Proceedings of Informing Science & IT Education Conference (InSITE) 2009

Regaining the 'Object' of Learning Objects

Namdar Mogharreban and Dave Guggenheim
Southern Illinois University, Carbondale, Illinois, USA

namdar@cs.siu.edu; dguggen@siu.edu

Abstract

Learning objects were to bring a seismic shift to the field of computer-based instruction by introducing transportability and reusability. Supposedly outfitted with the concepts taken from object-oriented (OO) design, learning objects have long promised dramatic savings of time and money in course and curricula development. However, they have failed to deliver the return on investment that seems a natural extension of their existence, in large part because the conceptual mechanisms adopted by OO design for transportability and reusability are lacking in learning objects.

Object-oriented software development, first discovered in the 1960s, had ushered in a new era of programmatic coding and design by the 1990s. Instead of thinking in terms of "verbs," or the processes that act upon information, developers could directly conceive of "nouns," or the objects that define the world around us, and provide these objects with real-world attributes. These transportable and reusable objects would then possess a library of ready-to-use actions that provide both a rich feature set as well as isolation for the user from implementation complexity. Software languages designed with support for such concepts as classes, methods, instantiation, overloading, overriding, inheritance, polymorphism, and encapsulation, achieved this tectonic shift in computer engineering and resulted in dramatic improvements in reliability, reusability, and cost.

In response to this shortcoming, we have proposed a new entity - the learning pod (Mogharreban & Guggenheim, 2008). The learning pod is our conception of the Learning Object. Engineered with the original concepts behind object-oriented development, the proposed conception uses OO technology to create an experientially seamless interconnection between disparate learning variables and delivers on the promise of sharing and reuse. The proposed learning object is construed as a class in OOP. A class may be considered as a blueprint, a schematic for replicating an object. Using a class begins with instantiating a new object based on that blueprint. Instantiation is the process by which a new copy of an object is created for use by invoking a constructor. This "instance" of a class referred to as an object has all the properties of the original, and is immediately available for application.

Introduction

In 1994, the term "learning objects" made its first appearance in the title of Wayne Hodgins CedMA working group, “Learning Architectures, APIs and Learning Objects” (Polsani, 2003). This reference is ostensibly made toward object-oriented programming, a paradigm of software engineering where software programs are built using modules that are interoperable, reusable, and easier to maintain than their monolithic counterparts. In a similar fashion, an academic course can be broken up into computer-mediated instructional units that possess these same qualities – portability, adaptability, reusability, and ease of maintenance.

Object-oriented software programming, first conceived by Kristen Nygaard and Ole-Johan Dahl in the 1960s (Campbell-Kelly, 2002), achieves these qualities by introducing and manipulating the concepts of classes, methods, instantiation, overloading, overriding, inheritance, polymorphism, and encapsulation. (Korson & McGregor, 1990).

Following the Object Orientation paradigm we have proposed a new entity - the learning pod (Mogharreban & Guggenhiem, 2008). Designed as an object-oriented structure with the capabilities offered by instantiation, encapsulation, inheritance, and polymorphism, the learning pod consists of learning variables and several software modules – Theme Builder, Styler, Learner, and Evaluator - that drive their learner-dependent selection, positioning, and reuse. These engines, when combined with feedback mechanisms, create a structure that is highly reusable, from the entire course down to a single digital element of instruction.

A learning pod contains learning variables (LVs) which are context-free digital elements, whether text, audio, video, animation, etc., that become part of a learning pod only when applied within a context of learning. The context of learning is defined by the “Theme Builder” of the learning pod. A pod is a collection of learning variables connected with sound pedagogical principles by way of software engines driving metadata, sequencing, personalization, and reusability in the form of encoded functional and aesthetic information. Most importantly, the learning pod utilizes available technology, albeit state-of-the-art, to accomplish its tasks.

The proposed learning pod is construed as a class in OOP. A class may be considered as a blueprint, a schematic for replicating an object. Using a class begins with instantiating a new object based on that blueprint. Instantiation is the process by which a new copy of an object is created for use by invoking a constructor. This “instance” of a class referred to as an object has all the properties of the original, and is immediately available for application. For example, an abstract data type (class) called Employee may contain the information and actions associated with a person engaged for hire. Name, contact information, hire date, pay period, base pay rate, vacation allowance, and other relevant data is contained within the class Employee, and when a new object is instantiated (a new employee is hired), this information would be gathered and input as part of the constructor process.

