The READY Model: Patterns of Dynamic Behavior in REA-Based Accounting Applications

Dinesh Batra and Thant Sin
Florida International University, College of Business Administration, Miami, FL

Abstract The Resource-Event-Agent (REA) model has gained considerable attention in accounting literature. While REA denotes a data model, which represents only the static aspect of a system, the dynamic aspect has now been introduced as the scenario concept in a recently proposed REA ontology. Using the Unified Modeling Language (UML) sequence diagram—a popular method of showing interactions among objects—and building on the REA framework and the scenario notion, the paper presents the READY model to illustrate patterns of dynamic behavior in accounting scenarios.

Keywords Resource-Event-Agent model, Unified Modeling Language (UML), and analysis patterns

The last four decades have yielded significant developments in the field of Information Systems (IS) in general and Accounting Information Systems (AIS) in particular. Some of these developments have caused a significant shift in the way we conceptualize and represent business, information systems, and accounting phenomena. Two key developments in the area of business, and information systems data modeling are the introduction of the relational model (Codd, 1970) for representing and manipulating business data, and the entity relationship (ER) model (Chen, 1976) for conceptualizing data that can be implemented using the relational model.

In this domain, accounting researchers have made a key contribution. The Resource-Event-Agent (REA) model (McCarthy, 1979, 1982) is not only an innovative way of studying AIS, it also provides fresh lenses with which to view and understand business systems. The REA model is based on a frequently occurring pattern of transactions (economic events) wherein agents inside and people outside of an enterprise (economic agents) exchange things of value (economic resources). McCarthy (2003) reviews how the REA model was assimilated into and included in the AIS pedagogy as a useful framework for understanding. However, one can make an argument that the REA approach, which essentially links business events with IS analysis and design, can even be employed in core MIS courses such as database applications, and systems analysis and design to enhance their usability.

The REA model has gained considerable attention in accounting literature. In 2007, a symposium was dedicated specifically to celebrating 25 years of REA. Several textbooks in AIS (e.g., Murthy & Groomer, 2003) have employed the REA approach as a foundation for teaching accounting topics. The REA approach is not only an intuitive way to render AIS concepts, it can also encapsulate the double-entry bookkeeping approach and hide its unappealing and, at times, myopic features. There is, indeed, empirical support (Dunn & Grabski, 2000) that the REA approach leads to higher task accuracy and is perceived as more expressive when compared to the double-entry approach. Accounting software packages also seem easier to understand when one takes the REA approach. Whether it is a smaller package like QuickBooks (Ivens, 2008) or a large enterprise package like SAP, whose reference models are exemplified by Scheer (1998), it seems that the REA approach is well suited to the understanding of accounting processes and models. The REA model is also found to enhance user comprehension of conceptual models (Poels, 2003).

However, the REA approach denotes only a data model, which is the static aspect of a system. What is not included is how the data is processed. For example, a sale not only represents data, it includes both how the sale actually takes place and the key activities involved. Ostensibly, the dynamic piece—how the data is processed—is missing from the REA model. To fill this void, this paper proposes READY (DYnamic REA), an
extension of the REA model that incorporates dynamic modeling. Given that the REA framework itself is a pattern, READY employs a similar analysis patterns approach to reveal the typical interaction scenarios in accounting applications.

The motivation for our paper stems from the initial work by Murthy and Wiggins (2004) who proposed Object-Oriented REA (OOREA) to extend the REA framework beyond the confines of data modeling. They adopted an object-oriented approach, but did not employ a standard object-oriented language like the UML to show interactions among accounting objects. For example, OOREA lacks the means to show actors (e.g., customers, vendors, and other agents), forms through which the interaction takes place (e.g., order form), and the controller objects that embed various business rules and enforce double entry bookkeeping. We employ the UML sequence diagram, the most popular method of showing interactions among objects, to propose an REA-based approach for modeling dynamic behavior of AIS. Our coverage is more detailed and comprehensive than OOREA, and is based on the extended REA ontology reported in Geerts and McCarthy (2002). The use of UML makes the diagrams amenable for implementation.

The REA Ontology

Geerts and McCarthy (2000) contend that the REA model provides an ontology of enterprise information systems in general, and accounting information systems in particular. They have extended the original REA model (McCarthy, 1982) to build the ontological foundation for representing an AIS (Geerts & McCarthy, 2000, 2002). Figure 1 provides a summary of the REA ontology. Not all of the elements may be relevant in a given system. Key elements are briefly described below.

