1.1 What Is REA?

There are several concepts that are present in almost all business software applications. Understanding these concepts makes it much easier to design business applications, to ensure that they do not violate the domain rules, and to adapt the applications to changing requirements without the need to change the overall architecture.

These concepts are known as REA (Resources, Events, Agents). Fig. 1 illustrates the most fundamental REA concepts, which are economic resource, economic agent, economic event, commitment, and contract.

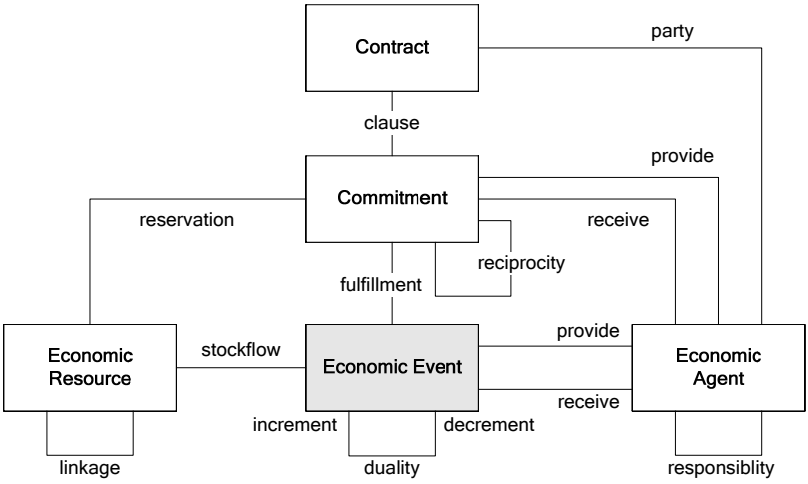

Fig. 1. Fundamental REA concepts

Economic Resource is a thing that is scarce, and has utility for economic agents, and is something users of business applications want to plan, monitor, and control. Examples of economic resources are products and services, money, raw materials, labor, tools, and services the enterprise uses.

Economic Agent is an individual or organization capable of having control over economic resources, and transferring or receiving the control to or from other individuals or organizations. Examples of economic agents are customers, vendors, employees, and enterprises. The *enterprise* is an economic agent from whose perspective we create the REA model.

Economic Event represents either an increment or a decrement in the value of economic resources that are under the control of the enterprise. Some economic events occur instantaneously, such as sales of goods; some

occur over time, such as rentals, labor acquisition, and provision and use of services.

Commitment is a promise or obligation of economic agents to perform an economic event in the future. For example, line items on a sales order represent commitments to sell goods.

Contract is a collection of increment and decrement commitments and terms. Under the conditions specified by the terms, a contract can create additional commitments. Thus, the contract can specify what should happen if the commitments are not fulfilled. For example, a sales order is a contract containing commitments to sell goods and to receive payments. The terms of the sales order contract can specify penalties (additional commitments) if the goods or payments have not been received as promised.

REA also specifies the domain rules assuring soundness and consistency of business software applications from the business perspective. There are several other approaches attempting to describe the fundamental modeling entities, such as *archetypes* (Coad, Lefebvre, DeLuca. 1999) and *pleomorphs* (Arlow, Neustadt 2003), for the business domain, and many *business patterns* on more detailed levels; our favorite books include (Fowler 1996), (Hay 1996), (Silverstone 1997), (Marshall 2000), and (Evans 2003). The patterns and modeling entities described in these books can be expressed in terms of the REA concepts. These patterns are more specific, as they focus on certain subdomains within the business domain. They provide for further concepts, but do not change the concepts defined in the REA. Therefore, REA defines a ubiquitous language for business systems.

The fundamental idea of the REA model is

> If an enterprise wants to increase the total value of resources under its control, it usually has to decrease the value of some of its resources.

An enterprise can increase or decrease the value of its resources either by *exchanges* or by *conversions*.

