Why do so many companies hesitate to adopt OO technology? The authors use organizational learning theory to illuminate the obstacles involved, then offer guidelines that can help managers mitigate them.

Understanding and Managing OOT Adoption

Anol Bhattacherjee and James Gerlach, University of Colorado at Denver

Object-oriented technology offers methodologies, techniques, and tools that provide clear advantages for corporate software developers and users alike. OOTs enable fast and cost-effective application development with reusable components, provide connectivity across heterogeneous computing platforms, and support integration of data, applications, and business processes.[1] As such, OOTs have been heralded as the next generation of software process technologies beyond structured methodologies, 3GL/4GL, and relational databases.

However, despite widespread awareness of OOTs' benefits and the commercial availability of reusable libraries, databases, methodologies, programming languages, and packaged applications, many businesses remain cautious about deploying OOT for major system development.[2] One important reason for this reluctance is that OOT adoption requires significant changes in organizational roles and processes, which can engender resistance. Lacking knowledge and experience with OO projects, many managers fear that their companies will be overwhelmed by the new technologies and their unanticipated impacts on mission-critical activities. Organizational learning (OL) theory helps us understand this hesitancy and suggests ways to manage OOT adoption that can be tailored to the adopting organization's specific needs.

0740-7459/98/$10.00 © 1998 IEEE

In this article, we address three questions:

1. How do companies decide whether or not to adopt OOTs?
2. What factors affect OOT adoption?
3. How can managers overcome these obstacles?

OL theory gives us a theoretical lens to explore these issues, addressing not only the ability to adopt a new technology but also how institutional and market structures such as consultants, information centers, and outsourcing agents can lower barriers to adoption by promoting learning of complex technologies such as OOT.[3] OL theory is timely given the growing currency of such concepts as "learning organizations" and "knowledge-based organizations" in the management literature.[4] Because it helps companies build the structures and processes required for adaptation, OL can enhance competitiveness in today's rapidly changing business environment.

Successful OOT adoption often requires retraining system developers, and replacing traditional systems analysis and design techniques with OO approaches.

Earlier Models

Two theoretical camps have dominated technology adoption research: innovation diffusion theory and the theory of network externalities. Innovation diffusion theory models technology adoption based on a company's communication patterns, in which prior adopters inform potential adopters of a new technology's availability and persuade them to adopt it.[5] In contrast, network externality theory argues that adopter behavior is influenced by relevant parties outside the company (such as vendors, consultants, or standards-making bodies).[6] As more companies adopt a technology, a body of accumulated wisdom develops, a pool of qualified personnel emerges, complementary products become available, and third-party support groups form to facilitate adoption. This motivates subsequent adoptions and creates a "bandwagon effect," which eventually makes the process self-sustaining.

Though these theories reasonably predict individual adoption of simple technologies such as spreadsheets and e-mail, they do not adequately explain organizational adoption of complex technologies such as OOT.[3] There are several reasons for this. First, several individuals make organizational decisions based on a common goal (for example, reduce cycle time). Moreover, companies may choose different levels of OOT adoption based on their needs and technological sophistication (for example, adopt OO programming languages but not OO analysis and design techniques), as opposed to the binary adoption/nonadoption decision for most individual-use technologies.

Second, both theories assume that all adopters have equal abilities to adopt and are constrained only by the availability of information—hence, employing appropriate information sources (internal or external) should enhance adoption. However, despite widespread knowledge of OO benefits and tools, disseminated via journals, professional associations, and vendors, OOT has been unable to replace entrenched system development practices. We argue that a company's adoption behavior derives from its ability to understand and use OOT.

Third, a comprehensive model of technology adoption should consider both internal and external influences, then examine their effects in light of organizational ability to learn and the availability of market-based mechanisms to promote that learning. OOTs exhibit increasing returns as knowledge and experience from early OO projects influence subsequent projects in the same or different companies. Consultants, vendors, and outsourcing agents capture this knowledge and act as "change agents" in promoting OOT to others.[7] However, this requires a critical mass of adopters. By acknowledging a company's ability to learn and promote that learning, the OL perspective addresses both the demand and supply sides of OOT adoption.

Organizational Learning Theory

OL refers to the process of acquiring knowledge, expertise, and insights about complex innovations, and institutionalizing this wisdom by modifying organizational roles, processes, structures, routines, strategies, technologies, beliefs, and values as needed.[8] Successful OOT adoption often requires retraining system developers, encouraging cross-functional sharing of reusable components, and replacing traditional systems analysis and design

techniques with OO approaches.

