December 2004

OOREA: An Object-Oriented Resources, Events, Agents Model for Enterprise Systems Design

Uday Murthy
University of South Florida

Casper Wiggins, Jr.
University of North Carolina at Charlotte

OOREA: AN OBJECT-ORIENTED RESOURCES, EVENTS, AGENTS MODEL FOR ENTERPRISE SYSTEMS DESIGN

Uday S. Murthy
School of Accountancy
University of South Florida
Tampa, FL U.S.A.
umurthy@coba.usf.edu

Casper E. Wiggins, Jr.
Belk College of Business Administration
University of North Carolina at Charlotte
Charlotte, NC U.S.A.
cwiggins@email.uncc.edu

Abstract

A number of modeling approaches have been proposed in the literature for designing business information systems. This paper critiques prior data modeling approaches and presents an integrated object-oriented modeling approach that captures both the structural and the behavioral aspects of the business domain. Although there is considerable interest in object-oriented (OO) technologies in practice and in the information systems literature, there is no widely accepted OO modeling approach that facilitates the identification of objects from a business information processing perspective. Based on McCarthy's (1982) resources, events, agents (REA) framework, the business process focused object-oriented ontology presented in this paper identifies the key ***resources****,* ***events****, and* ***agents*** *in an enterprise information systems context. Termed OOREA, the ontology extends McCarthy's REA model by capturing both the structural aspects of modeling, in terms of the objects of interest in the domain, and also the behavioral aspects in terms of the processes that modify objects. Application of the model is illustrated in the context of sales and related events for a retailing enterprise.*

Keywords: Enterprise systems development, business process modeling, object-oriented modeling, systems analysis and design, entity-relationship modeling

Introduction

With the move toward enterprise-wide information systems, there is an emergent need for ontologies to guide enterprise systems design and development using the latest tools and techniques. While relational databases still dominate enterprise systems, there is considerable interest in object-oriented database systems for meeting the complex information processing needs of large corporations (Sultan and Chan, 2000). A key advantage of the object-oriented approach is the integration of structural (static) and behavioral (dynamic) aspects in one model (Booch et al. 1999; Navathe 1992; Ochuodho 1992). Although UML, proposed by Booch et al. (1999), is generally accepted as the de facto standard for object-oriented analysis and design, it is not oriented specifically toward the identification of objects of interest within a business information processing framework. Thus, what is lacking is a *business process oriented* methodology for guiding object-oriented analysis and design for business information systems. The purpose of this paper is to propose a business process based ontology for modeling enterprise systems within an object-oriented framework.

Several logical modeling approaches have been proposed for the purpose of designing database-driven information systems. Chen's (1976) entity-relationship (ER) modeling approach was one of the first logical models that allowed designers to focus on the entities of interest and the logical relationships among them independent of DBMS-specific features. A number of extensions have been proposed to ER modeling, all aimed at capturing more of the semantics of the application domain being modeled (Hull and King 1987; Teorey et al. 1986). Extended entity-relationship models (EER) focus on the data-oriented (structural) aspects of the enterprise's information domain, necessitating the development of separate process-oriented models using techniques such as process flowcharts and data-flow diagrams to represent the behavioral aspects of the domain. Prior to systems implementation, the data-oriented model and process-oriented models must be reconciled and integrated with one another. Several researchers have called for the development of models that integrate the data and process aspects of the system in one view (Hull and King 1987; Navathe 1992; Ochuodho 1992).

While the lack of integration of structural and behavioral aspects is problematic from a systems design perspective, a more fundamental problem in the systems analysis phase of enterprise modeling is the identification of entities/objects to be represented in the model. McCarthy (1982) proposed an ontology for identifying the key *resources*, *events*, and *agents* in an enterprise context, which is referred to as the REA framework. The REA ontology has its origins in Chen's ER modeling approach and is a means of representing information about economic resources, economic events, and economic agents and relationships among them. McCarthy's original work has led to a substantial body of research on extensions to and applications of the REA model (David et al. 1999; Geerts and McCarthy 1999, 2000, 2002). However, a key limitation of REA modeling is that it takes a structural view of the system, with behavioral aspects having to be modeled using techniques such as data-flow diagrams.

There has been considerable interest in the object-oriented paradigm from both modeling and technology perspectives. Object-oriented modeling approaches can represent both the data- and process-oriented views of the system using the unifying metaphor of the *object* (Booch 1991; Coad and Yourdon 1991a, 1991b). A number of alternative object-oriented modeling (OOM) approaches were initially proposed in the literature (for a review of the alternative proposals, see Monarchi and Puhr 1992). Booch et al. proposed the unified modeling language (UML) for object-oriented modeling, and there appears to be widespread acceptance of UML as a standard. A significant advantage of object-oriented modeling is that semantics represented in the logical model correspond directly to constructs in object-oriented technologies. For example, objects, their characteristics, and associated procedures shown in an object-oriented modeling correspond to object classes, their attributes, and their encapsulated methods in object-oriented programming environments such as C++, C#, or Java. The purpose of this paper is to extend McCarthy's (1982) REA framework by proposing an object-oriented ontology for modeling enterprise-wide phenomena. Termed OOREA, the proposed ontology is used to model **r**esources, **e**vents, and **a**gents in the enterprise in terms of object classes, with each object class having certain attributes and encapsulated methods.

The next section briefly reviews related research on semantic data modeling and object-oriented modeling. The drawbacks of these approaches are discussed. In the third section, the OOREA model is presented and an illustrative OOREA model for a retailing enterprise is developed. This model is then compared with McCarthy's (1979) ER model for a retailing enterprise. The concluding section summarizes the paper and discusses future research directions.

Background and Prior Research

This section discusses semantic data modeling and the problems associated with models such as the ER and REA models. Prior research on OO modeling approaches and OO extensions to the ER model are also reviewed. The limitations of these models are discussed.

Semantic Data Modeling

Semantic data models seek to naturally and directly incorporate more of the semantics, or meaning, of the application domain into the database schema (Hammer and McLeod 1981). Semantic models provide systems analysts and designers with a higher level of abstraction for modeling data, such that the resulting model closely reflects the real meaning of data (Hull and King 1987). One of the first semantic data models was the entity-relationship (ER) model in which the domain being modeled is viewed as a collection of real world entities and relationships among the entities (Chen 1976). Other semantic modeling approaches include Hammer and McLeod's semantic data model and Hull and King's generic semantic model.

