Research Frontiers in Object Technology

Salvatore T. March

Charles A. Wood

Gove N. Allen

Information and Decision Science Department

Carlson School of Management

312 19th Avenue South

Minneapolis, MN 55455

{smarch, cwood, gallen } @csom.umn.edu

Abstract

Object technology has been widely acclaimed as offering a revolution in computing that will resolve a myriad of problems inherent in developing and managing organizational information processing capabilities. Although its foundations arose in computer programming languages, object technology has implications for a wide range of business computing activities including: Programming, Analysis and Design, Information Management, and Information Sharing. We examine six fundamental research frontiers in each activity: Common Business Classes, Organizational Barriers Applications and Tools, Management of Classes, Instances and Reuse, Standards, Testing, and Metrics, and Technology Investment. The cross product of the business computing activities with these fundamental research frontiers yields a taxonomy within which to position the research needed to realize the promises offered by object technology.

Introduction

The move to object technology represents a significant paradigm shift in the development and management of organizational information processing capabilities. In one example, the Object management Group (OMG) envisions a world in which objects in various organizational computing systems interact seamlessly with each other despite differences in hardware platforms, operating systems, database technologies and programming languages. These objects would provide organizational information processing capabilities that could be bought and sold, individually or as part of a "component design." Furthermore they would be "plug-compatible" with each other enabling seamless and effortless system upgrades. They would be re-usable and thereby eliminate redundancy and the classic effort, quality and maintenance problems associated with it. Object technology, it seems, is not only the elusive "silver bullet" sought by Fred Brooks (1987) to develop information systems, but also the cure for the organization and management woes faced by system developers over the past 30 years.

While we may be overstating the case, it is clear that there is significant hype surrounding "object technology" and precious little evidence of delivery on its promises. In this paper we present a taxonomy of business activities related to the delivery and management of information processing capabilities and the research frontiers in object technology that address them. In concept, object technology has tremendous potential for addressing many of the problems that have plagued software developers and IS managers. However, there are many technical and organizational issues that must be addressed before this potential can be realized.

An object is a software construct that represents something in real world. A class is the structural definition of an object. Classes are instantiated to create objects. These instances have identity, capabilities and attributes. Objects respond to messages corresponding to their capabilities and, when necessary, send messages to other objects.

The organizational structure defined for objects is based on commonality of capabilities, similar to the organizational structure of modern businesses. Marketing personnel are organized into the marketing department because they all perform similar marketing functions. When a marketing person needs information about finance, they ask a finance person. When they need information about operations, they ask an operations person. The marketing person does not know *how* the finance or operations person is able to respond, they only know that they *can* respond. Similarly, an object does not know how other objects respond to its messages, only what messages it can send to which objects. Thus object implement the basic principle of *encapsulation* that enables objects to change how they perform tasks without effecting any other objects, provided the interface remains the same.

As with human organizations, there may be differences among the capabilities of different objects within the same broad organization. These are defined as specializations in the class structure. From an object technology perspective, such a class structure enables "*inheritance*. Objects in a class inherit all capabilities from their superclass(es). Thus a class structure can be developed such that capabilities shared by multiple classes can be assigned to a superclass and inherited from the superclass rather than being developed in each subclass.

The main benefit of object technology is extensibility. When a class is created, that class becomes a part of the information processing capability of the organization. It is not differentiated from classes whose objects perform more system-oriented functions such as opening windows, recognizing user interactions on those windows, accessing files, or sending electronic mail messages. The classes in the class structure define the language of implementation and execution. This changes the role of developers from that of writing programs for specific applications to extending the programming language of the organization so that any application can reuse the capabilities created.

We identify four broad categories of business computing activities necessary for the development and management of information processing capabilities: Programming, Analysis and Design, Information Management, and Information Sharing. Programming is the activity of constructing a software artifact in a computer programming language. Analysis is the process of identifying an appropriate set of capabilities sufficient to meet the needs of a particular domain. Design is the process of organizing those capabilities into an appropriate class structure that can also be utilized by other domains. Information Management is the organizational activity of planning, evaluating and overseeing the development of information processing capabilities, including the selection of applications to be developed and the institutionalization of tools and technologies with which to develop them. Information Sharing constitutes those activities necessary to enable organizations to effectively work together to accomplish common goals. In this context, the term, organization, is used in a very broad sense to include both organizational units in the same corporate entity as well as entirely different corporate entities. Such units may be separated geographically or organizationally, requiring communication and cooperation among their information processing capabilities to achieve mutual goals.

Each of these activities has specific challenges for which object technology, if applied properly, can provide effective solutions. For example, object technology offers programming languages that enable effective program construction by supporting these basic concepts: class, object, encapsulation, inheritance/ extensibility, and message passing. If applied properly, these can significantly enhance programmer productivity and software quality. However, they also raise the challenges of how to apply them properly. They represent a fundamental shift from the application-oriented programming paradigm common to conventional programming environments.

Similarly, object analysis and design techniques extend these basic concepts into the earlier stages of system development. Neither code reuse nor sharing of components occurs naturally within an organization. These require organizational and management structures that are in direct conflict with application-oriented or database-oriented structures common in many IS organizations.

Numerous challenges beset an organization attempting to take advantage of Object technology. These include, among a myriad of other issues, how to organize methods within a class to minimize redundancy, how to find and effectively use existing classes and components, how to acquire the necessary developer and management support, how to estimate programming effort, how to evaluate the resultant artifacts, how to manage re-use, how to program for extensibility, what training and consultation mechanisms to put in place to enable this type of development, and what standards to use.

We explore six research frontiers in object technology: Common Business Classes, Organizational Barriers Applications and Tools, Management of Classes, Reuse and Object Management, Standards, Testing, and Metrics, and Technology Investment. Common Business Classes provide information-processing capabilities needed in a variety of organizations. They are organized into components such that the classes within a component operate together to provide a coherent set of capabilities for specific functional areas such as sales order processing, procurement, finance, or human resources. They can be shared across organizations and effectively define a standard vocabulary for implementing applications using these capabilities. Furthermore, as an object technology, these classes are extensible to enable significant customization within an organization without sacrificing the basic functions that make them sharable. Basic research in this frontier must address the feasibility of developing, managing and sharing such classes and components.

The second frontier, Organizational Barriers, represents the organizational change issues inherent to object technology. While such issues are certainly not new to object technology, given the wide ranging paradigm shifts in development and IS management practices, they are particularly significant in this context. The successful transition from conventional to object technology requires a fundamental change in the IS organization, from a focus on application development and data administration functions to a focus on object management and management of reuse. Existing organizational structures must be migrated to this new paradigm, while sustaining the management of exiting legacy systems. Basic research in this frontier must address the specific issues in this transition, including the tendency to avoid new technologies that cause existing expertise to become obsolete (March 1991).

The third frontier, Applications and Tools, represents the fundamental power underlying object development. A significant amount of research has focused on developing programming languages, systems analysis and design methodologies and development environments. A number of commercial products exist; however, little work has been done to evaluate or compare such applications and tools either with each other or with conventional applications and tools. Almost no theoretical work has been done to demonstrate or enable an understanding of the advantages of object technology over conventional development technologies. Furthermore, many object tools are in their infancy, fraught with awkward interfaces and significant instability. Research in this frontier must focus on understanding why object technology offers advantages in information system development and leveraging this understanding into new and more powerful tools.

The fourth frontier, Reuse and Object Management, is fundamental to the large scale success of object technology. Classes must be managed so they can be reused. A class repository serves as the storage medium for class definitions, making them available to developers as needed. It is not enough, however, simply to support searching for existing classes to reuse. Classes must evolve in response to changes in the organization. This requires significant coordination between object managers and developers. Developers must not be allowed to implement functionality outside of the class in which it logically belongs. Furthermore, while the object technology concept of encapsulation is an effective means to minimize the impacts of such changes it by no means guarantees complete insulation. Finally, the management of instances is one of the major unresolved issues in object technology. Conceptually, an instance represents one "thing" in the world and that "thing" in the world corresponds to exactly one instance in the information system. The instance must be persistent, that is, have existence independent of a program using it. Thus it must be identified and there must be mechanisms through which it can be located and messages sent to it from any application, independent of platforms, operating systems and implementation languages. Research in this frontier must focus on tools and techniques for repository management and well as mechanisms to identify and locate specific instances. Object Oriented Database Management Systems (OODBMS) have the latter as their focus, however, they typically address object identification only within the database, leaving a significant disconnect between the OODBMS and the programming language.

The fifth frontier, Standards, Testing, and Metrics, addresses the development, evaluation and utilization of standards, testing procedures, and metrics both within and among organizations. Standards provide a base of commonality without which it is impossible to achieve the goals of Object technology described above. Significant areas requiring standardization range from programming languages such as Java and C++ to common business class definitions such as those suggested by IBM's San Francisco project (Arnold, Bosch et al. 1997) to representations formalisms such as the Unified Modeling Language (UML) [Booch, 1998] to mechanisms for accessing remote objects such as OMG's Common Object Request Broker Architecture (CORBA) (Maffeis and Schmidt 1997) and Microsoft's Distributed Common Object Model (DCOM) (Wood 1998). Metrics are especially needed to measure complexity, to evaluate object representations including common business classes, and to produce effort estimations. Testing is the process of controlling quality. Research in this frontier must focus not only on the development of standards, testing procedures, and metrics, but also on their value and utility in developing and managing organizational and inter-organizational information processing capabilities.

The final frontier, Technology Investment, addresses an issue that is crucial to the success of any new technology, but especially for one as far reaching as object technology. To be successful, both developers and managers must understand the nature of the basic object technology concepts. Not only must tools be purchased and developers trained in their use, but also mechanisms must be put in place to manage the overall transition to object thinking. Understanding the nature of the investment in object technology cuts to the heart of managing this new technology. Research in this frontier must focus on what tools, methods, technologies and management structures in which to invest. Research must also focus on evaluating the effects of levels and types of investments and on managing joint inter-organizational investments. Fundamental economic issues such as first mover costs and benefits, externalities and opportunity costs must also be examined.

This paper examines current research and illuminates needed research in each of these frontiers as it effects the four activities discussed above.

Previous Calls for Research of Object-Oriented Development

Other authors have made calls for more object-oriented research. Hsu, et al. (Hsu, Gerhardt et al. 1994) develops a research program called Adaptive Integrated Manufacturing Enterprises (AIME). Research in AIME concentrates on exploring how object-oriented techniques can be applied to four major problems. These problems are management distributed systems, architecture development that integrates legacy systems with new systems, manage heterogeneous views in production systems, and production requirements modeling. This work concentrates on the object-oriented programming and design efforts as applied to manufacturing and production. The topics brought up in Hsu's work could be applied and expanded to include other areas of OO development and to include a more general research approach to OO development that could be applied to all business areas rather than concentrating on production.

Guerraoui's (Guerraoui 1996) research provides a history of OO development, and groups OO research into technologies integration, software components, and distributed programming. Guerraoui's work helps researchers discover the application development aspects of OO. However, Guerraoui limits himself to discussion of OO as a development technology. Thus, topics such as organizational barriers, training, repositories and object oriented database management systems (OODBMSes) are not discussed. This paper expands Guerraoui's work to include the management aspects of object-oriented development.

Bjornestad (1994) attempts to define a universal object-oriented research program. Bjornestad claims that object-oriented theories need to be viewed within a comprehensive theoretical framework. This would enhance a researcher's ability to compare object-oriented theories.

Our call for object-oriented research differs from the others in three important ways. First, it is based on a panel discussion of eight accomplished scholars at the 18th International Conference on Information Systems (ICIS '97), a review of which is forthcoming in the Communications of the ACM (Chandra, et al.). This panel discussed the need for the object-oriented research frontiers discussed in this paper. The research frontiers described by that committee and discussed in this article are more expansive and complete than other calls for object-oriented research thus far. Second, this article is the first to review a substantial portion of existing OO literature. Finally, this article is the first article to show the trends of object-oriented research as it has developed. The analysis of the trends shown in this paper is important because they allow researchers to see how research in different object-oriented areas is progressing.

