same time.
2. These tools support very few forms of student input.
3. They are closed systems. Each has its own representational format for subject and pedagogical knowledge and its own set of interface elements.
4. They are rarely matched to student and community needs because development is a one-way process.
5. They are difficult, if not impossible, for a classroom teacher to adapt to the particular needs of a given set of students.
6. They neither take into account nor integrate well into the curriculum and the social structure of the classroom.

The current breed of web-based course delivery software offers little relief. Most of these software provide student management, simple organization and/or navigation of web pages and some form of conferencing. They are electronic books with multiple-choice quizzes packaged with some conference capability. How content experts can create engaging web pages has largely been ignored or assumed. [Ip et al., 1997] discussed the need and design of a technical framework which

1. recognizes the specialised roles of different members in a courseware development team,
2. provides economies of scale and scope, and
3. embraces continuous courseware improvement as a mechanism to extend the useful life of the courseware.

Component-based software architectures, such as Virtual Apparatus (VA) Framework [Ip & Canale 1996] to build engaging interactivity within an educational web pages, are likely to provide a medium to long-term solution to the current problem. However, building every educational web page from scratch, even when most of the education components are re-used, is still a significant effort. To meet the mass education trend and the need to continuously improve the courseware, OXYGEN (Object eXtensible analYsis and Generation of Education coNtent) is created. It provides a mechanism for insulating the content author from the technical details of a highly interactive web page and a database backend for storing the educational content. The ability to extend the features of the templates by server side objects in the OXYGEN engine turns out to be more significant than we originally intended. As we continue to develop OXYGEN, we find that we can build a new pedagogical type of interactive components which cannot be done by components on the client side. This compliments nicely with our efforts basing on the VA framework. This paper discusses the design of OXYGEN and illustrates how OXYGEN is being used in The University of Melbourne.

Brief Outline of the Digital Delivery Architecture where OXYGEN sits

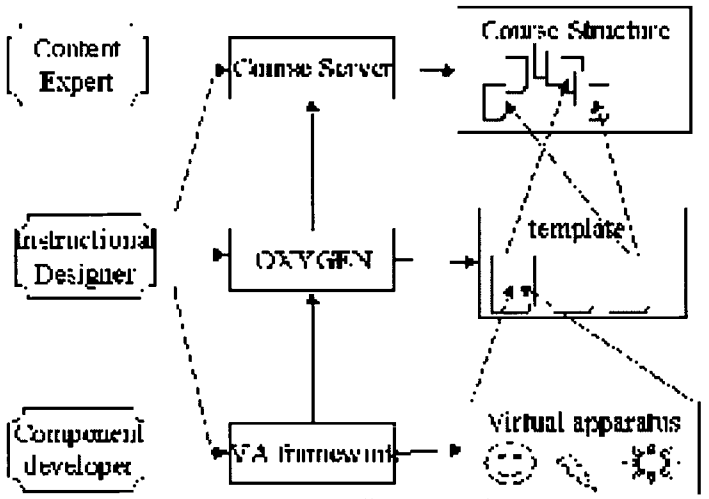

Figure 1: Digital Delivery Architecture

In our approach, content expert, instructional designer and component developer work collaboratively. Component developers represent the lowest tier of development. They are highly skilled technical people creating interactive components (client side components using the VA framework or server side component as OXYGEN templates) for educators' consumption. On the top tier are academics whose expertise lies in content and instruction. They may not have the skill and/or time to work with the technical details of creating the interactive components. Having articulated this dichotomy, we acknowledge the existence of other team members whose role may lie between these two ends and/or people whose interest span the whole continuum.

Typically and being overly simplified, we start by using "scenario capturing" techniques. The content expert articulates how a course will be delivered. Instructional designer converts these scenarios into "use cases", applies his domain expertise in evaluating the pedagogical soundness of the "use cases" and creates templates by recognizing common delivery patterns. Those "use cases" which are not covered by common delivery patterns will be created as standard web pages. The component developers build the web pages and the templates by re-using components, create new component and/or write more HTML codes. These components effectively encapsulate the pedagogical principles of the content expert and instruction designer.

