Augmented Intensional Reasoning in Knowledge-Based Accounting Systems

Guido L. Geerts
University of Delaware

William E. McCarthy
Michigan State University

Send page proofs to:

William E. McCarthy
N270 – Department of Accounting
North Business Complex
Michigan State University
East Lansing, MI 48824-1121
TEL: 517-432-2913
EMAIL: mccarth4@msu.edu

Augmented Intensional Reasoning in Knowledge-Based Accounting Systems

Guido L. Geerts
University of Delaware

William E. McCarthy
Michigan State University

ABSTRACT: A limitation of existing accounting systems is their lack of knowledge sharing and knowledge reuse, which makes the design and implementation of new accounting systems time consuming and expensive. An important requirement for knowledge sharing and reuse is the existence of a common semantic infrastructure. In this article we use McCarthy's (1982) Resource-Event-Agent (REA) model as a common semantic infrastructure in an accounting context. The objective is to make knowledge-intensive use of REA to share accounting concepts across functional boundaries and to reuse these concepts in different applications and different systems, an approach we call *augmented intensional reasoning*. Intensional reasoning is the active use of conceptual structures in information systems operations such as design and information retrieval. For augmented intensional reasoning, the conceptual structures are extended with domain-specific REA knowledge. Sections II and III describe different dimensions of augmented intensional reasoning: the REA primitives, the technological features needed to support augmented intensional reasoning, the need for epistemologically-adequate representations to make augmented intensional reasoning feasible, and the practical necessity of implementation compromises. Sections IV and V explore two uses of augmented intensional reasoning: *design* and *operation* of knowledge-based accounting systems. The example in section V explains how augmented intensional reasoning works: (a) define the conceptual schema, (b) structure the conceptual schema in terms of REA (knowledge augmentation), (c) define a shareable and reusable accounting concept (claim), and (d) use the concept (claim) to derive information in different accounting cycles (revenue and acquisition).

Key Words: Augmented intensional reasoning, Epistemological adequacy, Implementation compromise, Knowledge reuse and sharing, Procedural-declarative tradeoff, REA accounting.

We would like to acknowledge the helpful comments of Jan Pukite, Thomas Verghese, the members of the Michigan State AIS Workshop, the editor, and three anonymous referees on the previous versions of this paper. An early version of some of the ideas in this paper was presented at the Twelfth International Workshop on Expert Systems and Their Applications, June, 1992, Avignon, France (Geerts and McCarthy 1992). Financial support for this work was provided by Arthur Andersen and the Free University of Brussels. Accepted by the previous editor, A. Faye Borthick.

I. INTRODUCTION

In the late 1990s, there has been a strong movement toward more active use of enterprise knowledge structures, movements characterized as enterprise modeling, knowledge management, and enterprise ontology development (Hayes-Roth 1997; Bernus et al.1998; Gomez-Perez 1998; Guarino 1998; Rolstatdas 1999). In these areas, there is a strong overriding insistence on explicit and persistent representation of enterprise knowledge structures so that these structures graduate from being simple requirements definition and analysis tools to being active components in systems operation and information retrieval.

The objective of this article is to extend this knowledge management research by illustrating the active use of domain-specific knowledge structures in accounting applications. We use the Resource-Event-Agent (REA) model (McCarthy 1982) to augment the enterprise schema with domain-specific knowledge. The active use of conceptual structures is known as *intensional reasoning*. We use the term *augmented intensional reasoning* for the active use of conceptual structures augmented with the domain-specific REA structures imposed on top of the enterprise schema. The main advantages of augmented intensional reasoning are knowledge sharing across functional borders and knowledge reuse across different implementations.

To apply augmented intensional reasoning, a number of technological and design requirements need to be fulfilled: (1) persistent existence of a common semantic infrastructure, (2) explicit representation of knowledge structures, and (3) existence of epistemologically adequate representations. The common semantic infrastructure should support the homogeneous representation of domain-specific phenomena in a manner that endures after initial system analysis. In our case, the infrastructure is the REA model, which supports an explicit and persistent semantic representation of the economic activities of a company across the value chain. Epistemological adequacy is a metric we propose that expresses the degree to which a conceptual schema structures the economic activities of a company in terms of the REA model.

We compare existing artificial intelligence (AI) accounting applications with our proposed knowledge-based systems to elucidate the advantages and requirements of augmented intensional reasoning as well as to demonstrate its implications for the design and operation of accounting information systems (AIS). Knowledge technology has been applied in accounting since the 1980s, in particular as expert systems that support audit and tax problem solving, e.g. *ExperTAX* (Shpilberg and Graham 1986), *Planet* (McGowan 1996) and *Comet* (Nado et al. 1996). Although they have been successful for specific, well-defined tasks, existing AI accounting applications have some important limitations:

- They lack a common knowledge architecture, which prevents knowledge sharing across functional borders. Current systems are stand-alone applications, which means there is no interface with other intelligent accounting systems or with the production accounting (transaction processing) information system.
- They are designed from scratch. Existing accounting and non-accounting knowledge structures are not reused, which makes the design and implementation of knowledge-based accounting systems time consuming and expensive.

Figure 1 depicts the limitations of existing knowledge-based accounting systems. The boxes on top show stand-alone intelligent systems, and the top portions of the boxes portray

task-specific knowledge. Task-specific knowledge is application-specific, such as the rules used to determine the creditworthiness of a customer. The crosshatched areas represent knowledge that could be shared among two or more intelligent systems but which instead is embedded within each system. The dotted lines show that intelligent applications routinely are not linked to the actual accounting information system and that the data needed from the AIS must be retrieved and formatted separately.

-- INSERT FIGURE 1 HERE --

The importance of knowledge sharing and reuse has been recognized in recent years (Neches et al. 1991; Hayes-Roth 1997; Gomez-Perez 1998). The major challenge for the next generation of intelligent systems is achieving a common knowledge architecture. Achieving a common knowledge architecture requires overcoming many integration obstacles: heterogeneity of representation formalisms, heterogeneity of implementation platforms (Prolog, Lisp, expert system shells, etc.), conflicting lexicons, and the lack of semantic interoperability (Musen 1992; Gomez-Perez 1998). In this article we focus on one dimension of knowledge sharing and reuse: achieving semantic interoperability.

Semantic interoperability requires a knowledge-based infrastructure that is administered across functional boundaries and that is employable in different systems. For accounting systems, this implies a semantic framework that can be shared across traditional cycle-oriented subsystems (such as accounts receivable, accounts payable and payroll) along the enterprise value chain and that can be reused by systems in companies of different sizes and in different sectors. In order to support semantic interoperability in knowledge-based accounting systems, we use REA accounting rather than double-entry accounting as a starting point. We do this because the double-entry paradigm gives primacy to account-oriented classification, which conceals the semantic structure of the enterprise being modeled. When the double-entry filter is applied, most of the accountability data for a company (arising from its economic transactions with workers, customers, creditors, etc.) cannot be used in any knowledge-intensive fashion for non-financial decision purposes. REA accounting does not filter economic data, and it structures accounting and non-accounting data in a homogeneous way. All elements in the economic process are assigned a domain-specific role, and we use those role assignments to create a shareable and reusable semantic infrastructure. Semantic interoperability would facilitate the use of transaction data in both accounting applications, such as claim materialization, and non-accounting applications, such as customer relationship management and supply chain coordination.

Figure 2 shows a knowledge-based accounting system architecture for supporting knowledge sharing and reusability that has three major components: the accounting information system (ellipses), the REA-based semantic infrastructure (rectangles), and the augmented intensional reasoning component (cylinder).

-- INSERT FIGURE 2 HERE --

The AIS component is represented as two ellipses in the middle of figure 2. The data part contains multidimensional descriptions of economic phenomena across the value chain. Current ERP-type AIS store similar data (Davenport 1998). ERP systems, however, lack the

explicit and persistent representation of a semantically-structured schema (represented on the diagram by the outer ellipse), which contains a detailed description of a company's economic phenomena in an REA model. The enterprise schema has a dynamic nature. Each company will have its own specific enterprise schema which changes over time. An enterprise schema where elements are congruent with all parts of the REA model is called *epistemologically adequate* or *full-REA.* Epistemologically adequate representations are the heart of shareable and reusable knowledge-based accounting systems. When the enterprise schema fails to comply with this representation commitment, the REA-based inference engine needs additional knowledge to draw conclusions. The epistemological adequacy metric can be considered on a continuum where decreases in epistemological adequacy (known as implementation compromises) result in a decrease in knowledge sharing and reuse.

The REA-based semantic infrastructure of figure 2 consists of REA primitives and a taxonomy of shareable and reusable accounting concepts. REA primitives include the basic objects and the relationships between these elements. The taxonomy is a dynamic set of accounting concepts that are defined in terms of REA primitives or other REA-based concept definitions.

The cylinder in figure 2 represents the special-purpose inference engine that uses REA primitives and the taxonomy of REA-based concepts to reason with the elements of the enterprise schema. Reasoning with conceptual schema definitions or intensions is called intensional reasoning. Because it uses the REA structures imposed on top of the conceptual schema, it is called augmented intensional reasoning. Augmented intensional reasoning is a reusable technique, and the degree of that reusability depends on the epistemological adequacy of the enterprise schema.

The dotted boxes at the top of figure 2 represent task-specific knowledge needed for individual applications. They correspond to the top white boxes of figure 1. Because it is neither shareable nor reusable, this knowledge is not part of the common knowledge architecture.

