THE NEED FOR A STRATEGIC ONTOLOGY

Kay M. Nelson
Fisher College of Business
The Ohio State University
2100 Neil Avenue
Columbus, OH 43210
(614) 292-7692
fax (614) 292-2118
nelson_k@cob.osu.edu

H. James Nelson
Fisher College of Business
The Ohio State University
2100 Neil Avenue
Columbus, OH 43210
(614) 292-3903
fax (614) 292-2118
nelson_j@cob.osu.edu

ABSTRACT

Although it is often not obvious or intuitive, advances in technology have historically driven organizational strategy that incorporates technical components. This has not proven to be successful. We advocate that advances in technology should be driven by strategy, and that one way of accomplishing this is to have strategic ontologies that serve as foundations for strategic patterns, than arbitrary technological patterns. This paper describes ontologies and how they have been used, somewhat loosely, in the definition of technological patters for software development. It discusses the implementation of enterprise resource planning (ERP) systems, and how they could have been more successful had they been introduced with strategic patterns based on a social ontology. In conclusion, this paper encourages the MIS community to lead strategic IT initiatives, rather than follow the opportunities created by technology.

Keywords: Ontology, Strategy, Architecture, Patterns, Business processes.

INTRODUCTION

As technology increases at a rapid rate, it presents organizations with new and varied opportunities. Unfortunately, these new technical opportunities often drive strategic positioning by organizations, rather than having technology use being driven by the organizational strategy. ERPs are an example of this phenomenon. While organizations often thought they were strategically changing their processes for competitive advantage, they were, in effect, having their strategy implicitly driven by the processes embedded in the software – the technological imperative at work.

MIS professionals and academics, versus computer scientists, have both an opportunity and an obligation to mitigate technology from driving strategy. While MIS has successfully impacted the standardized operation of computing through the creation and support of standards such as SEI/CMM, our discipline has failed to keep senior non-MIS managers from being at the mercy of technology projects that have up to a seventy percent failure rate (Hildebrand, 1998). This treatise proposes that one way to significantly shift the balance of technological power from the technologists that create the technology to the managerial organizations that use technology can be accomplished by creating ontologies for emerging technologies that are written in a language that business decision makers understand and control. By articulating business technology needs through understandable and consistent ontologies, business will proactively drive competitive needs down to the technologists to configure, rather than attempting to “mate” with what the technologists present to them. Business executives are not going to learn to speak “techie,” and the time has come for the MIS community to create technological solutions in the language of business.

ERPs are an example where a lack of understandable standards and ontology has contributed to massive failures (Sumner, 2000). Since technical terms are often unclear, non-standard, and not described in the language of business, many managers are at a disadvantage when negotiating with technical firms who have essentially created the rules and ontology of outsourcing whether they are implicit or explicit. These examples demonstrate the need for the MIS discipline to become actively involved in standard setting and ontology development expressed in the language of business.

As a discipline, we are uniquely positioned to be the gatekeepers between business executives and basic technical scientists. We can accomplish this by setting standards of semantic consistency that are understandable to the people who make technology ultimately useful, high level executives. While the CIO often plays this translation role, it is virtually impossible to keep track of every new technology becoming available. MIS academicians need to get involved as new technologies develop, helping define and develop standard semantics and ontologies. By doing so, we can significantly increase the success of technology projects which will be driven by strategic needs rather than being fit to technology. This should also increase the speed by which non-IT executives understand the implications and opportunities presented by new technologies and improve the quality of their IT investment decisions.

The purpose of this paper is not to create a specific standard but to send a challenge to the MIS community to use our unique position between business and computer science to set IT standards using consistent, business-based semantic ontologies. The next section of this paper discusses ontologies from the technologist's perspective, highlighting the inconsistencies inherent even within one discipline. The following sections discuss ontology as a social phenomenon, arguing that it is possible to create ontologies for IT that business can understand and use. Next, we discuss patterns and how they also vary by author and lack consistency. This leads to a discussion of how our discipline failed to predict and prevent many ERP failures – failures that may have been reduced by a more fundamental understanding of the technology by the business executives making these large investments.