Drawbacks of Semantic Modeling Approaches

Semantic models focus primarily on the *structural* aspects of the domain and therefore represent a data-oriented view of the system. From a process-oriented perspective, the *behavioral* aspects of the system refer to the procedures affecting each entity and the transfer of data among entities. This process-oriented view of the system is typically modeled separately using tools such as data flow diagrams, process flowcharts, and state-transition diagrams. The data- and process-oriented views must then be integrated during detailed systems design—a process that can be quite cumbersome, even if CASE tools are used to automate the process (DeChampeaux and Faure 1992; Fichman and Kemerer 1992). This separate modeling of the structural and behavioral aspects was well suited for conventional systems implemented on hierarchical or relational DBMS. In these DBMSs, the database itself reflects only the structural aspects of the domain while the behavioral aspects of the domain are represented in application programs. Although this separation of data from processes, referred to as data independence, has many advantages, it is critical to ensure that the behavioral semantics affecting a database entity are consistently implemented across all applications accessing that entity. For example, if there are four application programs that access an inventory table, and if there is a business rule that a purchase order for an inventory item should be triggered whenever the reorder point for that item is reached, then all four applications must be programmed to reflect that business rule. Given these problems, systems developers are increasingly seeking to seamlessly represent both the behavioral and structural aspects of the domain in one integrated representation (Coad and Yourdon 1991a; Ochuodho 1992).

Data Models for Enterprise Information Needs

McCarthy (1979) demonstrated how the ER approach can be applied to modeling enterprise information requirements in transaction processing contexts. McCarthy (1982) later extended this work and proposed the REA (resources, events, and agents) framework for representing information about economic resources, economic events, and economic agents and relationships among them. The key focus in the REA methodology is on the *events* that occur in an enterprise, the *resources* that are affected by these events, and the *agents* that control and are associated with the events. Thus, using the REA modeling framework, entities in an enterprise-wide information processing context are events such as sales and purchases, resources such as cash and inventory, and agents such as customers and suppliers. Resource and event entities are connected by means of relationships that represent the stock-flow interactions between them, such as a sale event representing an outflow of finished goods that is (eventually) coupled with an inflow of cash. Economic agents and units are represented by means of their participation in and control over events and resources. By focusing on the significant events that occur in an enterprise, the REA approach fosters a focus on business processes.

Although McCarthy's REA approach is a powerful tool for modeling the structural aspects of an enterprise information system, the behavioral aspects must be modeled separately using data-flow diagrams, process flowcharts, and state-transition diagrams. Another drawback of an REA-based enterprise model is that the modeling formalisms used have no direct counterpart in technologies that might be used to implement the model. For example, there is no way to distinguish between resource, event, and agent tables in a relational DBMS—they are all implemented as tables with only the table names conveying their meaning or intended purpose; relational database systems do not allow a grouping or categorization of like tables. Generalization hierarchies, which can be represented in an REA model, also cannot be implemented in a relational DBMS. The limitations of relational database technology also prevent some features of semantic data models from being fully implemented. For example, although many semantic modeling approaches including REA can depict generalization and aggregation hierarchies, most relational DBMS cannot represent nested entities or aggregations of data items that are hierarchically related (Jackson 1990; Ochuodho 1992). As another example, there is no way to distinguish between an entity table and a relationship table in a relational DBMS (Jackson 1990). Relationships among entities depicted in an ER diagram must be inferred by searching for cross-reference keys in the entity tables in a relational implementation.

Object-Oriented Modeling

In the object-oriented paradigm, *objects* provide a unifying metaphor for systems design and development activities from the initial design stage through the implementation stage. Systems analysts, designers, and programmers all focus on a common set of objects and inter-object relationships which results in a smoother and more seamless systems development process (Korson and McGregor 1990). Recognizing these limitations of semantic models such as the ER model, some researchers have sought to equip the ER model with object-oriented extensions, while others have focused on developing "pure" object-oriented modeling approaches. Object-oriented modeling seeks to capture both the structural and the behavioral semantics of the application domain using the unifying metaphor of the object. Monarchi and Puhr (1992) provide a good review of the different object-oriented modeling approaches. Murthy and Wiggins (1993) explored the implications of the object-oriented paradigm for accounting systems and reviewed several object-oriented modeling approaches in light of the needs of accounting systems. Murthy and Wiggins call for further research on object-oriented modeling approaches specifically designed for modeling accounting systems in an enterprise context.

One object-oriented modeling approach that extends Chen's ER model, providing it with object-oriented features, is Gorman and Choobineh's (1991) object-oriented entity-relationship model (OOERM). However, a limitation of OOERM for modeling enterprise information systems is that, unlike REA, it does not explicitly address the unique needs of business event information processing contexts. These object-oriented modeling approaches hold two key advantages over conventional semantic modeling approaches like ER and REA. First, they are capable of representing both structure and behavior in one integrated model. Second, they use a metaphor for modeling (the object) which has a direct correspondence in the technology used to implement the model (object classes in object-oriented programming languages or database systems).

Chu (1992a, 1992b) explored the applicability of object-oriented approaches for accounting systems, but employed the chart of accounts as the basis for constructing object classes. Generalization hierarchies that are implicit in the chart of accounts are easily represented by means of superclass-subclass object relationships. Another advantage of using the chart of accounts as the basis for developing an OO model is that the inheritance concept in object-oriented systems allows more general accounting attributes and methods, such as account number and balance, to be defined at the highest level in an account class hierarchy. However, using the chart of accounts as the basis for an OO accounting model has its limitations. The chart of accounts is essentially an accounting artifact which does not always consider and represent the needs of non-accountants (McCarthy 1982). Entities (objects) of interest which do not have a direct impact on the organization's financial statements would likely be excluded in an OO model based on the organization's chart of accounts. For example, there is no logical place in a chart of accounts to store information about employees and their skills. Thus, given that the chart of accounts does not capture all aspects of the organization's information domain, an OO model based on that chart of accounts will be limited to representing only those data and procedures that have a financial statement impact. As such, a chart-of-accounts based OO model has limited applicability for modeling enterprise-wide information needs.