In summary, the objective of this article is to explore the *knowledge-intensive* use of REA for *sharing* accounting concepts across functional boundaries and for *reusing* these concepts in different applications and different systems. The technique we use to accomplish this objective is augmented intensional reasoning. Augmented intensional reasoning requires:

- The existence of a common semantic infrastructure (the REA model),
- The explicit representation of the knowledge structures,
- The congruency of the knowledge structures with all parts of the REA model, and
- The existence of a specific inference engine that can reason with REA primitives, the taxonomy of REA concepts, and the explicitly recorded knowledge structures.

II. THE RESOURCE-EVENT-AGENT MODEL

Adopting a semantic or conceptual description of an accounting object system has the benefits of: (1) harmonizing human communication to give consistent definitions across different accounting and non-accounting user views, and (2) focusing attention on the economic phenomena instead of on implementation and access details. These benefits permit the complexity of system development and use to be managed. Adopting a semantic

description, however, requires a representation formalism, "a set of conventions about how to describe a class of things" (Winston 1992, 16). The Entity-Relationship (E-R) model (Chen 1976; McCarthy 1979; Batini et al. 1992) is used here as representation formalism.

The Resource-Event-Agent (REA) model (McCarthy 1982) is a generalized semantic representation of accounting phenomena that guides the conceptual modeling of an enterprise schema and is used as the basis for the semantic infrastructure developed here. A simplified version of the REA model is illustrated in E-R form in figure 3. Without a loss of generality, we use two binary control relationships instead of the original ternary control relationship (McCarthy 1982) and omit the responsibility relationship between inside agents (like departments that report to each other). Figure 4 illustrates a possible instantiation resulting from applying the REA model with the E-R conventions of Batini et al. (1992). This example illustrates a typical REA-based conceptual description for the revenue cycle.[1]

-- INSERT FIGURE 3 HERE --
-- INSERT FIGURE 4 HERE --

The REA model (figure 3) can be considered as a generic description of an *Economic Event*,[2] and when both sides of the *Duality* relationship are filled, it is a generic description of an economic exchange. The domain theory suggests that at least the following three aspects of an *Economic Event* must be described:

1. Its ***Stock-Flow*** relationship with *Economic Resources*. Figure 4 illustrates that the *Economic Event* Sale results in an *Outflow* of the *Economic Resource* Product and the *Economic Event* Cash Receipt results in an *Inflow* of the *Economic Resource* Cash. *Economic Events* cannot be modeled without identifying the *Economic Resources* (or scarce means) they affect. Additionally, each modeled *Resource* should participate twice in *Stock-Flow* associations: one for *Inflow* and one for *Outflow*.[3]

2. Its ***Control*** relationship with *Economic Agents* inside and outside of the firm. The example illustrates that an *Economic Agent* Salesperson will be held responsible (*Control*) for an *Economic Event* Sale and the corresponding *Outflow* of the *Economic Resource* Product. An *Economic Agent* Cashier will *Control* the *Inflow* of the *Economic Resource* Cash through the *Economic Event* Cash Receipt. Customer is the *External Agent* for both of the *Economic Events* Sale and Cash Receipt.

3. Its ***Duality*** relationship with another *Economic Event* where the increment *Events* are paired with decrement *Events* in a give-take relationship or exchange. The example in figure 4 illustrates a give-take or *Duality* relationship between the *Economic Event*

---

1. The revenue cycle shown here shows only one resource decrement. More realistically, most cycles typically would also include other less prominent resource decrements (such as the use of labor or the use of an asset like a vehicle) called transaction costs in the process representation. In the interest of simplicity, these decompositions and additional decrements have been omitted from the figure. A more complete enumeration of a revenue cycle is available in Geerts and McCarthy (1997).
2. A capitalized term refers to an entity type or object type (e.g. Sale). A capitalized term in italics refers to an REA primitive (e.g. *Economic Event*).
3. Because of space limitations, the example of Figure 4 does not show both flows for Product and Cash. See figure 7 later.

Sale (decrease in *Economic Resource* Product) and the *Economic Event* Cash Receipt (increase in *Economic Resource* Cash).

This article explores the use of the REA model as the foundation for a semantic infrastructure for knowledge-based accounting systems. Resource, event, agent, stock-flow, control, and duality are the REA primitives, represented as the smaller rectangle in figure 2. A conceptual schema that describes all aspects of economic events in the accounting object system in accordance with the REA model of figure 3 is termed *full-REA* (Geerts 1993; Geerts and McCarthy 1994). Full-REA means that economic events participate fully in each of the three relationships (stock-flow, control, and duality) mentioned above.

III. AUGMENTED INTENSIONAL REASONING IN KNOWLEDGE-BASED ACCOUNTING SYSTEMS

Different Modes of Knowledge-Intensive Assistance in the Design and Operation of an AIS

A definition of a concept is also called the *intension* of the concept. The *extension* of a concept consists of all the actual objects the definition applies to. For example, the intension of the object Sale defines the common characteristics of Sale (also known as attributes), the relationships the object Sale is involved in, supertypes and subtypes of Sale, and constraints that apply to Sale. The set of economic transactions to which the Sale definition applies, the actual sales transactions, constitutes its extension. A conceptual schema contains intensional descriptions for objects and relationships among objects in the application domain. This article focuses on the potential of *augmented intensional reasoning* in an accounting environment, that is, on reasoning processes that use intensional structures augmented with domain-specific knowledge. The actual domain-specific augmentation proposed here involves REA classifications. In terms of REA, Sale is an *Economic Event*, and Sale participates in a *Stock-Flow* relationship (with Product), a *Duality* relationship (with Cash Receipt), a *Control* relationship with an *Internal Agent* (Salesperson) and a *Control* relationship with an *External Agent* (Customer). The structuring of the objects in terms of REA primitives adds domain-specific knowledge to the conceptual schema.

There are three modes of knowledge-intensive assistance in the design and operation of accounting systems: (a) routine design and operation of an AIS, (b) knowledge-intensive design and routine operation of an AIS, and (c) knowledge-intensive design and operation of an AIS. The three modes are shown from top to bottom in figure 5 where all three instances follow the left-to-right representation and implementation mapping of an accounting universe of discourse (AUoD) through a design process into a conceptual description and then further through a conclusion materialization process to operational user views. The crosshatched portions in figure 5 indicate knowledge augmentation or assistance. All three modes use methods knowledge, which helps the designer build grammatically correct conceptual descriptions. For example, methods knowledge can be used to report grammatical inconsistencies in an E-R model such as "an entity exists for which no relationship has been specified." Methods knowledge is used now in many CASE tools to analyze conceptual representations (De Troyer et al. 1988; Batini et al. 1992; Booch et al. 1999).

-- INSERT FIGURE 5 HERE --

Routine Design and Operation of an AIS

The design and operation processes in figure 5a are *routine* in the sense that they are unguided by any formal enterprise model (such as REA). No common semantic infrastructure is used for design or operation. This will most likely result in inconsistent representations of similar phenomena across functional boundaries, and the form of the conceptual descriptions will vary from company to company. Further, no taxonomy of shareable and reusable accounting concepts can be built without a common semantic infrastructure. Instead, an artifactual set of procedures or programs will be constructed to meet the reporting and decision needs of individual users.

Knowledge-Intensive Design and Routine Operation of an AIS

In knowledge-intensive design and routine operation (figure 5b), REA is used as a formal enterprise model to *structure* the conceptual description of the accounting object system. The extra crosshatched portion in the design process box represents the use of domain-specific knowledge to assist the design process. CASE tools can support knowledge-intensive design.[4] For example, such a CASE tool would tell the user that an *Economic Event* needs to participate in a *Duality* relationship. The REA model is of great value in developing a consistent conceptual structure for the accounting object system even though the REA structures are employed solely as a starting point for final implementation design. The enterprise schema in figure 5b is neither explicit nor persistent, and the conclusion materialization process will actually use artifactual sets of procedures, just like the routine design approach did. This is a common situation when database technology is used as implementation platform for an AIS. Database technology is not capable of making intensional structures explicit, and thus it is not able to use them for reasoning on an ongoing basis. The domain-specific knowledge that is used to structure the accounting object system is scattered during the actual database design and implementation phases. A common semantic infrastructure does not suffice by itself to accomplish augmented intensional reasoning; technological features are important as well.

Knowledge-Intensive Design and Operation of an AIS

In knowledge-intensive design and operation, domain-specific knowledge consisting of repeated, guided instantiations of the REA model is used as knowledge-intensive assistance during the design process. Figure 5c contains two additional crosshatched portions: the explicitly recorded conceptual schema and the taxonomy of shareable and reusable accounting concepts (declarative descriptions). The crosshatched conceptual structure in the middle corresponds with the enterprise schema in figure 2 and is used to assist both the design process and the conclusion materialization process. The crosshatched portion in the conclusion materialization box in figure 5c contains the REA-based definitions that are part of the larger box in figure 2. The full effect of augmented intensional reasoning is realized during the conclusion materialization process. During routine operation (figures 5a and 5b), all user views are produced entirely with programs, and they do not share generic patterns that are part of the common semantic infrastructure. This is very different from our proposed new ways of thinking in which information retrieval relies heavily on explicitly recorded REA structures. Economic phenomena like claims are materialized first with an intensional pattern match and second with a summation of extensional occurrences like individual amounts of specific sales. The main difference between the architectures in figures 5b and 5c is the ability of the technology used in figure 5c to represent the enterprise schema explicitly and to use it for reasoning on a regular basis. The white portion in the conclusion materialization box in figure 5c represents task-specific knowledge and corresponds with the dotted boxes on top of figure 2.