- *Exchange* is a process in which an enterprise receives economic resources from other economic agents, and it gives resources to other economic agents in return.
- *Conversion* is a process in which an enterprise uses or consumes resources in order to produce new or modify existing resources.

The data associated with exchanges and conversions are the primary business data about the enterprise in REA software applications. Account-

ing artifacts such as debit, credit, journals, ledgers, receivables, and account balances are derived from the data describing the exchanges and conversions. For example, the quantity on hand for an inventory item can be calculated from the difference between the purchase and sale events, or between the production and consumption events, for that inventory item.

For comparison, in most current business software applications, whose paradigms are derived from double entry accounting, it is the opposite – they focus on the accounting artifacts, and economic data is derived from them. This, in some sense, puts the consequences before the cause and makes the models more complicated.

The fact that REA operates on primary and raw economic data explains why it offers a wider, more precise, and more up-to-date range of reports than models based on the traditional double entry accounting system that operates on derived accounting data.

REA was originally proposed as a generalized accounting model. It was first published by William E. McCarthy of Michigan State University (McCarthy 1982). McCarthy in his doctoral thesis at the University of Massachusetts analyzed a large number of accounting transactions, and identified their common features and formulated a general model describing and explaining the accounting transactions. Since then, the original REA model has been extended by McCarthy and Guido Geerts to a framework for enterprise information architectures and ontology for business systems (Geerts, McCarthy 2000a, 2002). REA became the foundation for several electronic interchange standards, such as ebXML and Open-edi (an ISO standard), which influenced the extensions of the original REA model into commitments and contracts.

Last but not least, an increasing number of business analysts have found that the models they develop become better when they have REA in mind.

1.2 Joe's Pizzeria

We will create an REA model for Joe's Pizzeria

Joe makes revenue by selling pizza to his customers. Joe's Pizzeria has employees whose task is to sell pizzas, as well as to produce pizzas from raw materials such as dough, tomatoes, cheese, pepperoni and other toppings. There are also other things necessary to produce pizza, such as the oven where the pizza is baked, electricity consumed to heat the oven, various kitchen equipment and many other things. Joe is interested in tracking information about some of them; in REA, the things that Joe is interested in planning, monitoring and controlling are called *economic resources*. Joe has decided that the economic resources that will be included in the business software application are the *Pizza*, *Cash*, *Labor* of the employees, and *Raw Materials and Ingredients* for producing pizza.

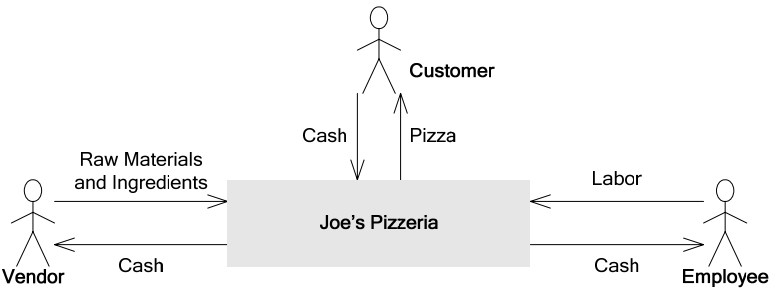

Fig. 2. Trading partners of Joe's Pizzeria

Trading partners of Joe's Pizzeria are customers, vendors and employees. They are capable of controlling economic resources; therefore, in the

REA application model the *Customer*, *Vendor*, *Employee,* and *Joe's Pizzeria* are economic agents, see Fig. 2.

