

Obligation-Fulfillment: A Pattern Language for Some Financial Information Systems

Lubor Sesera

Softec, Ltd.
Kutuzovova 23, 831 03 Bratislava, Slovakia
&
Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovicova 3, 842 16 Bratislava, Slovakia

e-mail: lubor.sesera@softec.sk
phone: (+ 421) 2 49 49 00 00
mobile: (+ 421) 9 03 78 60 55

This paper addresses the issue of analysis patterns for some financial information systems. In particular, they deal with information systems based on recording financial obligations of various kinds and their fulfillments. This spans a large number of domains and company/institution sizes: from commercial companies to public and government institutions and from small companies to complex institutions.

Although financial information systems are among the most expanded systems, few analysis (domain) patterns going into depth have been published. Coad's patterns [Coad 1997] are rather general and oversimplified. In [Hay 1996] the main focus is in modeling an enterprise and financial patterns play only a marginal role. Keller's patterns [Keller 1998] are restricted to a specific domain, insurance systems in particular. Fowler's patterns [Fowler 1997] are paragons of right abstraction, however, altogether they are just independent islands not forming a broad pattern language. Our aim has been both to provide patterns that are 'general enough' to cover a number of systems and still having enough 'depth' to be helpful when building specific models. This is highlighted with added examples.

Patterns described here are rather simple. However, together (including variants) they provide quite a powerful pattern language. The patterns can be combined in various ways in order to cover distinct types of systems and several options. Table 1-1 provides a summary of patterns in this paper. In the following two diagrams the main dependencies among patterns are shown. Figure 1-1 shows the 'evolution of patterns, i.e. how patterns 'grow' from simple patterns such as Basic Obligation through Summary Obligation to complex patterns such as Accounting. Complementing this, Figure 1-2 sketches out how patterns can be combined. For instance, Event-Based Claim can be combined not only with Payment Methods but also with Partial Fulfillment and Accounting. In the background of all of these patterns the general concept of Obligation can be found. Furthermore, in the first five patterns the key concepts (Basic Obligation, Recurring Claim, Event-based Claim, Partial Fulfillment and Summary Obligation) *are* Obligations and in this way they have the same fundamental structure. This structure is

sketched out in the Basic Obligation pattern. The difference is that these Obligations have distinct roles. In this paper the emphasis is put on these roles.

So far, patterns and pattern languages that have been published are ‘flat’. In [Sesera 2004] we introduced hierarchical patterns as a means to balance generality and usefulness forces in analysis patterns. In hierarchical analysis patterns several layers of analysis patterns are provided, each level offering the proper generality and abstraction with regard to its objective. The most general layer gives a framework while the most specialized layer provides a rather specific guideline for an analyst building a conceptual model of a real-world system. From the three layers hierarchy introduced in [Sesera 2004], patterns described in this paper form a specific second layer of ‘specialized analysis patterns’. Most of our patterns address the aspect of ‘why’ to specialize the Accountability general analysis pattern [Fowler 1997]. In Party’s Account, Accounting and Payment Methods patterns, however, the aspect of operations plays an import role as well.

Pattern	Problem	Solution
Basic Obligation	How do you represent many types of financial documents between parties in a common way?	Generalize various types of financial documents to the concept of Basic Obligation.
Recurring Claim	How do you handle recurring and variable obligations?	Split obligation into Basic Obligation and Recurring Claim.
Event-based Claim	How do you handle event-based obligations rooted in the same obligation?	Split obligation into Basic Obligation and Event-based Claim.
Partial Fulfillment	How do you represent partially honored obligations?	Decouple Partial Fulfillment from Basic Obligation/Recurring Claim/Event-based Claim.
Summary Obligation	How do you put together distinct obligations of the same party?	Create Summary Obligation that aggregates particular obligations.
Party’s Account.	How do you settle mutual obligations between two parties?	Create Party’s Account.
Accounting	How do you represent various points of view of the same obligation and maintain integrity?	Generalize Party’s Account to Account and use accounting transactions.
Payment Methods	How do you handle various methods of payments?	Generalize commonalities to the Payment and Payment Destination concepts.

Table 1-1 *Summary of patterns*

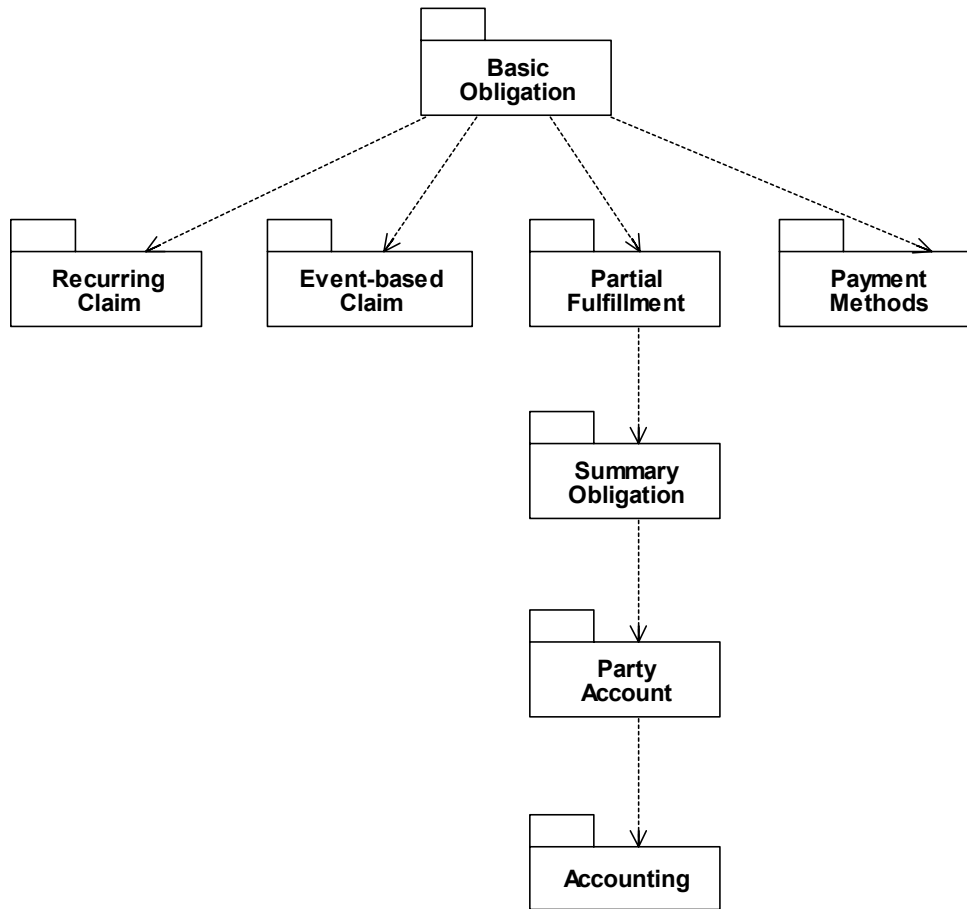


Figure 1-1 *Evolution of patterns*

Abstract models are patterns only when they are useful and have been used. There are always problems with analysis patterns with regards to known uses. Business models are usually company secrets and they are published rarely. That is why we cannot provide any real-world applications apart from our own. Although the essence is the same, many details may be different depending on the country. The social system is the example for the above.