- **Resource**: something of value. For example, a product or a service is a resource.
- **Event**: an economic transaction. For example, invoicing is an event that involves the sale of products or services.
- **Agent**: an economic representative such as a customer or an employee.
- **Commitment**: an obligation with a certain degree of enforceability. For example, an order is a commitment; it subsequently results in events such as shipments and corresponding invoices.
- **Duality**: the condition resulting from one event requiring a complementary event (the dual) to complete a transaction. For example, an invoice leads to one or more collections.
- **Linkage**: a relationship between a product (or service) and its components such as the bill of materials. This relationship may not be required in an event scenario.
- **Reciprocal**: a complementary relationship wherein one commitment leads to another commitment. For example, an order will eventually lead to a purchase.
- **Association**: a relationship between one agent and another, usually between an external and an internal agent. For example, a salesperson may be assigned to a customer. This relationship may not be required in an event scenario.
- **Reserved**: a commitment to provide a list of products or services, usually modeled as line items for a commitment.
- **Stock-Flow**: the list of products or services in a transaction, usually modeled as line items for an event.
- **Executes**: the materialization of a commitment into an event. For example, an order leads to shipments/invoices.

The relationships *accountability* and *involvement* are both self-explanatory and implicit in a transaction. The relationship *custody* is not a required element for an event scenario, but may be relevant in some housekeeping situations. For example, a laptop computer may be checked out to an employee for use during travel.

In addition to these elements, the extended REA ontology also introduces the notion of *scenarios*, which are abstract representations that describe an enterprise’s strategies or its policies on activities involved in business processes. In object-oriented systems analysis terminology, this is usually described in a use case, one of the most important UML artifacts.

The REA ontology is expressed as a data model. However, its elements are not merely static. An event, for instance, also represents the activities required to complete the transaction. In other words, an event also denotes a scenario. This is better understood by taking an example from the REA ontology.

---

**Figure 1.** Components of the REA ontology.
In Figure 2, the REA ontology is instantiated into an order commitment scenario. A sales order placed by a customer and taken by a salesperson may consist of one or more products, which are recorded in sales order line items. Then the order is filled and products are shipped out to complete the sales order, and the customer is invoiced for the shipment. Each invoice will have a list of products included in invoice line items. Since an order is realized in several shipments, one order may lead to one or more invoices. The product may need to be assembled or manufactured from standalone components or raw materials resulting in a bill of materials relationship.

However, a sales order and a sales invoice also represent scenarios. A sales order scenario may involve static elements such as the SalesOrder, Customer/Employee, Involvement, Product, and SalesOrderLineItem. Similarly, the sales invoice scenario will involve a number of elements. Key scenarios, in general, are likely to involve both event and commitment elements.

The interactions among these elements are not captured in the REA ontology, which is expressed at an abstract level. It would be useful to reveal the nature of the scenarios especially if there are underlying patterns, that is, if we can find interaction configurations that tend to repeat so that learning an interaction in one accounting scenario can help one understand an interaction in another scenario.

In Figure 3, we depict an example, adapted from the textbook by Murthy and Groomer (2003), showing the key data modeling aspects of the revenue side of an enterprise. The dark or shaded side of the relationship represents the ‘many’ cardinality, while the clear side represents the ‘one’ cardinality. The static model depicted using the ER model is, otherwise, self-explanatory and resembles the REA model in Figure 2. However, the model does not depict the dynamic aspects. In other words, the scenarios that describe how the static elements interact are not reflected in the data model. The ER model is not geared to representing the interactions, although some researchers use the class diagram and draw arrows to show the messages among the objects. The most popular method of representing interactions is the UML sequence diagram (Dobing & Parsons, 2006).

**On UML, Sequence Diagrams, and Analysis Patterns**

Typical scenarios in accounting can be determined by examining the revenue cycle, conversion cycle, and expenditure or acquisition cycle, following the business process and enterprise value chain frameworks (McCarthy, 2003). In IS development, a scenario is shown by a use case, which is one of the most popular UML artifacts. A use case shows behavior or functionality under various conditions as the system responds to requests from users (George, Batra, Valacich, & Hoffer, 2007). However, a use case only lists the activities that occur in a scenario, it does not show how objects actually interact to carry out the activities of the scenario; this job is done by another UML artifact called the sequence diagram. The use case and its sequence diagram together show high- and low-level behavior and interaction involved in a business process.

