UML Modeling与 Together For VS.NET

? ? Gordon Li
Borland大中华首席技术官

ULL NAME
TLE POSITION

Borland
Excellence Endures

Agenda

- ? ? ? ? ? Modeling
- Best Practices
- UML? ?
- UML Diagrams and Extensions
- Design pattern
- ? ? Borland Together For .NET? ? Modeling? ? ? ? ?

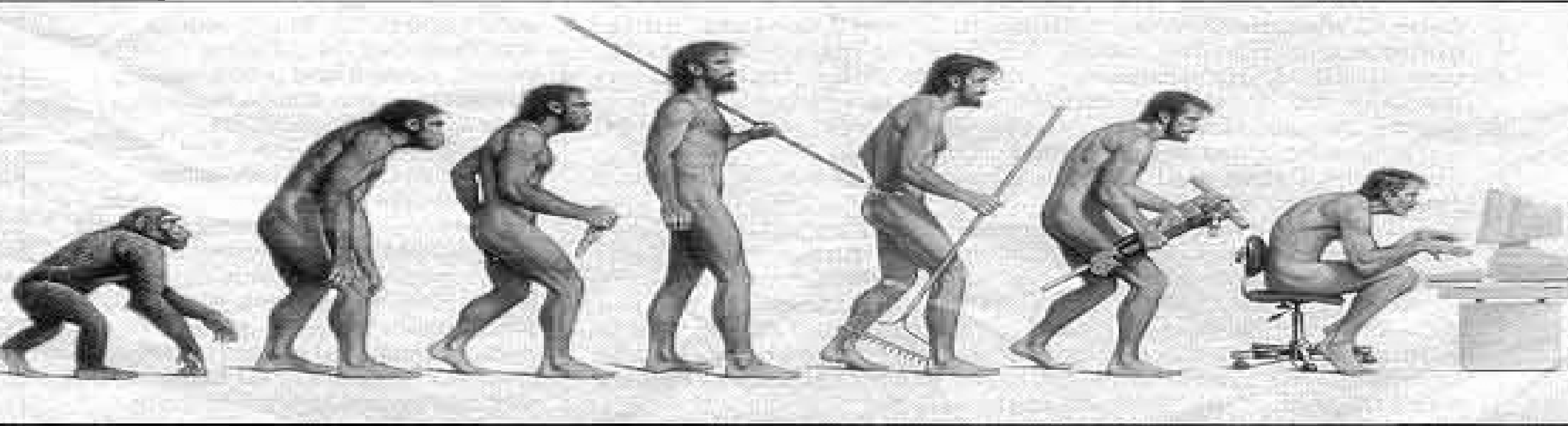
Borland®

为什么需要Modeling

其实我们一直在做Modeling!

为什么需要Modeling

其实我们一直在做Modeling!



Modeling Cycle

Modeling Process

Modeling Development

Modeling Code

Borland®

为什么需要Modeling

- 问一个简单的问题, 您了解你的产品/项目吗?
  - Microsoft Duwamish范例

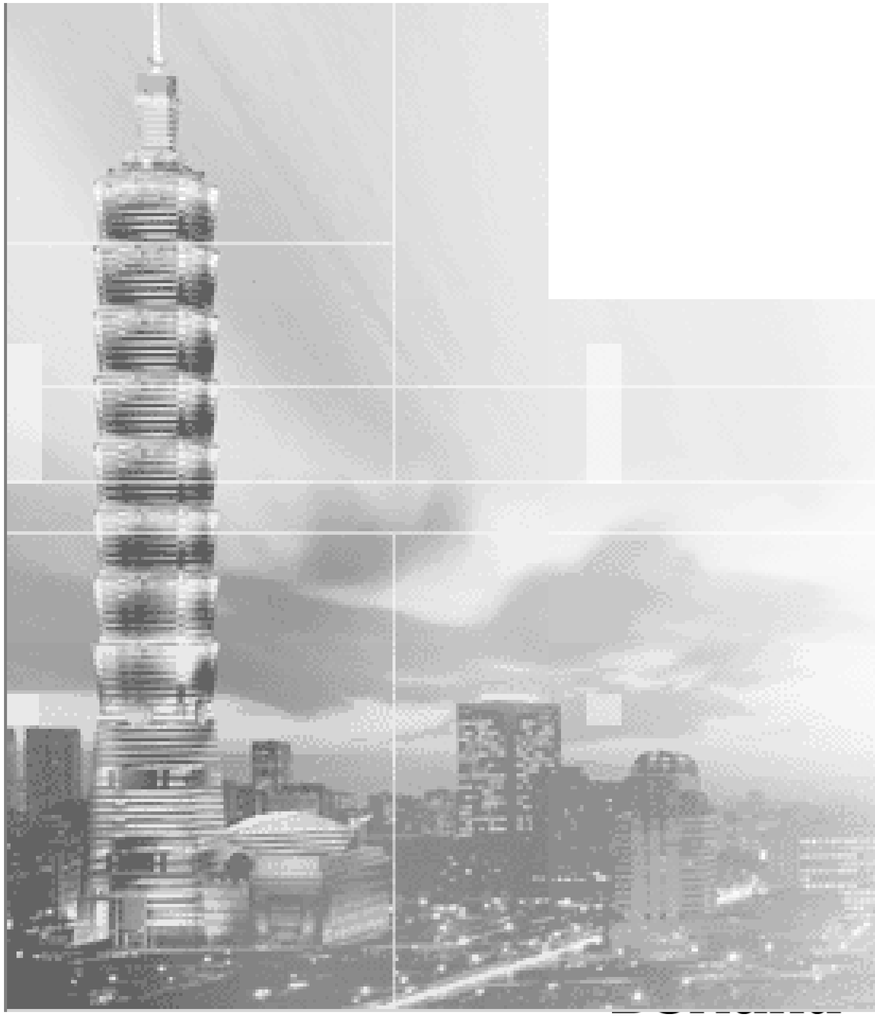
为什么需要Modeling

Simple vs. Complex

One man vs. Team work

Non-Critical vs. Critical

VS.



Model & Blueprint

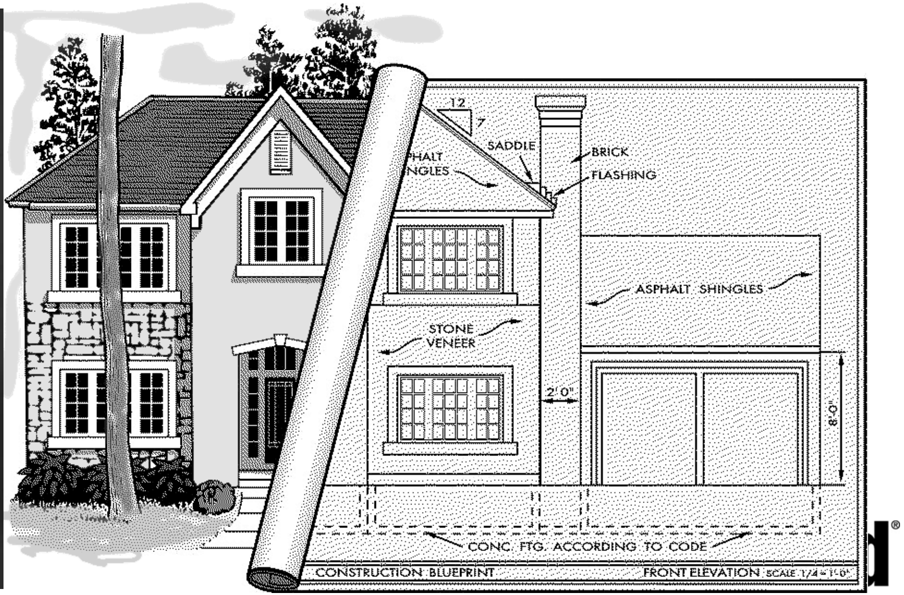
模型与蓝图是所有参与者的沟通根据

Business Model

Requirement Model

Analysis Model –Platform Independent Model

Design Model –Blueprint for implementation



From IBM Rational - RUP

Best Practices

Develop Iteratively
Manage Requirements
Use Component Architectures
Model Visually (UML)
Continuously Verify Quality
Manage Change

Component architectures is evolved into Service Oriented architecture.

Borland®

Once Upon A Time …

® ™ &© ? ? ? ? ? ? ? ? ? ?

If you only knew the power of the dark side.

® ™ & © ? ? ? ? ? ? ? ? ? ?

Preparation and Recommendation

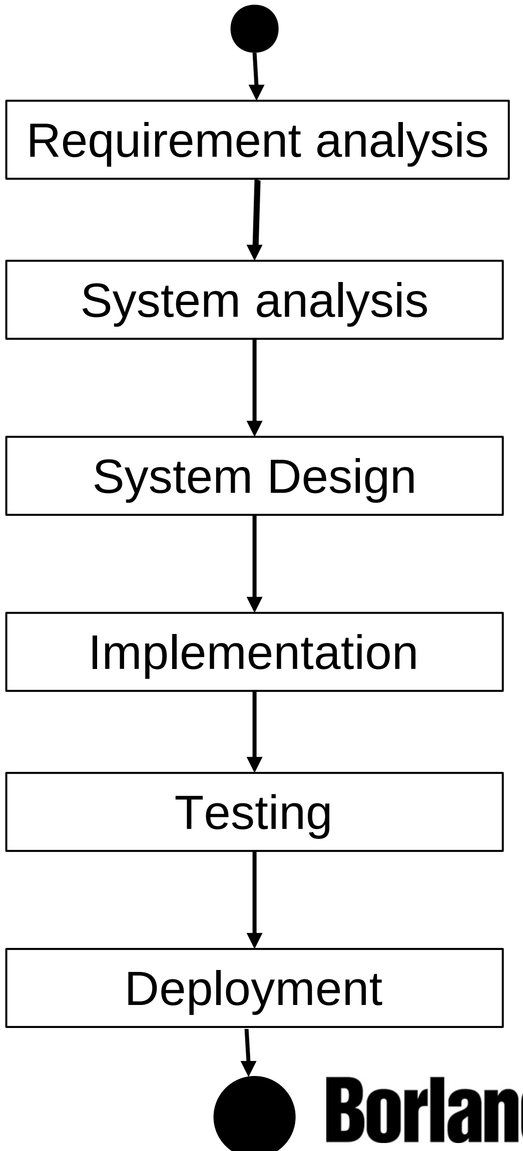
- Mindset
  - Not religion, but rigorous
  - Incremental adoption
- Training
  - Learning & Training
  - Apprenticeship
  - Mentors
- Tools are required
- Agile Process
- Start Small

® ™ &© ? ? ? ? ? ? ? ? ? ?

