# Modeling Business Enterprises as Value-Added Process Hierarchies with Resource-Event-Agent Object Templates

**Guido L. Geerts, Assistant Professor of Accounting**
**William E. McCarthy, Arthur Andersen Alumni Professor of Accounting**

*Department of Accounting, N270 North Business Complex*
*Michigan State University*
*East Lansing, MI 48824, USA*

*E-mail: Geerts "23358MGR@msu.edu" -- McCarthy "21277WEM@msu.edu"*
*Telephone: 517-355-7486 Fax: 517-432-1101*

***ABSTRACT**: Applying object technology to the task of modeling the value-added processes of a business enterprise is a task that should be examined both conceptually and practically. This paper does both, but its main theme is a conceptual reliance on a standardized object template -- the REA (resource-event-agent) model -- at various levels of abstraction as that template is used to model the economic activities of an enterprise. Deployment of REA concepts in business object design and implementation is a semantic strategy for increasing reusability and interoperability. We explain the components and use of REA models in the context of a simple example, and we discuss also its predictable patterns of implementation compromise. The paper finishes with a discussion of the adaptation of various object-oriented analysis and design techniques to the task of REA modeling of enterprises.*

***KEY WORDS**: REA model, enterprise information architecture, object analysis and design patterns, accounting systems, value chain*

## 1. Introduction

The enterprise information architecture of most businesses is composed primarily of data that concerns the input and output of various economic resources into a chain of value-adding processes or activities (Porter, 1985). For example, cash is exchanged for raw materials and labor, those materials and labor are converted into finished goods, and then the finished goods are exchanged for cash again in a cycle that (hopefully) produces more money in the end than was used at the beginning. A successful company or entrepreneur repeats this cycle many times a year or month in an effort to turn an initial outlay of cash (initial financing) into a surplus (profit) where expenditures are exceeded by revenues or where the enterprise's initial set of resources is exceeded in value by its ending set of resources.

The primary information system for tracking these chains of value-added activities in enterprises has traditionally been the firm's accounting system, an information architecture based upon bookkeeping principles originally promulgated over 500 years ago by a Franciscan monk -- Luca Pacioli -- in Venice. In its original form (which is actually the way in which most accounting instruction still takes place both in the USA and in the world), the Pacioli model of double-entry

bookkeeping (Geijsbeek, 1914) forces a very narrow filter upon a very rich data environment for the specific purpose of supporting the preparation of periodic measures of net worth and net income as expressed solely in monetary terms. The development and dissemination of double-entry bookkeeping in 15th and 16th century Europe was a circumstance of both improved technology (the ability to use negative numbers and the printing press for example) and a radically different environment for economic development (the trading ventures to the East and to the Americas). Pacioli's model certainly stands as one of the stellar achievements of the Renaissance. In the succeeding five centuries, his methods have been enhanced and augmented (especially with regard to internal cost accounting), but the basic set of ideas retains both its essential structure and its preemptive call on the object categories used to type economic data in companies (Dunn and McCarthy, 1992). Proof of this preeminence is the very significant role still played by both general ledgers and accounting feeder systems based upon subsidiary ledger structures (such as accounts-receivable, accounts-payable, job-costing, and payroll) in most modern companies. As noted by a number of forward-thinking accountants (Andros *et al*., 1992; Elliott, 1992; Fisher, 1994), the dominance of these information structures is clearly dysfunctional in modern commerce. However, their primacy in the accounting software market continues, although there are signs that information architectures with a more semantic orientation have evolved slowly and are starting to gain market share (McCarthy, 1995, Cherrington *et al*., 1993 ).

It is interesting to note that although improved technology and a different economic environment were the clear catalysts for the emergence of double-entry accounting in the 15th century, these same environmental forces are often resisted vigorously by modern accountants as they struggle to preserve the technology of 15th century Venice in the modern world. With the technological shift to more semantic information architectures and object-oriented programming and with the commercial economic environment shift to "virtual enterprises" that concentrate on their own core competencies first and outsource other processes to downstream and upstream partners, the marketplace should expect to see different ways of tracking economic phenomena in the environment of the firm (i.e., new ways of "doing accounting" that make it less insular and more integrated). For the short term however, what we see primarily instead are a spate of "advanced technology" general ledgers and job-costing systems. In analogous terms, such advanced systems are a lot like ornithopters -- the attempts of more than 100 years ago at heavier-than-air flight where designers simply took an old model (how birds fly by flapping their wings) and tried to make it work with man-made technology. The breakthroughs that culminated with the Wright brothers successful flights at Kitty Hawk in 1903 started occurring when people began to attack the problems of manned flight with a clean conceptual slate, unencumbered by how the task was accomplished with only muscle power. Ironically, some of the first designs for ornithopters were constructed by Leonardo Da Vinci -- a Renaissance contemporary and collaborator of Pacioli. In a certain sense, all-encompassing general ledger systems that use multitudes of coded accounts (sometimes exceeding half a million) are the ornithopters of the 1990s -- designs whose time has passed, but whose continued "kludge" existence is partially enabled by technology and whose owners are under the illusion that conceptual re-orientation is unnecessary. Reengineers often refer to the presence of such artifacts as "paving the cow-paths."

In this paper, we have a number of purposes that relate to the design issues discussed above, especially as those design issues relate to the deployment of object-oriented technology in modern business organizations. Our first purpose is to introduce to the object-oriented implementation

community a semantic framework for conceptualizing the data that tracks economic phenomena in an enterprise. This semantic framework is called the REA accounting model -- a framework for building accounting systems in a shared data environment that has been the subject of considerable conceptual and empirical research since its publication in *The Accounting Review* in 1982 (McCarthy 1982; Geerts and McCarthy 1994, 1995). As might be expected from the tone of our introductory comments, REA (**R**esource-**E**vent-**A**gent) accounting systems do not use double-entry artifacts as essential primitive elements. Instead they use object conceptualizations (such as economic events, economic exchanges, and value chains) derived with the same abstraction methods that gave rise to the object orientation paradigm. In Section 2, we illustrate the basic ideas of REA by applying them to a sample described enterprise. Our purpose in doing this is to relieve readers of the burden of researching and reading all of the normative REA research work as it has been applied in the fields of database design, artificial intelligence, and software engineering. Having familiarized readers with our basic ideas, we use Section 3 to consider two matters: (a) the correspondences of certain REA principles with the ideas of object-oriented methodologists like Coad and Jacobson, and (b) the possibilities for building and using both individual object-oriented software design patterns and a unifying framework (Gamma *et al*., 1995) grounded on REA templates and our own experiences with implementation heuristics. Based upon our own knowledge of the content of corporate data files and of corporate data warehouses and based additionally on empirical research like that of David (1995), we estimate that instantiations of the object patterns associated with full REA modeling could account for as much as 50-60 percent of normal corporate data stores. We finish the paper by speculating on possible research directions for work that combines semantic economic models with object orientation.