Due to space, only static models of patterns are provided in the patterns' description. For this UML class diagrams are used. Both requirements (e.g. in the form of uses cases) and dynamics (e.g. in the form of sequence diagrams) are omitted. Furthermore, meta-level is shown only in some diagrams, so that the patterns' essence can be highlighted. As Martin Fowler used to say: 'a pattern is an idea' [Fowler 1997] rather than a complete solution. This paper is nothing more than a small set of such ideas.

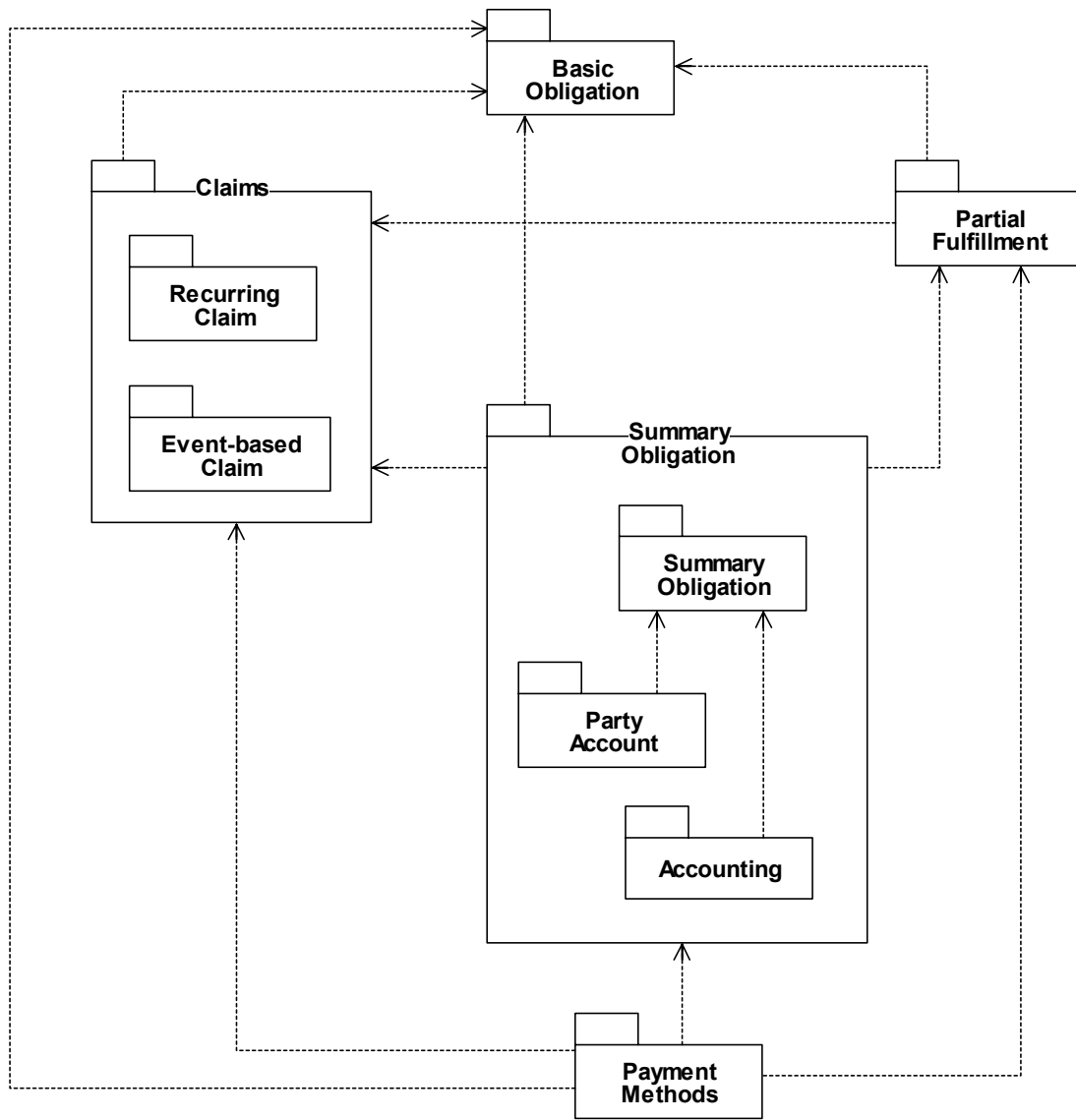


Figure 1-2 *Combination of patterns*

Basic Obligation

Also Known As

Contract, Accountability

Context

There are many kinds of financial relationship between parties. Historically, the most traditional ones are associated with selling goods. Later, businesses with intangible goods appeared: drafts, shares, bank obligations, etc. In the modern era¹, the importance of 'government-to-people' financial relationships has also increased: old age pension, social benefits, unemployment benefits, government-supported health care. Financial relationships are usually represented in the form of legal documents. As there are many kinds of financial relationships there are many types of legal documents: contracts, purchase orders, invoices, agreements, government registrations, applications and others. These documents share many common features but they also have their own specific aspects.

Example(s)

An insurance company has 'sold its product' as car insurance and makes a policy with a client to insure his car against damage. Prior to this policy the client has had an obligatory² insurance policy for the same car with this insurance company.

A government agency records applications for state social benefits. There are about a dozen basic types of social benefits, e.g. housing benefits, child benefits, disability benefits and others. The same person may apply for several such benefits fulfilling several application documents. The approved application is valid for one year.

A company selling many types of products receives an order from its regular customer for some product types.

Problem

How do you represent many types of financial documents in a common way so that they can be manipulated in a consistent manner and the implementation of behavior can be shared?

Forces

- There are many types of financial documents, each one having its specifics depending on a particular business.
- It requires a lot of effort to design, implement and maintain a system with lots of special cases.
- Different types of financial documents share many common features.
- Often documents include the same data on parties and/or objects of financial relationships.
- Documents also contain specific information on parties/objects and various departments within the same organization need to represent different internal information on parties and objects.

¹ In Europe at least.

² Prescribed by government law.

Solution

Generalize various types of fundamental financial documents to the concept of Basic Obligation. Define the uniform fundamental structure of Basic Obligation so that the representation of documents becomes as similar as possible. Basic Obligation contains the financial amount(s), the period of validity and due dates. Due to normalization the concept of Party³ is decoupled from Basic Obligation and Basic Obligation has only an association to the Party object. If it makes sense, Object of Obligation⁴ may be decoupled from Basic Obligation in the same way. Try to build a meta-level: meta-concepts and their relationships provide extendibility and declarative constraints.

The structure of the pattern in its simple variant is shown in Figure 1-3⁵.

Note 1: It would be more readable but less precise to use the term 'obligation' instead of 'Basic Obligation' here. 'Basic Obligation' represents the fundamental document which can be addressed by subsidiary documents (see other patterns later in this paper) while 'obligation' is the general concept generalizing obligations of both kinds, namely 'Basic Obligations' and subsidiary obligations. It is worth mentioning that the structure provided here for Basic Obligation is valid also for the general obligation and in this way (with some exceptions) applicable also to subsidiary obligations.

