Financial Information Systems: Teaching REA semantics within an Information Engineering framework

J Callaghan

Professor of Accounting, School of Business Administration, Oakland University, Rochester, Michigan, United States of America

A Savage

Assistant Professor of Accounting, School of Business Administration, Oakland University, Rochester, Michigan, United States of America

E Peacock

Professor of Accounting, School of Business Administration, Oakland University, Rochester, Michigan, United States of America

Received: March 2002
Revised: August 2002
Accepted: August 2002

SAJAR
Vol 16 No. 1
2002
pp.59 to 80

The purpose of this paper is to describe how the accounting *REA* model can be incorporated into the Information Engineering set of methodologies in the Accounting or Financial Information Systems classroom. We contrast the traditional Information Engineering approach with one that includes *REA* modeling. We argue that the *REA* model is an interaction model, and that its use enables accounting students to develop business information systems that more adequately depict business phenomena. We also advocate a hands-on approach to systems development by supplementing conceptual model building with actual systems building through the use of a systems development toolset that automatically generates both application and database programming code from the logical models developed by the students. Consequently, students can see how their abstractions have real-world consequences.

A prior version of this paper was awarded first prize in the 1999 Sterling Software University Program IT Research Contest, and was the topic of the keynote address at the 5th Annual Sterling Software University Program Conference (1999) in Dallas, Texas. Revised versions of the paper were presented at the 2000 AIS Educator Conference in Denver, Colorado and the 2002 American Accounting Association Information Systems Mid-Year Conference in Orlando, Florida.

KEY WORDS
REA model, Information Engineering, Systems Development Life Cycle

Contact
Email: savage@oakland.edu

INTRODUCTION

In the accounting discipline, a Resource-Event-Agent (*REA*) model has long been proposed as providing the necessary semantics for developing modern enterprise-wide information systems (McCarthy, 1979, 1982, 1999) capable of providing real-time information to users. Currently, more than 100 business schools in the United States teach *REA* modeling to some degree in Accounting or Financial Information Systems courses (see McCarthy, 1999). However, no rigorous attempt has been made to integrate *REA* modeling into the modeling techniques described in the Information Engineering for business literature.

We have successfully integrated the accounting *REA* model into the Information Engineering systems development set of methodologies in our new Financial Information Systems curriculum (comprehensively described in Callaghan, Savage & Peacock, forthcoming). Within this curriculum, our Model-Oriented, Tool-Enhanced (MOTE) approach (Callaghan, Lauer & Peacock 1998; Callaghan, Peacock & Savage 2000) aims to teach conceptual understanding and modeling skills in an accounting context.

Conceptual models are implemented through the use of a systems development toolset with downstream effects. The use of this toolset is important, because it enables students who are confronted with highly abstract concepts at the analysis stage of systems development to interact with these concepts more easily in the form of diagrams. Because of the downstream capabilities of the toolset, the software automatically generates the programming code. Therefore, accounting students engaged in systems development do not require sophisticated programming skills.

The abstract models developed by the students eventually lead to working business information systems, and students can see how the abstractions have real-world consequences. Accounting students, as future systems evaluators, can also use the diagrams to facilitate their understanding of the business processes and the underlying databases that capture and store the data about events (or activities) that are of interest to the business. Furthermore, by obtaining an in-depth understanding of systems development, prospective accountants and auditors learn how controls should be embedded in information systems in the form of business rules.

The purpose of this paper is twofold. First, it describes how we have incorporated the *REA* model into the Information Engineering set of methodologies as an interaction model within our Financial Information Systems curriculum. This provides our students with additional structure to assist in the development of information systems that more adequately depict business phenomena and thus produce more reliable financial reports. Second, we recommend the use of systems development software, used predominantly in management information systems and computer science courses, for use in Accounting or Financial Information Systems courses. These toolsets are valuable because they facilitate the practical software engineering component of the increasing trend towards enterprise-wide systems development, of which accounting is an integral part. For a powerful argument on the need for accountants to participate in enterprise-wide systems development, as opposed to focusing on traditional accounting systems, see Walker and Denna (1997).

By combining the accounting discipline's *REA* model with activity (process) modeling at the conceptual level, the student arrives at better interaction models. These, in turn, lead to better models of business semantics. This modeling approach is enhanced by the use of an appropriate systems development toolset, which would support accounting information systems development across functional areas of the business. The use of an integrated systems development toolset in the classroom is particularly advantageous because it supports the construction of information systems for a variety of operating systems, database management systems and programming languages. Consequently, it allows for the development of higher-level planning and analysis skills, with less emphasis on knowledge of programming languages and technology-dependent skills.

The remainder of the paper proceeds as follows. The next section summarizes the Financial Information Systems curriculum into which we position our teaching innovation. The third section gives an example of how a software development toolset with downstream effects implements the Systems Development Life Cycle logical framework, by following the conventional Information Engineering method of systems development. The fourth section describes the general approach to *REA* modeling and contributes to the *REA* literature stream in accounting by describing how this model can be incorporated into the Information Engineering framework as an interaction model. The final section concludes the study.

THE FINANCIAL INFORMATION SYSTEMS CURRICULUM

Our Financial Information Systems curriculum (see Callaghan *et al.*, forthcoming) allows students to integrate information technology (IT) and financial information into the development of business information systems. Our objective is to provide accounting graduates with the knowledge they need to leverage the latest information technologies to support the use of financial information in management decision-

making, and to integrate financial information and internal controls into business information systems. Our technology-rich approach retains the rigor of traditional programs for those students striving for professional accounting certification, but it also expands the horizons of the prospective accounting professional, from one of viewing accounting as a stand-alone, untimely, inflexible information system, capturing only accounting transactions and their limited characteristics, to one of a real-time, enterprise-wide, activity-driven information system, used by a variety of users with a various needs. We also shift the focus from implementing costly controls to that of embedding controls within information systems during the systems development phase (Callaghan *et al,* forthcoming).