UML is the standard object modeling language sponsored by an open consortium of companies, called the Object Management Group (OMG) (Kobryn, 1999; OMG, 2005). It has been widely used by software developers in object-oriented (OO) software development projects and adopted in many information systems (IS), information technology (IT), and computer science (CS) curricula around the world (Batra & Satzinger, 2006). In a recent study, the class diagram, the use case diagram, and the sequence diagram were found to be those most commonly used by practitioners (Dobing & Parsons, 2008).
The use case diagram is often developed in requirements or use case modeling, while the class and the sequence diagrams are modeled in static and dynamic modeling, respectively (Bolloju & Leung, 2006).

A sequence diagram captures behavior in a scenario by showing the interactions among objects. It is the linkage between the functional requirements captured in written use cases and the analysis classes depicted in class diagram (Ambler, 2004; George et al., 2007). The operations of object classes are identified from responsibilities modeled in sequence diagrams, whereas attributes are derived from domain data models.

Requirements of an IS are defined and organized into analysis models by systems analysts. This is often considered one of the most critical systems development activities because the quality of analysis models often determines the quality of the resulting IS (Bolloju & Leung, 2006). Analysis patterns are designed to facilitate the analysis modeling by enabling the reuse of template solutions that have proven to be useful in relevant contexts. By using template solutions for typical problems, analysis patterns help define software requirements. IS researchers have introduced a number of analysis patterns (Batra, 2005; Coad, 1992; Coad, North, & Mayfield, 1997; Fowler, 1997; Hay, 2006; Scheer, 1998). In AIS, McCarthy (1982) introduced the Resource-Event-Agent (REA) model that can be employed in building enterprise data models for an AIS by following the prescribed pattern of economic transactions.

According to Kodaganallur and Shim’s (2006) taxonomy of analysis patterns, the REA model is classified as an abstract building block that can be specialized and assembled into conceptual models. The REA model’s ability to provide an abstract conceptualization of a business enterprise based on a pattern of an economic event is invaluable in prescribing typical structures and relationships found in an accounting application (Geerts & McCarthy, 2000). An enterprise data model is developed by replacing these structural building blocks with domain-specific items such as product, order, and customer.

The REA model is suitable for developing enterprise information architecture, which specifies the static structure of an AIS; however, it is not suitable for specifying processes that manipulate the data. IS researchers have taken steps to extend the REA model beyond the structural modeling to the dynamic or behavioral modeling, as in Murthy and Wiggins’s (2004) object-oriented REA (OOREA) model. One problem with existing solutions such as the OOREA model however, is the sparse coverage of accounting scenarios. Another problem is the use of an ad hoc-style notation, which can create a barrier to widespread adoption. Therefore, there is a need for analysis patterns that can facilitate dynamic modeling using the well-established UML conventions.

The Generic READY Pattern

In this paper, we develop and present an important extension of the REA model, an extension designed for dynamic modeling, which has been termed READY (DYnamic REA). The READY model provides a pattern that can be used to model commitments and transactions in UML sequence diagrams. Standard UML notations are used in the READY model. We first provide the generic representation of the READY model, which is later instantiated into key scenarios of the revenue, expenditure, and conversion cycles of a business enterprise.

The REA model has a generic representation depicting a data model among the resource, event, and agent objects. In the same spirit, an abstract READY model, which includes key interactions in a transaction scenario, is presented in Figure 4. Key elements of the diagram are:

- the actor who initiates the scenario (the actor is usually one of the economic agents);
- the interface object, usually a form, through which the actor interacts with the system;
- the control object, which controls the message exchanges including the posting of entries to the accounts and which embeds the logic of business policies (e.g., tax or discount calculation), or outsources this logic to the transaction object or other objects; and
- the entity objects for resource, event/commitment, and agent, as well as for line item if a transaction contains more than one resource.