## 2. An REA modeling example

### *2.1 Sy's Fish*

We begin our exposition of REA object modeling with a simple commercial example called Sy's Fish. This example is based on an actual company, but its structure has been greatly simplified for use in our explanations. The paragraphs below explain Sy's business.

> *Sy's Fish* is a family-owned distributor of seafood. From humble beginnings, Sy has expanded rapidly into multiple cities, and he provides his base of restaurant customers with over 50 types of fresh fish. Each location or store can carry all types of seafood, but they usually specialize in local favorites. Fish are purchased from local fishers, cleaned at the store, marked up outrageously, and then delivered to restaurant customers. Luckily, because of all the good-health publicity of fish and because of Sy's sterling reputation for quality and service, customers are willing to pay almost anything for fish with his name on it. All stores are very successful at present.
>
> Customers are allowed to buy on credit, and all pay on the last day of the month. Most employees are generalists who can perform many duties such as purchasing, cleaning, and delivering fish. Employees fill out time cards fortnightly upon which they may note the percentage of time devoted each day to buying, cleaning, and selling fish. One employee at each store is designated as the boss, and he or she simply manages the input-processing-

output of the fish. There are also a few other non-generalist employees at each store (such as cashiers).

Sy's also possesses a fleet of trucks. Painted with the firm's Poseidon logo in blue and white, the trucks are used to bring fish from the docks and to deliver fish to the restaurants. Both the truck and the employees involved in each purchase and sale of fish are noted. All trucks are leased on yearly contracts, and lease payments are made monthly. Cash receipts and disbursements are made to/from one of the multiple checking accounts of the firm.

Again, this enterprise description is abbreviated. There are other possible phenomena (such as advertising and rent expenditures or the payment of taxes) that ought to be included, but being simple with these descriptions allows us clarity in our explanations. REA methods can be scaled up to include all types of economic activity.

### *2.2 The basic template -- Resource-Event-Agents*

A core concept in enterprise modeling of commercial activity and in microeconomics is the idea of an economic exchange -- a requited set of transactions where the enterprise gives up control over some resource (a decrement or give) in order to gain control over some other resource (an increment or take). Examples of simple exchanges might include: (1) a revenue process where inventory is decremented and cash is incremented or (2) an acquisition process where cash is decremented and supplies are incremented (we use the terms exchange, process, and activity to mean the same thing). REA modeling views all exchanges as two mirror-image economic events connected by a duality relationship that links the give and take of the exchange. The term REA comes from the object pattern of each event -- an *economic **R**esource* flows in or out of the enterprise in an *economic **E**vent*that has both an inside *economic **A**gent* and an outside *economic **A**gent*.

An example of a fully instantiated REA template is shown in entity-relationship (Chen, 1976) form in Figure 1 which models the economic activity in *Sy's Fish* of leasing the trucks used to transport today's catch. The decrement event in this exchange is the monthly cash disbursement made by one of Sy's employees (a cashier) to the truck vendor; the increment event is the yearly lease contract negotiated with the vendor by a buyer. In REA modeling, the connections between resources and events are termed **stock-flow** relationships, and the connections between events and agents are called**control** or **accountability** relationships. The connection between give events and take events is one of the central ideas of REA modeling, and it is called a **duality relationship**.

### *2.3 Economic event templates at different levels of abstraction*

In the middle of Figure 2, an instantiated REA template is shown at a higher level of abstraction as a**process** or **activity** where the decremented resource is shown as an input and the incremented resource as an output. When all duality relationships are fully specified for an enterprise, the entrepreneurial rationale of its owner or manager (who are presumed to be *homo economicus*) is laid bare. No money is spent or any other resource consumed unless an identifiably more valuable resource is acquired in return. Taken as a whole, duality relationships

are the glue that binds a firm's separate economic events together into rational economic processes, while stock-flow relationships weave these processes together into an **enterprise value chain** (Porter, 1985; Geerts and McCarthy, 1994) or **scenario** (Geerts, 1993). In its most general form, a value chain (as shown at the top of Figure 2 ) is a purposeful set of economic exchanges where an initial outlay of cash is successively converted into some types of more valuable intermediate resource and then finally converted back to cash.

Value chain processes can be decomposed into subprocesses multiple times before an enterprise modeler finds the level at which it is appropriate to explode into a full set of matched REA patterns. A working heuristic that gives an approximate start for deciding object tracking is to choose the level at which decision makers need to plan, control, and evaluate economic events and then go no lower (Hollander *et al*., 1995). Full REA decomposition leads to a process structure of the firm shaped like a tree with only the leaf nodes fully exploded to object patterns, although it will be clear from discussions later in the paper that enforcing all duality links at the disaggregated object level cannot usually be done unless the enterprise has a very simple traceability structure. One must instead construct a process tree where many of the branch nodes have only partial patterns of REA objects specified for resource acquisition and subsequent consumption with the rest of the objects specified at lower levels.

Choosing the appropriate level at which to use the object templates is a difficult analysis process. However, REA modeling posits that it is possible to explode fully in every case at the leaf node level; it just doesn't always make cost-benefit sense to establish a measurement system to do so (see Grabski and Marsh (forthcoming) for a good example of following the patterns down to very minute levels). Predictable implementation compromises (McCarthy and Rockwell, 1989) of REA patterns will be discussed in a later section of this paper, but they are certainly one of the most promising directions for application of object-oriented technology to the task of constructing enterprise information architectures. A good example of an implementation compromise is shown at the bottom of Figure 2 where an economic event (in this case, a purchase) is decomposed to the task level (Burch, 1994, chap. 10). Tasks in REA analysis are, by definition, compromises to full specification (that is, they are economic events where an analyst doesn't try to specify full patterns). Their usefulness in building an information architecture for an enterprise is difficult to assess generally, but at the process leaf level their enumeration can be useful in integrating workflow management and activity-based-costing (ABC) analysis into those architectures.

Looking at Figure 2 from top to bottom, one can see REA information architectures at various levels of abstraction, and concomitantly, one can envision how these architectures are designed:

a. First of all, a corporate chain of value-added processes (Porter, 1985) is specified in very general terms. These high-level processes are then divided into subprocesses until the lowest level at which management needs to plan, control, and evaluate is reached.

b. Second, each process at the lowest level is exploded to illustrate in object fashion its decrement and increment events along with their flow of resources and their internal/external agents.

c. Third, if necessitated by implementation and measurement considerations, some economic events are subdivided into tasks which are economic occurrences in time that do not have to adhere to the full pattern of REA exchanges.

d. Lastly, in an augmentation process not discussed or illustrated here, other object data types are added to the accountability infrastructure described above. Such additional objects might include those of a non-economic nature or those dealing more with hypothetical data types or opportunity costs (Geerts and McCarthy, 1994).

