DOCUMENT RESUME

ED 428 662 IR 019 323

AUTHOR Fritze, Paul; Ip, Albert
TITLE Learning Engines - A Functional Object Model for Developing Learning Resources for the WWW.
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. Figures may not reproduce clearly.
PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; Computer Interfaces; *Computer Mediated Communication; *Courseware; Educational Technology; Foreign Countries; Higher Education; *Instructional Design; *Integrated Activities; Learning Activities; *Material Development; Models; Resource Materials; Teacher Developed Materials; Teacher Role; Tutoring; World Wide Web
IDENTIFIERS Learning Environments; Object Orientation; *Technology Integration; University of Melbourne (Australia)

ABSTRACT

The Learning Engines (LE) model, developed at the University of Melbourne (Australia), supports the integration of rich learning activities into the World Wide Web. The model is concerned with the practical design, educational value, and reusability of software components. The model is focused on the academic teacher who is in the best position to conceive and apply novel learning objects (e.g., visualization, simulation, dialogue shell, interface) to meet particular discipline learning requirements. A key component of the model is the ability to engage the learner in tutorial-style dialogue. By incorporating other content resources and customized interface objects, the learning environment can be effectively extended. LE objects operate within an open, scalable technical framework that provides mechanisms for inter-object communication, database management, delivery, and authoring. This paper describes the LE model, including the Tutorial Itemset (a dialogue shell) and design issues, and the LE technical structure, including the communication framework, script, and Director core libraries. Future directions are also addressed. Three figures present: the Tutorial Itemset object displaying one of a set of question items; objects combined to create a tutorial dialogue centered around a simple visualization; and the Graphing Engine input device as focus of a tutorial dialogue. (Author/DLS)

Learning Engines - a Functional Object Model for Developing Learning Resources for the WWW

Paul Fritze
Multimedia Education Unit, The University of Melbourne, Australia
email: p.fritze@meu.unimelb.edu.au
Web: http://www2.meu.unimelb.edu.au/le

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

Abstract: The Learning Engines model is about the integration of rich learning activities into the WWW. It is concerned with the practical design, educational value and re-useability of software components. The model is focused on the academic teacher who is in the best position to conceive and apply novel learning objects to meet particular discipline learning requirements. A key component of the model is the ability to engage the learner in tutorial-style dialogue. By incorporating into this other content resources and customised interface objects, the learning environment can be effectively extended. Learning Engine objects operate within an open, scalable technical framework that provides mechanisms for inter-object communication, database management, delivery and authoring.

Background

The University of Melbourne is, like many other institutions around the world, re-thinking its approach to teaching in the light of changing economic and societal factors. There is a particular focus on restructuring whole curricula, supported in part by funding incentives for projects integrating educational technology into courses. The Multimedia Education Unit plays a central role in this process providing academic and technical support to departments and faculties. It is from this position that we perceive a need to enhance the often rather didactic delivery of information often associated with the World Wide Web (WWW). We are particularly interested in developing conditions for the active involvement of students, the process of 'doing', reflecting and of iterative adjustment of understandings. In this light we are seeking to extend the conversational nature of the teaching-learning process with some form of dialogue between the student and teacher, in this case by way of intermediary WWW-based software objects. The purpose of the model is not to provide the complete on-line interactive system. Rather, it is a practical opportunity for teaching staff to add rich and customised activities to their courses to complement existing on-line and traditional modes of learning. There are many solutions already available to efficiently deliver banks of multiple choice questions for example. The activities we are proposing however, with careful design and application, do have the potential to provide the student with a more engaging experience than might otherwise occur.

Despite an expanding collection of interactive WWW resources such as visualisations, simulations and tutorial systems, we see the continued need to customise on-line activities to meet the cultural, institutional and pedagogical positions of local teaching staff. From the technical development viewpoint, many challenges also remain. Appropriate cross-platform development systems are few in number and relatively immature, the foremost contenders being Java and Macromedia Shockwave. In the light of the traditional CAL approach within the University environment, we have initially employed, but are not restricted to, Shockwave for Director. Shockwave objects operate within a standard WWW browser page by way of an extension 'plugin'. They can offer a high degree of interactivity, control over various forms of media and are compact in size making them ideal for on-line delivery [Macromedia 1997].

