Universal Enterprise Model: Business Pattern Language

Pavel Hruby

Microsoft Business Solutions
Frydenlunds Allé 6
DK-2950 Vedbaek, Denmark
E-mail: phruby@acm.org

Abstract

Have you ever tried to describe an object model of a business system and struggled to find the right relationships between business entities, such as customers, business partners, products, sales and purchase orders, invoices and credit memos? Have you ever wanted to know a simple rule for modeling the business system in a consistent manner? The Universal Enterprise Model is a pattern language for building extensible models of business systems. The fundamental structure of the business system is derived from the resources-events-agents (REA) pattern, which is extended by a number of behavioral patterns, such as roles, due dates, addresses, classifications and accounts.

Context

Relationships between business partners are at the core of business. Various business software solutions implement functionality that focuses on various aspects of relationships between business partners, such as customers and vendors, employers and employees, service providers and service receivers. Almost all of these relationships are in some way intended to, or directly related to exchanges of economic resources, such as the purchase and sale of products and services, corresponding payments. Some relationships specify commitments and constraints for the exchanges.

Some examples of such relationships are as follows:

- *Purchase order*: a commitment for the vendor to deliver goods and obligation for the customer to pay for it. In addition, a purchase order specifies payment and delivery terms, such as the delivery date and what happens if the delivery date is not met.

- *Invoice*: a declaration of the claim that the buyer owes a specific amount of money to the seller. In addition, an invoice typically specifies other properties such as payment terms.

- *Employment contract*: specifies details about relationship between employee and employer. In addition, the employment contract specifies other conditions of the employment, such as position and compensation.

- *Shipment*: a movement of materials between business partners, warehouse sites, or between a warehouse site and a business partner.

- *Payment*: a transfer of money from one business partner to another.

I call the abovementioned artifacts *business relationships*. The term *business relationship*, as used in this paper, covers relationships between parties in various scopes and at various levels of abstraction. An example of a general scope relationship is a contract for providing a maintenance service in a given period of time. This contract can result in more specific relationships, such as service orders, and they result in more specific relationships, such as material movements, payments and other business transactions.

I call the business partners participating in a business relationship *parties*. In keeping with the authors of other publications [1], [3], [4], [6], [7], I use the term *party* to mean a business entity that can participate in a business relationship with another party, such as a person, company, legal entity, team, or organizational unit.

The *resource* is a subject of trade. I use the term resource to mean a concrete physical product, asset, inventory, or service that has identity. For example, a product that has a serial number, or a service that has a start time and end time.

Problem

Have you ever tried to describe an object model of a business system and struggled to find the right structure of the model and the right relationships between business entities? Have you ever wanted to know a simple rule for modeling the business or economic system in a consistent manner?

Forces

1. Parties have relationships between each other. We can model these relationships as associations between the parties. However, these relationships often have specific attributes, describing details of this relationship rather than details of one of the parties. Examples of such attributes are delivery due, validity period, payment terms and employment position. Because of these attributes, the relations between parties cannot be modeled as pure associations.

2. Relationships between parties involved in business are manifested in various documents, such as purchase and sales orders, quotes, invoices, payments, and delivery receipts. These documents vary in complexity and it is not possible to determine a complete set of business documents that fit all situations in all businesses. However, you want to capture all these relationships in a uniform way in the object model.

3. If a relationship exists between parties, this relationship may, under certain conditions, create or cause the creation of other relationships between the same parties. For example, a purchase order may result in a delivery of goods. You want to capture this fact in the object model. However, traditional object-oriented modeling techniques do not give any hints for how to describe the fact that one relationship between objects can imply another.

Solution

Encapsulate the relationship between parties in an entity called *business relationship*. The business relationship entity is related to at least two *party* entities: the supplier and the consumer. Examples of parties are customer, vendor, or employee. Each business relationship is related to one or more

resources. Examples of resources are an asset, a physical product, service, work, or another subject of trade.
The conceptual structure of the business relationship pattern is illustrated in Fig.1.

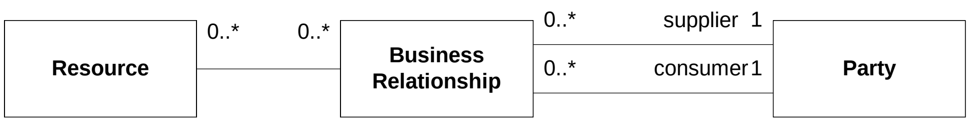


Fig.1. Conceptual structure of the business relationship pattern

Known Uses

Purchase order is a business relationship that specifies a commitment for the vendor to deliver goods and an obligation for the customer to pay for it. Purchase order is related via the reconciliation to the business relationships shipment and payment.

Invoice is a business relationship that declares the claim that the buyer owes a specific amount of money to the seller.

Customer phone call requesting a sales quotation is a business relationship, related to certain resources. It is related via the reconciliation relationship to the quotation.

Resulting Context

If you are starting to look at your enterprise through the perspective of business relationships, then you will need to decide what relationships between parties exist, and model these relationships as entities.

The business relationship pattern allows you to define the fundamental infrastructure of the enterprise model, which can be extended by other patterns. See the section Related Patterns for details. However, besides this structure, the business relationships pattern does not specify any constraints and modeling rules. The system compliant with economic rules is described in the REA, COMMITMENT and CLAIM patterns.

Related Patterns

The pattern BUSINESS TRANSACTIONS specifies how to record history of business relationships. The patterns REA, COMMITMENT and CLAIM extend the BUSINESS RELATIONSHIPS pattern with general economic rules valid for every economic system. These patterns determine the structure and skeleton of the enterprise model.

The rest of the patterns extend the structural patterns with specific functionality and features. Currently, in this group are DUE DATES, ROLES, ACCOUNTS, CLASSIFICATIONS, and ADDRESSES. This is not a final list, and the pattern language can (and will) be extended by adding more patterns to this group. The pattern map is illustrated in Fig. 2.