Within classes, methods are built-in actions that may be taken on an object, and constructors are the first method called when instantiating a new object. What makes a constructor different from an ordinary subroutine is the fact that they are frequently overloaded. Constructor overloading provides an extra degree of flexibility by allowing objects to be instantiated with varying degrees of input details. For example, the class Employee may have several constructors:

1. Name and address only
2. Name, address, and hiring information
3. All of the above, plus scanned résumé
4. All of the above, plus physical test results

Each of these constructors will create a new object titled “Employee,” and each is to be used depending on the information available at the time of processing. Other methods contained within the class Employee may include affecting a departmental transfer, marking a change in supervisor, entering a probationary review, placing on suspension, noting vacation status, etc.

In the field of learning objects, this would equate to the simple process of duplicating a learning object and entering the learner-centric information necessary to activate its use. At the lowest level of object orientation, a LO-based course is an instance of a learning pod which could be copied and used exactly as it was originally intentioned.

Extending instantiation is the prospect of inheritance. Instantiating an object from a class, and then adding user-defined data fields and/or methods over and above those already built-in is the essence of inheritance and a basic component of reusability. An example would be a company that wishes to use the object class Employee, but has different policies and procedures than those inherent in the base class. It could create fields and methods as an addition to the class Employee without having to rewrite the class or alter the basic blueprint. A key facet of inheritance is method overriding, or the ability to replace certain built-in actions with custom methods without disrupting any details not specifically addressed. Under the umbrella of inheritance, learning objects could be instantiated and then adapted - learning elements, sequencing, and/or testing – for a particular purpose or requirement.

Polymorphism (Greek for “many forms”) may be expressed in a number of ways, but the most relevant to this discussion is the use of inheritance and method overriding to achieve new uses for an object. A popular example is a class of geometric objects with a method to calculate surface area. Regardless of whether the object is a rectangle, square, circle, or triangle, the area method will correctly calculate the answer because of its polymorphic capabilities that allow it to “recognize” different shapes. A corollary with learning objects would be to copy a LP-based course and then shift its focus to an ancillary topic by introducing new learning elements with their concomitant sequencing and testing schemas. For example, a history lesson could be shifted from a character perspective to one of commerce or geography, retaining as much material as possible in the conversion to save development time and cost. Of course, the learning variables that comprise a unit of instruction, whether textual, visual, or audible, are inherently polymorphic and must be kept context-free through careful meta-tagging in order to promote the widest possible uses.

Encapsulation, also known as information hiding, is an OO technique that hides underlying complexity or private information, or both, from the user. A favorite real-world example is the vending machine, in which a user knows the operational parameters (put money in, press selection, take item), but does not need to understand the mechanical engineering that accommodated their purchase. Similarly, in a paradigm of learning objects, the instructional designer need not be concerned with the coding details of learning elements placement, learning style interpretation, page sequencing or performance testing. Encapsulation would occlude this complexity from view, and allow the designer to focus on the course instead of its construction and execution details.

A Return to Object Orientation

The single most-cited advantage of OO design principles is the ability to build transportable, reusable, and adaptable software components (Pancake, 1995). Because a single hour of online instruction can take up to 300 hours to develop (Kapp, 2003), reusability is also the core return on investment (ROI) message offered by learning object promoters, from the earliest days to the present (Churchill, 2005, 2007; Downes, 2003; du Plessis, 2005; García-Barriocanal et al., 2007; Hodgins, 2000; Liber, 2005; Liu et al., 2005; Polsani, 2003; Wiley 2000). Yet, after 12 years of successive evolution, learning objects are still primarily a collection of stand-alone modules that rarely interconnect outside of strictly controlled regimes, such as those imposed by corporate and military training guidelines.

The engineered reusability of a learning pod begins with importing or creating learning elements in the form of digital, context-free audio, video, text, or interactive components that may be affixed to a browser-based presentation layer or web page. By remaining context-free at this level, the learning variable is inherently reusable. These learning variables do have metadata, but only that which defines the structural specifications of the object instead of imposing, accidentally or otherwise, restrictions on its potential use.