Each transaction follows a predictable pattern of messages. The actor interacts with the interface, and the interface communicates with the control object, which acts as a director managing the exchange of messages and data. The scenario generally starts with a search for information for a resource or an agent. Required information may need to be obtained from one or more entity objects. Once information is displayed via the interface object, the actor may request that a transaction object be created. The transaction object may be required to manage the creation or update of data in the line item object. Generally, more than one line item will be added. Once line items are added, the detailed information of the transaction, including line item details and all related information (e.g., fees, discounts, taxes, etc.) can be assembled to allow the actor to review the transaction. In the final step, the transaction is confirmed and committed to by posting any updates to relevant objects. This process is usually performed by the control or the transaction object and concluded with a confirmation of the completed transaction. Overall, the process usually follows a pattern that follows this sequence: search, select, create transaction, add line items, review transaction, and confirm transaction.
The static model of the revenue cycle is shown in Figure 3. The starting point in the revenue cycle is the sales order, which is a commitment to provide the customer with a number of products or services. This commitment can then be realized in one or more shipments/invoices. The issuance of an invoice increases the receivables. A dual process—collection—results in both a reduction of receivables and an increase in cash. Products may need to be manufactured from raw materials converted into work-in-process, using resources such as machine operations and workers, in the conversion cycle.

The sequence diagram shown in Figure 5 provides an example of a high-level interaction view of the sales order process. We assume that the customer initiates the commitment (e.g., in a web- or phone-based purchase), although an employee (e.g., sales representative) may initiate the commitment on behalf of the customer. The first activity is a search for and perusal of products. When the customer is ready to purchase, the sales order object is created. Messages here do not explicitly indicate what data is being input. For example, for the commitment, a sales order number will be generated, and the...
sales date will be recorded. Line items are then added for products selected by the customer.

When line items have been added and the order is complete, the customer is ready to review the order. In addition to sales order and line items, information such as taxes, fees, shipping charges, discounts, and estimated shipping date will also be displayed. The determination of these figures can involve tedious logic. The sales order object may be responsible for calculating the figures, or the control object may ask an entity object such as sales order and discount tables (not shown in the figure) to provide the figures. Although order and line item details are saved into respective database entities, the confirmation of the order will not lead to the update of inventory or receivable. When the sales order is confirmed, the changes are committed to the respected entity objects, and the customer is sent a copy of the sales order. The sales order transaction has the search, create transaction, create line items, review, and commit activities.

The READY Model of Event: Sales Invoice/Shipments

The sales order, therefore, does not usually result in a monetary exchange, although in simple situations, such as retail stores that charge the customer when the items are checked out, there is an immediate monetary exchange since there is no difference between an order and an invoice. In general, however, a sales order leads to shipments of products, which then results in invoices. Since backordering is usually allowed, one sales order can result in multiple sales invoices. The customer is gen-
generally not invoiced until a shipment is made, so invoice and shipment are similar events. Figure 6 shows the sequence diagram for the invoice/shipment scenario.

The invoice/shipment process starts with the search for an open order to be filled. The order information is obtained so that ordered line items can be viewed. Then an invoice is opened by creating the invoice object. Line items are filled, and the quantity on hand for products is adjusted. When the invoice is completed, it can be reviewed by obtaining the data from the invoice, line item, and product objects. At this stage, additional information such as a discount, taxes, and shipment charges are calculated based on information in the sales order and according to company policies. The commitment to the transaction updates various entity objects. Specifically, the customer balance is updated to reflect an increase in receivables.

The quantities in the invoice line item may not match those in the order line item because of the shortage of inventory. This will invoke backordering, which has not been included in this sequence diagram. The backordering process can be considered a separate revenue sce-
The backordering scenario is straightforward since the quantity backordered is a calculated entry.

**The READY Model of Duality: Collections**

Payments received from a customer are collections, which decrease the amount of accounts receivables for the customer and increase cash amounts for the company. The collection process is the dual of the sales invoice, which increases the accounts receivables and decreases the inventory. In general, a collection can be applied to one or more invoices. Conversely, an invoice can be settled using one or more collections. In a sense, the line item concept can be extended to the collection scenario to record how much of a given collection is apportioned to a given invoice. Queries and programs can ensure that a collection is applied to the oldest invoices first. The company may have other collection policies such as incentives for paying early in the form of discounts, and disincentives for paying late in the form of penalties. Beyond that, the collection scenario is routine. Using the customer identification, open invoices can be listed chronologically, and the customer payment applied to one or more invoices. Then, the collection transaction is reviewed and confirmed. The sequence diagram in Figure 7 illustrates the collection scenario, which has the search, create transaction, update line items, review, and commit activities.