Methodology

We began this review of the existing research on object orientation by identifying, through group consensus, the major journals which publish (or should publish) research regarding object oriented topics. We then searched online periodical indices (Infotrac and ABI Informs) to identify articles that had "object orientation," "object-orientation," "object oriented," or "object-oriented" as keywords in the past five years. One periodical that is noticeably absent from our survey is the Journal of Object-Oriented Programming (JOOP—SIGS Publications). SIGS does not make abstracts for JOOP available in any electronic format (SIGS 1998) and is thus not included in this study. This search identified 467 articles in the following journals:

Journal	Count
Communications of the ACM	89
IEEE Software	60
IEEE Transactions on Knowledge and Data Engineering	52
IEEE Transactions on Software Engineering	30
ACM Computing Surveys	24
IBM Systems Journal	19
IEEE Communications Magazine	18
Decision Support Systems	15
IEEE Transactions on Systems	14
ACM Transactions on Programming Languages & Systems	12
IEEE Transactions on Magnetics	12
IEEE Transactions on Power Systems	11
IEEE Expert	9
IIE Transactions	9
Information Systems Management	9
ACM Transactions on Database Systems	7
ACM Transactions on Information Systems	7
IEEE Transactions on Computers	6
Journal of Database Management	6

IEEE Spectrum	5
Information & Management	5
IEEE Transactions on Circuits and Systems for Video Technology	4
IEEE Transactions on Semiconductor Manufacturing	4
ACM Transactions on Mathematical Software	3
IEEE Network	3
IEEE Transactions on Parallel and Distributed Systems	3
International Journal of Information Management	3
Journal of Management Information Systems	3
Journal of the Association for Computing Machinery	3
European Journal of Information Systems	2
IEEE Transactions on Professional Communication	2
IEEE Transactions on Robotics and Automation	2
Communications of the ACM	1
European Management Journal	1
IEEE Intelligent Systems	1
IEEE Transactions on Biomedical Engineering	1
IEEE Transactions on Broadcasting	1
IEEE Transactions on Energy Conversion	1
IEEE Transactions on Engineering Management	1
IEEE Transactions on Image Processing	1
IEEE Transactions on Pattern Analysis and Machine Intelligence	1
IEEE Transactions on Power Delivery	1
IEEE Transactions on Reliability	1
IIE Solutions	1
Information Systems Research	1
Management Science	1
Omega	1
Sloan Management Review	1
Total	**467**

It is interesting to note that some of the most desirable outlets for research in the IS discipline included only one article or no articles that matched our search criteria. These outlets include Information Systems Research, Management Information Systems Quarterly, and Management Science.

Once the articles were identified, each abstract was reviewed and classified in a category that was defined as the intersection between a business computing activity with a research frontier. We attempted to limit each article to only one category; however, some clearly dealt with multiple categories and therefore were assigned to multiple categories. Of the 467 articles identified by the search, 134 of them did not represent research on object orientation. These 134 articles included four calls for research and ranged from book reviews to research that only addressed object orientation in passing (including one obituary). These articles were removed from consideration in the taxonomy. The following matrix is the result of the juxtaposition of the six research frontiers with the four business computing activities. Each cell on the table represents a research category. The number in each cell refers to the number of articles that addressed the cell's issues.

	Common Business Classes	Organizational Barriers	Applications and Tools	Class, Instance and Reuse Management	Standards, Testing, and Metrics	Technological Investment
Programming Activity	**4**	**6**	**63**	**51**	**9**	**3**
Analysis & Design Activity	**2**	**4**	**72**	**31**	**9**	**4**
Information Management Activity	**2**	**5**	**19**	**17**	**5**	**13**
Inter-organizational Activity	**3**	**2**	**12**	**5**	**10**	**0**

Two research assistants who were Ph.D. students in the Information and Decision Science department at the University of Minnesota were used to classify the articles into each cell of the grid. These assistants had combined experience of 17 years as professional systems developers. These assistants worked together to classify the first 100 articles so that the definitions of various terms used in the abstracts could be clarified before the bulk of the articles were classified. The remaining 367 articles were divided between the two assistants and the rest of the classification was completed independently. Fifty articles were then selected for cross-validation. Of those articles, the two assistants agreed on 47, giving an estimated interrater reliability of 94 percent.

Common Business Classes

Programming Activity	(Bohrer, Johnson et al. 1998); (Comerford 1995); (Gray, Hotchkiss et al. 1998); (Kozaczynski and Booch 1998)
Analysis & Design Activity	(Cho and Zeigler 1997); (Sutherland 1995);
Information Management Activity	(Arnold, Bosch et al. 1997); (Maring 1996)
Inter-organizational Activity	(Arnold, Bosch et al. 1997); (Elliot, Diedrichsen et al. 1996); (Haggerty and Seetharaman 1998)

It is estimated that 80% of the development effort is dedicated to developing noncompetitive functionality that is essentially the same across most organizations (Arnold, Bosch et al. 1997). Business classes that are developed to support functionality that is common to many organizations can give the organizations the competitive parity or competitive advantage that allows them to compete successfully in an increasingly aggressive marketplace.

There are two main areas that need research with regard to how programming is affected by the advent of standard business components. First, when standard business classes are used in a development effort, the impact on the rest of the development process is uncertain. For example, "What impact will future releases of standard business classes have on current development efforts?" or "When and how should common business classes be extended for customization?" Second, research on this frontier should include studies to improve our understanding of how standard components should actually be implemented in different object-oriented languages. The articles in this area have done little more than suggest that standard business components should be used in the programming activity(Kozaczynski and Booch 1998).

At the analysis and design level, the research regarding common business classes should be interested in identifying what classes should be standardized and made available to developers. Likewise, once classes are identified as candidates for standardization, identifying the functionality that should be implemented in each class is worthy of research. Moreover, the interaction between the domain classes and the interface classes needs to be specified. For example, "Should interface classes request values from domain classes and then populate their own components, or is it better to pass a reference of the interface class' component to the domain class and allow the domain class to populate the interface?" Additional research should seek to determine how the availability of common business components should change the process of object-oriented analysis and design. The existing research has addressed none of these questions and has been limited to either reporting on the design of a particular class intended for large-scale reuse (Cho and Zeigler 1997) or examining the costs associated with developing all systems from the ground up instead of using common business components (Sutherland 1995).

In the area of information management, the primary need for research lies in understanding what is necessary to move the systems of an organization from an existing state of incongruous software systems to attain the conjunctive benefits promised by using common business classes. Further, understanding the real costs and benefits of building applications using commonly available components should receive real consideration. Research in this area should seek to increase our understanding of the affect common business classes will have on the ultimate goal of providing the right information to the right individuals at the right time.

There are only a few articles that discuss some of these issues. These articles focus on some of the benefits available by the use of common business components(Maring 1996). The research in this area should continue to identify compelling reasons for businesses to consider using common business components for the development of their systems.

The need for common business classes is fundamental to successful interorganizational coordination. If compatible common business classes are implemented then cooperating organizations could share not only information, but functionality as well. More importantly, if organizations attempt to work together to achieve some common goal, their efforts should be enhanced—not encumbered—by their information systems. Thus, research in this area needs to enlighten our understanding of functionality that should be included in common business classes to advance interorganizational cooperation, as well as to describe the results of attempts at interorganizational cooperation with and without the use of common business classes. So far, the research presented is limited to examples of how building a class structure that is commonly used by to many organizations in cooperation can be extremely beneficial (Elliot, Diedrichsen et al. 1996).

Organizational Barriers

Programming Activity	(Deubler and Koestler 1994); (Dumas and Parsons 1995); (Rabin 1995); (Rosson and Carroll 1996); (Sheetz, Tegarden et al. 1994); (Wilde, Matthews et al. 1993)
Analysis & Design Activity	(Flint 1997); (Parsons and Wand 1997); (Rajkumar and Dawley 1996); (Sutherland 1995)
Information Management Activity	(Bhattacherjee and Gerlach 1998); (Fayad, Tsai et al. 1996); (Fichman and Kemerer 1993); (Kozaczynski and Kuntzmann-Combelles 1993); (Pittman 1993)
Inter-organizational Activity	(Kochikar 1998); (Kunieda, Sugimoto et al. 1993)

Central to the problem of overcoming organizational barriers to adopting object orientation is the need to move an experienced base of developers from an existing procedural or functional environment to a new and unfamiliar environment. Research in this area should seek to help us understand what factors can help to gain support for the move to object orientation. The research identified in this study covers several areas—but none in detail. Articles have identified that some object-oriented programming languages are more easily implemented if certain procedural languages were previously in use, given examples of merging object and legacy technology, presented approaches to migrating a team of programmers to object orientation, identified some of the problems to be expected in the migration, presented techniques for identifying problems, and put forth warnings about making the transition (Comerford 1995). Only one article (Dumas and Parsons 1995) provides any empirical evidence for understanding any issues that may arise overcoming organizational barriers to object-oriented programming. Also addressed are issues of how to design systems so that they can be merged with existing legacy systems (Flint 1997).

Research should continue to examine these areas. Specifically, now that many ideas have been expressed regarding the transition to object-oriented programming, these ideas should be validated by empirical means. Other questions should also be explored, for example, what hardware- and operating-system-level developments make the transition easier? Under what conditions should developers with a strong domain understanding but low object-orientation skills be replaced with developers competent in a new methodology but relatively domain ignorant?

With regard to the management of information, research should be conducted to enlighten our understanding of the proper way to manage and control organizational information resources. For example, currently applications often serve as the locus of control. This can lead to different applications (even within the same department) being unable to communicate effectively even when both are built using strong object-orientation because each application may necessitate a different view of a particular class (the customer class, for example). Research in this area should provide insight into understanding the ramifications of shifting the locus of control over the information from the application to a “Class Manager” with organization-wide responsibility of reconciling and maintaining the divergent views of a particular class. Further research should examine the impacts of using this model as opposed to relying on a database administrator for this kind of management.

So far, articles in this area have focused on understanding the hesitancy that organizations experience in moving to object orientation (Bhattacherjee and Gerlach 1998), suggesting that for successful migration to object orientation, strong management practices must be in place (Fayad, Tsai et al. 1996), and identifying specific barriers and necessary conditions for the successful migration to object orientation (Kozaczynski and Kuntzmann-Combelles 1993). Empirical work has not been done to confirm any of the assertions made in these articles, nor has any work been done to indicate the viability of restructuring the management function around classes instead of applications. This represents a substantial area for future research

When addressing organizational barriers to interorganizational cooperation, research should examine the factors allow individuals to trust the classes developed in another organization. For the encapsulation characteristic of the object-oriented approach to reach its potential, developers must be able to trust that instances of classes will perform as described without an in-depth examination of the class code. Yet, when an organization has a vested interest in the successful outcome of a joint venture, relying blindly on error-filled classes can be costly. Research in this area should be primarily concerned with understanding how to gain assurance of classes developed another organization that must be used for the success of a venture. Additionally, research in this area should look at organizational barriers that have arisen because of alliances with various vendors of hardware and software.

Of the articles reviewed in this study, none address the issue of trust by one organization of another organization’s classes. However, articles that address organizational trust may shed light in this area even though these studies did not deal directly with classes and objects. The role of the Internet and of systems specifically designed to bridge platform differences is discussed lightly by the research (Kochikar 1998); however, no empirical research has been attempted specifically in this area.