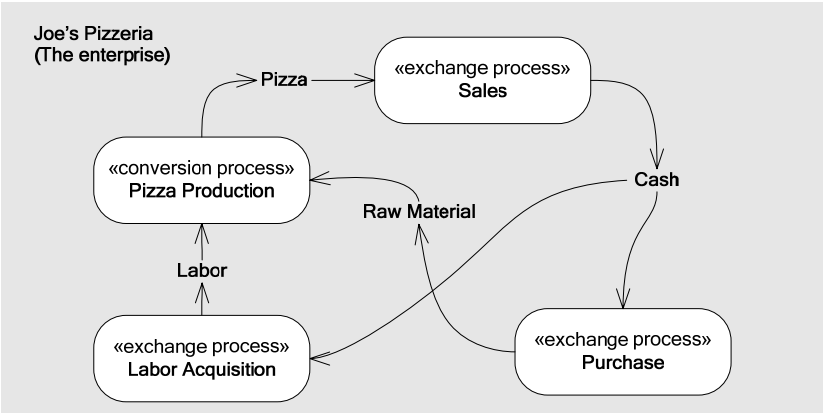

Fig. 3. Trading processes of Joe's Pizzeria

The main trading processes of Joe's Pizzeria, see Fig. 3, are selling pizza to the customers (the *Sales* process), purchasing raw materials from the vendors (the *Purchase* process), and purchasing labor from the employees (the *Labor Acquisition* process). We will construct the REA model for each process.

1.2.1 Sales Process

The process of selling pizza to the customers is essentially an exchange of pizza for cash; Joe's Pizzeria gives *Pizza* to the customer, and receives *Cash* in return. For Joe's Pizzeria, the *Sales* process represents an outflow of *Pizza* and an inflow of *Cash*, see Fig. 4.

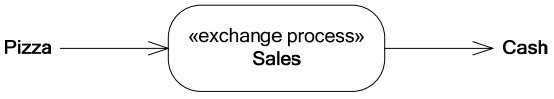

Fig. 4. Selling pizza is an exchange of pizza for cash

The REA model for the process of selling pizza is illustrated in Fig. 5. *Joe's Pizzeria* and the *Customer* are economic agents, and the *Pizza* and *Cash* are economic resources. One economic event is the transfer of own-

ership of the *Pizza* from *Joe's Pizzeria* to the *Customer* (we call this event *Sale*); in this transaction *Joe's Pizzeria* provides *Pizza*, and *Customer* receives it. Another economic event is the transfer of ownership of *Cash* from the *Customer* to *Joe's Pizzeria* (we call it *Cash Receipt*); in this transaction the *Customer* provides *Cash*, and *Joe's Pizzeria* receives it.

For *Joe's Pizzeria*, the *Sale* event (the transfer of ownership of the *Pizza* to the *Customer*) is a decrement event, because it decreases the value of the resources under the control of *Joe's Pizzeria*. The *Cash Receipt* is an increment event, because it increases the value of the resources under the control of *Joe's Pizzeria*. The terms decrement and increment are relative to the model viewpoint; they depend on the economic agent which is in the focus of the model. If we modeled the same process from the perspective of the *Customer*, the transfer of pizza would be an increment (would be called Purchase) and the transfer of cash would be a decrement event (would be called Payment or Cash Disbursement).

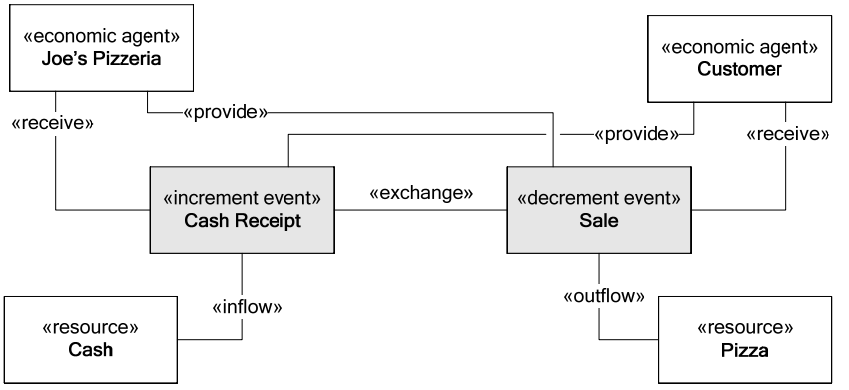

Fig. 5. The REA model for Joe's Pizzeria sales process

The REA model of the sales process in Fig. 5 focuses on the core economic phenomena, and therefore it covers many special cases. For example, most customers pay when they purchase pizza, but some customers may receive an invoice, and pay for all their purchases in a certain period at once. If the case of Internet sales, customers must provide their credit card information before the pizza is delivered, and Joe's Pizzeria receives money from the customer's bank later. When the sale occurs in the restaurant, the customers pay after they get pizza, either using cash or a credit card.