Note 2: Similarly, it would be more precise to use the terms Basic Obligation Type instead of just Obligation Type or Object of Basic Obligation instead of Object of Obligation. For readability, however, we use the shorter names although they are less precise.

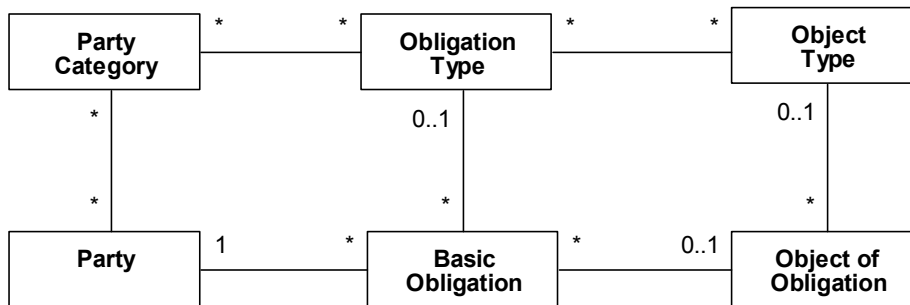


Figure 1-3 *The Simple Basic Obligation pattern*

³ Party is the general concept from [Hay 1996] and [Fowler 1997] for person, organization and position held by a person.

⁴ Here Object of Obligation is the general concept including Hay's [Hay 1996] concepts of Object, Activity and Party.

⁵ Due to accepted practice in analysis patterns, we do not show attributes and methods of objects in this paper. In Figure 1-3 the obligation amount, validity period and due dates attributes are not shown.

Variants

- Object of Obligation does not have to be a separate object but rather an implicit part of Basic Obligation. This variant is used when Object of Obligation is not shared among Basic Obligations.
- Aggregate Basic Obligation (see Figure 1-4⁶). Basic Obligation includes several parts of potentially many types. To preserve sharing of features these many types of parts are generalized to the Obligation Line Item concept. Obligation Line Item behaves like a 'mini-Obligation' that can have its amount and it may also refer to a subset of Obligation Parties and Objects of Obligation. Basic Obligation usually utilizes only a small subset of potential Obligation Line Item Types.
- Multi-party Basic Obligation with party roles (see Figure 1-5). There are several parties associated with Obligation. Distinct associations to parties are generalized to the Party Role concept. A Party Role Type is assigned to Party Role (another variant is that complementary to Party Role, there is the direct association between Obligation and the primary Party.) The Party Role concept can be used also to record party specific information related to certain types of Obligations.
- Combined Aggregate and Multi-party Basic Obligation.
- Basic Obligations can be associated, e.g. where an obligation is a replacement of another obligation, e.g. a new insurance policy replaces an old insurance policy. The new policy needs to reference the old policy so that previous benefits, claims, etc. can be ascertained. Similarly, Obligation Line Items can be associated.
- In many cases meta-level is not built.

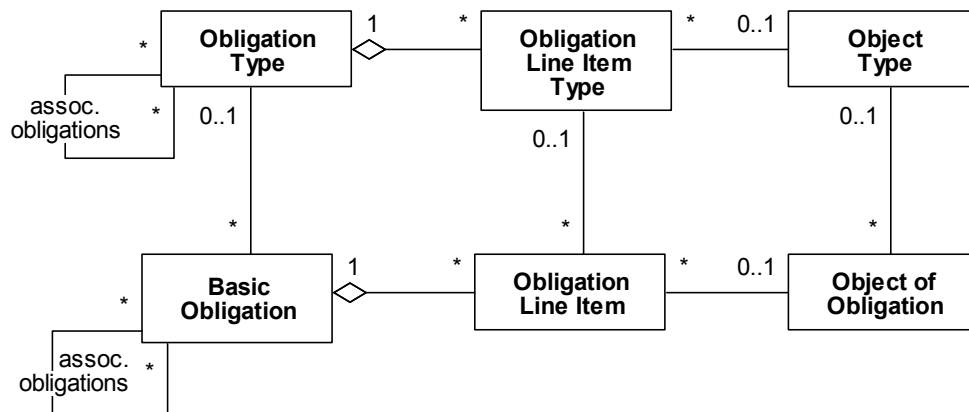


Figure 1-4 The Aggregate Basic Obligation subpattern with associated obligations

⁶ For simplicity the concept of Party is omitted in the diagram.

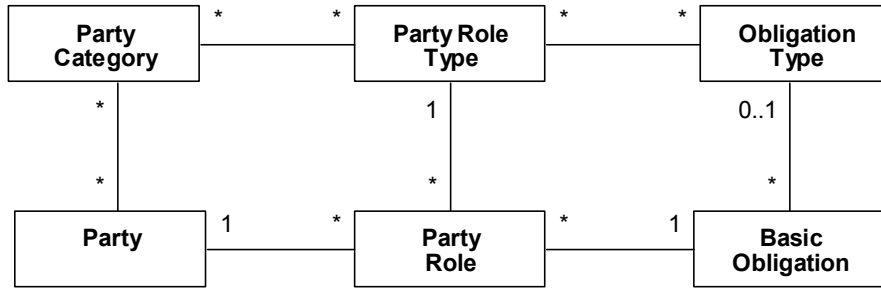


Figure 1-5 *The Multi-party Basic Obligation subpattern*

Example(s) Resolved

The insurance policy is (a special kind of) Basic Obligation. It references the client object (Party) as the client can have more than one policy.⁷ It also references the car object as the same car can be the Object of Obligation of several policies. Product is the Obligation Type. For more complex variants see Known Uses.

The approved application for a state benefit is Basic Obligation of the government to the applicant. The applicant is represented as the Party object that can be referenced from several benefits. The benefit type represents the Obligation Type.

The order is Aggregate Basic Obligation. It contains line items (Obligation Line Item). Each line item references a product type (Object of Obligation). As many clients are regular clients the object of client (Party) is decoupled from Basic Obligation.

Resulting Context

Different types of financial documents share the same fundamental representation.

Basic Obligations are decoupled from Parties so that these objects can be handled independently.

Several Basic Obligations can share the same Party and/or Object of Obligation.

Complex Basic Obligations consisting of parts and/or addressing several Parties can be handled in a uniform way.

Implementation is less efficient as table joins are needed.

As complex Basic Obligations consist of several objects, a sophisticated user interface is needed to mask normalization.

⁷ Furthermore, he can be Object of Obligation in another policy.