Applications and Tools

Programming Activity	(Abadi and Cardelli 1996) ; (Achauer 1993); (Attali, Caromel et al. 1996); (Berman, Larson et al. 1997); (Bommel and Weddell 1994) ; (Bruaset and Langtangen 1997); (Bruno and Agarwal 1997); (Buneman and Ohori 1996); (Campbell, Islam et al. 1993); (Cardenas, Ion et al. 1993); (Caromel 1993) ; (Chang and Gehringer 1996); (Chapman and Bahill 1993); (Chen and Suda 1997); (Chen 1997); (Chikayama 1993); (Cho and Park 1998); (Cockburn 1993); (Cools, Laurie et al. 1997); (Corkill 1997); (Davis and Morgan 1993); (Ehlers and Rensburg 1996); (Eustache, Meunier et al. 1996); (Francois, Escande et al. 1994); (Gagliardi and Spera 1997); (Guenther and Wackerbarth 1993); (Gupta and Fuchs 1993); (Hakavik and Holen 1994); (Hill, Brinck et al. 1993); (Holzle and Ungar 1996); (Jacky 1995); (Jean, reoli et al. 1996); (Karacal and Mize 1998); (Kifer, Lausen et al. 1995); (Kishimoto, Kotaka et al. 1995); (Kunz, Jin et al. 1996); (Kurumbalapitiya, Ratnajeevan et al. 1993); (Laddaga and Veitch 1997) ; (Lee and Zachary 1995); (Leone, Rullo et al. 1997); (Lieberherr, Silva-Lepe et al. 1994); (Liu, Offutt et al. 1998); (Lu, Swidenbank et al. 1995); (Luckham, Kennedy et al. 1995); (Machiels and Deville 1997); (Maeda and Ikeda 1996); (Magalhaes and Mesquita 1998); (Mohan and Kashyap 1993); (Norton, Szymanski et al. 1995); (Palsberg, Xiao et al. 1995); (Palsberg 1996); (Papazoglou, Delis et al. 1997); (Popescu, Munteanu et al. 1998); (Radin 1996); (Su, Jawadi et al. 1998); (Taivalsaari 1996); (Thomas 1995); (Wiener 1998); (Wood 1993); (Wu, Chan et al. 1994); (Yoo and Sheu 1993); (Zaharioudakis and Carey 1998); (Zeigler, Cho et al. 1996)
Analysis & Design Activity	(Agarwal, Sinha et al. 1996); (Amadio and Cardelli 1993); (Baclawski and Indurkhya 1994); (Baumer, Gryczan et al. 1997); (Beeri and Spiegler 1996); (Billo and Bidanda 1995); (Blaha, Premerlani et al. 1994) ; (Bock and Ryan 1993); (Borgida, Mylopoulos et al. 1995); (Bourdeau and Cheng 1995); (Castagna 1995); (Cockburn 1993); (David 1996); (Dori and Tatcher 1994) ; (Dujardin, Amiel et al. 1998); (Edwards 1997); (Embley, Jackson et al. 1995); (Fayad, Tsai et al. 1994); (Fayad, Hawn et al. 1993) ; (Fotouhi, Ahmad et al. 1994); (Frank, Rupprecht et al. 1997); (Gaing, Lu et al. 1996); (Geihs, Heite et al. 1993); (Gottlob, Schrefl et al. 1996); (Harrison, Kilov et al. 1996); (Hevner and Mills 1993); (Holsing and Yen 1997); (Honiden, Kotaka et al. 1993); (Honiden, Nishimura et al. 1994); (Hotter 1994); (Isakowitz, Schocken et al. 1995); (Janneck and Naedele 1998); (Kemper, Kilger et al. 1994); (Larsen, Fitzgerald et al. 1996); (Lauesen 1998); (Lavazza 1993); (Lea, Jacquemot et al. 1993); (Lenard 1993); (Li and Lochovsky 1998); (Lieberherr 1993); (Low, Henderson-Sellers et al. 1995); (Lu and Dillon 1994); (McConnell 1996); (McDavid 1996); (Mentzas 1997); (Mycroft 1996); (Palsberg and Smith 1996); (Parsons and Wand 1997); (Peckham, Maryanski et al. 1996); (Pei and Cutone 1995); (Pinheiro and Goguen 1996); (Potts 1993); (Poulovassilis and Levene 1994); (Qing and Huang 1995); (Quilici 1994); (Ram, Vivekananda et al. 1997); (Schwabe and Rossi 1995); (Schwabe and Rossi 1995); (Seiter, Palsberg et al. 1998); (Shlaer, Mellor et al. 1994); (Shlaer and Mellor 1997); (Shoval and Frumermann 1994); (Silberschatz, Korth et al. 1996); (Snoek and Dedene 1998); (Song 1997); (Teuhola 1996); (Vaishnavi, Buchanan et al. 1997); (Wang 1994); (Wang 1995); (Wang 1995); (Wang 1996) ; (Yamazaki, Kajihara et al. 1993)
Information Management Activity	(Agarwal, Tanniru et al. 1995); (Bussche, Gucht et al. 1997); (Davis, Grimes et al. 1996); (Handschin, Heine et al. 1998); (Hayne and Pendergast 1995); (Henderson-Sellers 1997); (Jaccheri and Conradi 1993); (Jungthirapanich and Benjamin 1995); (Krause, Byers et al. 1994); (Lee and O'Keefe 1996); (Lefrancois and Montreuil 1994); (Leymann and Altenhuber 1994); (Lieberherr and Xiao 1993); (Love 1995); (Ma 1995); (Ma 1997); (Muhanna 1993); (Pitoura, Bukhres et al. 1995); (Rettig, Simons et al. 1993)
Inter-organizational Activity	(Bocking 1996); (Chen and Suda 1997); (Davies and Davies 1997); (Fekete, Kaashoek et al. 1998); (Maffeis and Schmidt 1997); (Moore and Stanphill 1994); (Price, Trinkle et al. 1993); (Roy, Haridi et al. 1997); (Schmidt, Gokhale et al. 1997); (Shrivastava and McCue 1994) ; (Singh 1998); (Vinoski 1997)

Object-oriented development techniques can simplify application development, improve communcation, and motive personnel (Davis and Morgan 1993). Research into object-oriented applications and tools shows how powerful and potentially cost effective object-oriented development can be. Even though this frontier has experienced more research than any other frontier, studies conducted have all but ignored theory building and empiricism.

When researching the process of programming applications and tools, the primary interest concerns the development of tools and languages for writing object-oriented applications. Appropriate research in this area includes design-science studies which seek to demonstrate the feasibility of a building a particular concept of object-orientation into a programming language. Once the feasibility of a specific implementation has been demonstrated, the next step for research in this area is to evaluate whether significant improvements in efficiency and effectiveness can be realized by its use.

Although a large portion of the articles examined for this study are classified as programming research in the area of applications and tools, many of the articles thus classified are simply examples of solving a problem using an object-oriented language and do not represent research on how programming is affected because object-oriented applications are being developed. Most of the articles dealing with applications or tools and the programming activity were descriptions, comparisons or reviews of object-oriented programming languages. A few articles presented descriptions of algorithms which could be implemented in most object-oriented languages. Some of the articles addressed in this section were simple "how to" tutorials and others offered a historical perspective of the development of different languages or features. Only two articles—(Cockburn 1993), (Davis and Morgan 1993)—addressed how the programming activity is enabled or otherwise affected by the move to object orientation.

Painfully absent from this articles in this survey is any discussion of research questions regarding the affects of different object-oriented programming languages on programmer productivity. For example, "Which language allows programmers to be more effective or efficient: C++, Java, or SmallTalk?", "What characteristics of an application make it more suited to a particular language?", "Do people systematically apply object-oriented programming techniques more correctly if they learn an object-oriented language as their first language than if they have programming experience in a procedural language first?" These are the kinds of research questions that have the potential to make a substantial mark on the object-oriented landscape—the kind of mark that presentation of yet another object-oriented language or development of another object-oriented tool cannot hope to make.

Research on how analysis and design are impacted by various applications and tools includes research on object-oriented CASE tools and the development of new methods for conducting analysis and design. Many articles have been written describing new ways of formalizing the description of a problem space or a solution space. Like research pertaining to the programming activity, research here can take both the form of design science and natural science. Comparative studies that examine the benefits of one approach to analysis and design as compared to another are also appropriate in this area of research.

The research in this study has focused on the following topics: how to design systems using object-oriented analysis and design, comparing modeling approaches (and how to compare them), examples of interesting designs, converting object-oriented designs to other formats for implementation, comparing modeling formalisms, solution to common design problems, discussions of particular design methods, example applications designed using different object-oriented design approaches, discussions of tool design, model management, formal foundations of object-oriented analysis and design, discussions of tools to assist in design and studies examining the impact of object-oriented analysis and design on application development.

In some respect, the work in this area is acceptable because it covers a wide array of topics; however, the research questions that must be answered for object orientation to have its potential impact in the area of tools and applications for analysis and design remain unanswered or only lightly addressed. Although some research has been done to determine the impact of object-oriented analysis and design on the application development process—(Cockburn 1993), (Dujardin, Amiel et al. 1998), (Pei and Cutone 1995),and (Larsen, Fitzgerald et al. 1996)—there are still many questions that should be addressed. The continued development of new design approaches and methodologies is of questionable value until we have the discipline to determine relative benefits of the proposed approaches over the existing ones.

To understand how the activity of information management is affected by object-oriented applications and tools, research efforts should focus on how planning, evaluating and overseeing the development of information processing capabilities are changed when object-oriented methods are used or mixed with traditional methods of application development. Specifically, should the traditional notion of application managers (individuals overseeing the development of an application) be replaced with class managers (individuals responsible for the definition and functionality of individual classes regardless of which applications use instances of the class)? Further, research in this area should seek to identify how object-oriented applications might help the management of information resources in a particular domain.

Some articles examined in this area have delved into new object-oriented methods for managing model development, how object-oriented languages affect the management of heterogeneous database management systems, various object-oriented tools for managing information resources, processes for managing the evolution of object-oriented software, and techniques for managing people developing object-oriented software. Others made arguments for the need to build decisions capabilities into various classes, presented implementations of object-oriented knowledge-based decision support systems, or compared the management of object-based systems to rule-based systems.

While some of the research in this area has addressed a portion of the important questions in the area of information management, there has been very little research examining how management of information resources is different when the object approach is implemented. We were unable to identify any research that examined whether management is easier or harder, more expensive or less expensive, more efficient or less efficient, or more effective or less effective because object-oriented tools and applications are being managed.

Research on object-oriented applications and interorganizational activities is primarily concerned with understanding how applications can be built to foster cooperation among organizations. This kind of research should focus on the implementation of object communication functionality to increase the benefits that organizations get from working together. Moreover, research in this area should also be concerned with determining the impact that applications using object technology has had on joint ventures and with comparing object-oriented joint venture systems with their procedural counterparts.

In the term of our study, research in this area has primarily been concerned with building applications to facilitate communication among objects that exist on different platforms, that are created using different languages, that are in contact with each other only intermittently, and that achieve communication using various communication protocols. Some work has been done in developing stable modules in distributed environments as well as implementing communication between object and legacy systems. Some examples of using object technology to facilitate inter-organizational cooperation are also examined.

Some of the main research questions that remain unanswered in this area are "Can object oriented applications provide advantages in inter-organizational communication?", "To what extent can cross-platform object communications utilities facilitate joint ventures?", "Can tools be built to allow companies using vastly different computing architectures and platforms to cooperate effectively?", "What are the fundamental characteristics of joint venture systems and is object technology fundamental to their success?"

Reuse and Object Management

Programming Activity	(Alfred 1994) ; (Arjomandi, O Farrell et al. 1995) ; (Arnold and Fuson 1994); (Bellinzona, Fugini et al. 1995); (Bertino, Ferrari et al. 1998); (Bouguettaya 1996); (Budinsky, Finnie et al. 1996); (Buhr 1995); (Chen, Hull et al. 1995); (Chien, Xue et al. 1997); (Cockburn 1993); (Cooper 1998); (Czejdo, Eick et al. 1993); (Deng and Fuhr 1995); (Diaz and Paton 1994); (DiPippo and Wolfe 1997); (Fabre and Perennou 1998); (Henrotte, Meys et al. 1996); (Huang and Lin 1996); (Ishikawa, Kozakura et al. 1993); (Ishikawa, Yamane et al. 1996); (Islam and Campbell 1996); (Johnson 1995); (Johnson 1997); (Karaorman and Bruno 1993); (Kesim and Sergot 1996); (King and Novak 1993); (Laddaga and Veitch 1997); (Lee and Wiederhold 1994); (Lee and Lee 1998); (Lieberherr and Xiao 1993); (Lohr 1993); (Oomoto and Tanaka 1993); (Papzoglou 1995); (Richardson, Carey et al. 1993); (Robertson 1997); (Samarati, Bertino et al. 1997); (Schauble and Wuthrich 1994); (Schmidt 1995); (Schnase, Leggett et al. 1993); (Silva, Mesquita et al. 1994); (Silva and Mesquita 1996); (Singhal and Nguyen 1998); (Srinivasan and Chang 1997); (Staringer 1994); (Su, Hyun et al. 1998); (Thakore, Su et al. 1995); (Wolfe, Lau et al. 1994); (Wong and Wilson 1993); (Wong and Tojo 1996); (Zhou, Rundensteiner et al. 1997)
Analysis & Design Activity	(Artale, Cesarini et al. 1996); (Beneventano, Bergamaschi et al. 1998); (Bertino and Foscoli 1997); (Cockburn 1993); (Coplien 1997); (Demeyer, Meijer et al. 1997); (Fayad and Cline 1996); (Gabel 1994); (Griffin 1995); (Gronbaek and Trigg 1994); (Gyssens, Paredaens et al. 1994); (Huh and Chung 1995); (Islam and Devarakonda 1996); (Kim and Subbaraman 1997); (Kwon and Park 1996); (Liu and Meersman 1996); (Liu, Zicari et al. 1997); (Manola 1995); (Mellor and Johnson 1997); (Meyer 1993); (Mili, Mili et al. 1995); (Monroe, Kompanek et al. 1997); (Ng, Butler et al. 1995); (Peters and Ozsu 1997); (Poston 1994) ; (Purao, Jain et al. 1998); (Rajlich and Silva 1996); (Rundensteiner, Bic et al. 1994); (Sanchez and Choobineh 1997); (Su, Guo et al. 1993); (Aarsten, Brugali et al. 1996)
Information Management Activity	(Bancilhon 1996); (Basili, Briand et al. 1996); (Bist 1996); (Bordoloi and Lee 1994); (Cook, Wolf et al. 1998); (Driesen, Fransen et al. 1998); (Fernandez, Gudes et al. 1994); (Gillenson and Frost 1993); (Hyman 1993); (Lee and Liou 1996); (Majetic and Leiss 1997); (Olivier and Solms 1994); (Qing and McLeod 1994); (Schmidt and Fayad 1997); (Thomas and Sandhu 1996); (Watanabe 1997); (Wolf 1994)
Inter-organizational Activity	(Cobb, Iii et al. 1998); (Gogac, Dengi et al. 1998); (Haase 1996); (Muhlhauser, Gerteis et al. 1993); (Shvartsman 1993)

Traditional, pre-object-oriented development techniques had a huge trade-off between customization and standard software products. Managers were reluctant to change code inside a pre-packaged software product since those changes would need to be incorporated into every new release of the product. In addition, developers often started a new program by copying a similar program and making modifications. This lead to huge maintenance costs as a single change request could often affect several programs.

One of the premier reasons for the acceptance of object-oriented development is extensibility. Object-oriented development techniques allow developers to take existing objects and extend them through inheritance without changing the original code. This makes customizable pre-packaged software more feasible. Programmers can now reuse other programmer's code through inheritance rather than through making copies similar code, so that change requests affect only one ancestor class, therefore reducing development and maintenance costs. However, as programmers start to use one object to contain other objects, inherit from ancestor objects, or instantiate objects within class methods, the need for strong encapsulation discipline is vital to protect instances of the class from unwanted interaction from other classes. In addition, the need for new business structures such as an object-oriented database administrator and a class repository manager who manages objects and reuse is imperative with any object-oriented effort.