All these cases are covered by the model in Fig. 5; this is very useful if we would like to create a robust skeleton of a software application.

Customers may order pizza over the Internet. In this case, a software business application creates an electronic *Sales Order*, which specifies a commitment of *Joe's Pizzeria* to sell a specified *Pizza* to the *Customer*, and a commitment of a *Customer* to pay for the *Pizza* a specified amount of *Cash*.

The *Sales Order, see* Fig. 6*,* is an example of a contract between the economic agents *Joe's Pizzeria* and the *Customer*. The *Sales Line* and the *Payment Line* are not economic events; they are commitments to perform the economic events in well-defined future. The *Sales Line* is a commitment to perform the event *Sale*, and the *Payment Line* is a commitment to perform the event *Cash Receipt* in the future.

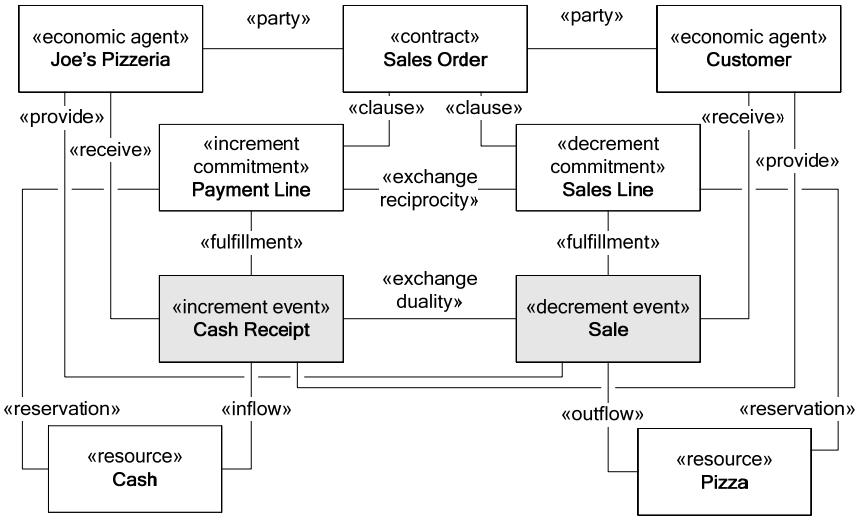

Fig. 6. The REA model for the sales process with sales order

The *Sales Order* often contains terms specifying what should happen if the commitments are not fulfilled, such as when the payment arrives late, or the customer is not satisfied with the pizza. The fact that a contract can be represented as a computer model is important for automatic tracking of the state of the contract at runtime, and also for computer-assisted evaluation of complicated financial contracts.

1.2.2 Purchase Process

When Joe's Pizzeria purchases tomatoes, cheese, pepperoni, flour and other raw materials, it essentially exchanges the raw material for cash. *Vendor* gives *Raw Material* to *Joe's Pizzeria*, which gives it *Cash* in return. For Joe's Pizzeria, the *Purchase* process represents an outflow of *Cash* and an inflow of *Raw Material*, see Fig. 7.

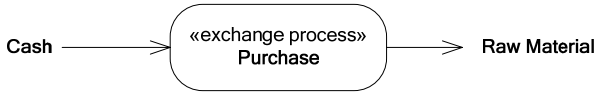

Fig. 7. Purchasing raw material is an exchange of raw material for cash

The REA model for the process of purchasing raw material is illustrated in Fig. 8.

