Patterns of Business Software Applications In Model Driven Architecture

- Pavel Hruby
- Microsoft Business Solutions
- www.microsoft.com/BusinessSolutions

- www.phruby.com
- phruby@acm.org

Agenda

- Structural Patterns (13.30 – 14.50)
- Behavioral Patterns (14.50 – 16.05)
  - Coffee break (15.00 – 15.30)
- Model-Driven Architecture for Business Systems (16.05 – 17.00)
About me

- Pavel Hruby
  - (pronounced "ruby"; but call me Pavel, please)
- Work at Navision Software
  - Headquarters in Copenhagen, Denmark
  - Products: Navision (Attain), Axapta, C5
- Work at Microsoft Business Solutions
  - Microsoft has acquired Navision in July 2002
  - Products: + Great Plains, Solomon + more
  - [www.microsoft.com/BusinessSolutions](http://www.microsoft.com/BusinessSolutions)
- E-mail: phruby@acm.org

Description of a Pattern

- Name
  - Proxy
- Context
  - Client is depending on services from another component. Direct access is technically possible, but may not be the best approach.
- Forces
  - Client needs to access services from another component
  - We do not want do hard-code a component's physical location to its clients
  - Direct access may be inefficient or insecure
- Solution
  - Let client communicate with a representative (surrogate), rather than the component itself.
Pattern Definition

- Each pattern describes a problem that occurs over and over again in our environment,
- and then describes the core of the solution to that problem
- in such a way that you can use this solution a million times over without ever doing it the same way twice.

- Christopher Alexander

Light on Two Sides of Every Room

- When people have a choice, they will always gravitate to those rooms which have natural light coming in from at least two sides.
- In designing a home, it's best to locate each room so that it has outdoor space on at least two sides, with ample windows to capture natural light in every room from more than one direction.

Patterns of Business Software Applications

- Business Relationships
- REA (Resources Events Agents)
- Contract
- Roles
- Business Transaction
- Account
- Identification
- Classification
- Due Date
- …
Part 1: Structural Patterns

Business Relationships
REA
Business Transaction
Contract
Roles
Business Component

Business Relationships: Business Problem
Business Relationships: Structure

- Unfortunately, reality is more complicated than this.

Business Relationships: Forces

- You want to model the rules that apply to all economic systems
- You want to model things that are shareable and reusable across various application domains
- You want to build an extendable system, which can be extended by new concepts without changes to the foundations of the system.
Example: Sales Order

Other examples: Service level agreement, invoice, phone call

Business Relationship: Structure
Business Relationships: Resulting Context

- Parties (agents, business partners) are "never" directly related to each other.
- The pattern does not specify any additional constraints and rules.
- The pattern defines the fundamental infrastructure of the enterprise model, which can be extended by other patterns.

Exercise: Examine this Model

Pattern Language for Business Systems

Business Transaction: Problem

Customer

Company

Record

Database

Term "transaction" has two meanings:
- Communication involving two or more parties
- A record of business, especially a business agreement or exchange

I use the term transaction to mean the record of business.
Business Transaction: Forces

- Record all relevant information about relationships between business partners

- An event that affects the business relationship might imply other events to occur.
  - For example, shipment of goods implies payment to occur.

- 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.

Business Transaction: Solution

![Diagram of business transaction solution]
Reconciliation (Apply): Example

6/1 2003 Purchase 20 USD

10/1 2003 Payment 25 USD

10/1 2003 Purchase 25 USD

20/1 2003 Payment 45 USD

25/1 2003 Purchase 50 USD

30/1 2003 Payment 5 USD

Business Transaction: Example

«party» Customer C
consumer

4 pcs $40

«resource» Product A
$10 per 1 piece

«business transaction» Invoice
initiator

$40

«party» Vendor V
supplier

«resource» Cash
terminator

«business transaction» Payment
reconciliation
Business Transaction: Resulting Context

- Special strategies for corrections of errors
- Business transaction (in the REA conceptual framework) does not have a navigable relationship to account.