In conclusion, we call the MIS community to action to take the lead in creating semantically consistent ontologies for technology opportunities, in the language of business rather than past ontologies such as UML, CORBA, or XML. We need to insure that MIS investments are made wisely, and standard setting for language and patterns is one way to address this issue, with MIS academicians and practitioners leading in a collaborative manner.

ONTOLOGIES

There many definitions for the term *ontology*. Guarino and Giaretta (1995) identify seven different interpretations. Many of the definitions for ontology tend to be technical, as the art and practice of constructing ontologies has moved from a relatively obscure philosophical area of study to an important area of software engineering and knowledge management. Some sample definitions can be seen below:

> *Ontologies aim at capturing static domain knowledge in a generic way and provide a commonly agreed upon understanding of that domain, which may be reused and shared across applications and groups. (Pinto & Martins, 2001)*
>
> *An ontology can be thought of as a sharable representation of knowledge that: (1) provides a shared terminology that various applications can jointly understand and use; (2) defines the meaning of each term (a.k.a. semantics) in a precise and unambiguous*

> *manner; and (3) implements the semantics in a set of axioms that allows one to automatically deduce the answer to many "common sense" questions. (Gruninger, Schlenoff, Knutilla, & Ray, 1997)*
>
> *In our work, an ontology serves as a partial specification of the knowledge representation to be built in a later stage. The specification is partial because it supplies concepts in which states of affairs can be expressed but does not actually specify states of affairs. (van der Vet & Mars, 1998)*
>
> *An ontology is a representation vocabulary specialized in some domain or subject matter. … Ontologies are quintessentially content theories, because their main contribution is to identify specific classes of objects and relations that exist in some domain. (Falbo et al., 2002)*
>
> *In the simplest case, an ontology describes a hierarchy of concepts related by subsumption relationships; in more sophisticated cases, appropriate axioms are added to express other relationships between concepts and to constrain their intended interpretations. (Wang, Chan, & Hamilton, 2002)*

One of the most concise and most often cited definitions for ontology is: "An ontology is an explicit specification of a conceptualization" (Gruber, 1993b), or more specifically, a systematic account of Existence (Gruber, 1993a). A conceptualization is an abstracted, simplified view of the world containing the things, the concepts, and the other entities of interest along with the relationships that hold between them (Genesereth & Nilsson, 1987). An ontology can be thought of as a vocabulary (a set of words), a grammar (the set of rules for combining words into larger structures), and semantics (the meanings of the words and the larger structures) all defined within a specific domain.

An ontology begins with a particular view of a domain, for example: the interaction of subatomic particles (physics), how the human body moves (biomechanics), the definition and manipulation of data (various programming languages and databases), and areas of business such as strategy, finance, management, accounting, marketing, information systems, and so on. This view, typically a shared view among practitioners within the domain, contains the foundational theories and axioms of the things and "how the things work" within the domain. The ontological foundation allows the practitioners to reason about and understand their domain (Gruninger et al., 1997).

An ontology is a formalization of this domain-view foundation. The ontology identifies the objects of interest within the domain and assigns various symbols to represent them in a declarative formalism. This set of objects becomes the universe of discourse. For example, an ontology based in technical language that formally describes processes (a "data flow diagram") may include objects such as *external entities*, *processes*, *data stores*, and *data flows*. Further, there is a set of rules that defines when a process description is "correct," such as how data flows stand between processes, data stores, and external entities, and so on.

There are several design criteria for formal ontologies (Gruber, 1993a). An ontology must be clear and effectively communicate their intended meaning through objective, complete, and correct definitions. It must be coherent so that it supports inferences consistent with its definitions. It must be extendable so that it anticipates new uses of the shared vocabulary without changing the existing definitions. It must have minimal encoding bias so that the meanings in the world are preserved without making representation choices merely for the

convenience of the representation. Finally, it must have a minimal ontological commitment, allowing the practitioners in the domain the freedom to specialize and instantiate the ontology as needed.

