INTERACT
INTEGRATE
IMPACT

Proceedings of the 20th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE)

Adelaide, Australia
7–10 December 2003

Editors
Geoffrey Crisp, Di Thiele, Ingrid Scholten, Sandra Barker, Judi Baron

Citations of works should have the following format:

Author, A. & Writer B. (2003). Paper title: What it's called. In G.Crisp, D.Thiele, I.Scholten, S.Barker and J.Baron (Eds), *Interact, Integrate, Impact: Proceedings of the 20th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education*. Adelaide, 7-10 December 2003.

ISBN CDROM 0-9751702-1-X WEB 0-9751702-2-8

ascilite
Australasian Society For Computers
In Learning In Tertiary Education

Published by ASCILITE www.ascilite.org.au

REUSE IN PRACTICE: LEARNING OBJECTS AND SOFTWARE DEVELOPMENT

Maeve Paris
School of Computing and Intelligent Systems
University of Ulster, NORTHERN IRELAND
m.paris@ulster.ac.uk

Abstract

Learning objects (LOs) may be considered from two perspectives: the learning perspective, with a focus on learning objectives, content, and assessment in order to derive small instructional components from existing resources; and the object perspective, stemming from the object-oriented paradigm in computer science, with a focus on the development of small, reusable components, which are characterized in terms of accessibility, reusability, and interoperability. This dual perspective reflects the interests of the protagonists in the LO movement: the education community and the learning technology community. While the technologists are concerned with the development of technical systems designed to meet educational needs, these systems must also conform to pedagogical theories and concepts of instructional design, which are the domain of the education community. Many commentators disassociate learning objects from the object-oriented paradigm; at the same time, the LO community is preoccupied with the issue of reuse, which is a fundamental of this paradigm. This review draws parallels between both communities, as the same concerns are mirrored; the fundamentals of object technology should be applied to the entire learning object development process if higher levels of reuse are to be achieved.

Keywords

Reuse, reusability, learning objects, object-oriented development

Introduction

The learning object community is drawn from the worlds of business, higher education, and government, and includes authors, publishers, resellers, and end users such as individuals and academic departments. Cross-sector participation is a key theme of current initiatives to develop a learning object economy; indeed, learning objects (LOs) have emerged from two sectors of professional practice: object-oriented programming, and learning objectives. Higher education initiatives have championed the development of LO repositories (for example, the Co-operative Learning Exchange at the University of Waterloo, and the Use of ICTs in Flexible Delivery at the University of Wollongong), but the approach is not yet pervasive. Major obstacles to the more widespread adoption of learning objects by authors and users include the lack of a common language (Johnson, 2003, p.13).

Motivated by this gap in understanding, this review explores reuse in order to clarify its meaning for authors and end users in higher education, by drawing parallels across sectors. Until a single definition is agreed, it will be difficult to measure the extent of reuse, since the concept means different things to users, learners, developers and e-learning professionals. It will also be difficult to generalize findings from surveys and questionnaires. This review attempts to ascertain what reuse of learning objects means in practice, and to compare this with the practice of reuse in the software industry, where the concept of reusability has a stricter definition in relation to software development. If LO reuse is to become more widespread, there must be agreement to its definition. Reuse has long been a concern for software

developers, and lessons from developers could inform the LO community: realistic levels of reuse can only be achieved if development occurs in the context of an overall object-oriented approach to development.

Learning Objects and Software Objects

Many commentators are at pains to disassociate learning objects from the object-oriented paradigm; at the same time, the learning objects community is preoccupied with the issues of reuse and granularity, which are fundamentals of this paradigm. Perhaps the learning objects community denies this association at its cost, as the same concerns are mirrored in the software development community. Some studies (such as Sosteric & Hesemeier, 2002) have striven to deny links between learning objects and software objects, but they have only considered the object-oriented approach from an implementation perspective (using programming constructs and referring to lines of code). Object technology is much more than that: it is a development philosophy, with its own associated visual language, the Unified Modeling Language (UML). Other studies (such as Boyle, 2003) have argued the relevance of object technology by applying general design principles (such as coupling and cohesion) to the development of learning objects, but such principles are not necessarily object-oriented principles; indeed, they predate the object-oriented approach as they arose out of structured design, and while cohesion supports modularity, coupling tends to be somewhat in conflict with object principles of inheritance and polymorphism.