More Information

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

- 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.
REA (Resources Events Agents): Problem

- Customer
- Product
- Order
- Company
- Exchange
- Cash

REA (Resources Events Agents): Problem

- Agent
- Event
- Agent
- Customer
- Company
- Order
- Product
- Cash
- Exchange
In Fact, there are Two Economic Events

Customer → Shipment → Company

Product

In Fact, there are Two Economic Events

Customer → Payment Receipt → Company

Payment Receipt

Cash
REA (Resources Events Agents): Problem

REA (Resources Events Agents): Example
**REA (Resources Events Agents): Axioms**

- At least one inflow event and one outflow event exist for each economic resource; conversely, inflow and outflow events must affect identifiable resources.

- All events effecting an outflow must be eventually paired in duality relationships with events effecting and inflow and vice-versa.

- Each exchange needs an instance of both the inside and outside parties.

---

**REA (Resources Events Agents): Structure**

![Diagram of REA (Resources Events Agents): Structure]

- Economic Agent
  - Inside participation: 1
  - Outside participation: 1

- Economic Resource
  - Stock flow: 1..*

- Economic Event
  - Duality: 1..*
### REA: Wholesale Example

- **Customer**
  - **Shipments** (outflow)
  - **Cash Receipts** (inflow)
- **Wholesaler**
  - **Purchases** (inflow)
  - **Payments** (outflow)
- **Vendor**
  - **Products**
  - **Cash**

### REA: Supply Chain Example

- **Procurement**
  - **Payment**
  - **Purchase**
- **Transportation**
  - **Logistical Operation**
  - **Delivery**
- **Human Resources**
  - **Payment**
  - **Labor Acquire**
- **Sales & Marketing**
  - **Sale**
  - **Cash Receipt**
- **Manufacturing**
  - **Material Issue**
  - **Manufacture Operation**
- **Finance**
  - **Cash Receipt**
  - **Cash Payment**

Source: ebXML BP Catalog
**REA: More Information**


- Jesper Kiehn, Model REAL World, VikingPLOP 2002, for details see [www.plop.dk/vikingplop](http://www.plop.dk/vikingplop)
  - This paper describes REA in the pattern form.
In Fact, there are Two Commitments

Commitment: Forces

- 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.

- 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.
**Commitment: Example**

- **Resource:** Product A
  - Price: $10 per 1 piece
  - Quantity: 4 pcs
  - Total: $40

- **Commitment:** Commitment to Ship
  - Fullfillment: 1 pc
  - Total: $40

- **Commitment:** Commitment to Receive Cash
  - Total: $40

**Contract as a Bundle of Commitments**

- **Resource:** Product A
  - Price: $10 per 1 piece
  - Quantity: 4 pcs
  - Total: $40

- **Commitment:** Commitment to Ship
  - Fullfillment: 1 pc
  - Total: $40

- **Commitment:** Commitment to Receive Cash
  - Total: $40
Commitment: Structure

Commitments: Axioms

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.
Commitments: Resulting Context

- Prognosis of future events, for example budgeting, is not covered by this pattern. Budgeting is addressed by classification / grouping pattern.

- There might be a hierarchy of contracts: order, service level agreement, government regulations … Higher level contracts specify policies for lower level contracts.

Claim: Business Problem
Claim: 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.

Claim: Forces

- Exchange of economic resources must be fair, that is, agreed by both parties.
- Business partners do not know the exact unbalanced value of the exchanges.
- 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.
Claim: Solution

- Claim: Examples
  - 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!
Roles: Business Problem

If nurse becomes a patient, wouldn't it be nice to reuse data already existing in the system?

Roles: Context

- The same physical entity participates in different types of business relationships
- Agents
  - Customer, employee, vendor
- Resources
  - Work in progress, raw material, finished goods
Roles: Forces

- The same physical entity participates in different types of business relationships.
- Entity 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.
- You want to capture the information that customers, vendors and employees can be, in the real world, the same physical entity.

Roles: Solution
Roles: Solution in UML

Roles: Resulting Context

- The roles pattern adds flexibility and extensibility to the model. Information can be shared, not duplicated.

- However, this is not always the desired behavior.
  - Security and legal issues
Roles: More Information

- Martin Fowler: Dealing with roles
  - Implementation considerations in Java.  