Reuse and object management consists of managing class definitions and instances of classes throughout a development process. Classes are managed using a class repository, where all the classes are stored and wait to be reused by developers. Instances of classes are managed using an object-oriented database management system (OODBMS) or an interface to a relational database via an object translation class. Additions to the body of knowledge in this area could concentrate on development techniques for developing reusable code, design methodologies that incorporate repository and OODBMS use, managing repositories and OODBMSes, repository development, repository use, and implications of interorganizational repositories for joint object-oriented development.

Frameworks and patterns are a large part of reuse. Frameworks are collections of classes or class libraries that simplify the programming effort by allowing developers to use inheritance to customize their applications. Frameworks include Java's AWT (Advanced Windowing Toolkit), and the MFC (Microsoft Foundation Class) library. Research in this category has demonstrated different ways to design and reuse classes inside frameworks, and to identify common patterns that exist across classes (Coad and Yourdin 1991), and to store reusable classes via frameworks or repositories (Bellinzona, Fugini et al. 1995; Schmidt 1995).

Much research in this category concentrates on development efforts using an OODBMS. Research has been done on OODBMS creation (modeling, clustering, indexing, etc.), OODBMS concurrency and security issues (Olivier and Solms 1994; Majetic and Leiss 1997), multi-corporation or distributed OODBMSes (Cobb, Iii et al. 1998), using an OODBMS in design or development (Bertino, Ferrari et al. 1998), concurrent OODBMS use (Karaorman and Bruno 1993; Buhr 1995), OODBMS evaluation (Fernandez, Gudes et al. 1994), and tools that help manage distributed OO environments (Shvartsman 1993).

There are several research questions in this frontier that still need to be addressed, including "What is that best way to administer a class repository?", "What management structures or business processes are needed for effective object-oriented reuse?", "What are the organizational or development impacts of multi-organizational control on distributed OO development efforts?", and "What are the organizational or development impacts of reuse of classes across organizations?"

Standards, Testing, and Metrics

Programming Activity	(Binder 1994); (Chambers and Leavens 1995); (Comerford 1995); (Lee and Zachary 1995); (McGregor and Korson 1994); (Murphy, Townsend et al. 1994); (Poo and Chew 1996); (Poston 1994); (Roy and Fang 1996)
Analysis & Design Activity	(Binder 1994); (Chidamber and Kemerer 1994); (Churcher, Shepperd et al. 1995); (Du and Wolfe 1997); (Henning 1998); (Hitz and Montazeri 1996); (Jackson, Giauque et al. 1996); (Jorgensen and Erickson 1994); (Kung, Gao et al. 1995)
Information Management Activity	(Basili, Briand et al. 1996); (Fayad 1997); (Harrison, Counsell et al. 1998); (Nesi 1998); (Chidamber, Darcy et al. 1998)
Inter-organizational Activity	(Dupuy, Nilsson et al. 1995); (Fayad and Schmidt 1997); (Genesereth and Ketchpel 1994); (Hardwick and Bolton 1997); (Klerer 1993); (Meyer 1993); (Moore and Stanphill 1994); (Schmidt 1998); (Siegel 1998); (Vinoski 1998)

When standards are enforced, developers must develop a specific way. This may help developers when reusing unfamiliar classes since all classes would have similar features. Standards are especially important for object development since different programmers—possibly from different departments or even different organizations—need to access the same classes. OMG's CORBA and Microsoft's Distributed Common Object Model (DCOM) are both standards for multi-organizational development.

Metrics are a useful way of developing standards. Metrics allow the measurement of execution time, programming productivity, or code complexity. Metrics are needed to effectively measure a program's or developer's performance so it can be judged against a standard. Testing ties in to metrics and standards by allowing an end user (or end user representative) to ensure the quality of a product by making sure an object functions in a particular environment. Research in testing and metrics includes built-in testing capabilities within modules (Binder 1994; Poston 1994), error tolerance standards (Kim and Subbaraman 1997; Roy and Fang 1997), and development of metrics (Chidamber and Kemerer 1994; Churcher, Shepperd et al. 1995; Hitz and Montazeri 1996).

Recently, the CORBA standard has generated much research in multi-organizational standards use and development (Fayad and Schmidt 1997; Schmidt, Gokhale et al. 1997). Additional research on multi-organizational standards includes developing with communication standards for multiple object-oriented systems or languages within a single organization (Chambers and Leavens 1995).

Research questions that still need to be explored include "How do Microsoft's DCOM and OMG's CORBA standards compare with each other in efficiency, simplicity, or productivity?", "How can an organization or set of organizations best establish enterprise-wide or cross-industry cooperation when developing standards?", "When are standards between organizations appropriate, and how do you determine these standards?", "Which metrics or standards are most efficient and effective for different organizational types when developing object-oriented systems?", "What are the best techniques for motivating developers or analysts to adopt standards during object-oriented development?", "What are the most effective object-oriented testing patterns?", and "What are the quality or time effects of programming standards on programmer productivity?"

Technological Investment

Programming Activity	(Auer 1995); (Cockburn 1993); (Jacobson 1993);
Analysis & Design Activity	(Cockburn 1993); (Goldfedder and Rising 1996); (Muller 1994); (Potok and Vouk 1997)
Information Management Activity	(Adamczyk 1995); (Berg, Cline et al. 1995); (Cockburn 1993); (Fayad and Tsai 1995); (Fayad, Tsai et al. 1996); (Fuerte-Esquivel, Acha et al. 1998); (Hatton 1998); (Henderson-Sellers 1997); (Marchisio, Ronco et al. 1993); (Mierop, Tax et al. 1993); (Pancake 1995); (Yamagishi, Sasaki et al. 1995); (Zhou 1996)
Inter-organizational Activity	

Research regarding technological investment examines the willingness of management to invest in and adopt object-oriented technology. Research about the investment required for effective object-oriented development is key to information management research because object-oriented development requires a significant support and dedication from management—including new tools and substantial training—to be effective. Without management buy-in, object-oriented development is next to impossible (Cockburn 1993). However, relatively little research has been done in this area. Research in technological investment concentrates on the effects of investment on analysis, design, and programming; when, why, and how much to invest; and how to handle cross-company investment in shared object-oriented technology.

Research in this frontier deals with how to improve object-oriented programming through investment, how to decide which programming tool to purchase, and the effects of low or high investment on developer productivity. Current research in this area includes studies about the significant training costs that need to be incurred when starting object-oriented development (Cockburn 1993), the advantages that organizations receive when their programmers are among the first to adopt object-oriented development techniques as object-oriented technology matures (Jacobson 1993), and on the benefits of using tools for specific tasks such as legacy system pattern development (Auer 1995). Appropriate tools that need to be purchased to aid in object-oriented analysis and benefits from investment (Muller 1994); (Potok and Vouk 1997) have also been discussed, as have techniques used for cost estimation of object-oriented development (Henderson-Sellers 1997).

Perhaps the most interesting unanswered question revealed by our survey in the area of technological investment is "How do companies divide investment the costs and benefits of object-oriented development during joint ventures?" The issues raised by joint technological investment are difficult to resolve. Using object technology compounds the problem because joint object-oriented development efforts can result in an extremely valuable and portable software components that are designed for reuse. In addition, since the deliverables are classes instead of entire programs, drastically more planning is required to determine which company expends which effort during development.

More research is needed to answer questions like "How much investment is needed for programmers to become effective object-oriented developers?", "What are the types of object-oriented training that are most effective for different types of language and different types of developers?", "What types of tools should be purchased to aid in object-oriented programming?", "What are the effects of high or low investment on quality and timeliness of object-oriented development efforts?", "What is an effective methodology for determining the amount of object-oriented investment", and "How do you perform break-even analysis of an object-oriented investment?"

Conclusion

Object-oriented development has clearly made great strides over the last five years. Almost every mainstream development tool, such as Java, PowerBuilder, C++, Visual Basic, and even COBOL, now has or claims to have object-oriented capabilities. The object-oriented paradigm, however, is vastly different from traditional software development paradigms.

Far-reaching implications can result from implementing the object-oriented paradigm. Object-oriented development provides for extensible, off-the-shelf software packages that are easily customizable such that customizations need to be made only once and will span different versions of the software, whereas traditional customizable software packages force the developer to reapply any changes from version to version. Effective object-oriented development may require a different IS structure, where employees are arranged by functionality (e.g., user interfaces, database access, etc.) rather than by represented department (e.g., financial systems, marketing systems, sales support, etc.). New positions could be created, such as repository manager or class manager. Object-oriented development could even serve as a bridge between joint ventures as organizations develop object communication standards, such as OMG's CORBA, Java's Remote Methods Interface (RMI), and Microsoft's DCOM, that allow different organizations share software components through a common standard.

Although the object-oriented paradigm has been with us since the late 60s, it is relatively new to the business world gaining acceptance with the advent of C++ in the mid 80s. As such, there are still many vital research areas that need to be addressed. How does an organization (or should an organization) convince its managers, analysts, and developers to start using object-oriented development techniques? What is the best way for companies to share software while protecting their interests at the same time? What tools, packages, or techniques are best for object-oriented development? How much, or how little, should be invested on object-oriented training, tools, and infrastructure? What standards should be adopted, and how does an organization choose between competing standards?

One of the most pressing issues is the area of software reuse. Often, developers complete assignments without much thought about what has been done before in other areas of the organization. Object-oriented development has promised an easier development task since encapsulated modules lend themselves to reuse through inheritance. What are the best practices, organizational designs, or managerial efforts that can lead to effective object-oriented development and reuse, and what are the implications of small module reuse in an organization? What are the cognitive, organizational, and industry implications of reuse? What is the best way to design or program reusable code? How can the knowledge of a developer regarding classes he or she developed best be captured and disseminated throughout an organization? These questions have not been addressed in the academic literature nor in the trade journals. For the object-oriented paradigm to have its full impact, the questions regarding reuse must be explored.

In this article, we have explored six frontiers in object-oriented development. These lack of research in several of these frontiers shows academic researchers have not kept pace with object-oriented practices. More empirical articles are needed to shed light on the difficult questions discussed in this article.

Bibliography

Aarsten, A., D. Brugali, et al. (1996). “Designing concurrent and distributed control systems.” Communications of the ACM 39(10): 50-58.

Abadi, M. and L. Cardelli (1996). “On subtyping and matching.(object-oriented programming).” ACM Transactions on Programming Languages & Systems 18(4): 401.

Achauer, B. (1993). “The DOWL distributed object-oriented language.” Communications of the ACM 36(9): 48-55.

Adamczyk, J. (1995). “Smalltalk reaches crossroads in the insurance industry.” Communications of the ACM 38(10): 107-109.

Agarwal, R., A. P. Sinha, et al. (1996). “Cognitive fit in requirements modeling: A study of object and process methodologies.” Journal of Management Information Systems 13(2): 137-162.

Agarwal, R., M. R. Tanniru, et al. (1995). “Knowledge-based model validation support for end-user computing environments.” Decision Support Systems 15(1): 1-18.

Alfred, C. (1994). “Maximizing leverage from an object database.” IBM Systems Journal 33(2): 280-299.

Amadio, R. M. and L. Cardelli (1993). “Subtyping recursive types. (computer programming).” ACM Transactions on Programming Languages & Systems 15(4): 575.

Arjomandi, E., W. O Farrell, et al. (1995). “ABC++: Concurrency by inheritance in C++.” IBM Systems Journal 34(1): 120-137.

Arnold, T. R. and W. A. Fuson (1994). “Testing 'in a perfect world'.” Communications of the ACM 37(9): 78-79+.

Arnold, V. D., R. J. Bosch, et al. (1997). “IBM business frameworks: San Francisco project technical overview.” IBM Systems Journal 36(3): 437-445.

Artale, A., F. Cesarini, et al. (1996). “Describing database objects in a concept language environment.” IEEE Transactions on Knowledge and Data Engineering 8(2): 345.

Attali, I., D. Caromel, et al. (1996). “A Natural Semantics for Eiffel dynamic binding.” ACM Transactions on Programming Languages & Systems 18(6): 711.

Auer, K. (1995). “Smalltalk training: As innovative as the environment.” Communications of the ACM 38(10): 115-117.

Baclawski, K. and B. Indurkhya (1994). “The notion of inheritance in object-oriented programming.” Communications of the ACM 37(9): 118-119.

Bancilhon, F. (1996). “Object databases.” ACM Computing Surveys 28(1): 137.

Basili, V. R., L. C. Briand, et al. (1996). “A validation of object-oriented design metrics as quality indicators.” IEEE Transactions on Software Engineering 22(10): 751-761.

Basili, V. R., L. C. Briand, et al. (1996). “How reuse influences productivity in object-oriented systems.” Communications of the ACM 39(10): 104-116.

Baumer, D., G. Gryczan, et al. (1997). “Framework development for large systems. (object-oriented software development projects).” Communications of the ACM 40(10): 52.

Beeri, Y. and I. Spiegler (1996). “Synergetic expert systems.” Decision Support Systems 17(2): 73-82.