A real need exists for this type of systems development knowledge in the business world. Often, financial information systems are developed entirely by non-financial systems developers, because accounting professionals lack the IT knowledge to be an integral part of the development team. A recent study by New York-based Cap Gemini Ernst & Young shows that 63% of 265 Chief Financial Officers believe that they are hampered by inadequate financial information systems, and that their efforts towards full and accurate financial disclosures are impeded (Hoffman, 2002). Their dilemma is described as follows:

> Part of the problem is that many companies continue to rely on outmoded legacy systems that fail to draw needed financial information from across all facets of the business. In many cases, companies have automated accounting and finance functions but have failed to properly link their financial processes with those systems. And in some cases, the processes themselves are poorly designed. (Hoffman, 2000:1)

The successful completion of our curriculum provides accounting students with skills to address these shortcomings in the contemporary business environment, by using the previously mentioned MOTE approach (Callaghan *et al.*, 1998:57-65). The model-oriented aspect uses systems development methods that permit high-level abstractions of real-world financial information systems. Included are activity, data, and interaction models. The tool-enhanced aspect uses advanced systems development software that converts these models into code executable in different technical environments. This approach makes it possible to provide an even balance of the conceptual and the practical, while at the same time allowing for the development and implementation of actual financial information systems (Callaghan *et al.*, 1998:61).

Our curriculum has four required financial information systems courses covering (1) introduction to financial information systems and databases, (2) financial information systems analysis, (3) financial information systems design, and (4) information systems audit and control (for course descriptions, objectives, and major

topics, see Callaghan *et al.*, forthcoming). We cover the entire Systems Development Life Cycle (discussed in the next section) during the delivery of these courses.

IMPLEMENTING THE SYSTEMS DEVELOPMENT LIFE CYCLE

The Systems Development Life Cycle framework is a systematic and orderly theoretical approach used to guide systems development, while the Information Engineering set of methodologies is used to implement this conceptual framework and to create a working, enterprise-wide business information system.

We acknowledge that Unified Modeling Language (UML), an object-oriented approach, could be used instead of an Information Engineering approach, as UML could also follow the phases of the Systems Development Life Cycle. However, from a pedagogical perspective, we prefer the Information Engineering approach because of the emphasis that is placed on the analysis phase of systems development, and because it follows the Systems Development Life Cycle stages lockstep. In the Information Engineering approach, data models and activity models are initially separated in a logical manner, while in UML they are presented in an integrated form that facilitates the object-oriented methodology. The logical separation of data and activity models allows students to focus on these issues separately, while interaction modeling permits formal coupling. In any event, once Information Engineering interaction models are developed (which *REA* facilitates), they can be used traditionally or used in an object-orientated approach. For students well versed in the Information Engineering approach, the switch to UML could be made with less difficulty.

The Systems Development Life Cycle framework defines the phases that are essential to any systems development project, namely, systems planning, analysis, design, and construction.

1. ***Systems planning*** is concerned with how information architecture can be used to implement business strategy, and is covered comprehensively in our introduction to financial information systems and databases course.

2. In ***systems analysis***, business processes needed to implement business strategy are identified, and models (conceptual or logical representations of reality) are built. This should be done free of any technical implementation considerations, as the student should be concerned with what the business does and what information it needs, and not with how the system should be implemented. Systems analysis is covered in our financial information systems analysis course.

3. ***Systems design*** is concerned with describing and documenting how specific procedures are used to implement the business processes identified in systems analysis. Design has two parts: external design and internal or technical design. External design, which is technologically independent of the targeted environment, is when the student builds user interfaces. In internal design, the system is then mapped to the chosen technical environment. We cover this material in our financial information systems design course.

4. In the ***construction*** stage, the design outputs lead directly to their implementation through the generation of computer code, user interfaces, and the physical creation of a database through standard data definition language. This stage is covered in the financial information systems design course.

This top-down approach to teaching systems development offers a "divide-and-conquer" approach that facilitates further refinement as one proceeds down the Systems Development Life Cycle. Because of business complexity, systems development on an enterprise-wide basis cannot effectively be achieved without automated tools (Martin 1989b:viii). James Martin's (1989a, 1989b, 1989c) seminal works on Information Engineering are widely considered to be the most advanced and complete set of systems development methodologies (e.g., Texas Instruments, 1990; Whitten & Bentley, 199:123, 175). Martin (1989a:1) defines Information Engineering as "the application of an interlocking set of formal techniques for the planning, analysis, design, and construction of information systems on an enterprise-wide basis or across a major section of the enterprise." Consistent with this, Martin (1989a:1) also defines Information Engineering with reference to automated techniques, as follows: "An interlocking set of *automated techniques* in which enterprise models, data models, and process models are built up in a comprehensive knowledge base and are used to create and maintain data processing systems."

In our information systems courses, we chose to use Advantage™ Gen[1] in the classroom because it is one of the most integrated and technologically independent of these automated toolsets. It supports the development of information systems for a wide variety of operating system, database management system and programming language combinations. Oracle Designer™ combined with Developer™ is an alternative to Advantage™ Gen for use in the classroom.