Aspects of Augmented Intensional Reasoning

Three related aspects of augmented intensional reasoning are: (1) the *procedural-declarative transformations* effected by the domain theory, (2) the importance *of epistemologically-adequate representations*, and (3) the implications resulting from *implementation compromises*. The procedural-declarative transformations make information retrieval in AIS more general and theoretical. Epistemologically-adequate representations and

4. See Geerts et al. (1996) for an enumeration of such CASE possibilities.

implementation compromises are related to the economic feasibility of augmented intensional reasoning. These three subjects warrant individual explanation.

Procedural-Declarative Transformations

"As theory progresses [in a particular field of inquiry], more of the knowledge can be removed from procedures and put in declarative form" (Sowa 1984, 24). With respect to the actual construction of accounting systems, the availability of a domain theory of accounting makes procedural-declarative transformations possible.

Sowa (1984, 23, following Simon 1969) uses the example of constructing a circle to illustrate the consequences of procedural-declarative transformations:

How: To construct a circle, rotate a compass with one arm fixed until the other arm has returned to its starting point.

What: A circle is the locus of all points equidistant from a given point.

Other procedures can be conceptualized for constructing a circle, such as rolling a piece of clay and cutting a cross-section. These procedures illustrate Sowa's (1984) conclusions:

1. Without the declarative description (the what), it is difficult to show how the different procedures (the how) relate to each other.

2. The declarative description covers the different procedures.

Similar conclusions appear when we compare how accounting phenomena like claims are supported by database accounting systems (figure 5b) and knowledge-based accounting systems (figure 5c).

To support claims information in a database accounting system, conclusion materializations (also named view definitions) are needed for the different individual types of claims that may result from the accounting information structure under consideration (e.g., accounts receivable, accounts payable or advance receipts). For each of these, we need to describe precisely how they are materialized from the data. A possible definition for accounts receivable is:

How: Determine trade accounts receivable by subtracting the total amount of the cash receipts from customers from the total amount of sales made by customers.

Some immediate consequences of such an organization are:

1. Modifications in the enterprise schema (extending the object structure or changing the constraint values) may imply a revision of such definitions.

2. The definitions for accounting phenomena like claims are largely application specific.

On the other hand, to support claims information with a knowledge-based AIS such as the ones in figures 2 and 5c, it suffices to describe what a claim is in terms of the domain theory (REA). Such a declarative definition for claims is:

What: A claim with an outside agent exists where there is a flow of resources with that agent without the full set of corresponding instances of a dual flow.[5]

The reasoning component of an AIS may determine the different claim types starting from the declarative description of claim. The actual occurrences of the claim concept for the accounting object system under consideration depend on the enterprise schema. Information about a specific claim type can be obtained from a more precise declarative description. As a result of these procedural-declarative transformations, the extension of claims will vary with modifications in the enterprise schema, and the definitions of accounting phenomena like claims are largely application-independent.

This independence represents an important opportunity for accounting systems designers because it opens up the possibility of developing a general accounting framework consisting of REA-based definitions of phenomena like claim, asset, and activity (Geerts 1993). This general accounting framework is represented by the taxonomy box in figure 2. The concepts that are a part of this framework can be *shared* across functional boundaries and can be *reused* by all accounting information systems that commit to an REA-based semantic infrastructure. The extensions for the different accounting phenomena defined would be determined from the specified enterprise database. A formal measure of an accounting system's ability to support such a scheme is *epistemological adequacy.*[6] We extend this AI notion to accounting next.

Epistemological Adequacy

The result of the knowledge augmentation process described above is that accounting information is derived in a much more general and theoretical manner. The main tasks are developing operational definitions of accounting phenomena in terms of the domain theory and then explicitly describing them. The *feasibility* of augmented intensional reasoning in knowledge-based accounting systems depends on the epistemological adequacy of the enterprise schema.

McCarthy (1987) explained epistemological adequacy in accounting representations as "the ability to represent definitional features of the environment faithfully." Satisfying the feasibility criterion leads to our heuristic for determining such adequacy: *if a representation allows the full extent of intensional reasoning in materializing data-dependent conclusions and in enforcing integrity constraints, its epistemological features are adequate. Anything less means that we strive for a higher degree of representational faithfulness.*

---

5. For simplicity purposes in the example of Figure 4, we assume a maximal cardinality of "1" for the participation of both economic events in the duality relationship.
6. See McCarthy and Hayes (1969) for John McCarthy's original ideas on epistemological adequacy. Lifschitz (1990, 3) characterizes McCarthy's notion as "A representation is *epistemologically* adequate if it can be used to express the facts that can actually be discovered with the available opportunities."

Epistemological adequacy is a very high standard for accounting systems, and even approximate adherence to such a standard is unlikely soon. Impediments to attaining such a standard include the storage structures and data access mechanisms of existing computer systems and the representational difficulties of some REA primitives.

Implementation Compromise

Implementation compromise means taking some modeled component of the real world (gathered during initial systems analysis) and discarding it during actual system construction. In present-day accounting systems, incomplete REA schemas (as in McCarthy 1982, 573-4; Geerts 1993, 69-96; Rockwell and McCarthy 1999, figure 4) are acceptable and even expected from a cost/benefit point of view. The lack of epistemological adequacy will limit the reasoning capabilities of a system or burden its reasoning component with exception handling. Procedural specifications will be needed to handle cases with insufficient knowledge about the underlying phenomena or to deal with exceptions. Let us suppose, for example, that the *Duality* relationship between Sale and Cash Receipt in figure 4 is not explicitly represented. Arguments for the substantive dismissal of this *Duality* link can be found in the case where the company tracks receivables only by amount (i.e., a balance-forward system). While this procedural solution is acceptable from an information systems implementation point of view, the loss of knowledge is extensive. Pattern matching for deriving instantiations for a generally-defined concept like claims becomes hard or impossible. Geerts (1993) contains different examples of situations where a decrease in epistemological adequacy is compensated by extra procedural specifications.

IV. KNOWLEDGE-INTENSIVE AIS DESIGN: THE CREASY ENVIRONMENT

Designing knowledge-based accounting systems with the REA model as the common semantic infrastructure requires a development environment. We have built one for this purpose called CREASY: **C**onceptualizing **REA SY**stems. The environment attempts to support knowledge-intensive design by exploiting both methods knowledge and domain-specific knowledge. Existing CASE tools are inadequate for this purpose because they do not use domain-specific knowledge to support conceptual design (Geerts et al. 1996). This section explains how CREASY employs augmented intensional reasoning in support of knowledge-intensive design and how CREASY refines and integrates augmented intensional structures into applications. The CREASY environment is implemented in Arity Prolog (Arity 1988).

-- INSERT FIGURE 6 HERE --

The first module, supporting **conceptual schema design** (northwest part of figure 6), provides the ability to develop a specification in terms of the E-R model. The module converts the semantic specifications for entities, relationships, attributes, and constraints of an E-R schema to a format manageable by the inference engine. This methods knowledge will be used for different purposes during different phases of system construction and use. During design, it is used to check the grammatical validity of added specifications. At the operational level, it may be used to ensure that the behavioral implications of the static specifications are attainable (Geerts 1993).

The structuring of the accounting object system is supported by the **domain-specific design** module (northeast corner of figure 6), which permits reclassification of all entity types during the conceptual analysis as *Resources, Events,* or *Agents*. Utilizing this extra layer of knowledge augmentation, the system searches for relationships between the different reclassified entities and labels them as *Stock-Flow* relationships, *Control* relationships, or *Duality* relationships. For the detected *Stock-Flow* relationships, an extra classification as *Inflow* or *Outflow* is required. Upon request, CREASY analyzes a specified enterprise schema. Inconsistencies, such as duality relationships connecting two inflows, events not participating in a duality relationship, and resources for which no stock-flow relationship can be found, are reported to the designer for refinement or corrective action. In the CREASY environment, implementation compromises are dealt with procedurally.

With a frame editor, the **operational design** module (southeast part of figure 6) refines the generic frame knowledge and slot definitions. Default values and procedural attachments can be added to specific slots. For frames classified as REA primitives, slot definitions can be extended with role assignments.

The **operational system** module (southwest corner of figure 6) can generate a prototype for the system. A user interface will be built by CREASY according to the defined structure. Concept definitions and necessary procedural code (e.g., for implementation compromises) can be added with the application builder interface. The prototype will support operational activities such as data entry, schema analysis, and query formulation. Each of these activities can rely on the different types of knowledge made explicit throughout the design process.

V. KNOWLEDGE-INTENSIVE AIS OPERATION

Code Components of Knowledge-Based AIS

The operation of a knowledge-based AIS based on our prior definitions is largely reduced to the management of a set of declarative descriptions related to the accounting object system. The application of these descriptions in solving problems is principally the task of the reasoning component of the system. Geerts (1993) provides examples of applications relying on the explicitly-recorded enterprise schema and augmented intensional reasoning. Here, we illustrate a portion of a complete system as proof-of-concept for augmented intensional reasoning and the use of REA accounting as a common semantic infrastructure. The accounting object system is the one portrayed in figure 7.[7] Part of the Prolog code for this system appears in the appendix,[8] organized in five components: the conceptual schema definitions, the accounting-specific classifications, the REA-based definitions of accounting phenomena, the database, and the supportive definitions. Each appendix component is explained with a narrative below. In the first two components, an intensional structure (including its interpretation in terms of the domain theory) is defined. This structure corresponds to the REA-structured enterprise schema (outer ellipse) in figure 2. The third component corresponds to the large box in figure 2 and illustrates declarative and domain-specific programming. The REA-based definitions of accounting phenomena appear as

7. This object system is a combination of a revenue and an acquisition cycle with inside agents not depicted because of space limitations.
8. For the complete set of definitions see Geerts (1993, 184-192).

declarative descriptions in the conclusion materialization process in figure 5c. The database includes the extensional description of the accounting object system (i.e., the actual data occurrences) and corresponds with the inner ellipse in figure 2. Finally, the fifth part reflects the programming code needed to make intensional reasoning work and corresponds to the cylinder in figure 2.

