A Learning Technology Application of MIC

Larry Howard, Sandor Palfy, Zsolt Reményi
Institute for Software Integrated Systems
Vanderbilt University

NSF Engineering Research Center for Bioengineering Educational Technologies (VaNTH)

© 2004 Vanderbilt University. All Rights Reserved.
Overview of the Talk

- Part I: The Application
- Part II: Model-Integrated Computing
Learning on the Internet

- **Distance Learning**
  - Interest motivated by convenience
    - “Anything, anytime, anywhere”
  - Focus on self-contained learning “objects”
  - Academic community ambivalent
    - Is it a threat? An opportunity?

- **Blended Learning**
  - Roles for on-line learning within the traditional learning environment
Blended Learning
Adaptive learning environments

Before Class
- Pre-Tests
- Preparatory resources
- Adaptive exercises

In Class
- Traditional sensing (Q&A, quizzes)
- Feedback systems (PRS, VOS, VSAS)

After Class
- Adaptive exercises
- Remedial resources
- Post-Tests

• Pre-Tests
• Preparatory resources
• Adaptive exercises

• Traditional sensing (Q&A, quizzes)
• Feedback systems (PRS, VOS, VSAS)
Authoring On-Line Learning

Aspects

Instructional design

Pedagogy

Integration and validation

Content design and development

Web deployment

Digital media

Technologist

Instructional Designer

Media Designer

Can we expect this from individual faculty?
Roles for Technology

Instructional design
- Support for design patterns (or templates)
- Easily elaborate detailed design

Integration and validation
- Support for design-time checking and testing
- Easily collect and analyze usage data

Content design and development
- Support for content made with familiar tools
- Easily author assessments and dynamic content
Adaptive Courseware: Authoring and Delivery

Authors
- Metadata
- Sequencing Models
- Assessments
- Learning Objectives
- Learning Materials
- Create/Integrate

Instructors
- Rosters
- Delivery Records
- Courseware Assignments
- Experiential Data
- Repository
- Upload

Learners
- eLMS Delivery Platform
- Model-Based Delivery Engine

CAPE Authoring Environment
- Packages
- Courseware
- Security
- Versioning
- Session Mgmt
- Interoperability
- Services
- DHTML+
- Flash
- XML-RPC
- Interfaces
Innovations

- **CAPE** is a general-purpose, visual language environment for authoring interactive, adaptive courseware.
  - Powerful adaptive sequencing capabilities
  - Model abstraction facility for instructional design patterns
  - Integrated authoring for assessments and dynamic content
  - Data modeling facility for adaptations and integration
  - Embedded scripting language
CAPE: A Visual Authoring Environment

- Adaptive delivery
- Learning objectives and domain taxonomies
- Instructional design patterns
- Metadata
- Assessments
Innovations

- **eLMS** is a repository-based, web services based platform for adaptive courseware delivery.
  - Instrumented delivery of CAPE courseware
  - Mining of data from delivery records
  - Integration framework for embedded LTs
  - Administration of classes
  - Reuse of courseware assets
Repository Architecture

eLMS Repository

Authors

- CAPE
- Learning Scientists
- Instructional Designers
- Media Designers
- Media Technologists
- Learning Technologists
- Instructional Designers
- Media Designers
- Media Technologists
- Learning Technologists
- External Web-based Access

- eLMS Librarian
- eLMS Delivery Engine
- eLMS Integration Services

- Embedded Technologies
- FBD
- Indie
- Sask

Repository Architecture
Part II: Application of Model-Integrated Computing
Domain-Specific Visual Language

- **CAPE** is a large visual language
  - 70+ concepts and relationships
  - 2nd generation reduced from 100+

- **Novel aspects**
  - Procedural nature of courseware sequencing models
  - Integration of dynamic programming language
  - Data modeling facility
    - Scope
    - Derived data
  - Positional semantics in assessment authoring facility
Authoring Affordances

- GME extensions in dynamic language (Python)
- Context-sensitive automation
  - GME “add-on” and CAPE wizards
- Context-specific help
- Python dynamic evaluation and import/export
- Library-based asset sharing
- Web services-based integration with delivery infrastructure
Delivery Infrastructure

- Embedded model interpretation
- Dynamic Python evaluation in “sandbox”
- Web services delivery engine interfaces
  - Integration of embedded interactive content
  - CAPE-authored data interchange and computational extensions (Py-lets)
- Delivery records
  - Fine-grained delivery instrumentation
  - Model-based data mining