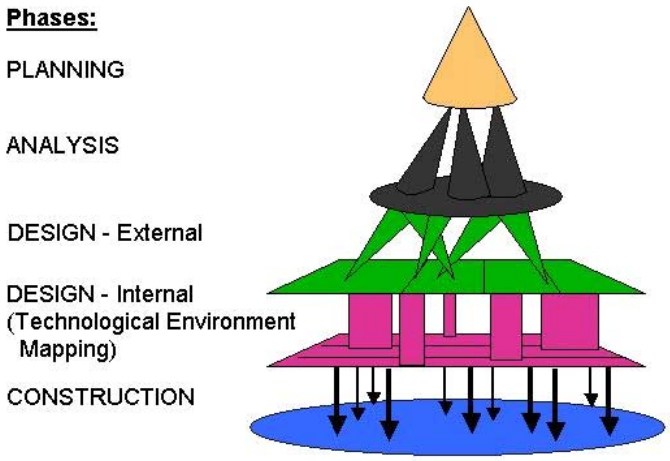
For accounting students, the use of a highly integrated software development toolset such as Advantage™ Gen or Oracle Designer™/Developer™ facilitates the leveraging of higher-level planning and analysis skills, with much less emphasis of technology dependent skills, such as detailed knowledge of particular operating

[1]This toolset, to which the seminal works of Martin (1989a:45, 50; 1989c:24, 26, 60, 77) repeatedly refer., was formerly known as Texas Instruments IEF (Information Engineering Facility).

systems, database management systems, or programming languages. One of the greatest benefits for students is that these toolsets have downstream effects and, based on the models that the students draw, generate the programming code. This eliminates the struggle over the traditional tradeoff between diagramming and programming skills, as it allows for the development of systems without the possession of programming skills, which most accounting students lack.

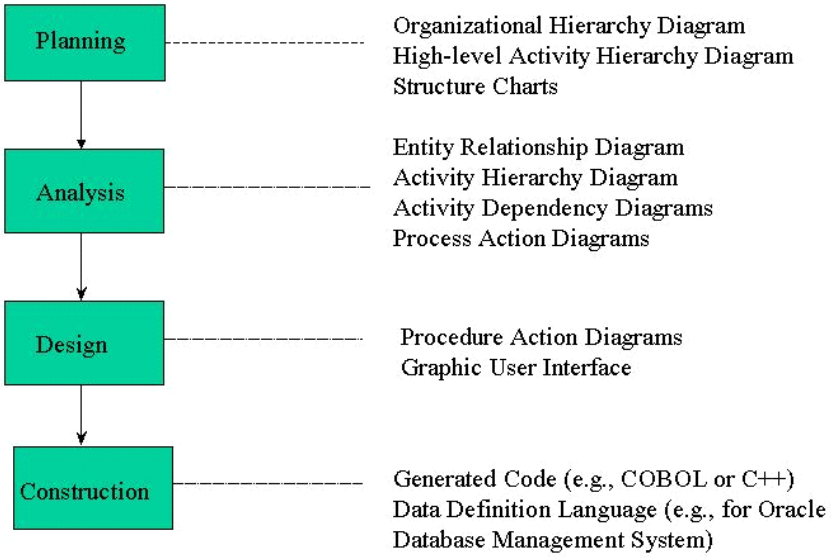
Furthermore, the planning, analysis, and external design phases of systems development are technologically independent, leaving only internal design and construction to specify particular technological combinations (see Callaghan *et al.,* 1998). Figures 1-A and 1-B illustrate how a toolset like Advantage™ Gen provides an integrated set of tools for all parts of the Systems Development Life Cycle, as opposed to having a separate tool for each of the four stages of systems development.

Figure 1-A
Toolset Implements Entire Systems Development Life Cycle



Source: Adapted from Sterling Software, Inc., 1997, p. 17.

Figure 1-B
Toolset Outputs



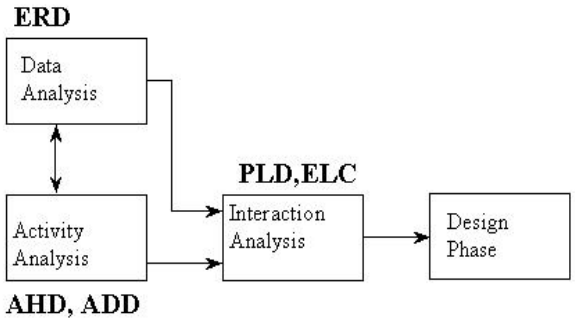
Traditionally, *data modeling* (a conceptual representation or "picture" of what the database would look like in the fully developed system) and *activity modeling* (diagrams of the processes and events of interest to the business, also referred to as *process modeling*) are done separately, with informal, iterative confirmation between the two. Once the systems designer is satisfied with the individual models, (s)he would proceed to formal *interaction modeling* (a representation of how activities/events and data interact with each other). Interaction modeling normally comes at the tail end of systems analysis, and precedes the systems design phase of the Systems Development Life Cycle. Its main purpose is to subsequently confirm the data and activity models.

A discussion of data models, activity models and interaction models follows:

(1) **Data Model**: The Entity Relationship Diagram (ERD) is a frequently used data model in which *entities* (fundamental things of interest to the organization about which data must be kept) and the *relationships* between the various entities are diagrammed. Examples of entities are *customer*, *inventory*, *store, salesperson,* and *customer order.* A relationship is a business association that exists between one or more entities, for example, *customer* may place one or more *customer orders*, and

customer order must always be placed by only one *customer*. Relationships and their optionalities and cardinalities represent *business rules*, and lead to foreign keys during the design phase of systems development. It is important for accountants and auditors to be involved in building data models, as they are in a good position to ensure that internal controls are embedded in the information system by well-specified relationships.