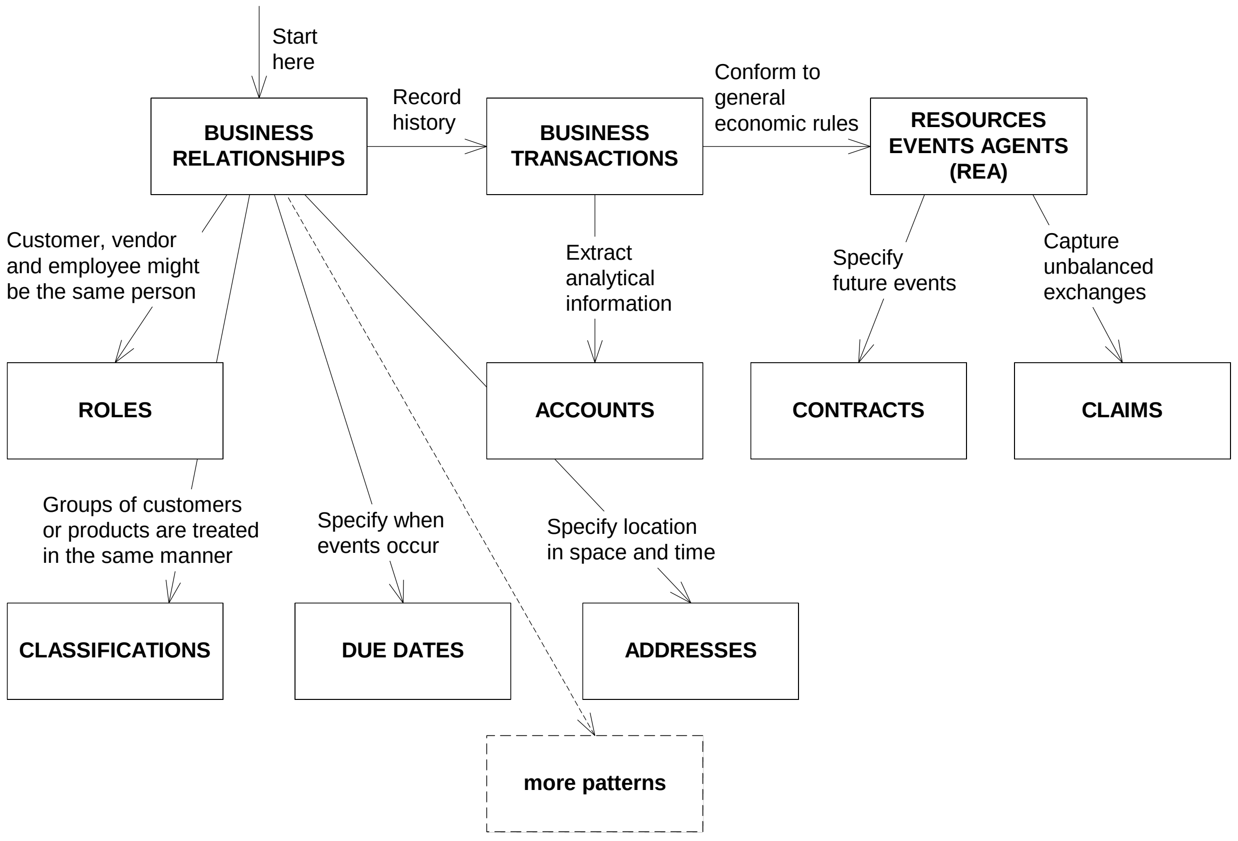

Start here
BUSINESS RELATIONSHIPS
Record history
BUSINESS TRANSACTIONS
Conform to general economic rules
RESOURCES EVENTS AGENTS (REA)
Customer, vendor and employee might be the same person
Extract analytical information
Specify future events
Capture unbalanced exchanges
ROLES
ACCOUNTS
CONTRACTS
CLAIMS
Groups of customers or products are treated in the same manner
Specify when events occur
Specify location in space and time
CLASSIFICATIONS
DUE DATES
ADDRESSES
more patterns

Fig. 2. Pattern Map

Credits and Sources

1. Christian Vibe Scheller of Denmark, personal communication, 1998 – 2000, Lars Hammer, Jesper Kiehn, Henning Kjersgaard Nielsen, Erik B. Jakobsen of Microsoft Business Solutions, personal discussion, 2000 – 2002.

2. David Hay in [6] describes the business relationships Purchase and Sale Orders, Lease, Permit and Employment Contracts. David Hay uses the terms *contract* for business relationships, *line item* for business relationship line and *asset* for the resource. The business relationship pattern, as described in this paper, is more general than David Hay's contact. It also covers material movements, invoices, etc.

3. Peter Coad in [1] uses this pattern to model various business relationships, including Request for Quote, Purchase and Sales orders, Delivery, Invoice, Service, Production Requests, Manufacturing process. The complete list is available at http://www.oi.com/services/publications/jm_book.zip, Peter Coad uses the terms *Moment-Interval* for Business relationship, *Description* for resource and *Party Role* for party. The universal enterprise model pattern language uses this pattern as a skeleton to bind together additional business patterns.

4. Customer Contact Pattern by Dirk Riehle is a specific instance of this pattern, in which one of the parties is the customer. For details, see http://www.riehle.org.

Business Transactions

Context

Registering and analyzing history of the business relationships between business partners is an important part of the functionality of most business software solutions. The history of business relationships is typically related to realized or intended exchanges of economic resources, such as purchase and sale of products and services, invoices and corresponding payments.

Problem

How do we keep track of the history of business relationships?

Forces

1. You want a system that records all relevant information about relationships between business partners.

2. You want to model the fact that an event that affects the business relationship might imply other events to occur. For example, shipment of goods implies payment to occur.

3. If user of the system made an error when making a record, there are often legal requirements for correction of the error. The original (erroneous) information should often not just be deleted or overwritten, but instead a new record that eliminates the effect of the error should be made.

Solution

Encapsulate the event that changed the relationship between parties in an entity called *business transaction*. The business transaction is related to two *party* entities: the supplier and the consumer. Each business transaction is related to one or more *resources*. Business transaction might have additional relationships to other entities, such as to the category of party or category of resource (see the CLASSIFICATIONS pattern for details).

The operation *commit()* persists the business transaction and makes is immutable, that is, no changes in the properties and relationships are allowed after this operation has successfully finished. If the business transaction contained erroneous information, the only way to undo the effect of this transaction is to create and commit another transaction that eliminates the effect of the error. See reference [5] for discussion on adjusting transactions.

The attribute *when occurred* denotes a time interval or point in time when the transaction occurred, and *when committed* denotes the time when the transaction was registered.

The *reconciliation* relationship represents a cause-and-effect association between the business transactions. For example, a quotation might be followed by an order, shipment by payment, or payment by shipment. Please note that *initiator* and *terminator* of the reconciliation do not imply any time order. That is, the terminator business relationship can take effect before the initiator business relationship and vice versa.

The structure of the business transaction pattern is illustrated in Fig.3.

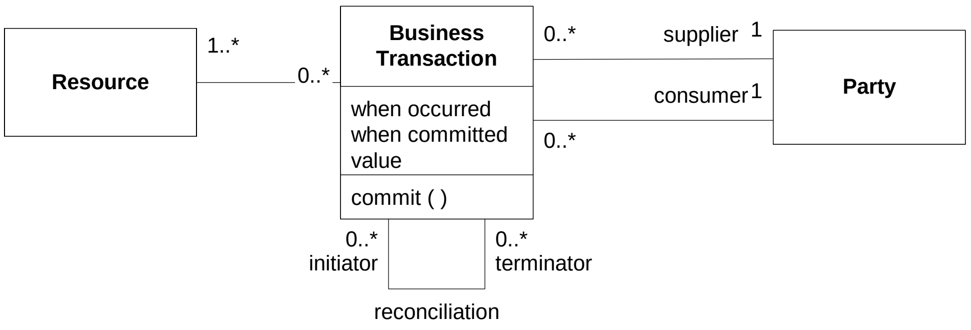


Fig.3. Structure of the business transaction pattern

An example of the business transaction pattern is illustrated in Fig. 4. The figure illustrates two business transactions: invoice and payment. The parties are Customer C and Vendor V. The invoice line specifies four pieces of Product A. The reconciliation relationship links together the payment and the invoice.

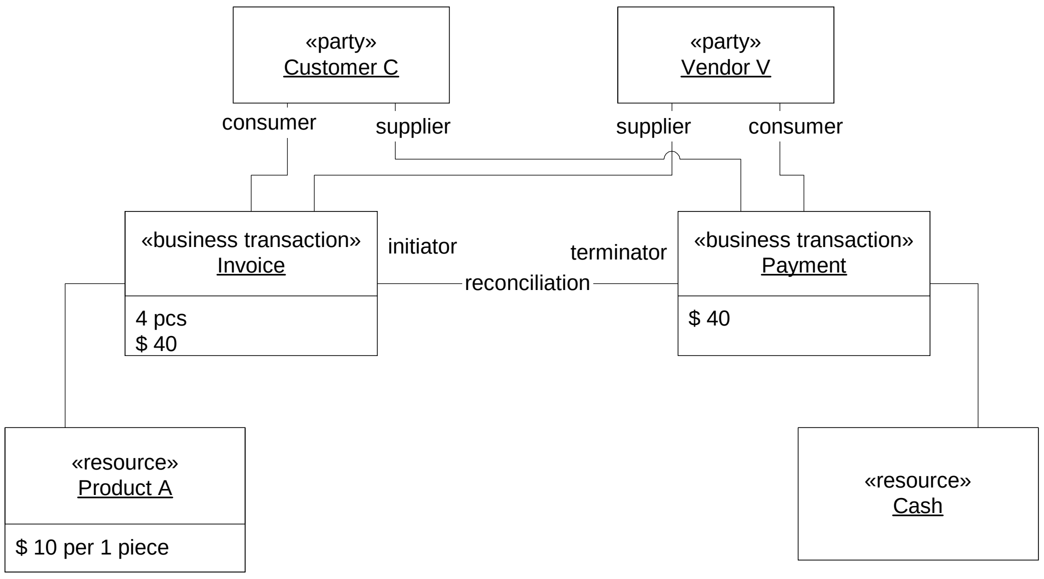


Fig. 4. Example of business transactions

Known uses

Material movement. The warehouse locations are the parties, and the responsibility for the material is the resource.

Sales order line. Sales order line is related (via sales order header) to the customer and seller. Sales order line specifies a product or service, which are the economic resources for both parties.

Rental. The premises is the resource; yielding the right to occupy to the tenant is an outflow of resources. Payment of rent is an inflow of resources. Rental is an example of business transaction that occurs over an interval of time.

Deposits, withdrawals of cash, and interest payments are the transactions in a banking system. Bank and customer are the parties, and responsibility for cash, and interest payments are the resources.