Some commentators (such as Wiley, 2000) have highlighted particular aspects of learning objects which stem from the object-oriented approach; for example, learning objects are defined in terms of reuse, or in terms of interoperability, but these are applied in terms of attributes of objects, rather than objects themselves. There is a preoccupation with design for reuse, but this is a characteristic of good design: it does not tell us anything about the design itself, and the design process. For some commentators, learning objects are components (Douglas, 2001): again, this fails to tell us anything about the fundamentals, since components are largely an implementation construct (as executable units they do not occur in the analysis phase). Similarly, the concept of interoperability also relates to the implementation phase. It may be opportune to consider object-orientation as part of the overall approach to the creation of learning objects, and not just in relation to implementation constructs, by using UML to model objects and all artefacts of the development process, accompanied by a suitable methodology to guide the process.

Reuse of Learning Objects in Practice

Concrete initiatives based on learning objects have emerged in recent years, particularly in the commercial domain. Hewlett Packard, for example, designed a training and support application for call centre agents, of which reusable learning objects were a significant component, and the system is evolving into sets of meta-tagged objects which the company hopes can be reused across product families. DocworksCPTI offers a service for migration of content to a reusable learning architecture, based on assets (such as text, graphics or animation), where a reusable learning object is defined as a collection of such assets which are grouped together to teach a task based on a single learning objective. Cisco Systems have pioneered object-based e-learning courses to support employees and customers. These commercial initiatives contrast with free exchanges of learning objects which have been a characteristic of academic projects such as MERLOT and the EOE, but which have been difficult to sustain without substantial subsidies.

Despite the emergence of these initiatives and much discussion of the topic in e-learning circles, the LO approach does not appear to be influencing practice as much as was anticipated. Dodani (2002) observed that LO technology was far from reaching its potential due to ‘the complexity of the problem it is trying to solve, the need for more advanced architecture and design of learning objects to address this complexity, and the tools, infrastructure and skills that are needed to address the problem effectively’ (Dodani, 2002, p. 38). Clark and Rossett (2002) described a ‘picture of halting acceptance’, attributing the slow uptake to ‘elementary lessons about sharing’. They also observed that the learning community was

engaged in the same debates as the software community when the object-oriented paradigm first emerged, ‘questions about standards, customization and quality and discomfort with giving up the control and creativity involved in building from scratch’ (Clark and Rossett, 2002).

Wagner (2002) identified barriers to adoption including the need for methodological rigour, while Lin (2001) identified the problem as the absence of a design and development methodology: ‘course development now needs to follow a systems development methodology and needs to [be] governed by something like a systems development life cycle’ (Lin, 2001). Douglas (2001) steered the debate closer to the object-oriented paradigm: ‘there is scope for adapting analysis and design methods from object-oriented software design ... making use of a modified version of the unified modeling language (UML) for the analysis and design of courses utilizing reusable learning objects’ (Douglas, 2001). He also observed that ‘there are relatively few studies and tools relating to the systematic analysis, design and documentation that should precede construction and delivery, and none that incorporate the emerging object model’ (Douglas 2001). Similarly, Polsani (2003) identified ‘a need to reengineer the design and development process of LOs’ (Polsani, 2003).

Reuse in software development

For the software development community, reuse is one of the pillars of the object-oriented paradigm. The success and spread of object-oriented programming has led to the techniques of object-oriented analysis and design (OOAD), where OO methods are used to organize information and related processing of that information according to the real world objects described by that information. Brown (2002) observed that the label object-oriented applies to anything presenting with the features of inheritance and polymorphism, as this allows reuse, and not just of program code (Brown, 2002, p. 80). The ability to reuse extends to analysis results too: ‘both code and analysis results are being reused on object-oriented projects’; however, ‘because of the learning curve for object-oriented thinking (...) the reuse benefits don’t become significant until about the third or fourth object-oriented project’ (Brown, 2002, p. 81). Promotion of reuse is one of the greatest claims for the object-oriented approach, yet it is difficult to assess how widespread reuse is; there is little empirical research on reuse; the dearth of research activity is illuminated by ‘a few glimmers of evaluative research light in an otherwise dark universe’ (Glass, 1999). Reuse remains an ideal for developers: ‘software reuse is not a matter of routine practice, the promises of software reuse remain for the most part unfulfilled’ (Mili et al, 1999).

Reusability offers many potential benefits to the software community: these include lower development costs, faster development times with a consequence of faster times to market, higher quality products, and lower maintenance costs, yet software reuse is not as prevalent as might be expected even though sixty to seventy percent of a system’s functionality is common to more than one system. Lack of uptake may be due a combination of factors: project isolation, lack of management commitment, unrealistic expectations, and the lack of a standard development process (Rosheim, 1999). Some of these factors are institutional, but many of Rosheim’s observations are of interest to learning technologists: he cited a timescale of three to five years in order to develop and realise benefits from reuse initiatives, and he warned that payback may not be realised until two to three years after a project is completed. Some of the solutions Rosheim proposed are equally applicable to development of advanced learning technologies: development teams should ‘speak the same language’ (Rosheim, 1999); in other words, standardisation of artefacts and activities is a pre-requisite for encouraging reuse This concern with standardisation has extended to the learning objects community, where metadata has long been a preoccupation, but this alone will not lead to the holy grail of reusability since it only applies to the finished implementation: models and processes for the development of learning objects have no standard language or process. Rosheim also focused on the need to set realistic expectations in terms of the level of reuse that might be achieved. Reuse efforts should be tailored to the type of organisation: ‘some organisations may be ready to begin building reusable components, but most organisations are better suited to begin focusing at the use case level, or by looking at analysis and design patterns’ (Rosheim, 1999). Reusability should permeate the entire development process, therefore, and not just be an attribute of the finished product.