Figure 2
Systems Analysis Stage of Systems Development Life Cycle:
Traditional Information Engineering Analysis Tasks



Legend: ERD - Entity Relationship Diagram
AHD - Activity Hierarchy Diagram
ADD – Activity Dependency Diagram
PLD - Process Logic Diagramming
ELC – Entity Life Cycle Diagramming

(2) **Activity Models**: In activity modeling (also called process modeling), analysts typically use some form of *decomposition* and *dependency diagrams*. Decomposition means breaking down a system in increasing detail, until the lowest levels of decomposition are reached. In this manner, manageable subsets of the overall system can be identified. An example of a decomposition diagram is an *Activity Hierarchy Diagram*, in which the highest levels of business processes are identified and agreed upon. Each of these processes are then analyzed in more detail, and decomposed into lower-level processes. This decomposition is continued, until groupings of lowest-level activities (also called elementary processes) that fully support each of the highest-level processes are reached. Dependency diagrams document the sequence in

which processes, and the events of which the processes are composed, occur. The main role of dependency analysis is to confirm activity decomposition. An example would be an *Activity Dependency Diagram*.

(3) **Interaction Models**: Analysts typically prepare data and activity models, as described above, and then proceed to formal interaction analysis to confirm these models. Interaction models show the interaction between business events and the data about the event that the business wishes to collect. Two forms of formal interaction modeling are *Process Logic Diagramming* and *Entity Life Cycle Diagramming*. Process Logic Diagramming focuses on activities, with analysts confirming their understanding of the data model by ensuring that each entity or activity depicted by activity models would have a corresponding data store (i.e., a mapping from activity model to data model). An event or activity without data would be unacceptable and would require amendments to the models. Entity Life Cycle Diagramming focuses on data stores, with systems analysts performing a mapping from the data model to activity models, in an attempt to identify data without an event or activity that would at least create the data.

In the next section, we integrate two frameworks: (1) the traditional Information Engineering framework (Martin, 1989a, 1989b, 1989c) described above, and (2) the accounting *REA* framework (McCarthy, 1979, 1982; Hollander, Denna & Cherrington, 1996, 2000). By doing this, we put *REA* modeling into its proper systems development context.

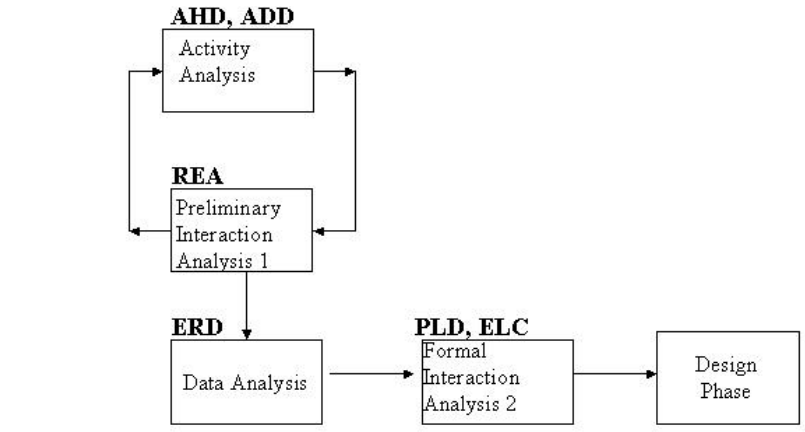
THE *REA* MODEL AS AN INTERACTION MODEL

A useful extension of McCarthy's (1979, 1982) *REA* model adds "location" as a modeling element, and is also known as the Resource-Event-Agent-Location (*REAL*) model (Hollander *et al*., 1996, 2000). Our application of *REA* modeling within an Information Engineering framework is illustrated in Figure 3. It can be compared to the traditional approach depicted in Figure 2.

The integration of the *REA* model into the traditional framework in our courses proceeds as follows:

(1) The enterprise is initially decomposed (using an Activity Hierarchy Diagram) into three possible high-level business process types, which will subsequently be decomposed into various stages of lower level processes. This first level decomposition is made up of (1) Acquire-Maintain-Pay (AMP) of Resources, (2) Conversion Processes, and (3) Market-Sell-Collect (MSC) of Products/Services (Hollander *et al*., 2000:4-5). Figure 4 depicts a template business process breakdown of an enterprise, using the three process types described above.

Figure 3
Analysis Tasks Incorporating *REA* Interaction Modeling

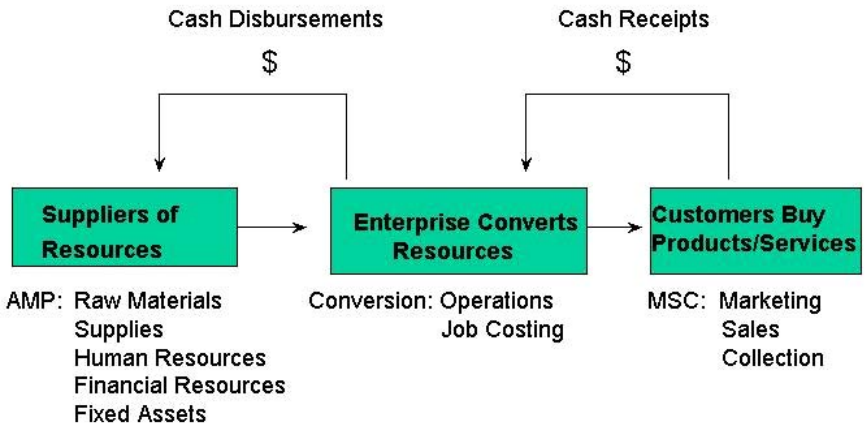


Legend: AHD - Activity Hierarchy Diagram
ADD - Activity Dependency Diagram
ERD - Entity Relationship Diagram
PLD - Process Logic Diagramming
ELC - Entity Life Cycle Diagramming

On the input side of the enterprise, the AMP template is used to identify processes for acquiring, maintaining, and paying for the resources needed by the business. For instance, an AMP of Raw Materials may be discovered for a manufacturer in this manner. Others could be AMPs of Human Resources, Financial Resources, Fixed Assets, and Supplies. The second and third types are Conversion and MSC processes, corresponding to value-added and output processes, respectively.