Ideally, the information architecture design process proceeds top down. However, the strong typing and structuring of the REA model actually allows construction to proceed at the middle or bottom levels first.

Figure 3 illustrates what an REA enterprise value chain might look like for our very simple example of *Sy's Fish*. Within each process, only the give (-) and take (+) events are shown along with their duality relationships. In narrative terms, Sy's "entrepreneurial script" proceeds as follows:

> Sy uses cash from initial financing to acquire labor and trucks. His people use their own labor, his trucks, and cash to acquire and transport fish. Workers then use additional labor and the purchased stock to produce cleaned fish. Finally, Sy's employees use labor, the trucks, and the cleaned fish to acquire cash from customers, some of which is used to help repay the initial financing. At a more general level, the labor of managers (and perhaps other employees) is used to facilitate and supervise the overall set of buying, cleaning, and selling activities for each store.

The process hierarchy for Sy's Fish would have a root level process with cash in and cash out, thus representing the long-term behavior of the firm at a very abstract level. A second level would have three leaf processes (financing, payroll and truck acquisition) and one branch process (store supervision and facilitation). The buying, cleaning, and selling processes would be leaf nodes off of the store processing.

Figure 4 shows Sy's value chain in a slightly different fashion that emphasizes the driving definitional rationale for value-added processing. "Value" means value to the firm's customers, and in the final analysis, all economic exchanges or processes in a firm must be evaluated in light of their contribution to customer value. For such analysis, it is useful to think of the enterprise's final product as consisting of a portfolio of attributes, each of which customers value and are thus willing to pay for. In the case of Sy's Fish, this portfolio consists of the cleaned fish, the location of the fish (delivered via truck to the restaurant), the reputation of the fish, and the potential service that comes with the fish if needed. Reputations usually cost money (in advertising, patents, or quality control for example), as does service potential (such as being able to deliver on short notice or replace substandard catch without question). However, the rational economic entrepreneur is always willing to pay that money (the give in an upstream exchange) if it is exceeded in value-added to the customer (the take in the same exchange).

As Figure 4 illustrates in very abstract terms, an enterprise is constantly cycling through its overall value-added processing and turning cash into more cash (making a profit). In Porter's (1985) strategic terms, the company should (1) look at its overall process structure at various levels of abstraction, (2) decide where its core competencies lie by analyzing which of its processes work most efficiently (by producing output with less input) or most effectively (by producing an differentiated output), and (3) manage most carefully those processes associated with their core competencies. Companies like *Sy's Fish* gain their competitive advantage with a differentiated product and service, so he needs to pay special attention to the purchase, cleaning, and delivery of fish. Other processes can be outsourced or less carefully managed. For example, it is apparent that Sy has made such a decision with regard to his trucks, thinking that his leasing company can manage the purchase, maintenance, and management of these resources better than he can.

### *2.4 Full REA modeling, interoperability of object components, and implementation compromises*

In papers that discusses the possibilities for intensional reasoning with REA-modeled economic phenomena, Geerts and McCarthy (1992a, 1995) champion the notion of **Full-REA Modeling** (or as it is called there *epistemologically adequate* enterprise schemas). In very simple terms, this notion means that there are decided benefits to an enterprise information architecture that uses the REA event pattern and its process level abstractions in a repeated top-down fashion. One of these benefits is due to the wide applicability of pattern-matching procedures on such a repeated-pattern architecture. This enables characterization of procedural business definitions (such as how to materialize and value claims) at a relatively high level of abstraction. With ad hoc enterprise information models, such procedures are simply not as applicable, and definitions tend to be single case programs.

The notion of full REA modeling has a related benefit that is especially applicable in an object-oriented environment-- it enables and enhances interoperability. At the process level of REA -- exploded to include matched give-take object patterns as portrayed in Figure 2b. -- business activities are highly congruent, both within and between firms. This means that the various acquisition cycles of a particular firm (for labor, for raw materials, for capital assets, etc.) all look like each other, and indeed, like all other cycles (revenue, conversion, logistics, etc.) in both the same company and other companies. This does not mean that all enterprises have the same information architecture; the idiosyncratic mixing and matching of economic activities in a particular company is what gives that firm its distinctive competitive advantages. Additionally, there are ample opportunities at the subprocess or task level of an REA architecture to tailor an object schema to fit a particular method of doing business. Thus, we see the opportunities to be great for enhanced interoperability in an REA environment, but we do not believe that its consistent repeated use of a single object pattern leads to monolithic implementations, a criticism sometimes heard of software packages like SAP that have similar design philosophies to REA (Semich, 1995).

One of the important lessons we have learned from years of adapting REA conceptual structures to actual implementation platforms (like files, network databases, relational databases, logic programming, and frame-structured representations) is that the patterns of implementation compromises necessitated by such adaptation become predictable (McCarthy and Rockwell, 1989). As we have mentioned previously, we believe that object-oriented technology holds great

promise in this area because it will allow the programming associated with these compromises to become both reusable and transparent. Some of the most common REA implementation compromises are illustrated with *Sy's Fish* examples in Figure 5 and explained below.

a. Figure 5a shows a compromise that occurs in REA modeling when an exchange only involves parties internal to the enterprise **and** when the two halves of the exchange (give and take) are the same level of granularity. When this occurs, there is no reason to model both events independently as they are absolutely congruent. A good example of this is an issue of raw materials from stores into manufacturing (McCarthy, 1982). At the implementation level, the congruent events are folded into each other, so Sy (for example) would only track his cleaning events once.

b. Figure 5b illustrates in a variety of ways what is certainly the most prevalent type of REA compromise in common legacy-type file systems for commercial enterprises. Temporal aggregation means that objects representing occurrences in time are folded into more stable objects (usually representing people, things, or types). For example, instead of keeping a record of individual sales, common marketing or accounts-receivable packages for *Sy's Fish*might aggregate the effect of sales onto either the resource in the REA template (sales per week or month for a product), or the external agent (customer monthly sales or outstanding balance), or the internal agent (salesperson weekly sales total). Less commonly, the aggregated temporal effect of sales could be tracked on a upstream or downstream process in the value chain (aggregate sales due to a certain promotional effort or marketing campaign). Impounding the logic of temporal aggregation procedures in object design patterns (Gamma*et al*., 1995) will be an important part of adapting object-oriented REA systems to the task of wrapping legacy systems (Winsberg, 1995).

c. Figure 5c illustrates with elements of *Sy's Fish* revenue cycle the possibilities for trading off some REA declarations (the explicit representation of an object or a relationship between objects) for procedures. For example, in many cases the relationship shown in dotted lines between the "restaurant" agent and the "cash receipt" event could be replaced at the implementation level by a procedure that circles clockwise back through the "sale" event to identify when needed the external agent for "cash receipt." In another example that actually uses elements of the temporal aggregation heuristics mentioned above, *Sy's Fish* might decide not to maintain the explicit duality link between "cash receipt" and "sale," choosing instead to aggregate the event effects over time and maintain those totals in the "restaurant" object. Such a compromise decision is a common feature (called "balance-forward") of legacy accounting systems, and again, understanding it well is a key to adapting object technology to working with those packages. Conceptual aspects of procedural-declarative tradeoffs in REA systems are discussed extensively by Geerts and McCarthy (1995).