The Process

- Business Model
- Requirement Model
- User Experience Model
    - look-Feel & Interaction
- Analysis Model
- Design Model
    - Architecture design
    - Data Model
- Implementation Model
- Test Model
- Deployment Model

From IBM Rational - RUP



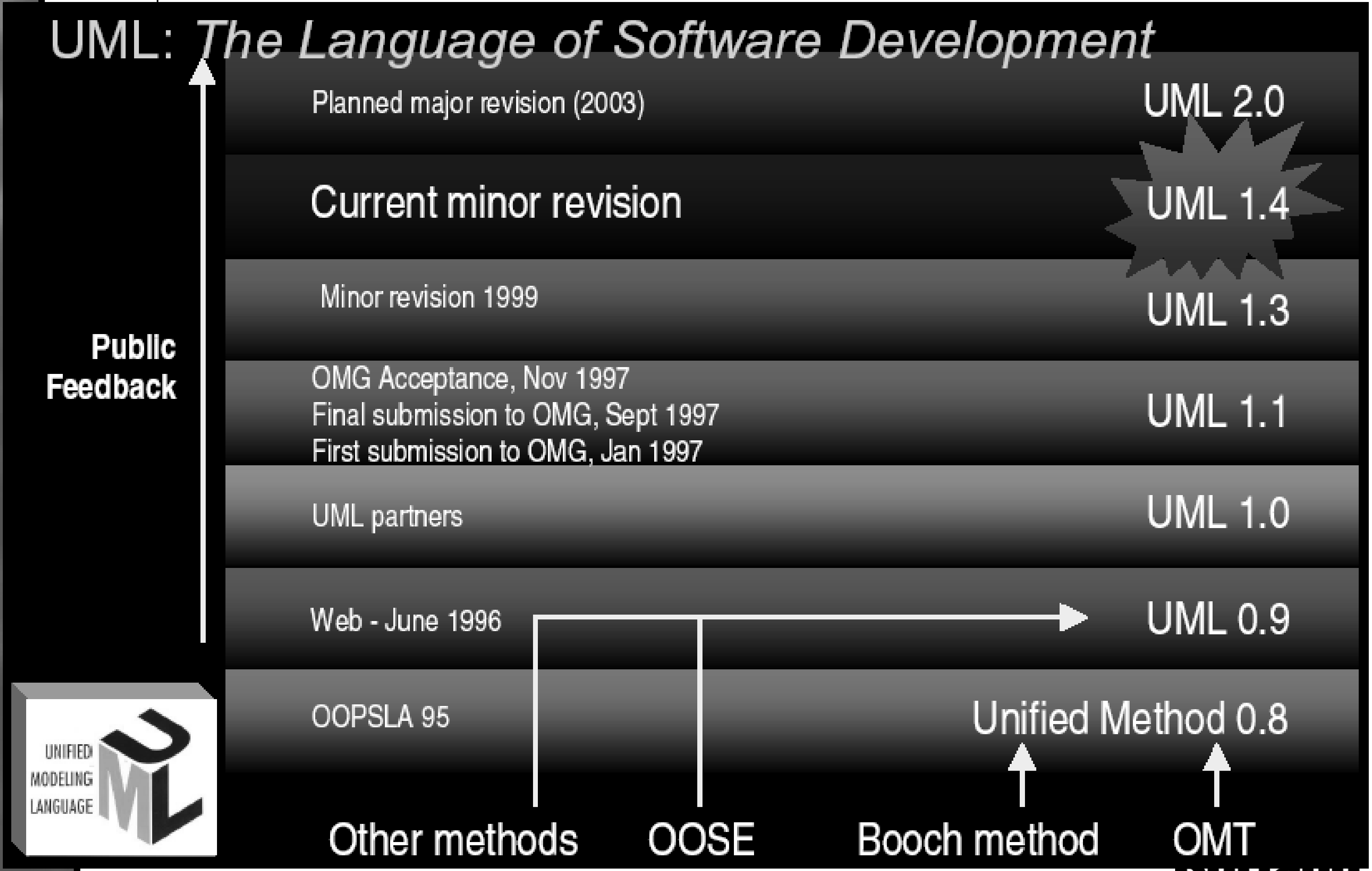
Borland®

UML: Unified Modeling Language

The UML is the standard language for visualizing specifying, constructing, and documenting the artifacts of a software-intensive system

From IBM Rational - RUP

UML? ? ? ?


UML: *The Language of Software Development*
Planned major revision (2003)
UML 2.0
Current minor revision
UML 1.4
Minor revision 1999
UML 1.3
Public Feedback
OMG Acceptance, Nov 1997
Final submission to OMG, Sept 1997
First submission to OMG, Jan 1997
UML 1.1
UML partners
UML 1.0
Web - June 1996
UML 0.9
OOPSLA 95
Unified Method 0.8
UNIFIED MODELING LANGUAGE
Other methods
OOSE
Booch method
OMT

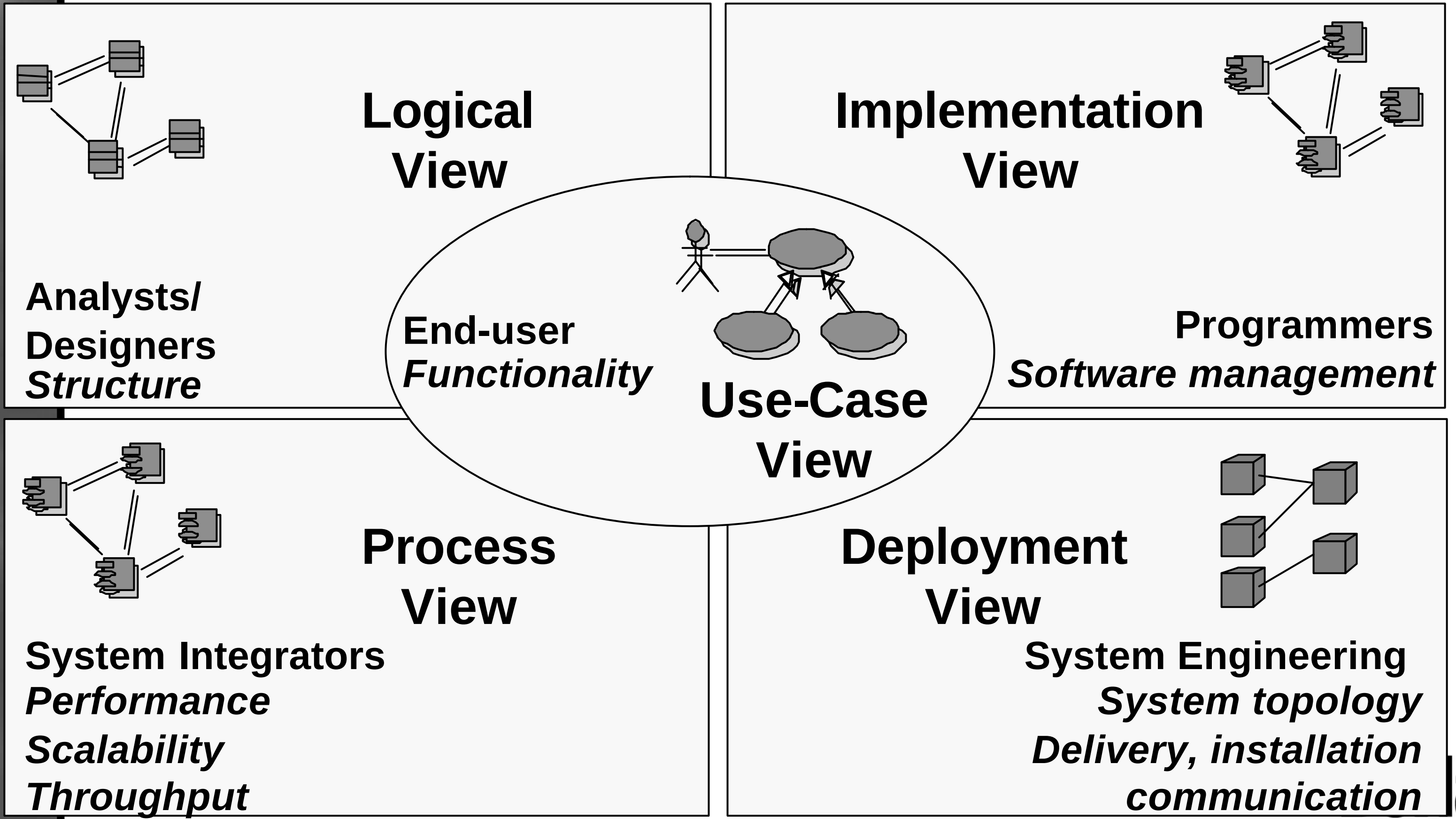
The Value of the UML

- Is a standard
- Supports the entire software development lifecycle
- Supports diverse applications areas
- Is based on experience and needs of the user community
- Supported by many tools

Model, View, Diagram

Model contains views for different purposes.

View contains multiple diagrams.



UML 12 Diagrams

Behavior :

- Use Case
- Activity
- Sequence
- Collaboration
- State Chart

Structural:

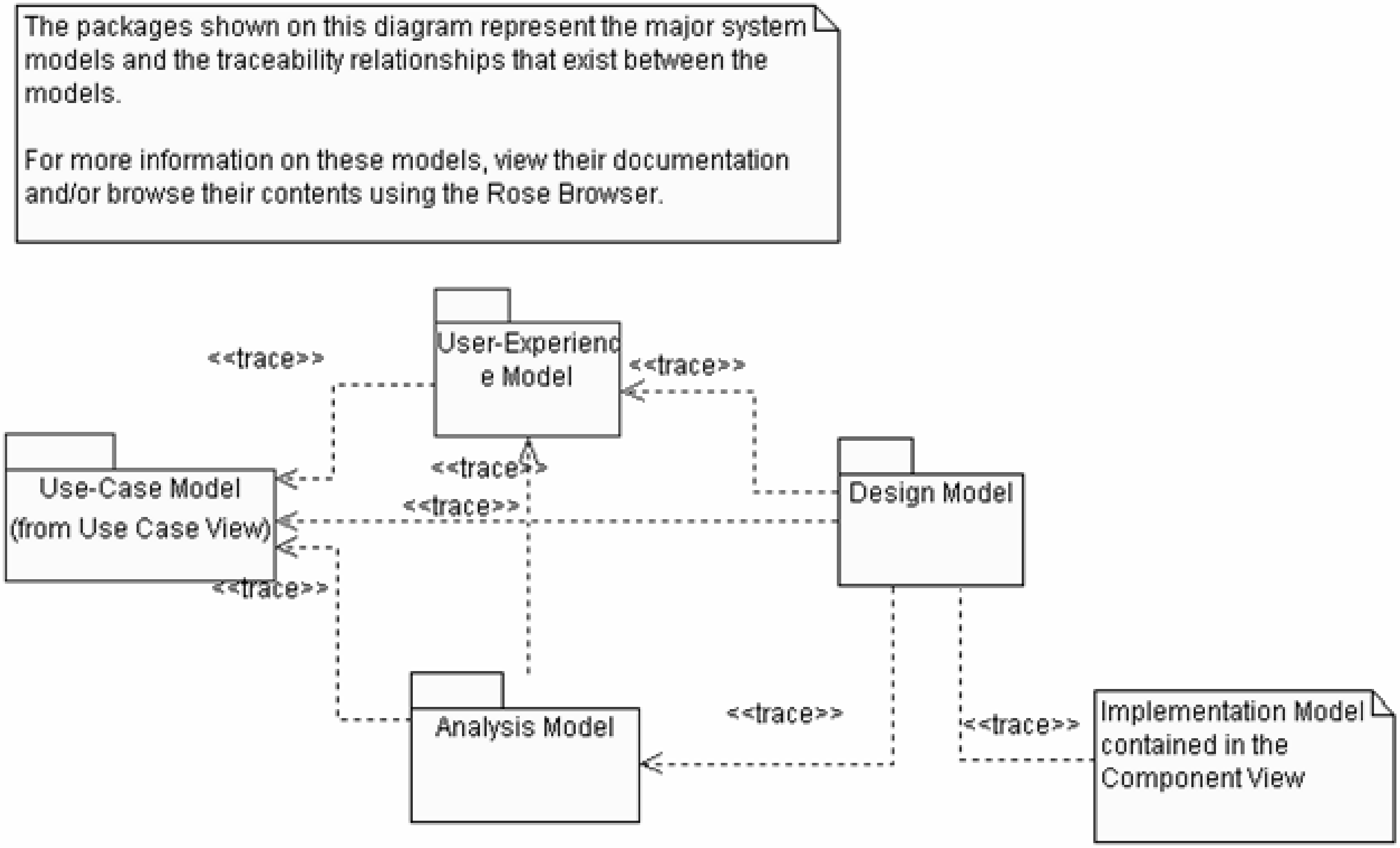
Class
Component
Deployment
Object

Model Management:
Packages (class diagram contains packages)
Subsystems (class diagram contains subsystems)
Models (class diagram contains models)

Borland®

Model Diagram

使用Model Diagram**描述所要执行的**Modeling**工作**


The packages shown on this diagram represent the major system models and the traceability relationships that exist between the models.