(2) Once an actual named business process is identified, for example, AMP of Raw Materials, the business *events* (synonymous with business *activities*) associated with that process are identified. This set of related business events is intended to adequately describe, at some proper level of disaggregation, the semantics of interest to the ultimate users of the system. If an AMP of Raw Materials is identified as a business process, it might be determined that there are three related events: (1) a receipt of raw materials event, (2) a storage (and maintenance) of raw materials event (this refers to physical storage and maintenance of the actual asset, as opposed to the storage and maintenance of the data in the database), and (3) a payment to the vendor for the raw materials event.

Figure 4
Template For Modeling An Enterprise:
A Collection of Template Processes



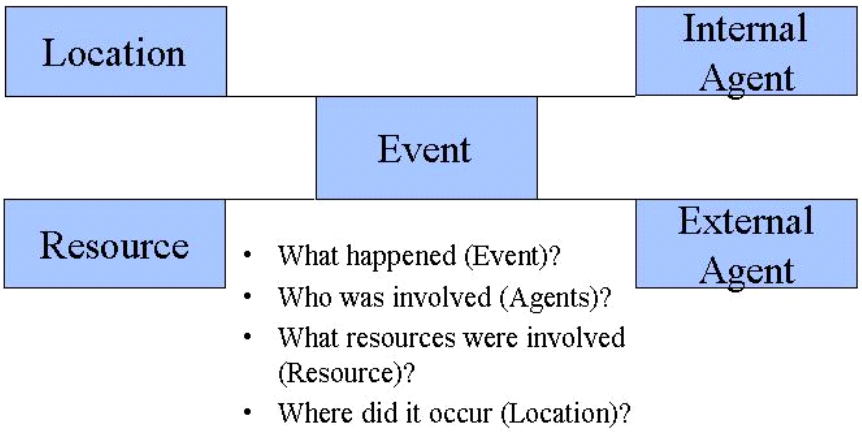
(3) For each identified business event (the "E" in *REAL*) of a given process, e.g., receive raw materials, the resource ("R") involved (e.g., raw materials), the agents ("A") involved (the vendor as the external agent and the purchasing agent as the internal agent), and the location ("L") (e.g., warehouse or division) would be identified. Figure 5 illustrates the *REA* model framework of a general event (see McCarthy, 1979, 1982, 1999; Hollander *et al*., 2000).

(4) Event triggers and special business rules are then described. For example, a business condition of low stock may be identified as the trigger for the "acquire raw materials" process. A special business rule might be that each purchasing agent is assigned to specific vendors.

(5) At this stage, a rough hand-drawn interaction model can be presented (although the existing *REA* literature does not identify the *REA* model as an interaction model), with the identified business events, their surrounding resources, agents, and locations (RALs), and their implied relationships. Interaction models define how the things that the business does (events or activities) affect the things of interest to the business (data pertaining to event and its RALs), and vice versa. These models

attempt to permit the systems analyst a deeper insight into data and activities by formally modeling how they interact with one another.

Figure 5
***REA* Model of A Business Event:**
The Event and Surrounding Resource/Agents/Location



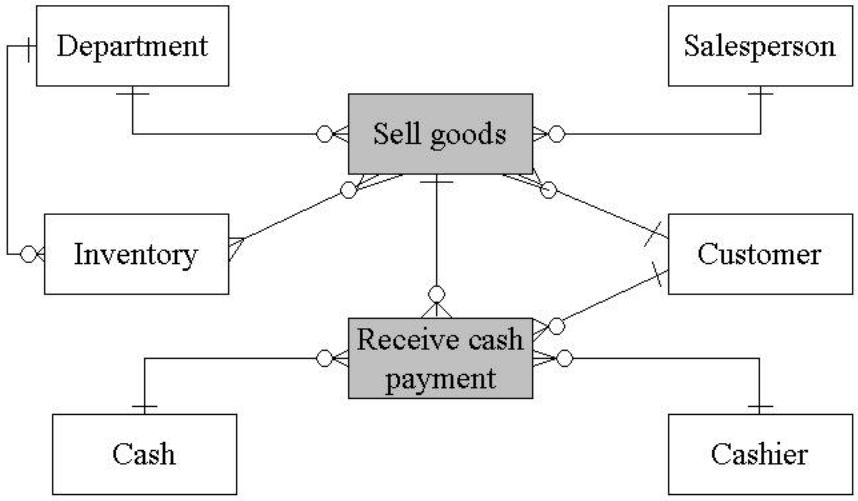
(6) After confirming the rough interaction model with users from different functional areas and at different organizational levels within given functional areas, the model would be refined, as follows: First, the analyst would consider the best way to record the business events and their surrounding RALs, usually through a process of normalization (through third normal form for most business applications). At this point, what were activities (i.e., business events) are now the proper data objects (i.e., entities) of the information process of recording these activities. Secondly, after realigning the direct relationships implied from the rough model to take account of the refinements of the first step, the nature of the relationships among the entities of the model would be re-specified, by indicating the cardinalities and optionalities of these relationships. Proper specification of relationships at the model level will ensure that business rules are embedded in the system, once it is implemented in the form of a relational database management system.

(7) The unique identifiers (which become primary keys in the database) and the non-key attributes for each entity, the data types for each attribute, and the domains from which attribute values could derive would then be specified.

Completion of these seven steps would facilitate the development of the ERD, which is a normalized (to third normal form) data model for the analyzed business process. In practice, the process is iterative, constantly factoring in user feedback.

Once the events of interest have been specified and modeled, an information systems application can readily be designed. Figure 6 shows the *REA* model for a simple MSC business process, using Advantage™ Gen notation (for a comprehensive review of *REA* modeling, see Hollander *et al*., 2000).