Kandelin and Lin (1992) present a computational model of an events-based object-oriented accounting information system for inventory management. Their objective was to integrate both data and knowledge representation in an events-based accounting system. As Kandelin and Lin indicate, object-oriented technologies are well suited for integrating data structures along with knowledge represented in encapsulated procedures for each data structure (object). The three main components of the Kandelin and Lin computational model are an event message database system, an accounting report object subsystem, and an accounting intelligence subsystem. However, they do not present a model or a methodology for designing a logical model of an OO events accounting system.

More recently, Verdaasdonk (2003) put forth an object-oriented model for handling *ex ante* information in the context of operations management. Verdaasdonk argues that the REA model focuses only on the modeling of static accounting phenomena and is, therefore, not able to provide relevant *ex ante* accounting information for operations management decisions. Although the object-oriented model proposed by Verdaasdonk has the advantage of integrating both the static (structural) and dynamic (behavioral) aspects of the domain, his model is not intended as an ontology for object-oriented modeling of enterprise systems. Rather, the Verdaasdonk object-oriented model is narrow in scope, aimed exclusively at supporting operations management decisions. What is lacking, then, is a generalized object-oriented model for describing both the structural and the behavioral aspects of accounting and non-accounting phenomena in an enterprise-wide context. In the next section, we describe such an object-oriented model for designing enterprise information systems.

The Object-Oriented REA Model

The OOREA ontology proposed in this paper is essentially an object-oriented extension to McCarthy's REA model. Resources, events, and agents are all viewed as object classes, each with different attributes. Unlike McCarthy's (1982) REA model, the OOREA model shows the interaction among objects in terms of the processes that modify their values. Further, the OOREA model's use of the object metaphor for modeling simplifies the implementation process in object-oriented environments such as Smalltalk, C++, ObjectStore, or Gemstone. An illustrative OOREA model of a retailing enterprise is also presented. This OOREA model is then compared with an equivalent ER model (McCarthy 1979).

The OOREA Model

As noted earlier, a key advantage of McCarthy's REA ontology is the focus on events (business processes), the resources that affect and are affected by events, and the agents who perform and are associated with events. The OOREA model proposed here harnesses this business process focus for object-oriented modeling in an enterprise-wide context. In essence, the OOREA approach involves focusing on the structural aspects of the enterprise's domain in terms of the key business events, associated resources, and agents, through an object-oriented lens. This object-oriented lens necessitates a simultaneous focus on the behavioral aspects of events, resources, and agents, while capitalizing on OO features of inheritance and encapsulation. Recent research by Agarwal and Sinha (2003) suggests that novice users with prior experience in process-oriented modeling approaches found UML to be easier to use than those without such experience. Agarwal and Sinha call for research to enhance the usability of UML diagrams by addressing the close interdependence of class and interaction diagrams. The OOREA model proposed in this paper, which is a variant of UML, represents one possible approach to addressing this issue.

The notational conventions used in the OOREA model, shown in Figure 1, are based primarily on the unified modeling language (UML), proposed by Booch, Rumbaugh, and Jacobson (1999). As in UML class diagrams, object classes are represented by rectangles; the top part of the rectangle specifies the unique name the object class, the bottom part contains the attributes for that object class. Attributes that require simple data types are represented in simple typeface and those attributes requiring complex data types are represented in boldface. For ease of exposition, resource objects are shown in shaded rectangles with shadows, event objects are shown in bold rectangles, and agent objects are shown in shaded rectangles with rounded corners.

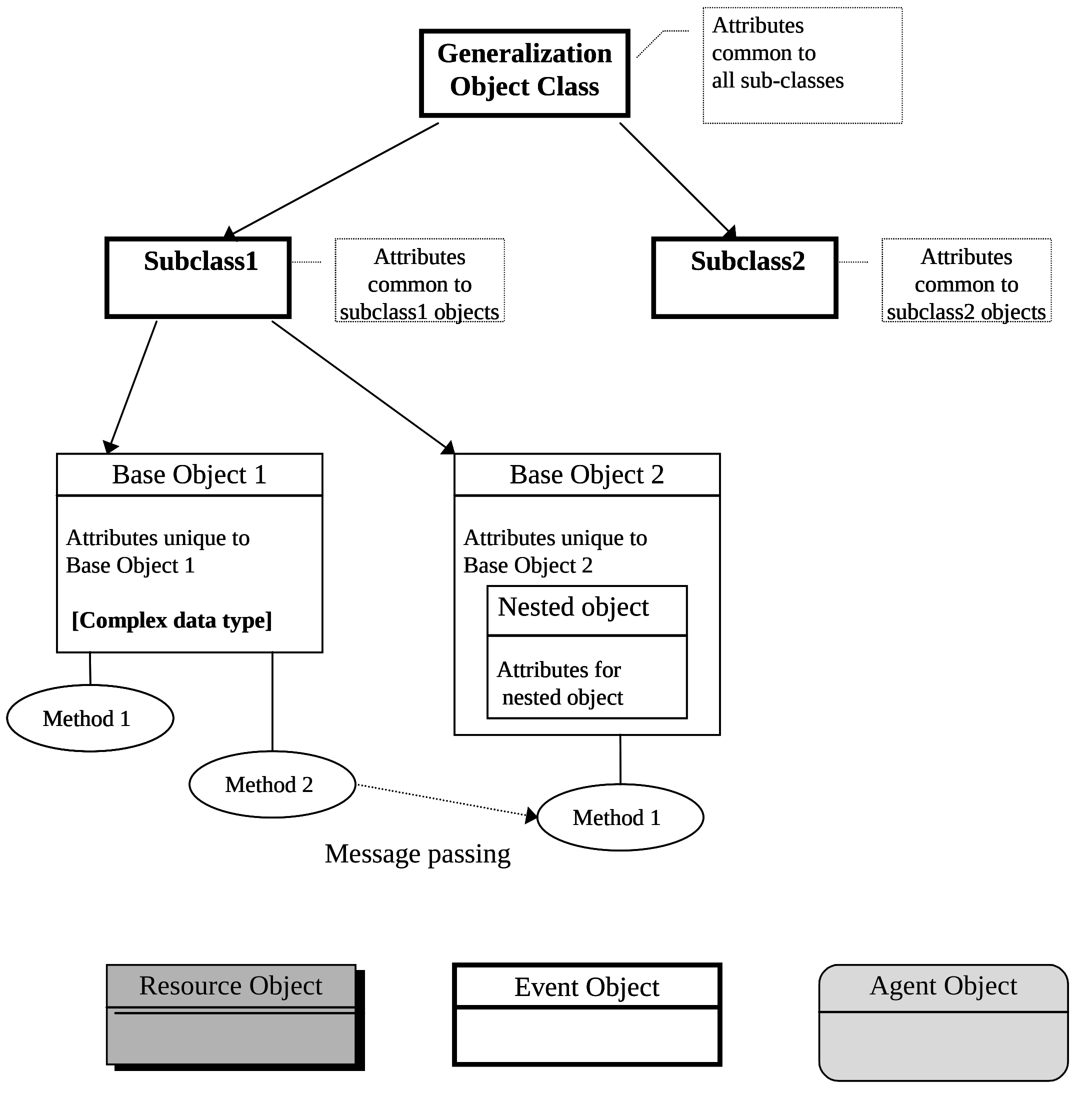
Generalization Object Class
Attributes common to all sub-classes
Subclass1
Attributes common to subclass1 objects
Subclass2
Attributes common to subclass2 objects
Base Object 1
Attributes unique to Base Object 1
[Complex data type]
Base Object 2
Attributes unique to Base Object 2
Nested object
Attributes for nested object
Method 1
Method 2
Method 1
Message passing
Resource Object
Event Object
Agent Object