Known Uses

- Simple Basic Obligation: In the CDPK card management system at VUB bank [CDPK 2003] Basic Obligation due to denormalization includes Basic Obligation merged with Object of Obligation, i.e. both a contract and a card. The model contains quite complex meta-level as cards are classified from several points of view.
- Simple Basic Obligation: In the Broker system for selling insurance products [Broker 2000], [Sesera 2003] Basic Obligation is a contract between a self-employed broker and a broker company. The contract is quite simple: neither the Object of Obligation nor the Obligation Type concepts have been applied.
- Aggregate Basic Obligation: In the Koop system for insurance companies [Koop 1999], [Sesera 2000a] Basic Obligation is specialized to the policy concept (an insurance contract). Obligation Line Items are policy risks addressing insured objects.
- Aggregate Basic Obligation: In the Information System of the General Health Insurance [Health 1996] Basic Obligation is a contract between General Health Insurance and a health care provider. Where the health care facility consists of departments, the contract contains line items.
- Multi-party Basic Obligation: In the Koop system a policy is associated with several parties such as a policyholder, a premium payer, a broker, an underwriter and others. Number of Party Roles involved depends on the product and the complexity of the policy.
- Multi-party Basic Obligation: In the ISOP system of state social support [ISOP 1999], [Sesera 2000a] Basic Obligation takes the form of an application for a specific benefit. The application is usually associated with several parties playing various roles. The permissible roles are defined as Party Role Types for each benefit type.
- Multi-party Basic Obligation: In the EPP system for collecting obligatory employment insurance [EPP 2001], [Sesera 2000b] Party Role is decoupled from Party as Party can play several roles simultaneously, e.g. being an employee and a self-employed person at the same time or Party Role can be temporal. In the EPP system the registration concept represents a specialization of the Basic Obligation concept.
- Contrary to a previous application, the Factoring system [Factoring 1996] is an example of a system with a fixed number of parties. There are four parties involved at most: seller, buyer and two factors. A 'virtual' multi-party contract has three direct associations with parties, i.e. the Party Role concept is not used.

Related Patterns

In general, the Basic Obligation pattern corresponds to the Accountability pattern [Fowler 1997], the Contract pattern [Hay 1996] and the Transaction patterns [Coad 1997]. There are however some differences. The Basic Obligation pattern is constrained to financial obligations and does not include other applications such as an organization structure incorporated into the Accountability pattern. On the other hand it is also more general as it includes not only traditional business obligations such as contracts, orders and invoices, but also 'government' business obligations such as applications, registrations, etc.

Hay's Contract pattern includes also Line Items [Hay 1996].

The Multi-party variant of the pattern is the variant of the Role Object pattern [Bäumer+ 2000]. It can also be found as Contract Roles in [Hay 1996].

Basic Obligation is the fundamental pattern of this pattern language. The other patterns are its add-ons.

Recurring Claim

Context

Obligations that regularly reoccur at specific time periods. Each such obligation can differ slightly from its predecessors. This may be due to an agreement made at the beginning of the financial relationship or due to conditions that have since changed.

Example(s)

In insurance a client and an insurance company sign a policy specifying fundamental terms and conditions. Both parties agree that the insurance premium is to be paid gradually, e.g. quarterly. The insurance amount and premium are indexed, i.e. they are evaluated and changed to follow inflation rates. Furthermore, there might be some agreed deferrals on payments for some periods, penalties due to late payments, etc. There is a requirement ascertain the obligation amount for any chosen time in the past.

Problem

How do you handle recurring and variable obligations?

Forces

- The model is easier to understand when recurring obligations are kept in the form of a single object.
- Recurring obligations may differ slightly in the amount of money to pay, which is difficult to represent within a single object.
- It is space effective to have the recurring obligation as a single object.
- Queries are more effective when the same computation is not performed again and again.
- It should be easy to ascertain the amount of the obligation at any point in time.

Solution

Split the obligation into its solid and (potentially) variable part. Basic Obligation represents fundamental terms and conditions while Recurring Claim⁸ represents a particular claim valid

⁸ Another option might be to name it Recurring Obligation. Here, the term Recurring Claim was chosen to set off the concept role - that can be amended by its Fulfillment (discussed later).

within a time period. Generate Recurring Claim from Basic Obligation in specified time intervals.

Variants

Recurring Claim itself is an obligation. Thus it can appear in any variant described with Basic Obligation:

- Simple Recurring Claim
- Aggregate Recurring Claim
- Multi-party Recurring Claim
- Combined Aggregate and Multi-party Recurring Claim

Example(s) Resolved

Apart from the insurance policy representing Basic Obligation insurance subscriptions are generated and stored in time periods defined within the insurance policy. Insurance subscriptions can be easily checked by a user and they are stored for further queries and computation.

Resulting Context

Recurring Claims decoupled from Basic Obligation make it easier to ascertain the actual obligations and to trace their history.

Queries and computation based on stored Recurring Claims are performed faster and programs are less volatile to programming errors. This is achieved at a cost of space.

When Recurring Claims do not diverge from Basic Obligation and/or computing procedure is simple, representing recurring claims as separate objects yield redundancy.

Known Uses

- In the CDPK card management system [CDPK 2003], monthly fees are charged to cards of specific types. The model is the application of the Aggregate Recurring Claim variant. Card-contract represents Basic Obligation. The fee operation plays the role of aggregate claim; particular fees are line items. Note that there can be several fees associated with the operation.
- In the insurance business an insurance premium can be paid gradually. In the Koop system [Koop 1999], [Sesera 2000a] premium subscriptions are generated for policies monthly. The premium subscription plays the role of Simple Recurring Claim.
- In the ISOP system [ISOP 1999], [Sesera 2000a], certain state benefits are paid monthly or quarterly. The generated benefit is the Multi-party Recurring Claim. Contrary to previous systems where the Recurring Claim pattern was used for money income (receivables), in the ISOP system the pattern is applied for money outstanding (liabilities).

- The EPP system for collecting employment insurance [EPP 2001], [Sesera 2000b] is a mixture of Recurring Claim and Event-Based Claim. As 'government' insurance must be paid monthly, a premium subscription (Recurring Claim) is generated monthly for each valid registration. However, the premium subscription also depends on an 'income document' (e.g. individual tax return, employer's monthly statement, etc.) The 'income document' plays the role of Event-Based Claim. If this document is missing values from the previous month and other defaults are used.

Related Patterns

The Recurring Events pattern [Fowler 1996] focuses on the issue of many different events recurring at different times without addressing any (financial) obligations. On the contrary, the aim of the Recurring Claim pattern is a financial obligation and not the time at which its generation is scheduled. All kinds of Recurring Claims are usually generated together and regularly (e.g. each day or once a week). If there is a need, however, these patterns can be used together (combined).

Event-based Claim is a pattern in the same category: both are claims supplementing Basic Obligations. While Recurring Claim is triggered by a recurring time event, Event-based Claim is triggered by an explicit request of a party.

Sometimes Recurring Claim is combined with Summary Obligation. However, combination with Partial Fulfillment is rare.

Event-based Claim

Context

Obligations that are based on the same Basic Obligation are emerging due to specific events. Unlike in Recurring Claims, these events are not (time) events known in advance to the party, but they are explicit requests of the counterparty.

Example(s)

In insurance the client has a car accident and requests the insurance company to cover the cost of car repairs.

Problem

How do you handle event-based obligations rooted in the same Basic Obligation?

Forces

- The model is easier to understand when all the event-based obligations are kept in the form of a single object.
- Timing and obligation amounts of event-based claims are hard to predict

- It is space effective to have obligations as a single object.
- Queries are more effective when the same computation is not performed again and again.