Resulting Context

The BUSINESS TRANSACTIONS is a special case of BUSINESS RELATIONSHIPS. It covers those business relationships, for which all changes are recorded, especially the exchanges of economic resources between business partners, and commitments for such exchanges.

Some business transactions last over a period of time, and recording these changes is not always natural. Some changes, like usage of the equipment or shrinkage of the inventory, happen continuously, and not at discrete points in time. Nevertheless, the accounting practice has standard way of solving these issues.

Please note that a business transaction does not need to have a navigable relationship to financial accounts. See the ACCOUNTS pattern, and the reference [6] for more information. However, many traditional accounting systems specify an account already when the transaction is committed. See the reference [5] for more information on this approach. Although this is commonly used in the accounting practice, this solution has its limits for very large business information systems. Please see the reference [8] for discussion on scalability.

Related Patterns

The patterns REA, COMMITMENTS and CLAIMS extend the BUSINESS TRANSACTIONS pattern with general economic rules valid for every economic system. The ACCOUNTS pattern shows how to extract analytical data from the recorded transactions.

Credits and Sources

1. Business transactions are part of the Transactions and Accounts pattern language, available at http://c2.com/cgi-bin/wiki?TransactionsAndAccounts

2. Martin Fowler' paper Accounting Patterns has a concept of event that is close to business transaction. The paper is available at http://martinfowler.com/apsupp/accounting.pdf.

The REA (Resources Events Agents) Pattern

Context

Business is based on exchanges of economic resources. For example, a customer at a shop buys a product. The product is the resource and the sale is the outflow of resources. The payment for the product is an inflow or resources and the cash is the resource. If the deal is fair, both partners agree that the value of the exchanged resources is the same. That is, the agreed value of the product is the same as the amount of cash received in return.

For business or legal reasons, it is important to keep track of what resources have been exchanged between business partners and when the exchange of resources occurred.

Some exchanges occur over an interval of time, for example, providing or receiving services, or labor work. Even the sale in a shop might have a non-zero duration, if we want to keep track of various stages of the selling process. It is not that important to distinguish whether exchange occurred instantaneously or during an interval of time. It depends on the point of view. From this pattern perspective it is important that the exchange occurred and that the resource changed ownership.

Problem

How do we model exchanges of economic resources?

Forces

1. You want to model the rules that apply to *all* economic systems. However, most analysis patterns and data models are often domain dependent, because they reflect experience of specific software consultants.

2. You want to model things that are shareable and reusable across various application domains, and where details of specific applications are ignored. You could build your object model from user requirements, but it would be very difficult to find the right abstractions valid for *all* economic systems.

3. You want to build an extendable system, which can be extended by new concepts without changes to the foundations of the system. This is difficult, because you must sort out or generalize the domain specific knowledge. Otherwise, your system would result in changes, instead of extensions.

4. You want well-defined modeling rules that enable you to ensure that your model is consistent. For example, how do you keep track of a party who gives resources away, and eventually receives other resources in return? Other examples of consistency questions include: if company sells a product, how does the company obtain it? Or what does a company get in return for providing a customer with certain benefits?

Solution

Consider the economic activities as a sequence of exchanges of resources – the process of giving up some resources to obtain others. The following entities model the exchanges of resources.

Economic event represents the interval in time, or a moment in time when economic exchange occurs. Further more, economic event keeps track of the *value* of exchanged resources. The *time interval* specifies a moment or interval in time when the exchange occurred. Some exchanges occur instantaneously, such as sales of goods; some occur over interval of time, such as rentals or services. Note that the change of ownership (the economic event) sometimes occurs at a different time to that of the physical movement of goods.

The economic event represents either *inflow* or *outflow*. Inflow is an event when a party receives ownership of resources, for example, incoming payment. Outflow is an event when a party yields ownership of resources, for example, a shipment of goods.

Resource represents the subject of trade. Resource is something of value that is under the control of the party.

Party represents an economic unit, or legal entity capable of exchanging economic resources with other parties.

Duality is a special kind of reconciliation. It is a relationship between inflow economic events and outflow economic events. When the deal is closed and fulfilled, the values of inflow and outflow economic events are the same. It is *duality's* responsibility to keep track of the balance between the outflow and inflow. Duality is a many-to-many relationship. For example, several sales can be paid by one check, and one sale can be paid by several installments.

The model is illustrated in Fig. 5.

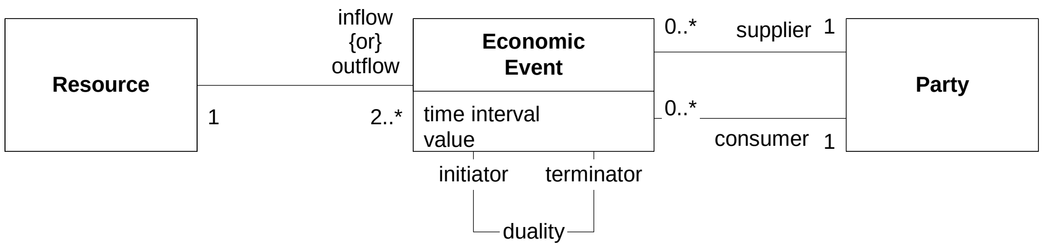


Fig. 5. The economic event pattern as a special case of business transaction

The following three rules apply for the model. They can be used to ensure consistency of a specific instantiation of this pattern.

1. At least one inflow and one outflow economic event exist for each resource. Conversely, each inflow and outflow economic event must be related to a resource.
For example, goods related to the sales event must also be related to the purchase or production event (or some other event specifying how those goods are going to be obtained).

2. Each outflow economic event must have a duality relationship to an inflow economic event, and vice versa. For example, shipment to a customer (outflow) must be related to a customer payment (inflow).

3. Each economic event must have a relationship to two parties participating in the exchange. For example, each economic event must be related to the customer and vendor, employer and employee, and so on.

An example of a simple system is illustrated in Fig. 6. In this example, a wholesaler buys products from vendors and sells them onto customers. The *Customer*, *Vendor* and *Wholesaler* are the parties; the *Product* and *Cash* are the resources; *Goods Receipt*, *Payment*, *Shipment* and *Cash Receipt* are the economic events.

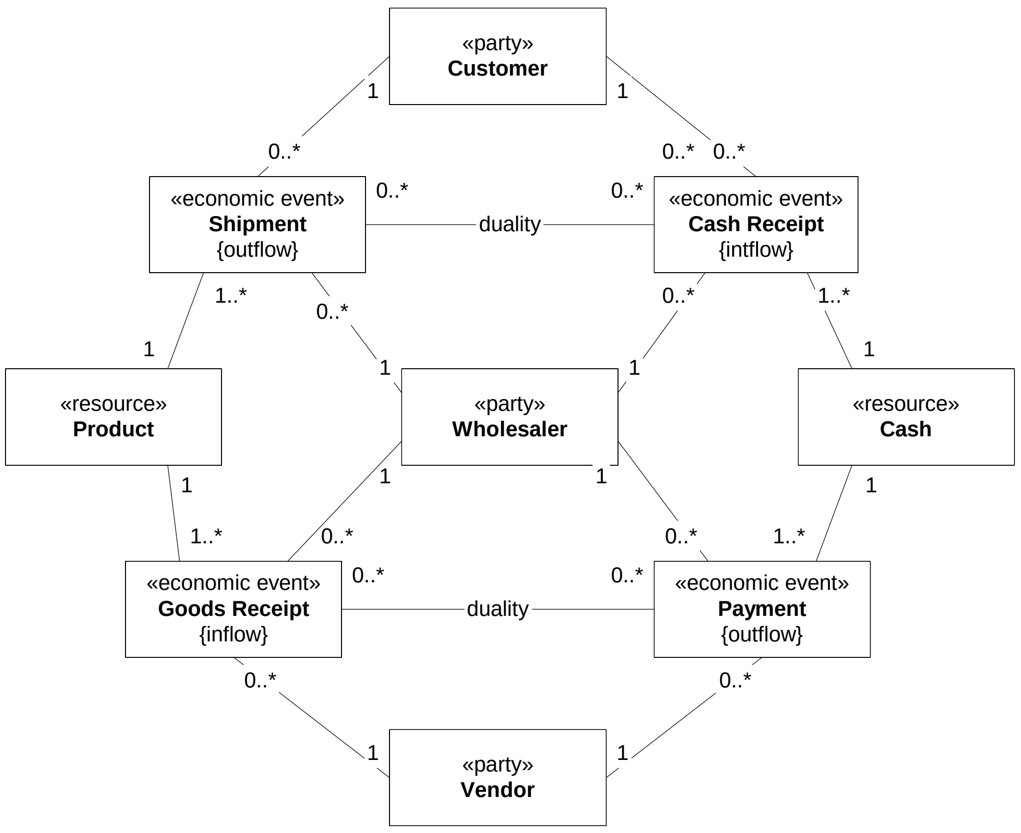


Fig. 6. Economic events for a wholesale system

Fig. 7 illustrates and example, in which the upper part of the example from Fig. 6 is enhanced by two economic events, Return of Goods and Cash Refund. The duality here is a ternary relationship between these four economic events. The duality ensures that when the deal is closed, the sum of values of the initiator events is equal to the sum of the terminator events.