Bellinzona, R., M. G. Fugini, et al. (1995). “Reusing specifications in OO applications.” IEEE Software 12(2): 65.

Beneventano, D., S. Bergamaschi, et al. (1998). “Consistency checking in complex object database schemata with integrity constraints.” IEEE Transactions on Knowledge and Data Engineering 10(4): 576.

Berg, W., M. Cline, et al. (1995). “Lessons learned from the OS/400 OO project.” Communications of the ACM 38(10): 54-64.

Berman, O., R. C. Larson, et al. (1997). “Scheduling workforce and workflow in a high volume factory.” Management Science 43(2): 158-172.

Bertino, E. and P. Foscoli (1997). “On modeling cost functions for object-oriented databases.” IEEE Transactions on Knowledge and

Data Engineering 9(3): 500.

Bertino, E., E. Ferrari, et al. (1998). “Navigational accesses in a temporal object model.” IEEE Transactions on Knowledge and Data Engineering 10(4): 656.

Bhattacherjee, A. and J. Gerlach (1998). “Understanding and managing OOT adoption. (object oriented technology).” IEEE Software 15(3): 91.

Billo, R. E. and B. Bidanda (1995). “Representing group technology classification and coding techniques with object oriented modeling principles.” IIE Transactions 27(4): 542-554.

Binder, R. V. (1994). “Design for testability in object-oriented systems.” Communications of the ACM 37(9): 87-101.

Binder, R. V. (1994). “Object-oriented software testing.” Communications of the ACM 37(9): 28.

Bist, G. (1996). “Applying the object-oriented model to technical information.” IEEE Transactions on Professional Communication 39(1): 49-57.

Blaha, M., W. Premerlani, et al. (1994). “Converting OO models into RDBMS schema. (object oriented; relational database management system)(includes related article on OMT object-modeling notation).” IEEE Software 11(3): 28.

Bock, D. B. and T. Ryan (1993). “Accuracy in modeling with extended entity relationship and object oriented data models.” Journal of Database Management 4(4): 30-39.

Bocking, S. (1996). “Sockets++: a uniform application programming interface for basic-level communication services.” IEEE Communications Magazine 34(12): 114.

Bohrer, K., V. Johnson, et al. (1998). “Business process components for distributed object applications.” Communications of the ACM 41(6): 43-48.

Bommel, M. F. v. and G. E. Weddell (1994). “Reasoning about equations and functional dependencies on complex objects.” IEEE Transactions on Knowledge and Data Engineering 6(3): 455.

Booch, Grady, Jim Rumbaugh, Ivar Jacobson, and James Rumbaugh, The Unified Modeling Language User Guide, Addison-Wesley Publishing Company, Inc., New York, 1998, ISBN: 0-2015-7168-4.

Bordoloi, B. and M.-H. Lee (1994). “An object-oriented view: Productivity comparison with structured development.” Information Systems Management 11(1): 22-30.

Borgida, A., J. Mylopoulos, et al. (1995). “On the frame problem in procedure specifications.” IEEE Transactions on Software Engineering 21(10): 785-798.

Bouguettaya, A. (1996). “On-line clustering.” IEEE Transactions on Knowledge and Data Engineering 8(2): 333.

Bourdeau, R. H. and B. H. C. Cheng (1995). “A formal semantics for object model diagrams.” IEEE Transactions on Software Engineering 21(10): 799-821.

Brooks, Fred P., Jr., "No Silver Bullet: Essence and Accidents of Software Engineering", IEEE Computer, April 1987, 10-19.

Bruaset, A. M. and H. P. Langtangen (1997). “Object-oriented design of preconditioned iterative methods in Diffpack.” ACM Transactions on Mathematical Software 23(1): 50.

Bruno, G. and R. Agarwal (1997). “Modeling the enterprise engineering environment.” IEEE Transactions on Engineering Management 44(1): 20-30.

Budinsky, F. J., M. A. Finnie, et al. (1996). “Automatic code generation from design patterns.” IBM Systems Journal 35(2): 151-171.

Buhr, P. A. (1995). “Are safe concurrency libraries possible?” Communications of the ACM 38(2): 117-120.

Buneman, P. and A. Ohori (1996). “Polymorphism and type inference in database programming.” ACM Transactions on Database Systems 21(1): 30.

Bussche, J. v. d., D. v. Gucht, et al. (1997). “On the completeness of object-creating database transformation languages.” Journal of the Association for Computing Machinery 44(2): 272.

Campbell, R. H., N. Islam, et al. (1993). “Designing and implementing CHOICES: An object-oriented system in C++.” Communications of the ACM 36(9): 117-126.

Cardenas, A., T. L. Ion, et al. (1993). “The knowledge-based object-oriented PICQUERY + language.” IEEE Transactions on Knowledge and Data Engineering 5(4): 644.

Caromel, D. (1993). “Toward a method of object-oriented concurrent programming. (construction of program illustrating sequential unification of object-oriented techniques using Eiffel program development software) (one of eight articles on concurrent object-oriented programming; special issue) (Technical).” Communications of the ACM 36(9): 90.

Castagna, G. (1995). “Covariance and contravariance: conflict without a cause.” ACM Transactions on Programming Languages & Systems 17(3): 431.

Chambers, C. and G. T. Leavens (1995). “Typechecking and modules for multimethods.” ACM Transactions on Programming Languages & Systems 17(6): 805.

Chandra, Jagdish, Les Gasser, Salvatore March, Satyen N. Mukherjee, Will Pape, R. Ramesh, H. Raghav Rao, and Ray Waddoups, "Information Systems Frontiers: Emerging Vistas – I", Communications of the ACM, forthcoming.

Chang, J. M. and E. F. Gehringer (1996). “A high-performance memory allocator for object-oriented systems.” IEEE Transactions on Computers 45(3): 357.

Chapman, W. L. and A. T. Bahill (1993). “A hypertext software package to help document system designs.” IEEE Transactions on Systems 23(2): 584.

Chen, I. M. A., R. Hull, et al. (1995). “An execution model for limited ambiguity rules and its application to derived data update.” ACM Transactions on Database Systems 20(4): 365.

Chen, J.-Y. J. (1997). “CSPL: An Ada95-like Unix-based process environment.” IEEE Transactions on Software Engineering 23(3): 171-184.

Chen, L. T. and T. Suda (1997). “Designing mobile computing systems using distributed objects.(Distributed Object Computing).” IEEE Communications Magazine 35(2): 62.

Chidamber, S. R. and C. F. Kemerer (1994). “A metrics suite for object oriented design.” IEEE Transactions on Software Engineering 20(6): 476-493.

Chidamber, S. R., D. P. Darcy, et al. (1998). “Managerial use of metrics for object-oriented software: an exploratory analysis.” IEEE Transactions on Software Engineering 24(8): 629.

Chien, S. Y. P., L. Q. Xue, et al. (1997). “Task planning for a mobile robot in an indoor environment using object-oriented domain information.” IEEE Transactions on Systems 27(6): 1007.

Chikayama, T. (1993). “Personal perspectives: Launching the new era.” Communications of the ACM 36(3): 82-90.

Cho, D.-S. and R.-H. Park (1998). “An object-oriented coder using block-based motion vectors and residual image compensation.” IEEE Transactions on Circuits and Systems for Video Technology 8(3): 316.

Cho, T. H. and B. P. Zeigler (1997). “Simulation of intelligent hierarchical flexible manufacturing: batch job routing in operation overlapping.” IEEE Transactions on Systems 27(1): 116.

Churcher, N. I., M. J. Shepperd, et al. (1995). “Comments on 'A Metrics Suite for Object Oriented Design'.” IEEE Transactions on Software Engineering 21(3): 263-265.

Coad, P. and Yourdon, E., Object-Oriented Design, Prentice Hall, Englewood Cliffs, N.J., 1991.

Cobb, M. A., H. F. Iii, et al. (1998). “An OO database migrates to the Web. (object oriented methods applied to the Internet World Wide Web)(includes related article on supporting technologies to convert a mapping database to an OO paradigm and migrate it to Internet).” IEEE Software 15(3): 22.

Cockburn, A. A. R. (1993). “The impact of object-orientation on application development.” IBM Systems Journal 32(3): 420-444.

Comerford, R. (1995). “Software engineering.” IEEE Spectrum 32(1): 62-65.

Cook, J. E., A. L. Wolf, et al. (1998). “A highly effective partition selection policy for object database garbage collection.” IEEE Transactions on Knowledge and Data Engineering 10(1): 153.

Cools, R., D. Laurie, et al. (1997). “Algorithm 764: Cubpack++: a C++ package for automatic two-dimensional cubature.” ACM Transactions on Mathematical Software 23(1): 1.

Cooper, J. W. (1998). “Using design patterns.” Communications of the ACM 41(6): 65-68.

Coplien, J. O. (1997). “Idioms and patterns as architectural literature.(includes related article on Richard Gabriel's 'Simply Understood Code').” IEEE Software 14(1): 36.

Corkill, D. D. (1997). “Countdown to success: dynamic objects, GBB and RADARSTAT-1. (includes related article on scheduling effort).” Communications of the ACM 40(5): 48.

Czejdo, B., C. F. Eick, et al. (1993). “Integrating sets, rules, and data in an object-oriented environment. (Tanguy knowledge-based management system) (Technical).” IEEE Expert 8(1): 59.

David, D. W. (1996). “Business language analysis for object-oriented information systems.” IBM Systems Journal 35(2): 128-150.

Davies, B. and V. B. Davies (1997). “Patching onto the Web: Common Lisp Hypermedia for the Intranet.” Communications of the ACM 40(5): 66.

Davis, J. and T. Morgan (1993). “Object-oriented development at Brooklyn Union Gas. (mainframe implementation of a Smalltalk-like execution environment supports a critical task)(includes related article on the need for the Brooklyn Union sy stem) (Technical).” IEEE Software 10(1): 67.

Davis, M. E., J. D. Grimes, et al. (1996). “Creating global software: Text handling and localization in Taligent's CommonPoint application system.” IBM Systems Journal 35(2): 227-243.

Demeyer, S., T. D. Meijer, et al. (1997). “Design guidelines for 'tailorable' frameworks. (balancing flexibility and customizability in open systems architectures).” Communications of the ACM 40(10): 60.

Deng, P.-S. and C. L. Fuhr (1995). “Using an object-oriented approach to the development of a relational database application system.” Information & Management 29(2): 107-121.

Deubler, H.-H. and M. Koestler (1994). “Introducing object orientation into large and complex systems.” IEEE Transactions on Software Engineering 20(11): 840-848.

Diaz, O. and N. W. Paton (1994). “Extending ODBMSs using metaclasses.” IEEE Software 11(3): 40.

DiPippo, L. C. and V. F. Wolfe (1997). “Object-based semantic real-time concurrency control with bounded imprecision.” IEEE Transactions on Knowledge and Data Engineering 9(1): 135.

Dori, D. and E. Tatcher (1994). “Selective multiple inheritance. (includes related article on the mathematics of selective multiple inheritance).” IEEE Software 11(3): 77.

Driesen, J., J. Fransen, et al. (1998). “Object oriented storage of material data for coupled problems.(Selected Papers from the 11th Conference on the Computation of Electromagnetic Fields - Compumag '97).” IEEE Transactions on Magnetics 34(5): 3415.

Du, T. C.-T. and P. M. Wolfe (1997). “An implementation perspective of applying object-oriented database technologies.” IIE Transactions 29(9): 733.

Dujardin, E., E. Amiel, et al. (1998). “Fast algorithms for compressed multimethod dispatch table generation.” ACM Transactions on Programming Languages & Systems 20(1): 116.

Dumas, J. and P. Parsons (1995). “Discovering the way programmers think about new programming environments.” Communications of the ACM 38(6): 45-56.

Dupuy, F., G. Nilsson, et al. (1995). “The TINA Consortium: toward networking telecommunications information services. (Telecommunications Information Networking Architecture).” IEEE Communications Magazine 33(11): 78.

Edwards, S. H. (1997). “Representation inheritance: a safe form of "white box" code inheritance.” IEEE Transactions on Software Engineering 23(2): 83.

Ehlers, E. M. and E. v. Rensburg (1996). “An object-oriented manufacturing scheduling approach.” IEEE Transactions on Systems 26(1): 17.

Elliot, P. J., J. Diedrichsen, et al. (1996). “An object-oriented system for 3D medical image analysis.” IBM Systems Journal 35(1): 4-24.

Embley, D. W., R. B. Jackson, et al. (1995). “OO Systems analysis: is it or isn't it?(That's Debatable!).” IEEE Software 12(4): 19.

Eustache, P., G. Meunier, et al. (1996). “Finite element toolbox for generic coupling (magnetic, thermal, etc.).” IEEE Transactions on

Magnetics 32(3): 1461.

Fabre, J.-C. and T. Perennou (1998). “A metaobject architecture for fault-tolerant distributed systems: the FRIENDS approach.(Special Issue on Dependability of Computing Systems).” IEEE Transactions on Computers 47(1): 78.

Fayad, M. and M. P. Cline (1996). “Aspects of software adaptability.” Communications of the ACM 39(10): 58-59.