Figure 1. OOREA Modeling Conventions

The typical application of the UML methodology entails the use of separate diagrams to depict structural and behavioral aspects, i.e., class and object diagrams for the structural aspects, and use-case and activity diagrams for the dynamic or behavioral aspects. As suggested by Dori (2002), the proliferation of diagrams in UML is one barrier to its widespread acceptance. By contrast, the OOREA approach we propose employs a single diagramming convention for depicting all structural and behavioral aspects. Thus, the OOREA approach requires a mechanism for depicting interaction between object classes. Accordingly, one significant departure from UML is in the convention used in OOREA to indicate the methods for each object class. In UML, methods are indicated within the bottom tier of object rectangles. However, this convention makes it difficult to show inter-object communication in terms of the messages passed between specific methods. In our OOREA modeling approach, we employ Gorman and Choobineh's (1991) convention for their object-oriented entity-relationship model (OOERM), in which methods are represented by ovals below the object class. The separate depiction of each method in its own oval makes it easier to show messages passing between methods of interrelated objects. As with OOERM, message passing is indicated with a dashed line; the parameters being passed are indicated in italics next to each dashed line.

The inheritance feature in the object-oriented paradigm naturally supports generalization hierarchies. As shown in Figure 1, generalization classes are shown at a higher level than the specialization classes. There may be any number of levels of generalizations. Each generalization level consists of a *superclass* being decomposed into a number of *subclasses.* Each subclass inherits all of the attributes and methods of its superclass. OOREA also supports modeling of aggregate or "nested" objects. Multiple line items associated with a sale event is an example of a nested object. An association relationship specifies a connection between two independent objects. In an ER model, associations among entities are depicted by means of the diamond symbol connecting the associated entities. In OOREA, association type relationships are indicated by means of a message being passed between the related objects. For instance, the customer and sale object classes would be associated by virtue of sale occurrences, each of which would result in a message being passed from the sale object class to the customer object class with parameters such as the amount of the sale to update the appropriate customer object instance (i.e., customer account in conventional terms).

An OOREA Model of a Retailing Enterprise

This section illustrates an object-oriented REA model of a simple retailing enterprise. The model includes the sales, purchasing, cash receipts, cash disbursements, and other selected functions for a typical retailing enterprise. Both conventional transaction processing capabilities and complex data handling capabilities (such as document images) are represented. Claims such as accounts receivable, accounts payable, and wages payable are represented as attributes of their related agent objects (i.e., customer, vendor, and employee, respectively), and are discussed later in this section. An object class hierarchy for the resulting OOREA model is shown in Figure 2. For ease of exposition, Figure 2 does not show the methods associated with each object. Figure 3, which is a subset of Figure 2, depicts the methods for each object within this topography and identifies the message passing which would likely occur between objects for typical sale and purchase transaction events for a retailing enterprise.

The class object hierarchy depicted in Figure 2 consists of an Information Entity superclass, Resource, Event, and Agent subclasses and 14 base objects. The Information Entity superclass at the topmost level of the hierarchy is defined at a very general level to allow for the representation of any resource, event, or agent about which the organization would like to store information. Attributes and methods defined for the Information Entity superclass are inherited by all subclasses. The 14 base objects, which include 3 Resource objects (Cash, Inventory, and PPEQ), 7 Event objects (Sale, Purchase, Cash Receipt, Cash Disbursement, Employee Service, G&A Service, and Capital Transaction), and 4 Agent objects (Customer, Vendor, Stockholder, and Employee), are identified in Figure 2. The attributes (instance variables) which give uniqueness to each class and object are also depicted.

Subclass objects inherit all attributes and methods of their superclass(es). For example, in Figure 2 all instances of the PPEQ object inherit the Resource Type attribute of the Resource subclass and all four attributes (ID#, name, Description, and current balance) defined for the Information Entity superclass, in addition to the six attributes uniquely defined for PPEQ objects. Similarly, all instances of the Employee object inherit all four attributes defined at the Information Entity superclass level, and inherit the agent type and address attributes from the Agent subclass. Line_Item is a nested object within the Purchase and Sale objects, indicating that each Purchase or sale object instance may have more than one Line_Item instances associated with it. **Invoice Image** and **P.O. Image** are document images and reflect complex data types.

Although the resources, agents, and events shown in Figure 2 all have a financial accounting orientation, it should be noted that the OOREA model could easily represent information about system elements with a managerial accounting orientation. In addition, information about non-accounting resources, events, and agents can also be represented. Examples of managerial and non-accounting resources, events, and agents that do not have any direct bearing on the organization's financial statements are shown in Table 1.