-- INSERT FIGURE 7 HERE --

Conceptual Schema Definition

The conceptual schema definitions contain an explicit representation of the E-R schema elements: the intensional structure. The elements are represented as Prolog clauses (facts), and they may be considered as the output of CREASY's conceptual schema design module.

Entity type and attribute type definitions are self-explanatory. In the three-argument relationship definition (relationship/3),[9] the first argument expresses the relationship name, the second argument expresses the list of entity types involved in the relationship (only binary relationships are included), and the third argument expresses a unique identifier for the whole relationship. This identifier is used as a surrogate for the actual identification, which consists of relationship name and participating entity types. Its use simplifies the implementation.

Constraints are also declaratively specified (constraint/4). The first data value (third argument) expresses the minimal cardinality constraint; the second data value (fourth argument) designates the maximal cardinality constraint. The second constraint clause for instance "constraint(sale,r1,1,1)" expresses that "a Sale occurrence appears exactly once in the relationship r1" or alternatively "exactly one Customer is involved in a Sale." Constraint specifications for attributes have been omitted.

We also record explicitly the attributes used for identification (id) purposes. For example, customer_nr identifies customers (description(customer,customer_nr,id)). These clauses are examples of role declarations.

Some uses of the explicitly-represented conceptual schema elements are:

1. At the design level, they can be used for consistency checking in terms of methods knowledge which occurs in CREASY's conceptual schema design module.

2. They can be used in support of a data dictionary system. Queries like: "Which entities are specified?", "Which data elements are used to describe an entity?", and "To which constraints is the participation of an entity in a relationship subjected?" can be answered easily. An example of such a query is illustrated below where the first line is the natural language query, the second line is the query in Prolog, and the last three lines are answers from the system.

   ```
   ** Which data elements are used to describe sales?
   ?- attribute(sale,AttributeName).
   AttributeName = sale_nr;
   ```

9. In Prolog, the notation "clause-name/number-of-arguments" is used to refer to clause definitions. Thus we have five different definitions in the first subsection of the appendix: entity/1, relationship/3, attribute/2, constraint/4, and description/3.

AttributeName = sale_amount;
AttributeName = sale_date;

3. At the operational level, they can be used to determine the behavioral implications of constraint definitions, e.g., "Which data elements are mandatory for a valid instance of an entity?"

Accounting-Specific Classifications

Entity types and relationship types are classified in terms of generic classes whose specific roles were explained in section II: {*Resource, Event, Agent, Stock-Flow, Duality, Control*}. For *Economic Events*, flow direction must be specified as well: {*Inflow,Outflow*}. The consistency of these definitions must be ensured at the design level, checking that is performed by CREASY's domain-specific design module.

Other types of accounting-specific classifications exist that are excluded from the code in the appendix. One of the most important of these is the assignment of specific roles to attributes. For the materialization of accounting reports, knowledge of the relationship between accounting reporting items and the actual data is required. This knowledge can be made explicit by the assignment of specific roles to attributes. Financial accounting applications, for example, require at least amount and recording date attributes for economic events. For product costing purposes, cost amount declarations must be specified. A wide range of applications relying on role declarations is illustrated in Geerts (1993). Such accounting-specific role declarations can be added by CREASY's operational design module.

REA-Based Definitions of Accounting Phenomena

A limited set of REA-based definitions of accounting phenomena is included in the knowledge base. The definitions for *Control, Stock-Flow*, and *Duality* relationships are similar and combine object definitions and REA classifications. The *Control* relationship is defined as:

1. A relationship that has been classified as being of the REA-type ***Control***, and
2. For which one of the entity types participating in the relationship has been classified as being of the REA-type ***Agent***, and
3. For which the other participating entity type has been classified as being of the REA-type ***Event***.

An exchange combines an *Inflow Economic Event* and an *Outflow Economic Event* by means of a *Duality* relationship with the same external *Agent* involved. In Prolog, the latter condition is ensured by using the same variable name (*Agent*) for the external *Agent* in both control definitions. Exchange is a declarative definition that combines other definitions.

These REA-based definitions of accounting phenomena are largely independent of specific accounting object systems. This makes them useful for developing a reusable accounting framework based on concepts like *Duality* relationships and exchanges. They may be embedded in the knowledge base as standard concepts on which further applications may be built. These standard concepts are represented as shareable and reusable concepts (the large box) in figure 2. The extension of these concepts (i.e., "the set of all existing things to which

the concept applies" (Sowa 1984, 11)) depends on the actual conceptual description of an accounting object system.

These definitions can be used to draw conclusions from the explicitly-recorded accounting information structure. The application of augmented intensional reasoning for semantic information retrieval is illustrated by the following queries.[10]

```
** In which economic events are vendors involved?
?- control(Event,_,vendor).
   Event = cash_disbursement;
   Event = purchase;

** Which events may lead to spending?
?- duality(EventI,_,cash_disbursement).
   EventI = purchase;

** Which events affect the availability of product?
?- stockflow(Event,_,product).
   Event = sale;
   Event = purchase;
```

Such queries can be posed using the user-interface of CREASY's operational system module.

The Database

While the three previous components contained general statements defining the accounting object system, the database component illustrates how concrete data occurrences or tokens are stored in the database. The database is a set of value facts. The value/4 predicate describes both entity type instantiations and relationship type instantiations.

For entity type instantiations, the arguments of the value/4 predicate are entity type name, identification value of the entity occurrence, attribute name, and value. For simplicity, the id attribute value identifies entity instances.

For relationship type instantiations, the arguments of the value/4 predicate are surrogate identifier of the relationship type, identification value of the relationship type occurrence, name of the participating entity type, and the id attribute value of the participating entity type occurrence.

Supportive Definitions

Extra clauses, necessary for making the system work, have also been specified in the appendix. Four supportive predicates employed for interpreting the actual data in terms of REA-based accounting definitions (the intensional-extensional link) are id/2, occurrence/2, part/4 and relpart/5.

id/2 Determines the attribute name used to identify particular occurrences of a real world object.

10. The outcomes of these queries are based on the actual implementation of Figure 7.

occurrence/2 Determines the extension of a particular entity type in terms of the identification attribute.

part/4 Determines, depending on the existence of a corresponding value, either (a) the relationship type in which an entity instance is expected to occur or (b) the relationship occurrence in which an entity instance actually occurs.

relpart/5 Determines the entity type occurrences participating in a specific relationship type occurrence.

A fifth supportive definition (dualrel/3), unrelated to the intensional-extensional link, has also been included. The previously mentioned duality/3 relationship imposes a fixed inflow/outflow pattern of "duality(Inflow,_,Outflow)." The dual/3 relationship is similar to the duality relationship with the exception that the fixed order is not imposed. The invocation of dualrel/3 as a substitute for duality/3 is situation specific.

Claim Materialization

Claim Definition

We illustrate operation of a knowledge-based accounting system by showing the effect of augmented intensional reasoning on the materialization of "claim with external party" information. An accounting concept like claim with outside party is implemented by adding the REA-based definition of the concept to the knowledge base (the taxonomy of shareable and reusable concepts box in figure 2). The following Prolog clause defines the claim concept:

```
** Generic Definition for the Claim concept.
claim(Agent,AgentValue,Event,EventValue,FutureEvent) :-
    dualrel(Event,Dual,FutureEvent),
    part(Event,Dual,DualValue,EventValue),
    not relpart(Event,EventValue,Dual,FutureEvent,
            FutureEventValue),
    control(Event,Control,Agent),
    relpart(Event,EventValue,Control,Agent,AgentValue).
```

The arguments of the claim/5 predicate are a mixture of intensional (type) and extensional (occurrence) elements. The three intensional elements are Agent, Event, and FutureEvent. The extensional arguments are values for Agent (AgentValue) and Event (EventValue). The values are those of the identification attribute of the entity types (Agent and Event). The different elements of the definition are explained next.

dualrel(Event,Dual,FutureEvent).

Intensional. The explicitly-recorded accounting information structure (intension) is consulted to determine the *Economic Event* entity types (Event) that are involved in a *Duality* relationship (Dual).

part(Event,Dual,DualValue,EventValue).

not relpart(Event,EventValue,Dual,FutureEvent,
FutureEventValue).

Extensional. The part/4 predicate determines all Event occurrences that are assigned to the Dual relationship. The relpart/5 predicate determines Event occurrences that are part of completed exchanges. Negation (not) leads to the set of unrequited Event occurrences. In combination, these two predicates determine the subset of Event occurrences that are assigned to the Dual relationship and are not yet linked to their give-take dual. They represent partially-performed exchanges with FutureEvent as terminator.

control(Event,Control,Agent).

Intensional. The control/3 predicate is used to determine the Control relationship type in which the Event type under consideration is involved and the Agent type involved in the same relationship.

relpart(Event,EventValue,Control,Agent,AgentValue).

Extensional. The relpart/5 predicate determines the actual instantiation of the Agent type corresponding with the Event type occurrence.