Though OL arises from individual learning, it represents more than the cumulative effect of individual learning.[8] Individual learning results from individual cognitive effort and is constrained by individual abilities and access to resources. In contrast, OL results from a company-wide change in behaviors, mental maps, norms, and values, which provides a basis for subsequent organizational actions and outlasts the tenure of individuals initiating that learning. OL is incomplete until institutionalized—typically the case with companies that distance their OO projects from their mainstream system development.

OL may be incremental or radical. Incremental (evolutionary, single-loop) learning is short-term, localized, and based on repetition of past behaviors; this typically leads to minor refinements in organizational process (such as changes in formal rules). Radical (revolutionary, double-loop) learning is long-term, pervasive, and based on the development of heuristics, insights, and tacit knowledge; this can lead to radical changes in organizational norms, frames of reference, and assumptions.[8] For most companies, upgrading a relational database (RDB) requires incremental learning because it minimally impacts organizational processes and actors, while migrating software projects from structured approaches to OO will require radical learning.

Forcing OOT on traditional system developers may result in resentment, turf wars, and subsequent project failure because people resist change and generally dislike radical learning. Successful OO projects require building relevant knowledge incrementally, diffusing interest in OOT among different constituencies, gradually modifying organizational roles and processes, and carefully orchestrating both the technological and human dimensions of OL.

Factors Affecting Organizational Learning

Studies indicate that several factors influence a company's ability to learn OOT.

The technology's complexity and immaturity

Many people consider OOT difficult to understand and use because it requires adopting a new lexicon, learning new concepts (such as encapsulation, inheritance, and polymorphism), implementing new data modeling and programming techniques, and so forth.[1,9] This imposes a formidable learning requirement on incumbent companies. This situation is exacerbated by the absence of complementary tools to support OO development. Because OOT is still evolving, it lacks comprehensive, domain-specific class libraries that developers can use directly in OO projects.[9] Also, few tools exist to integrate OOT with other technologies such as RDBs;[7] replacement technologies such as OO databases do not yet provide the efficiency and scalability that industry demands.

Knowledge diversity

The breadth of in-house experience and expertise can partially offset technological difficulties in OOT adoption. Companies with greater knowledge diversity may have OO-literate people who can initiate and manage the OL required for successful OO projects. Most early OOT adopters (including Hewlett-Packard, IBM, and Toshiba) are large, technology-based companies with substantial in-house knowledge.

Forcing OOT on traditional system developers may result in resentment, turf wars, and subsequent project failure.

Compatibility with organizational culture

OOT adoption represents a major cultural change for most system developers trained in structured approaches and tools. It also requires creation of new roles (for example, object architects, class designers, and object programmers) and new work processes (such as reuse-based system development). Further, a large installed base of prior, mature technologies such as procedural methodologies and RDB creates an inertial effect, hence resistance to new technologies.[1] Radically changing system development approaches not only disrupts organizational functioning but also alters power distribution within the company as traditional developers lose control of systems they built or manage. Companies with open cultures receptive to change will find it easier to learn OOT.

Radical OL for OOT adoption is expensive. OO projects require extensive training and outside consultants are expensive, as are OO system develop-

ment tools. Incentive mechanisms must be built to promote reuse. Learning to use OOT effectively may take several years, and the time spent on learning may detract from ongoing system development and maintenance efforts. Furthermore, because the costs of OOT adoption are immediate, while potential benefits such as reuse are long-term and uncertain, managers may face difficulty cost-justifying OO projects. A company with adequate slack resources may find adoption easy because it can shelter OO projects from organizational cost-reduction drives.

Many OO projects fail because management does not grasp the magnitude of OL required and cannot effectively manage the change.

Industry competitiveness

A company may be willing to learn OOT and implement necessary organizational changes if it helps them improve their competitive position. Software companies such as Oracle, Baan, and SAP have built OO interfaces to their current products because they view technological advancement as critical to survival in the highly competitive software industry. Organizations in less competitive industries may be less willing to undertake such extensive changes and thus less motivated to adopt OOT. Furthermore, if the external environment is too dynamic and complex, less technologically sophisticated companies may postpone OOT adoption until the environment stabilizes—a "wait-and-see" adoption policy.