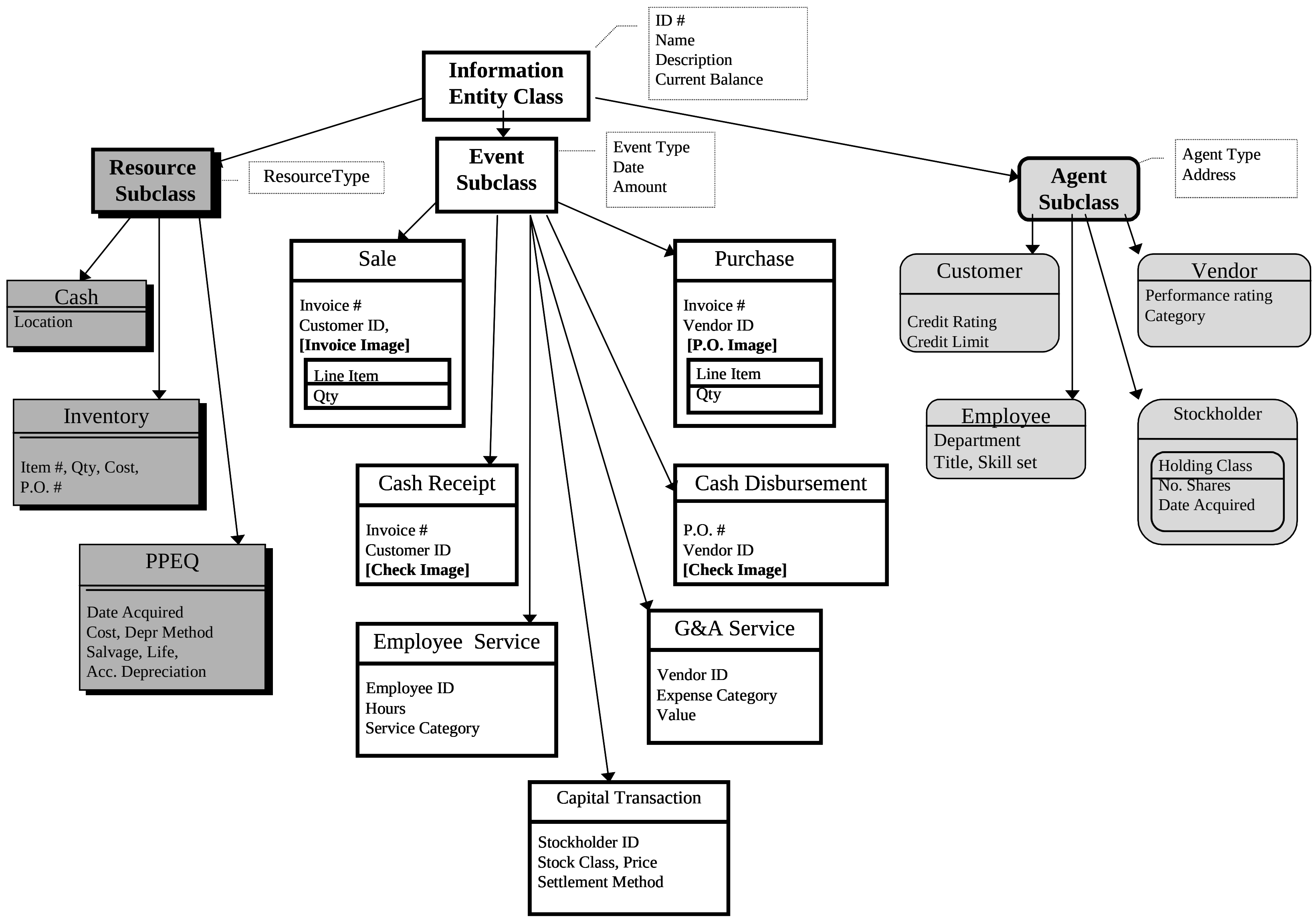

Figure 2. OOREA Object Classes

Table 1. Examples of Non-Accounting Resources, Events, and Agents

Resources	Events	Agents
• Patents	• Customer inquiries/complaints	• Potential customers
• Research and development projects	• Contacts with potential customers	• Potential stockholders
• Leased equipment	• Machine breakdown	• Consultants