The claim concept definition can be improved by capturing more domain knowledge as part of the definition, which will improve the efficiency of Prolog's inference engine. For situations where the participation of the Event type in the Dual relationship is constrained to a [1,1] relationship for example, claims cannot occur. This means that receivables or payables will not occur in situations constrained to cash transactions. Making this knowledge explicit increases efficiency because the system can avoid searches for which no result can exist. The improved definition for claim is:

```
** Improved Definition for the Claim Concept.
claim(Agent,AgentValue,Event,EventValue,FutureEvent) :-
    dualrel(Event,Dual,FutureEvent),
    not constraint(Event,Dual,1,1),
    part(Event,Dual,DualValue,EventValue),
    not relpart(Event,EventValue,Dual,FutureEvent,
                FutureEventValue),
    control(Event,Control,Agent),
    relpart(Event,EventValue,Control,Agent,AgentValue).
```

The claim definition can be extended to distinguish between positive and negative claims. A declarative description for the positive claim concept is:

> *A positive claim with an outside party exists where there is an outflow of resources without a full set of corresponding instances of a dual flow.*

Positive claims are the subset of claims where the initiator *Event* effects an *Outflow* of *Resources*. In Prolog, this definition can be expressed as:

```
positive_claim(Agent,AgentValue,Event,EventValue,FutureEvent) :-
```

rea(Event,outflow),
claim(Agent,AgentValue,Event,EventValue,FutureEvent).

Execution of the Claim Definition

The claim/5 predicate describes partially-performed exchanges. The extension of the claim concept for the actual implementation of figure 7 is illustrated in table 1.[11] The gray-colored row in table 1 emphasizes the mixture of intensional and extensional elements in semantic information retrieval.

-- INSERT TABLE 1 HERE --

This table mixes four claim types:

- An *Inflow* of Cash (Event: cash_receipt) with no corresponding *Outflow* of Products (FutureEvent: sale)
- An *Inflow* of Product (Event: purchase) with no corresponding *Outflow* of Cash (FutureEvent: cash_disbursement)
- An *Outflow* of Product (Event: sale) with no corresponding *Inflow* of Cash (FutureEvent: cash_receipt).
- An *Outflow* of Cash (Event: cash_disbursement) with no corresponding *Inflow* of Product (FutureEvent: purchase).

Rather than being explicitly defined, claim types are determined by the inference engine based on the explicitly-represented accounting information structure. Claim types are added or removed by modifying this structure. If the example included the payroll cycle, the inference engine could recognize an incomplete exchange between Labor Service Acquisition and Cash Disbursement as an additional instance of claim (wages payable).

Actual claims are instantiations of claim types. The extension of a claim type varies as a result of database modifications. For example, inserting the 'value(r3,r32,cash_receipt,'0003')' fact to the database would cause the fourth row in table 1 to be removed.

The claim definition can be refined further. We can ask, for instance, for all actual instantiations of a particular claim type. The example in table 2 shows the Product *Outflow* (Sale) with no corresponding *Inflow* of Cash (Cash Receipt).

-- INSERT TABLE 2 HERE --

Even more precision can be obtained by assigning a value to the AgentValue variable. For instance, the example in table 3 shows all Product *Outflows* to Customer '0002' with no corresponding *Inflow* of Cash.

-- INSERT TABLE 3 HERE --

11. The complete set of data values used in our examples is given in Geerts (1993).

The execution of these definitions generates the substance for conclusion materializations of items like accounts receivable and accounts payable. Actual implementations for these items have been specified in Geerts (1993).

The examples in this section illustrate shareability and reusability of accounting concepts across functional boundaries. Using generic accounting concepts for semantic information retrieval requires the REA-based semantic infrastructure, epistemological adequacy for the enterprise schema, and use of augmented intensional reasoning. Lack of a common semantic infrastructure would make it impossible to define accounts receivable and accounts payable as instantiations of the more generic accounting concept claim. The definition of claim illustrates the importance of epistemological adequacy. The definition would not recognize accounts receivable as an instance of claim in cases where the *Dual* relationship between Sale and Cash Receipt is not explicitly represented. As a result, this representation cannot share the generic concept, which means that an extra procedure would be required to support accounts receivable, e.g., determine accounts receivable by subtracting the total amount of the cash receipts from customers from the total amount of sales made by customers.

Augmented intensional reasoning glues components together by using the enterprise schema definitions as data in combination with the knowledge base containing REA primitives and a taxonomy of accounting concepts. The extensional-intensional link supported by augmented intensional reasoning makes knowledge-based accounting systems as represented in figure 2 capable of semantic information retrieval. The knowledge base represented as the REA-based semantic infrastructure in the boxes in figure 2 is reusable in all AIS in different companies and different sectors that commit to the REA-based common semantic infrastructure. Practically, that implies that the enterprise schema should be structured in terms of the REA model and that the REA classifications should be accessible by the intensional reasoning component.

VI. CONCLUSIONS AND IMPLICATIONS FOR RESEARCH

We have demonstrated the knowledge-intensive use of REA for knowledge sharing and reuse in an accounting context through augmented intensional reasoning. Augmented intensional reasoning in knowledge-based accounting systems has important benefits: the design and implementation of AIS becomes less time consuming and less costly, some of the design tasks can be done by the system itself, and accounting information is derived in a much more general and theoretical manner. Limitations of augmented intensional reasoning result primarily from the implementation technology currently available. The technology must be capable of explicitly recording the enterprise schema and explicitly reasoning with it. Such technology is not yet, however, supported and implemented on a widespread basis.[12] Further, the feasibility of augmented intensional reasoning depends largely on the epistemological adequacy of the enterprise schema. As explained in section III, compromised REA structures are often acceptable and even expected from a cost/benefit point of view. Advances in information technology such as faster processing/storage and improved data capture are the

12. For progress on overcoming technology constraints for REA-based systems, see Nakamura and Johnson (1998).

main drivers for future increases in the epistemological adequacy of enterprise schemas. For these reasons, progress towards economically-feasible implementations of knowledge-based accounting systems strongly depends on further advances in information technology.

For many of the ideas presented in this paper, much more research is needed. Two possible research directions are: (1) refining and extending the accounting taxonomy of shareable and reusable concepts, and (2) addressing the multiple dimensions of knowledge sharing and reuse.

The extent of knowledge sharing and reuse depends largely on the depth of the taxonomy. Future work could look at declarative knowledge that can be shared across a variety of applications. The Resource-Event-Agent model has served as semantic infrastructure for different types of information systems: AIS (Gal and McCarthy 1986), decision support systems (Denna and McCarthy 1987), manufacturing information systems (Grabski and Marsh 1994), supply chain management (Haugen 1997), and value chain analysis (Geerts and McCarthy 1997; 1999). In addition to accounting-oriented concepts such as claim, exchange, and asset, non-accounting REA-based concepts across the value chain, such as business process, are required.

Semantic interoperability is only one dimension of a common knowledge architecture. Researchers in the field of ontological engineering address many other dimensions including the heterogeneity of representation formalisms and the heterogeneity of implementation platforms. To overcome such hurdles, they design declarative, expressive, semantically well-defined, and machine-readable grammars, which are independent of any particular data modeling formalism or machine-readable language (Gomez-Perez 1998). Examples of such grammars are CYC (Lenat and Guha 1990) and Ontolingua (Gruber 1993). Future research work could include the coding of REA primitives and the REA-based taxonomy of shareable and reusable concepts with such grammars. With sufficient extensions this work might permit REA to be considered a candidate for a full domain ontology in the manner specified by Guarino (1998).

REFERENCES

Arity Corporation. 1988. *The Arity Prolog Language Reference Manual*. Concord, MA: Arity Corporation.

Batini, C., S. Ceri, and S. B. Navathe. 1992. *Conceptual Database Design. An Entity-Relationship Approach*. Redwood City, CA: Benjamin/Cummings.

Bernus, P., K. Mertins, and G. Schmidt, eds. 1998. *Handbook on Architectures of Information Systems.* Berlin: Springer-Verlag.

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

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

Davenport, T. H. 1998. Putting the enterprise into the enterprise system. *Harvard Business Review* (July): 121-131.

Denna, E. L., and W. E. McCarthy. 1987. An events accounting foundation for DSS implementation. In *Decision Support Systems: Theory and Application*, edited by C. W. Holsapple, and A. B. Whinston, 239-263. Berlin: Spinger-Verlag.

De Troyer, O., R. Meersman, and P. Verlinden. 1988. RIDL* on the CRIS case: a workbench for NIAM. Internal report, Infolab Tilburg, The Netherlands.

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

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

_____, and W. E. McCarthy. 1992. The extended use of intensional reasoning and epistemologically adequate representations in knowledge-based accounting systems. *Proceedings of the Twelfth International Workshop on Expert Systems and Their Applications*, Avignon, France.

_____, and _____. 1994. The economic and strategic structure of REA accounting systems. *300th Anniversary Program*, Martin Luther University, Halle-Wittenberg, Germany.

_____, and _____. 1997. Modeling business enterprises as value-added process hierarchies with resource-event-agent object templates. In *Business Object Design and Implementation*, edited by J. Sutherland and D. Patel, 94-113. London: Springer-Verlag.

_____, and _____. 1999. An accounting object infrastructure for knowledge-based enterprise models. *IEEE Intelligent Systems & Their Applications* (July/August): 89-94.

_____, _____, and S. R. Rockwell. 1996. Automated integration of enterprise accounting models throughout the systems development life cycle. *International Journal of Intelligent Systems in Accounting, Management and Finance* (September): 113-128.