For more information on these models, view their documentation and/or browse their contents using the Rose Browser.

需求模型： Use-Case Model

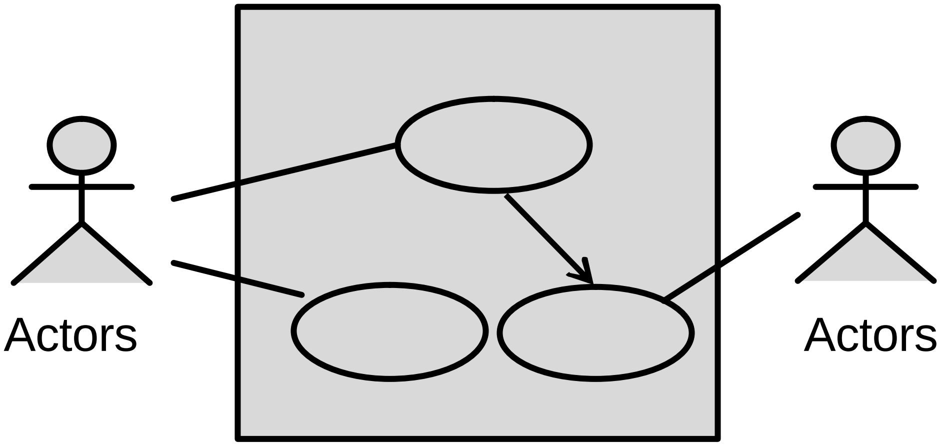
Use case model描述系统功能，是与客户沟通的工具

Use Cases: 系统行为或功能

Actors: 系统的各种使用者

Use Cases Diagrams

Use-Case Reports or Use-Case Specifications



Use-Case Diagram

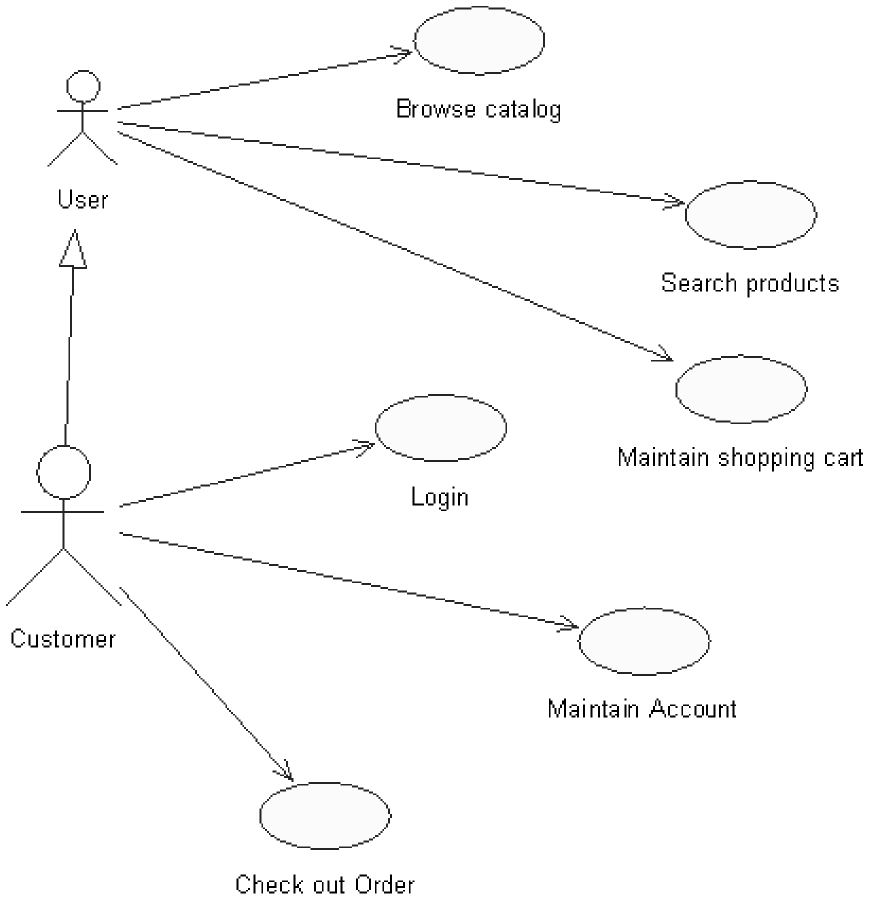
网络购物系统的需求模型

Use cases:

- Browse catalog
- Search products
- Maintain shopping cart
- Logon
- Maintain account
- Check out order

Actors

- User
- Customer



Borland®

Use case report

This is requirement model version.