The compromise patterns and examples illustrated above are not exhaustive. For example, Activity-Based Costing (ABC) systems are hybrid versions of REA models where procedural tradeoffs are made for many declarative links based upon similarity of event occurrence patterns (Geerts and McCarthy, 1992c). We intend in future work to encapsulate these ABC compromises and others into object design patterns (Gamma *et al*., 1995). Readers should not also presume that

all implementation compromises lead to less objects. For example, adaptation of REA to accommodate certain types of long-term claims processing (McCarthy, 1982; 1984) like stocks and bonds would lead to the basic patterns being enlarged (again in a predictable manner).

This discussion of implementation compromises concludes our survey of REA work that was conducted in the context of the *Sy's Fish* example. We move next to a discussion of how these ideas reflect similar object-oriented work by authors such as Coad, Jacobson, and Gamma *et al.*.

## 3. The REA model and object-oriented analysis/design methods

### *3.1 REA implementation platforms -- databases, knowledge-bases, and object-oriented systems*

Until now, the REA Framework has primarily been used for design of accounting systems in a shared database environment (Cherrington *et al*., 1993; Hollander *et al*., 1995). Although relational database technology affords REA systems a robust implementation platform (allowing integration and flexibility in information retrieval), information technology has actually been a major constraint on taking full advantage of REA's capabilities. As described by Geerts (1993, pp. 150-2), a considerable amount of domain specific knowledge is lost during the mapping process when relational databases are used as implementation vehicles. The hierarchic and pattern-oriented descriptions of accounting phenomena are scattered over different tables and are no longer visible to the user. Additionally, procedural abstractions (such as being able to use set difference operations to calculate some claims) are lost in the details of the implementation (such as matching posted keys or identifying tuples with null values). Relational technology simply does not allow one to exploit REA's structural knowledge after implementation occurs, thereby reducing its applicability to domain-specific analysis and some heuristic design guidance.

Geerts (1993) and Geerts and McCarthy (1992a, 1995) explore the use of knowledge-based systems to overcome these restrictions. Knowledge technology enables both the explicit recording of hierarchical intensional structures and reasoning with them. Once these structures are in place, additional concepts (like the accounting definitions of claim, asset, cycle, etc.) may then be characterized in terms of REA primitives, and the implemented system will rely on these definitions to materialize conclusions. We foresee a major research opportunity here in developing a general accounting and economic phenomena framework consisting of REA-based definitions that may be shared and reused by many enterprises. However, for reasons of software reusability plus embedded support for both structuring and procedural abstraction, we believe that object-oriented tools may allow us and others to pursue this goal more readily than the knowledge-based tools (like Prolog) that we have been using thus far. Reusability is a key issue in object solutions for both analysis work as well as design work. Analysis Patterns as described by Coad (1995) clearly illustrate the reusability of analysis efforts, while Gamma *et al*. (1995) describe the reusability of design efforts. Both Coad (1995) and Jacobson *et al*. (1995) address the ideas of structuring and behavioral abstraction. In the sections below, we relate the ideas of these authors in a preliminary way to REA analysis, design, and implementation issues.

### *3.2 Coad patterns for analysis and behavioral abstraction*

Coad (1995, p.xiv) describes patterns as something observed from something in actuality, as plans rather than specific implementations, and as templates to be followed during construction of a system. Briefly, they are blueprints providing practical and repeatable "how to" advice helpful for building object models. Clearly, important similarities as well as differences exist between Coad's patterns and the REA model as discussed in the previous section of the paper. These similarities and differences plus the implications of adapting his ideas to modeling economic phenomena with REA patterns are discussed next.

The generalized forms of the REA framework (McCarthy, 1979, 1980, 1982; Geerts and McCarthy, 1994) were derived by semantic abstraction (Chen, 1976; Smith and Smith, 1977) of actual economic transactions and by analysis of abstract accounting theories whose terms resembled the derived primitives. Additionally, REA modeling has been touted as a good blueprint for automated support of semantic database design (McCarthy and Rockwell, 1989), so it does seem that the model fits Coad's definitions for pattern use. Actually, the REA model may be considered as a constellation of integrated patterns (a kind of meta-pattern) in the Coad sense, because some of its major relationships (such as duality, stock-flow, control) are similar to Coad patterns (Transaction--Subsequent-Transaction, Transaction--Transaction-Line-Item, Participant--Transaction) for transactions which he recommends using in an integrated fashion.

Some significant differences do exist between REA models of enterprise economic phenomena and Coad transaction patterns. The REA patterns are first order models grounded in accounting and microeconomic theory, and they have well-defined design heuristics and implementation compromises associated with them. Coad's example patterns on the other hand are more generic and more wide-ranging. In the future, we expect to look at his ideas as we try to impart advice to enterprise information architects on the problem of adding non-economic objects to the REA accountability infrastructure.

As expected by object-oriented analysis, Coad describes behavioral abstractions relevant for each of his patterns. Although recognized as being important in McCarthy (1982) and as being the subject of some very general advice in Gal and McCarthy (1986), behavioral abstractions have been largely ignored for REA patterns. One of our current research efforts is to find relevant behavioral abstractions for each of the REA patterns as well as for REA-specific definitions of accounting and economic concepts. The generic patterns described by Coad are an excellent starting point for such efforts, although REA behavioral abstractions will be clearly different because of their economic specificity and their concomitant availability of domain heuristics. A good example of such adaptation is described in Geerts (1995) for the REA concept of claims. Claims are imbalances in duality relationships, and examples of behavioral abstractions (interactions) described by Coad for the highly similar but more generic *Transaction--Subsequent-Transaction* pattern are *How-Many* and*Calc-Over-Subsequent-Transactions*. Adapting and defining the relevant range for applying these behavioral abstractions involves a precise matching of different cardinality patterns corresponding to different business rules (for example, do we make just cash sales, do we allow installments, etc.).