Business Component


- Peter Coad, Eric Lefebvre, Jeff De Luca: Java Modeling in Color with UML, Prentice Hall, 1999
David Hay’s Contract Pattern

Adapted from David C. Hay: Data Model Patterns: Conventions of Thought

Peter Coad’s Four Archetypes

Adapted from Coad et al.: Java Modeling in Color
Peter Coad’s Domain-Neutral Component

Comparing the Pattern Descriptions
Part 2: Behavioral Patterns

Identification
Account
Due Date
Calendar
Classification

Pattern Language for Business Systems
Identification: Business Problem

Name
CPR number

Author, Title
ISBN
DOC Number

Identification: Context

• There are often specific business rules for creating identifications.
Identification: Structure

- Business Entity
  - Event
  - Commitment
  - Contract
  - Resource
  - Party
  - Business Document

Identification Type
- auto number
- unique
- mandatory

Number Series
- last ID
- creation rule

Identification
- identification: string

Identification: Resulting Context

- Business Entities can also be identified by their Descriptions (another pattern). However, the purpose of description is different.

- Some users prefer to think about two patterns
  - Number Series (unique identifications)
  - Names (might be both unique or non unique).
Account: Business Problem (1)

- Purchase 20 pcs
  - 6 January

- Sale 2 pcs
  - 10 January

- Sale 5 pcs
  - 12 January

- Athena chair (product)

- Purchase 10 pcs
  - 20 January

Account: Business Problem (2)

- Sales 150 USD
  - 6 January

- Cash receipt 100 USD
  - 10 January

- Cash receipt 50 USD
  - 15 January

- Joe Smith (customer)

- ... Sales 50 USD
  - 20 January
Account: Context

- In some cases, keeping track of the individual entities is not possible or desirable.

- For instance, once a glass of water has been poured into a tub, it is no longer possible to distinguish between the water that was in the tub before the water was poured into it and the water that comes from the glass.

- The only thing that it is possible to keep track of is the total amount of water in the tub.

Account: Forces

- The business transaction pattern describes how to register and keep track of individual 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.

- You would like to know a rule that would automatically derive types of aggregations from the business transactions, events, and commitments available in the system.
Inflow: SUM(inflow) – SUM(outflow)

Number of Athena chairs on stock:

Outflow:
Account: Solution in the REA Framework

The accounts are separated from the transactions. Account number does not need to be specified at the commit (post, register) time.

However, traditional double-entry bookkeeping systems specify accounts when transaction is committed (posted).  
- legal issues
Account: Solution in Double-Entry System

Accounting Entry

Business Transaction

Account

value

value

sum of values of entries = 0

Cash Account

Accounting Entry

+ 100 USD

Business Transaction

+ 100 USD

Checking Account

Accounting Entry

- 100 USD

Account: More Information

- Martin Fowler: Accounting patterns
  - Double-entry bookkeeping patterns

- Paul Keefer wrote a MS thesis on Accounts
Due Date: Business Problem

Also known as Deadline and Milestone

Due Date: Context

- Certain actions have to be taken and things have to be done on or before certain date.

- Due dates are often coupled together, to specify two points in time, or time interval.
  - For instance, starting date and end date are due dates, where end date occurs, for example, five days after the starting date.
Due Date: 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.

Due Date: Example

Simple due dates

duration

start date    end date

time

Recurring due dates

start  end  start  end  start  end
time
Due Date: Solution

Due Date

Due Date Type
Recurrence Rule
Reminder Rule
«is a type of»

Due Date
Start Date
Duration
Has Expired
Recurrence
Range of recurrence
Reminder
«create»

Reminder

Due Date
Commitment,
Claim
Economic Event

Business Relationship

Due Date: Resulting Context

Calendar provides an aggregated view on due dates.
### Calendar: Business Problem

- Meeting 8.30-9.15
- Meeting 10.00-11.30
- Lunch 12.00-12.30
- Dinner 19.00-21.00

### Calendar: Solution

- **Calendar Type**
  - Work week
  - First day of week
  - First week of year
  - Holidays

- **Due Date**
  - Start date
  - Duration
Classification: Problem

- Also known as
  - Categorization
  - Grouping

- Customers
  - high-volume customers
  - small-yield customers

- Furniture
  - Office
  - Home

- Employee
  - Outstanding performance
  - Average performance
  - Under average performance

Typification vs. Grouping

Source: Guido Geerts, 2003
### Classification: Example

- Categories of web pages in Yahoo

### Classification: Forces

- You want to make a uniform model for, for example, event categories, agent categories, resource categories.
- It should be possible to categorize different entity types using the same classification hierarchy
  - For example, different resources share the same VAT (sales tax) categories.
- When an object changes its category, does not change its type.
  - For example, when customer changes from small-volume to high-yield, it still has attributes name, address, phone number…
- The entity typically does not change its category when its attributes change.
  For example, let’s suppose that the payment entity belongs to a certain priority category. If due date of the payment is over, the payment entity still belongs to the same priority category.
Classification: Solution

The Types can also be used to categorize objects.
- Use types if objects in different categories have different attributes and methods.
- Use classification if objects in different categories have the same attributes and methods.