Ontologies are useful because they encourage the standardization of the terms used to represent knowledge about a domain (Jurisica, Mylopoulos, & Yu, 1999). It is very difficult to describe phenomena using a natural language even with a shared domain view. An ontology gives a formal, common language for representing, sharing, reasoning about, and making inferences from knowledge. This formalization of a domain of knowledge allows the formation of industry-wide standards (McGuinness, 2002). For example, NIST (the National Institute of Standards and Technology) is encouraging the development of controlled vocabularies and ontologies. RosettaNet (http://www.rosettanet.org) is a consortium of organizations in information technology, electronic technology, electronic components, and semiconductor manufacturing that is using ontologies to develop e-business standards and a language for business processes. Unfortunately, all of these ontologies are written in technical languages.

SOCIAL ONTOLOGY

A social, semantically consistent approach to strategic technology ontologies has the potential to capture a more complete description of the realities of using technology for strategic business purposes. Unfortunately, we (as a discipline) have failed to make this a standard method in describing strategic business needs to implementers. Project managers do not get to use their knowledge of social issues to influence technical architecture (Cockburn, 1996). Similarly, social and transactional knowledge of executives is not communicated to the project level managers, making the problem more dire. Technologists see architectures as “blueprints” for managing the construction and operation of the computing and telecommunications operations of the organization. However, architectures should also be representations of the social reality of the organization (Cockburn, 1996).

Habermas (1990) describes three separate “worlds”: the object world, the social world, and the subject world (Weigand, De Moor, & Heuvel, 2000). While the ontologies and patterns created by technologists do a good job of representing the object world (and a sometimes adequate job of describing the social world), the subject world is represented from their perspective of computer codes, technical objects, and technical transactions. A key to creating ontologies that lead to patterns that truly represent the social and subject world of the strategic business is to have the MIS community act as advocates for the non-MIS executive to insure that a semantically consistent ontology exists as a foundation for strategic patterns. Although some work has been done in this area with a focus on business communication semantics (Kimbrough & Moore, 1997; Winograd & Flores, 1986), the results are quasi-ontologies that are incomprehensible to non-technologists.

Semantically consistent patterns and ontologies have lifted the level of the design discourse (May & Taylor, 2003) and contributed to improved understanding and faster translation of knowledge and requirements between technologists. Since most large IT projects fail between the interpretation of the strategic business need and the technological implementation (Luftman & Brier, 1999; Luftman, 1996), the use of social ontologies and patterns at the strategic level will make a substantial contribution to standards led by the MIS community, plus may potentially save organizations millions of dollars in misunderstood requirements and technology capabilities.

PATTERNS

An organization encounters many problems in its day-to-day operations and in its strategic positioning against other organizations. These problems occur over and over in slightly different forms but with the same fundamental characteristics. A pattern is a "core of the solution" to these common problems (Alexander et al., 1977). Recognizing that a problem is one that has been seen before, and applying a semi-customized solution pattern to it, allows faster reaction times, and it "enables efficiency in both the communication and the implementation of software design, based on a common vocabulary and reference" (Adams, Koushik, Vasudeva, & Galambos, 2001, p. 11). However, the common vocabulary and reference may not be so common. The business side of the organization has the task of recognizing and communicating about the problem. However, the language of the solution pattern is often derived from technology rather than communicated in the language of business. In other words, from a technological ontology rather than a strategic ontology. We use patterns in this paper as an example of where a social, business semantic ontology would make the approach more successful.

For example, Schmidt and Buschmann (2003) provide an overview of patterns and describe three main types of patterns. Design patterns refine the elements and the communication structure of a software system. Architectural patterns express the overall structural organization of a software system by defining subsystems, subsystem responsibilities, and subsystem interrelationships. Pattern languages are a set of patterns that define a vocabulary for talking about software development.

Perhaps the most technical, earliest of the pattern works was written by Gamma, Helm, Johnson, and Vlissides (1995), well known as the "gang of four." This book describes design patterns, which describes solutions to problems in object oriented design. This is a very low-level, implementation oriented approach where software design problems are matched with object oriented solutions that describe the objects, relationships, responsibilities, and collaborations. The authors provide a catalog of twenty-three design patterns to help software developers create more efficient, easier to maintain object oriented code. Despite this very technical beginning, the concept of design patterns has been stretched from a technological solution set to be called upon at implementation time to a driver of organization strategy.