In addition to the metadata, each learning variable has associated keywords, which include language of origin, copyright owner, type of object (text, audio clip, etc.), statement of purpose, and potential applications. The statement of purpose describes the content, such as a speech classified as “speech.” But potential applications for that speech may include speech communications, history, management style, biography, etc. Encompassing the metatags that allow selection, and indexed keywords that promote the broadest possible use, a software engine could search, select, and place learning objects on a presentation layer using either manual or automated processes, or both.

The Theme Module (Class, Instantiation, and Inheritance)

The Theme is a collection of search criteria, metatags, design data, and keywords about the contents and structure of the unit of instruction contained within the learning pod that allow the pod to be found and instantiated. (See Figure 1.) Conforming to the principles of object-oriented design, the Theme presents a layer of abstraction (encapsulation) that provides external contact as the aim of a search or as the export agent for the contents and design of the learning pod. Primarily, Theme contains the overarching topic that describes the mission and content of the learning pod, appropriate grade level or background, and packaged page layout, individual learning object title, type and placement, sequencing, and aesthetic information provided by the Styler. Populated with this information, the Theme has enough data to export a blueprint of the baseline learning pod (prior to Learner adaptation), allowing a receiver to reconstruct the unit of coursework as a clone of the original – the essence of instantiation. Once instantiated, a learning object can be extended through inheritance, overloading, and overriding.

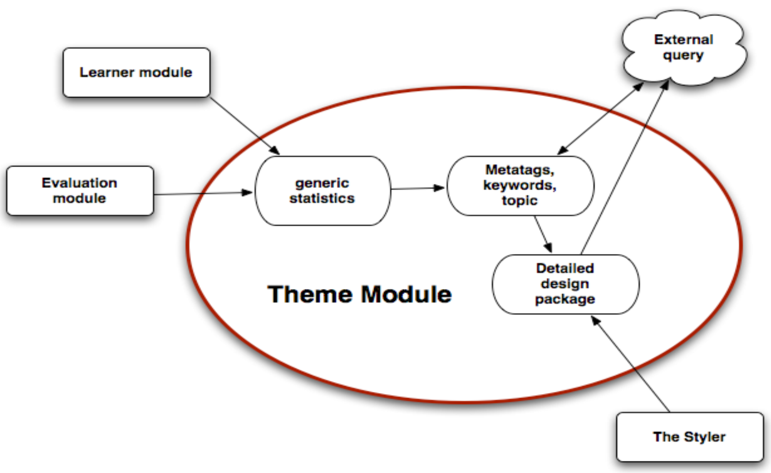


Figure 1 – The Theme module

Linking with the Learner module, Theme has to potential to contain additional generic information extracted from learning episodes (personal data removed), such as mean score by grade level, 5-number summary of scores (minimum, 1st quartile, median, 3rd quartile, maximum), pass/fail ratios – even student satisfaction ratings if desired. This will provide a qualitative measurement of the learning pod to the outside world for inspection prior to its reuse.

The Styler (Overriding, Polymorphism, and Encapsulation)

The Styler is an internally-focused multi-use software engine that performs the following tasks (See Figure 2.):

1. Determination of learning style
2. Learning object selection based on learner cognitive style
3. Presentation-layer authoring tool
4. Sequencing of instructional and assessment pages
5. Packaging and export of design information separate from content