1. Name: Browse Product Catalog
2. Flow of Events

   2.1 Basic flow: **显示所有的次类别及促销产品**

      1.**使用者开启首页**

      2.**系统显示所有的次类别**, **与当时的促销产品**

      3.**使用者选取次类别**

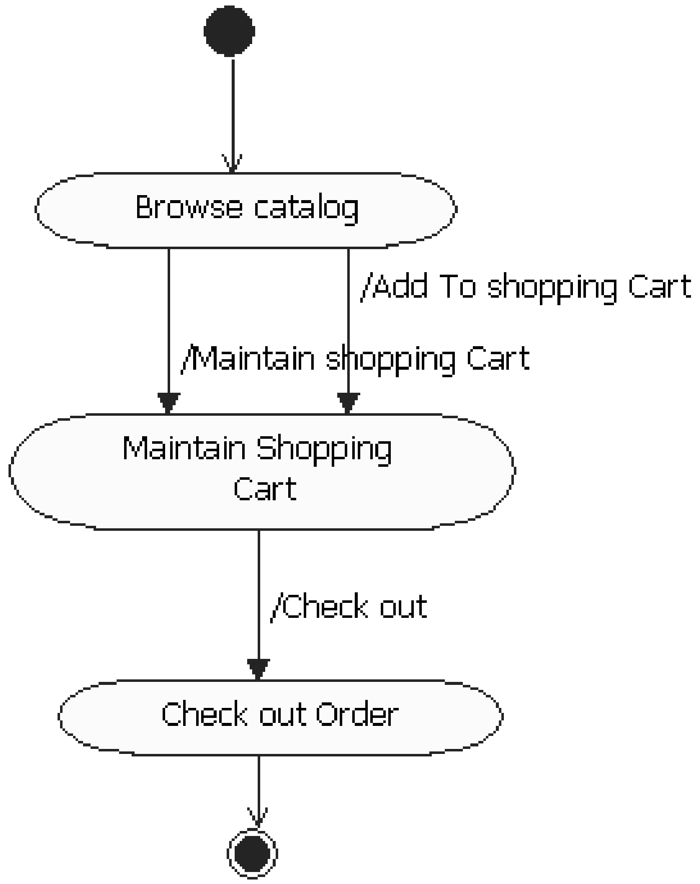
      4.**系统显示该类别的全部产品简介**

      5.**使用者选择产品**

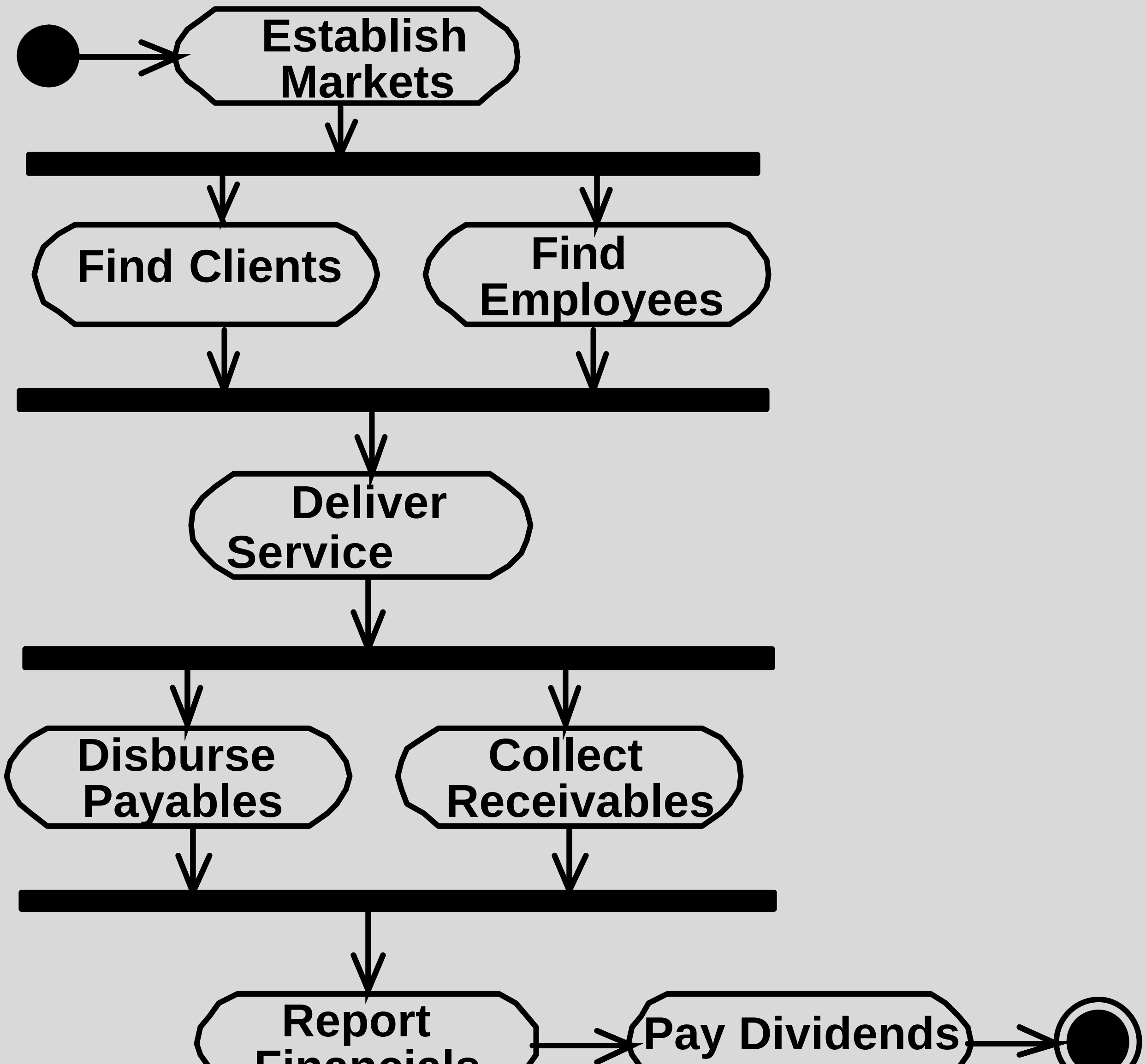
      6.**系统显示该产品的信息**

Activity diagram

描述Business process**或**use case**的操作流程**



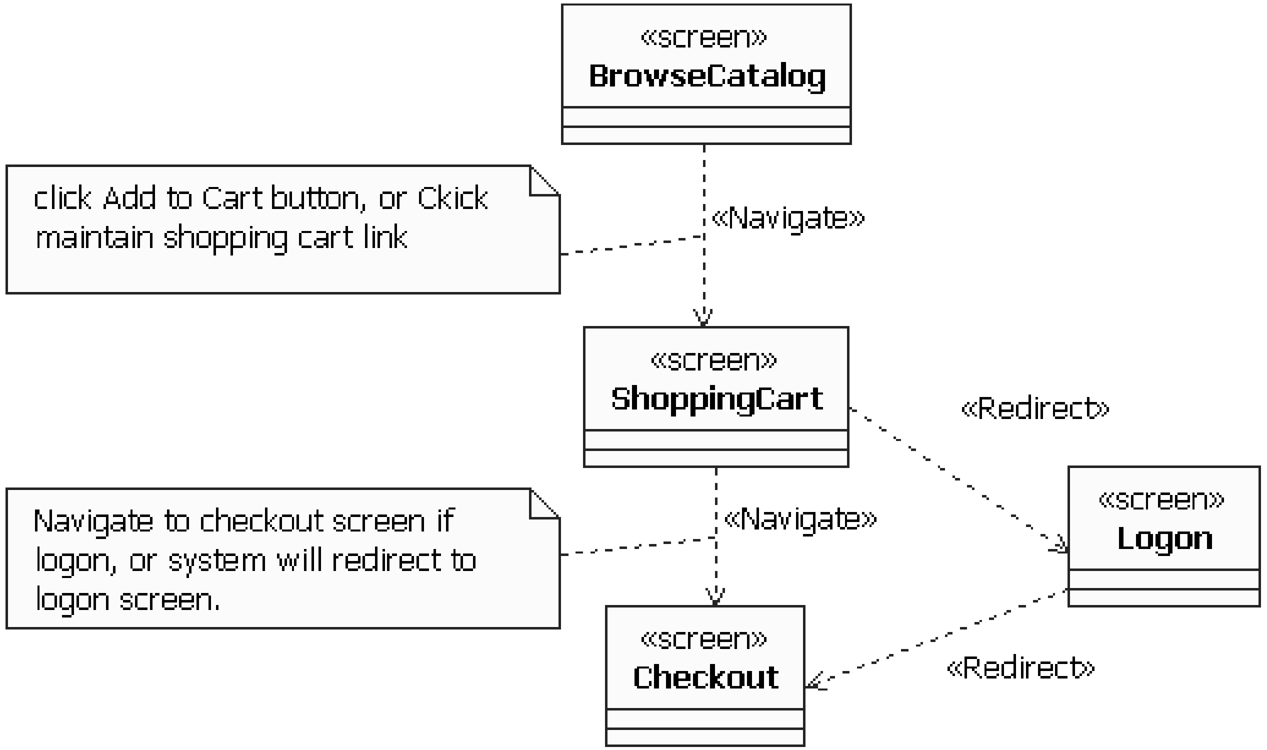
Business Modeling



User Experience Model

UI Prototype

Navigate flow: use UI class with screen stereotype


«screen»
BrowseCatalog
click Add to Cart button, or Ckick maintain shopping cart link
«Navigate»
«screen»
ShoppingCart
«Redirect»
«screen»
Logon
Navigate to checkout screen if logon, or system will redirect to logon screen.
«Navigate»
«screen»
Checkout
«Redirect»

More Detailed Use Case

This is User Experience Model version, Optional.

1. Name: Browse Product Catalog

2. Flow of Events

2.1 Basic flow: ? ? ? ? ? ? ? ? ? ? ? ? ?

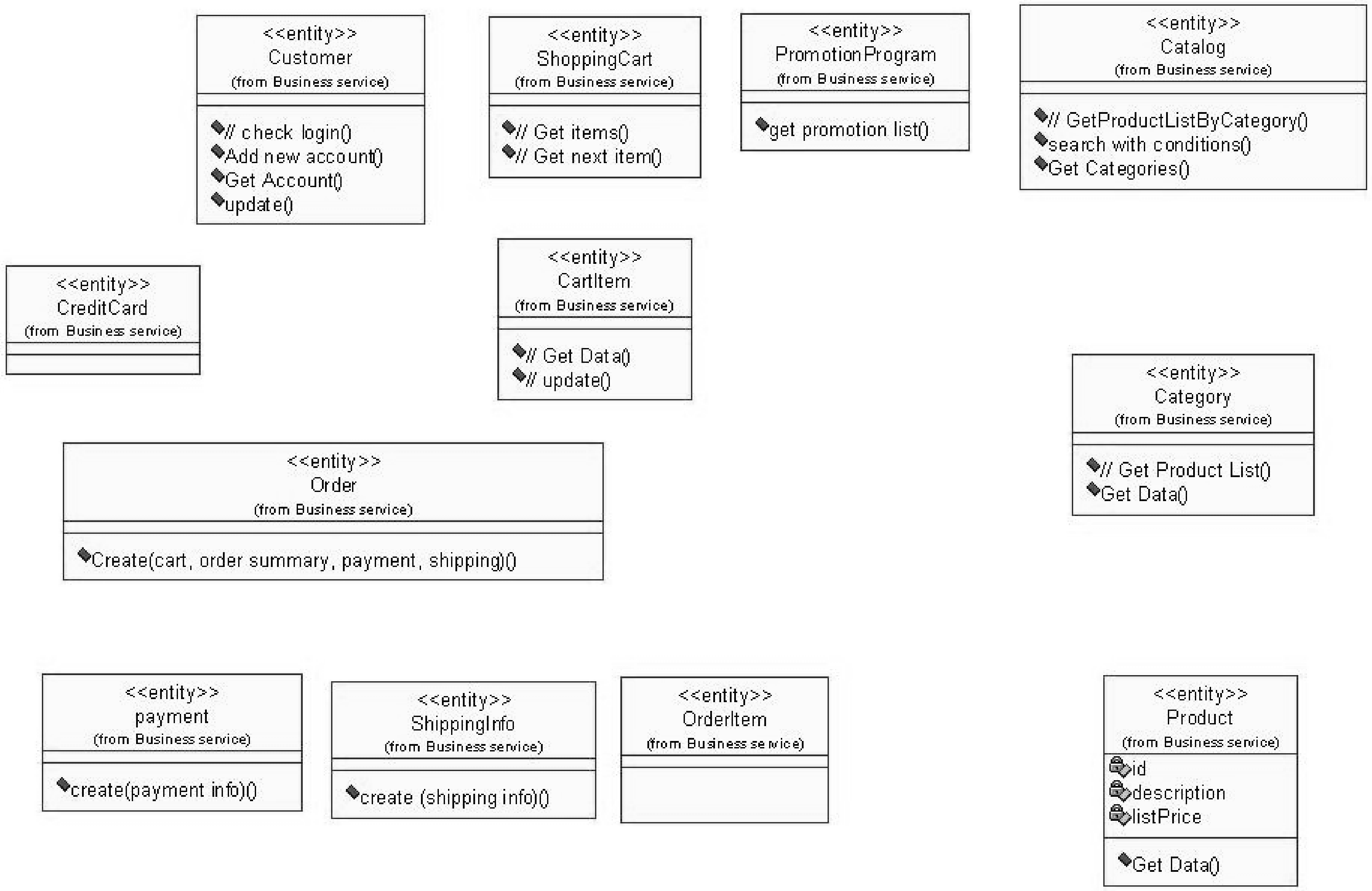
1. ? ? ? ? ? ? ?
2. ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
3. ? ? ? ? ? ? ? ? ? ? pick of today? ? ? ? ? ? ?
4. ? ? ? ? ? ? ? ?
5. ?
6. ? ? ? ? ? ? ?
7. ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? , ? ? ? AddToCart?

Analysis model

- Platform Independent Model
- **找出完成**use cases **所需要的类别**
  - Analysis classes
- **规划**Classes**的**relationship
- **指派**use-case behaviors**给适当的类别**
  - ？ ？ Operations
- **找出每个**class**的**attributes

Analysis Classes

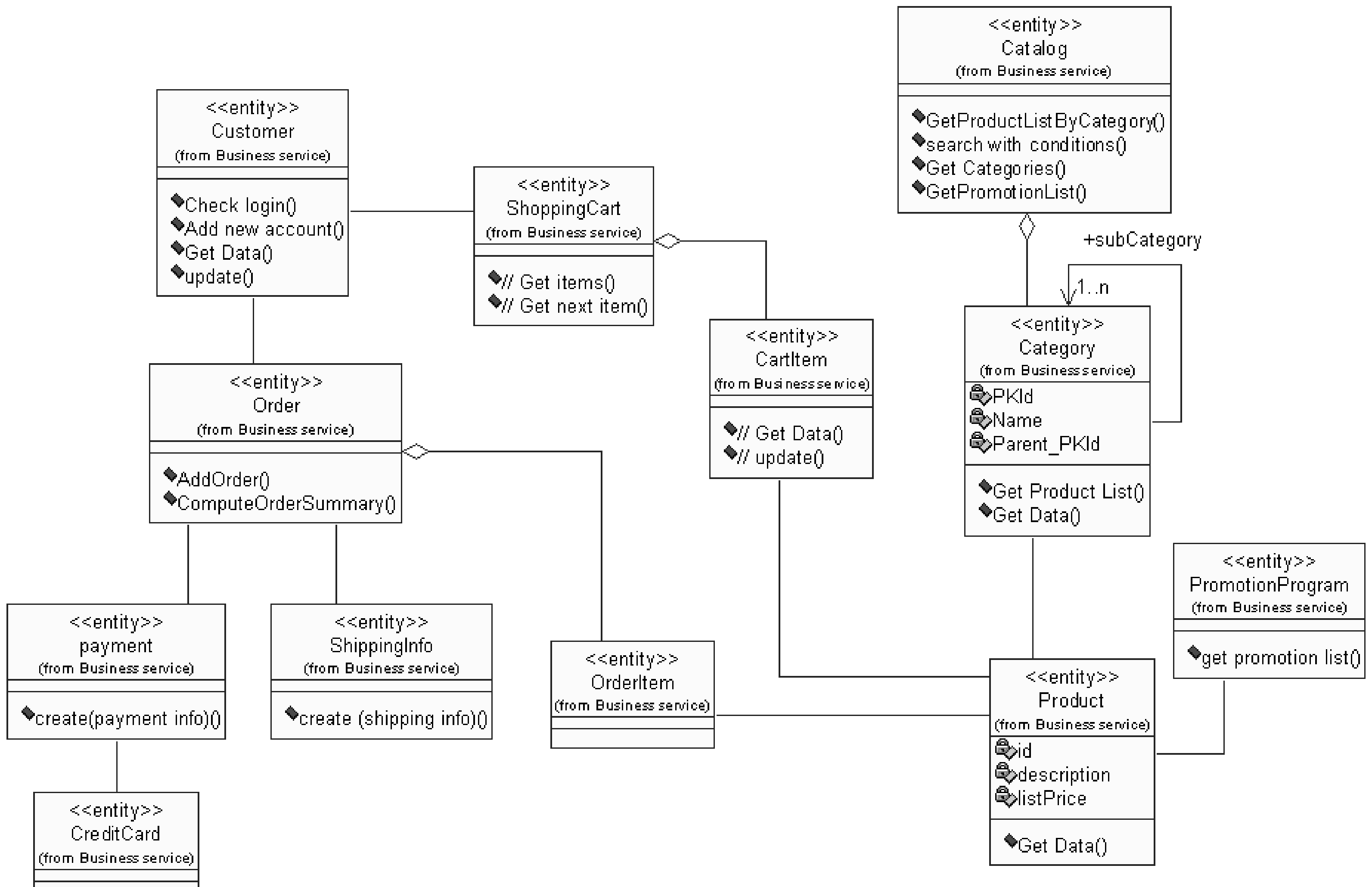
Class name, operations, attributes



<<entity>> Customer (from Business service)
// check login()
Add new account()
Get Account()
update()
<<entity>> ShoppingCart (from Business service)
// Get items()
// Get next item()
<<entity>> PromotionProgram (from Business service)
get promotion list()
<<entity>> Catalog (from Business service)
// GetProductListByCategory()
search with conditions()
Get Categories()
<<entity>> CreditCard (from Business service)
<<entity>> CartItem (from Business service)
// Get Data()
// update()
<<entity>> Category (from Business service)
// Get Product List()
Get Data()
<<entity>> Order (from Business service)
Create(cart, order summary, payment, shipping)()
<<entity>> payment (from Business service)
create(payment info)()
<<entity>> ShippingInfo (from Business service)
create (shipping info)()
<<entity>> OrderItem (from Business service)
<<entity>> Product (from Business service)
id
description
listPrice
Get Data()