The Lifecycle (State Machine) can also be used to categorize objects.
- The states can be viewed as categories.
- Use lifecycle if you need to specify rules for transitions from one category (state) to another.
- Use lifecycle if an event can change the object's category (state).
Classification: Exercise

- "Categories" of sales order: quotation, accepted order, shipped order, paid order.
  - Would you use classification (grouping), type, or lifecycle (state machine) patterns?

- The library has different types of books: computer books, history books, crimi books, etc.
  - Would you use classification (grouping), or type patterns?

Part 3: Implementation

Model Driven Architecture (MDA)
Business Patterns and MDA
Implementation Approaches
Model-Driven Architecture™

- An approach to using models in software development 1).

  - specify the kinds of models to be used
  - relationships of the models

- MDA Goals
  - specify a system independently of the platform
  - portability, interoperability, reusability


MDA Concepts (OMG approach)

- CIM – Computation Independent Model (informal)
- PIM – Platform Independent Model a “blueprint” UML model
- PSM – Platform Specific Model implementation model
- Platform CORBA, .NET
Model Transformation

- A process of converting one model to another model of the same system.

PIM – Platform Independent Model

Transformation

PSM – Platform Specific Model

Metamodel Transformation

- PIM metamodel is mapped to PSM metamodel
Pattern Application

- PIM patterns are mapped to PSM patterns

Model Merging
"Virtual Machine" Approach

- PIM is interpreted by a framework at runtime
- No model transformation

Developer-Centered Model (Traditional)
User-Centered Model (Vision and Goal)

Can it be Achieved?

- Can a single model be
  - intuitive for the users AND
  - precise enough to generate a software application?

- Yes, if the model is at the right level of abstraction.

- Common modeling language between the consultants, developers and users.
  - Modeling language and symbols are identical to the language and symbols of the user's daily work.
UML Model-Driven Architecture

Language Used

PIM → Metamodel

Terms like:
- Namespace
- Class
- Visibility
- Method
- Attribute
...

UML Metamodel

Business Model-Driven Architecture

Language Used

PIM → Metamodel

Terms like:
- Commitment
- Account
- Address
- Category
...

REA++ Metamodel

More info: Kasper Østerbye: http://www.it-c.dk/people/kasper/
Business Patterns as Components

Merging Structural and Behavioral Patterns
Merging Structural and Behavioral Patterns

Aspect-Oriented Programming (AOP)

- AOP captures cross-cutting concerns in a modular way.
  - objects capture primary division of labor
  - aspects capture cross-cutting concerns

- It is useful to think about
  - structural patterns as objects
  - behavioral patterns as aspects

Merging Business Components

Structural Components
REA ++

Behavioral Components
Identification
Classification
Due Date
…

Configured System

Merge can occur at
- design time
- compile time

Merging Business Components: Example

Structural Components
Economic Event
Payment

Behavioral Components
Payment Date
Discount Date
Payment Number
…

Configured System
-Economic Event-
-Payment-
-Due Date-
-Payment Date-
-Payment Number-
-Identification-
-Payment Number Type-

Start Date
Duration
…

Payment Date
Discount Date
Payment Number

User Interface Model Transformation

Business Logic (PIM) \(\rightarrow\) "REA" Language \(\rightarrow\) "REA" Metamodel

Transformation \(\rightarrow\) UI generator specification

User Interface (PSM) \(\rightarrow\) html, asp, … \(\rightarrow\) UI Metamodel

Data Model Transformation

Business Logic (PIM) \(\rightarrow\) "REA" Language \(\rightarrow\) "REA" Metamodel

Transformation \(\rightarrow\) DB generator specification

Data Model (PSM) \(\rightarrow\) SQL \(\rightarrow\) "SQL" Metamodel
Contact

Pavel Hruby

- phruby@acm.org
- www.phruby.com