

©1997 MASTER SERIES

Adaptive Learning Technologies for Bioengineering Education

An Authoring Tool That Puts Educators in the "Driver's Seat" When Creating Individualized, Web-Based Learning Experiences

LARRY HOWARD

Learning technologies employed both inside and outside the classroom are increasingly influencing the nature of teaching and learning. Web-based learning technologies are enabling powerful possibilities for learning activities outside the classroom, both in preparation for in-class activities and in following them up. Of particular importance among these possibilities is the opportunity to address learners as individuals, assessing their strengths and weaknesses and adapting learning activities in response. Over the past few years we have been developing and maturing learning technologies that target these aims in the context of the National Science Foundation's VaNTH (Vanderbilt University; Northwestern University; University of Texas at Austin; and Health, Science and Technology at Harvard/MIT) Engineering Research Center (ERC) (www.vanth.org). In this article we will describe these technologies and discuss their roles in bioengineering education.

The idea of learning technologies that acquire and respond to knowledge about individual learners is certainly not new. Work on intelligent tutoring systems (ITS) [1] can trace its origins back some 30 years [2], and related work in areas such as adaptive hypermedia [3] are extending ITS concepts to Web-based learning experiences. What can be generalized about these kinds of artifacts is 1) that they are custom-crafted using specialty technologies requiring unusual technical expertise, 2) that their development is difficult and expensive, and 3) that evolving them in the face of changing domain knowledge and learner populations requires the continued involvement of their developers.

In pursuing adaptive learning technologies for VaNTH we have chosen a different path. Our primary motivation has been to make the means for authoring such activities accessible to bioengineering educators while providing enough expressive power to enable ambitious applications and the ability to incrementally acquire the needed skills. To address this aim we have created an authoring technology called the courseware authoring and packaging environment (CAPE). This system is described below.

CAPE: An Environment for Authoring Adaptive Courseware

Using CAPE, authors primarily address three interrelated sets of concerns:

- ▶ integrating learning materials into instructional units at various levels of aggregation and determining in what sequences and under what conditions materials are delivered to learners
- ▶ establishing learning objectives for instructional units and associating them with elements of content (or domain) knowledge
- ▶ supplying metadata that describes the materials and instructional units to instructors, to learners, and to the delivery infrastructure.

To address these concerns, CAPE provides a graphical modeling language. In this language, iconic nodes represent authoring concepts, and edges represent various kinds of relationships among these concepts (Figure 1). This visual language is rich, incorporating nearly 90 distinct concepts and relationships. Yet the author can focus on particular tasks (such as defining learning objectives) and CAPE automatically subsets the language to those concepts and relationships appropriate to the task. This feature is called *aspects*. To enable complex representations to be created, the language supports hierarchy; i.e., larger definitional units can be built up from smaller units. To enable reusability, the language supports abstraction and refinement; i.e., definitional units can be used as the starting point for other definitional units, and the latter "inherit" changes from the former. Any definitional unit can serve as the basis for any number of instances of it that likewise inherit changes but cannot be further refined. An abstraction mechanism called *references* allows typed placeholders to be defined that can later be satisfied by instances of the corresponding modeling element. References can be used in conjunction with refinement as a parameterization technique.

None of the above features of the CAPE language were developed specifically for CAPE. Rather, they arise from the fact that CAPE is based on a meta-programmable infrastructure called the generic modeling environment (GME) [4]. We use the metamodeling representation of the GME [5] to define and evolve the CAPE authoring language. From this metarepresentation, GME is specialized into the authoring environment for CAPE. GME provides extensibility features that support the creation of authoring services that are specific to CAPE. This implementation strategy allows us to focus on the authoring language and unique authoring services rather

than on building and evolving the general support environment.

In designing the CAPE language, we have benefitted from a number of external influences. CAPE was initially influenced by the shareable content object reference model (SCORM) [6]. This is a standard courseware representation from the Advanced Distributed Learning Initiative that makes a clear separation between learning content and the description of how the content can be delivered to learners. The design of CAPE also reflects this fundamental separation of concerns that promotes the reusability of learning materials. CAPE supports metadata tagging using the IMS Learning Resource Metadata Specification [7]. CAPE provides assessment-authoring capabilities influenced by the IMS Question and Test Interoperability Specification [8]. In supporting learner profiles in CAPE, we are using the IMS Learner Information Package Specification [9] as an influence. In all of these circumstances we have been less concerned with standards compliance and more concerned with leveraging the valuable insights of talented individuals who have contributed to these standards.