Solution

Split the obligation into its solid and (non-predictable) variable part. Basic Obligation represents fundamental terms and conditions while Event-based Claim represents a particular claim of the counterparty. Create Event-based Claim when it is received from the counterparty.⁹

Variants

Event-Base Claim itself is an obligation with variants:

- Simple Event-based Claim
- Aggregate Event-based Claim
- Multi-party Event-based Claim
- Combined Aggregate and Multi-party Event-based Claim

Example(s) Resolved

The client's loss notification is the Event-based Claim referencing the insurance policy (Basic Obligation). It is an Aggregate Event-based Claim as there could be several cars and other objects¹⁰ damaged in the accident.

Resulting Context

Events and Event-based Claims are decoupled from Basic Obligation. This clarifies the model and enables event tracking at a later date.

Event-Based Claims can be associated due to their relationships to Basic Obligation.

Known Uses

- The CDPK card management system of the VUB bank [CDPK 2003]. Aggregate Claim is a client operation with a card, e.g. an application to change certain contract values or an application for a card recovery. Particular fees are line items.
- The Koop system for the Kooperativa insurance company [Koop 1999], [Sesera 2000a]. A policyholder's insurance claim is a specialization of Aggregate Multi-party Event-based Claim.

⁹ In contrast to Recurring Claims that are periodically generated.

¹⁰ For example a traffic sign or a crash barrier.

- The Broker system for selling insurance products [Broker 2000]. A broker provides a draft policy to his company. The draft policy with assured risks plays the role of Aggregate Event-based Claim.
- The Factoring system [Factoring 1996]. An assignment of an account receivable (i.e. a transfer of commercial invoices) sent by seller is Aggregate Event-based Claim. Particular invoices are its line items.
- Information System of the General Health Insurance [Health 1996]. In this case hospital claims and physician claims are Aggregate Multi-party Event-based Claims. Health care services, that have been provided, are their line items. The hospital/physician claim is associated with several parties: a patient, a doctor, an outpatients' department. As the number of parties is fixed the Party Role object is not required.

Related Patterns

Recurring Claim is a pattern of the same category; both are claims supplementing Basic Obligations. While Recurring Claim is triggered by a recurring time event, Event-based Claim is triggered by an explicit request of a party.

Aggregate Event-based Claim is quite often combined with Aggregate Partial Fulfillment. Combination with Summary Obligation is less common.

Partial Fulfillment

Context

An obligation is not always paid in full or it is paid gradually.

Example(s)

In health insurance a pharmacy sends an invoice to an insurance institution to honor drugs it has given to patients as patients do not pay for prescribed drugs. The invoice is supplemented with prescriptions provided in an electronic form. Each prescription contains the amount the pharmacy is claiming. The Ministry of Health (periodically) passes a regulation containing the list of maximum drug prices. The insurance institution verifies the prescriptions provided: whether a patient is insured and a doctor has a valid contract, if a drug is licensed and its price does not exceed the amount in the Ministry price list, etc. The insurance institution honors only prescriptions that have been verified. It can decrease a claimed drug price (and pay a prescription gradually, e.g. after a reclaim).

Problem

How do you represent partially honored obligations?

Forces

- The obligation may not be honored in full.

- Obligations may be fulfilled gradually.
- It is more understandable and space effective to have the fulfillment together with its basic obligation in a single object.
- Partial fulfillments can differ in the cash amount and other attributes.
- It must be clear what, when and how much has been paid.

Solution

Create the Partial Fulfillment object. Each instance of Partial Fulfillment refers to its Basic Obligation/Recurring Claim/Event-based Claim.

Variants

- Simple Partial Fulfillment. Here the Simple Fulfillment is decoupled from Simple Basic Obligation/Simple Recurring Claim/Simple Event-based Claim.
- Aggregate Partial Fulfillment utilizing Obligation Line Items of Aggregate Basic Obligation (Similarly, Aggregate Recurring Claim/Aggregate Event-based Claim). This variant is used when Aggregate Basic Obligation can be partially honored or gradually paid, but Obligation Line Item cannot be paid gradually.
- Aggregate Partial Fulfillment with Fulfillment Line Items. This variant is used when Obligation Line Item can be paid gradually. This is also true for Aggregate Recurring Claim/Aggregate Event-based Claim.

Example(s) Resolved

After the invoice is verified the insurance institution creates a settlement document.

If any prescription cannot be paid gradually the electronic settlement document contains the original prescriptions that are honored. Each such prescription contains the actual honored amount.

If prescriptions can be paid gradually the settlement document contains its line items. Each settlement line item refers to its prescription.

Resulting Context

Partial Fulfillments are decoupled from Basic Obligation/Simple Recurring Claim/Simple Event-based Claim that make clear what and how much is to be/has been paid.

Separation of Partial Fulfillments from Basic Obligation/Simple Recurring Claim/Simple Event-based Claim is worthy of added complexity only when it is common that Basic Obligation/Simple Recurring Claim/Simple Event-based Claim are not paid in full. Otherwise, it is sufficient to extend the original Basic Obligation/Simple Recurring Claim/Simple Event-based Claim (and their line items) with actually honored attributes.

Known Uses

- In the Koop insurance systems [Koop 1999] (accepted) Claim is decoupled from policyholder's Claim Report. Here loss notification is an elaboration of the Event-based Claim pattern while the insurance claim is an elaboration of the Partial Fulfillment pattern.¹¹ Several insurance Claims may be associated with one Claim Report.
- In the Broker system for selling insurance products [Broker 2000], [Sesera 2003] brokers are paid based on acquired units. Units are of a different type depending on the stage of policy and the broker role. In this way Claim Line Item (i.e. assured risk) can be associated with several units. The Settlement object plays the role of Aggregate Partial Fulfillment. Fulfillment Line Items are associated with one type of unit.
- Information System of the General Health Insurance [Health 1996]. In this example, Aggregate Claim of a health care provider is carefully validated. The settlement object representing Partial Fulfillment directly aggregates Claim Line Items (health care services).

Related Patterns

The Partial Fulfillment pattern is often neglected and superseded by Aggregate Payment Methods. However, in several cases we have found it useful when combined with Aggregate Event-based Claim.

Summary Obligation

Also Known As

Portfolio

Context

There are several obligations of the same Party. There is a requirement to simplify financial transactions between the parties and to reduce bank fees.

Example(s)

A person in need has several approved applications for state social benefits, e.g. a housing benefit, a child benefit and a disability benefit. The government agency pays these benefits monthly, all benefits for a person in the same calendar day. If every benefit is paid separately the institution needs to pay a bank fee (or a postal fee) for every benefit transaction, i.e. the bank fee while paying the housing benefit, the bank fee for the child benefit and the bank fee for the disability benefit.

¹¹ How domain names can be confusing☺.

Problem

How do you put together distinct obligations of the same Party?

Forces

- Obligations are of distinct types that are difficult to combine.
- When obligations are handled together the other party might be more satisfied as this party need to pay only one summary bill.
- When obligations are handled together bank fees are reduced.

Solution

Create Summary Obligation that aggregates particular obligations of the same Party.