For example, organizations are trying to understand the external strategic and the internal tactical problems of e-commerce and e-business. In their book, Adams, Koushik, Vasudeva, and Galambos (AKV&G) (2001) discuss patterns for e-business. "These patterns help us understand and analyze complex business problems and break them down into smaller, more manageable functions that can then be implemented using lower-level design patterns" (Adams et al., 2001, p. 13). They "extend the domain of software patterns to earlier phases of the application development life cycle" by introducing four major types of patterns for e-business: Business, Integration, Application, and Runtime patterns.

Business patterns and Integration patterns are the highest level patterns, with Application and Runtime patterns representing successively more implementation-oriented solutions. Business patterns are closest to the strategic problems of the organization. They describe the high-level components of the solution as well as the interactions between these components. The components of the solution are: the solution users (customers, investors, etc.), the environment in which the users interact (company, organization, system, etc.), and the organizational data. AKV&G describe four examples of Business patterns: self service (user to business, collaboration (user to user), information aggregation (user to data), and extended enterprise (business to business).

Although the Business pattern is supposed to be closest to the organizational view of problems, it is described primarily in technical terms. The self-service Business pattern, for example, is described as typically consisting of three things: users, a network, and enterprise systems as shown below in Table 1. The other three sample Business patterns are described in similar technical terms.

Table 1. Self-service Business pattern.

Users	• In the enterprise, in partner organizations, or any other location. • Accessing the solution using a Web browser.
Network	• Based on TCP/IP and other Internet technologies. • Can be a dedicated LAN, broadband, or dial-up connection
Enterprise systems	• Custom systems • ERP systems such as SAP, BAAN, and PeopleSoft • Databases

Integration patterns are similar to Business patterns in that they are close to the organizational view of problems. They are also similar to business patterns as they are described in technical terms as shown below in Table 2. The other two types of pattern, application and runtime patterns, are very technical and represent the patterns used to implement the business and the integration patterns.

Table 2. Access integration pattern.

Users	• In the enterprise, in partner organizations, or any other location. • Accessing the solution using a Web browser.
Network	• Based on TCP/IP and other Internet technologies. • Can be a dedicated LAN, broadband, or dial-up connection
Business applications and data	• Custom developed systems (old and new) • ERP systems • Databases
Access integration services	• Device support • Presentation • Personalization • Security and administration

Table 2 makes a great deal of sense to a technologist, but does not give the non-IT business executive a conceptualization of what this pattern contributes to the business.

These patterns are useful for solving many problems that occur as organizations try to tackle e-business. They are indeed the “core of the solution” to these business problems. However, there is a disconnect between recognizing the problem and applying the pattern. An organization’s executives may recognize a strategic e-business problem but need to find and implement a strategic, rather than jump into a patterned technical, solution. This solution may eventually be realized as business, integration, application, and runtime patterns, but the patterns described by AKV&G are more useful for technicians who need to implement the solution than for executives who need to recognize and solve a problem at the strategic level.

AKV&G note that there is a business problem that needs to be matched to the pattern, which leads to a technical solution. They begin with the business problem (but disconnected from the business strategy) to define their patterns. Other authors note the business problem, but design

their patterns around very technical concepts, missing the step of decomposing the business problem into business patterns and moving directly from business problem to technical pattern. For example, Fowler (2003) discusses enterprise applications and the patterns that can be used to support them. In his introduction, Fowler describes several business problems: a business-to-consumer online retailing, processing leasing agreements, and tracking expenses. Each of these problems has a different enterprise architecture solution. However, the patterns composing the architectures are highly technical. Patterns such as a *Remote Façade*, a *Data Transfer Object*, a *Transform View*, or a *Front Controller* are very useful for defining and implementing enterprise application software, but are little help in identifying and/or describing the overarching business problem. Again, there is a gap between the technical language of the pattern and the business language of the organization, making pattern technology of limited use to executive decision-makers.