Figure 6
***REA* Interaction Model for a MSC Process:**
Two Events and Surrounding RALs



Our approach integrates *REA* modeling into the analysis phase of Information Engineering, by explicitly incorporating two activity models, the Activity Hierarchy Diagram and the Activity Dependency Diagram, which in turn are interactive. Recall that activity models record the activities of interest to the business (i.e., the things the business does or should do). The Activity Hierarchy Diagram involves decomposition of business processes from the highest level (AMP of Resources, Conversion, MSC of Products/Services) to the lowest elementary processes, which include the *events* discovered by using *REA* modeling.

This is where accounting's *REA* model makes a contribution to the MIS discipline's systems analysis methods (see Figure 2). The *REA* model provides a framework for specifying a well-developed ERD. Elementary processes are the smallest units of business activity of meaning to users which, when complete, leave the business in a consistent state (when the elementary process is complete, it must not violate any of the rules for an ERD in third normal form, and this would then leave the business in a consistent state).

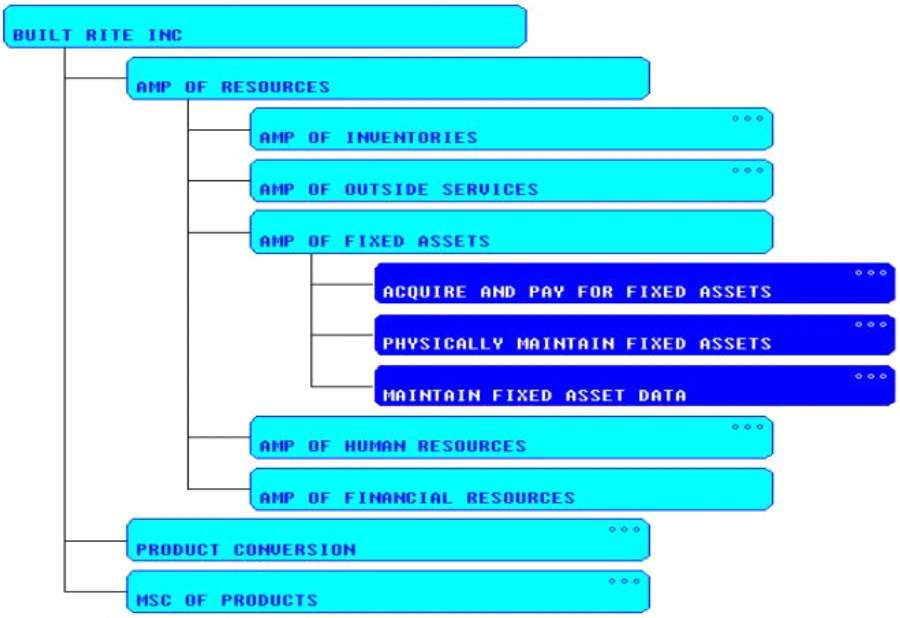
The result must support a business need (Sterling Software, 1997:65). These lowest-level processes are typically the most difficult processes for the systems analyst to identify. The *REA* model helps to solve this MIS problem and provides the developer with a pattern to arrive at the events at the lowest level of decomposition. The Activity Hierarchy Diagram is complete once all the elementary processes (*REA* events, plus the events needed for the maintenance of the RALs) have been specified for each high-level process.

The Activity Dependency Diagram is done concurrently with the Activity Hierarchy Diagram. The Activity Dependency Diagram involves modeling, at each level of decomposition, the dependencies among all processes or, at the lowest level of decomposition, dependencies between all events. These dependencies could be sequential, parallel, or mutually exclusive. Additional interaction modeling (Process Logic and Entity Life Cycle Diagramming) would be used to confirm the Activity Hierarchy Diagram, Activity Dependency Diagram, and ERD models.

Figure 3 graphically illustrates how the interactive *REA* model is incorporated into the traditional MIS approach (which was shown in Figure 2).

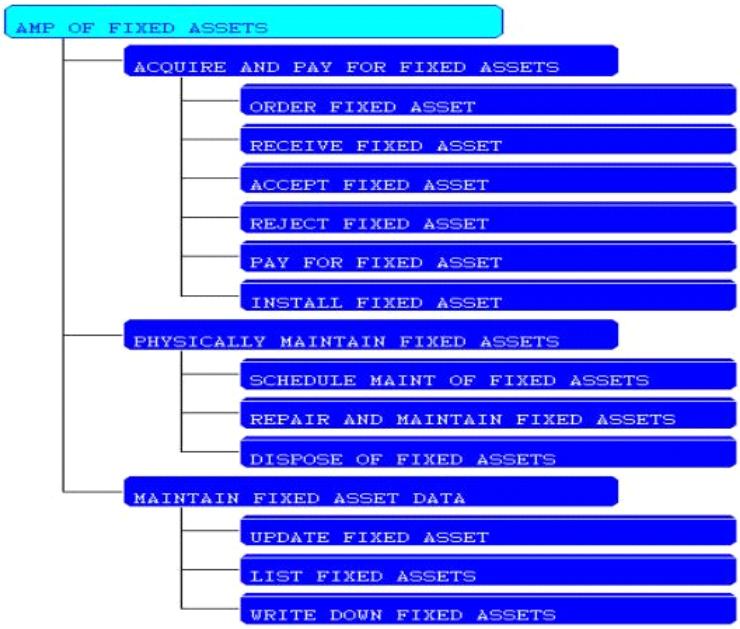
Activity Hierarchy Diagramming can use the template shown in Figure 4 for first-level business decomposition into high-level AMP, Conversion, and MSC business processes. During planning, an analyst could arrive at a higher-level model similar to that shown in Figure 7-A, which is taken from a case that we developed for teaching purposes, called Built Rite Inc.