Example(s) Resolved

A summary benefit is created each month that aggregates all the benefits of this person. This summary benefit is used for generating a payment order/postal order.

Resulting Context

Summary Obligations reduce bank fees and make the other party more satisfied.

On the other hand, Summary Obligations yield complexity. Furthermore, they obscure primary obligations to some extent. That is why in addition to Summary Obligation, particular obligations should be listed so that the other Party can check it for correctness.

Known Uses

- In the ISOP system [ISOP 1999] a person can have several valid applications; each for a distinct type of benefit. There is a summary benefit object summarizing the particular benefits for a given period.
- The insurance system of Wiener Städtische¹² uses summary premium subscriptions so that a client can pay several of his policies en bloc.

Related Patterns

Fowler's Portfolio concept is also used for putting together distinct contracts of the same Party. His motivation is different, however. It is to manage market risks.

Summary Obligation can be naturally combined with Basic Obligation and Recurring Claim.

In more complex systems it is superseded by Party's Account.

¹² Wiener Städtische is the parent company of Kooperativa.

Party's Account

Also Known As

Account

Context

Obligations between two parties are mutual: the party has both receivables and liabilities. Such mutual obligations should be combinable at any time. The simple concept of Summary Obligation is not enough because its aggregation of obligation is static and it is suited to unilateral obligations.

Example(s)

Company *A* buys products from company *B*. In addition, company *A* also sells other products to company *B*. Therefore company *A* may have both invoices sent to company *B* and invoices received from the company *B* at the same time.

Problem

How do you settle mutual obligations between two parties?

Forces

- Obligations are of distinct types that are difficult to treat uniformly.
- When obligations with the same party are combined it is clear who owes who.
- When obligations are combined bank fees are reduced.
- The simple concept of Summary Obligation is not enough as obligations can be accumulated from previous periods.

Solution

Create Party's Account that contains both credit obligations and debit obligations. Record credit obligations on the credit side of Party's Account and debit obligations on the debit side of Party's Account. Party's Account balance shows the outstanding claim or debt.

The structure of the pattern is shown in Figure 1-6.

In the diagram the Obligation concept represents Basic Obligation, Summary Obligation, Recurring Claim, Event-based Claim and Partial Fulfillment.

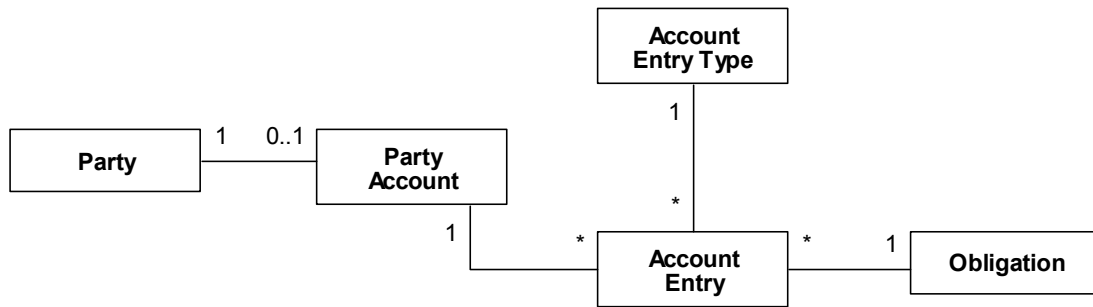


Figure 1-6 *The Party's Account pattern*

Variants

- Multi-party Account

Example(s) Resolved

Party's Account is defined for each company. On the credit side of this account, invoices that have been sent are recorded while on the debit side received invoices are recorded. Monthly, depending on the account balance the final invoice is sent or expected.

Resulting Context

Party's Account enables one to combine mutual obligations between two Parties. Furthermore, these obligations can be of various types. The account balance can be easily computed and at any time it is clear who owes who. It simplifies computing interest as well.

However, Party's Account is less efficient as it introduces another level of indirection. In addition, computing account balance can be time consuming. To improve this computed totals to defined time points (e.g. end of a month) are stored.

Known Uses

Information System of the State Social Support [ISOP 1999]. A person drawing state benefits is assigned a personal account. On the credit side of the account there are calculated benefits and repayments. On the debit side there are debts and benefits paid.

Related Patterns

The Party's Account pattern is similar to the Account analysis pattern of Fernandez [Fernandez 2002]. In both patterns accounts are associated with parties and address monetary issues. Our pattern, however, is the part of the pattern language that further elaborates objects of Account Entries.

Party's Account is the special case of a general account in the Accounting pattern (see below) in which Account can also be associated with concepts other than Parties. Such a more general Account corresponds to Fowler's Account [Fowler 1997].

Party's Account is a special case of the most general concept of AnyAccount [Fayad+ 2003] that deals also with accounts that are not constrained to monetary accounts. Unlike in the general AnyAccount pattern, monetary accounts cannot exist without Account Entries.¹³ In this way Party's Account is the specialization of a combination of AnyAccount and AnyEntry.

Party's Account can be combined with Basic Obligation, Recurring Claim, Event-based Claim and Partial Fulfillment.

The pattern supersedes Summary Obligation in more complex systems. Unlike Summary Obligation, Party's account aggregates obligations in a dynamic way and it contains both claims and debts of the counterparty.

Accounting

Context

A mutual obligation addresses several Parties (including government and a bank) and/or is internally composed of partial obligations (e.g. a reserve, a disbursement, VAT etc.) Such obligations can be examined from various points of view. Furthermore, there is a requirement for calculated amounts to be safely stored so that a discrepancy can be discovered easily.

Example(s)

A broker company employs brokers selling insurance products. A broker makes a policy draft, which is sent to an insurance company that examines the draft. The broker is given monthly provision that is the combined total of particular provisions; each particular provision depends on the product sold and the insured amount/insurance premium. The broker's provision is based on the value of sold regardless policy drafts have already been approved by the insurance company. However, as some policy drafts may not be approved, they may be changed or the broker may leave the company's employment, the broker is not paid in full but rather needs to fill a reserve fund until the prescribed maximum in this fund is reached. If any policy draft is later not approved or it is changed, the amount is subtracted from the following month's provision (if possible) or from the reserve fund. At any time for any broker the broker company needs to know their claims, debts, approved provisions, balance of the reserve fund, etc.

Problem

How do you represent various points of view of the same obligation and maintain integrity in such a multiple representation?

Forces

- Single objects are easy to understand.

¹³ Furthermore, the purpose of account as defined by Fowler is to record a history of changes to a value.

- For safety reason there is a need for double storing of values and to record the history of changes.
- Multiple objects may cause integrity problems.
- Defining points of view should be flexible.

Solution

Generalize Party's Account to the Account concept. Use various accounts to represent particular points of view and the accounting transactions to maintain integrity. Double-entring accounting transactions increase safety.

The structure of the pattern including the meta-level is shown in Figure 1-7.