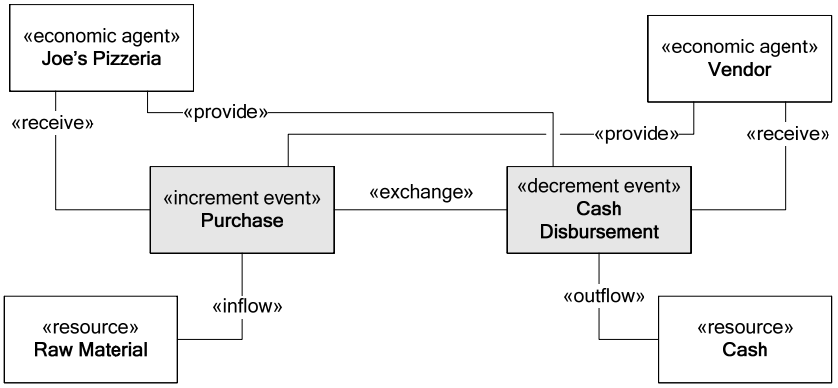

Fig. 8. The REA model for the purchase process

The *Vendor* and *Joe's Pizzeria* are economic agents, the *Raw Material* and *Cash* are economic resources. The transfer of ownership of the *Raw Material* from the *Vendor* to *Joe's Pizzeria* is an increment economic event (we call it *Purchase*), and the transfer of ownership of *Cash* from *Joe's Pizzeria* to the *Vendor* (we call it *Cash Disbursement*) is a decrement economic event; the increment and decrement are from *Joe's Pizzeria* perspective.

Similarly as for the REA model for sales, the REA model for purchases covers many special cases. Some raw materials can be paid by check or bank transfer; some can be made in different currencies. There can be sev-

eral purchases paid using a single payment, and a single purchase can be paid in several installments. The model tracks the information about which purchases correspond to which cash disbursements, but abstracts from technical details and does not specify the order of these transactions. Again, this is useful if the skeleton of a software application is based on this model, because it does not have to be changed if some technical aspects of the purchase process change.

1.2.3 Labor Acquisition Process

Joe's Pizzeria employees provide their work (they produce and sell pizzas during specified periods of time) and receive their salary in return. Labor acquisition is essentially an exchange of *Labor* (the worked hours) for *Cash*. *Employee* sells his labor to *Joe's Pizzeria*, which gives him *Cash* in return. For Joe's Pizzeria, the *Labor Acquisition* process represents an outflow of *Cash* and an inflow of *Labor*, see Fig. 9.

Fig. 9. Labor acquisition is an exchange of worked hours for cash

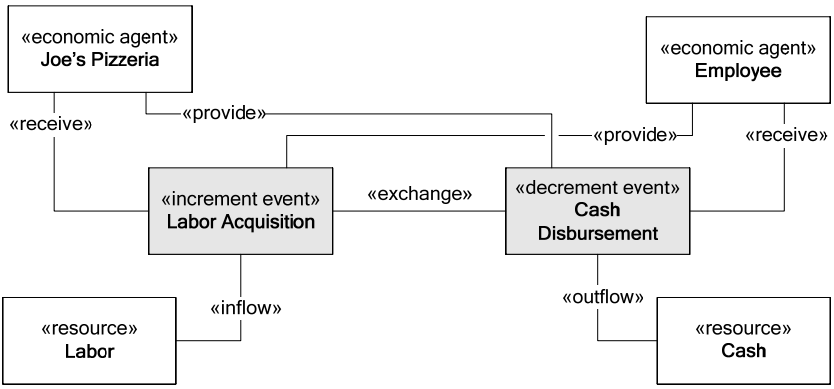

Fig. 10. The REA model for the labor acquisition process

The REA model for the labor acquisition process is illustrated in Fig. 10. The *Employee* and *Joe's Pizzeria* are economic agents; the *Employee* provides *Labor* and receives *Cash*, and *Joe's Pizzeria* provides

Cash and receives *Labor*. *Labor* (the worked hours) and *Cash* are economic resources. The *Labor Acquisition* is an economic event that occurs over periods of time (during the employee's working hours), while *Cash Disbursement* is an instantaneous event that occurs once a week or month when the *Employee* receives his paycheck.