The digital learning architecture reflects our approach. This architecture, of which OXYGEN can be the middle

and/or bottom layer, is the subject of on-going study but beyond the scope of this paper. Briefly, the architecture is three-tiered, each tier being populated with inter-operable components and each tier can be replaced by an engine serving similar functions. The lowest tier of this architecture supporting the higher tiers is the component layer. Our framework for the lowest tier is the VA framework which is essentially a component-based development framework for providing interactivity at the client-side of the delivery [Ip & Canale, 1996; Ip et.el. 1997; Ip & Fritze, 1998]. Our VA components can be built using the common web technologies and are inter-operable, cross-platform and cross-browser. In some situations, these components can be built as OXYGEN templates with additional computational processing on the server side.

In this architecture, prototype web pages are built with or without components. These are converted into templates for the server side OXYGEN engine. The content author interacts with the OXYGEN authoring environment to provide contents and hence built content pages. Content author can also arrange the delivery order of the content pages using the course server.

Virtual Apparatus Framework is for re-use of other people's effort in building interactive functionalities into our own courseware web pages. OXYGEN is about re-using academic's own effort of creating a pedagogically sound content model and additional components requiring the support of the web server. Typically, the content expert designs the content model, commissions the components to be built and then connects with other components to create a prototype page. With the prototype page as a template, the content model can be used in different parts of the course, a different course or to deliver alternate content for mass customization to meet a wide range of student expectations and ability. This systemic approach towards re-use improves our courseware development effectiveness significantly.

Object Extensible Analysis and Generation of Education Content (OXYGEN)

Architecture Description

Although the term "template" is used in this discussion, it must be stressed that the connotation of a boilerplate with no flexibility should not be associated with the word "template" in this context. First of all, the template is created by the academic, with the collaboration of other parties, to address a specific pedagogical need. Secondly, there can be as many templates as needed in a courseware. Thirdly, the OXYGEN templates can allow variations. This approach ensures easy creation of content and/or alternate content delivery.

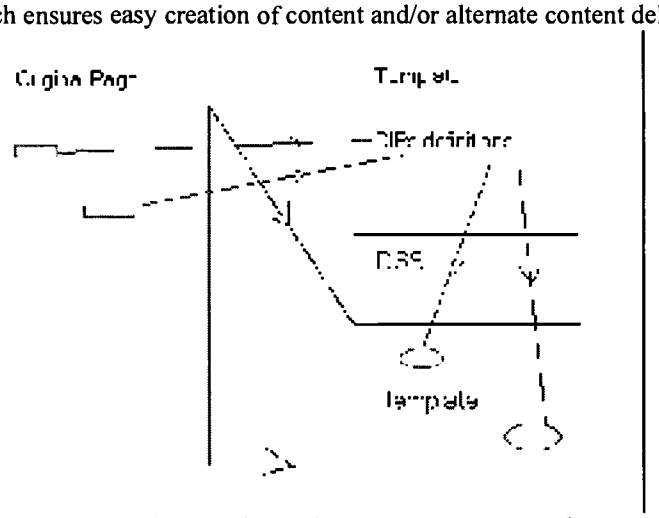

Figure 2: Converting a webpage into an OXYGEN template

In the OXYGEN engine, the fixed (e.g., common elements of the template) and changeable data (e.g., course content) are identified in a Web page. The changeable data components are replaced by Data Insertion Points (DIP) as in conventional template design.

DIPs act primarily as placeholders in the template for content to be inserted. These DIPs are given names so that

they can be individually referred to in different parts of the template. However, in the OXYGEN engine each DIP is supported by a Data Support Structure (DSS) on the server-side.

Among other things, the Data Support Structure consists of three views: the authoring view, what the intended user sees, and data storage.

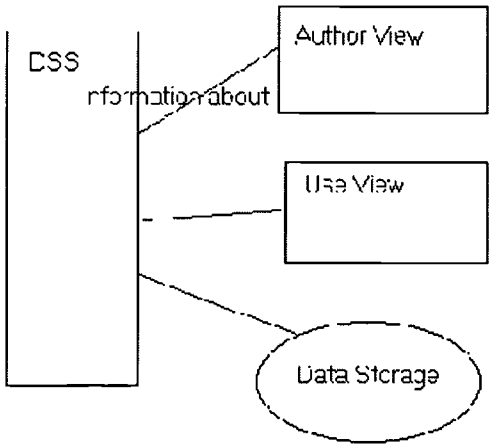

Figure 3: Multiple views of Data in the OXYGEN template

The authoring view stores the visual representation for the content expert. It is typically a block of HTML code of part of a form. It contains instruction to the content expert about what is expected for the DIP in question. Typically, the original data from the prototype page is kept and is used as the default in the authoring view. This provides a good example for the author.