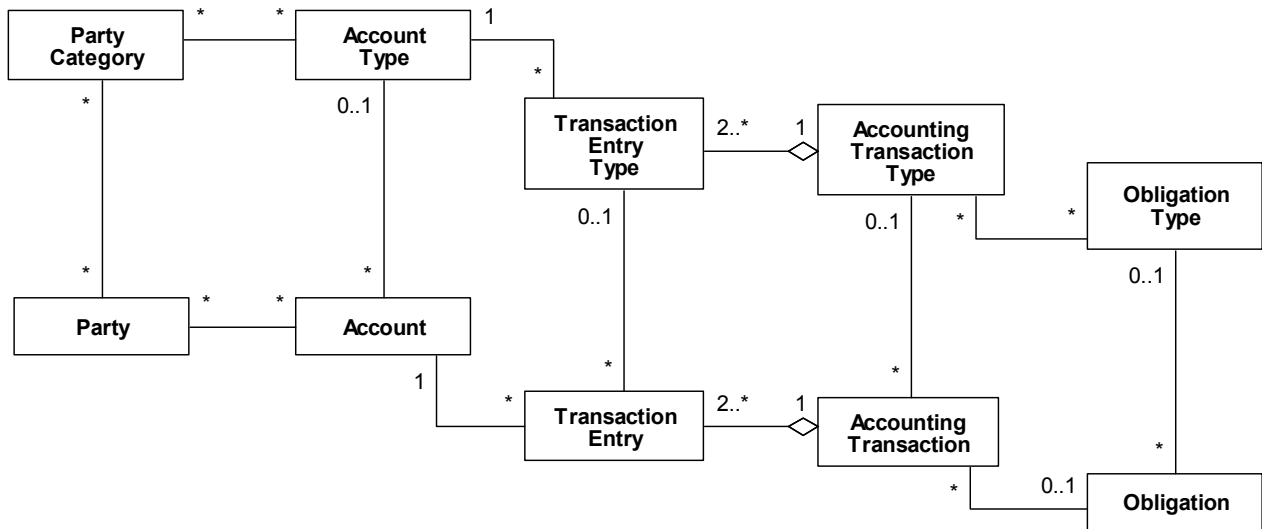


Figure 1-7 *The Accounting pattern*

Although the pattern can be used for general accounting, in the realm of the Obligation-Fulfillment patterns it is constrained to representing obligations.

Variants

- Accounting Blocks. Accounting transactions may be complex: some of them addressing more than a dozen accounts. In such a case it can be useful to define accounting blocks (usually accounting pairs) and then to compose transactions in a flexible way within these accounting blocks. The variant is shown in Figure 1-8.

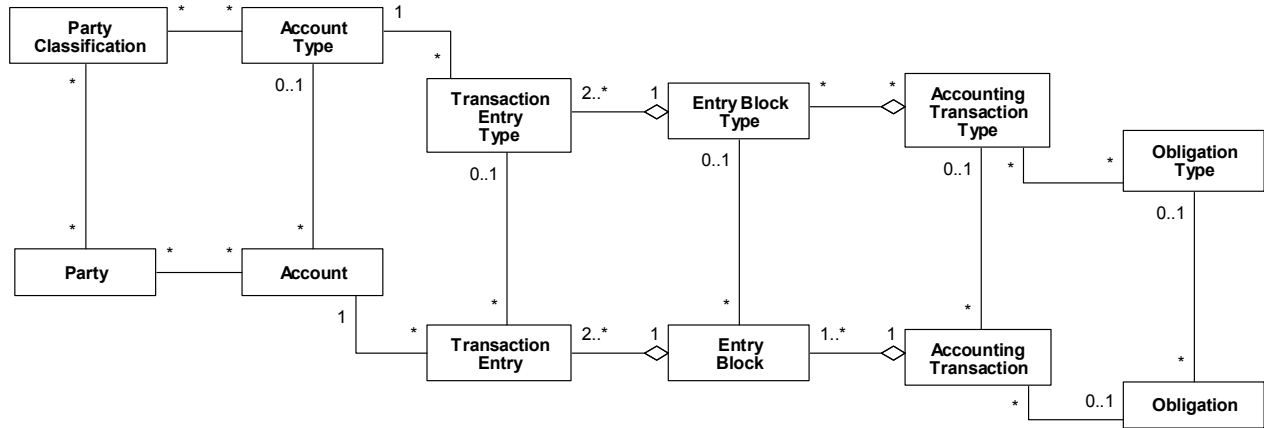


Figure 1-8 *The Accounting Blocks pattern*

Example(s) Resolved

For each broker a set of accounts is defined: provision earned, a reserve, debts, disbursements, VAT, etc. As the same set of accounts is defined for every broker and the same account numbers are used it is easy to find out the combined totals for all or a set of brokers within a specified time interval. Business transactions calculate requested amounts and then call particular accounting transactions for security reasons and to maintain integrity.

Resulting Context

Accounting transactions are the proven way to make transactions much safer. They help to maintain integrity among many Accounts.

On the contrary, an accounting transaction is a low-level technique for safety. In particular:

- It might not be easy to design accounts and accounting transactions in a proper way.
- It requires some effort to implement the accounting engine.
- Although a symptom of an error can be detected quickly, to find the cause can be difficult.

Known Uses

- In the Broker system for selling insurance products [Broker 2000], [Sesera 2003] the Accounting pattern is used. Partial obligations such as a reserve, a disbursement, a debt payment, etc. are represented in separate accounts.
- In the Factoring system [Factoring 1996] the Accounting pattern is also used. Here, some accounts are multi-party accounts addressing several parties, e.g. a factor together with a buyer, a factor together with a seller. As the accounting transactions were quite complex the second version of the system utilized the Accounting Blocks variant of the pattern.

Related Patterns

The Accounting pattern supersedes Party's Account in more complex systems. Here Party's Account is generalized to Account and Account Entries are put together using transactions.

This concept of Account corresponds to Fowler's Account [Fowler 1997]. When Accounting Transactions are added the pattern is similar to Fowler's Transaction pattern. More elaborated patterns for double-entry accounting can be found in [Hay 1996]. There are some slight differences between our pattern and patterns mentioned above. One difference is in the consistent usage of meta-level in our pattern. While it is quite unusual in general accounting, we have found it useful in more specific applications of financial systems. The other difference is our innovation of using Accounting Blocks in the case of complex transactions.

Accounting can be further augmented by Posting Rules as proposed by [Fowler 1997].

Payment Methods

Context

Obligations are paid using various methods of payments.

Example(s)

State social benefits can be paid using bank transfers or postal orders or exceptionally they may be paid in cash.

Problem

How do you handle various methods of payments in a uniform way?

Forces

- There are various methods of payment: bank account transfer, cheque, money order, credit card, cash, etc.
- There are some commonalities among the methods of payments.
- Payments should be associated with obligations.

Solution

Generalize commonalities among the various methods to the Payment and Payment Destination concepts. Associate Payment with its obligation. If Payment can be associated with several obligations, decompose Payment to Payment Entries. Assign Payment Entry to obligation.

The structure of the Simple Payment Methods version of pattern is shown in Figure 1-9¹⁴. The structure of the Aggregate Payment Methods version is sketched out in Figure 1-10.