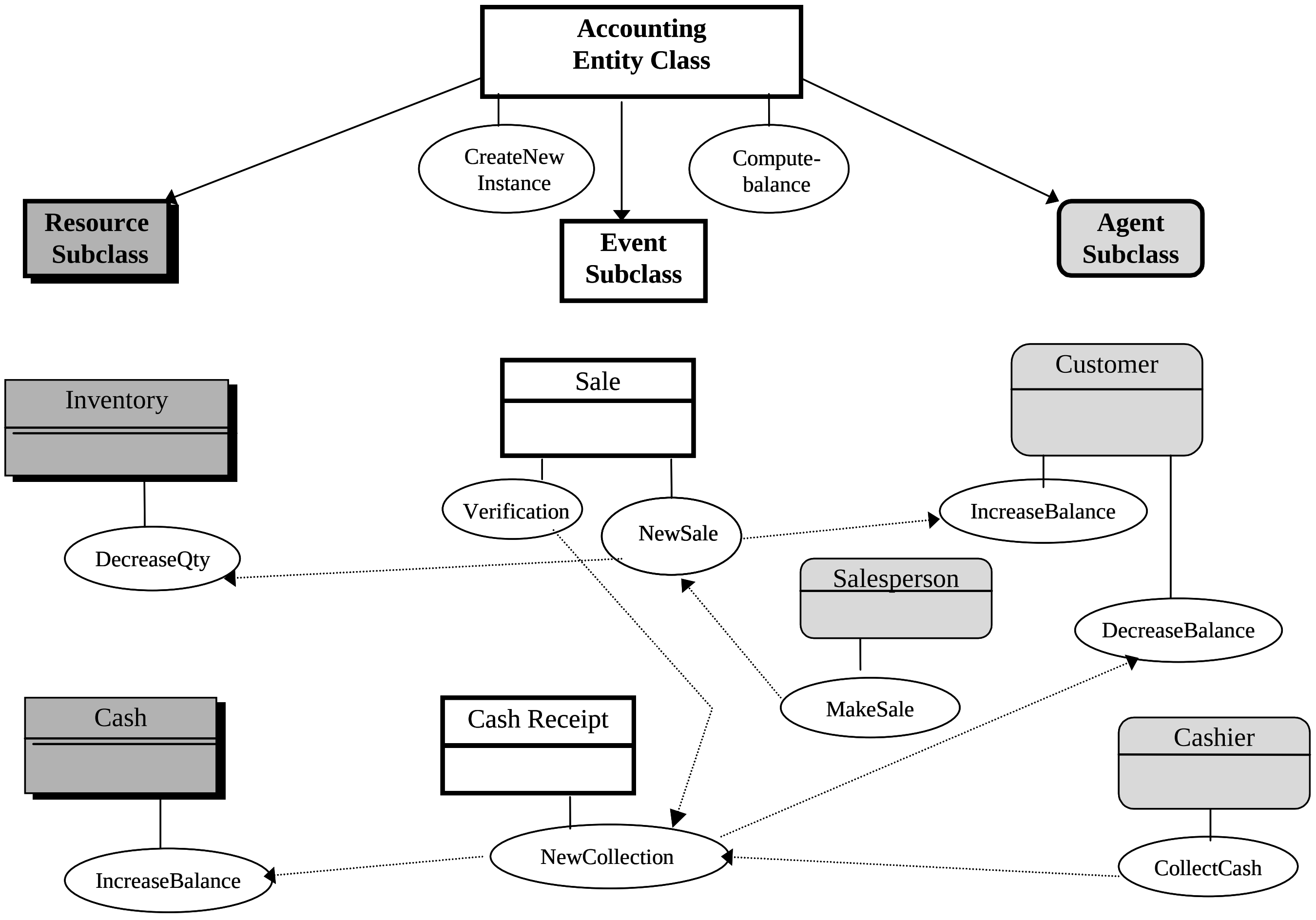

Figure 3. OOREA Methods and Message Passing (for credit sales and cash receipts)

In the OOREA model, events (transactions) are triggered through message passing between related objects, i.e., objects associate (or interact) by sending messages to each other. Each message invokes a method within the target object which causes appropriate actions to occur such as updating a system element. The methods and message passing characteristics for the sales and cash receipts events are presented in Figure 3 and are now discussed. Note that Figure 3 is a subset of Figure 2, showing the resource, event, and agent classes of relevance for sales and cash receipts.

When a sale or cash receipt event occurs, a new instance of the appropriate event object is created and caused to "fire" messages to the related objects. For example, a Sale event causes the following actions: (1) the CreateNew method in the Information Entity Class is invoked which creates and initializes a new instance of the Sale object (referred to as instantiation in OOP), (2) a message is passed to the new sale instance which invokes its NewSale method, (3) the NewSale method passes an IncreaseBalance message to the indicated customer object and a DecreaseQty message to the appropriate inventory object. In the customer object, the Increase Balance method causes the customer balance to be increased to reflect the new sale. In the inventory object, the Decrease Qty method causes the inventory balance to be reduced for the items sold. Similarly, when a cash receipt occurs, a new cash receipt object is instantiated via CreateNew Instance, the NewCollection method is invoked, an IncreaseBalance message is sent to the Cash object and a DecreaseBalance message is passed to the Customer object to activate the appropriate updates to these objects through their respective methods. Note that the cash receipt event receives a message from the sales event, which serves as verification that the prior (required) event did occur. All other events would trigger similar creation, message passing to related objects, and updating activities as those shown in Figure 3.

It should be observed that the Figure 3 OOREA representation reflects and is consistent with the REA concepts of stock-flow relationships and duality relationships. For example, Figure 3 may be viewed as depicting sales and cash receipts as duality-related event set pairs. Within each event set pair the stock-flow relationship is evidenced by the IncreaseBalance and DecreaseBalance methods in the Resource objects which perform the respective increment and decrement roles for the event set pair. For example, in the sales and cash receipts event set pair, the increment role is performed by the IncreaseBalance method in the Cash object and the decrement role is performed by the DecreaseQty method in the Inventory object. The OOREA model presented here represents claims such as accounts receivable, accounts payable, and wages payable as attributes of their related agent objects. As depicted in Figure 3, the accounts receivable claim is maintained as a running balance in the Current Balance attribute of each customer instance. Alternately, this claim could be periodically determined through the ComputeBalance superclass method which, in the case of a customer, would determine the excess of sales to the customer over cash receipts from the customer, less any adjustments or allowances. Balances of other claim types would be similarly determined.

Financial reporting aspects of OOREA are not illustrated in Figures 2 and 3 but could be implemented by means of a GenerateReports event. When periodic reports are desired, the GenerateReports event would send a ComputeBalance message to each object. As a ComputeBalance message is sent, it would be interpreted differently for the various objects (polymorphism) but in general would aggregate the current balance field for each instance of each base object type. As an illustration, for Cash and most base objects, the ComputeBalance method would simply aggregate the Current Balance attributes of each instance of the base object. However, for the Inventory object class, ComputeBalance would first stratify inventory object instances into purchase-quantity layers and then calculate the inventory balance based on a prescribed cost flow assumption. For Event objects, ComputeBalance would aggregate the Current Balance fields for all instances for the current year only.

Comparison of OOREA and ER Models

For the purpose of comparison, we show both McCarthy's (1979) complete ER model of a retailing enterprise in Figure 4, and also an REA model of sales and cash receipts in Figure 5.

The OOREA model in Figure 2 is similar in some respects to McCarthy's complete ER model shown in Figure 4. Both approaches attempt to capture the semantics of the information domain for a retailing enterprise. In the ER model (Figure 4), objects and events are shown as entities and relationships, whereas in the OOREA model (Figure 2), resource entities, agent entities, and events are shown as object classes. Both models depict association type relationships. In the object-oriented model, inter-object associations are indicated by messages being passed between them, whereas in the ER model, the relationship symbol (diamond) depicts associations between entities. Generalization hierarchies are not depicted in the ER model, although the REA model (McCarthy 1982) is capable of depicting generalization hierarchies. In the OOREA model, generalization-specialization hierarchies are depicted by means of superclass-subclass relationships. The OOREA model shows nested objects, as in the example of several line-item instances nested within both the sale and purchase event classes. Neither the ER model nor the REA model is capable of depicting such nested relationships. Further, the OOREA model differentiates between simple and complex data types associated with each object.

A significant point of difference between the OOREA model and the ER model is that the ER model depicts only the structural aspects of the domain. In comparing the OOREA model in Figure 3 with the equivalent REA model in Figure 5, note that the OOREA model indicates the operations affecting each object, that is, the behavioral aspects of the sales and cash receipts domain. The OOREA model also shows the dynamics of the interaction between objects by way of messages being passed between them. The focus of the REA model in Figure 5 is on the nature of the *structural relationships* between resource, event, and agent entities. For example, the sales event is related to the inventory resource and the customer and salesperson agents. The OOREA model in Figure 3 shows these structural relationships and also shows the dynamics of the methods in which they interact.