Fayad, M. E. (1997). “Software development process: A necessary evil.” Communications of the ACM 40(9): 101-103.

Fayad, M. E. and D. C. Schmidt (1997). “Object-oriented application frameworks. (includes related article on related trends).” Communications of the ACM 40(10): 32.

Fayad, M. E. and W.-T. Tsai (1995). “Object-oriented experiences. (includes related article).” Communications of the ACM 38(10): 50.

Fayad, M. E., L. J. Hawn, et al. (1993). “Using the Shlaer-Mellor Object-Oriented Analysis Method. (object identification method) (includes related article on McDonnell Douglas' Missile Systems Division's Mission Generation System cruise missile system) (Tutorial).” IEEE Software 10(2): 43.

Fayad, M. E., W.-T. Tsai, et al. (1994). “Adapting an object-oriented development method. (includes related article on a missile simulation system and a new object-oriented programming methodology).” IEEE Software 11(3): 68.

Fayad, M. E., W.-T. Tsai, et al. (1996). “Transition to object-oriented software development.” Communications of the ACM 39(2): 108-121.

Fekete, A., M. F. Kaashoek, et al. (1998). “Implementing sequentially consistent shared objects using broadcast and point-to-point communication.” Journal of the Association for Computing Machinery 45(1): 35.

Fernandez, E. B., E. Gudes, et al. (1994). “A model for evaluation and administration of security in object-oriented databases.” IEEE Transactions on Knowledge and Data Engineering 6(2): 275.

Fichman, R. G. and C. F. Kemerer (1993). “Adoption of software engineering process innovations: The case of object orientation.” Sloan Management Review 34(2): 7-22.

Flint, E. S. (1997). “The COBOL jigsaw puzzle: Fitting object-oriented and legacy applications together.” IBM Systems Journal 36(1): 49-65.

Fotouhi, F., I. Ahmad, et al. (1994). “TOS: A temporal object-oriented system.” Journal of Database Management 5(4): 3-14.

Francois, F., E. Escande, et al. (1994). “A methodology for knowledge representation in expert system, in the field of electrical engineering. (Special Issue: 9th Conference on the Computation of Electromagnetic Fields, Part 2).” IEEE Transactions on Magnetics 30(5): 3656.

Frank, J., B. Rupprecht, et al. (1997). “Knowledge-based assistance for the development of drugs.(includes related articles)(AI in Manufacturing).” IEEE Expert 12(1): 40.

Fuerte-Esquivel, C. R., E. Acha, et al. (1998). “Efficient object oriented power systems software for the analysis of large-scale networks containing FACTS-controlled branches. (flexible alternating current transmission system).” IEEE Transactions on Power Systems 13(2): 464.

Gabel, D. A. (1994). “Technology 1994: Software engineering.” IEEE Spectrum 31(1): 38-41.

Gagliardi, M. and C. Spera (1997). “BLOOMS: A prototype modeling language with object oriented features.” Decision Support Systems 19(1): 1-21.

Gaing, Z.-L., C.-N. Lu, et al. (1996). “An object-oriented approach for implementing power system restoration package.” IEEE Transactions on Power Systems 11(1): 483.

Geihs, K., R. Heite, et al. (1993). “Protected object references in heterogeneous distributed systems. (Technical).” IEEE Transactions on Computers 42(7): 809.

Genesereth, M. R. and S. P. Ketchpel (1994). “Software agents.” Communications of the ACM 37(7): 48-53.

Gillenson, M. L. and R. D. Frost (1993). “The evolution of the meta-data concept: Dictionaries, catalogs, and repositories.” Journal of Database Management 4(3): 17-26.

Gogac, A., C. Dengi, et al. (1998). “Distributed object computing platforms.” Communications of the ACM 41(9): 95-103.

Goldfedder, B. and L. Rising (1996). “A training experience with patterns.” Communications of the ACM 39(10): 60-64.

Gottlob, G., M. Schrefl, et al. (1996). “Extending object-oriented systems with roles.” ACM Transactions on Information Systems 14(3): 268.

Gray, D. N., J. Hotchkiss, et al. (1998). “Modern languages and Microsoft's component object model: Programming COM made simple.” Communications of the ACM 41(5): 55-65.

Griffin, W. G. (1995). “Lessons learned in software reuse.” IEEE Software 12(4): 11.

Gronbaek, K. and R. H. Trigg (1994). “Design issues for a Dexter-based hypermedia system. (Special Section: Hypermedia) (Technical).” Communications of the ACM 37(2): 40.

Guenther, W. and G. Wackerbarth (1993). “Object-oriented design of ISDN call-processing software. (integrated services digital network).” IEEE Communications Magazine 31(4): 40.

Guerraoui, R. (1996). “Strategic directions in object-oriented programming.” ACM Computing Surveys 28(4): 691-700.

Gupta, A. and W. K. Fuchs (1993). “Garbage collection in a distributed object-oriented system.” IEEE Transactions on Knowledge and Data Engineering 5(2): 257.

Gyssens, M., j. Paredaens, et al. (1994). “A graph-oriented object database model.” IEEE Transactions on Knowledge and Data Engineering 6(4): 572.

Haase, K. (1996). “FramerD: Representing knowledge in the large.” IBM Systems Journal 35(3,4): 381-397.

Haggerty, P. and K. Seetharaman (1998). “The benefits of CORBA-based network management.” Communications of the ACM 41(10): 73-79.

Hakavik, B. and A. T. Holen (1994). “Power system modelling and sparse matrix operations using object-oriented programming.” IEEE Transactions on Power Systems 9(2): 1045.

Handschin, E., M. Heine, et al. (1998). “Object-oriented software engineering for transmission planning in open access schemes.” IEEE Transactions on Power Systems 13(1): 94.

Hardwick, M. and R. Bolton (1997). “The industrial virtual enterprise.” Communications of the ACM 40(9): 59-60.

Harrison, R., S. J. Counsell, et al. (1998). “An evaluation of the MOOD set of object-oriented software metrics.” IEEE Transactions on Software Engineering 24(6): 491.

Harrison, W. H., H. Kilov, et al. (1996). “Technical note--From dynamic supertypes to subjects: A natural way to specify and develop systems.” IBM Systems Journal 35(2): 244-256.

Hatton, L. (1998). “Does OO sync with how we think? (object orientation paradigm).” IEEE Software 15(3): 46.

Hayne, S. C. and M. Pendergast (1995). “Experiences with object-oriented group support software development.” IBM Systems Journal 34(1): 96-119.

Henderson-Sellers, B. (1997). “Corrigenda: Software size estimation of object-oriented systems.” IEEE Transactions on Software Engineering 23(4): 260-261.

Henderson-Sellers, B. (1997). “OO Project management: the need for process.” IEEE Software 14(4): 96.

Henning, M. (1998). “Binding, migration, and scalability in CORBA.” Communications of the ACM 41(10): 62-71.

Henrotte, F., B. Meys, et al. (1996). “An object-oriented decomposition of the F.E. procedure. (finite element).” IEEE Transactions on Magnetics 32(3): 1441.

Hevner, A. R. and H. D. Mills (1993). “Box-structured methods for systems development with objects.” IBM Systems Journal 32(2): 232-251.

Hill, R. D., T. Brinck, et al. (1993). “The Rendezvous language and architecture.” Communications of the ACM 36(1): 62-67.

Hitz, M. and B. Montazeri (1996). “Chidamber and Kamerer's metrics suite: A measurement theory perspective.” IEEE Transactions on Software Engineering 22(4): 267-271.

Holsing, N. F. and D. C. Yen (1997). “Integrating computer-aided software engineering and object-oriented systems: A preliminary analysis.” International Journal of Information Management 17(2): 95-113.

Holzle, U. and D. Ungar (1996). “Reconciling responsiveness with performance in pure object-oriented languages.(object-oriented programming and use of optimization techniques).” ACM Transactions on Programming Languages & Systems 18(4): 355.

Honiden, S., K. Nishimura, et al. (1994). “An application of artificial intelligence to object-oriented performance design for real-time systems.” IEEE Transactions on Software Engineering 20(11): 849-867.

Honiden, S., N. Kotaka, et al. (1993). “Formalizing specification modeling in OOA. (object-oriented analysis still needs a specification process defined in detail)(Technical Article: includes related article on automating helicopter landings) (Cover Story).” IEEE Software 10(1): 54.

Hotter, M. (1994). “Optimization and efficiency of an object-oriented analysis-synthesis coder.” IEEE Transactions on Circuits and Systems for Video Technology 4(2): 181.

Hsu, C., L. Gerhardt, et al. (1994). “Adaptive integrated manufacturing enterprises: information technology for the next decade.” IEEE Transactions on Systems 24(5): 828.

Huang, Y.-M. and S.-H. Lin (1996). “An efficient inductive learning method for object-oriented database using attribute entropy.(Special Section on Mining of Databases).” IEEE Transactions on Knowledge and Data Engineering 8(6): 946.

Huh, S.-Y. and Q. B. Chung (1995). “A model management framework for heterogeneous algebraic models: Object-oriented database management systems approach.” Omega 23(3): 235-256.

Hyman, R. B. (1993). “Creative chaos in high-performance teams: an experience report. (ARINC Research Corp.'s Software Reusability Department uses object technology to create libraries of reusable software components) (Project Organization and Management special section).” Communications of the ACM 36(10): 56.

Isakowitz, T., S. Schocken, et al. (1995). “Toward a logical/physical theory of spreadsheet modeling.” ACM Transactions on Information Systems 13(1): 1.

Ishikawa, H., F. Kozakura, et al. (1993). “The model, language, and implementation of an object-oriented multimedia knowledge base management system. (Technical).” ACM Transactions on Database Systems 18(1): 1.

Ishikawa, H., Y. Yamane, et al. (1996). “An object-oriented database system Jasmine: implementation, application, and extension.” IEEE Transactions on Knowledge and Data Engineering 8(2): 285.

Islam, N. and M. Devarakonda (1996). “An essential design pattern for fault-tolerant distributed state sharing. (Recoverable Distributor).” Communications of the ACM 39(10): 65.

Islam, N. and R. H. Campbell (1996). “Latest developments in operating systems.” Communications of the ACM 39(9): 38-40.

Jaccheri, M. L. and R. Conradi (1993). “Techniques for process model evolution in EPOS.” IEEE Transactions on Software Engineering 19(12): 1145-1156.

Jackson, R. B., W. C. Giauque, et al. (1996). “Concepts of single-paradigm object-oriented development, with application to a manufacturing information system.” IEEE Transactions on Systems 26(5): 583.

Jacky, J. (1995). “Specifying a safety-critical control system in Z.” IEEE Transactions on Software Engineering 21(2): 99-106.

Jacobson, I. (1993). “Is object technology software's industrial platform? (object oriented approach is unprecedented in computer technology) (Cover Story).” IEEE Software 10(1): 24.

Janneck, J. W. and M. Naedele (1998). “Modeling a die bonder with Petri nets: a case study.” IEEE Transactions on Semiconductor Manufacturing 11(3): 404.

Jean, M., reoli, et al. (1996). “Integrated computational paradigms for flexible client-server communication.” ACM Computing Surveys 28(2): 297.

Johnson, R. E. (1997). “Frameworks = (components + patterns). (frameworks for object-oriented software development).” Communications of the ACM 40(10): 39.

Johnson, T. (1995). “Characterizing the performance of algorithms for lock-free objects.(concurrency control procedures).” IEEE Transactions on Computers 44(10): 1194.

Jorgensen, P. C. and C. Erickson (1994). “Object-oriented integration testing.” Communications of the ACM 37(9): 30-38.

Jungthirapanich, C. and C. O. Benjamin (1995). “A knowledge-based decision support system for locating a manufacturing facility.” IIE Transactions 27(6): 789-799.

Karacal, S. C. and J. H. Mize (1998). “Object-oriented software implementation for manufacturing systems.(A Formal Structure for Discrete Event Simulation, part 2).” IIE Transactions 30(3): 217.

Karaorman, M. and J. Bruno (1993). “Introducing concurrency to a sequential language.” Communications of the ACM 36(9): 103-116.

Kemper, A., C. Kilger, et al. (1994). “Function materialization in object bases: design, realization, and evaluation.” IEEE Transactions on Knowledge and Data Engineering 6(4): 587.

Kesim, F. N. and M. Sergot (1996). “A logic programming framework for modeling temporal objects.” IEEE Transactions on Knowledge and Data Engineering 8(5): 724.

Kifer, M., G. Lausen, et al. (1995). “Logical foundations of object-oriented and frame-based languages.” Journal of the Association for Computing Machinery 42(4): 741.

Kim, K. H. and C. Subbaraman (1997). “Fault-tolerant real-time objects.” Communications of the ACM 40(1): 75-82.

King, R. and M. Novak (1993). “Designing database interfaces with DBface. (Technical).” ACM Transactions on Information Systems 11(2): 105.

Kishimoto, Y., N. Kotaka, et al. (1995). “Adapting object-communication methods dynamically.(includes related article).” IEEE Software 12(3): 65.

Klerer, S. M. (1993). “System management information modeling.” IEEE Communications Magazine 31(5): 38.

Kochikar, V. P. (1998). “The object-powered Web. (object-oriented programming applied to the Internet World Wide Web).” IEEE Software 15(3): 57.