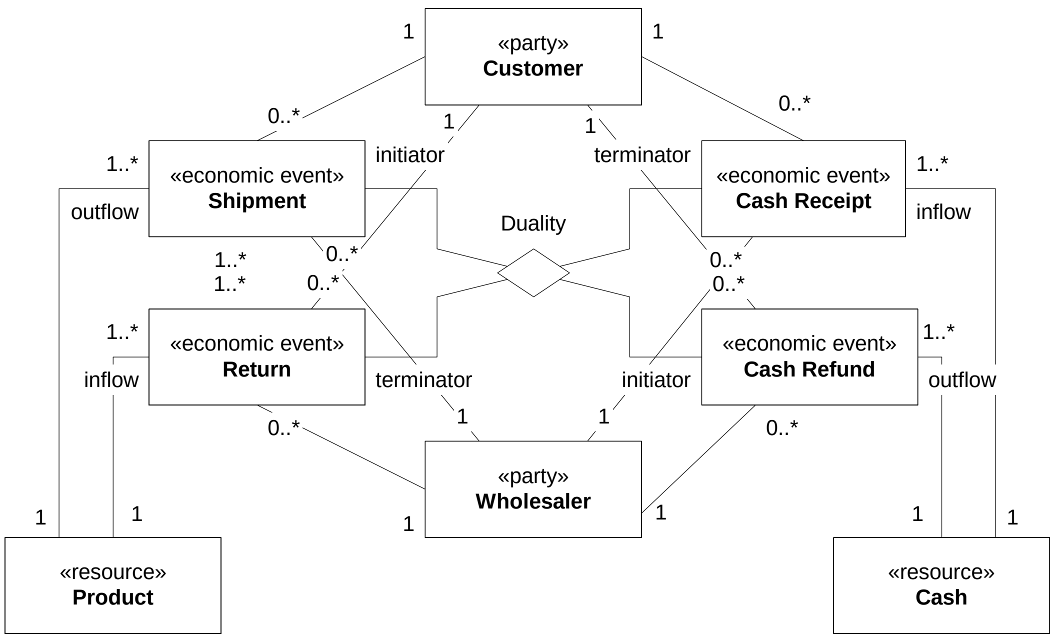


Fig. 6. Return of goods

Known uses

Cash sale. Product and cash are the resources; delivery and payment are the economic events.

Rental. The premises is the resource; yielding the right to occupy to the tenant is an outflow of resources. Payment of rent is an inflow of resources. Rental is an example of economic event that occurs over an interval of time.

Employment. From employer's perspective, an employee's time is the inflow of resources, salary is the outflow. Employment is another example when an economic event occurs over an interval of time.

Resulting Context

The economic event is a special case of business transaction, see BUSINESS TRANSACTIONS pattern. It covers those business transactions that change ownership of resources. Moreover, this pattern defines rules valid for *all* economic systems. This pattern forces developer to make a *complete* model of the economic exchanges, and think about non-obvious questions like "what do I get in return for paying my taxes?" While this is often useful, sometimes the overall burden from increased complexity overweighs the benefits of getting a complete and economically correct model.

As the economic event may occur over interval of time, sometimes it is useful to model interactions between business partners within the scope of one economic event. The LIFECYCLES[1] pattern addresses this issue.

Related Patterns

The COMMITMENTS pattern is a special kind of BUSINESS TRANSACTIONS that specifies the economic events scheduled to occur in the future.
The CLAIMS pattern is a special kind of BUSINESS TRANSACTIONS that deals with unbalanced exchanges of resources.

The ADDRESSES pattern specifies where the exchange occurred. The ACCOUNTS pattern specifies how to create a report that aggregates values across sets of economic events. The CLASSIFICATIONS pattern specifies how to handle various categories of economic events.
The DUE DATES pattern specifies the time interval in the case the economic event occurs over interval of time.

Credits and Sources

1. Bob Haugen, e-mail discussion
2. Guido Geerts, personal communication
3. William E. McCarthy and Jesper Kiehn, teleconference
4. Henning Kjersaard Nielsen and Erik B. Jakobsen, personal discussion
5. McCarthy, W.E. 1982. "The REA Accounting Model: A Generalized Framework for Accounting Systems in A Shared Data Environment." The Accounting Review (July), pp. 554-578.

[1] The LIFECYCLES pattern is not part of this pattern language, yet.

Contracts

Context

Some relationships between business partners specify promises of future exchanges of resources. For example, a purchase order is a promise to receive goods from a vendor, and pay for the goods. An employment contract is a promise that an employee will give his time to the employer and a promise by the employer to provide compensation in return.

Problem

How to model promises of future exchanges of economic resources?

Forces

1. Most economic events do not occur unexpectedly. They have been agreed between business partners beforehand. You would like to have a mechanism specifying details about the commitments of economic events.

2. If a party commits itself to give resources away (to pay for a product or accept a purchase order), the party would like to be informed about whether it will actually have the resources available at the time specified by the commitment.

Solution

Consider the economic contracts between business partners as a collection of commitments to trigger economic events in a well-defined future. The following entities model the contracts and commitments.

Commitment entity specifies an economic event or events to occur in the future. *Commitment* has a relation to *Resource*, and either reserves the resource for future stock outflow, or expects a resource for future stock inflow.

Fullfillment (a special kind of reconciliation) is a one-to-many relationship between the commitment and the economic event. One commitment can be fulfilled by several economic events.

Economic contract is a set of commitments, which specify the future inflow or outflow of resources. At least one commitment in this set should be related to the inflow economic event, and at least one to the outflow economic event.

Reciprocity (a special kind of reconciliation) is a many-to-many relationship between commitments. Reciprocity guarantees that the total value of expectation claims is equal to the total value of the reservation claims.