The Learning Engines model

The Learning Engines (LE) model is an object-based approach to thinking about, constructing and interconnecting educational objects. By 'object-based' we mean here modular software objects that can be independently developed to certain technical and functional standards and set up to operate collaboratively. The model is not dependent on any single object, any object can be replaced, with the value coming from the combination. This value is both in effectiveness of learning and the ability to reuse resources in different teaching contexts.

Four broad classes of object in the LE model have been described [Fritze & McTigue 1997]. These include client-side *visualisations* such as animations, hyperlinked data sets, movies or other representations of information. *Simulations* may be pieces of virtual equipment or graphic representations of more complex relationships in which parameters can be adjusted, providing a higher degree of intrinsic feedback and in which there are multiple 'dimensions' for the student to explore. For both these types of resource, consideration should be given as to how they are employed in teaching. Unless a learning context is made clear, the student experience may well be one of knowledge reception or simply unfocussed exploration, rather than constructive engagement in learning activity. This context may be established by direct involvement of the teacher, by integrating the resource into other classroom activities or by providing guidance in some form of supplementary instructions. In the LE model, we introduce the idea of constructing an on-line dialogue between the student and the resource using a third class of object, the *dialogue shell.* A dialogue can be constructed by each teacher to provide an extended sequence of tutorial-style learning interactions to guide the student through a graded set of activities, provide immediate remedial feedback and give the teacher indication of problems and progress of the student. The 'shell' object is configured with content information by the teacher using a separate script template described below. Input or *interface objects* can extend the capabilities of a dialogue shell with customised and novel forms of interface to the subject matter. They could, for example, provide virtual 'pens' and 'papers' optimised for different subjects. It is worth briefly describing the main dialogue shell object and how it can be applied to other LE objects to create rich learning activities.

The Tutorial Itemset - a dialogue shell

The Tutorial Itemset is an instance of a dialogue shell with which we can create a sequence of question items for the student [Fig. 1]. It is based on the TutorialTools instructional authoring system used since 1993 to create over 80 hours of undergraduate CAL workshop materials for the School of Chemistry [McTigue et al 1995]. An earlier version of the WWW dialogue shell was described in [Fritze 1996]. It supports a variety of question styles and multiple levels of student feedback. Students can also register comments at any stage. Multiple assessment criteria can direct feedback according to subtleties in a student's response and the interaction styles have been extended to address the difficulty of open-ended responses.

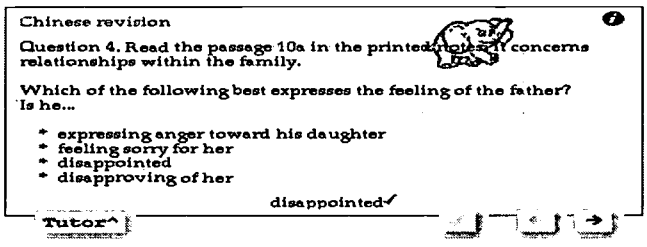

Figure 1: The Tutorial Itemset object displaying one of a set of question items

In this example the Itemset object is operating as an independent object, under the direction of a question script. The real power of the LE model is in being able to extend this interaction with other LE objects to provide a richer and more customised environment. This has been described in [Fritze & McTigue 1997] and [Kennedy et al 1997] but it is worth summarising with two examples.

The example in [Fig. 2] illustrates how a very simple resource in veterinary anatomy can be used as the focus of a tutorial dialogue. The WWW page contains two LE objects, on the left a simple object, essentially a digital video of a leg that the student can position to various angles, and on the right, the regular Itemset dialogue object. The first question item requires that the leg is moved to full extension, subsequently the position of the leg is read by the Itemset as the answer to the question. The next question item not shown here will physically position the leg to a particular angle and pose a standard multiple choice or text entry question based on the indicated circumstance. As a simple visualisation of a body function the Leg resource is probably of limited value by itself. By using it as an extension interface to the Itemset, we have the ability to more fully explore the issues concerning this particular body system.