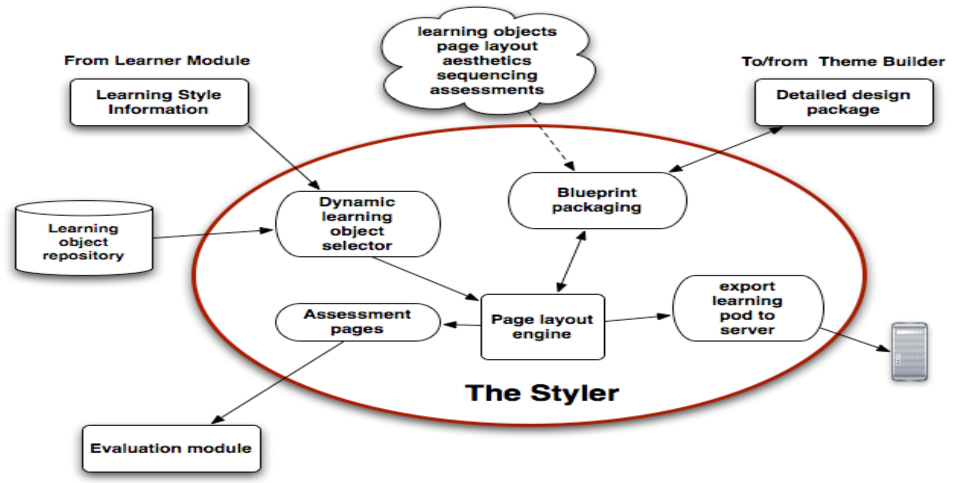


Figure 2 – The Styler module

Finding and selecting appropriate learning variables for a unit of study is potentially an automated process using fuzzy-logic determination combined with keyword analysis. The instructional designer would input the desired topic and characteristics into a structured query-driven search engine designed to retrieve digital learning elements based on fuzzy-logic metatags, and the result would be the population of a local library with learning variables meeting the criteria.

Another function of the Styler is to generate pre-assessment, instructional, and assessment pages using drag-and-drop technology via standard page layout software interface. Realization of a learning pod authoring system, though, mandates that aesthetic design data both sensory and immersive, be encapsulated apart from the content using purpose-built style sheets. This isolation of form and function calls for a new specification linked to XSL visual design data. In addition to metadata, a lesser degree of abstraction is necessary to catalog the aesthetics of a presentation-layer web page. Design principles such as layout, font, color schemes, frame placement, button style, white space, movement, and object relationships on a 'page' are essential aspects of sound instructional design, yet these components are not codified among the technical standards available for specific reuse. It has been suggested that each learning object developer create multiple styles using XSL/XML style sheets, in the hopes that one of those will aesthetically match a module developed by another course designer (Polsani, 2003). Rather than invite probabilistic determination in deciding whether learning objects are reusable, we propose a means of achieving compatibility between disparate learning pages and pods that relies on the electronic exchange of stylistic attributes with both manual and automatic reactions. The location of learning varaibles on a page will be automatically generated and kept in a data file as standard page layout information (object metatags & title and x-y coordinates along with x-y pixel size information). This geometric mapping, the sequence of pages, and other aesthetic information will be codified and transmitted to the Theme module as the blueprint of the learning pod. In this way, the Styler and Theme modules work together to achieve encapsulation and inheritance of the learning pod.

In addition to designing and building single or multiple pages of pre-assessment, instruction or assessment, the Styler sequences the pages using a page-sorter view according to the pedagogical principles applied by each individual instructor/designer. Sequencing may be accomplished on a page-by-page basis or by using an instructional algorithm/template such as those proposed by Component Display Theory (Merrill, 1994), Elaboration Theory (Reigeluth, Merrill, & Wilson, 1978; Reigeluth, Merrill, Wilson, & Spiller, 1979), Instructional Transaction Theory (Merrill, 1999), or the collection offered by Kaur et al. (2007). Using web pages as the basis for the presentation layer not only offers the greatest flexibility in learning object selection (text, audio, animation, full-motion video, interactive applets, etc.), it makes the job of sequencing using a page-sorter view rather simple. Standards such as IMS Global's Common Cartridge Format (IMS Global, 2007) and ADL's SCORM (ADL, 2006) will find applicability only when adapting learning pods to third-party Learning Management Systems (LMS) because learning pods and their contents are inherently interchangeable, adaptable, and reusable using only the generic standards for XHTML within HTML 4 as specified by the W3C (1999).

Beyond the baseline layout and sequence, adaptive sequencing of instructional pages, driven by an understanding of the learner's prior knowledge and skill set, is integral to the learning pod. Work done by García-Valdez et al. (2007), Brusilovsky & Vassileva (2003), Brusilovsky & Peylo (2003), and Weber & Brusilovsky (2001) establish the means by which artificial intelligence techniques may be used to define a learner-centric path through coursework.

Regardless of whether the pages are assembled ad-hoc, divined through an algorithm, or prepared using a template, a page-sorter view with color-coded page margins indicating basic instruction sequence, intermediate assessments, and progression thresholds will be available to the designer. Furthering the object-oriented design principle of polymorphism, imported pages or sequences of pages may be used as-is, redesigned with altered or new learning variables, and/or rearranged according to the teacher's pedagogical principles. In all cases, the imported pages will adopt the aesthetic parameters of the destination instructional program. When sequencing selection is complete, the system will generate a data file of the pages, objects, and all other visual, auditory and kinesthetic information.