**Figure 7.** Sequence diagram for collection scenario.
The collections scenario can become quite simple in the case of a cash sale because the exchange of products/services and payment occurs simultaneously. In fact, a separate scenario is not required because the collection operations can be included in the invoice since there is no waiting period as in the case of account receivables.

**READY Scenarios for Conversion Cycle**

For a business enterprise that manufactures or assembles the products it sells, the conversion of raw materials or components into finished products usually involves work-in-process inventory. A finished good may be assembled from components, or manufactured from raw materials, using machine operations and/or workers. The labor input eventually leads to payments in the forms of wages and salaries. A product can alternatively be purchased and then resold, as is the case with several retail products, in which case, the expenditure cycle addresses the procurement of the goods.

**TheREADY Model of Work-in-Process Inventory**

When a finished good is assembled or manufactured, two main costs need to be tracked. The costs of components or raw materials are the material cost. The costs incurred because of the operations to be completed result in employee wages. Each company may have its own algorithm for estimating and measuring these costs.

Product specifications need to be retrieved to determine raw material, labor, and machine time requirements of a product. Availability and information of different types of machine operations and classes of labors are also recorded in separate objects. A detailed schedule of machine time and workers may be added if required. Figure 8 shows the sequence diagram for the production or conversion scenario.

**READY Scenarios for Expenditure Cycle**

The expenditure cycle is almost a mirror image of the revenue cycle discussed above. Sales order, invoice/shipment, and collection from customer in the revenue cycle have respective counterparts in the expenditure cycle. For example, the complement of sales order from the revenue cycle is purchase order from the expenditure cycle, the complement of sales invoice is receipts, and the complement of collections is payments. Since the revenue cycle has been illustrated in a fair amount of details, the expenditure cycle is exemplified by just one scenario. The purchase order scenario is demonstrated in the following section.

**TheREADY Model of Commitment: Purchase Order**

Just as a sales order constitutes a commitment from a customer, a purchase order embodies a commitment by the company to purchase products or services from one of its vendors or suppliers. The purchase order scenario starts with searching for goods, components, or raw materials that need to be replenished from respective vendors. A purchase order is then created, and line items are added. After all items have been added, the purchase order is reviewed, and confirmed. The confirmation saves the purchase order transaction and the line item details. Accounts payable and inventory level are updated upon receipt of a vendor’s invoice or shipment. The purchase order scenario will lead to the receipts (or purchase invoice) scenario when the goods are received. In turn, this will lead to the payment scenario. To avoid redundancy, we do not provide sequence diagrams of expenditure-related scenarios.

**Conclusion**

Some other accounting scenarios can also be modeled using a similar approach. Returns and exchanges can be modeled as extensions of regular invoice transactions. One can also include the purchase of fixed assets and their depreciation. The purchase of a fixed asset is like any purchase transaction. Depreciation is a straightforward adjustment that employs an algorithm, which can easily be programmed into relevant control objects.

Another accounting scenario is payroll. Each payroll remittance can be considered as a transaction involving a number of line items such as the various kinds of deductions, which may include overtime, FICA, Medicare, tax withholdings, pension plan deductions, and medical and other insurance deductions.

Some transactions are straightforward enough so as not to warrant inclusion. For example, a transfer of funds from one bank account to another is a simple scenario when expressed as a sequence diagram. Similarly, maintaining time sheets for employees or creating a budget are also simple scenarios.

The paper has shown a number of sequence diagramming patterns; other scenarios such as fixed assets and payroll are expected to follow similar structures. These structures have been shown at a high level of abstraction to allow specialization and assembly into useful analysis models. Thus, there may be certain omissions, which can be clarified if the diagrams are detailed. For example, the sales as well as purchase transactions can be extended to include the role of employees.

The READY model has predictable activities: search, select, create transaction, add line items, review transaction, and commit transaction. This underlying pattern can
be found in the key revenue, expenditure, and conversion accounting cycles. By providing patterns of dynamic behavior of accounting scenarios, the READY model has been shown to be a valid and useful extension of the REA model.

**Author Bios**


Thant Sin is a doctoral student in the MIS program at the Florida International University. He completed his MBA with IT concentration from the International University of Japan in 2004. He has presented at various conferences such as *ICIS SIG-ISCoRE 2006, AIS SIGSAND 2007*, and AMCIS 2006 and 2007.
References