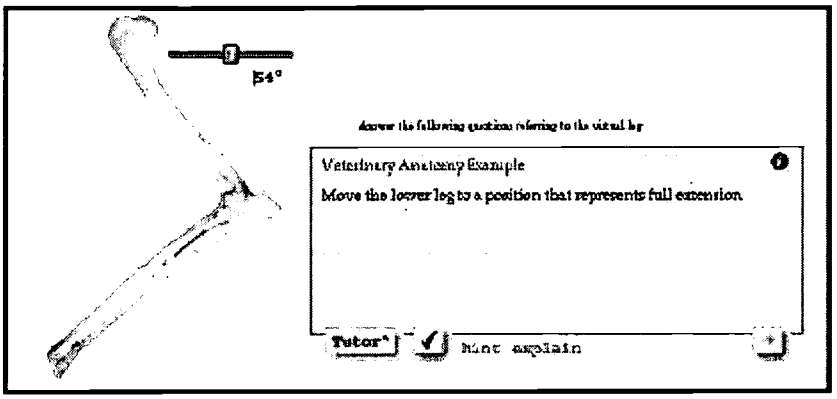

Figure 2: Objects combined to created a tutorial dialogue centred around a simple visualisation

The Interactive Graphing Object in [fig. 3] is another example of an interface component that extends the capacities of the general purpose tutorial object [Kennedy & Fritze 1997]. With this the student can sketch a curve on 'graph paper'. The attempt is also classified using a three point Bezier interpretation and reported to the Itemset. The Itemset can also specify curves to be plotted in the graph.
In the example below, the Itemset and Graph object have been combined within a WWW page. A curve has been displayed on the graph as directed by the script of the Itemset. The student's sketched response has failed one of the Itemset criteria conditions which returns an appropriate comment in response. The level of detailed question and response using such combined objects is potentially very high and applicable across a wide variety of disciplines.

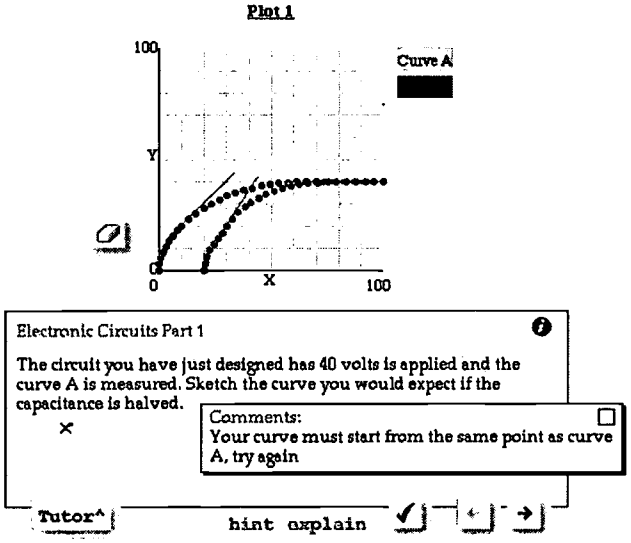

Figure 3: Graphing Engine input device as focus of a tutorial dialogue

Additional components under development in 1998 include interfaces and objects for open-ended text, simple algebraic input, image labelling, constructing images from picture elements, concept mapping, simple film storyboarding, a virtual dog for veterinary anatomy, a digital video front end and various language exercises.