Robertson and Sribar (R&S) (2002) note that there is a "clear misalignment" between the business organization and the IT organization. Executives work with consultants to determine the strategic direction for the organization, but this strategic direction reaches the infrastructure planners only through rumors and hearsay (Robertson & Sribar, 2002). However, R&S's patterns are also technical. They are derived from the application infrastructure consisting of transact patterns, publish patterns, and collaborate patterns. The disconnect between the language of the business and the technical language still exists.

There is no doubt that patterns have had a positive impact on the way that software solutions are found, are designed, and are implemented. Patterns have altered the way that developers speak and think about software design (May and Taylor, 2003). When properly applied, patterns aid the capture, transfer, and management of software design knowledge. These patterns begin and end with the technical side of software development. Unfortunately, little progress has been made in a top-down approach to pattern definition, beginning with strategic patterns, moving to business problem patterns, and then into software patterns, all with consistent business-oriented specifications and conceptualization and language. This is where MIS professionals need to step in and define standards through a strategic pattern ontology.

THE LESSON FROM ENTERPRISE RESOURCE PLANNING

The evolution of Enterprise Resource Planning (ERP) in organizations illustrates the need for business-based ontologies that are semantically consistent and can be understood by executives who hold the purse strings. Less than 30% of ERP installations are considered successful (Brown, 2001). This has been attributed to many causes, such as: CEOs who do not understand the strategic implications of ERP, consultants, users, lack of fit between requirements and product capabilities, and so on (Al-Mashari, 2001; Willcocks & Sykes, 2000). Nowhere in the literature has it been suggested that a lack of standardized language about these products caused this great deal of confusion and led to many poor investments. ERPs were sold as strategic enterprise solutions, even though at their core is a set of integrated, somewhat standardized business processes (Lee, Siau, & Hong, 2003). This approach to product development had virtually no strategic intent, and therefore ERPs, while sometimes solving process level problems, often create more strategic problems than they solve (Davenport, 2000; Ezingeard & Chandler-Wilde, 1999). Although advertised as flexible, recent articles have identified the need for far move flexibility in ERPs (Al-Mashari, 2001; Willcocks & Sykes, 2000).

The MIS academic community was "asleep at the wheel" with the advent of ERPs. This is clear since most of the extant literature on the subject reports on practice, rather than proactively prescribing solutions and standards for what has been one of the biggest IT investments in the

last decade. We had a minimal role in ERP development or implementations, and we did nothing to help business managers avoid multi-million dollar mistakes. We did not translate the vendor or technical jargon to decision-makers in language they could understand and we did not identify the potential pitfalls of these monolithic systems. Had MIS academicians and professionals presented ERPs to senior management in an ontology based on business language, the results of these installations may have been much improved or the installations may not have been viewed as viable investments.

There is support from the MIS research community for the creation of standards for ERP and other technologies. Willcocks and Sykes (2000) suggest that MIS leadership devise strategy and structures to ensue that technology delivers value. We propose that standardized patterns based on business-based ontologies are a critical element of doing this. Interestingly, in the same article, Willcocks and Sykes argue for architectural planning but do not link it to strategy and structures. This represents the disconnect that is rampant in the MIS literature. Sprott (2000) suggests that ERPs need to be adaptable to the complexities of today's business, but again, fail to suggest deriving this adaptability from the strategic level. Without intervention and a mind shift on the part of MIS academics, we will continue to report on failures resulting from the disconnect between technology and strategy, and we will miss the opportunity to be the leaders in making technology truly ubiquitous by presenting it in business ontology form and making technology decisions as easily accessible to senior executives as other strategic decisions.

CONCLUSION

Technology project successes and failures have shown that greater structure, standards, and understanding at higher levels greatly improves the chances for a successful and useful implementation (Schmidt, 1996). Structured software development techniques organizes "spaghetti code" into more easily understood blocks. Object oriented software development techniques organize code and data. Software development patterns further organize object oriented code into "chunks" that match generally recognizable development problems. All this is good, but it begins with the very technical and stretches toward (but is still far from reaching) organizational strategic problems. This is starting from the wrong direction if technology is to lead to business value and competitive advantage. This disconnect between organizational strategy and technology is one of the causes for the enormous amounts of money wasted on failed technology projects (Luftman & Brier, 1999; Luftman, 1996).