While CAPE started out as a purely visual language, it has evolved into a hybrid language. To increase its expressive power, CAPE has added a general purpose, object-oriented dynamic language called Python [10] as an extension language. This decision reflects several considerations. First, we recognized that certain aspects of authoring adaptive learning activities are very much like programming, and that to support these aspects with a purely visual language would require many extensions to what is already a large visual language. Second, Python is an easy-to-learn language for those with limited programming skills and a very powerful language for those with more advanced skills. Limited users are only re-

quired to know how to write logical expressions in Python, where Python syntax has much in common with other programming languages. Advanced users can employ Python to create powerful reasoning components for deciding how to adapt learning activities in light of what is known about individual learners, the learning situation, or the learning environment. Finally, CAPE developers use Python to create application-specific extensions to the GME, and so this public domain language was already distributed as part of CAPE.

Authoring with CAPE

CAPE is not used for authoring learning content. Web-based learning materials are authored using traditional content authoring tools such as HTML editors, presentation authoring tools, word processors, and simulation or multimedia authoring tools. CAPE only requires that the learning materials involved in its designs be deliverable in a Web browser. Instead, CAPE is principally used to design when, and under what conditions, learning materials are delivered to learners during the course of a learning activity; i.e., to design the form of the instruction. CAPE does provide an intrinsic capability for authoring assessments, since assessments are an important source of information about individual learners that can be used to trigger adaptations. But CAPE is not limited to assessments authored using this intrinsic capability as the exclusive basis for adaptation, as will be discussed later. Otherwise, all content is authored externally to CAPE, and CAPE merely needs to be made aware of the existence of content in order to author adaptive learning activities that involve it.

This leads to two primary use cases for CAPE that we might term *top down* and *bottom up*. In the *top-down* use case, CAPE is used to design the learning activity prior to the creation of learning content. In this case CAPE is purely a visual design representation that can be used to evaluate a design,

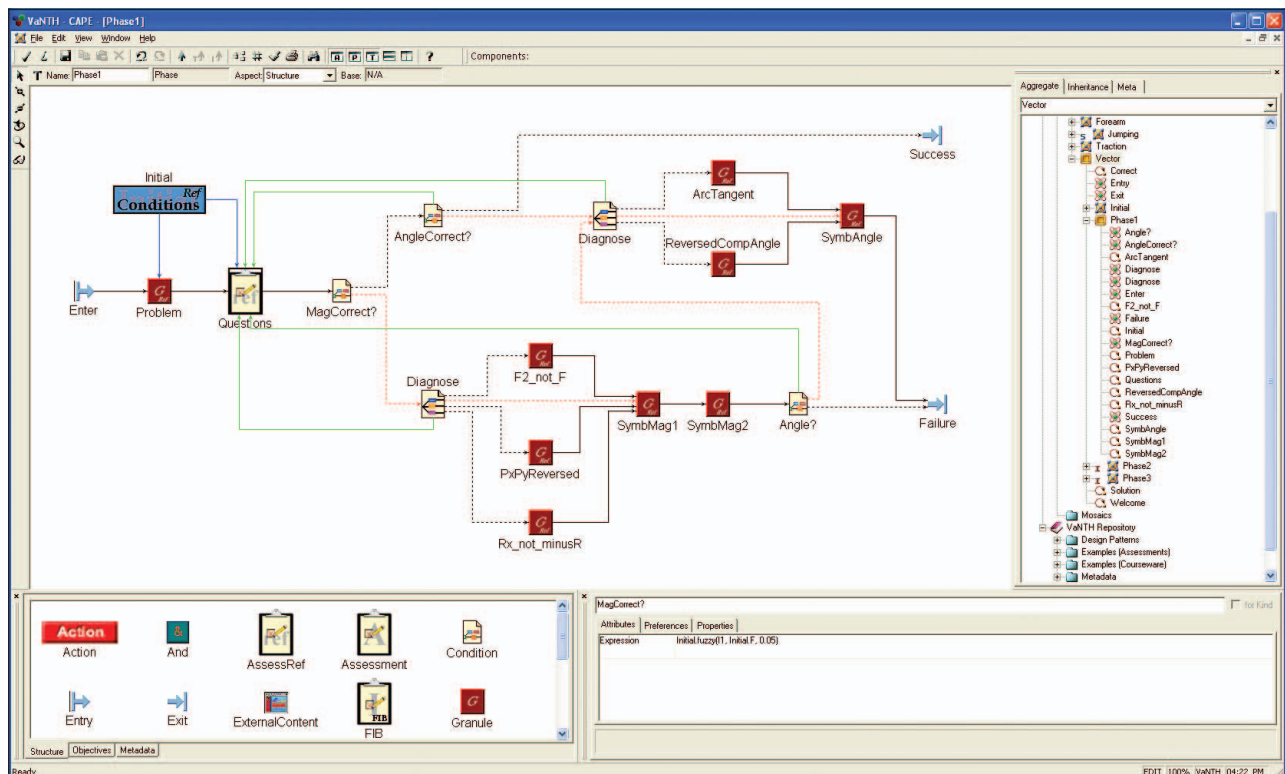


Fig. 1. Adaptive sequencing for a vector arithmetic tutor.