### *3.3 Use Cases of Jacobson*

Currently, the REA model provides domain-specific guidance in determining the information structure (static data model) of an enterprise. The behavioral emphasis posed by the object-oriented approach implies a research challenge to expand these horizons. Currently, we are looking at the integrated use of REA domain analysis and Jacobson's Use-Case analysis (Jacobson, 1992; Jacobson*et al*., 1995). As suggested by Booch (1994, p.158), both approaches may considered as complementary. A Use Case is *a behaviorally related sequence of transactions* (Jacobson, 1992, p.127). For example, the task level diagram in Figure 4 could have be defined as four Use Cases: (1) Buy Fish Decision, (2) Actual Fish Purchase, (3) Fish Transportation, and (4) Fish Preparation. For each of the Use Cases, analysts and designers look for stereotypical patterns, for how events are related, and for how the Use Case affects objects. Each of the Use Cases is like a separate object capable of managing the variety of possible states and state transitions. In actuality for the *Sy's Fish* example, tasks such as "Load Fish" and "Drive Truck to Store" could have been considered as types of **Economic Event**, each of which would be subject to REA specifications. Instead, we decided to gather transportation information only at a higher level of abstraction. However, the Use Case provides us a stereotypical behavioral pattern that a transaction may go through. Each of the tasks may trigger a change in the "Transportation Activity' state. Jacobson et al. (1995) discuss at length how Use Cases help in reengineering. The extent to which REA value chain analysis (as discussed in Section 2) may benefit from integrated application with Use Cases needs to be explored further. However, the real challenge is to find domain-specific abstract Use Cases. Stated differently, we have to look for behavioral abstractions which hold for REA accounting and economic analysis. For example, what similarities exist between purchase orders and sales orders, what similarities exist between Use Cases dealing with different instances of duality relationships, etc.

### *3.4 Design patterns of Gamma, Helm, Johnson, and Vlissides*

Reusability of design efforts may be accomplished at different levels of abstractions ranging from idioms to frameworks with design patterns lying somewhere in between. The design patterns of Gamma *et al.* offer solutions applicable to many different applications not just the tracking of economic phenomena, but the implementation of an REA system may rely greatly on some of these generic patterns. However, what is even more meaningful to advances in our work is the idea of building an REA system **Framework.**

Gamma et al. (1994, p.26) describe a *Framework* as "a *set of cooperating classes that make up a reusable design for a specific class of software,."* and we believe that Frameworks may be the key solution to make REA accounting operational. As it stands today, the design of such systems relies inordinately on heuristic guidance not embedded (and hence not reusable by anyone except the human expert) in any software. The design of enterprise accounting solutions could benefit from a framework that supports the objects and interactions among objects which express the core (declarative and procedural) REA knowledge. Currently, we are building such a framework called FREACC (FRamework for REA ACCounting), and we hope to embed in it much of the analysis and design guidance (as both structural and behavioral abstractions) that was outlined in Section 2 of this paper.

## 4. Summary

This paper was designed first to bring the body of research associated with REA modeling to practitioners concerned with business object design and implementation. Most moderately complex suites of accounting software (i.e., client-server level) do not support reusability, interoperability, and portability very well, because they are based on a non-semantic model of enterprise business (double-entry accounting) that works well only for simple companies in manual and non-integrated environments. When business activities and organizational forms become complicated, object technology provides a platform for controlling complexity. However, for this technology to work well, its essential components must be based on object models of enterprise economic activity that have multiple levels of abstraction and integrated semantic patterns. The REA model fits these specifications very well, and we hope that readers of this work will become interested in our other empirical and normative work (available on request). A secondary purpose of the paper was to speculate on possible new directions in research that combines the ideas of REA modeling with object orientation. In the past, we have implemented and conceptualized most REA systems with database (Geerts and McCarthy, 1992b) or AI technology (McCarthy, 1987)., and most commercial implementations have used traditional file structures (Cherrington *et al*., 1993). However, this will clearly change because of the object model's embedded abilities to support structural and procedural abstractions. With the help of others, we hope to expand the model's basic components and definitions and to illustrate a wider range of its applicability.

## 5. References


Andros, D.P., J.O. Cherrington, and E.L. Denna. 1992. Reengineer your accounting the IBM way. *Financial Executive*. July/August. 28-31.

Booch, G. 1994. *Object-oriented analysis and design*. Benjamin Cummings, Redwood City, CA.

Burch, J. G. 1994. *Cost and management accounting: A modern approach*. West Publishing, St. Paul, MN.

Chen, P.P. 1976. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*. March. 9-36.

Cherrington, J.O., W.E. McCarthy, D.P. Andros, R. Roth, and E.L. Denna. 1993. Event-driven business solutions: implementation experiences and issues. *Proceedings of the Fourteenth International Conference on Information Systems*. Orlando. 394.

Coad, P. (with D. North and M. Mayfield). 1995. *Object models: Strategies, patterns, & applications*. Prentice-Hall, Englewood Cliffs, NJ.

David, J.S. 1995. An empirical analysis of REA accounting systems, productivity, and perceptions of competitive advantage. Doctoral dissertation, Michigan State University.

Dunn,C.L. and W.E. McCarthy. 1992. Conceptual models of economic exchange phenomena: History's third wave of accounting systems. *Collected Papers of the Sixth World Congress of Accounting Historians.* Kyoto, Japan. Volume I. 133-164.

Elliott, R.K. 1992. The third wave breaks on the shores of accounting. *Accounting Horizons*. 1-21.

Fisher, J.S. 1994. What's ahead in accounting: The new finance. *Journal of Accountancy*. August. 73-76.

Gal, G. and W.E. McCarthy. 1986. Operation of a relational accounting system. *Advances in Accounting*. v.3. 83-112.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1995. *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley, Reading, MA.

Geerts, G. 1993. Toward a new paradigm in structuring and processing accounting data. Doctoral dissertation. Free University Brussels.

Geerts, G. 1995. The semantic modeling of accounting phenomena, Working paper, Michigan State University .

Geerts, G. and W.E. McCarthy. 1992a. The extended use of intentional reasoning and epistemologically adequate representations in knowledge-based accounting systems. *Proceedings of the Twelfth International Workshop on Expert Systems and Their Applications*. Avignon, France. EC2: 321-32.

Geerts, G. and W.E. McCarthy. 1992b. Database accounting systems. *IT and accounting: The impact of information technology*. B.C. Williams and B.J. Spaul (eds.). Chapman & Hall. 159-183.

Geerts, G. and W.E. McCarthy. 1992c. The cost revolution from a data modeling point of view. Paper presented to the Congress of the European Accounting Association, Madrid, Spain. April.

Geerts. G. and W.E. McCarthy. 1994. The economic and strategic structure of REA accounting systems. Paper presented to the 300th anniversary program, Martin Luther University, Halle-Wittenberg, Germany. September.

Geerts. G. and W.E. McCarthy. 1995. Augmented intensional reasoning in knowledge-based accounting systems. Paper submitted to *Journal of Information Systems*.

Geijsbeek, J. B. 1914. *Ancient double-entry bookkeeping*. Scholars Book Co., Houston.

Grabski, S.V. and R.J. Marsh. Forthcoming. Integrating accounting and advanced manufacturing information systems: An ABC and REA approach. *Journal of Information Systems*.

Hollander, A. S., E. L. Denna, and J. O. Cherrington. 1995. *Accounting, information technology, and business solutions*. Richard D. Irwin, Chicago, IL.