We propose that MIS academics, professionals and non-IT executives begin the investigation of standardized strategically-oriented ontologies as the basis for strategic pattern development. Organizations such as the Society for Information Management (SIM) can play a key role in both developing and certifying these standards. These strategic patterns can, in turn, drive the development and fit of technology patterns so that a seamless path from organization strategy to implemented technology may be forged with business driving technology, rather than the reverse that often happened with ERP implementations. We have a responsibility not to repeat the costly mistakes of the ERP era.

This paper defines a problem and describes an approach rather than prescribing standards. This must be done as a community of academics and practitioners. It is a "call to action" for MIS academicians to begin leading practice, instead of merely reporting on its "state of the art." There is a true need for a greater chance of success and return of IT investments. A strategic ontology linked to technology implementation may be one solution that will lead to these greater returns.

We believe that an ontology based on strategic business language that describes strategic technical needs can lead to true strategic business patterns that can then be used as requirements that link to technical patterns. This approach builds on a promising new technology method – patterns. As MIS professionals, we have the unique opportunity to develop standards that help the people that fund technology investments understand more clearly what they are paying for. At the same time, we are enabling technologists to extend the pattern concept in a way that will lead to more successful IT projects and more satisfied users and executive stakeholders.

We are currently engaged in an empirical proof of concept study in a major insurance company. The CTO (an expert in patterns, and who is referenced in this paper), with the cooperation of the CIO, is giving us access to all levels of the organization from strategic planning through IT pattern implementation. This study will develop a strategic ontology that can then be tested and improved in future research.

REFERENCES

Adams, J., Koushik, S., Vasudeva, G., & Galambos, G. (2001). *Patterns for e-Business: A Strategy for Reuse*. Double Oak, TX: IBM Press.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., I., F.-K., & Angel, S. (1977). *A Pattern Language*. New York: Oxford University Press.

Al-Mashari, M. (2001). Process Orientation through Enterprise Resource Planning (ERP): A Review of Critical Issues. *Knowledge and Process Management, 8*(3), 175-185.

Brown, J. (2001). ERP Doomed by Poor Planning. *Computing Canada, 27*(3), 11-12.

Cockburn, A. (1996). The Interaction of Social Issues and Software Architecture. *Communications of the ACM, 39*(10), 40-46.

Davenport, T. (2000). *Mission Critical: Realizing the Promise of Enterprise Systems*. Cambridge: Harvard Business School Press.

Ezingeard, J., & Chandler-Wilde, R. (1999, November 1999). *Evaluating How ERP Can Provide Competitive Advantage: Basis for a Research Framework.* Paper presented at the Sixth European Conference on IT Evaluation, Brunel University.

Falbo, R. A., Guizzardi, G., Natali, A. C. C., Bertollo, G., Ruy, F. F., & Mian, P. G. (2002, July 2002). *Toward Semantic Software Engineering Environments.* Paper presented at the International Conference on Software Engineering and Knowledge Engineering, Sant'Angelo d'Ischia, Italy.

Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Boston: Addison-Wesley.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading: Addison Wesley.

Genesereth, M. R., & Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. San Mateo: Morgan Kaufmann Publishers.

Gruber, T. R. (1993a). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino & R. Poli (Eds.), *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Dordrecht: Kluwer Academic Publishers.

Gruber, T. R. (1993b). A Translation Approach to Portable Ontologies. *Knowledge Acquisition, 5*(2), 190-220.

Gruninger, M., Schlenoff, C., Knutilla, A., & Ray, S. (1997). Using Process Requirements as the Basis for the Creation and Evaluation of Process Ontologies for Enterprise Modeling. *SIGGROUP Bulletin, 18*(2), 52-55.

Guarino, N., & Giaretta, P. (1995). Ontologies and Knowledge Bases: Toward a Terminological Clarification. In N. Mars, J. I. (Ed.), *Toward Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*. Amsterdam: IOS Press.

Habermas, J. (1990). *Moral Consciousness and Communication Interaction*. Cambridge: Polity.

Hildebrand, C. (1998). If At First You Don't Succeed. *CIO Enterprise Magazine*.