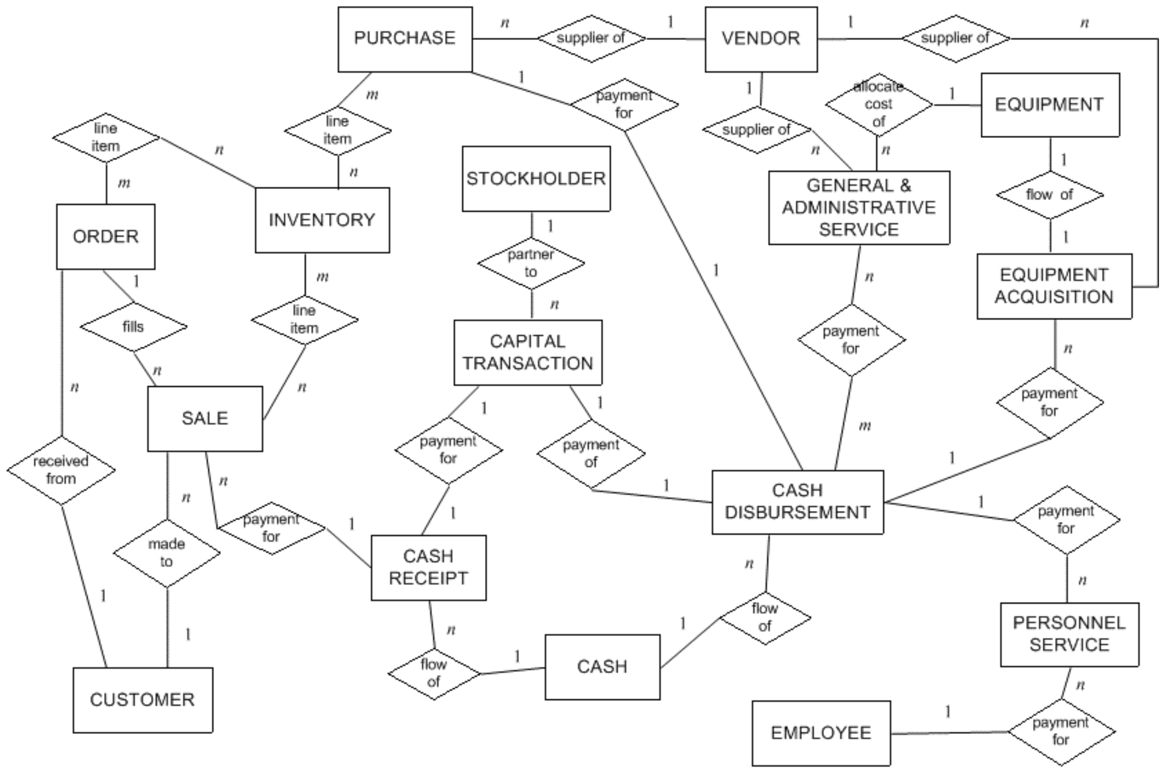

Figure 4. ER Diagram for the Entire Retail Enterprise
(Source: "An Entity-Relationship View of Accounting Models," W. E. McCarthy, *The Accounting Review* (54:4), October 1979, pp. 667-686. Copyright © 1979, The American Accounting Association; used with permission.)

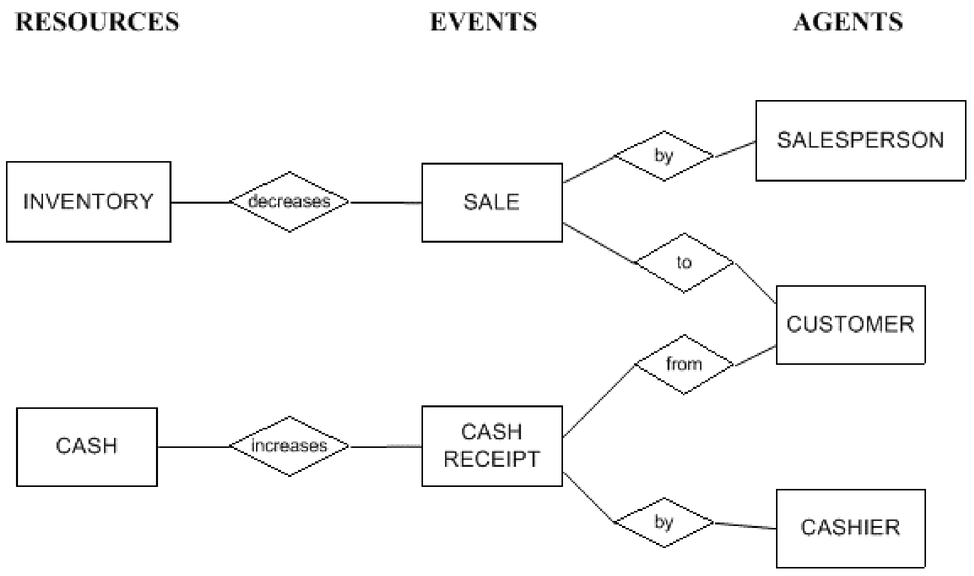

Figure 5. REA Diagram for Credit Sales and Cash Receipts

The ability to represent both the structural and behavioral aspects of the domain in one model provides a number of advantages. First, the semantic expressiveness of the model should be enhanced, since analysts and users work with a single model rather than separate models. (Using the ER/REA approach, analysts and users would work with the ER/REA model for the structural aspects of the domain and then with data-flow diagrams for the behavioral aspects.) Thus, the use of OOREA facilitates cooperation between the business domain experts (end users) and the information technology experts (systems analysts/designers) via the use of a single model and a common vocabulary. With the incorporation of generalization hierarchies via superclass-subclass relationships, the OOREA model should facilitate the proper identification and placement of controls in the system. Specifically, controls that affect all objects within a given superclass-subclass hierarchy should be placed within the highest object in that hierarchy. Finally, the joint modeling of structural and behavioral aspects is essential if the implementation domain is an object-oriented environment. A key feature of OOREA is its use of the object metaphor for modeling, with a focus on the attributes and methods of each object and the superclass-subclass relationships between objects. These constructs (object classes, attributes, methods) correspond directly with the features in all object-oriented environments such as C++, C#, and Java. Object-orientation is thus considered to be a unifying paradigm in which systems analysts, designers, and programmers all use the object metaphor at each stage of the systems development process (Korson and McGregor 1990).