Finally, the Styler has the ability to catalog what type of learning varaible occupies the page or portion thereof and to make dynamic substitutions or additions based on the learner's cognitive style. Several methods for capturing learning styles and using that information to drive learning object selection have been proposed (García, et al., 2007; García-Valdez et al., 2007; Kaur et al., 2005; Mustaro & Silviera, 2006; Santally & Senteni, 2005; Wolf, 2002). Whether the learning style is discerned from analyzing responses to a questionnaire, like the 118-question survey proposed by Wolf (2002) in the iWeaver project, or if it is detected in real-time during the course of instruction by providing alternative content models to be rated by the learner and corroborated by learner responses, the relevant information from which the Styler will base its placement decisions comes from the Learner module.

The Learner Module (User-centric Polymorphism)

Information about the learner, such as grade level, prior knowledge, course histories, and learning style(s) is kept in a secure, password-protected and encrypted data file capable of being contained on a smart card. Interacting on a two-way channel with the Styler and

Evaluation modules, the Learner Module contains derived learning style data and pre-assessment test scores (so as to establish a prior knowledgebase before engaging in the main instructional unit), with the caveat that the learning style may dynamically drive learning object selection, and prior knowledge may then drive sequencing and progression. (See Figure 3.) When the student has completed a section of instruction, assessment results are returned to the Learner module from Evaluation and stored in a secure environment. When a course has been finished, a blueprint of their coursework, including learning objects, layout, sequencing, assessments and scores, is saved to the Learner data file. Using this information, it would be possible to reconstruct a static reproduction of the exact instruction he or she received, thereby offering reuse at an individual level.

With regard to learning styles, each time the learner interacts with a learning pod, available data is captured and kept in this secure environment. By maintaining a long-term history, the Learner has the ability to catalog learning styles based on topic, content, time of day, or any other available parameter with an associated value.

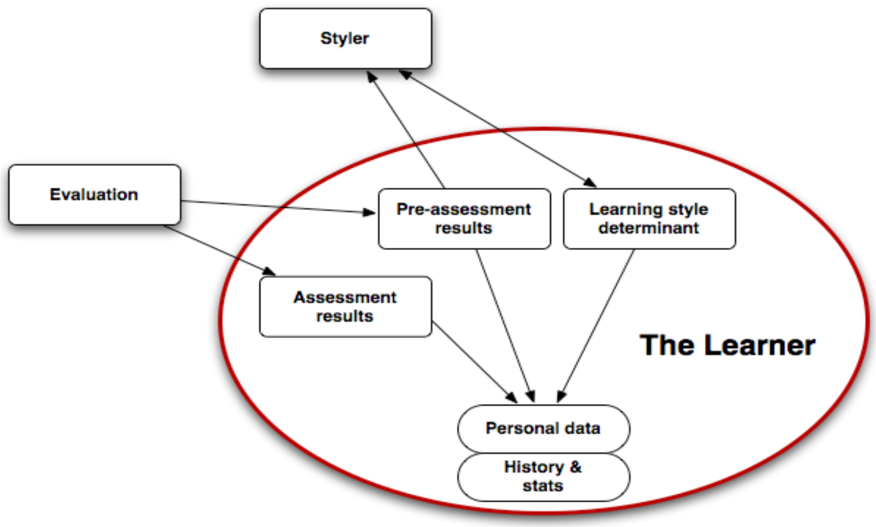


Figure 3 – The Learner Module

Evaluation (Overloading, Overriding, Polymorphism)

Every course of instruction must include some means of assessing the learner's performance at three general waypoints: 1) prior to engaging with the coursework to establish prior knowledge, 2) performance hurdles offered during the course of instruction, and 3) post-instruction qualification. Assessments and evaluations are conducted by the Evaluation module, a discrete program within the learning pod that is tightly integrated with the Styler and Learner modules. (See Figure 4.) When sequencing pages of instruction with the Styler during the design and build phase, the designer will denote those pages reserved for assessment purposes. Templates defining assessment types (multiple choice, fill-in-the-blank, matching, essay response) and any relevant learning objects are selected, placed on a page, and populated with questions, answers, and when revealed with an incorrect answer, page locations where the correct answer is discussed.