Jurisica, I., Mylopoulos, J., & Yu, E. (1999). *Using Ontologies for Knowledge Management: An Information Systems Perspective.* Paper presented at the Annual Conference of the American Society for Information Sciences, Washington, DC.

Kimbrough, S. O., & Moore, S. A. (1997). On Automated Message Processing in Electronic Commerce and Work Support Systems: Speech Act Theory and Expressive Felicity. *ACM Transactions on Information Systems, 15*(4), 321-367.

Lee, J., Siau, K., & Hong, S. (2003). Enterprise Integration with ERP and EAI. *Communications of the ACM, 46*(2), 54-60.

Luftman, J. F., & Brier, T. (1999). Achieving and Sustaining Business-IT Alignment. *California Management Review, 42*(1), 109-123.

Luftman, J. N. (Ed.). (1996). *Competing in the information age: Strategic alignment in practice*. New York: Oxford University Press.

May, D., & Taylor, P. (2003). Ontology Applications and Design. *Communications of the ACM, 45*(2), 39-41.

McGuinness, D. L. (2002). Ontologies Come of Age. In D. Fensel & J. Heldler & H. Liberman & W. Wahlster (Eds.), *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. Boston: MIT Press.

Pinto, H. S., & Martins, J. (2001). *A Methodology for Ontology Integration.* Paper presented at the First International Conference on Knowledge Capture, New York.

Robertson, B., & Sribar, V. (2002). *The Adaptive Enterprise: IT Infrastructure Strategies to Manage Change and Enable Growth*. Boston: Addison-Wesley.

Schmidt, D. C. (1996). Using design patterns to guide the development of reusable object-oriented software. *ACM Computing Surveys, 28*(4).

Schmidt, D. C., & Buschmann, F. (2003, May 2003). *Patterns, Frameworks, and Middleware: Their Synergistic Relationships.* Paper presented at the International Conference on Software Engineering, Portland, Oregon.

Sprott, D. (2000). Componentizing the Enterprise Application Packages. *Communications of the ACM, 43*(4), 63-70.

Sumner, M. (2000). Risk factors in enterprise-wide/ERP projects. *Journal of Information Technology, 15*(4), 317-327.

van der Vet, P. E., & Mars, N., J. I. (1998). Bottom-up Construction of Ontologies. *IEEE Transactions on Knowledge and Data Engineering, 10*(4), 513-526.

Wang, X., Chan, C. W., & Hamilton, H. J. (2002, July 2002). *Design of Knowledge-Based Systems with the Ontology-Domain-System Approach.* Paper presented at the International Conference on Software Engineering and Knowledge Engineering, Sant'Angelo d'Ischia, Italy.

Weigand, H., De Moor, A., & Heuvel, W. J. (2000). Supporting the Evolution of Workflow Patterns for Virtual Communities. *Electronic Markets, 10*(4).

Willcocks, L. P., & Sykes, R. (2000). The Role of the CIO and IT Function in ERP. *Communications of the ACM, 43*(4), 32-40.

Winograd, T., & Flores, F. (1986). *Understanding Computers and Cognition*. Reading: Addison Wesley.

Authors' biographies

Kay Nelson is an associate professor of MIS and director of The Center for Information Technologies in Management at the Fisher College of Business, The Ohio State University. She holds a Ph.D. from the University of Texas at Austin in Information Systems. Dr. Nelson has published articles about IT strategy issues, software engineering, and IT/Business partnership in publications such as *MIS Quarterly*, *European Journal of Information Systems*, and *Decision Support Systems*. Her research awards include the ICIS best paper award and

the WITS best paper award. Dr. Nelson was recently named a National Science Foundation Career Scholar.

Jim Nelson is an assistant professor of MIS at The Ohio State University. He received his BS in Computer Science from California Polytechnic State University, San Luis Obispo, and his MS and PhD in Information Systems from the University of Colorado, Boulder. His research interests include developing theoretically grounded models and metrics for evaluating business processes, investigating the problems people have shifting to emerging technologies, and determining the business value of IT. Jim generally teaches the more technical courses in information systems including object oriented technology, systems analysis and design, database theory and practice, and telecom.