Summary and Conclusion

This paper presents an object-oriented extension to McCarthy's (1982) REA model. Existing OO modeling approaches in accounting and object-oriented extensions to the ER model were reviewed. The advantages of the OOREA model relative to McCarthy's original REA model were discussed. An illustrative OOREA model of a retailing enterprise was presented and compared with an equivalent ER model (McCarthy 1979). The two key advantages of OOREA are (1) the integrated representation of the data- and process-oriented views of the system and (2) the employment of a modeling formalism (i.e., the object), which corresponds directly with constructs in object-oriented technologies, thereby easing the implementation process.

Future research could focus on the implementation of the OOREA model presented here in one or more OO environments. Practical considerations and problems would more likely be revealed as a result of such an implementation. The extent to which the OOREA model is perceived as being more semantically expressive is an issue that can be experimentally investigated. Similarly, whether the use of the OOREA model improves communication between the analyst and the user is another empirical question worthy of investigation. Research is also needed to better understand the information needs of functional areas other than accounting so that integrated enterprise-wide information systems can be designed and implemented using OOREA. While we contend that the OOREA model represents a superior method of identifying objects in a business information processing context, the ability of organizations to adopt the OOREA modeling approach is unclear. Thus, research regarding the potential adoption of OOREA by organizations and the barriers to such adoption would be fruitful. As object-oriented programming languages and object-oriented databases become increasingly available, it seems likely that many business information systems in the future will be developed using an object-oriented approach.

References

Agarwal, R., and Sinha, A. P. "Object-Oriented Modeling with UML: A Study of Developers' Perceptions," *Communications of the ACM* (46:9), 2003, pp. 248-256.

Booch, G. *Object-Oriented Design with Applications*, Benjamin/Cummings, Redwood City, CA, 1991.

Booch, G., Rumbaugh, J., and Jacobson, I. *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA, 1999.

Chen, P. P. S. "The Entity-Relationship Model—Toward a Unified View of Data," *ACM Transactions on Database Systems* (1:1), March 1976, pp. 9-36.

Chu, P-C. "Applying Object-Oriented Concepts to Developing Financial Systems," *Journal of Systems Management* (43:5), May 1992a, pp. 28-34.

Chu, P-C. "An Object-Oriented Approach to Modeling Financial Accounting Systems," *Accounting, Management, and Information Technologies* (2:1), 1992b, pp. 39-56.

Coad, P., and Yourdon, E. *Object-Oriented Analysis*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1991a.

Coad, P., and Yourdon, E. *Object-Oriented Design*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1991b.

David, J. S., Dunn, C. L., McCarthy, W. E., and Poston, R. L. "The Research Pyramid: A Framework for AIS Research," *Journal of Information Systems* (13:1), 1999, pp. 7-30.

DeChampeaux, D., and Faure, P. "A Comparative Study of Object-Oriented Analysis Methods," *Journal of Object-Oriented Programming* (5:1), 1992, pp. 21-33.

Dori, D. "Why Significant UML Change Is Unlikely," *Communications of the ACM* (45:11), 2002, pp. 82-85.

Fichman, R. G., and Kemerer, C. F. "Object-Oriented and Conventional Analysis and Design Methodologies: Comparison and Critique," *IEEE Computer* (25:10), October 1992, pp. 22-39.

Geerts, G., and McCarthy, W. E. "An Accounting Object Infrastructure for Knowledge-Based Enterprise Models," *IEEE Intelligent Systems & Their Applications* (14:4), July-August 1999, pp. 89-94.

Geerts, G., and McCarthy, W. E. An Ontological Analysis of the Primitives of the Extended-REA Enterprise Information Architecture," *The International Journal of Accounting Information Systems* (3), 2002, pp. 1-16.

Geerts, G., and McCarthy, W. E. "Using Object Templates from the REA Accounting Model to Engineer Business Processes and Tasks," *The Review of Business Information Systems* (5:4), 2001, pp. 89-108.

Gorman, K., and Choobineh, J. "The Object-Oriented Entity-Relationship Model (OOERM)," *Journal of Management Information Systems* (7), Winter 1991, pp. 41-65.

Hammer, M., and McLeod, D. "Database Description with SDM: A Semantic Database Model," *ACM Transactions on Database Systems* (6:3), September 1981, pp. 351-386.

Hull, R., and King, R. "Semantic Database Modeling: Survey, Applications, and Research Issues," *ACM Computing Surveys* (19:3), September 1987, pp. 201-260.

Jackson, M. S. "Beyond Relational Databases," *Information and Software Technology* (32:4), May 1990, pp. 258-265.

Kandelin, N. A., and Lin, T. W. "A Computational Model of an Events-Based Object-Oriented Accounting Information System for Inventory Management," *Journal of Information Systems* (6:1), Spring 1992, pp. 47-62.

Korson, T., and McGregor, J. D. "Understanding Object-Oriented: A Unifying Paradigm," *Communications of the ACM* (33:9), 1990, pp. 40-60.

McCarthy, W. E. "An Entity-Relationship View of Accounting Models," *The Accounting Review* (54:4), October 1979, pp. 667-86.

McCarthy, W. E. "The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment," *The Accounting Review* (57:3), July 1982, pp. 554-78.

Monarchi, D. E., and Puhr, G. I. "A Research Typology for Object-Oriented Analysis and Design," *Communications of the ACM* (35:9), September 1992, pp. 35-47.

Murthy, U. S., and Wiggins, Jr., C. E. "Object-Oriented Modeling Approaches for Designing Accounting Information Systems," *Journal of Information Systems* (7:2), Fall 1993, pp. 97-111.

Navathe, S. B. "Evolution of Data Modeling for Database," *Communications of the ACM* (35:9), September 1992, pp. 112-123.

Ochuodho, S. J. "Object-Oriented Database Support for Software Project Management Environments: Data-Modeling Issues," *Information and Software Technology* (34:5), May 1992, pp. 283-307.

Sultan, F., and Chan, L. "The Adoption of New Technology: The Case of Object-Oriented Computing in Software Companies," *IEEE Transactions on Engineering Management* (47:1), 2000, pp. 106-226.

Teorey, T. J., Yang, D., and Fry, J. P. "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model," *ACM Computing Surveys* (18:2), June 1986, pp. 197-222.

Verdaasdonk, P. "An Object-Oriented Model for *Ex Ante* Accounting Information," *Journal of Information Systems* (17:1), 2003, pp. 43-61.