The authoring view is tightly coupled with a data storage instruction. For a simple substitution type of DIP, the data storage can be very simple. However, for a DIP which represents a set of questions, each question is stored as a separate record in the database. In another situation, the data may be a URL and the real data for delivery will be fetched when the template is being used.

The user view takes the data from the data storage and renders it according to the original formatting instruction.

The current implementation of OXYGEN is based on a Windows NT server running IIS 3.0. Because there are server side components, the implementation may not port over to other platforms easily. However, the cross-platform requirement may not be an issue for authoring and delivery of web courses as they are a one time capital investment.

OXYGEN in Action

At the time of this writing, the authoring interface is a form-based Web interface consisting of "author views" of some of the DIPs found in an OXYGEN template. Some of DIP may not have an "author view". Other forms of authoring interface can be created by virtue of OXYGEN's object-oriented nature.

As the content expert enters data into the authoring form, the data is stored at a database. Special mechanism is available to enter a "rule" of finding the data instead of the actual data. This allows alternate content delivery as discussion in a later section.

When it comes to the time a page based on one of the templates is to be delivered, the web server will either fill in the DIPs with specific content just before delivery. The content can be fetched either directly from the database or found using the rule as specified.

Current uses OXYGEN

NALSAS

OXYGEN is being used as the main delivery mechanism for NALSAS project. NALSAS (National Asian Languages and Studies in Australian Schools Taskforce) project is a multimedia course to teach Mandarin to teachers via the Web. The NALSAS course is divided into four topic areas. Each topic area is further divided into three modules. Each module has two sets of instruction content, a magazine and a set of assessment activities. One typical use of the power of OXYGEN is in one of the assessment for all the instruction content (24 in total). The content expert has determined that for retention of the new vocabulary learnt in the set, a flash card exercise will be used. Since the new vocabulary being learnt is stored in the database for other purposes, instead of specifying the vocabulary for the 24 exercises, a rule is used in the flashcard template.

The navigation in this project is also interesting. By retrieving the last location of a student, OXYGEN customizes the navigation to highlight the next topic the student should take.

LEO

Learning Evaluation Online (LEO) [Ip & Kennedy, 1998] is an on-line service available via the Web for developing, constructing, delivering survey on-line. Our evaluation of LEO has been very positive.

The template has five parts. In the first four parts the survey author inserts data into an introduction, the objectives of survey (optional), the respondent's bibliography (optional), and specifies the survey questions. The final part provides a unique ID to identify each respondent.

From the implementation point of view,

- The introduction. The survey designer provides a brief description and purpose of the survey. This is a simple substitution type of DIP. The introduction entered by the survey author is stored directly in the database. When the survey is delivered, the introduction DIP is the data in the database.
- The survey objectives. A hypertext link to the survey objectives. This is a typical example of "referencing" data. The actual data inserted at the DIP is another URL which access the real data. In this implementation, it is another LEO template.
- The bibliography. By selecting some check boxes, the survey author generates a set of questions (e.g., name, course number, etc.) to be completed by the survey respondents. This is a situation where the authoring view is very different from the use view. Instead of asking the survey author to key in some preset items, a simple selection makes life much easier.
- The survey questions. There are several question templates available— e.g. a Likert-style question, true-false type question, and open-ended question. The number and content of survey questions are determined by the author during survey design time. Hence, in the LEO template, the survey question DIP represents a set (which may have more than one item) and is further complicated by the available question types.
- The Unique ID. This is generated when a respondent submits a survey. This is a typical "late binding" DIP.

The implementation of LEO using OXYGEN is interesting in the sense that it requires most of OXYGEN's capability and illustrates the flexible nature of the OXYGEN engine.

LEO is being used in several projects for evaluation, such as the Interactive Graphing Tool project [Kennedy & Fritze, 1998], CyberShakespear, NALSAS, Survey of Student Expectations About Carrying Out Assignments.