Design Issues

The main focus of LE object design is directed at the educational level, but supported by an underlying technical and broader course-level framework. This focus addresses most directly the curriculum requirements and it is our intention that the content teacher plays a central role in the conception, functional design, implementation and evaluation of these objects. There are a number of reasons why the LE approach makes this practical. Firstly, as relatively simple resources or extensions to the dialogue shell, LE objects are finite in their functionality. Their purpose and operation are clearly understood by teaching staff who can discuss their requirements with programmers without the need for involved software development or detailed technical specifications. Secondly, the delivery of LE activities and navigation through the course is handled by broader scale WWW course delivery systems, whether a basic WWW server or a commercial delivery system such as WebCT, CREATOR or TopClass. The focus of the teacher remains on the 'coal-face' learning interaction. Thirdly, the construction and operation of LE objects is supported by an underlying communication framework and a library of standard software functions described below.

With smaller, modular constructions has come the opportunity to create objects suiting the purposes of multiple disciplines. The graphing object for example has applications in medicine, economics and the sciences. Techniques for working with open-ended questions apply across all faculties. Consequently we have been conscious of the need for a design methodology to draw together educational ideas and requirements from a wide range of interests. This has involved discussion and negotiation with academic staff from different discipline areas. Although it is beyond the scope of this paper, we draw attention to the technique of scenario-based design. This software design approach is centred on the use of brief descriptions of representative usage sequences as a catalyst in this process. This form of documentation is concise, couched in plain English and can represent the ideas from those with even a minor interest in a project. For more details refer to [Carroll 1995].

We have also found that the attempt to translate existing standalone tutorial exercises into the new Web-based object format, despite an apparent similarity of function, has highlighted differences in the nature of the two approaches. Most obviously, current WWW browsers struggle to run the on-line components as quickly and as seamlessly as their standalone counterparts written in HyperCard, ToolBook or Director etc. Despite having scrolling windows, browsers running on small monitors make it difficult to organise screen layout, with the increase in modularity and generic function also leading to a loss of ability to fine tune and optimise the display. The time to load a page and associated objects across the WWW can be highly variable and is generally much slower than the standalone. Activity grain size is something that should be considered in examining the dynamics of a student's passage through the course materials. In the case of LE components, we have found it most satisfactory when a page contains an extended sequence of activities, rather than just a single question.

Learning Engines Technical Structure

While the main thrust of this paper has been to present a teacher's view of the model and nature of LE objects, a brief summary of the technical structure is given here in terms of the supporting communications framework, the object content script and software core libraries.

Communication Framework

At present LE objects are Director Shockwave movies embedded within an HTML page. Each object has the ability to receive and send commands to and from other LE objects, to report a student response, set a simulation to a particular state or to demonstrate a correct answer. There are two methods in which this is implemented. Initially communication between LE objects as Shockwave movies was achieved using

preferences files of the browser plugin and simple JavaScript functions. A more scalable communication framework is now employed which is described elsewhere [IP et al, 1997]. The VA framework provides the lower level mechanisms for inter-object communication, server database connection and management of object scripts. A key component here is the VAmessenger in the form of Javascript on the enclosing browser page which directs messages between objects, even those created in different development systems, for example, in Shockwave, Java, Javascript or ActiveX. This underlying framework also provides the glue for integrating LE objects with Web servers, databases and independent course delivery systems.

LE script

Many objects such as the Itemset can be pre-configured with content data. These objects can parse an external script file to update their structure [Fritze, 1996]. The script can be loaded either from a local file or remote database during startup, providing a means of separating and storing content data and for basic authoring. Each object has its own 'vocabulary' of commands and parameters but follows a common syntax. Currently two script formats are supported. The original version is illustrated by the following script extract from the above charting object:

```
graphset
  plot "Plot 1"
    curve "Curve A", "type=growth1, y2=40"
  plot "Plot 2"
    curve "Curve B", "type=line, x1 = 50, y2=50, x2 = 100, y2=100"
  ...
```

The object is therefore able to be 'authored' in relatively plain English language, couched in terms of the teacher's view of the object. In adopting the VA framework as mentioned above and in anticipation of a move to Extended Markup Language (XML) as a standard for WWW communication, we are adapting the parser to the VAscript format described in [Ip & Fritze 1998].