such as through a peer review process. Such designs can be “prototyped” by creating low-quality learning materials for all content elements in the design, and the result can then be “alpha-tested” with actual learners. When the design is accepted, high-quality materials can be created to replace the low-quality materials with no changes to the design itself. The learning activity is then immediately available for final testing prior to delivery in the targeted learning situation. In the *bottom-up* use case, learning materials already exist. This use case typically involves adding interactions (assessments or other kinds of interactive content) and adaptations that improve the efficacy or robustness of the learning activity. These two use cases could occur together in a single application of CAPE. For example, when proceeding from the bottom up, new adaptive paths in an activity might follow the top-down use case. Various iterative design scenarios might also involve the two use cases coincidentally.

The authoring task in CAPE begins by identifying what level of activity is being created. An “aggregation model” defines the basic structural concepts that govern how learning activities are created and assembled. VaNTH has defined an aggregation model that provides four levels:

- ▶ *Granules*—atomic content elements that provide basic resources for authoring. Granules can be used within all higher-level aggregations.
- ▶ *Modules*—the basic instructional unit, where learning objectives address teaching sets of interrelated domain concepts.
- ▶ *Mosaics*—compositions of modules with some unifying theme or challenge.
- ▶ *Courses*—compositions of mosaics or modules intended to provide coverage of some body of knowledge from a bioengineering curriculum.

VaNTH granules correspond to the SCORM content aggregation model (CAM) concept of “assets.” VaNTH modules, mosaics, and courses are compositions that correspond to SCORM’s “shareable content objects.”

Just as in the basic uses cases described earlier, CAPE designs can be elaborated in either a top-down or bottom-up manner. Top-down elaboration begins by designing the highest-level element selected from the aggregation model and then proceeding downwards by designing subordinate elements. Bottom-up elaboration begins by designing the lower-level elements and then assembling them together to form higher-level elements. In the top-down approach, granules can be defined using placeholders called “references.” In the bottom-up approach, a set of granules can be defined initially and then used by reference as needed in any number of higher-level aggregations.

VaNTH is interested in the disciplined application of learning science in the creation of its learning experiences, particularly the “how people learn” (HPL) framework [11]. This framework provides a macrostructure for attributes of effective teaching and learning experiences and environments that informs pedagogical styles and instructional designs. In CAPE, the latter can be represented as canonical design forms that capture recurring pedagogical strategies using abstraction facilities provided by the GME described earlier. The resulting *instructional design patterns* are abstract models that authors refine to ultimately form instances with particular learning content. Such patterns provide contexts for extended

scaffolding of the authoring task and can be organized into libraries with their supporting resources.

In CAPE designs, simple sequencing relations define the order of delivery of content elements and higher-level aggregations using typed edges in a directed graph. These graphs define basic ordering relations, where the graph $A \rightarrow B$ means that access to the leaning resource B is contingent on the completion of the resource A. Branching in such graphs supports learner path selection; i.e., the learner must choose from a list of available alternatives defined by the branches. Logical connectives and set relationships influenced by SCORM add expressiveness to these simple sequencing designs.

Adaptive sequencing is based on conditional delivery predicates. These predicates are defined using two CAPE authoring concepts, *condition* and *select*. The former is a binary test-and-branch concept with the predicate associated with the node and branching via outbound *true* and *false* edges. The latter is a multiway branching concept with delivery predicates associated with individual outbound edges, together with an *else* edge that is followed when none of the predicates defined by conditional edges are true.

The conditional delivery predicates employed by these concepts are logical expressions written in the Python language. The context (namespace) for evaluating these predicates is defined by authors in the design representation. Two primary sources for terms are available in defining these namespaces. The first is the set of student responses to any previously delivered assessments. The second is a general-purpose data container called a *condition set*, described below. Any number of assessments or condition sets can establish the namespace for evaluating the delivery predicate(s) of a particular conditional delivery concept.

Condition sets are a general-purpose data and computation definition facility in CAPE. The facility supports simple and composite data definitions. Simple data types include binary and character strings, integers, floating-point numbers, booleans, and dates. The values of integers and floating-point numbers can be randomized. Composite types include arrays and dictionaries that can be used recursively to define “deep” composite structures. Condition sets support symbolic values, called *derived conditions*, whose values are defined by means of expressions. Finally, functions can be defined and employed by the expressions of derived conditions or by delivery predicates. The definitions of derived conditions and functions in CAPE condition sets employ the Python language. To support the development of these definitions, the CAPE environment supports the ability to import and export condition sets in a form that enables the use of traditional Python development tools. The environment further provides an extension that can evaluate derived conditions directly.