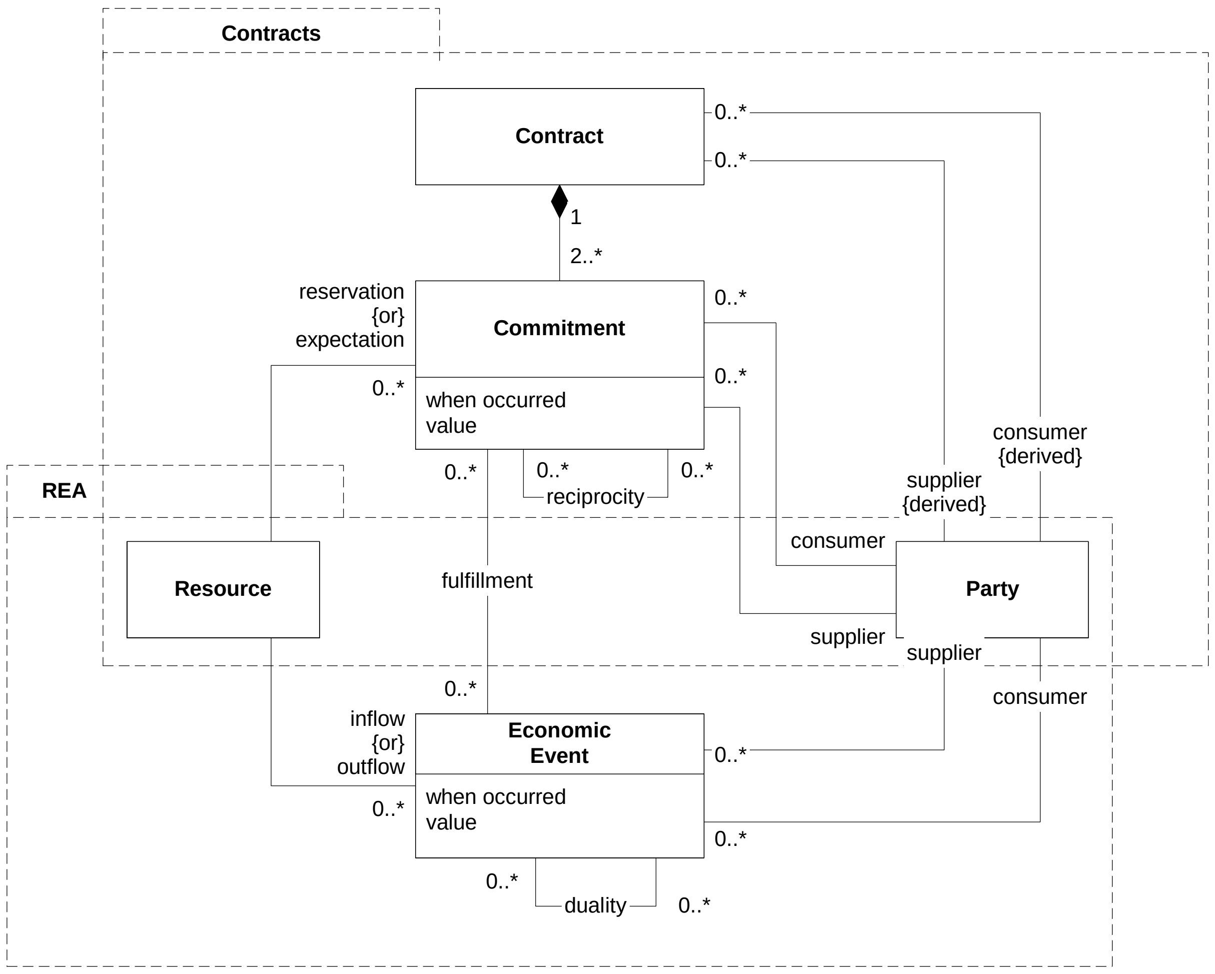


Fig. 7. The commitments pattern as a special case of business transaction

The following rules apply for the model. They can be used to ensure consistency of a specific instantiation of this pattern.

1. Each commitment must be related to a resource. For example, a sales order line must specify the goods to be sold.

2. Each commitment must have two relationships to parties that agree on the future exchange of resources. The customer and vendor, the employer and employee are the examples of parties related in contractual relationship.

3. Each commitment must have the fulfillment relation to at least one economic event.
For example, the purchase order line specifying the goods must be related to the shipment of these goods.

Fig. 8 illustrates an example of a sales order. Sales order is a contract composed of two commitments. The reservation commitment is to sell 4 pieces of product A, of a value of $ 40. The expectation commitment is to receive $ 40 compensation in cash.

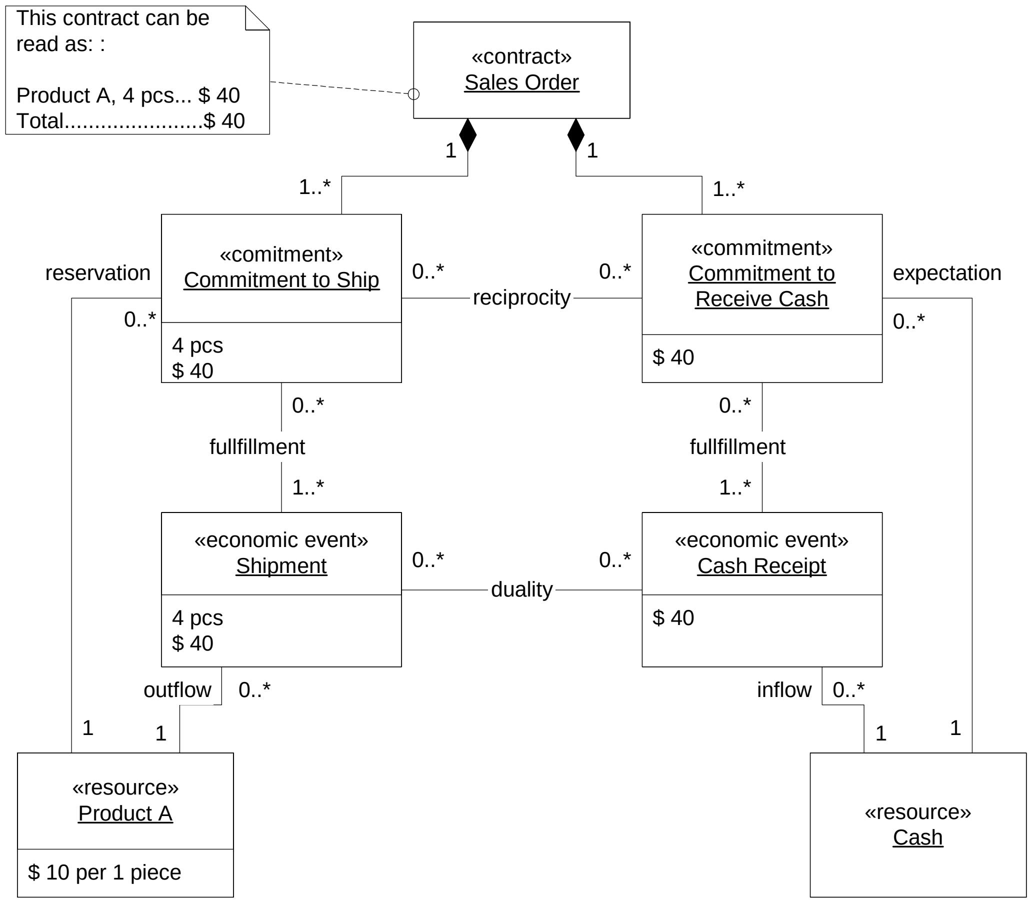


Fig. 8. Sales order contract with commitments and economic events

Known uses

Purchase order is a contract that consists of two commitments: a commitment to receive the ordered goods and a commitment to pay for the goods.

Employment contract represents a commitment by the employee to give his or her time to the employer, and a commitment by the employer give compensation in kind. Employment contract is an example of a contract when both parties agree on a stream of outflow economic events and a stream of inflow economic events.

Resulting Context

The commitment is a special case of business transaction, see BUSINESS TRANSACTIONS pattern. It covers those business transactions that are commitments, promises for changes in ownership of resources. Moreover, the contract pattern defines rules valid for *all* economic systems.

Prognosis of future events, for example budgeting, is not covered by this pattern.

Sometimes, this pattern forces developer to include to the model resources that are difficult to quantify and feel redundant. For example, the commitment to pay taxes to the government are reservation of resources, however, resources expected to be received in return from the government are difficult or even impossible to quantify precisely. The solution to this problem is called IMPLEMENTATION COMPROMISE: developers develop an ideal model of the enterprise that ensures that they have not forgot anything. Then, some contracts, resources or parties are omitted from the model. The resulting model is sufficient for the purpose of the software solution, and simpler than the complete model.

Related Patterns

The ADDRESSES pattern specifies at what place the commitment occurred and where the exchange of resources is supposed to occur.
The ACCOUNTS pattern specifies how to represent unbalances between commitments and the corresponding economic events.
The CLASSIFICATIONS pattern specifies how to handle various categories of commitments.
The DUE DATES pattern specifies when the economic event is scheduled to occur.

Credits and Sources

1. Geerts, G.; McCarthy, W.: The Ontological Foundation of REA Enterprise Information Systems, November 1999, March 2000, and August 2000.

2. Guido Geerts, The University of Delaware, Newark; personal communication, April 2002

Claims

Context

When a vendor ships goods, it usually does not receive customer payments at the very same moment. Outflow and inflow economic events usually do not occur simultaneously, and the duality relationship between the economic events is out of balance for certain period of time. In this case, a common practice is to send an invoice, a requirement to the business partner to settle the owed amount.

Problem

How can we model unbalanced economic exchanges?

Forces

1. Exchange of future economic resources must be fair, that is, agreed by both parties. How can we keep track of the fact that a party gives resources away and will eventually receive some resources in return?

2. Both business partners might not automatically know the exact unbalanced value between the inflow and outflow events. For example, a service contract might specify payments according to consumption. Or a vendor sometimes adds shipping fee to the price of products, whose value might not be known to the purchaser. You want to assure that both parties agree upon the unbalanced value.

3. Legal reasons might require a document specifying the unbalanced value. For example, VAT (value added tax) in Denmark is calculated as a percentage from the invoiced amount.