Guidelines for Managing OOT Adoption

Many OO projects fail because management does not grasp the magnitude of OL required and cannot effectively manage the change.[2] Here we synthesize our experiences with recent OO projects at US West and TCI, and several published case studies, into eight guidelines for building the OL needed for successful OOT adoption.

Select an appropriate level and pace of OL

Complete OOT adoption represents a major paradigm shift, from procedural system development to a new and evolving object-based paradigm. This means modifying all stages in the software life cycle—from requirements modeling to analysis, design, coding, testing, and maintenance. This radical process and cultural change may take several years to realize.[9]

Given that radical learning is difficult and potentially disruptive, a company should evolve a gradual, incremental learning strategy. The level and pace of OL should accommodate business needs as well as key players' comfort level. Selected OO projects should have clearly defined boundaries that can be implemented within budget and time, and their scope should be neither unwieldy nor insignificant. Such a strategy informed US West's stepwise approach to implementing OOT, which used C++ and the Forte OO programming environment to develop a customer billing application as a first step toward learning OOT. Subsequent steps, according to the project manager, will include incorporating Web forms and Java, and planning for reusable objects.

Build realistic expectations

OO projects are often criticized as promising too much and delivering too little.[11] Setting realistic expectations and conveying them clearly to all involved parties, including system developers, consultants, and managers will motivate OOT adoption. Management can also help by better scheduling and budgeting the transition process, and by designing metrics that can measure adoption success more accurately.

Employ change agents

A company lacking adequate in-house expertise should consider employing external vendors or consultants for building knowledge diversity. OO projects benefit from collaboration among internal developers and external specialists acting as mentors. External consultants can act as "change agents" in jump-starting an OO project while also bringing in valuable industry-specific best practices and "spreading the gospel." TCI, for example, hired external consultants to manage internal OO projects. Consultants' prior OO project experience helps them capture economies of scale in learning, which can be disseminated to clients at low costs.

However, because consultants are typically viewed as "outsiders," their efforts may invite skepticism and resentment from internal staff; managers must therefore maintain open communication

between the project team and internal staff. Eventually, as the company builds OO expertise and implements necessary organizational changes, it may discontinue consultants' services and employ in-house experts for future projects.

Encourage learning-by-doing

OOT's complexity and frequent incompatibility with existing development approaches and philosophies make formal training inadequate if not complemented with a learning-by-doing program. A continuous learning environment may require small teams of internal developers and on-the-job mentors working on OO projects, as revealed in the CAE-Link case.[3] Such teams can facilitate diffusion of technical expertise that is otherwise difficult to obtain, and help build knowledge diversity that can be reused in subsequent OO projects.

Leverage existing technologies

Incremental learning requires blending OOT into current system development processes and technologies to realize benefits without disrupting existing processes. Familiarity with existing technologies and processes helps system developers appreciate, understand, and learn OOT. Fusing OOT with a mature technology such as RDBs helps overcome the inertial effect of adopting a new technology while reducing chances of adverse impacts on mission-critical activities. For example, US West decided to retain its existing RDB-based production databases (instead of migrating to OO databases) and use C++ and Forte for building OO applications to interact with these back-end RDBs. However, inappropriate integration with existing technologies increases the learning barrier and, more significantly, leads to a new generation of legacy systems.

Build an OO architecture

This provides a detailed blueprint of products, languages, databases, modeling techniques, middleware, CASE tools, class libraries, people, and processes needed to guide and manage an OO project.[9] Many OOT benefits depend on the architecture's completeness and consistency. For instance, an OO programming language may yield little benefit unless combined with a compatible analysis and design methodology, knowledgeable developers, a workable database access approach, mechanisms to promote reuse, and so forth.

An effective OO architecture is the end result of the OL process, and should serve as the basis for subsequent OO projects. Keep architectural demands simple and consult proven architecture examples from other companies, if available. If needed, hire professional OO architects well versed in OO design principles, alternative system architectures, project management techniques, and the application domain.

Continuously scan for new toolsets

Lack of complementary toolsets such as reusable components and RDB integration tools greatly increases the costs of learning OOT. Attempting to build these infrastructure tools from scratch often leads to wasted time and subsequent project delays. Management seldom views such efforts positively because they do not add any value to organizational functioning.

However, such toolsets are slowly becoming available as third-party vendors develop reusable business components and object libraries specific to, for example, insurance, manufacturing, securities, and publishing. Many RDB vendors such as Oracle, Sybase, and Informix are increasing OO support of their RDBs via binary large object data types, GUI support, rules and triggers, and ANSI SQL 3.0 support, thus providing a migration path from RDB technology to OOT.