The REA model in Fig. 10 can be applied to many forms of acquiring labor; it can be applied for full employment, temporary work, consulting, as well as for work acquired according to various other forms of contracts.

1.2.4 Summary

The REA model focuses on the core economic phenomena and abstracts from technical and implementation details. This has several advantages.

Firstly, the REA model abstracts from the technical aspects of the transfer of the resources. Cash can be transferred as bills and coins, as a check or as a credit card transaction. Customers can pick pizza themselves, or pizza can be delivered to their address. For all these cases we can apply the same REA model, which does not have to be modified even if the technical infrastructure supporting the business changes.

Secondly, the REA model abstracts from the order in which the economic events occur. Usually, pizza is paid at about the same time as it is given to the customer, but sometimes it is paid for beforehand, and sometimes it can be paid by credit card and there is a significant delay between the sale of pizza and the transfer of cash. If the business process was specified as a scenario consisting of a sequence of events, the business application would support only the scenarios identified at design time. The REA model allows the business application to flexibly record everything that actually happened. The actual order of events emerges at runtime, rather than being specified at design time.

Thirdly, for each REA model apply certain rules: each increment must be related to a decrement, each economic event must have a provider and recipient agent, and each resource must be related to both increment and decrement. Therefore, application designers can ask relevant questions leading to the discovery of missing information in the user requirements, and can construct the model even if the initial specification is incomplete.

1.2.5 The Illustrated Models Are Examples of a Pattern

The three illustrated models for the business processes *Sales*, *Purchase* and *Labor Acquisition* have many common features. They all model the trans-

actions between Joe's Pizzeria and its trading partners as exchanges of economic resources.

These models can be generalized into a model at a higher level of abstraction, illustrated in the next chapter. The models for sale of pizza, purchase of raw materials and labor acquisition are examples of the *REA EXCHANGE PROCESS PATTERN*.

1.3 REA Exchange Process Pattern

Trade is the voluntary exchange of goods, services, or money

Context

You are an application designer developing a business application. You are trying to create an object model of a business application and struggling to find the right structure for the model and the right relationships between entities in the model. You know the user requirements; they can be in a written document or non-written information obtained by an ongoing dialog with the users; but you know the requirements are incomplete. You want to know the right questions to ask to better understand the application domain. You also want the model to be consistent and robust enough for future changes in user requirements.

Problem

How does one create a robust skeleton of an object-oriented model for interactions between the enterprise and it trading partners? User requirements are not a sufficient source of information, because they are known to be incomplete, often contradictory, and to change over time, and it is often impossible to find what requirements are missing. Shortly, you would like to create a business application that will satisfy even some of user requirements that have not been communicated to you.

Forces[1]

The REA exchange process pattern resolves the following forces:

- The modeled software application should provide information about how the interactions between the enterprise and its trading partners change the value of the economic resources of the enterprise. The application should keep track of the increases and decreases of the value of the resources that are under the control of the enterprise, and should record which resources were exchanged for which others.
- Application designers want to concentrate on the fundamentals of the users' business, and separate those requirements which are likely to change. The fundamentals are often so obvious to the users of business applications that they do not communicate them, and they remain hidden until late stages of software development.
- The model should be consistent with the business domain rules. Application designers want to ensure that the model is consistent, complete, and correct, with respect to the domain rules.
- Application designers want to include business semantics into the entities in the application model. Semantics based only on the names of the entities is not good enough because it relies on common knowledge, and common knowledge is not available to software applications.

Solution

Model the interactions between the enterprise and its trading partners as *exchanges* of economic resources.