LE Director Core Libraries

LE objects are currently constructed in Director 'Lingo' using an object orientated programming approach. The artefacts seen on screen and the underlying information structure are in fact created at runtime, as the content script is parsed. The software to support this exists in a library of core behaviours common to all LE objects. Creating an entirely new LE object then is greatly simplified and focuses on the creation of parent scripts and special behaviours for the new sub-components of the object. In the above example this would involve the parent scripts for the 'graphset', 'plot' and 'curve' sub-objects, defining their properties, behaviours and screen rendering characteristics. In addition to the parser, the core libraries support inter-object communications and other standard interface functions.

Future directions

The LE model has evolved out of the experiences of working with a number of projects in different disciplines including Physiology, Psychology, English, Chinese, Dentistry, Pharmacology, Economics and Commerce, Genetics, Chemistry, Veterinary Anatomy, Medicine and others. It is intended that departments will use the pool of LE objects as a shared resource and be able to develop at least simple additional objects to meet their own requirements. Evaluation of LE content materials for a number of these projects is being undertaken.

Administrative supports such as improved authoring tools, feedback and progress reports are being developed in conjunction with the VA framework and will be extended as the projects are implemented during 1998. The development of authoring layers with which components can be more easily configured by teaching staff remains a top priority and the subject of current work.

Integration into WWW-based courseware systems will provide more generalised solutions for whole course delivery, computer mediated communication, administration and assessment etc. We will be able to report

progress on work with MelbourneIT and their CREATOR course delivery system to incorporate LE components and the VA specification as extensions to that particular delivery system.

Bibliography

[Carroll 1995] Carroll, J. (1995). *Scenario Based Design - Envisioning Work and Technology in System Development*. New York: John Wiley & Sons, Inc.

[Fritze 1996] Fritze, P. (1996). An implementation of interactive objects on the Web, in J. Hedberg & S. McNamara (Eds.). *Learning technologies: Prospects and pathways.* Proceedings of the Biennial Conference of the Australian Society for Educational Technology, EdTech'96, Melbourne, 55-60.

[Fritze & McTigue 1997] Fritze, P. & McTigue , P. (1997). Learning Engines - a Framework for the Creation of Interactive Learning Components on the Web, in R. Kevill, R. Oliver & R. Phillips (eds), *What Works and Why*, Proceedings of the Australian Society for Computers in Tertiary Education Conference, Perth, Australia, 200-206.

[Ip et al 1997] Ip, A., Canale, R., Fritze, P. & Ji, G. (1997). Enabling re-useability of courseware components with Web-based 'Virtual Apparatus', in R. Kevill, R. Oliver & R. Phillips (eds), *What Works and Why,* Proceedings of the Australian Society for Computers in Tertiary Education Conference, Perth, Australia, 286-291.

[Ip & Fritze 1998] Ip, A. & Fritze, P. (1998). Supporting component-based courseware development using Virtual Apparatus Framework Script. To be presented at EdMedia'98.

[Kennedy et al 1997] Kennedy, D. M., Fritze, P. & McTigue, P. (1997). An interactive graphing tool: The meeting of pedagogy and technology, in R. Kevill, R. Oliver & R. Phillips (eds), *What Works and Why*, Proceedings of the Australian Society for Computers in Tertiary Education Conference, Perth, Australia, 331-337.

[Macromedia 1998] Macromedia, Inc. (1998). Macromedia - Shockwave. Available on-line at: http://www.macromedia.com/shockwave.

[McTigue et al 1995] McTigue, P. T., Tregloan, P. A., Fritze, P. A., McNaught, C., Hassett D. & Porter, Q., 1995. Interactive teaching & testing tutorials for first year chemistry. pp. in H Maurer (Ed.). *Educational Multimedia & Hypermedia 1995*. Proceedings of Ed-Media 95 - World Conference on Educational Multimedia and HyperMedia, Graz, Austria, 466-471.

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").