ANN

Annotation (ANN) allows web page owners to set up a facility for readers to annotate web pages. All annotation will be stored on our database and become accessible to other readers. ANN can be treated as a simple conferencing tool with a single thread and focuses on the topic in discussion. It is modeled as a Post-it sticker on the web page.

TAO

Text Analysis Object (TAO) is a new class of tool to analysis free text responses from learners. The answer consists of concepts and details which are scored separately. Feedback is provided to the student to help him/her to get the concepts and details correct. This is a reflective tool when the learner is encouraged to spend some time to think about a specific question and try to generate the best answer to a question through repeated attempts.

QPLUS

Question Plus (QPlus) allows a teacher to mimic and improve a classroom situation. In a real classroom, when a teacher asks a question as soon as one student attempts the question openly, the rest of the class is deprived of the privilege of attempting a "first-time" answer. QPlus enables the content author to pose a question and students will be able to see other's response only after they have submitted their version of the answer.

Conclusion

Mainstreaming digital education is costly and labour intensive. The work described here is an attempt to contain the cost and re-use the hard work various parties have put into the task. We need a mechanism to shield the technical complexity of the development of highly interactive courseware from the content author. This can enable the content author to concentrate on creating the content and can help to ease the already too heavy workload from the academic. With this, we hope to see more adaptive interactive courseware being produced. The tools (LEO, ANN, TAO and QPLUS) are available for trial and use by accessing this url:
http://www2.meu.unimelb.edu.au/oxygen/tools/

Reference:

[Fritze & McTigue 1997] Fritze P, McTigue P, 1997. "Learning Engines - A Framework For The Creation Of Interactive Learning Components On The Web" presented at ASCILITE 97.
[Ip, 1997] Ip, 1997. Feature of the Week article (17th Nov, 1997) at Education Object Economy "Higher Education & Web-based Learning: Five Challenges" online http://trp.research.apple.com/or http://www2.meu.unimelb.edu.au/virtualapparatusframework/papers/eoe.html
[Ip & Canale, 1996] Ip and Canale, 1996 "A model for Authoring Virtual Experiment for Web-based courses" presented at ASCILITE 96. Online
http://www2.meu.unimelb.edu.au/virtualapparatusframework/papers/ascilite96.html
[Ip & Canale, 1997] Ip & Canale, 1997. "Supporting mainstream adoption of digital technology using the "virtual apparatus" Model for Courseware Development", 15th August, 1997. Online
http://www2.meu.unimelb.edu.au/virtualapparatusframework/papers/onlineEd.html
[Ip et al. 1997] Ip, Canale, Fritze and Ji, 1997. "Enabling re-usability of courseware components with Web-based 'Virtual Apparatus'" presented at ASCILITE 97. Online
http://www2.meu.unimelb.edu.au/virtualapparatusframework/papers/ascilite97.html
[Ip & Fritze 1998] Ip & Fritze 1998. "Supporting component-based courseware development using the Virtual Apparatus Framework Script" presented at EdMedia 98.
[Kennedy & Ip 1998] Kennedy & Ip 1998. "Flexible, customisable evaluation online: An evaluation tool called LEO" presented at CALISCE'98
[NSF 1995] National Science Foundation 1995, NSF Educational Technology Workshop (Setting a Research Agenda for Computer Science in Education Technology), online http://www.cc.gatech.edu/gvu/edtech/nsfws/.
[Reisbeck 1995] Reisbeck et al 1995, Tools for Authoring Educational Technology, in NSF Educational Technology Workshop, online http://www.cc.gatech.edu/gvu/edtech/nsfws/tools.html [Roschelle & Kaput] Roschelle J, Kaput J. "Educational Software Architecture and Systemic Impact: The Promise of Component Software" online http://www.simcalc.umassd.edu/simcalc/library/S&SImpact.hqx
[Twigg 1996] Twigg, Carol (1996). "Academic Productivity: the case for Instructional Software" online http://ww.educom.edu/program/nlii/keudocs/boardmoor.html

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

REPRODUCTION BASIS

☒ This document is covered by a signed "Reproduction Release (Blanket) form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

☐ This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").