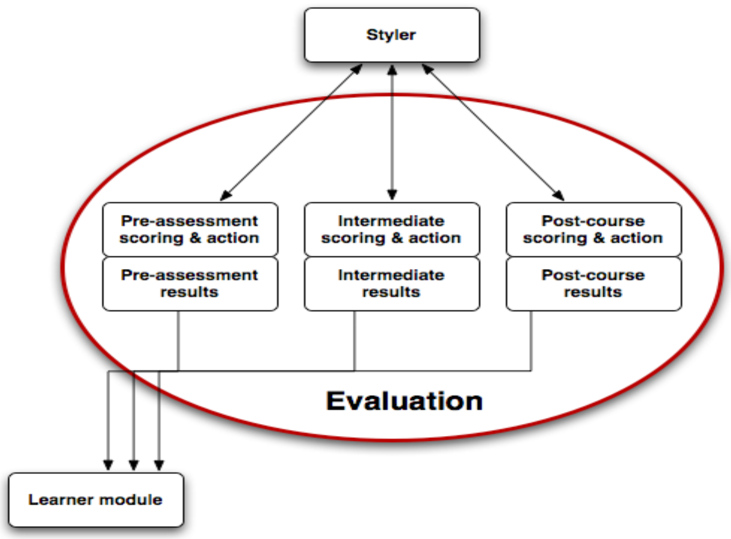


Figure 4 – Evaluation module

Logical rules-based expressions are programmed into the Evaluation module that instructs the Styler as to sequencing actions based on assessment results. For example:

Rule 1: If Assessment A > 80%, and Assessment B >= 50%, then unlock Page X (which in turn unlocks a new sectional study path).

Rule 2: If Assessment A < 80% and Assessment B is >= 50%, then return to Page Y, but skip over the section beginning with Page Z.

Results of assessments are provided to the Learner data file for long-term secure storage and future student-learning pod interaction as prerequisite information.

Another use for assessment data is to generate scoring statistics that reflect the intended use and quality of the learning pod. Transferring generic student grade level and scoring information from the Evaluation module to the Theme Builder for statistical processing and presentation to a query response adds a valuable indicator for future reuse in various situations and groups.

Conclusions

True hierarchal reusability - from a course to a section and from a page to a single learning variable – has been designed into the learning pod. Constructed in a paradigm of object-oriented programming and using off-the-shelf technology, the learning pod applies those concepts to the field of learning objects by promoting maximum reuse without placing limitations on aesthetic choice and academic freedom. A learning object is an instance of a learning pod which is "customized" for the learner by way of the methods and the data about the content and the user pass on to it during instantiation.

References

ADL. (2006). *SCORM 2004* (3rd ed.). Retrieved August 30, 2007, from http://www.adlnet.gov/scorm/index.aspx

Bloom, B. S. (1956). *Taxonomy of educational objectives, Handbook I: The cognitive domain*. New York: David McKay Co.

Brusilovsky, P. & Peylo, C. (2003). Adaptive and intelligent web-based educational systems. *International Journal of Artificial Intelligence in Education, 13*, 156-169. Retrieved September 11, 2007 from http://aied.inf.ed.ac.uk/

Brusilovsky, P., & Vassileva, J. (2003). Course sequencing techniques for large-scale web-based education. *International Journal of Continuing Engineering Education and Lifelong Learning*, *13*(1/2).

Campbell-Kelly, M. (2002, August 23). Obituary – Kristen Nygaard. *The Independent*. London. Retrieved January 20, 2008 from http://findarticles.com/p/articles/mi_qn4158/is_20020823/ai_n12639248

Churchill, D. (2005). Learning object: An interactive representation and a mediating tool in a learning activity. *Educational Media International, 42*(4), 333–349.

Churchill, D. (2007). Towards a useful classification of learning objects. *Education Tech Research Development, 55*, 479–497.

Cisco Systems. (2001). *Reusable learning object strategy: Designing information and learning objects through concept, fact, procedure, process, and principle template*. San Jose, CA: Cisco Systems, Inc.