Middleware solutions such as Sun's JDBC and Java Object Environment will also enable efficient and flexible distributed OO solutions that interact with RDB back ends. Baan, SAP, and SSA offer comprehensive, integrated OO systems covering core business processes for inexperienced users. Companies must continuously scan the marketplace for the latest relevant tools to reduce the learning curve associated with OOT adoption.

Many system developers believe that using OOT will automatically generate reuse, but such reuse has thus far proved elusive.

Institutionalize learned behaviors

The fruits of OL cannot be realized or retained long-term unless learned behaviors are institutionalized. Many system developers believe that using OOT will automatically generate reuse, but such reuse has thus far proved elusive.[9] Experience at Hewlett-

Packard shows that reuse only follows conscious planning and careful management of human issues. This includes gaining agreement on the desired level of reuse, designing mechanisms to promote reuse across functional boundaries, and developing standards for building reusable objects. Ultimately, incentives drive reuse; without them, project expediency may take priority over building for reuse.

OOT adoption imposes significant learning requirements by engendering major changes in organizational roles, processes, and assumptions. While OL of this magnitude is difficult and painful, it can provide significant long-term gains if managed effectively.

Our strategies for managing the transition to OOT emphasize the learning abilities of adopting companies. Committing to OOT adoption is an important managerial decision and affects careers, projects, and entire organizations. Some consultants and researchers advise against adopting OOT until it matures into a plug-and-play technology, while others recommend total immersion to fully realize OO benefits. We offer a balanced perspective based on an incremental learning process that not only informs managers of important constraining factors but also prescribes guidelines for managing OOT transition. The proposed guidelines' effectiveness will ultimately depend on the company's ability to learn and managers' ability to manage change. ❖

Acknowledgments

The authors thank the associate editor and five reviewers for their valuable comments and suggestions on an earlier draft of this paper.

References

1. R.G. Fichman and C.F. Kemerer, "Adoption of Software Engineering Process Innovations: The Case of Object-Orientation," Sloan Management Rev., Winter 1993, pp. 7-22.
2. M.E. Fayad, W. Tsai, and M.L. Fulghum, "Transition to Object-Oriented Software Development," Comm. of the ACM, Feb. 1996, pp. 108-121.
3. P. Attewell, "Technology Diffusion and Organizational Learning: The Case of Business Computing," Organization Science, Feb. 1992, pp. 1-19.
4. D.A. Garvin, "Building a Learning Organization," Harvard Business Rev., Jul.-Aug. 1993, pp. 78-91.
5. E.M. Rogers, Diffusion of Innovations, Free Press, New York, 1995.
6. M.L. Katz and C. Shapiro, "Technology Adoption in the Presence of Network Externalities," J. Political Economy, 1986, pp. 822-841.
7. B.C. Hardgrave, "Adopting Object-Oriented Technology: Evolution or Revolution," J. Systems Software, 1997, pp. 19-25.
8. C. Argyris and D. Schon, Organizational Learning, Addison Wesley Longman, London, 1978.
9. R.G. Fichman and C.F. Kemerer, "Object Technology and Reuse: Lessons from Early Adopters," Computer, Oct. 1997, pp. 47-59.
10. W. Wessale, D. Reifer, and D. Weller, "Large Project Experiences with Object-Oriented Methods and Reuse," J. Systems Software, 1993, pp. 151-161.
11. M.E. Fayad and M. Cline, "Managing Object-Oriented Software Development," Computer, Sept. 1996, pp. 26-31.

About the Authors

James H. Gerlach is an associate professor of information systems at the University of Colorado at Denver. His research interests include software engineering and management of information systems technology. Gerlach holds an MS in computer science and a PhD in management, both from Purdue University. He is a member of IEEE and ACM.

Anol Bhattacherjee is an assistant professor of information systems at the University of Colorado at Denver. His research interests include information technology adoption and use, and management of information technology within organizations.

Bhattacherjee received a PhD in information systems and an MBA from the University of Houston, and an MS in exploration geophysics from the Indian Institute of Technology, Kharagpur, India.

Address questions about this article to Gerlach at the Dept. of Information Systems, College of Business and Administration, University of Colorado at Denver, Denver, CO 80214-3364; jgerlach@evans.cudenver.edu.