Condition sets are a dynamic facility within CAPE-authored adaptive learning activities. In addition to defining datasets that can be randomized and rerandomized at delivery time, condition sets provide the means of containing and referencing data created and used by interactive content delivered to learners. Web services available to such content at delivery time support data interchange with condition sets. In this way, condition sets can be used to define arbitrary interfaces with embedded interactive content, and the data supplied by such content can then be evaluated and used to trigger adaptations. A further capability for the dynamic manipulation of condition sets is supported by an authoring concept

called an *action*. This concept allows Python statements to be executed that can alter the values of conditions in condition sets or define new conditions. Actions can be conditional and can be adaptively sequenced. More will be said about these capabilities in describing applications below.

CAPE assessments are sets of questions (items) rendered to learners as structured forms. CAPE employs the aggregation model from the IMS QTI Specification. This model provides assessments composed from sections (and references to sections) composed from items (and references to items). CAPE supports the creation of “banks” (or libraries) of items and sections that can be used by reference. Three item types defined by the IMS QTI Specification are currently supported:

- ▶ *fill in the blank* (FIB)—where the responses can be strings (single and multiline), integers, and floating point numbers
- ▶ *true/false* (TF)—where the “true” and “false” labels can be changed to support an arbitrary two-choice response
- ▶ *multiple choice* (MC)—where choices can be text or images.

Assessments authored with CAPE are both *dynamic* and *adaptive*. They are dynamic in that any text anywhere in an assessment, section, or item can be dynamically generated by reference to learner responses from earlier assessments or the value of any condition in a condition set. This extends to the entire question text of an item, so that entire questions can be dynamically changed based on current knowledge. Assessments are adaptive in that each section or item can be conditionally delivered by defining delivery predicates in the same manner, and with the same power, as the conditional delivery concepts described earlier. Conditional sequencing of assessments can be used in concert with conditional delivery and dy-

namic content of assessments to achieve very sophisticated adaptations; i.e., “when you ask,” “what you ask,” and “if you ask” can all be conditional on any available knowledge.

CAPE supports context-specific help resources that can be provided as “companions” to any delivered content. The mode of delivery for such resources can be specified and determines whether the resource is delivered in a separate browser window, whose size can be specified, or if it can be delivered directly by a help accessory within the delivery interface. Help resources can be organized hierarchically, and CAPE automatically generates menus for navigating this hierarchy.

So far our description of CAPE has focused on authoring support for the adaptive sequencing of learning activities. While these capabilities are a particular focus of CAPE, they are not the exclusive focus. In the next section, we describe the support CAPE provides from specifying descriptive information.

Descriptive Tasks with CAPE

CAPE allows authors to define learning objectives and specify metadata for learning activities and their constituent elements and materials. These capabilities are important for providing various kinds of descriptive information that can inform others, both humans and machines, about the learning activity.

The CAPE language defines means for creating collections of descriptive resources. VaNTH’s domain taxonomies and IMS metadata tags are examples. These are represented through sets of tag types, and instances of these types are used to create palettes of tagging resources. Authors drag instances of these tags on to a modeling canvas and associate them with other model elements. Attributes are specified, where required, to complete tag instances.

In the *objectives* aspect of CAPE models (Figure 2), authors define learning objectives and associate these with con-