Jacobson, I. 1992. *Object-oriented software engineering: A use case driven approach*. Addison-Wesley. Reading, MA.

Jacobson, I., M. Jacobson, and A. Jacobson. 1995. *The object advantage: Business process reengineering with object technology*. ACM Press. New York.

McCarthy, W.E. 1979. An entity-relationship view of accounting models. *The Accounting Review*. October. 667-86.

McCarthy, W.E. 1980. Construction and use of integrated accounting systems with entity-relationship modeling. in P. Chen, ed. *Entity-Relationship Approach to Systems Analysis and Design*. North-Holland. 625-37.

McCarthy, W.E. 1982. The REA accounting model: A generalized framework for accounting systems in a shared data environment. *The Accounting Review*. July. 554-578.

McCarthy, W.E. 1984. Materialization of account balances in the REA accounting model. Paper presented to the annual meeting of the British Accounting Association. Norwich, England. April.

McCarthy, W. E. 1987. On the future of knowledge-based accounting systems. The D.R. Scott Memorial Lecture Series. The University of Missouri. 19-42.

McCarthy, W. E. 1995. The evolution of accounting systems. Paper presented at Erasmus University, Rotterdam, The Netherlands. September.

McCarthy, W. E. and S.Rockwell. 1989. The integrated use of first-order theories, reconstructive expertise, and implementation heuristics in an accounting information system design tool. *Proceedings of the Ninth International Workshop on Expert Systems and Their Applications*. Avignon, France. EC2: 537-548.

Porter, M.E. 1985. *Competitive advantage*. The Free Press, N.Y.

Semich, J. W. 1995. C/S manufacturing: Build, buy, or reengineer? *Datamation*. September.

Smith, J.M. and D.C.P. Smith. 1977. Database abstractions: Aggregation and generalization. *ACM Transactions on Database Systems*. June. 105-133.

Winsberg, P. 1995. Legacy code: Don't bag it, wrap it. *Datamation*. May.

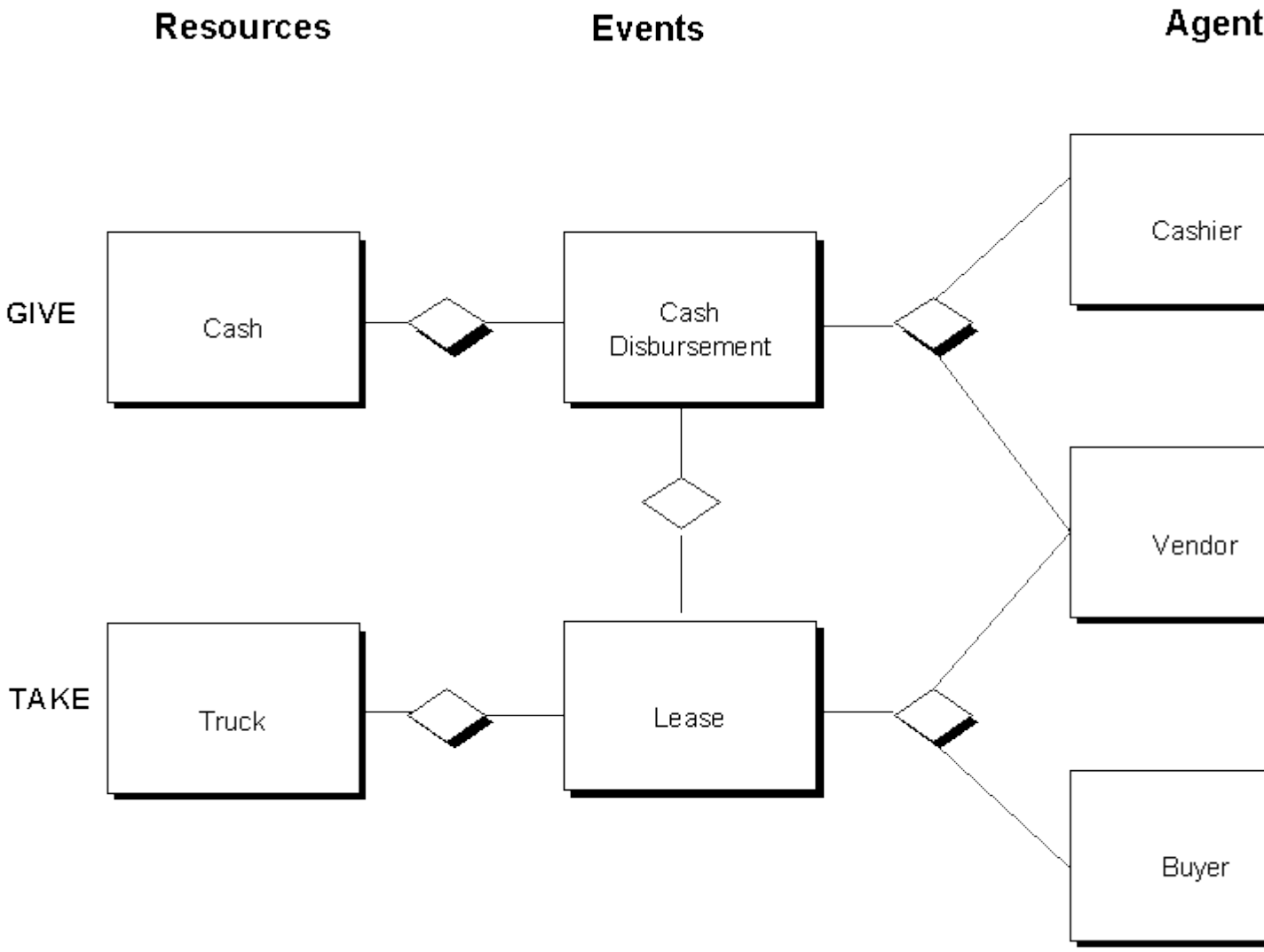
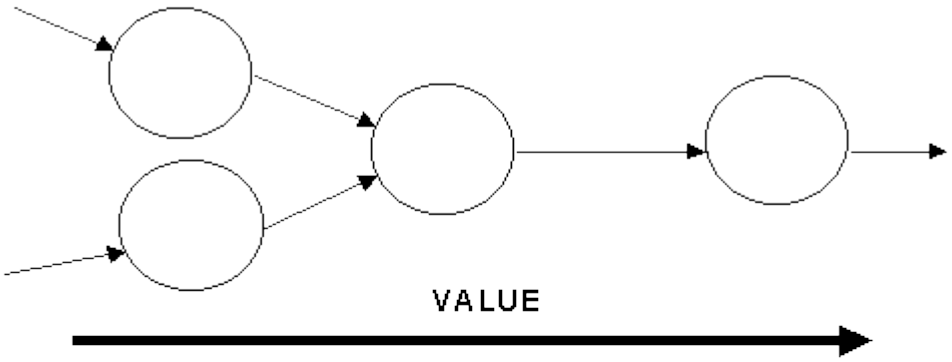
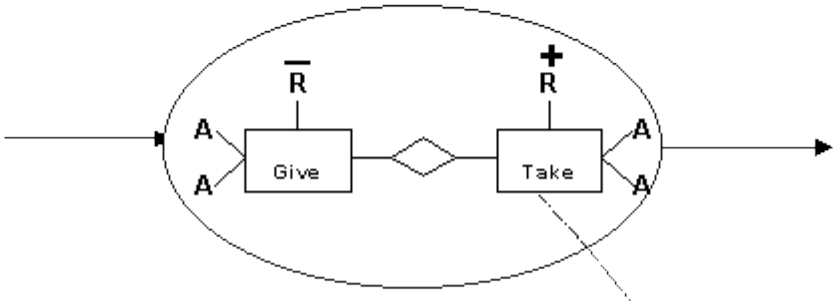