4. You want your model to be consistent from economic viewpoint. You could model the claim as a basic business relationship following the Fig.1, but you want your model give answers to questions like: if a customer receives a discount for early payment, what benefits does the vendor get in return?

Solution

Encapsulate the unbalanced value in the *duality*. The *claim* entity sums the value over one or more *dualities*. For example, when a vendor ships goods and the customer has not paid yet, the vendor can *materialize* a claim, that is, create an invoice. Each invoice line is related via the *materialization* relationship to the shipment events that have not been fully paid, yet. When the customer eventually pays for the goods, each invoice line will be related via *settlement* relationship to the payment. The *claim* entity is a placeholder for additional attributes such as DUE DATE.
This pattern is illustrated in Fig. 9.

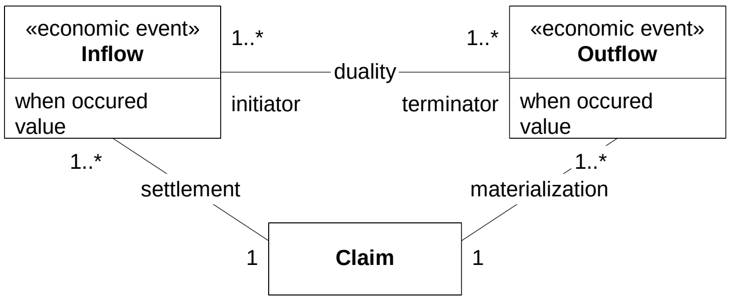


Fig. 9. The claims pattern

Related Patterns

The CLAIMS extends the REA pattern, as CLAIMS is applicable together with the duality relationship. Claim is usually contains due date, see the DUE DATES pattern.

Known uses

Invoice is a claim: pays us the money for the goods or services we provided to you.

Library's late notice that is sent out to you is a claim: bring back those books you owe us!

Credits and Sources

1. Bob Haugen, Logistical Software, communication via e-mail, November 2001- April 2002.

2. Guido Geerts, The University of Delaware, Newark; personal communication, April 2002

3. Daniel May, The University of Southern Denmark, personal communication, summer 2002.

Roles

Context

Sometimes, an employee of the company can also be a customer. For example, a bank clerk opens a bank account, or a nurse at the hospital becomes a patient. A vendor can become a customer, if he buys the company's product, and employees can become vendors, if the employer refunds them their purchases.

Similarly, a specific product can be considered as a raw material or finished goods. Or, the same working hour can be seen as an employee's time, or a consultancy hour sold to the customer. Although the resource is the same, when related to different business relationships, some of its properties will differ, for example, the unit price.

Problem

How can we model situations in which the same physical entity participates in different types of business relationships?

Forces

1. You used the business relationship pattern and your data model contains various kinds of business relationships, such as purchase orders, sales orders and employment contracts. A possible solution could be to replace all these kinds of business relationships by a "universal" business relationship entity, but this would not adequately capture the problem domain. For example, it is not very useful to have a single data entity representing both the sales order and the employment contract, because they have significantly different attributes.

2. The same party can be related to different business relationships, for example, to purchase and sales order. However, the party has different properties when participating in different business relationships. For example, the default ship-to-address is relevant on a party related to the sales order, but not on the party related to the employment contract.

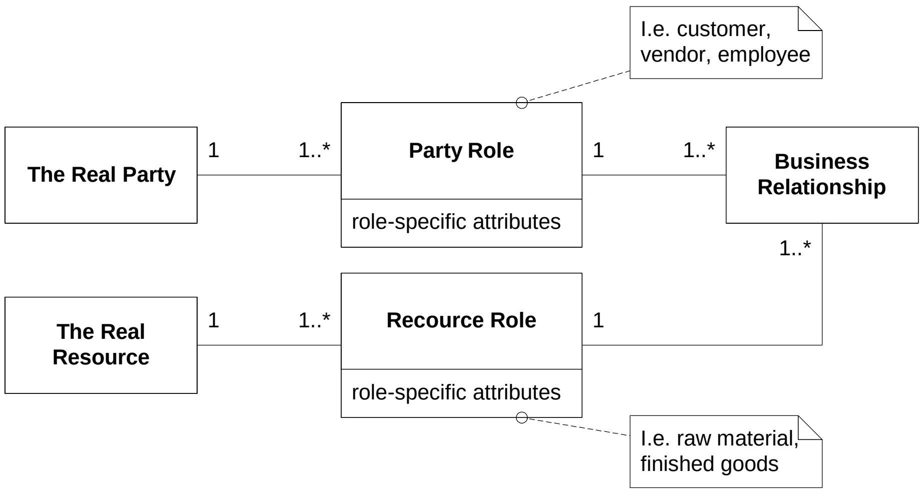
3. As a result of the second force, you could model customers, vendors and employees as parties in your model. However, you want to capture the information that all these three parties can be, in the real world, the same physical entity.

Solution

Split the party entity into *the real party* and the set of *party roles*. Split the resource entity into *the real resource* and the set of *resource roles*. The solution is illustrated in Fig. 10.

The real party and the *real resource* represent the physical objects participating in the business relationship.

The *party role* and the *resource role* are a way that *the real party* and *the real resource* can participate in a business relationship. Each real party and real resource have at least one role.



.
Fig. 10 The role pattern

Some might argue that the real party and resource have attributes common to all their roles. However, conceptually more correct would be to consider the common *role* attributes in the implementation of the *role* entity, for example, moving common attributes to the role superclass. Various implementation approaches to roles are discussed in [5].

Known uses

1. Specific product can have the roles of a material used in work, or finished goods sold to the customer. These two roles have different attributes: the product as material might be used together with mounting fittings. The same product as a goods sold is used without the fittings. Moreover, the unit price for the same product is different if used as a material or if sold directly. Source: Country Union of Danish Electricians (ELFO).

2. A person can play a role of a customer, vendor and employee.

Resulting Context

The roles pattern adds flexibility and extensibility to the model and prevents from duplicated information. However, this is not always the desired behavior. For example, in a hospital information system a user requirement was to physically separate the patient data and the employee data, even if they represent the same person (source: Ralph Johnson, personal communication).

Credits and Sources

1. Martin Fowler’s paper “Dealing with roles” focuses on implementation issues of the roles. For details see http://martinfowler.com/apsupp/roles.pdf

2. Dirk Riehle’s paper Role Object describes a generalization of the role concept for any component (a particular key abstraction). For details see www.riehle.org.

Accounts

Also known as balances

Context

The BUSINESS TRANSACTIONS pattern describes how to keep track of and registers business relationships, economic events, and commitments. However, merely keeping track of these entities is usually not the main interest of an enterprise's decision makers. Decision makers are mostly interested in *reports*, the aggregated data that summarize registered entities, and provide the information about the state of the enterprise.

Problem

How can we model the aggregated values across sets of business transactions?

Forces

1. The business transaction pattern allows for tracking of business transactions and their values. However, users of enterprise planning systems would like to get information about aggregated values from the sets of business transactions, economic events and commitments.

2. You could manually write an algorithm that aggregates the values from business transactions, for example, in the form of SQL statements. However, you would like to know a rule, valid for *all* economic systems, that would automatically provide for aggregated values across sets of business transactions, events, and commitments available in the system.

Solution

If a party or resource is related to at least two business transaction entities, then the party or resource has a property called *account*. The *account* summarizes the values of all related instances of one business transaction and subtracts them from a sum of the values of all related instances of the other business transaction. The solution is illustrated in Fig. 11.

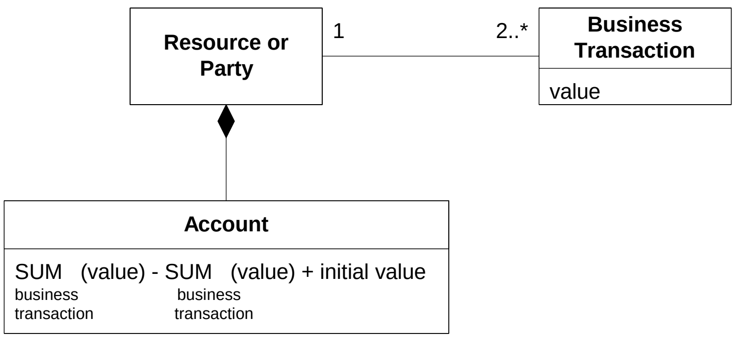


Fig. 11 The accounts pattern

The following figure shows an example of the system with four business transactions: the purchase order, receipt, sales order and shipment. The resource *goods* has three accounts. The account *goods* *on stock* is a sum of all goods receipts minus all shipments. The account *goods on order* (goods to be received) is a sum of all purchase order lines minus sum of all the receipts related to purchase order lines. The account *goods to ship* is a sum of all sales order lines minus the sum of all the shipments related to sales order lines.