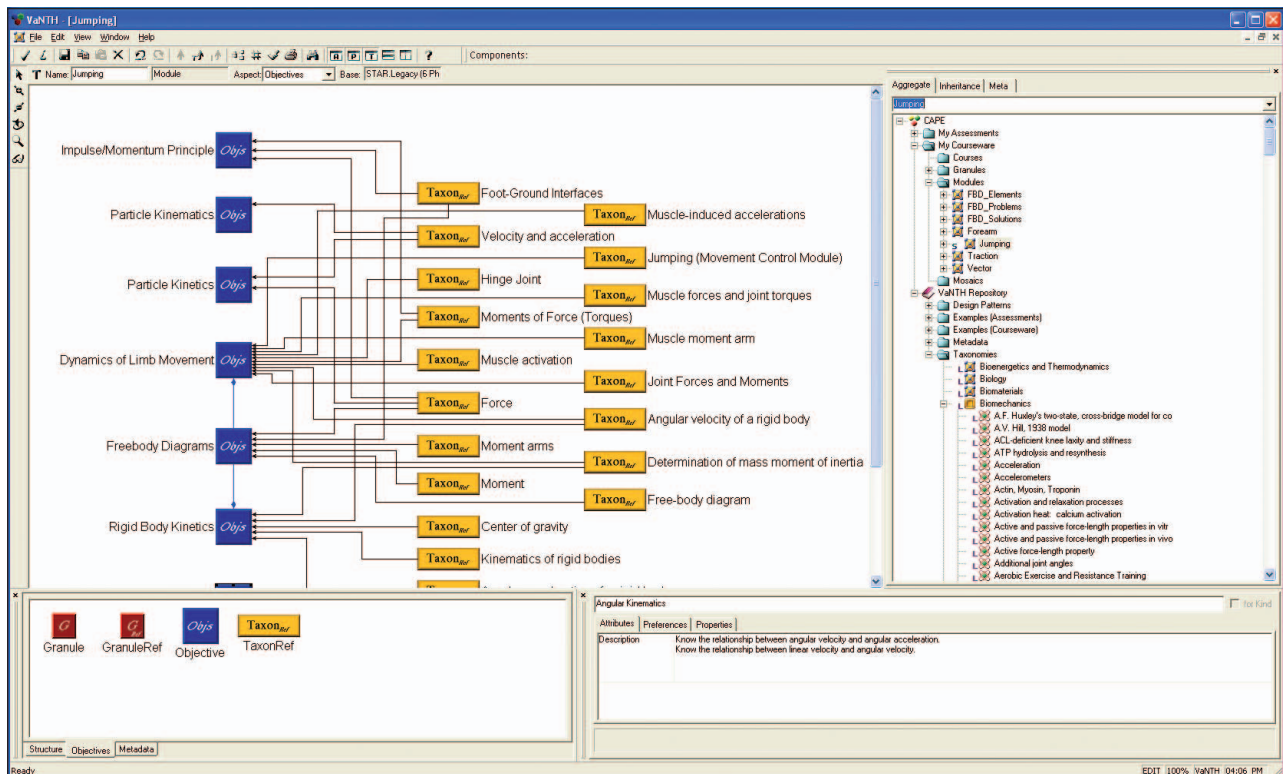


Fig. 2. Learning objectives associated with domain knowledge for a module on jumping.

CAPE is principally used to design when, and under what conditions, learning materials are delivered to learners during the course of a learning activity; i.e., to design the form of the instruction.

cepts from VaNTH's domain taxonomies. Objectives can be arranged into hierarchies where lower-level objectives contribute in some way to the attainment of higher-level objectives. The association of objectives with domain concepts merely signifies that an achieved objective contributes to the learner's knowledge of the concept. While such associations enable some basic auditing capabilities to map learning activities to, say, curricular models, we anticipate that more sophisticated associations will probably emerge from future work.

Metadata tags are elements of some descriptive framework that organize information that authors must or can supply about learning materials or learning activities. CAPE's metadata tag types are based on whether a tag is mandatory or optional and whether the tag can occur singly or multiply within such a framework. Using these types, palettes of tags have been created based on the IMS Learning Resource Metadata Specification as descriptive resources for use in CAPE.

One leverage point that CAPE has for such descriptive tasks concerns the use of instructional design patterns, described earlier. Metadata specifications can be made for such patterns where some tags are completed while other tags are left to be completed. As more abstract patterns are refined into more concrete patterns and instances, the metadata specifications can likewise be refined. Further, CAPE provides a "wizard" framework that can be used to reinforce these metadata specification tasks when instructional design patterns are reused.

When examining a rich authoring capability like CAPE, it is perhaps easier to communicate ideas through examples than through detailed explanations. In the next section we examine three applications of CAPE that highlight various kinds of capabilities.

Applying CAPE

Pioneering applications of CAPE have played an invaluable role in maturing the technology and in influencing the nature and direction of new capabilities. In this section, we will examine three applications that are somewhat representative of the kinds of applications currently being created with CAPE:

- ▶ a challenge-based inquiry cycle with integral assessments but no adaptive sequencing
- ▶ a vector arithmetic tutor that provides adaptive, progressive remediation
- ▶ a free-body diagram construction activity with embedded diagnosis and feedback.

Challenge-Based Instruction

This activity involves the use of an inquiry cycle grounded on a motivating challenge [12]. The cycle consists of: 1) presenting the challenge, 2) eliciting the learner's initial thoughts on the challenge, 3) presenting ideas from experts and others on aspects of solving the challenge, 4) performing activities and exploring materials that inform the solution to the challenge, 5) synthesizing learning into an approach to solving the challenge, and 6) reporting the solution to the challenge and reflecting on changes from initial thoughts.

This learning activity was authored for an undergraduate biomechanics course, and the challenge concerns how much muscle strength is required for a gymnast to hold the "iron cross" position. It was initially designed for classroom-based instruction, and some interactivity was supported through the use of personal response units provided to the students. Other aspects of the instruction involved asking the students to record information in their personal notebooks and some of this information was later collected. The learning materials were created using Microsoft PowerPoint presented by the instructor in a lecture format.

Since content was already available from the classroom-based version of this module, the task of creating the CAPE-authored version began with preparing this content for Web delivery. A CAPE accessory automates the HTML conversion capability of Microsoft PowerPoint and deaggregates the presentation into individual HTML pages. Granules for this content were created using a CAPE accessory that builds the CAPE representation and supporting metadata.