Class diagram

Find Relationships, be care of being too complex



Architecture design

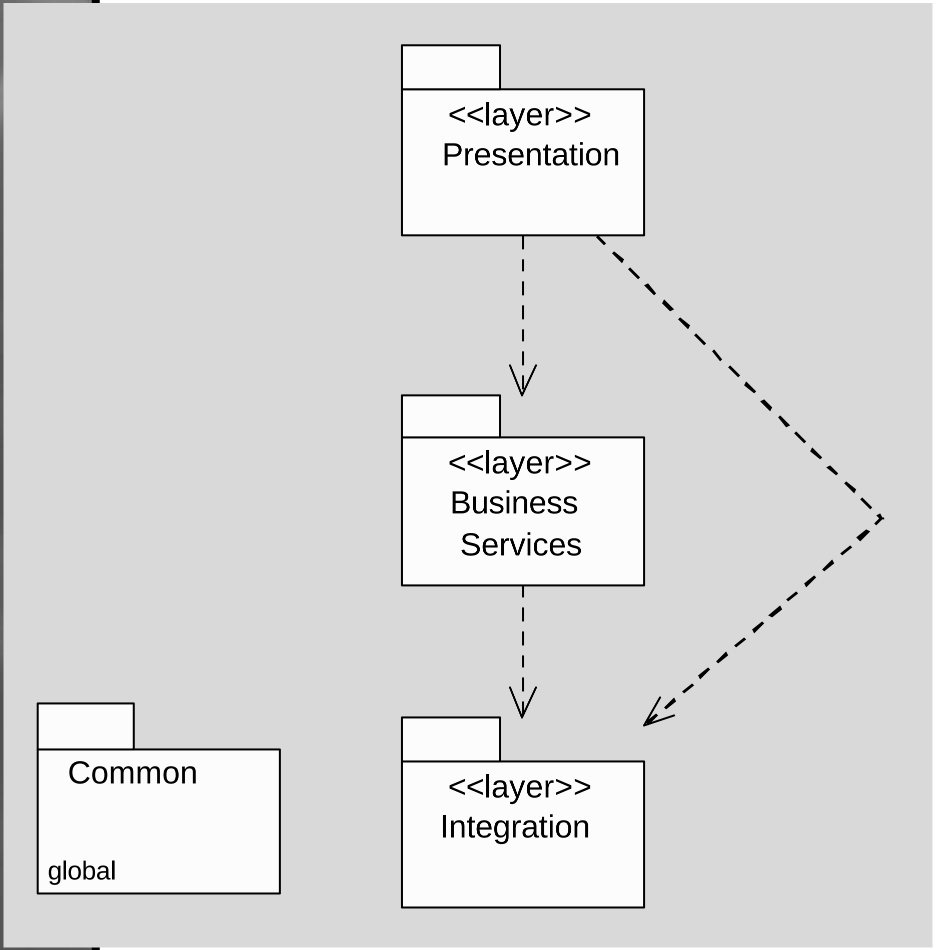
Architecture design处理较高层次的设计组件，包括package, Subsystem, Interface。

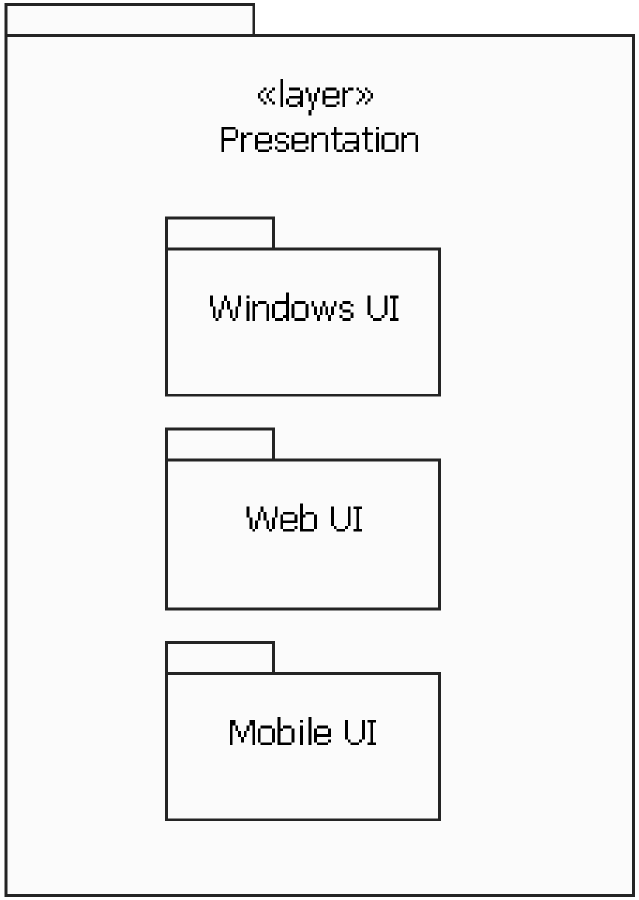
使用Package组织设计组件。

Subsystem包含多个classes, 透过Interfaces展现功能。

Package diagram

Package diagram is a class diagram which contains packages.