Figure 7-A
Activity Hierarchy Diagram: Example of Partial Decomposition



Therefore, during analysis for a given business process, successive decomposition would iteratively occur until business events are derived. Ultimately, this decomposition would arrive at elementary processes, which roughly correspond to the business events of *REA* models. Moreover, the Activity Hierarchy Diagram would identify other "events" that would include the maintenance of RALs, namely, Create, Retrieve, Update, and Delete (CRUD). For example, "create customer" is an important business event, although it does not involve an economic exchange. To make this distinction, the Activity Hierarchy Diagram is considered complete when arriving at the level of the elementary processes, which consist of *REA* model events, plus CRUD events. Elementary processes for an AMP of Fixed Assets are illustrated in Figure 7-B.

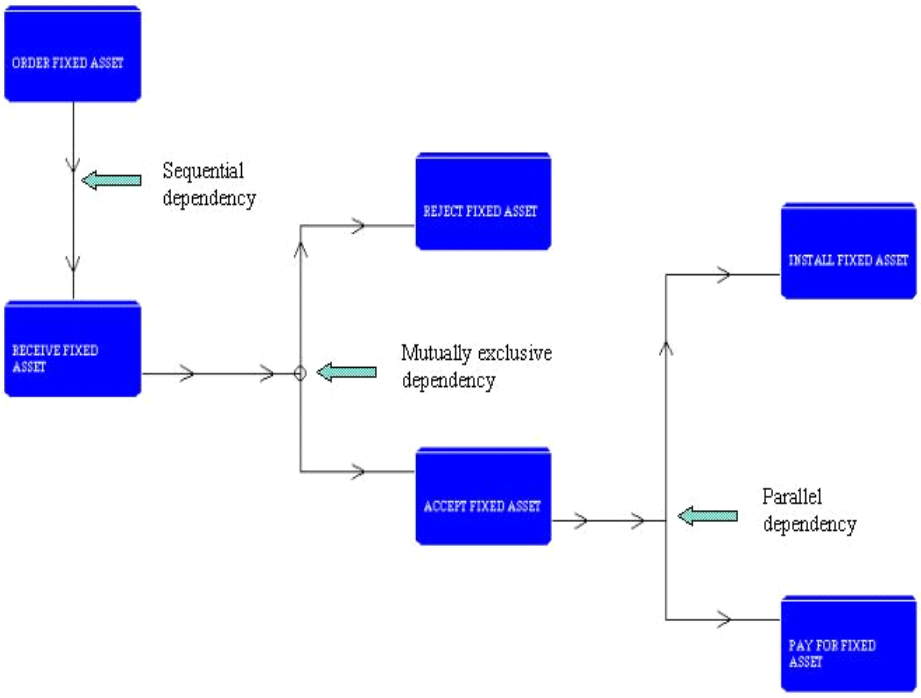
Figure 7-B
Activity Hierarchy Diagram: Elementary Processes of AMP of Fixed Assets



Another activity model, the Activity Dependency Diagram, will be done in conjunction with the Activity Hierarchy Diagram. An example of the Activity Dependency Diagram for the elementary processes of "Acquire and Pay for Fixed Assets" (following the Activity Hierarchy Diagram shown in Figure 7-B) is shown in Figure 8.

Concurrently and iteratively, during the analysis phase of the systems development life cycle, the activity models would be cross-checked against the preliminary data model developed under *REA* modeling, with changes occurring in all three models. Thereafter, the analyst would confirm all these models by employing Process Logic and Entity Life Cycle Diagramming, which explicitly model the interactions between the activity models and the data model. Of course, throughout the analysis phase, all models and techniques would be further confirmed and specified with feedback obtained from users of the system being developed.

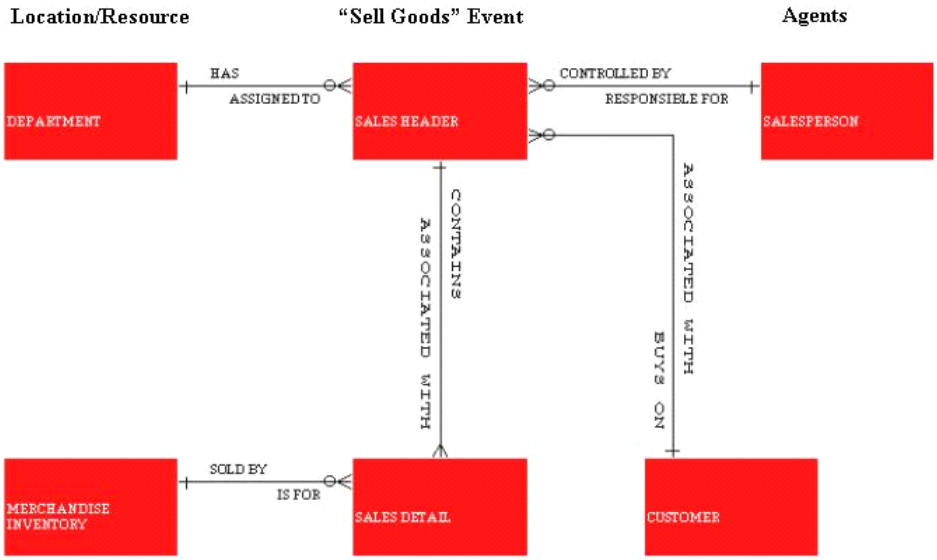
Figure 8
Activity Dependency Diagram for "Acquire and Pay for Fixed Assets"



With the increased possibility of identifying, through Activity Hierarchy Diagram and Activity Dependency Diagram modeling, the "correct" business events of a given business process, *REA* modeling could proceed as originally indicated, and then be converted to an ERD. Activity Hierarchy Diagram and Activity Dependency Diagram analyses assist in the identification of undiscovered business events, or lead to business events at higher or lower levels of aggregation or decomposition than originally contemplated. These redefined business events would be analyzed as originally outlined. That is, each event would be surrounded by redefined RALs, relationships would be respecified, and so forth. The results would lead to improved specification of ERDs. The ERD is a data model that depicts data (entities, attributes) and the relationships enforcing business rules between entities. Figure 9 illustrates an ERD built in Advantage™ Gen. The data model is physically implemented as a database in a developed system.