Challenge-based instruction is a preferred style for VaNTH modules and is supported by an instructional design pattern in CAPE that lays out the basic phases of the inquiry cycle. Within this framework, granules corresponding to some of the presentation slides were selected and sequenced.

The principal effort involved in finishing the module concerned replacing the content that instructed students to answer questions using their personal response units or to record their answers to questions using their notebooks with assessments in CAPE. One of the activities in the fourth phase of the cycle ("research and revise") asked the students to construct a free-body diagram (FBD). Here we reused an embedded FBD editor component (described below in the third example) to capture their diagram. Rather than perform diagnosis and remediation for the student diagram, the follow-up consisted of asking the student to choose from a set of FBDs the one that most resembled what they had constructed. This technique requires the student to first generate a solution prior to presenting them with a choice from alternatives.

CAPE allows authors to define learning objectives and specify metadata for learning activities and their constituent elements and materials.

The resulting Web-based “iron cross” module is approximately equal to the classroom-based module in its interactivity. What is important about the conversion is that in the Web-based version all student responses to interactions are automatically recorded by the delivery platform for later examination by the instructor. The CAPE-authored module can be evolved over time to add more interactivity and adaptations.

Vector Arithmetic Tutor

This activity (a VaNTH module) was created as a prototype for a suite of learning resources that students could use out of class to hone their vector arithmetic skills. The preparation for creating the module consisted of designing the vector arithmetic problem, the diagnostic algorithm, and the remediation. An instructor’s experience with errors students typically make with such problems was used to select the initial set of diagnosed mistakes.

The design called for providing students three attempts at solving the problem. For diagnosed mistakes, students would receive the corresponding remediation. Otherwise, general information about the solving the problem would be provided. The initial conditions for the problem would be randomized, providing a slightly different version to each student.

To support the latter aim of the design, a condition set was devised that symbolically represented the computations involved in solving the problem using derived conditions based on the randomized initial conditions. Additional derived conditions represented the diagnosed student mistakes. The diagnostic algorithm was represented using conditional delivery concepts. CAPE facilities for abstraction were used to replicate the diagnostic algorithm for each of the three attempts, while preserving the ability make changes to just the original.

The content for the activity was also created. Among this content was a Flash animation that visually presented the problem. To respond to the randomization of the problem, this animation used Web services of the delivery platform to retrieve the actual initial conditions from the condition set described above. Dynamic content techniques based on template features of the delivery platform (presented later) were used to make the content sensitive to the randomization, including the fully worked solution presented to learners who failed after three tries. These techniques involve altering the content to insert placeholders that reference initial and derived conditions in condition sets. While effective, this approach requires technical knowledge that we do not expect VaNTH authors to possess. As an alternative, we are investigating adding capabilities to CAPE that will support authoring dynamic

content similar to those currently available for CAPE assessments.

The initial version of this activity provided the same remediation at every attempt. A later refinement provides “progressive remediation”; i.e., less information is provided by the remediation for early attempts and progressively more information for later attempts. No structural changes to the CAPE design were required to implement this improvement. Rather, additional content was created and references to content in earlier attempts were adjusted to “point” to this new content. Additional diagnosed mistakes were also added as part of this refinement in response to initial uses of the activity.

Free Body Diagram Assistant

This is an example of more advanced techniques in CAPE involving embedded component integration and an advanced diagnostic component authored in CAPE/Python. The activity is actually a design pattern, one that can be reused for any number of activities that center on constructing free body diagrams.

The component is a Web-based free body diagram (FBD) editor [13] initially created through a collaboration with nTara, Inc., a VaNTH partner. We later joined with nTara to create a version of this editor that interoperates with VaNTH’s delivery platform, eLMS (described below). A companion authoring tool enables instructors to create solutions to FBD problems. This tool was also modified to export the solution in an XML representation that CAPE could import into a condition set.

The design of the activity allows the learner three attempts at creating a correct FBD, where “correctness” is determined using feature comparison with the instructor-created solution. Feature comparison is also used to provide feedback to the learner following a failed attempt. The feedback is provided by the FBD editor itself on subsequent attempts but is retrieved from the eLMS delivery platform using Web services. The feedback is generated by an FBD diagnostician component created in Python by Prof. Robert Roselli at Vanderbilt University. The algorithm is implemented as a set of Python functions in a CAPE condition set. Like the vector tutor example above, the level of feedback provided by this diagnostician is progressive.

The CAPE model of the FBD activity employs various abstraction techniques, multiple condition sets, and *actions* to invoke the diagnostician and prepare follow-up editing sessions. Dynamic content techniques of assessments are used to present the feedback following a third failed attempt and elicit

student reflections on their difficulties. A CAPE “wizard” is employed to simplify the correct instantiation of this collection of techniques in creating new FBD-based activities. Based on a few questions, this wizard generates the FBD activity ready for delivery or for composition into larger FBD-based exercises. The wizard further supports instantiating alternative design patterns involving FBD problems.