Conclusions

Reuse of learning objects is a major preoccupation of the LO community, but in the absence of an agreed definition of a learning object and a common understanding of reusability, the extent to which reuse can be encouraged is unclear. In the software and information systems industry, reuse in software projects remains largely aspirational. Its success depends on the culture of reuse permeating all aspects of development, for example, through agreement of a standard development process which is based on standardised models for all stages of analysis and design. The software development community is still searching for first principles, and reuse remains at a low level, although reasons for this and solutions have been identified; these findings are equally applicable to the LO community. The expectations of the learning technology community in relation to the area of reusability need to be tempered. By concentrating on the end product, the implemented (and meta-tagged) learning object, we may create unrealistic expectations and then wonder why extensive reuse is not automatic. Design for reuse should not just focus on the end products: it needs to be incorporated into the entire development process.

References

Boyle, T. (2003). Design Principles for Authoring Dynamic, Reusable Learning Objects. In *Australian Journal of Educational Technology*, 19(1), 46-58.
Also available: http://www.ascilite.org.au/ajet/ajet19/boyle.html

Brown, D.W. (2002). *An Introduction to Object-oriented Analysis: Objects and UML in Plain English, 2nd edition*, Wiley.

Buhrer, K. (2000). From Craft to Science: Searching for the First Principles of Software Development. In *The Rational Edge*. [Online]. Available: http://www.therationaledge.com/content/dec_00/f_craftscience.html [10 July 2003].

Clark, R. & Rossett, A. (2002). Learning Solutions - Learning Objects: Behind the Buzz. In *Chief Learning Officer Magazine*. [Online]. Available: http://www.clomedia.com/content/templates/clo_feature.asp?articleid=24&zoneid=30%20 [10 July 2003].

Dodani, M. (2002). The Dark Side of Object Learning: Learning Objects. In *Journal of Object Technology*, vol. 1, no. 5, November-December 2002, pages 37-42. [Online].
Available: http://www.jot.fm/issues/issue_2002_11/column3 [10 July 2003].

Douglas, I. (2001). Instructional Design Based on Reusable Learning Objects: Applying Lessons of Object-oriented Software Engineering to Learning Systems Design. In *Proceedings of 31st ASEE/IEEE Frontiers in Education Conference*, October 10-13, 2001. [10 July 2003].

Glass, R. (1999). The Realities of Software Technology Payoffs in *Communications of the ACM*, vol 42, no 2, pp. 74-79. Available: http://www.elearningmag.com/elearning/article/articleDetail.jsp?id=5043 [10 July 2003].

Johnson, L.F. (2003). Elusive Vision: Challenges Impeding the Learning Object Economy, *Macromedia White Paper*, June 2003.

Lin, F. (2001). A Critique of Stephen Downes' Article, "Learning Objects": A Chinese Perspective. In *International Review of Research in Open and Distance Learning*, July 2001 [Online].
Available: http://www.irrodl.org/content/v2.1/lin.html [10 July 2003].

Mili, A., Yacoub, S. , Addy, E., & Mili, H. (1999). Toward an engineering discipline of software reuse. In *IEEE Software* vol 16 no 5 pp. 22-31.

Polsani, P.R. (2003). Use and Abuse of Reusable Learning Objects. In *Journal of Digital Information* Volume 3 issue 4.

Rosheim, M. (1999). Reuse for Toddlers, *Rose Architect*, [Online].
Available: http://www.therationaledge.com/rosearchitect/mag/archives/fall99/f4.htm [10 July 2003].

Sosteric, M. & Hesemeier, S. (2002). When is a Learning Object not an Object: a First Step towards a Theory of Learning Objects. In *International Review of Research in Open and Distance Learning*, vol.3, no.2, October 2002.

Wagner, E.D. (2002). The New Frontier of Learning Object Design. In *The eLearning Developer's Journal*, The eLearning Guild, June 18 2002.

Wiley, D.A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In Wiley, D.A.(Ed.), *The Instructional Use of Learning Objects*, Association for Educational Communications and Technology, Bloomington.