Gomez-Perez, A. 1998. Knowledge sharing and reuse. In *The Handbook of Applied Expert Systems*, edited by J. Liebowitz, chapter 10. Boca Raton: CRC Press.

Grabski, S. V., and R. J. Marsh. 1994. Integrating accounting and manufacturing information systems: An ABC and REA-based approach. *Journal of Information Systems* (Fall): 61-80.

Gruber, T. 1993. A translational approach to portable ontologies. *Knowledge Acquisition* 5 (2): 199-220.

Guarino, N. 1998. Formal ontology and information systems. *Proceedings of International Conference On Formal Ontology in Information Systems*, Trento, Italy.

Haugen, R. 1997. Radically distributed supply chain systems. In *OOPSLA'97 Workshop on Business Object Design and Implementation III*, edited by J. Sutherland, Atlanta, GA.

Hayes-Roth, F. 1997. Artificial intelligence. What works and what doesn't? *AI Magazine* (Summer): 99-113.

Lenat, D. B., and R .V. Guha. 1990. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Reading, MA: Addison-Wesley.

Lifschitz, V., ed. 1990. *Formalizing Common Sense: Papers by John McCarthy*. Norwood, N.J.: Ablex Publishing Company.

McCarthy, J., and P. J. Hayes. 1969. Some philosophical problems from the viewpoint of artificial intelligence. In *Machine Intelligence 4*, edited by B. Meltzer and D. Michie. Edinburgh University Press.

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

_____. 1982. The REA accounting model: a generalized framework for accounting systems in a shared data environment. *The Accounting Review* (July): 554-78.

_____. 1987. On the future of knowledge-based accounting systems. *The D. R. Scott Memorial Lecture Series*. The University of Missouri.

McGowan, M. 1996. Implementation and acceptance of expert systems by auditors. In *Proceedings of the 1996 University of Kansas Audit Symposium*, edited by M. Etteridge. The University of Kansas.

Musen, M. A. 1992. Dimensions of knowledge sharing and reuse. *Computers and Biomedical Research* 25: 435-467.

Nado, R., M. Chams, J. Delisio, and W. Hamscher. 1996. Comet: An application of model-based reasoning to accounting systems. *AI Magazine* (Winter): 55-64.

Nakamura, H., and R.E. Johnson. 1998. Adaptive framework for the REA accounting model. In *OOPSLA'98 Business Object Workshop IV*, edited by J. Sutherland, Vancouver.

Neches, R., R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout. 1991. Enabling technology for knowledge sharing. *AI Magazine* (Winter): 36-56.

Rockwell, S. R., and W. E. McCarthy. 1999. REACH: Automated database design integrating first-order theories, reconstructive expertise, and implementation heuristics for accounting information systems. *International Journal of Intelligent Systems in Accounting, Management, and Finance* (September): 181-97.

Rolstatdas, A. 1999. *Proceedings of the 1999 IFIP International Enterprise Modeling Conference*, Verdal, Norway.

Shpilberg, D., and L.E. Graham. 1986. Developing ExperTAX: An expert system for corporate tax accrual and planning. *Auditing: A Journal of Practice and Theory* (1): 75-94.

Simon, H. A. 1969. *The Sciences of the Artificial*. Cambridge, MA: MIT Press.

Sowa, J. 1984. *Conceptual Structures. Information Processing in Mind and Machine*. Reading, MA: Addison-Wesley.

Winston, P. 1992. *Artificial Intelligence*. Reading, MA: Addison-Wesley.

APPENDIX
A Knowledge-Based Accounting System Implementation

1. Conceptual Schema Definition

Entity Definitions.

```
entity(customer).
entity(product).
entity(cash).
entity(sale).
entity(cash_receipt).
```

Relationship Definitions.

```
relationship(to,[sale,customer],r1).
relationship(of,[cash_receipt,customer],r2).
relationship(involved_in,[sale,cash_receipt],r3).
relationship(of,[cash,cash_receipt],r4).
relationship(of,[sale,product],r5).
```

Attribute Definitions.

```
attribute(customer,customer_nr).
attribute(customer,customer_name).
attribute(product,product_nr).
attribute(cash,cash_type).
attribute(sale,sale_nr).
attribute(sale,sale_amount).
attribute(sale,sale_date).
attribute(cash_receipt,cash_receipt_nr).
attribute(cash_receipt,cash_receipt_amount).
attribute(cash_receipt,cash_receipt_date).
```

Constraint Definitions.

```
constraint(customer,r1,0,n).
constraint(sale,r1,1,1).
constraint(customer,r2,0,n).
constraint(cash_receipt,r2,1,1).
constraint(sale,r3,0,1).
constraint(cash_receipt,r3,0,1).
constraint(cash,r4,1,n).
constraint(cash_receipt,r4,1,1).
constraint(sale,r5,1,1).
constraint(product,r5,0,n).
```

Role Definitions.

```
description(customer,customer_nr,id).
```

```
description(product,product_nr,id).
description(cash,cash_type,id).
description(sale,sale_nr,id).
description(cash_receipt,cash_receipt_nr,id).
```

2. Accounting-Specific Classifications

REA Classifications.

```
rea(sale,event).
rea(cash_receipt,event).
rea(customer,agent).
rea(product,resource).
rea(cash,resource).
rea(r1,control).
rea(r2,control).
rea(r3,duality).
rea(r4,stockflow).
rea(r5,stockflow).
rea(sale,outflow).
rea(cash_receipt,inflow).
```

3. REA-Based Definitions of Accounting Phenomena

```
control(Event,Relationship,Agent) :-
        relationship(Name,EntityList,Relationship),
        rea(Relationship,control),
        rea(Event,event), member(Event,EntityList),
        rea(Agent,agent), member(Agent,EntityList).

stockflow(Event,Relationship,Resource) :-
        relationship(Name,EntityList,Relationship),
        rea(Relationship,stockflow),
        rea(Event,event), member(Event,EntityList),
        rea(Resource,resource), member(Resource,EntityList).

duality(IEvent,Relationship,OEvent) :-
        relationship(Name,EntityList,Relationship),
        rea(Relationship,duality),
        rea(IEvent,event),
        rea(IEvent,inflow),member(IEvent,EntityList),
        rea(OEvent,event),
        rea(OEvent,outflow),member(OEvent,EntityList).

exchange(IResource,IEvent,Duality,OEvent,OResource,Agent) :-
        stockflow(IEvent,_,IResource),
        control(IEvent,_,Agent),
```

duality(IEvent,Duality,OEvent),
control(OEvent,_,Agent),
stockflow(OEvent,_,OResource).

4. Database

```
value(customer,'0001',customer_nr,'0001').
value(customer,'0001',customer_name,'Mead').
value(customer,'0002',customer_nr,'0002').
value(customer,'0002',customer_name,'Grabski').

value(sale,'0001',sale_nr,'0001').
value(sale,'0001',sale_amount,1000).
value(sale,'0001',sale_date,'07/10/95').
value(sale,'0002',sale_nr,'0002').
value(sale,'0002',sale_amount,1000).
value(sale,'0002',sale_date,'07/11/95').
value(sale,'0003',sale_nr,'0003').
value(sale,'0003',sale_amount,1250).
value(sale,'0003',sale_date,'07/13/95').

value(cash_receipt,'0001',cash_receipt_nr,'0001').
value(cash_receipt,'0001',cash_receipt_amount,1000).
value(cash_receipt,'0001',cash_receipt_date,'07/11/95').
value(cash_receipt,'0002',cash_receipt_nr,'0002').
value(cash_receipt,'0002',cash_receipt_amount,1250).
value(cash_receipt,'0002',cash_receipt_date,'07/12/95').

value(r1,r11,sale,'0001').
value(r1,r11,customer,'0001').
value(r1,r12,sale,'0002').
value(r1,r12,customer,'0001').
value(r1,r13,sale,'0003').
value(r1,r13,customer,'0002').

value(r2,r21,customer,'0001').
value(r2,r21,cash_receipt,'0001').
value(r2,r22,customer,'0001').
value(r2,r22,cash_receipt,'0002').

value(r3,r31,sale,'0001').
value(r3,r31,cash_receipt,'0001').
value(r3,r32,sale,'0002').
value(r3,r33,cash_receipt,'0002').
value(r3,r34,sale,'0003').
```

5. Supportive Definitions

```
id(Object,IdAttribute) :- description(Object,IdAttribute,id).

occurrence(Object,Value) :-
        id(Object,IdAttribute),
        value(Object,_,IdAttribute,Value).

part(Object,Relationship,RelationshipValue,Value) :-
        occurrence(Object,Value),
        value(Relationship,RelationshipValue,Object,Value).

relpart(Object1,Object1Value,Relationship,Object2,Object2Value) :-
        part(Object1,Relationship,RelationshipValue,Object1Value),
        part(Object2,Relationship,RelationshipValue,Object2Value),
        not Object1 == Object2.

dualrel(Event,Dual,OtherEvent) :- duality(Event,Dual,OtherEvent).
dualrel(Event,Dual,OtherEvent) :- duality(OtherEvent,Dual,Event).
```

FIGURE 1
Knowledge-Based Accounting Systems: Stand-Alone Intelligent Applications