Kozaczynski, W. and A. Kuntzmann-Combelles (1993). “What it takes to make OO work. (object oriented technology)(includes related article on books to read about object-oriented programming and data bases) (Cover Story).” IEEE Software 10(1): 20.

Kozaczynski, W. and G. Booch (1998). “Component-based software engineering.” IEEE Software 15(5): 34.

Krause, P. J., P. J. Byers, et al. (1994). “Formal specification and decision support.” Decision Support Systems 12(3): 189-197.

Kung, D., J. Gao, et al. (1995). “Developing an object-oriented software testing and maintenance environment.” Communications of the ACM 38(10): 75-87.

Kunieda, T., S. Sugimoto, et al. (1993). “A Synchronous Digital Hierarchy network management system.” IEEE Communications Magazine 31(11): 84.

Kunz, J. C., Y. Jin, et al. (1996). “Support for integrated value-based maintenance planning.(use of Intelligent Real-Time Maintenance Management system for maintenance of manufacturing or power plant).” IEEE Expert 11(4): 35.

Kurumbalapitiya, D., S. Ratnajeevan, et al. (1993). “An object-oriented representation of electromagnetic knowledge.” IEEE Transactions on Magnetics 29(2): 1939.

Kwon, O. B. and S. J. Park (1996). “RMT: A modeling support system for model reuse.” Decision Support Systems 16(2): 131-153.

Laddaga, R. and J. Veitch (1997). “Dynamic object technology.” Communications of the ACM 40(5): 36-38.

Larsen, P. G., J. Fitzgerald, et al. (1996). “Applying formal specification in industry.(Lessons Learned).” IEEE Software 13(3): 48.

Lauesen, S. (1998). “Real-life object-oriented systems.” IEEE Software 15(2): 76.

Lavazza, L. (1993). “Comments on considering 'class' harmful.” Communications of the ACM 36(1): 112-113.

Lea, R., C. Jacquemot, et al. (1993). “COOL: System support for distributed programming.” Communications of the ACM 36(9): 37-46.

Lee, A. H. and J. L. Zachary (1995). “Reflections on metaprogramming.” IEEE Transactions on Software Engineering 21(11): 883-893.

Lee, B. S. and G. Wiederhold (1994). “Outer joins and filters for instantiating objects from relational databases through views.” IEEE Transactions on Knowledge and Data Engineering 6(1): 108.

Lee, S. and R. M. O'Keefe (1996). “The effect of knowledge representation schemes on maintainability of knowledge-based systems.” IEEE Transactions on Knowledge and Data Engineering 8(1): 173.

Lee, S.-Y. and R.-L. Liou (1996). “a multi-granularity locking model for concurrency control in object-oriented database systems.” IEEE Transactions on Knowledge and Data Engineering 8(1): 144.

Lee, W.-C. and D. L. Lee (1998). “Path dictionary: a new access method for query processing in object-oriented databases.” IEEE Transactions on Knowledge and Data Engineering 10(3): 371.

Lefrancois, P. and B. Montreuil (1994). “An object-oriented knowledge representation for intelligent control of manufacturing workstations.” IIE Transactions 26(1): 11.

Lenard, M. L. (1993). “An object-oriented approach to model management.” Decision Support Systems 9(1): 67-73.

Leone, N., P. Rullo, et al. (1997). “A deductive environment for dealing with objects and nonmonotonic reasoning.” IEEE Transactions on Knowledge and Data Engineering 9(4): 539.

Leymann, F. and W. Altenhuber (1994). “Managing business processes as an information resource.” IBM Systems Journal 33(2): 326-348.

Li, Q. and F. H. Lochovsky (1998). “ADOME: an Advanced Object Modeling Environment.” IEEE Transactions on Knowledge and Data Engineering 10(2): 255.

Lieberherr, K. (1993). “Formal foundations for object-oriented data modeling.” IEEE Transactions on Knowledge and Data Engineering 5(3): 462.

Lieberherr, K. J. and C. Xiao (1993). “Object-oriented software evolution.” IEEE Transactions on Software Engineering 19(4): 313-343.

Lieberherr, K. J., I. Silva-Lepe, et al. (1994). “Adaptive object-oriented programming using graph-based customization.” Communications of the ACM 37(5): 94-101.

Liu, L. and R. Meersman (1996). “The building blocks for specifying communication behavior of complex objects: an activity-driven approach.” ACM Transactions on Database Systems 21(2): 157.

Liu, L., R. Zicari, et al. (1997). “The role of polymorphic reuse mechanisms in schema evolution in an object-oriented database.” IEEE Transactions on Knowledge and Data Engineering 9(1): 50.

Liu, S., A. J. Offutt, et al. (1998). “SOFL: a formal engineering methodology for industrial applications. (Structured-Object-based-Formal Language)(Special Section on Formal Methods in Software Practice).” IEEE Transactions on Software Engineering 24(1): 24.

Lohr, K.-P. (1993). “Concurrency annotations for reusable software.” Communications of the ACM 36(9): 81-89.

Love, T. (1995). “Seven deadly sins of object-oriented development.” Information Systems Management 12(3): 84-86.

Low, G. C., B. Henderson-Sellers, et al. (1995). “Comparison of object-oriented and traditional systems development issues in distributed environments.” Information & Management 28(5): 327-340.

Lu, S., E. Swidenbank, et al. (1995). “An object-oriented power plant adaptive control system design tool.” IEEE Transactions on Energy Conversion 10(3): 600.

Lu, X.-M. and T. S. Dillon (1994). “An algebraic theory of object-oriented systems.” IEEE Transactions on Knowledge and Data Engineering 6(3): 412.

Luckham, D. C., J. J. Kennedy, et al. (1995). “Specification and analysis of system architecture using Rapide.(Special Issue on Software Architecture).” IEEE Transactions on Software Engineering 21(4): 336.

Ma, J. (1995). “An object-oriented framework for model management.” Decision Support Systems 13(2): 133-139.

Ma, J. (1997). “Type and inheritance theory for model management.” Decision Support Systems 19(1): 53-60.

Machiels, L. and M. O. Deville (1997). “Fortran 90: an entry to object-oriented programming for the solution of partial differential equations." ACM Transactions on Mathematical Software 23(1): 32.

Maeda, Y. and Ikeda (1996). "Composite object modeling.(Part A: Systems and Humans)(A Composite Object Model of an Urban Water Network Based on Multidisciplinary Knowledge Bases, part 1)." IEEE Transactions on Systems 26(6): 718.

Maffeis, S. and D. C. Schmidt (1997). "Constructing reliable distributed communication systems with CORBA.(Common Object Request Broker Architecture)(Distributed Object Computing)." IEEE Communications Magazine 35(2): 56.

Magalhaes, A. L. C. D. C. and R. C. Mesquita (1998). "Requirements for a solid modeler coupled to finite-element mesh generators.(Selected Papers from the 11th Conference on the Computation of Electromagnetic Fields - Compumag '97)." IEEE Transactions on Magnetics 34(5): 3447.

Majetic, I. and E. L. Leiss (1997). "Authorization and revocation in object-oriented databases." IEEE Transactions on Knowledge and Data Engineering 9(4): 668.

Manola, F. (1995). "Interoperability issues in large-scale distributed object systems." ACM Computing Surveys 27(2): 268-270.

Marchisio, L., E. Ronco, et al. (1993). "Modelling the user interface." IEEE Communications Magazine 31(5): 68.

Maring, B. (1996). "Object-oriented development of large applications.(Lessons Learned)." IEEE Software 13(3): 33.

McConnell, S. (1996). "Missing in action: information hiding." IEEE Software 13(2): 128.

McDavid, D. W. (1996). "Business language analysis for object-oriented information systems." IBM Systems Journal 35(2): 128-150.

McGregor, J. D. and T. D. Korson (1994). "Integrated object-oriented testing and development processes." Communications of the ACM 37(9): 59-77.

Mellor, S. J. and R. Johnson (1997). "Why explore object methods, patterns, and architectures?(includes related articles)." IEEE Software 14(1): 27.

Mentzas, G. N. (1997). "Re-engineering banking with object-oriented models: Towards customer information systems." International Journal of Information Management 17(3): 179-197.

Meyer, B. (1993). "Systematic concurrent object-oriented programming." Communications of the ACM 36(9): 56-80.

Mierop, J., S. Tax, et al. (1993). "Service interaction in an object-oriented environment." IEEE Communications Magazine 31(8): 46.

Mili, H., F. Mili, et al. (1995). "Reusing software: Issues and research directions." IEEE Transactions on Software Engineering 21(6): 528-562.

Mohan, L. and R. L. Kashyap (1993). "A visual query language for graphical interaction with schema-intensive databases." IEEE Transactions on Knowledge and Data Engineering 5(5): 843.

Monroe, R. T., r. Kompanek, et al. (1997). "Architectural styles, designs, patterns, and objects.(includes related article on software architecture description)." IEEE Software 14(1): 43.

Moore, C. R. and R. C. Stanphill (1994). "The PowerPC alliance. (IBM, Apple, Motorola) (The Making of the PowerPC) (Cover Story)." Communications of the ACM 37(6): 25.

Muhanna, W. A. (1993). "An object-oriented framework for model management and DSS development." Decision Support Systems 9(2): 217-229.

Muhlhauser, M., W. Gerteis, et al. (1993). "DOCASE: A methodic approach to distributed programming." Communications of the ACM 36(9): 127-138.

Muller, N. J. (1994). "Applications development tools." Information Systems Management 11(3): 23-27.

Murphy, G. C., P. Townsend, et al. (1994). "Experiences with cluster and class testing." Communications of the ACM 37(9): 39-47.

Mycroft, A. (1996). "On integration of programming paradigms." ACM Computing Surveys 28(2): 309-311.

Nesi, P. (1998). "Managing OO projects better. (object-oriented software development)." IEEE Software 15(4): 50.

Ng, F., G. Butler, et al. (1995). "An intelligent tutoring system for the Dijkstra-Gries methodology." IEEE Transactions on Software Engineering 21(5): 415-428.

Norton, C. D., B. K. Szymanski, et al. (1995). "Object-oriented parallel computation for plasma simulation." Communications of the

ACM 38(10): 88-100.

Olivier, M. S. and S. H. V. Solms (1994). “A taxonomy for secure object-oriented databases. (Technical).” ACM Transactions on Database Systems 19(1): 3.

Oomoto, E. and K. Tanaka (1993). “OVID: design and implementation of a video-object database system. (object-oriented video information database).” IEEE Transactions on Knowledge and Data Engineering 5(4): 629.

Palsberg, J. (1996). “Type inference for objects.” ACM Computing Surveys 28(2): 358-359.

Palsberg, J. and S. Smith (1996). “Constrained types and their expressiveness.” ACM Transactions on Programming Languages & Systems 18(5): 519.

Palsberg, J., C. Xiao, et al. (1995). “Efficient implementation of adaptive software.” ACM Transactions on Programming Languages & Systems 17(2): 264.

Pancake, C. M. (1995). “The promise and the cost of object technology: A five-year forecast.” Communications of the ACM 38(10): 32-49.

Papazoglou, M., A. Delis, et al. (1997). “Class library support for workflow environments and applications.” IEEE Transactions on Computers 46(6): 673.

Papzoglou, M. P. (1995). “Unraveling the semantics of conceptual schemas.” Communications of the ACM 38(9): 80.

Parsons, J. and Y. Wand (1997). “Choosing classes in conceptual modeling.” Communications of the ACM 40(6): 63-69.

Parsons, J. and Y. Wand (1997). “Using objects for systems analysis. (object-oriented programming and software design).” Communications of the ACM 40(12): 104.

Peckham, J., F. Maryanski, et al. (1996). “Towards the correctness and consistency of update semantics in semantic database schema.” IEEE Transactions on Knowledge and Data Engineering 8(3): 503.

Pei, D. and C. Cutone (1995). “Object-oriented analysis and design.” Information Systems Management 12(1): 54-60.

Peters, R. J. and M. T. Ozsu (1997). “An axiomatic model of dynamic schema evolution in objectbase systems.” ACM Transactions on Database Systems 22(1): 75.

Pinheiro, F. A. C. and J. A. Goguen (1996). “An object-oriented tool for tracing requirements.(contains related articles).” IEEE Software 13(2): 52.

Pitoura, E., O. Bukhres, et al. (1995). “Object orientation in multidatabase systems.” ACM Computing Surveys 27(2): 141-195.

Pittman, M. (1993). “Lessons learned in managing object-oriented development. (includes related article on managing reuse) (current techniques require adaptation, especially to bring reuse under the manager's control) (Cover Story).” IEEE Software 10(1): 43.

Poo, G.-S. and C.-G. Chew (1996). “Modeling of the XOM/XMP application programming interface (API).(X/Open OSI Abstract Data Manipulation API, X/Open Management Protocols).” IEEE Communications Magazine 34(8): 134.

Popescu, M., I. Munteanu, et al. (1998). “An object oriented data structure for field analysis.(Selected Papers from the 11th Conference on the Computation of Electromagnetic Fields - Compumag '97).” IEEE Transactions on Magnetics 34(5): 3403.