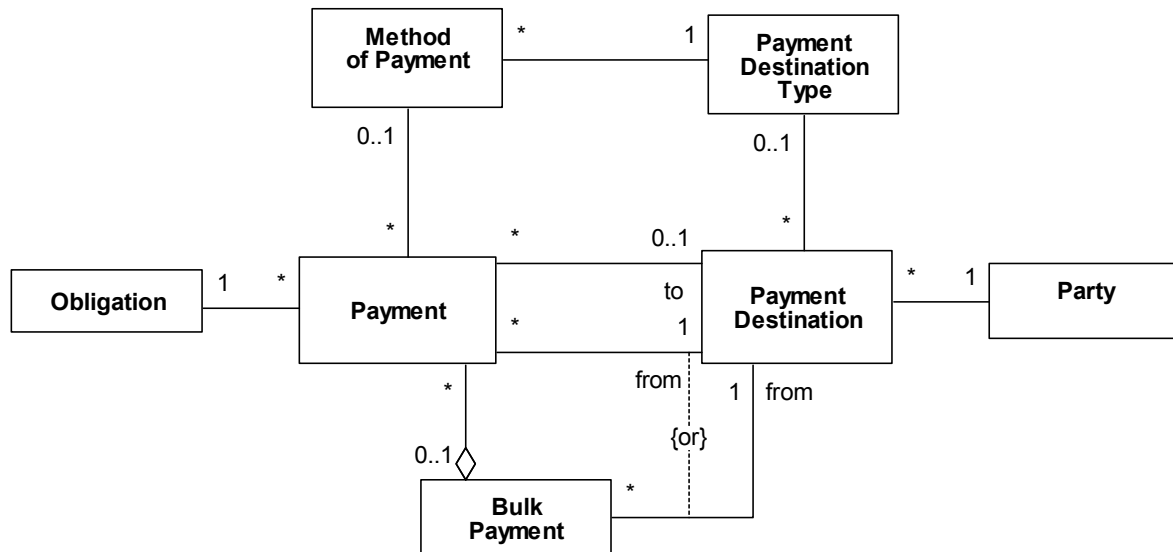


Figure 1-9 *The Simple Payment Methods pattern*



Figure 1-10 *The Aggregate Payment Methods subpattern*

The concepts in Figure 1-9 are specialized further using subtyping (power types). In this way required subtypes of Payments representing particular Methods of Payments are defined. Similarly, subtypes of Payment Destination such as Bank Account, Address, Cash Office, etc. are created.

Variants

- Simple Payment Methods
- Aggregate Payment Methods

Example(s) Resolved

Each person receiving benefits is assigned Payment Destination that is subtyped with the destination type this person has specified in his application¹⁵. Accordingly, an instance of particular subtype of Payment is created when the benefit is being paid.

¹⁴ Bulk Payment represents a payment from one account (and specifying this account only once) to many other accounts.

Resulting Context

The Payment Methods pattern puts together various methods of payments that simplifies computing.

The Aggregate Payment Methods variant of the pattern enables payment of several obligations en bloc while keeping associations to particular obligations.

The Aggregate Payment Methods variant imposes certain liabilities also. The algorithm concerning how Payment is decomposed to Payment entries might be difficult and/or domain specific.

In general, allowing several methods of payment increases the complexity of any system. This is not a drawback of the pattern, however. The pattern just makes the best of reality.

Known Uses

- Almost all systems mentioned in this paper use Aggregate Payment Methods in the form of decomposing Payment to Payment Entries.
- The ISOP system has used the pattern in its pure form in which there is a superclass for Payment Destination and a superclass or a type concept for Methods of Payments.

Related Patterns

While Basic Obligation is the foundation of other patterns, Payment Methods is the roof of these patterns. It can be combined with all other patterns.

Many times Aggregate Payment Methods is a substitution of Partial Fulfillment.

Acknowledgement

I would like to thank my shepherd Andy Longshaw for his many comments and suggestions that have helped me to clarify these patterns and describe them in a more readable form. I am also thankful to members of the writers' workshop at PLoP'2000, namely Ed Fernandez, Kyle Brown, Joseph Yoder and Kent Beck, for their valuable suggestions for improvements of the original small pattern and, especially, for encouraging me to grow the pattern into a pattern language. I thank also to Fergus White for proofreading this paper.

References

- [Bäumer+ 2000] Bäumer, D., D. Riehle, W. Siberski, M. Wulf. Role Object. PLoPD4, 2000.
- [Booch+ 1998] Booch, G., I. Jacobson, and J. Rumbaugh. Unified Modeling Language User Guide. Addison-Wesley, 1998.
- [Broker 2000] The Broker system for selling insurance products. Project documentation. Softec, 2000 (in Slovak).

¹⁵ The person can amend this destination during the validity of his application.

- [CDPK 2003] The CDPK card management system of the VUB bank. Project documentation. Softec, 2003 (in Slovak).
- [Coad 1997] Coad, P. Object Models: Strategies, Patterns and Applications. Yourdon Press, 1997.
- [EPP 2001] The EPP Information System for Collecting Employment Insurance for the National Labor Office. Project documentation, Softec 2001(in Slovak).
- [Factoring 1996] The Factoring system. Project documentation. Softec, 1996 (in Slovak).
- [Fayad+ 2003] Fayad, M. E. and H. Hamza. The AnyAccount Pattern. PLoP 2003.
- [Fernandez 2002] Fernandez, E., Y. and Y. Liu. The Account Analysis Pattern. EuroPLoP 2002.
- [Fowler 1996] Fowler, M. Recurring events, PLoP'96.
- [Fowler 1997] Fowler, M. Analysis Patterns: Reusable Object Models, Reading, MA: Addison-Wesley, 1997.
- [Gamma+ 1995] Gamma, E., R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Reading, MA: Addison-Wesley, 1995.
- [Hay 1996] Hay, D. Data Model Patterns: Conventions of Thought. New-York: Dorset House, 1996.
- [Health 1996] Information System of the General Health Insurance. Project documentation. Softec, 1996 (in Slovak).
- [ISOP 1999] Information System of the State Social Support. Project documentation. Phare # 951801, 1999.
- [Keller 1998] Keller, W. Some Patterns for Insurance Systems. PLoP'98. Also at: <http://ourworld.compuserve.com/homepages/WofgangWKeller/>
- [Koop 1999] The Koop software system for the Kooperativa insurance company. Project documentation, Softec 1999 (in Slovak).
- [Sesera 2000a] Sesera, L. A Recurring Fulfillment Analysis Pattern. Pattern Languages of Programs Conference, 2000.
<http://jerry.cs.uiuc.edu/~plop/plop2k/proceedings/proceedings.html>
- [Sesera 2000b] Sesera, L. Analysis Patterns. (Invited talk.) In: SOFSEM'2000. Lecture Notes in Computer Science series, Vol. 1963, Springer Verlag, 2000.
- [Sesera+ 2001] Sesera, L., A. Micovsky and J. Cerven, J. Data Modeling in Examples. Grada, 2001 (in Czech).
- [Sesera 2004] Sesera, L. Hierarchical Patterns: A Way to Organize (Analysis) Patterns. CITSA 2004. Orlando, FL, 2004.