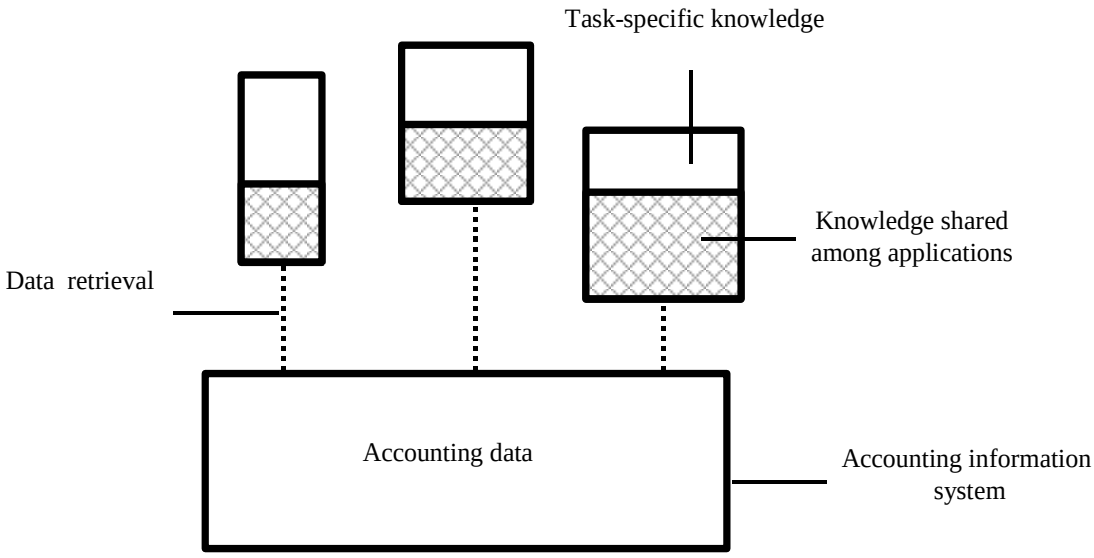


FIGURE 2
Knowledge-Based Accounting Systems: REA-Based Semantic Infrastructure

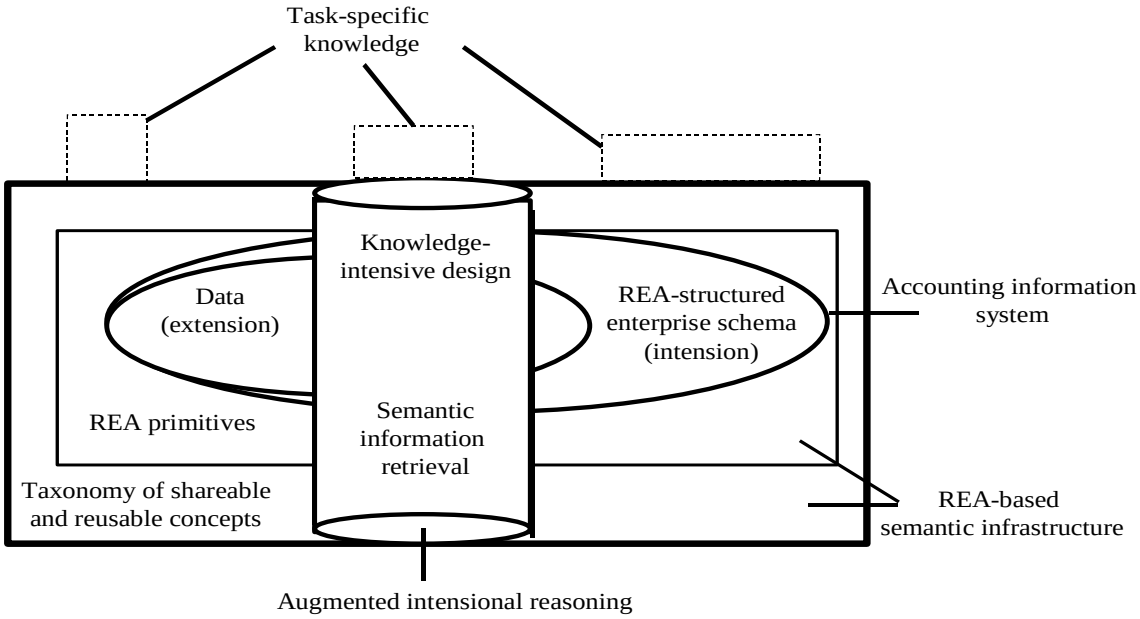


FIGURE 3
The REA Model

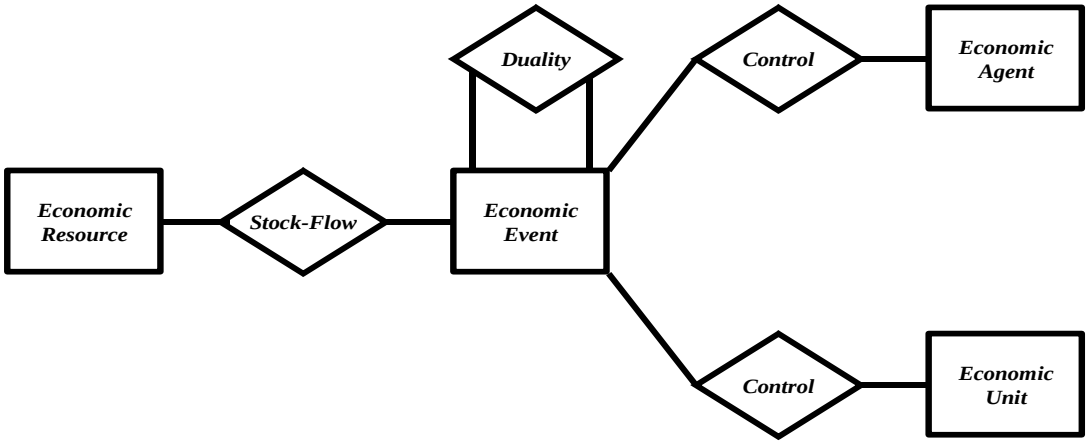


FIGURE 4
REA Instantiation

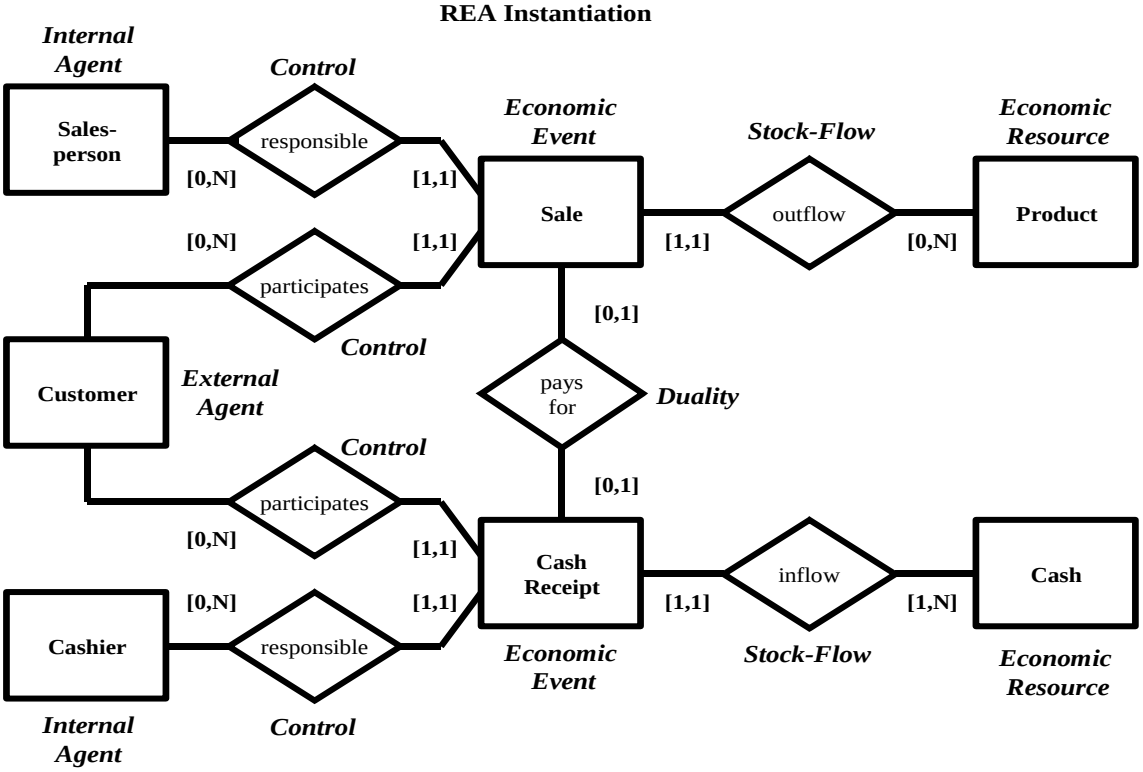
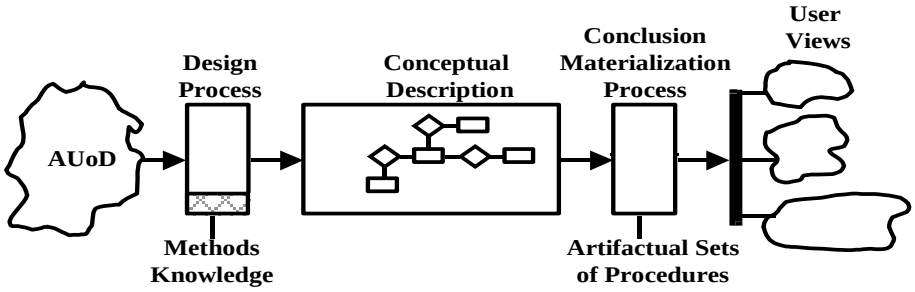
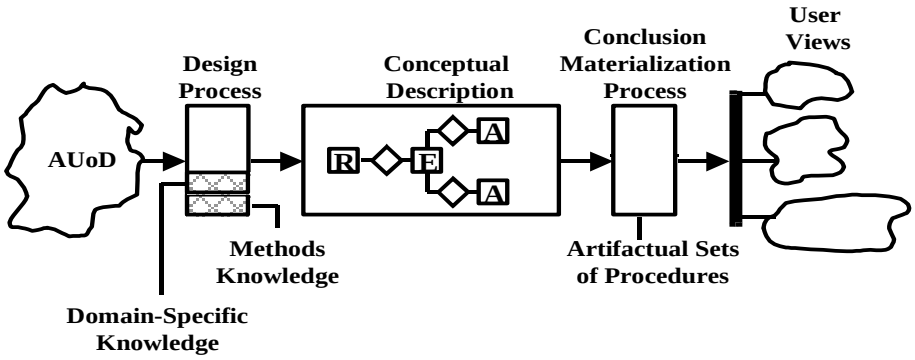