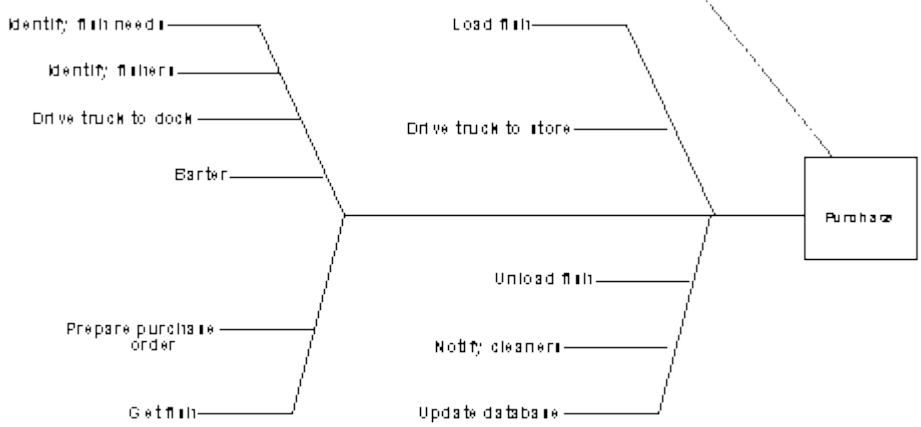

Figure 1 – REA Instantiation




a. Enterprise Level




b. Process Level Exploded to E-R Structure




c. Task Level

Figure 2 -- Different REA Abstraction Levels

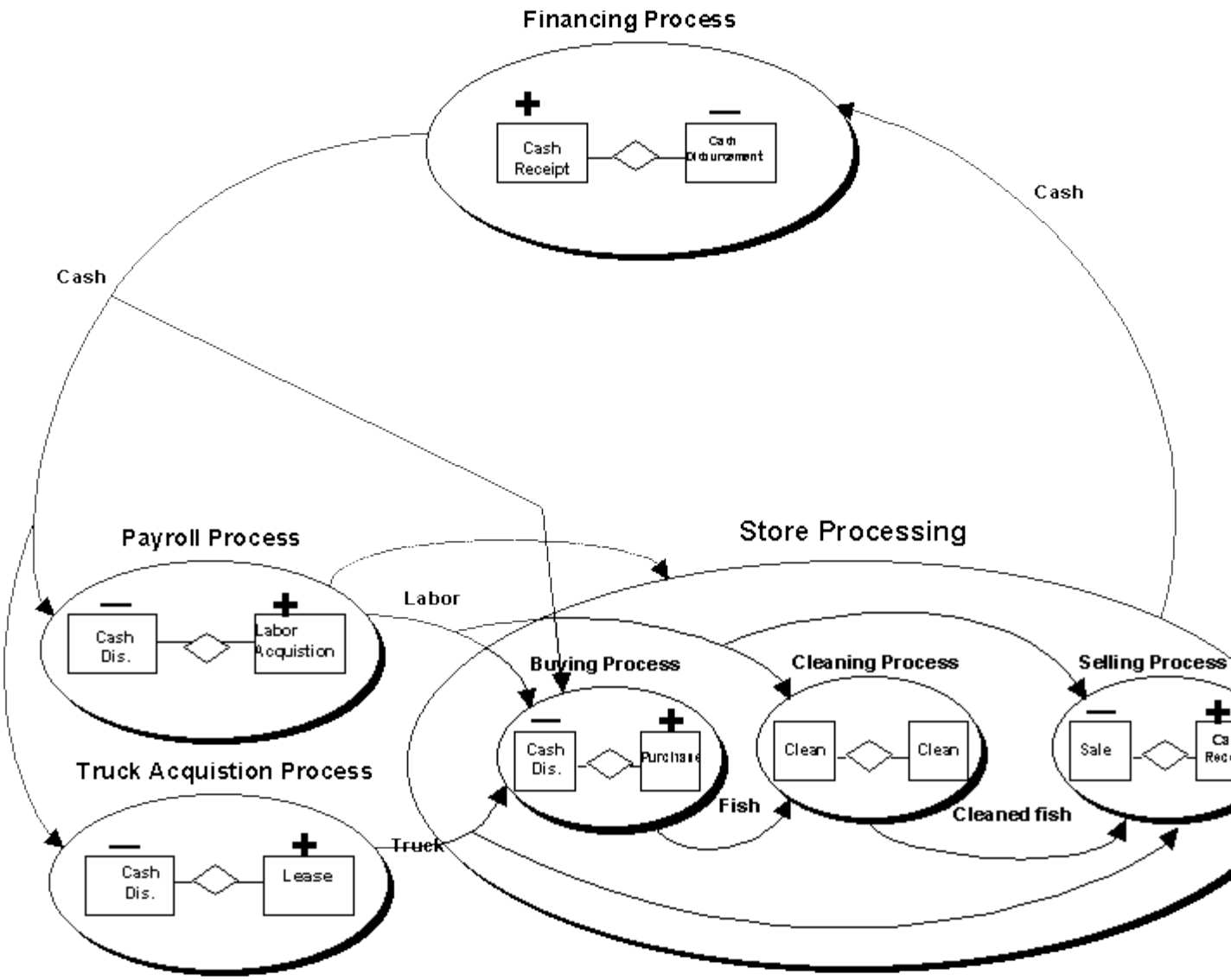



Figure 3 -- "Sy's Fish" Value Chain

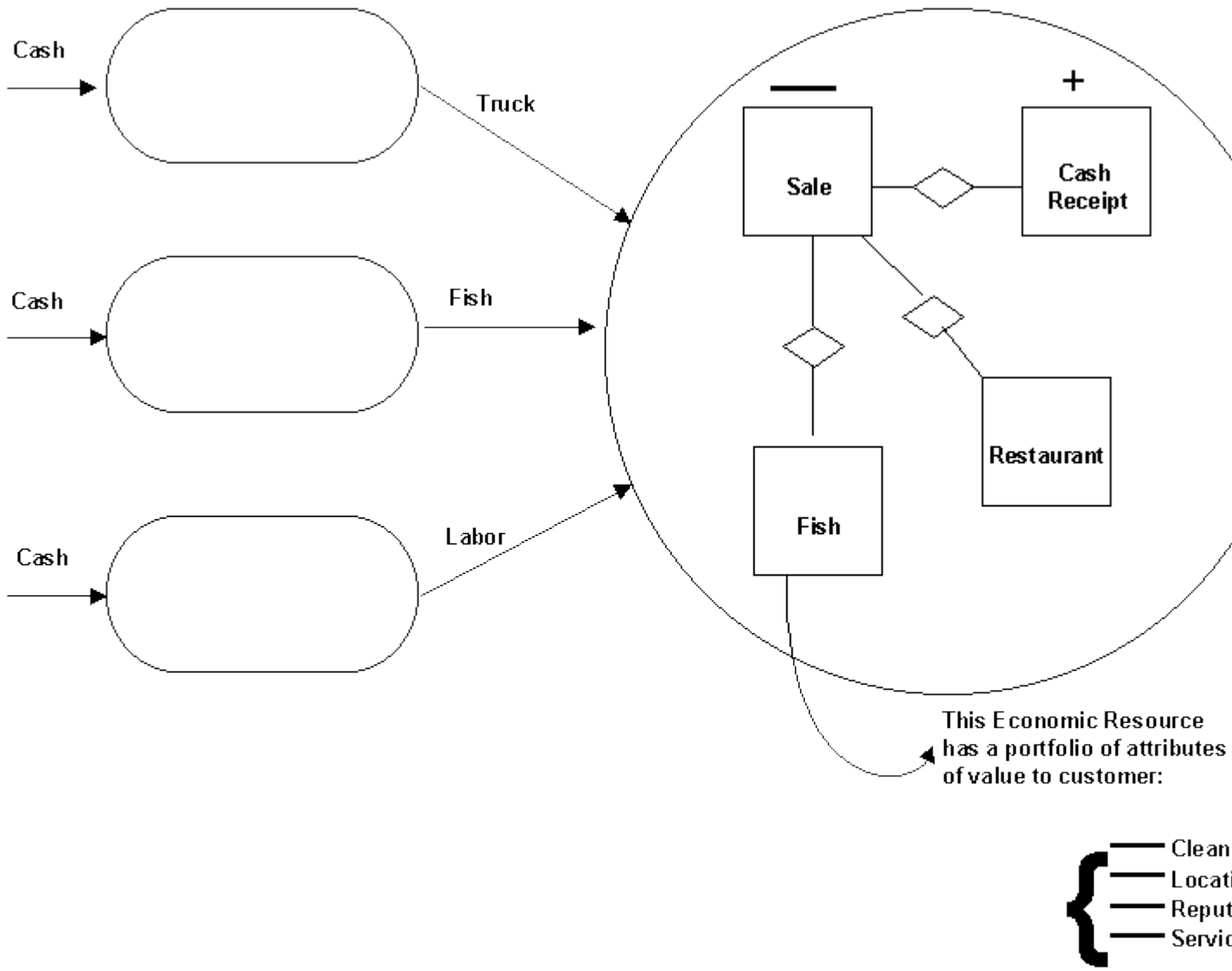