Cochrane, T. (2005). Interactive quicktime: Developing and evaluating multimedia learning objects to enhance both face-to-face and distance e-learning environments. *Interdisciplinary Journal of Knowledge and Learning Objects, 1*(1), 33–54. Retrieved from http://ijello.org/Volume1/v1p033-054Cochrane.pdf

Downes, S. (2003). Design and reusability of learning objects in an academic context: A new economy of education? *USDLA Journal: A Refereed Journal of the United States Distance Learning Association, 17*(1). Retrieved September 11, 2007 from http://www.usdla.org/html/journal/JAN03_Issue/article01.html

du Plessis, J. (2005). Learning objects: Using language structures to understand the transition from affordance systems to intelligent systems. *Interdisciplinary Journal of Knowledge and Learning Objects*, *1*(1), 55–67. Retrieved from http://ijello.org/Volume1/v1p055-066duPlessis.pdf

García-Barriocanal, E., Sicilia, M., & Lytras, M. (2007). Evaluating pedagogical classification frameworks for learning objects: A case study. *Computers in Human Behavior, 23*, 2641–2655.

García, P., Amandi, A., Schiaffino, S., & Campo, M. (2007). Evaluating Bayesian networks' precision for detecting students' learning styles. *Computers & Education, 49*, 794–808.

García-Valdez, M., Licea, G., Castillo, O., & Alanis, A. (2007). Simple sequencing and selection of learning objects using fuzzy inference. *Proceedings of the North American Fuzzy Information Processing Society*, NAFIPS '07 Annual Meeting, 628-632.

Griffiths, J., Stubbs, G., & Watkins, M. (2007). From course notes to granules: A guide to deriving learning object components. *Computers in Human Behavior, 23*, 2696–2720.

Hodgins, H. W. (2000). The future of learning objects. In D. A. Wiley (Ed.), *The instructional use of learning objects: Online version*. Retrieved August 21, 2007, from http://reusability.org/read/chapters/hodgins.doc

Hodgins, W., & Conner, M. (2000). Everything you ever wanted to know about learning standards but were afraid to ask. *Learning in the New Economy e-Magazine (LiNE Zine), Fall*. Retrieved September 11, 2007, from http://www.linezine.com/2.1/features/wheyewtkls.htm

IEEE. (2001). WG12: *Learning object metadata*. Retrieved September 4, 2007, from http://ltsc.ieee.org/wg12/.

IMS Global Learning Consortium, Inc. (2003). *IMS learning design information model*. Retrieved October 25, 2007, from the World WideWeb: http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html

IMS Global Learning Consortium, Inc. (2003). *IMS common cartridge forma*t. IMS Global Learning Consortium, Inc. Simple Sequencing.

Landauer, T. K., & Ainslie, K. I. (1975). Exams and use as preservatives of course acquired knowledge. *Journal of Educational Research, 69*, 99-104.

Liber, O. (2005). Learning objects: Conditions for viability. *Journal of Computer Assisted Learning*, *21*, 366-373.

Liu, J., Huang, B., & Chao, M. (2005). The design of learning object authoring tool based on SCORM. *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies*. Retrieved August 30, 2007 from http://ieeexplore.ieee.org/iel5/10084/32317/01508815.pdf?arnumber=1508815.

Kapp, K. M. (2003). How long does it take? Estimation methods for developing e-learning. *ASTD Learning Circuits*. Retrieved November 6, 2007 from http://www.learningcircuits.org/2003/jul2003/kapp.htm

Kaur, A., Dunning, J., Bhattacharya, S., & Ahmed, A. (2005). Re-purposeable learning objects based on teaching and learning styles. *Encyclopedia of Multimedia and Technology*. London, Melbourne, New York: Idea Group Publishing. Retrieved October 20, 2007 from www.indiana.edu/~geosci/people/faculty/dunning/pdf/encyc.pdf

Korson, T., & McGregor, J. D. (1990). Understanding object-oriented: A unifying paradigm. *Communications of the ACM, 33*(9), 40-60.

McGreal, R. (2004). Learning objects: A practical definition. *International Journal of Instructional Technology and Distance Learning*, 1(9), 21–32.

Merrill, M. D. (1999). Instructional transaction theory (ITT): Instructional design based on knowledge objects. In C. M. Reigeluth (Ed.), *Instructional-design theories and models: A new paradigm of instructional theory* (pp. 397- 424). Mahwah: Lawrence Erlbaum Associates.