These and many other CAPE-authored out-of-class learning activities are currently being delivered to students at VaNTH institutions. In the next section we describe the platform that supports the use of these activities.

The eLMS Delivery Platform

To support the delivery of adaptive learning activities authored with CAPE, we created the experimental learning management system (eLMS). This delivery platform is “experimental” in two senses. First, the platform is extensively instrumented, enabling its use as a vehicle for experimentation with Web-based learning experiences. These capabilities support the research mission of the VaNTH ERC. Second, since VaNTH is pioneering new concepts and capabilities in its authoring technology, solutions must be found for enacting these capabilities and eLMS provides a vehicle for experimenting with such solutions.

Particularly with the latter needs in mind, we chose to base the eLMS platform on an adaptable Web application framework called Zope. Zope [14] is an open source framework implemented in the Python language. In addition to extensibility in this language, Zope provides powerful dynamic content capabilities through its dynamic template markup language (DTML) and the newer Zope page templates (ZPT). Zope is backed by an object-oriented database called ZODB that en-

ables an object-oriented approach to constructing large Web applications.

The heart of the eLMS platform is a model-based delivery engine. This engine uses designs authored with CAPE as instructions for enacting learning experiences. A run-time representation of the design of a module, mosaic, or course is persisted in ZODB for each student and is “decorated” with artifacts produced by students to create the record of their use of the materials. The eLMS delivery engine is extensible and can be specialized with delivery semantics associated with instructional design patterns used for authoring in CAPE. This is accomplished using *delivery templates* that are the enactment counterparts of instructional design patterns used for authoring.

Web services for the eLMS platform are implemented in Python and can be accessed using HTTP requests or XML-RPC [15]. A novel Zope capability called *acquisition* allows Web services to be contextualized by objects identified by the called URL. This capability is used extensively by the eLMS delivery engine in invoking delivery-related Web services using the context of a particular student delivery. Web services also support the integration of embedded learning technologies, including execution coordination and data interchange. Through data interchange Web services, outcomes generated by embedded technologies are preserved in the student’s delivery record.

The courseware delivery interface (Figure 3) of the eLMS platform is another area where we are pioneering new capabilities. Rather than consuming browser “real estate” with navigation and other kinds of information and features for the learner, pull-in accessories are used to support extending the interface. These can be seen as tabs along the browser border. Using the approach of pull-in accessories we can provide a large amount of

Activity 2 - muscles and muscle groups

How does Muscle Force and Muscle Torque vary with position θ ?

Unit vector (insertion to origin): $u = \frac{(x_o - x_i)\mathbf{i} + (y_o - y_i)\mathbf{j}}{\sqrt{(x_o - x_i)^2 + (y_o - y_i)^2}} = u_x\mathbf{i} + u_y\mathbf{j}$

Muscle force vector: $F = |F|u, \quad r = x_i\mathbf{i} + y_i\mathbf{j}, \quad M_s = r \times F$

Muscle Torque: $M_s = |F| |x_i u_y - y_i u_x| k = |F| dk$

Fig. 3. eLMS delivery interface for learners.

interface functionality without overly intruding on the instruction. Accessories can be associated with a delivery template to provide interface extensions specific to an instructional design pattern—a specific review accessory, for example.

Student delivery records are an important resource that not only supports the evaluation of student performance but also informs the improvement of the learning activities themselves. eLMS provides good services for interactively accessing and reviewing such records. But especially as class sizes grow large, direct examination of student delivery records becomes burdensome as the exclusive means of gathering and synthesizing information. To address this concern we have recently created the first of what we expect will be a set of data mining tools that can be used to analyze delivery records. This tool is a pattern-based lexical scanner whose input is a set of delivery records rendered in XML and output is a spreadsheet summarizing pattern matches. We are currently investigating extensions to this tool that can automatically generate pattern lists based on the CAPE models themselves by leveraging regularities resulting from the use of instructional design patterns by authors.

While eLMS is primarily a vehicle for exploring delivery issues arising in the development of the CAPE authoring technology and for experimenting with the use of VaNTH courseware by learners, it must nonetheless provide more conventional capabilities found in production LMS platforms. For example, instructions can form classes, assign courseware, and review student delivery records. Authors can upload and update courseware. Infrastructural services provide authentication, session management, and access management. These capabilities are implemented by reusing or extending Zope platform services.

Conclusions

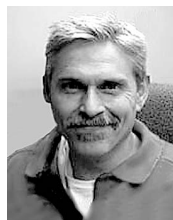
Web-based learning activities afford a unique opportunity to provide individualized learning experiences. In pursuing this opportunity, we have been motivated by the desire to put educators in the “driver’s seat,” rather than technologists. Educators are the ones with the intuition, born of their experience, to determine how adaptive learning activities can best help learners, for they are the adaptation mechanisms in today’s education system. Further, “ownership” of learning activities by educators greatly increases the possibility that the activities will be sustained and evolve over time.