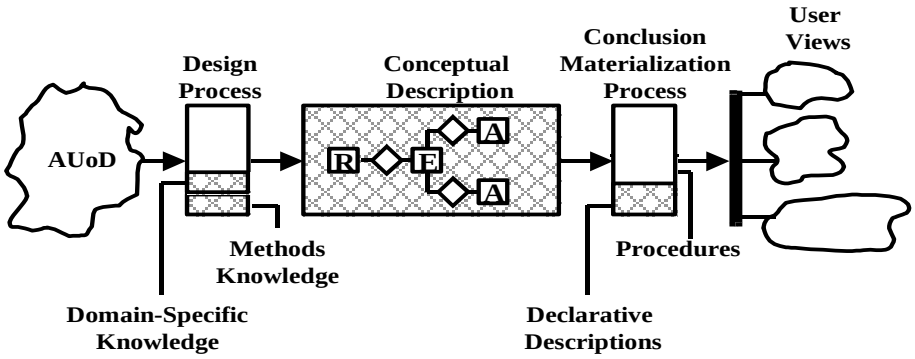
FIGURE 5
Knowledge-Intensive Assistance in the Design and Operation of AIS



a. Routine Design and Operation of an AIS



b. Knowledge-Intensive Design and Routine Operation of an AIS



c. Knowledge-Intensive Design and Operation of an AIS

FIGURE 6
Conceptual Structure of the CREASY Environment

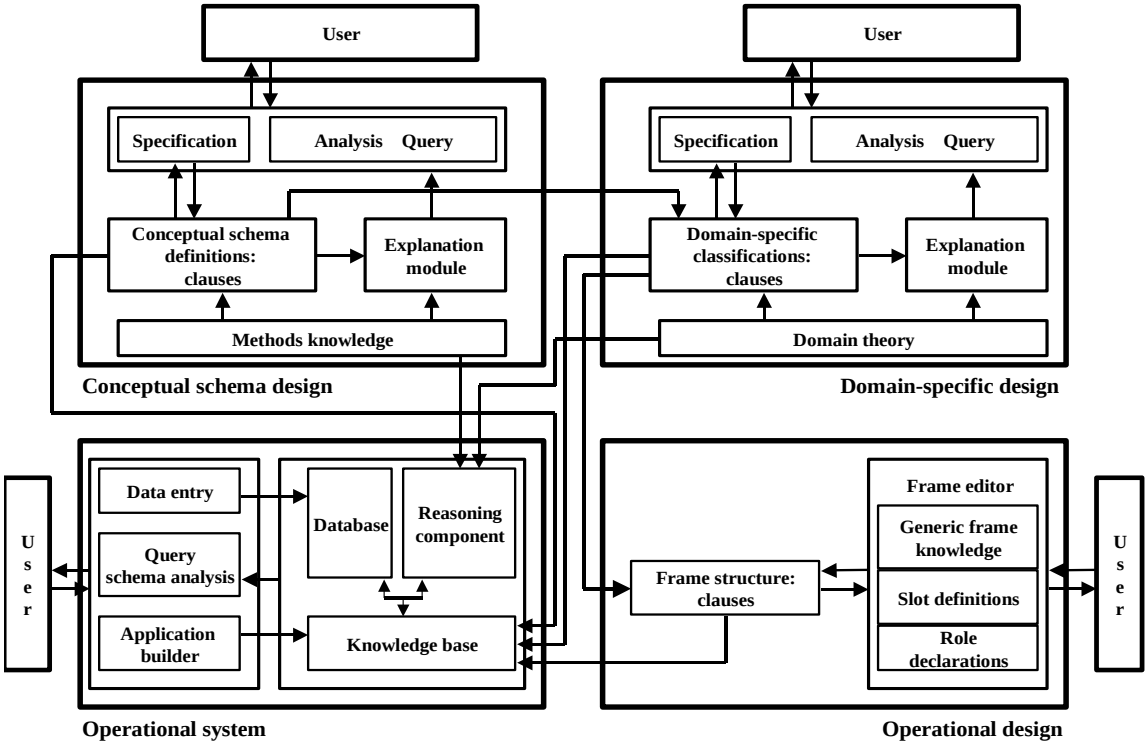


FIGURE 7
REA Example

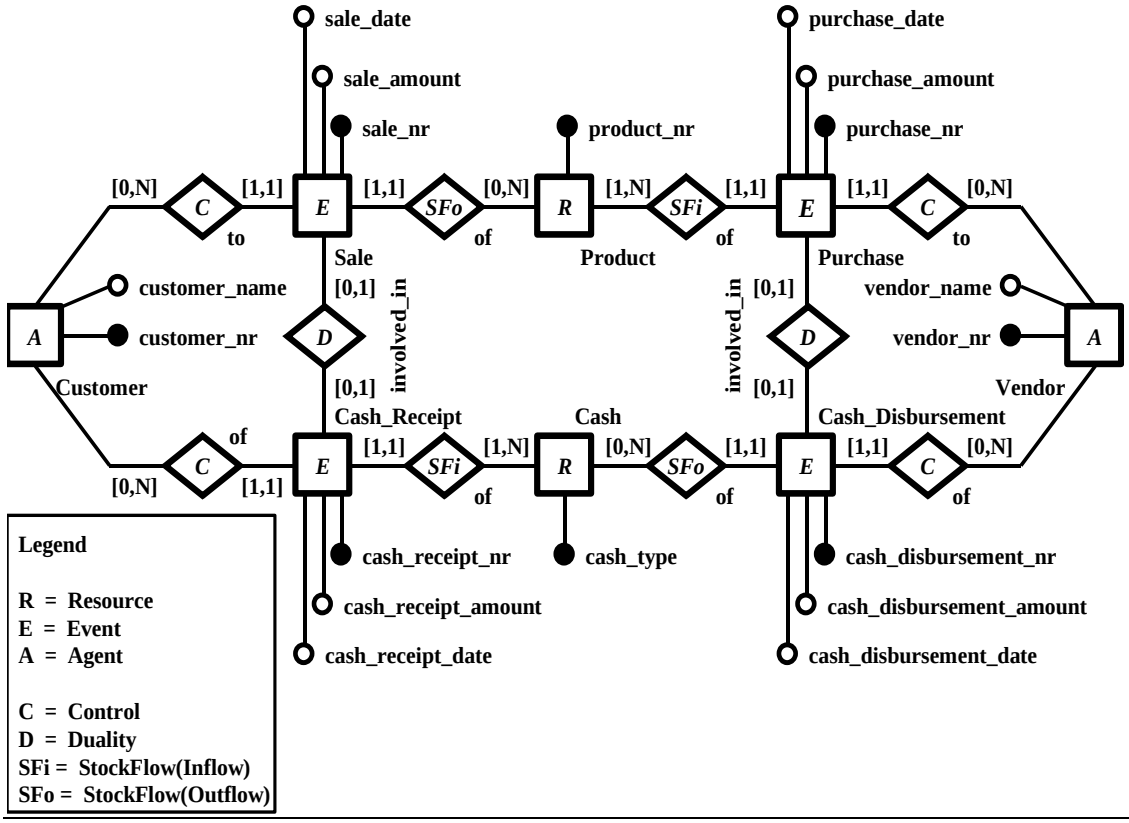


TABLE 1
Claim Type Extension (a)

** Claim Type Extension
?- claim(Agent,AgentValue,Event,EventValue,FutureEvent).

Agent	**AgentValue**	**Event**	**EventValue**	**FutureEvent**
Intension	Extension	Intension	Extension	Intension
customer	'0001'	cash_receipt	'0002'	sale
vendor	'0001'	purchase	'0002'	cash_disbursement
vendor	'0001'	purchase	'0003'	cash_disbursement
customer	'0001'	sale	'0002'	cash_receipt
customer	'0002'	sale	'0003'	cash_receipt
vendor	'0002'	cash_disbursement	'0002'	purchase

TABLE 2
Claim Type Extension (b)

** Claim Type Extension, Sale → Cash_Receipt.
?- claim(customer,AgentValue,sale,EventValue,cash_receipt).

Agent	AgentValue	Event	EventValue	FutureEvent
customer	'0001'	sale	'0002'	cash_receipt
customer	'0002'	sale	'0003'	cash_receipt

TABLE 3
Claim Type Extension (c)

** Claim Type Extension, Sale → Cash_Receipt, Customer = '0002'.
?- claim(customer,'0002',sale,EventValue,cash_receipt).

Agent	AgentValue	Event	EventValue	FutureEvent
customer	'0002'	sale	'0003'	cash_receipt