Porter, M. E., Competitive Strategy, 1980.

Poston, R. M. (1994). “Automated testing from object models.” Communications of the ACM 37(9): 48-58.

Potok, T. E. and M. A. Vouk (1997). “The effects of the business model on object-oriented software development productivity.” IBM Systems Journal 36(1): 140-161.

Potts, C. (1993). “Software-engineering research revisited. (includes related article on problems with the Research-Then-Transfer approach).” IEEE Software 10(5): 19.

Poulovassilis, A. and M. Levene (1994). “A nested-graph model for the representation and manipulation of complex objects. (Technical).” ACM Transactions on Information Systems 12(1): 35.

Price, C. R., J. C. Trinkle, et al. (1993). “Network-based infrastructure for distributed remote operations and robotics research. (Correspondence) (Technical).” IEEE Transactions on Robotics and Automation 9(5): 702.

Purao, S., H. Jain, et al. (1998). “Effective distribution of object-oriented applications.” Communications of the ACM 41(8): 100-108.

Qing, L. and D. McLeod (1994). “Conceptual database evolution through learning in object databases.” IEEE Transactions on Knowledge and Data Engineering 6(2): 205.

Qing, L. and L. S. Huang (1995). “A dynamic data model for a video database management system.” ACM Computing Surveys 27(4): 602.

Quilici, A. (1994). “A memory-based approach to recognizing programming plans.” Communications of the ACM 37(5): 84-93.

Rabin, S. (1995). “Host developers to object technicians: Transition strategies for OO development.” Information Systems Management 12(3): 30-39.

Radin, G. (1996). “Object technology in perspective.” IBM Systems Journal 35(2): 124-127.

Rajkumar, T. M. and D. L. Dawley (1996). “Designing and managing client/server DBMSs.” Information Systems Management 13(2): 49-57.

Rajlich, V. and J. H. Silva (1996). “Evolution and reuse of orthogonal architecture.” IEEE Transactions on Software Engineering 22(2): 153.

Ram, D. J., N. Vivekananda, et al. (1997). “Constraint meta-object: a new object model for distributed collaborative designing.” IEEE Transactions on Systems 27(2): 208.

Rettig, M., G. Simons, et al. (1993). “Extended objects.” Communications of the ACM 36(8): 19-24.

Richardson, J. E., M. J. Carey, et al. (1993). “The design of the E programming language. (Technical).” ACM Transactions on Programming Languages & Systems 15(3): 494.

Robertson, P. (1997). “Integrating legacy systems with modern corporate applications.” Communications of the ACM 40(5): 39-46.

Rosson, M. B. and J. M. Carroll (1996). “Scaffolded examples for learning object-oriented design.” Communications of the ACM 39(4): 46-47.

Roy, P. V., S. Haridi, et al. (1997). “Mobile objects in Distributed Oz. (concurrent object-oriented language).” ACM Transactions on Programming Languages & Systems 19(5): 804.

Roy, U. and Y.-C. Fang (1996). “Tolerance representation scheme for a three-dimensional product in an object-oriented programming environment.” IIE Transactions 28(10): 809-819.

Roy, U. and Y.-C. Fang (1997). “Optimal tolerance re-allocation for the generative process sequence.” IIE Transactions 29(1): 37-44.

Rundensteiner, E. A., L. Bic, et al. (1994). “Set restrictions for semantic groupings.” IEEE Transactions on Knowledge and Data Engineering 6(2): 193.

Samarati, P., E. Bertino, et al. (1997). “Information flow control in object-oriented systems.” IEEE Transactions on Knowledge and Data Engineering 9(4): 524.

Sanchez, N. G. and J. Choobineh (1997). “Achieving reuse with OO technology.” Information Systems Management 14(2): 48-55.

Schauble, P. and B. Wuthrich (1994). “On the expressive power of query languages.” ACM Transactions on Information Systems 12(1): 69.

Schmidt, D. C. (1995). “Using design patterns to develop reusable object-oriented communication software.” Communications of the ACM 38(10): 65-71+.

Schmidt, D. C. (1998). “Evaluating architectures for multithreaded object reguest brokers.” Communications of the ACM 41(10): 54-60.

Schmidt, D. C. and M. E. Fayad (1997). “Lessons learned: building reusable OO frameworks for distributed software.” Communications of the ACM 40(10): 85.

Schmidt, D. C., A. S. Gokhale, et al. (1997). “A high-performance end system architecture for real-time CORBA.” IEEE Communications Magazine 35(2): 72.

Schnase, J. L., J. J. Leggett, et al. (1993). “Semantic data modeling of hypermedia associations. (Technical).” ACM Transactions on Information Systems 11(1): 27.

Schwabe, D. and G. Rossi (1995). “The object-oriented hypermedia design model.” Communications of the ACM 38(8): 45-46.

Seiter, L. M., J. Palsberg, et al. (1998). “Evolution of object behavior using context relations.” IEEE Transactions on Software Engineering 24(1): 79.

Sheetz, S. D., D. P. Tegarden, et al. (1994). “A group support systems approach to cognitive mapping.” Journal of Management Information Systems 11(1): 31-57.

Shlaer, S. and S. J. Mellor (1997). “Recursive Design of an application-independent architecture.(software development)(includes related articles).” IEEE Software 14(1): 61.

Shlaer, S., S. J. Mellor, et al. (1994). “Research in object-oriented analysis and design; Response.” Communications of the ACM 37(1): 109-111.

Shoval, P. and I. Frumermann (1994). “OO and EER conceptual schemas: A comparison of user comprehension.” Journal of Database Management 5(4): 28-38.

Shrivastava, S. K. and D. L. McCue (1994). “Structuring fault-tolerant object systems for modularity in a distributed environment.” IEEE Transactions on Parallel and Distributed Systems 5(4): 421.

Shvartsman, A. A. (1993). “Dealing with history and time in a distributed enterprise manager. (Special Issue: Integrated Network Management) (Technical).” IEEE Network 7(6): 32.

Siegel, J. (1998). “OMG overview: CORBA and the OMA in enterprise computing.” Communications of the ACM 41(10): 37-43.

SIGS Publications (1998), Telephone discussion between Chuck Wood and SIGS customer service on 11/24/98.

Silberschatz, A., H. F. Korth, et al. (1996). “Data models.” ACM Computing Surveys 28(1): 105-108.

Silva, E. J. and R. C. Mesquita (1996). “Data management in finite element analysis programs using object-oriented techniques.” IEEE Transactions on Magnetics 32(3): 1445.

Silva, E. J., R. C. Mesquita, et al. (1994). “An object-oriented finite-element program for electromagnetic field computation. (Special Issue: 9th Conference on the Computation of Electromagnetic Fields, Part 2).” IEEE Transactions on Magnetics 30(5): 3618.

Singh, N. (1998). “Unifying heterogeneous information models: Semantic tags support knowledge webs.” Communications of the ACM 41(5): 37-44.

Singhal, S. and B. Nguyen (1998). “The Java factor.” Communications of the ACM 41(6): 34-37.

Snoek, M. and G. Dedene (1998). “Existence dependency: the key to semantic integrity between structural and behavioral aspects of object types.” IEEE Transactions on Software Engineering 24(4): 233.

Song, X. (1997). “Systematic integration of design methods.” IEEE Software 14(2): 107.

Srinivasan, V. and D. T. Chang (1997). “Object persistence in object-oriented applications.” IBM Systems Journal 36(1): 66-87.

Staringer, W. (1994). “Constructing applications from reusable components. (software reuse)(includes related article on graphical software).” IEEE Software 11(5): 61.

Su, S. Y. W., R. Jawadi, et al. (1998). “OSAM. KBMS/P: a parallel, active, object-oriented knowledge base server.” IEEE Transactions on Knowledge and Data Engineering 10(1): 55.

Su, S. Y. W., S. J. Hyun, et al. (1998). “Temporal association algebra: a mathematical foundation for processing object-oriented temporal databases.” IEEE Transactions on Knowledge and Data Engineering 10(3): 389.

Su, Y. W. S., M. Guo, et al. (1993). “Association algebra: a mathematical foundation for object-oriented databases.” IEEE Transactions on Knowledge and Data Engineering 5(5): 775.

Sutherland, J. (1995). “Business objects in corporate information systems.” ACM Computing Surveys 27(2): 274-276.

Taivalsaari, A. (1996). “On the notion of inheritance.” ACM Computing Surveys 28(3): 438-479.

Teuhola, J. (1996). “Path signatures: a way to speed up recursion in relational databases.” IEEE Transactions on Knowledge and Data Engineering 8(3): 446.

Thakore, A. K., S. Y. W. Su, et al. (1995). “Algorithms for asynchronous parallel processing of object-oriented databases.” IEEE

Transactions on Knowledge and Data Engineering 7(3): 487.

Thomas, D. (1995). “Ubiquitous applications: Embedded systems to mainframe.” Communications of the ACM 38(10): 112-114.

Thomas, R. K. and R. S. Sandhu (1996). “A trusted subject architecture for multilevel secure object-oriented databases.(Special Issue on Secure Database Systems Technology).” IEEE Transactions on Knowledge and Data Engineering 8(1): 16.

Vaishnavi, V. K., G. C. Buchanan, et al. (1997). “A data/knowledge paradigm for the modeling and design of operations support systems.” IEEE Transactions on Knowledge and Data Engineering 9(2): 275.

Vinoski, S. (1997). “CORBA: integrating diverse applications within distributed heterogeneous environments.(Common Object Request Broker Architecture)(Distributed Object Computing).” IEEE Communications Magazine 35(2): 46.

Vinoski, S. (1998). “New features for CORBA 3.0.” Communications of the ACM 41(10): 44-52.

Wang, S. (1994). “OO modeling of business processes.” Information Systems Management 11(2): 36-43.

Wang, S. (1995). “An object-oriented approach to work group support systems analysis.” International Journal of Information Management 15(3): 199-207.

Wang, S. (1995). “Object-oriented task analysis.” Information & Management 29(6): 331-341.

Wang, S. (1996). “Toward formalized object-oriented management information systems analysis.” Journal of Management Information Systems 12(4): 117-141.

Watanabe, S. (1997). “Professionalism through OO and reuse.” IEEE Software 14(1): 26.

Wiener, R. (1998). “Watch your language! (workable object orientation programming language other than those written in C++).” IEEE Software 15(3): 55.

Wilde, N., P. Matthews, et al. (1993). “Maintaining object-oriented software. (design practices and support tools are needed)(Technical Article) (Cover Story).” IEEE Software 10(1): 75.

Wolf, G. (1994). “Schedule management: An object oriented approach.” Decision Support Systems 11(4): 373-388.

Wolfe, V. F., K. F. Lau, et al. (1994). “Real-time object-oriented database support for program stock trading.” Journal of Database Management 5(2): 3-17.

Wong, S. and S. Tojo (1996). “A deductive object-oriented database system for situated inference in law.” IEEE Transactions on Knowledge and Data Engineering 8(3): 496.

Wong, S. T. C. and J. L. Wilson (1993). “A set of design guidelines for object-oriented deductive systems.” IEEE Transactions on Knowledge and Data Engineering 5(5): 895.

Wood, Chuck, Visual J++ 6 Secrets, IDG Books Worldwide, New York, 1998, p. 235.

Wood, E. J. (1993). “An object-oriented SECS programming environment. (SEMI Equipment Communication Standard) (Special Issue on Contributions from the 1992 Advanced Semiconductor Manufacturing Conference and Workshop and the 1992 International Semiconductor Manufacturing Science Symposium).” IEEE Transactions on Semiconductor Manufacturing 6(2): 119.

Wu, C.-Z., H.-C. Chan, et al. (1994). “An experimental study of object-oriented query language and relational query language for novice users.” Journal of Database Management 5(4): 16-27.

Yamagishi, K., N. Sasaki, et al. (1995). “An implementation of a TMN-based SDH management system in Japan. (Telecommunication Management Network in Synchronous Digital Hierarchy).” IEEE Communications Magazine 33(3): 80.

Yamazaki, S., K. Kajihara, et al. (1993). “Object-oriented design of telecommunication software. (NTT Software Laboratories develops a system to handle the complexity of telecommunication systems)(Technical Article) (Cover Story).” IEEE Software 10(1): 81.

Yoo, S. B. and P. C. Y. Sheu (1993). “Evaluation and optimization of query programs in an object-oriented and symbolic information system.” IEEE Transactions on Knowledge and Data Engineering 5(3): 479.

Zaharioudakis, M. and M. J. Carey (1998). “Hierarchical, adaptive cache consistency in a page server OODBMS. (object-oriented database management systems).” IEEE Transactions on Computers 47(4): 427.

Zeigler, B. P., T. H. Cho, et al. (1996). “A knowledge-based simulation environment for hierarchical flexible manufacturing.” IEEE Transactions on Systems 26(1): 81.

Zhou, E. Z. (1996). “Object-oriented programming, C++ and power system simulation.” IEEE Transactions on Power Systems 11(1): 206.

Zhou, L., E. A. Rundensteiner, et al. (1997). “Schema evolution of an object-oriented real-time data base system for manufacturing automation.” IEEE Transactions on Knowledge and Data Engineering 9(6): 956.