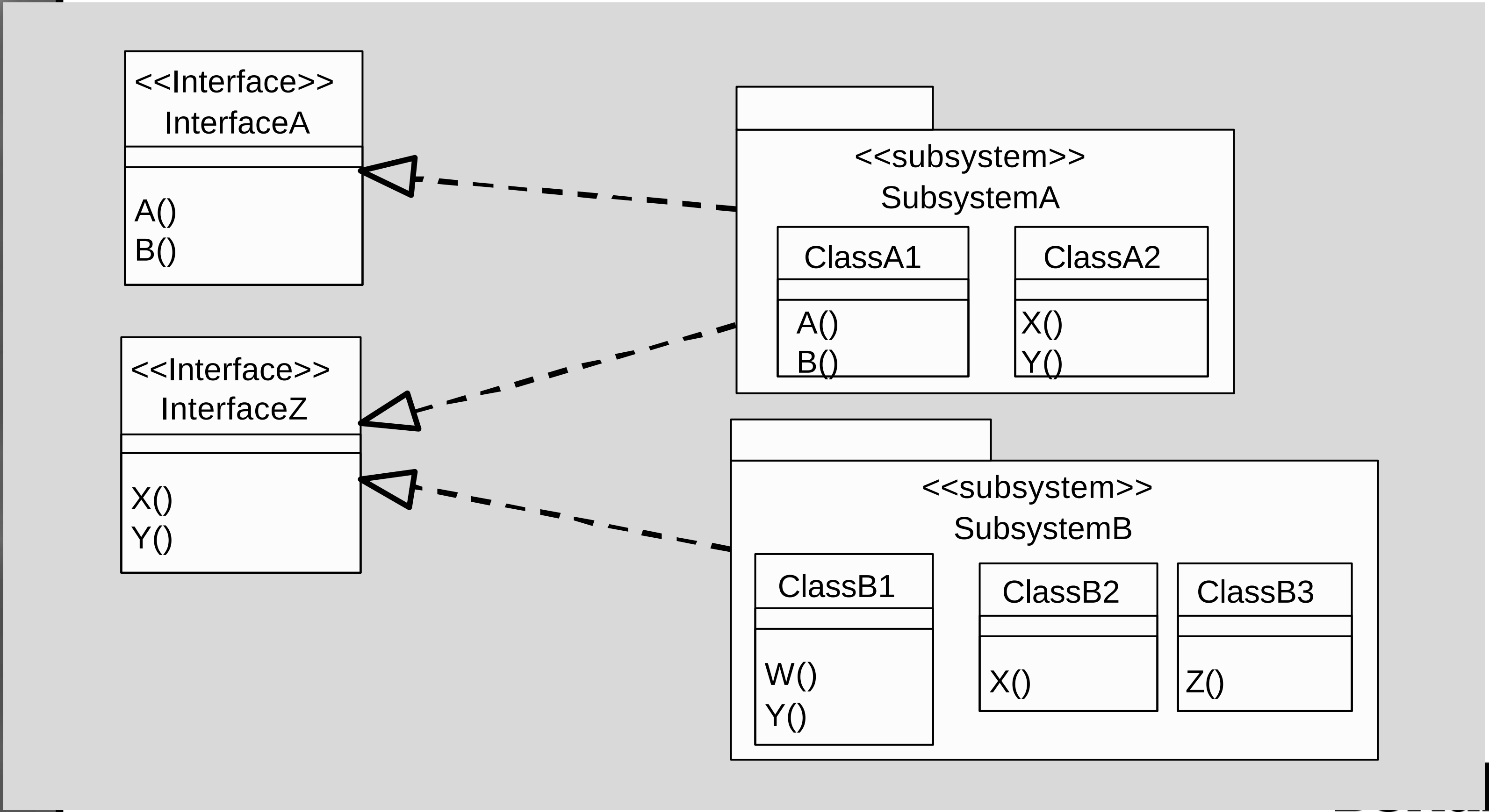




Subsystem diagram

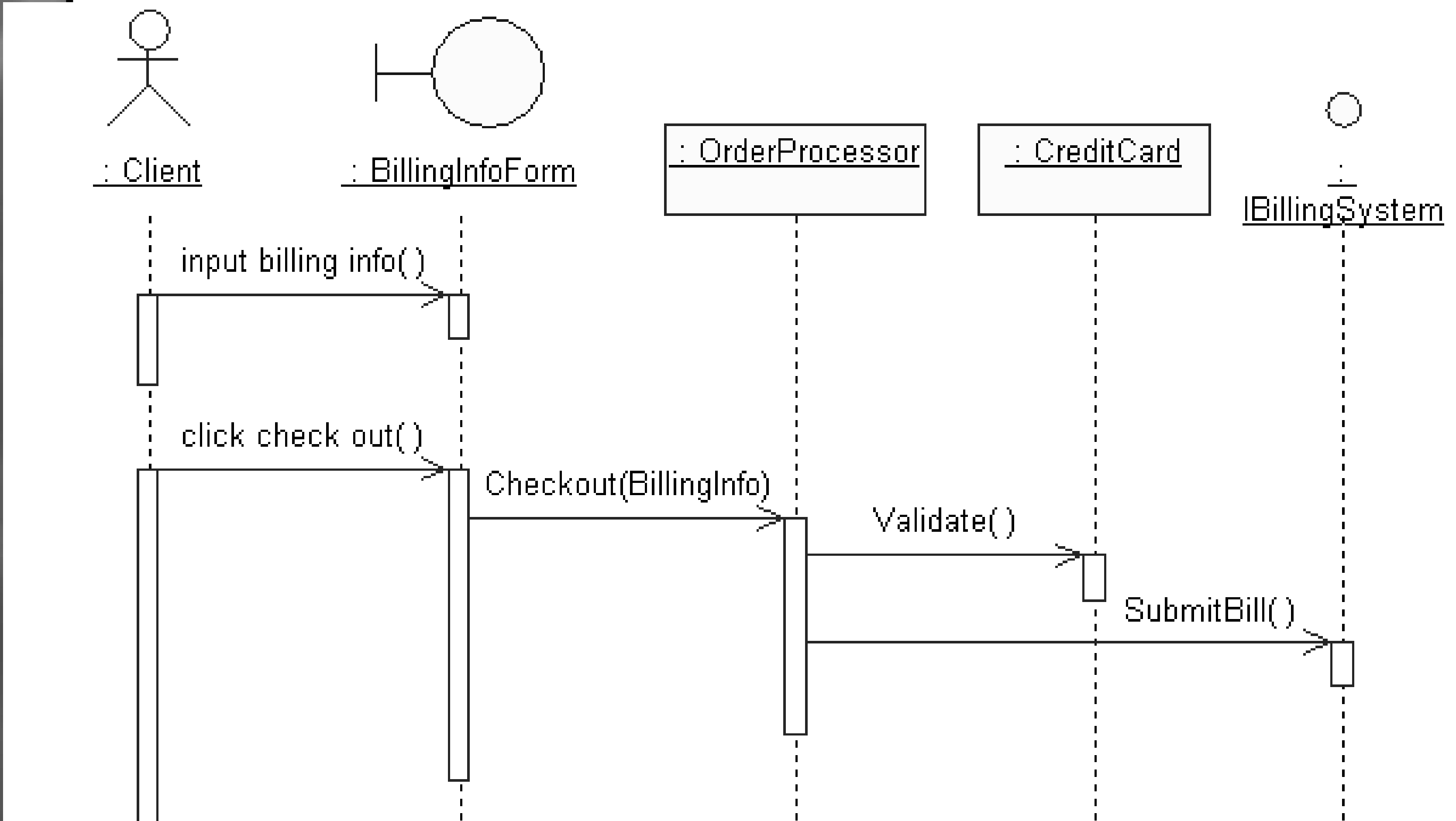
Subsystem可以realize两个以上的接口

一个接口可以被二个以上的子系统realize



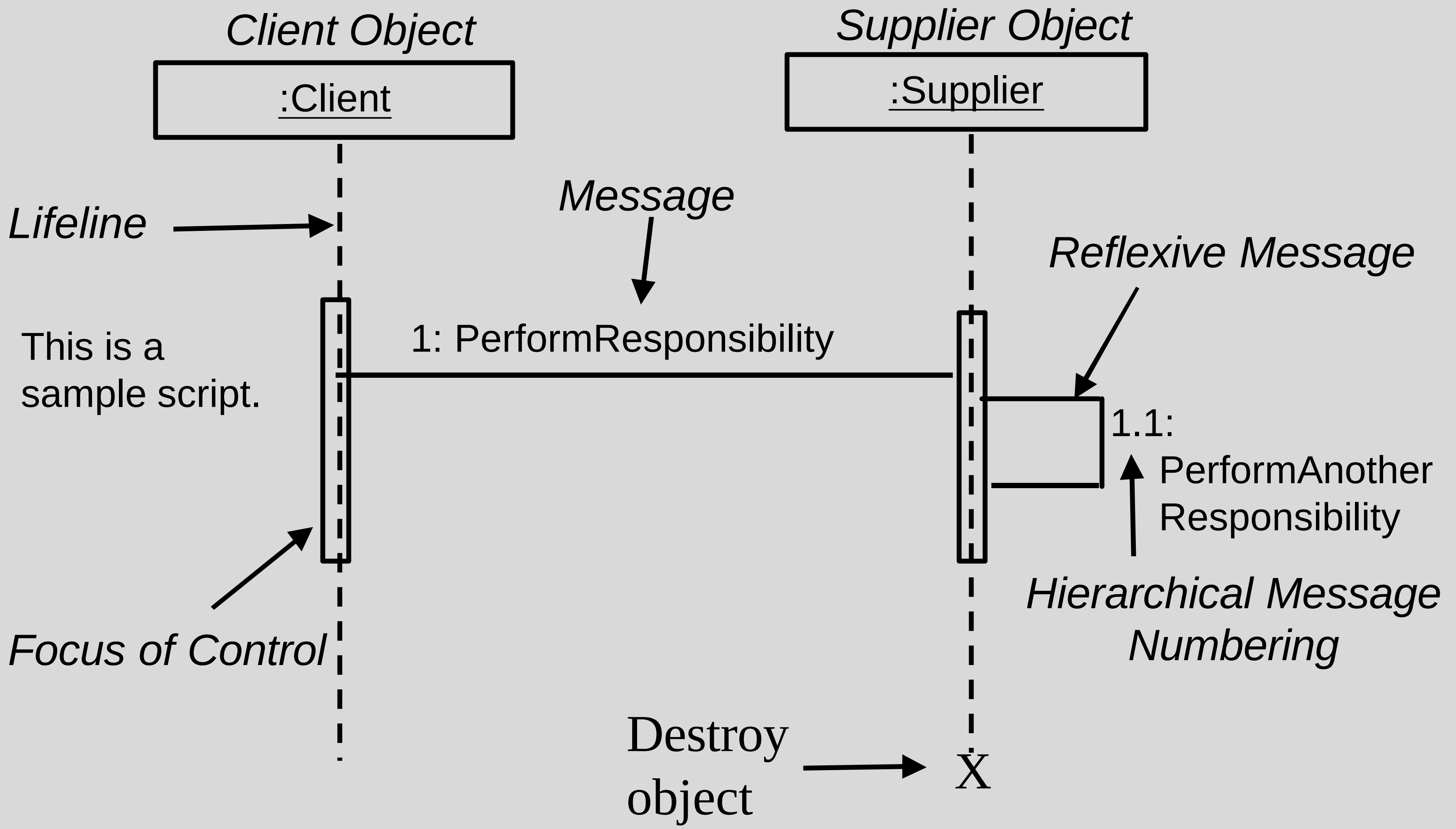
Use Case Design

Walk thru flow of events in use case by sequence diagram



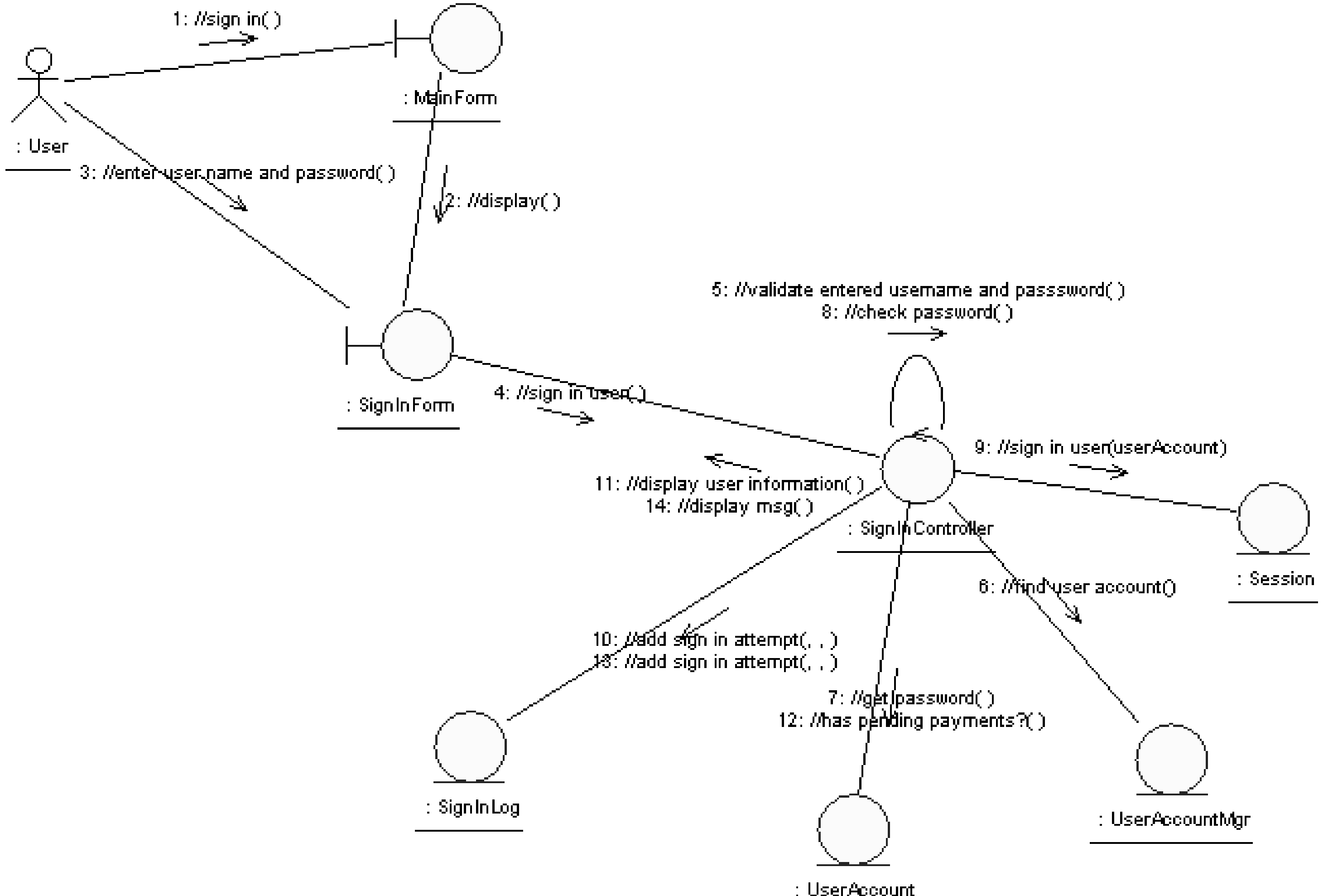
Sequence Diagrams

A sequence diagram displays object interactions arranged in a time sequence



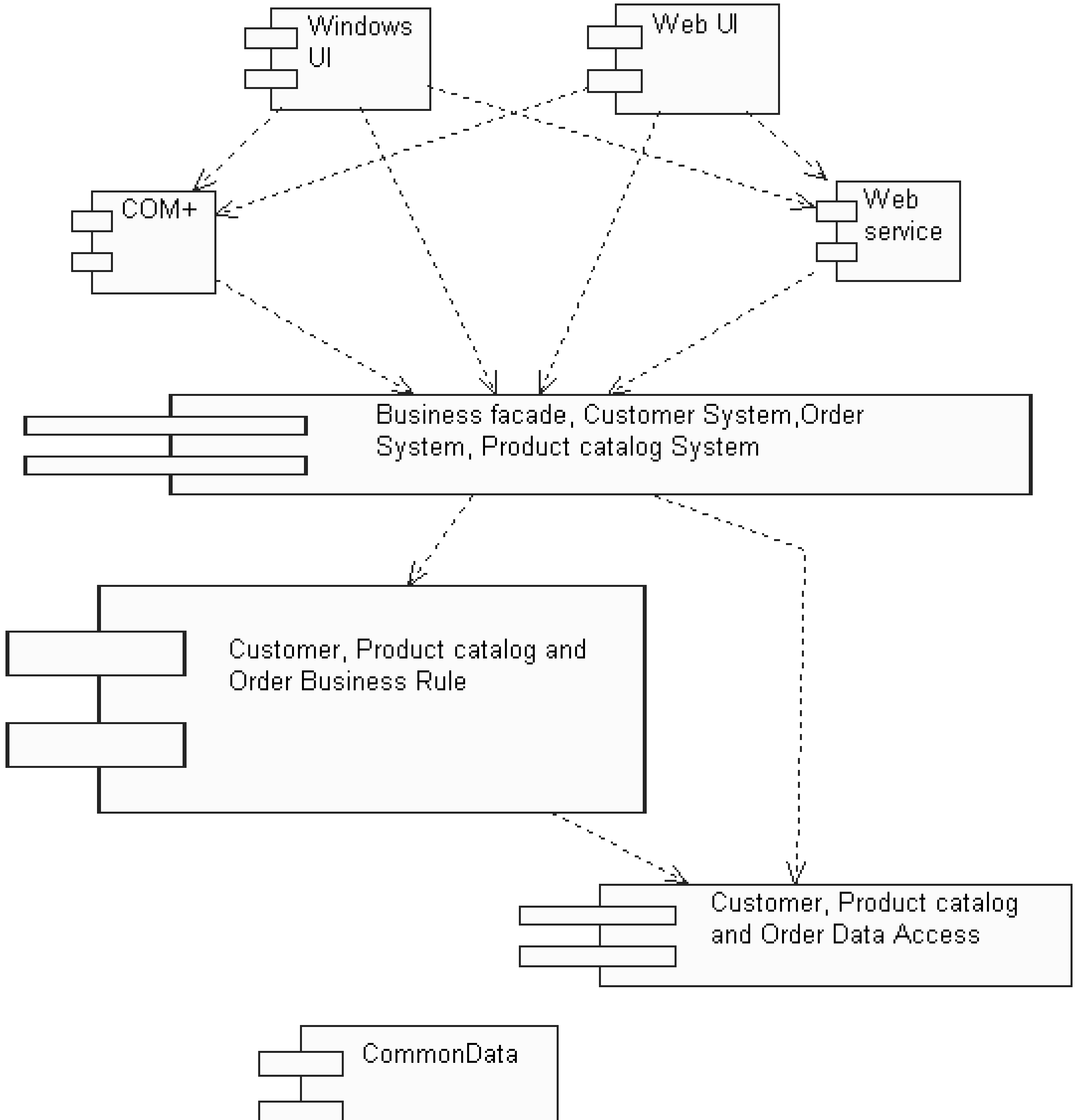
Collaboration Diagram

A collaboration diagram displays object interactions organized around objects and their links to one another

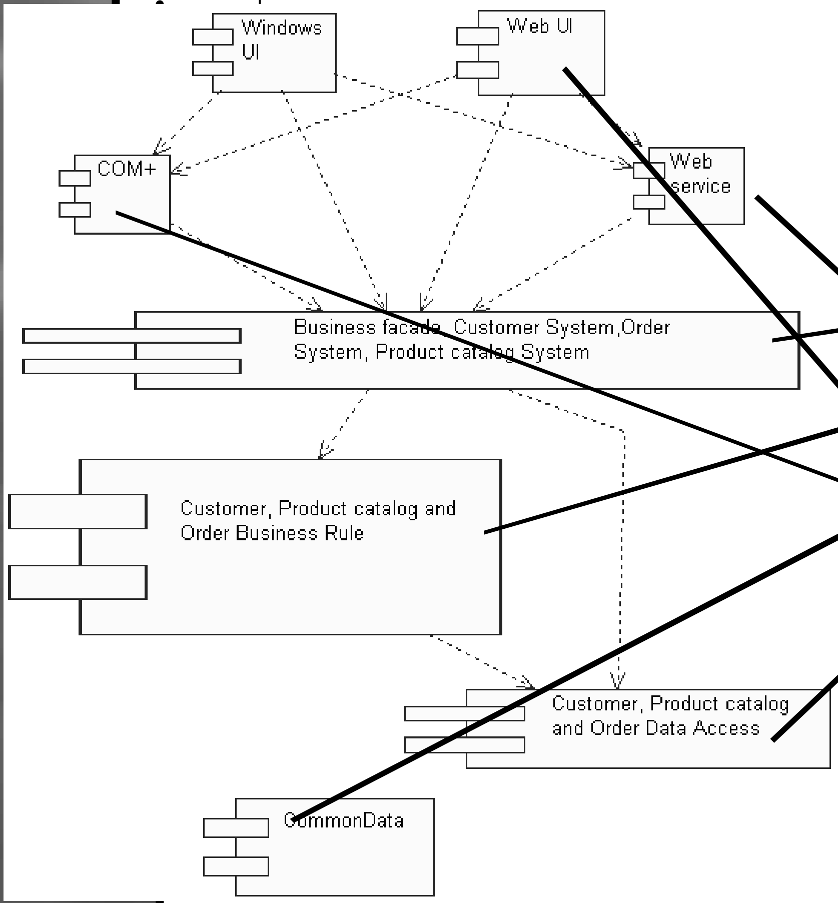

1: //sign in( )
: MainForm
: User
3: //enter user name and password( )
2: //display( )
5: //validate entered username and passsword( )