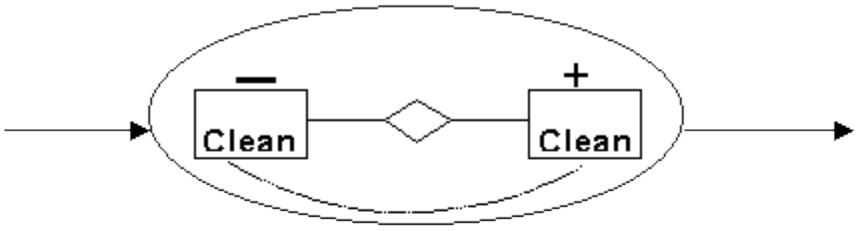
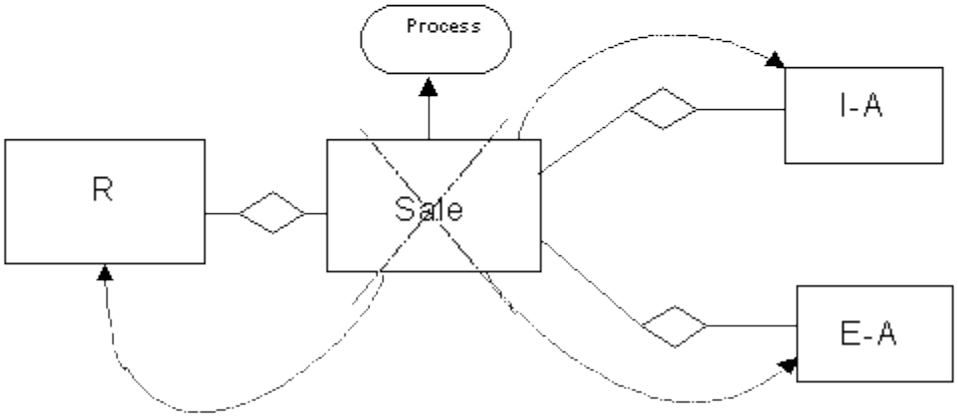


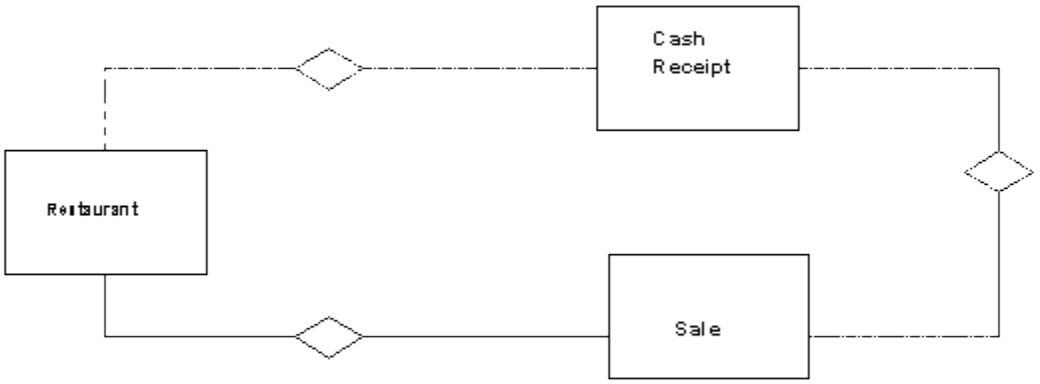

**Figure 4--Customer-Driven Value Chain**




a. combine give-take (internal transfer of resource)




b. temporal aggregation




c. procedural-declarative trade-off

**Figure 5 -- Possible REA Compromises for Sy's Fish**