The vehicle we have created for exploring this possibility is the CAPE authoring environment, and the journey we are making in this vehicle is one made with bioengineering educators and others from the VaNTH ERC. One of the unique facets of VaNTH that fosters direction and momentum for this journey is the practice of collaborative interventions in the design of new learning experiences. These interventions involve all thrusts of the ERC—learning science, assessment and evaluation, learning technology, and educators—as mutual stakeholders in the design process. As technologists, we bring to such interventions understandings of technological possibilities, resources, and constraints. Sometimes these are not the concerns that dominate a design, and other times they strongly influence it. What is clear from our experiences is that in the absence of influences of all stakeholders, designs tend to gravitate towards the concerns of the stakeholders most involved.

CAPE and its delivery platform eLMS are still quite young and continue to evolve rapidly. The snapshot presented in this article is intended to provide bioengineering educators a glimpse of the kinds of capabilities that are coming, and we extend to them the opportunity to join us for the journey that will shape the ultimate form and reach of these capabilities.

Acknowledgments

The author would like to gratefully thank Prof. Robert J. Roselli of the BME Department of Vanderbilt University, who, foremost among the pioneering users of CAPE and eLMS, has made invaluable contributions to the designs of these technologies. This work is sponsored by the Engineering Research Centers Program of the National Science Foundation under Award Number EEC-9876363.



Larry Howard is a senior research scientist at the Institute for Software Integrated Systems of Vanderbilt University’s School of Engineering. He leads the Adaptive Courseware Technology (ACT) Project in the Learning Technology Thrust of the VaNTH ERC.

Address for Correspondence: Larry Howard, Vanderbilt University, P.O. Box 1829, Station B, Nashville, TN 37235. E-mail: l.howard@vanderbilt.edu.

References

- [1] T. Murray, “Authoring intelligent tutoring systems: An analysis of the state of the art,” *Int. J. Artif. Intell. Educ.*, vol. 10, no. 1, pp. 98-129, 1999.
- [2] J. Hartley and D. Sleeman, “Towards more intelligent teaching systems,” *Int. J. Man-Mach. Stud.*, vol. 5, no. 2, pp. 215-236, 1973.
- [3] P. Brusilovsky, “Adaptive and intelligent technologies for web-based education,” in *Künstliche Intelligenz*, vol. 4, pp. 19-25, 1999.
- [4] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason IV, G. Nordstrom, J. Sprinkle, and P. Volgyesi, “The generic modeling environment,” in *Proc. Workshop Intelligent Signal Processing*, 2001, pp. 19-25.
- [5] A. Ledeczi, A. Bakay, M. Maroti, P. Volgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai, “Composing domain-specific design environments,” *Computer*, vol. 34, no. 11, pp. 44-51, Nov. 2001.
- [6] Advanced Distributed Learning (ADL) Initiative. *Shareable Courseware Object Reference Model*. Version 1.2. [Online]. Available: <http://www.adlnet.org>
- [7] Instructional Management Systems (IMS) Global Learning Consortium. *IMS Learning Resource Meta-data Specification*. Version 1.2.2. [Online]. Available: <http://www.imsproject.org/metadata>
- [8] Instructional Management Systems (IMS) Global Learning Consortium. *IMS Question and Test Interoperability Specification*. Version 1.2. [Online]. Available: <http://www.imsproject.org/question>
- [9] Instructional Management Systems (IMS) Global Learning Consortium. “IMS Learner Information Package Specification. Version 1.0. [Online]. Available: <http://www.imsproject.org/profiles>
- [10] D. Beazley and G. Van Rossum, *Python Essential Reference*, 2nd ed. Indianapolis, IN: New Riders, 2001.
- [11] J.D. Bransford, A.L. Brown, and R.R. Cocking, *How People Learn: Brain, Mind, Experience, and School*. Washington, DC: National Academy Press, 1999.
- [12] B.J. Barron, D.L. Schwartz, N.J. Vye, A. Moore, A. Petrosino, J.D. Bransford, and CTGV, “Doing with understanding: Lessons from research on problem and project-based learning,” *J. Learning Sci.*, vol. 7, nos. 3-4, pp. 271-312, 1998.
- [13] R.J. Roselli, B. Cinnamon, P. Norris, S.P. Brophy, D. Eggers, and J. Brock, “Development of an interactive free body diagram assistant for biomechanics,” in *Proc. 2nd Joint EMBS-BMES Conf.*, 2002, pp. 2617-2618.
- [14] A. Lattier and M. Pelletier, *The Zope Book*. Indianapolis, IN: New Riders, 2001.
- [15] D. Winer. *XML-RPC Specification*. [Online]. Available: <http://xmlrpc.com/spec>