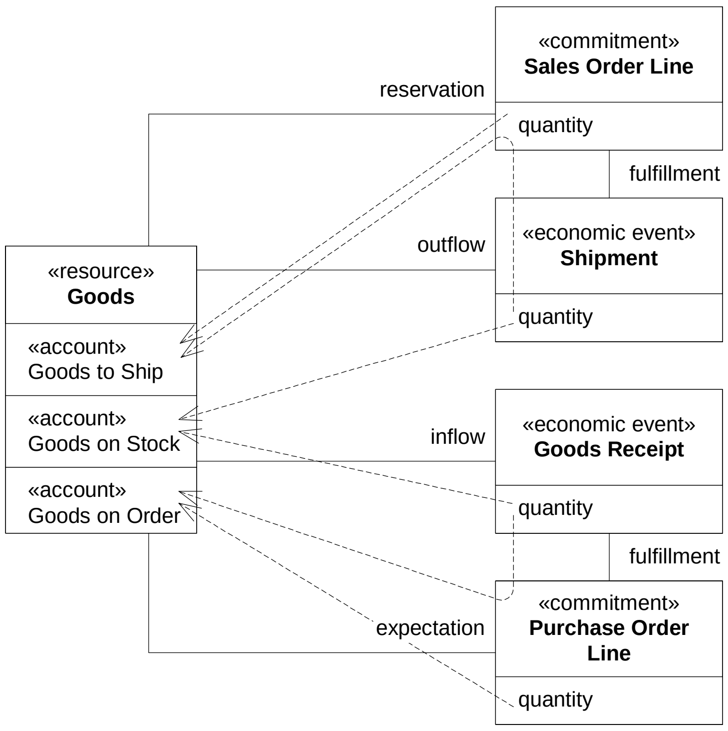


Fig. 12 Examples of accounts on resource

Please note this solution does not describe *all* accounts that exist in the accounting practice.

Known uses

Accounts receivable. For example, the customer balance is an account. It is a sum of all sales orders minus sum of all customer payments.

Accounts payable. For example, vendor balance is an account. It is a sum of all purchase orders minus the sum of all payments to the vendor.

Assets. For example, cash is an account on the company itself. It is a sum of all cash receipts minus sum of all cash disbursements.

Resulting Context

If the business transaction, in addition to its value property, specifies additional information, attributes and properties, the account pattern can be extended to provide selective sums, and aggregate balances for certain values of these additional attributes. For example, if a payment contains due date, see the DUE DATES pattern, the customer account can be refined to reflect all expected payments that are over due and the expected payments that are not over due.

Due Dates

Also known as Deadline

Context

Commitments are promises of the future economic events. Sometimes it is useful to specify when, at what time, the future economic events shall occur. Starting date, last payment date and renewal date are examples of due dates.

Problem

How can we model when should future economic events occur?

Forces

1. Dates and time intervals are attributes of business relationship, business transaction, economic event, commitment and claim. You could add the date-time attribute to all of them, but you want to have a uniform behavior for all these entities in your model.

2. Some future events can be recurrent. Examples of recurrent events are periodic shipments, or meetings that occur every week. There might be complicated rules specifying the recurrence.

3. You want to specify what happens when due date expires.

Solution

Encapsulate the attributes necessary to setup the due date in the *Due Date* entity. The time interval for due date is specified, for example, by *Start date* and *Duration*. The *has expired* attribute specifies whether the due date has passed. Examples of *recurrence rules* are “every day”, “every week”, “first Monday in a month”, etc. Examples of *reminder rule* are “on due date”, “15 minutes before start”, “one day later”, etc. The TYPE OBJECT often extends this pattern by providing for *due date type* that encapsulates a list of various rule types that might be set by the *due date*. The structure of the Due Date pattern is in Fig. 13.

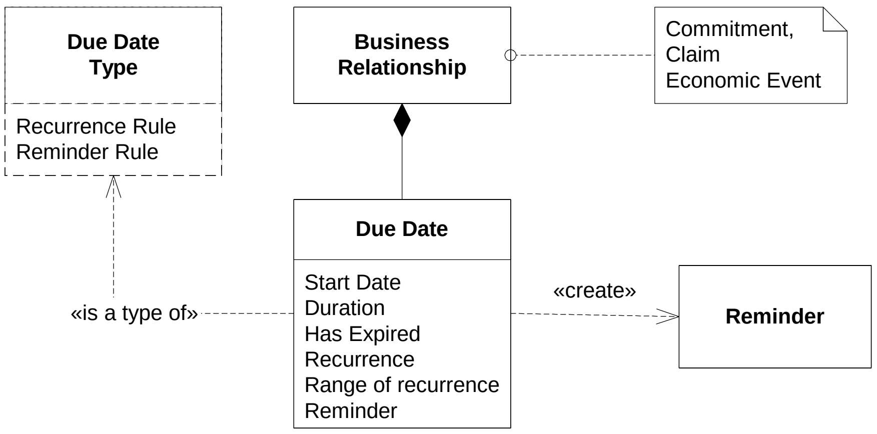


Fig. 13.The due date pattern

Known uses

Registration deadline for a conference is an example of a due date.
Payment schedule, or recurrent meeting in Microsoft Outlook and other personal information managers are examples of a recurrent due date.

Resulting Context

The due date defines the necessary attributes, behavior and rules relevant for setting up the due dates. But what actually happens when due date expires? This information has to be somehow forwarded to the rest of the system or users. The NOTIFICATIONS[2] pattern solves the user notification.

Related Patterns

TYPE OBJECT by Ralph Johnson et al. provides for flexibility by listing applicable recurrence and reminder rules in a *Due date type* entity.
CALENDARS[3] of a party or resource provides an aggregated view over all due dates of all business relationships related to this party or resource.

[2] The NOTIFICATIONS pattern is not part of this pattern language, yet.
[3]The CALENDARS pattern is not part of this pattern language, yet.

Addresses

Context

Parties and resources are usually located in a certain places in the real world. For example, the ship-to address specifies the location of the customer within the scope of the shipment event. However, the physical address is not the only way to contact a party. Phone number, e-mail and URL are examples of communication addresses. These addresses can change in time, for example, a person at the office can be contacted by direct phone number, but during meetings can be contacted by phone via the receptionist. Some communication addresses are public and accessible by any party, whilst others are available only to a restricted set of parties. Resources have communication addresses as well. For example, a product is placed in a certain warehouse location, and its description can be found through its URL.

Problem

How can we contact a party or locate a resource?

Forces

1. You want a uniform mechanism of establishing communication with a party or locating a resource. You can decorate the party or resource object with attributes such as e-mail, url, but this solution does not provide any information regarding when each is address valid.

2. You want to capture the fact that some communication mechanisms are restricted to a certain set of parties and not available to everybody.

3. You want to specify communication addresses for a party, such as ship-to address and bill-to address, but these addresses are often specific to the business relationship, rather than to the party.

Solution

Encapsulate the functionality about how to locate a resource or party in the *address* entity. The *address* is an abstract concept for the *communication address* and the *geographical address*. Examples of communication addresses are the phone number, e-mail and URL. Examples of geographical addresses are the postal address and geographical coordinates. The *business relationship* specifies the validity time interval of the address. The business relationship also specifies what parties the address is available to. The solution is illustrated in Fig. 14.