8: //check password( )
: SignInForm
4: //sign in user( )
9: //sign in user(userAccount)
11: //display user information( )
14: //display msg( )
: SignInController
: Session
6: //find user account()
10: //add sign in attempt(, ,)
13: //add sign in attempt(, ,)
7: //get password( )
12: //has pending payments?( )
: SignInLog
: UserAccountMgr
: UserAccount

Implementation Model and Component Diagram

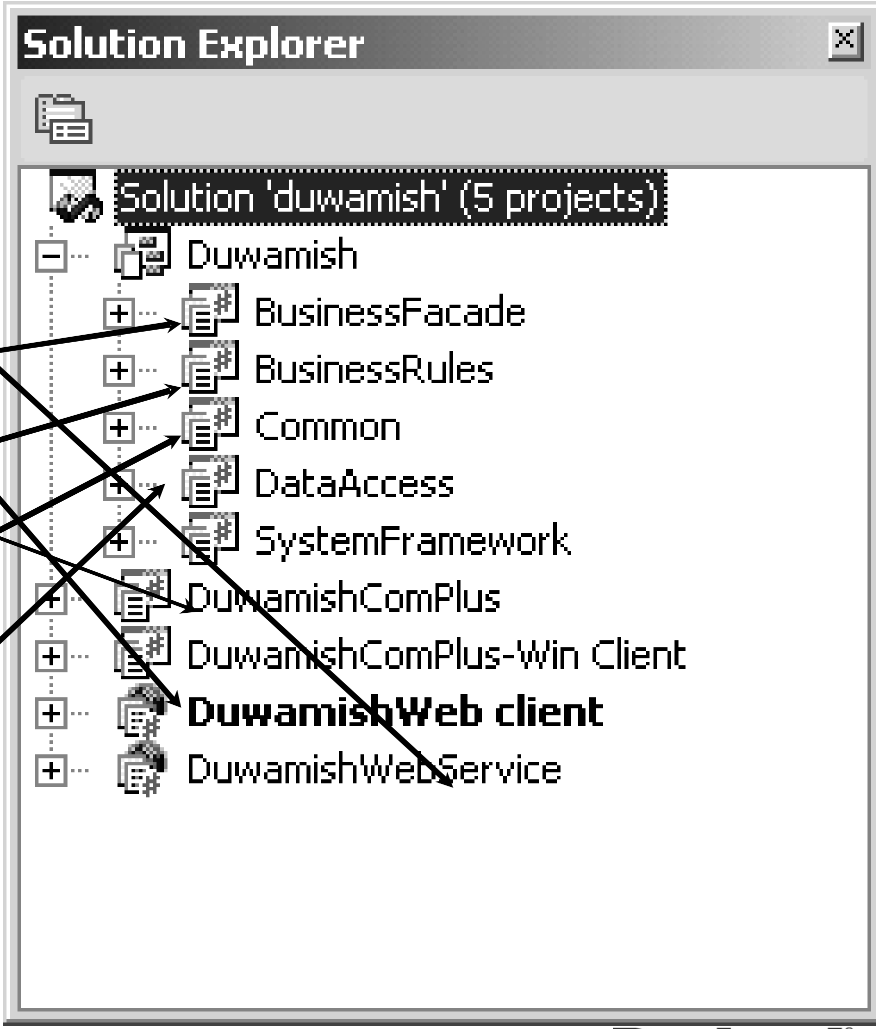
For small system, we use Business fa çade component Business rule component, a Data access component.


Windows UI
Web UI
COM+
Web service
Business facade, Customer System,Order System, Product catalog System
Customer, Product catalog and Order Business Rule
Customer, Product catalog and Order Data Access
CommonData