Each exchange consists of at least one *increment economic event* that increases the value of a resource of the enterprise by transferring rights to the resource to or from other economic agents. Every increment event is related to at least one *decrement economic event* that decreases the value of a resource of the enterprise by transferring rights to the resource to or from other economic agents. We call the relationship between the increment and decrement economic events *exchange duality*, or in short, *exchange.* The *exchange duality* is a many-to-many relationship, indicating that in the application model there must be at least one decrement for each increment, and vice versa. Therefore, the exchange duality in the applica-

[1] In the pattern literature the term *forces* is used for the constraints that restrict the solution of the problem, requirements, and properties that the solution should have. Appendix C describes the pattern form in detail.

tion model can be an n-ary relationship, that relates several increment and decrement entities.

In order for an exchange process to add value, the overall increase in value of the resources related to the increment events should be greater than the overall decrease in value of the resources related to the decrement events.

Each *economic event* is related to an *economic resource*, see Fig. 11. The relationship between an increment and a resource is called *inflow*, the relationship between a decrement and a resource *outflow*. In the application model there must be exactly one *economic resource* for each *economic event*, and at least one *increment* and one *decrement* for each *economic resource*.

Each economic event is related to two *economic agents*. The *economic event* in the exchange process transfers rights to the *economic resource* from one agent to another. When the event occurs, the *provider agent* loses rights to the resource, and the *recipient agent* receives the rights. In the application model for each economic event there must be at least one *provider* and at least one *recipient agent*. For each *agent*, there can be zero or more *economic events*.

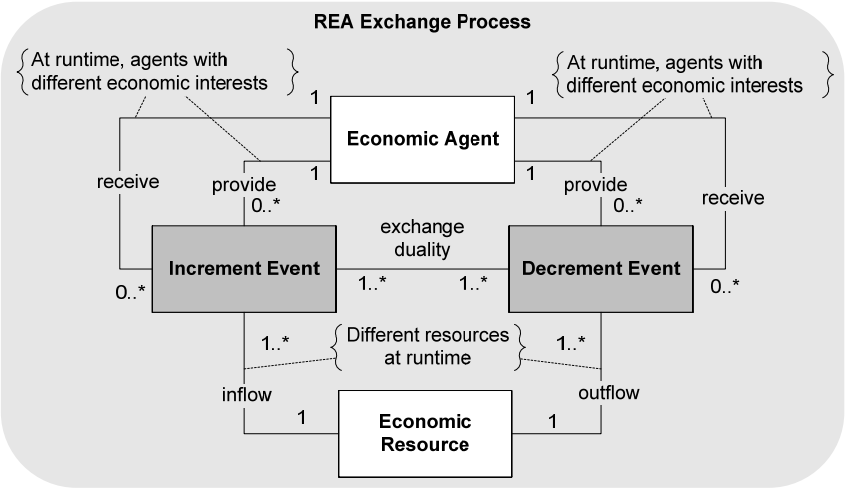

Fig. 11. The REA exchange process

Note that the model in Fig. 11 determines the rules for constructing the application model. The application model determines the structure of the runtime instances.

The following domain rules apply for any application model describing the REA exchange process.

Every increment economic event must be related by exchange duality to a decrement economic event, and vice versa.

Every increment economic event must be related by inflow relationship to an economic resource.

Every decrement economic event must be related by outflow relationship to an economic resource.

Every economic event must be related by a provide relationship to an economic agent, and by a receive relationships an economic agent. At runtime, these two agents must represent entities with different economic interests.

Resulting Context

The domain rules in this pattern allow application designers to derive new facts from the facts provided by the users, and to formulate questions leading to the discovery of new facts. Therefore, a business application can meet most or all user needs, even if the user requirements and the designers' knowledge of the user needs are incomplete.

Note that at runtime, for some period of time, there might exist an instance of an increment event that is not paired in exchange relationship with a decrement event. This temporary imbalance results in a claim between economic agents. The claim can be materialized, for example as an invoice. The concept of a claim is described in the chapter REA Exchange Process in Detail, and the materialized claim is described as a pattern in Part II of the book.

http://www.springer.com/978-3-540-30154-7

Model-Driven Design Using Business Patterns
Hruby, P.
2006, XVI, 368 p., Hardcover
ISBN: 978-3-540-30154-7