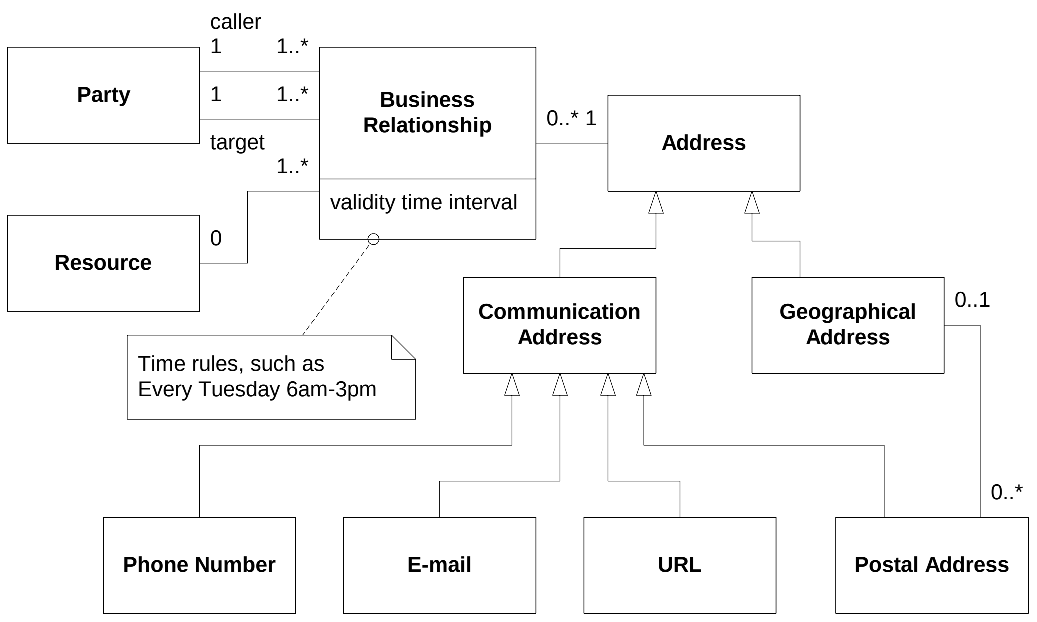


Fig. 14 Addresses

Known uses

A sales order must have a sell-to, bill-to, and ship-to address. These can all be postal addresses, as it may be necessary to send information material such as a confirmation to the customer. However, depending on the company and the customer these addresses could also be electronic addresses.

Within Human Resources department of a company, all addresses of employees must be stored. These addresses must be both postal addresses and phone numbers. The employee's phone numbers and physical locations during working hours and outside working hours are different.

Resulting Context

This pattern specifies that parties and resources are not directly related to their addresses, they are related via the business relationship entity. This is a consequence of keeping track of valid addresses over time, and focusing on the communication aspect of the address. In simple cases, this extra complexity would not pay off the benefits of using this pattern, and the address will be related directly to the party or resource.

Related Patterns

The CLASSIFICATIONS pattern can be used to specify a group of the caller parties. For example, in the postal address, which is public, the caller is usually the "any" party category. For e-mail address or direct phone number, which might be restricted, the caller might be a category "trusted party", which would imply that only parties with the "trusted party" classification might be related to (that is, to know about) this e-mail address or phone number.

Credits and Sources

1. The paper “How Do I Find You? (Let Me Count The Ways.)” by Terry Moriarty and Dwight Seeley describes the main ideas of this pattern. For details see www.inastrol.com

2. The address pattern is a part of IBM SanFrancisco framework, for details, see: http://www.ibm.com/software/ad/sanfrancisco

Classifications

Context

Sometimes, a company partitions its customers into various categories, for example, high-volume customers and small-yield customers. Sometimes, there is a hierarchy of categories. For example, the furniture products can be classified into the categories office and home. The home category has subcategories, such as sofas, chairs, tables, and beds. The sofas category has subcategories leather sofas, cloth sofas and sleeping sofas. Sometimes, the classified object can belong to several categories; for example, a specific sofa can be both sleeping sofa and leather sofa.

Problem

How can we model categories of products or parties in a uniform way?

Forces

1. You want to make a uniform model for, for example, event categories, party categories, product categories and business relationship categories.

2. It should be possible to categorize different business relationships, resources or parties using the same classification hierarchy.

3. When an object changes its category, it still belongs to the same class. That is, it still has the same attributes, properties and methods. It might change the values of the attributes, but, for example, does not get a new attribute when it changes its category. Note: the TYPE OBJECT should be used if this force is not applicable.

4. The entity typically does not change its category when its attributes change. For example, let's suppose that the payment entity belongs to different priority categories. If due date of the payment is over, the payment entity still belongs to its original priority category. Note: use the LIFECYCLES pattern if this force is not applicable.

Solution

The business relationship, economic event, commitment, contract, claim, resource and party can be related to the *category* entity. The category entity can belong to a hierarchy of category entities, that is, a category is related to a *supercategory* and *subcategories*. In general, the relationships between them are many-to-many. The business relationship, resource or party can belong to several categories. The category can belong to several supercategories and might have several subcategories. Solution is illustrated in Fig. 15.

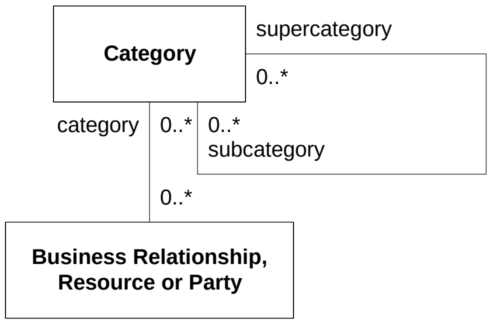


Fig. 15. The classifications pattern

Known uses

Subject is the classification of information resources into categories such as Business & Economy, Computers & Internet, Entertainment, Recreation & Sports, etc.

Skill (qualification) is a classification that assigns employee to certain skill groups. Skill is an example of a classification in which party can be assigned to more than one category.

VAT (value added tax) group is a classification of the product into the VAT groups.

Resulting Context

The business relationship, resource or party can be assigned into several categories simultaneously. The consequence of using this pattern is that the classified object does not change when its category changes. For example, if a product changes its VAT group, it is still the same product. On the contrary, we might consider a "party" object, categorize it into a "customer" or "employee" categories. Application of this pattern would imply that the attributes and methods are the same both for "customer" and "employee". If it is not the case, the TYPE OBJECT pattern should be applied instead of classification in this particular situation.

Related Patterns

The LIFECYCLES can be used to categorize the objects as well. For example, imagine that we create the following categories for the order: quotation, accepted, shipped, paid. We could use the classification pattern. However, it would be impossible to specify the rules like: the quotation is first accepted, then shipped and then paid; or the rules like: If the order is in the category "paid", it is not allowed to change the category to "quotation". In this case, the LIFECYCLES should be used, instead of CLASSIFICATIONS.

TYPE OBJECT can be used to categorize the objects as well. For example, imagine that we create the following categories for the sales order: quotation, accepted order, shipment, and payment. If the sales order in these four categories has different attributes and behavior, for example, the shipment specifies the products but not prices, and payment specifies the prices but not products, then TYPE OBJECT should be used. That is, the quotation, accepted order, shipment, payment should be different types, and not categories of the same type.

Credits and Sources

The classification pattern is a part of IBM SanFrancisco framework, for details, see:
http://www.ibm.com/software/ad/sanfrancisco
.

Acknowledgements for the whole Pattern Language

I would like to thank to Daniel May of Maersk Mc-Kinley Institute, University of Southern Denmark, Odense, Denmark, for shepherding this paper, and for his useful suggestions and comments. I do, of course, take full responsibility for any omission or errors.

I would like to thank to Jesper Kiehn, Microsoft Business Solutions, Copenhagen, Denmark, for fruitful discussions, valuable comments, observations and ideas.

References

[1] Coad, P., Lefebre, E., DeLuca, J.: Java Modeling in Color with UML, Prentice Hall PTR, 1999.
[2] David, J. S.: Three events that defined an REA methodology for systems analysis, design and implementation
[3] Ericsson, H., Penker, M.: Business Modeling with UML, John Wiley & Sons, 2000.
[4] Fowler, M.: Analysis Patterns, Addison Wesley Longmann, 1997.
[5] Fowler, M.: Analysis patterns (articles on the web), http://martinfowler.com/articles.html#
[6] Guido L. Geerts and William E. McCarthy: The Ontological foundation of REA Enterprise Information Systems, 1999-2000.
[7] Hay, D.: Data Model Patterns, Conventions of Thought, Dorset House Publishing, 1996.
[8] Hollander, A., Denna, E.L., Cherrington, J.O.: Accounting Information Technology and Business Solutions, Irwin McGraw-Hill, 2000.
[9] Riehle, D.: Association object patterns, http://www.riehle.org/practical-matters/patterns/business/association-objects/index.html
[10] Riehle, D.: Role Object. http://www.riehle.org/computer-science-research/1997/plop-1997-role-object.html
[11] Silverston, L., Inmon, W. H., Graziano, K,: The data model resource book, John Wiley & Sons, Inc. 1977.
[12] William E. McCarthy: The REA Accounting model: A generalized framework for accounting systems in a shared data environment. The accounting review (July) pp. 554-578, 1982.