Figure 9

ERD for a "Sell Goods" Event



Our approach combines *REA* modeling and activity analysis in an explicitly iterative manner. The development of data and activity models requires iteration, to validate (1) the data and process components individually, as well as their interactions, and (2) the business rules or internal controls embedded in the models.

Validation of business rules is accomplished by presenting successive versions of the models to the business users for discussion and eventual approval, resulting in the modification of business rules by which business processes are conducted. The iteration that must take place in order to refine and validate a model is often missed in classroom expositions of modeling. In practice, the cost of failing to iterate will result in invalid or missing rules and controls.

In contrast to formal interaction modeling, *REA* modeling (1) is done earlier, (2) gives semantics by helping with the development of activity and data models (as opposed to merely being confirmatory), and (3) is done in an iterative manner.

CONCLUSION

Recognizing the dramatic changes occurring in the accounting profession, especially with respect to information technology, there is a high and growing demand for IT-savvy accounting students. We want our graduates to be value-producing accounting and information systems professionals in the real world. Our MOTE approach achieves this objective by embodying a structured model-based approach to accounting and financial information systems development, using the systems development life cycle framework and by incorporating *REA* modeling into the information engineering methodologies. One of the major difficulties that students experience with modeling is to isolate the entities and processes. The *REA* model helps them achieve this by providing structure for the purpose of data and process modeling in the analysis stage of systems development.

We evaluated the content of our courses against the standards of the International Federation of Accountants' International Education Guideline No. 11, "Information Technology In The Accounting Curriculum" (see Callaghan *et al.*, 2000:1-12). We also assessed course content by conducting a survey of accounting and IT business professionals (see Callaghan, Peacock & Savage, 2001:51-60). In addition, our curriculum and teaching approach addresses many of the concerns raised by the Albrecht and Sack (2000) study, which found that accounting educators are not adequately exposing students to information technology, nor are we teaching them how technology can be leveraged to make business decisions.

We adjusted our curriculum and course content according to the results of these empirical studies and concerns raised by Albrecht and Sack (2000). This paper describes the teaching approach by which this is achieved.

REFERENCES

Albrecht, W.J. and Sack, R.J. (2000). *Accounting Education: Charting the Course through a Perilous Future.* Accounting Education Series No. 16. Sarasota: American Accounting Association.

Callaghan, J.H., Lauer, T.W. and Peacock, E. (1998). Developing a Comprehensive Curriculum for Accounting Information Systems: A Model-Oriented, Tool-Enhanced Approach, *The Review of Accounting Information Systems*, 2(4):57-65.

Callaghan, J.H., Peacock, E. and Savage, A. (2000). Assessment of an Accounting Information Systems Curriculum: An Analysis of the International Federation of Accountant's International Guideline 11, *The Review of Accounting Information Systems,* 4(1):1-12.

Callaghan, J. Peacock, E. and Savage, A. (2001). Feedback on Developing an AIS Curriculum, *The Review of Business Information Systems*, 5(4):51-60.

Callaghan, J., Savage, A. and Peacock, E. (forthcoming). Rethinking AIS: An Innovative Financial Information Systems Curriculum, *Advances in Accounting Education: Teaching and Curriculum Innovations*.

Hoffman, T. (2002). IT Woes Hinder Fiscal Reporting, *Computerworld*, 36(34):1, 47.

Hollander, A.S., Denna, E.L. and Cherrington, J.O. (1996). *Accounting, Information Technology, and Business Solutions*. Irwin McGraw-Hill.

Hollander, A.S., Denna, E.L. and Cherrington, J.O. (2000). *Accounting, Information Technology, and Business Solutions*, 2nd Edition. Irwin McGraw-Hill.

Martin, J. (1989a). *Information Engineering: Introduction, Book I*. Englewood Cliffs, NJ: Prentice Hall.

Martin, J. (1989b). *Information Engineering: Planning and Analysis, Book II*. Englewood Cliffs, NJ: Prentice Hall.

Martin, J. (1989c). *Information Engineering: Design and Construction, Book III*. Englewood Cliffs, NJ: Prentice Hall.

McCarthy, W.E. (1979, October). An Entity-Relationship View of Accounting Models, *The Accounting Review*:667-686.

McCarthy, W.E. (1982, July). The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment, *The Accounting Review*:554-578.

McCarthy, W.E. (1999). Semantic Modeling in Accounting Education, Practice, and Research: Some Progress and Impediments. In *Conceptual Modeling: Current Issues and Future Directions*, Chen, P., Akoka, J., Kangassalo, H., & Thalheim, B. (Eds.). Berlin: Springer Verlag.

Sterling Software, Inc. (1997, September). *Diagrams to Code: Student Guide*. Plano, TX: Sterling Software.

Texas Instruments (1990). *A Guide to Information Engineering Using the IEF™: Computer-Aided Planning, Analysis, and Design,* 2nd Ed. Texas Instruments.

Walker, K.B. and Denna, E.L. (1997, July). Arrivederci, Pacioli? A New Accounting System Is Emerging, *Management Accounting*:22-30.

Whitten, J.L. and Bentley, L.D. (1998). *Systems Analysis and Design Methods*, 4th Edition. Irwin McGraw-Hill.