Merrill, M. D. (2000). Knowledge objects and mental models. In D. A. Wiley (Ed.), *The instructional use of learning objects*. Retrieved August 21, 2007 from http://reusability.org/read/chapters/merrill.doc

Merrill, M. D., & ID2 Research Team. (1993). Instructional transaction theory: Knowledge Relationships among processes, entities, and activities. *Educational Technology. XXXIII*(4), 5.

Mogharreban, N., & Guggenheim, D. (2008). A new paradigm for reusability of learning objects. *Interdisciplinary Journal of E-Learning and Learning Objects, 4*, 303-315. Retrieved from http://ijello.org/Volume4/IJELLOv4p303-315Mogh477.pdf

Mustaro, P. N., & Silviera, I. F. (2006). Learning objects: Adaptive retrieval through learning styles. *Interdisciplinary Journal of Knowledge and Learning Objects, 2*, 35-46. Retrieved from http://ijello.org/Volume2/v2p035-046Mustaro.pdf

Pancake, C. M. (1995). The promise and the cost of object technology – A five year forecast. *Communications of the ACM, 38*(10).

Polsani, P. R. (2003). Use and abuse of reusable learning objects. *Journal of Digital Information, 3*(4), Article No. 164. Retrieved August 21, 2007, from http://jodi.tamu.edu/Articles/v03/i04/Polsani

Santally, M. I., & Senteni, A. (2005). A learning object approach to personalized web-based instruction. *European Journal of Distance Learning* (EuroDL). Retrieved October 20, 2007 from http://www.eurodl.org/materials/contrib/2005/Santally.htm

W3C. (2002) XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). A Reformulation of HTML 4 in XML 1.0. W3C Recommendation 26 January 2000, revised 1 August 2002. Retrieved October 20, 2007 from the World Wide Web at: http://www.w3.org/TR/xhtml1/

Weber, G., & Brusilovsky, P. (2001). ELM-ART: An adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education*, *12*, 351-384.

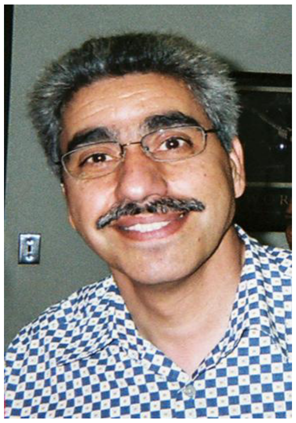
Wiley, D. A. (1999). *The Post-LEGO learning object*. Retrieved September 6, 2007, from http://wiley.byu.edu/post-lego/post-lego.pdf

Wiley, D. A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D. A. Wiley (Ed.), *The instructional use of learning objects*. Retrieved August 21, 2007, from http://reusability.org/read/chapters/wiley.doc.

Wiley, D., & Edwards, E. (2002). Online self-organizing social systems: The decentralized future of online learning. Retrieved October 25, 2007 from http://wiley.ed.usu.edu/docs/ososs.pdf.

Wolf, C. (2002). iWeaver: Towards an interactive web-based adaptive learning environment to address individual learning styles. *European Journal of Distance Learning* (EuroDL). Retrieved October 10, 2007 from http://www.eurodl.org/materials/contrib/2002/2HTML/iWeaver

Biographies



Dr. **Namdar Mogharreban** received his PhD in Computer Based Education in 1989. Since then he has been involved in various aspects of teaching and training in information systems and technology. Currently he is an associate professor at Southern Illinois University in the Department of Computer Science. The focus of his research has been end user computing, computing in special population, intelligent tutoring systems and data analysis. He teaches courses in Information Systems Design, Human and Computer Interface as well as programming languages.

David Guggenheim. Prior to returning to a university setting and engaging in learning objects research, Dave Guggenheim was responsible for overall product vision, strategic product planning, market-driven product development, and the resulting launch and sale of a broad range of software products directed toward the telecommunications industry. Most recently, as vice president of global marketing and product management, he oversaw worldwide marketing and product management functions for Ushacomm, an Indian provider of operations support systems, with headquarters in London and Kolkata. Dave has over 20 years of industry experience, including achievements in product development, product marketing, market entry, and global software product launches, which have resulted in awards for technical innovation and overall product excellence.