组件设计与VS.NET项目规划

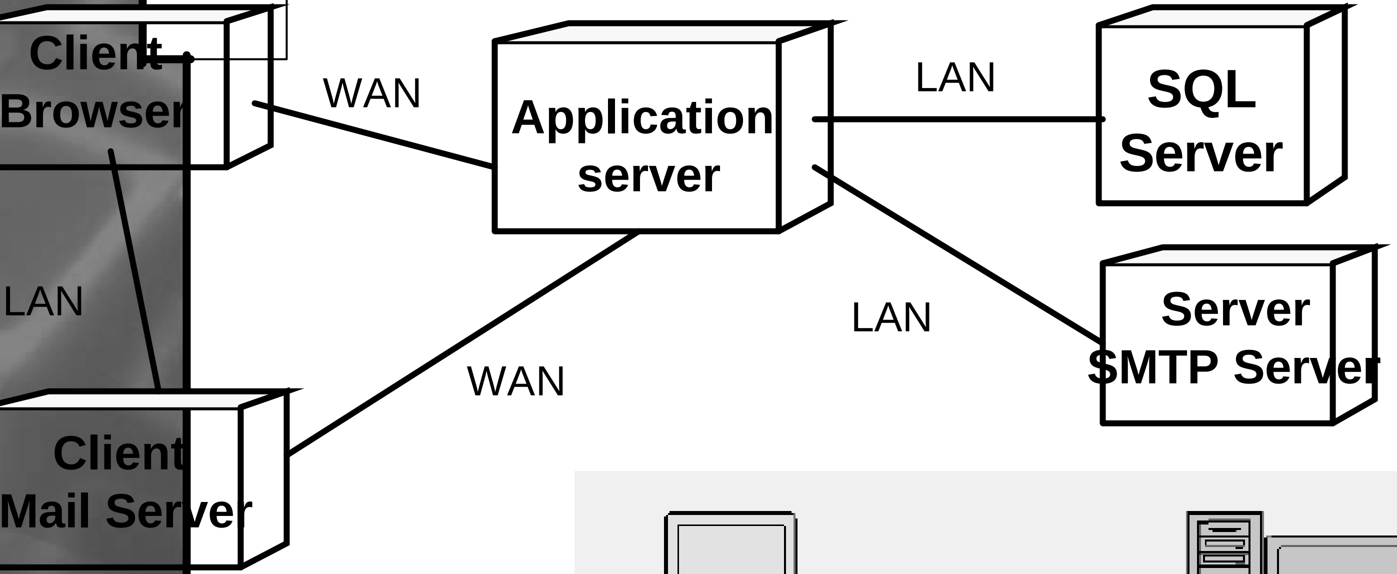


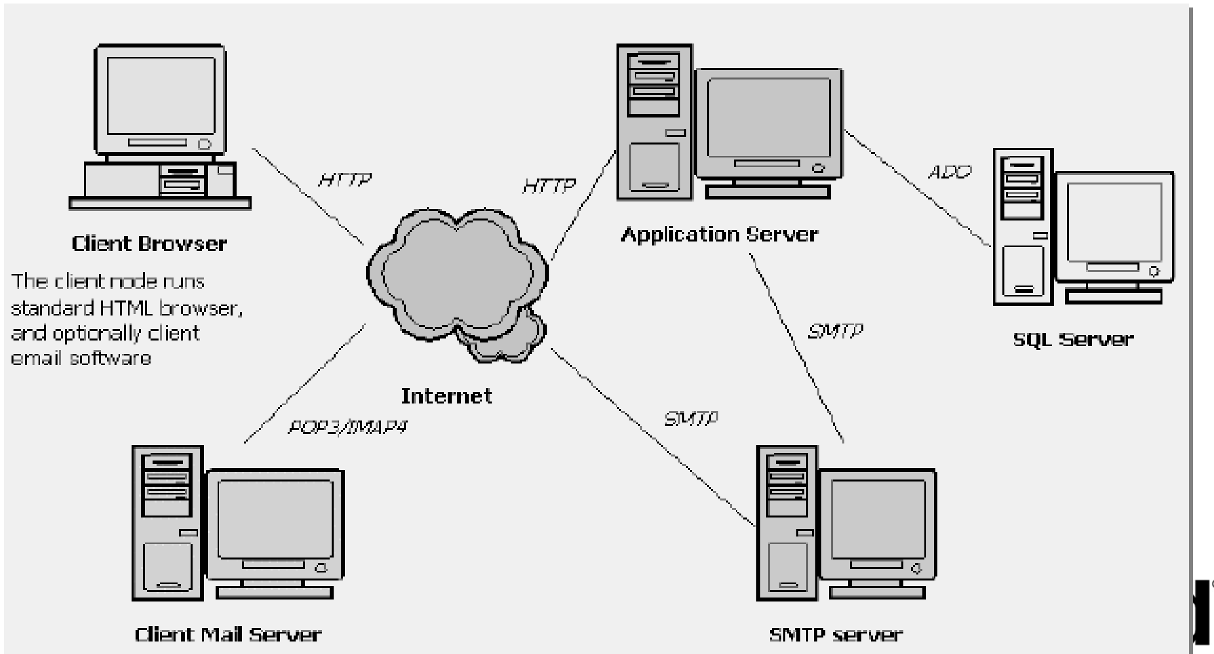
Windows UI
Web UI
COM+
Web service
Business facade, Customer System,Order System, Product catalog System
Customer, Product catalog and Order Business Rule
Customer, Product catalog and Order Data Access
CommonData



Borland

Deployment Diagram





The client node runs standard HTML browser, and optionally client email software

What we learn about UML Diagrams ?

Behavior :

- ✓ Use Case
- ✓ Activity
- ✓ Sequence
- ✓ Collaboration
- ☐ State Chart

Structural:

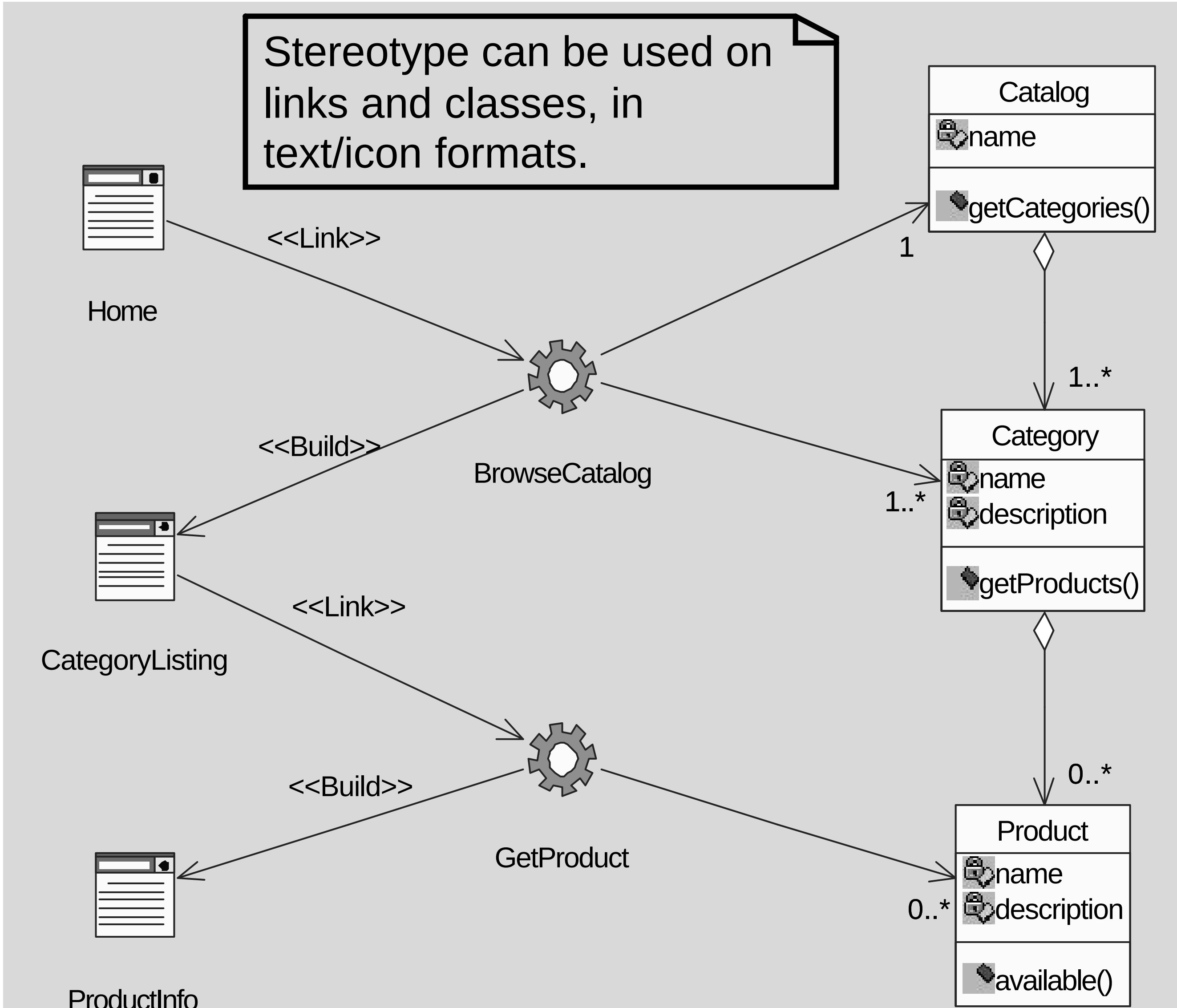
- Class
- Component
- Deployment
- Object

Model Management:

- Packages (class diagram contains packages)
- Subsystems (class diagram contains subsystems)
- Models (class diagram contains models)

扩充UML的描述能力

- Stereotype
- Note
- Tag



Design Pattern

A pattern is a solution to a problem in a context, it documents in an abstract and compact form

- ➢ the problem
- ➢ the context in which it occurs
- ➢ a good solution
- ✓ and it embodies wisdom about how the solution addresses the problem.

Different levels

- ◆ basic building blocks
- ◆ design patterns
- ◆ architecture patterns

Façade Design Pattern

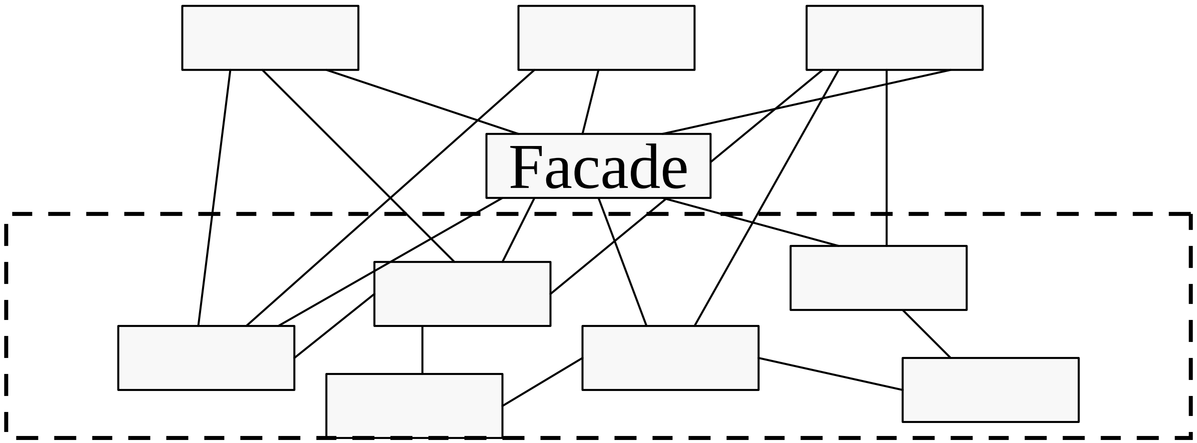
Façade: defines a clean, high-level interface to a subsystem.

Context: building easy-to-use and maintain subsystems

Problem: Each class in the subsystem provides part of the subsystem's functionality, clients has to know the inside, changes to the subsystem may require changes to the clients.

Solution: Add an interface class (the façade class) that knows the structure of the subsystem and forwards requests…

Consequences: no or less dependency of client from structure of subsystem, ideal for layered subsystems



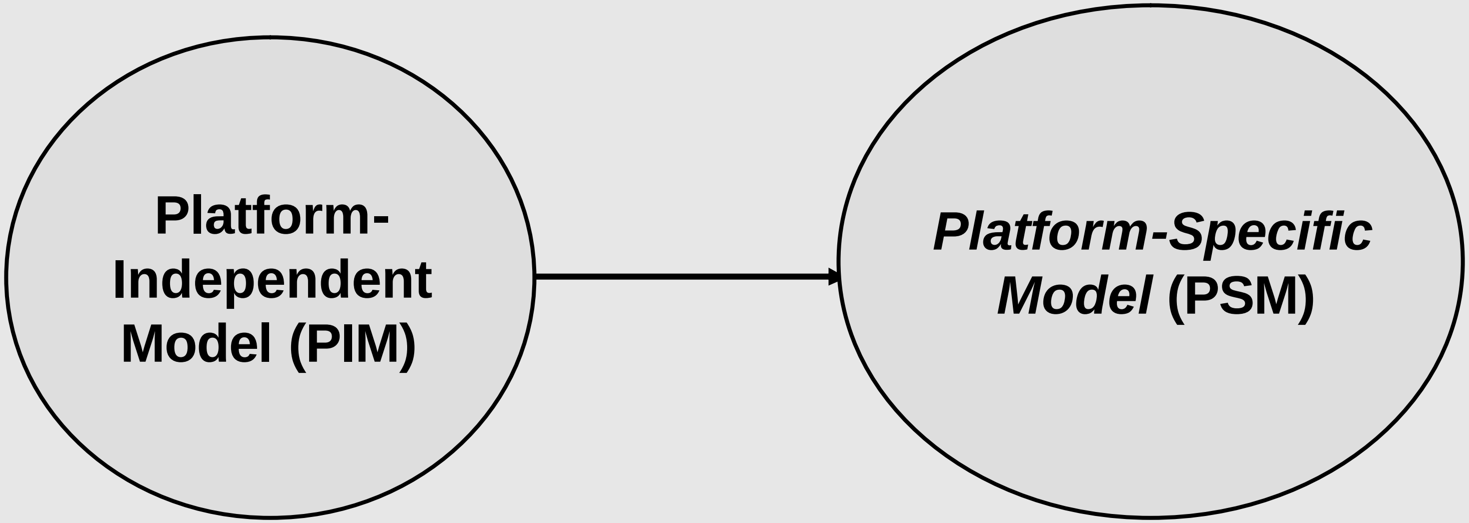
What is Model Driven Architecture?

A New Way to Specify and Build Systems

- **2001年由 OMG 制定的新开发架构**
- **以 UML 塑模为基础**
- **支持完整开发周期** : analysis, design, implementation, deployment, maintenance, evolution & integration with later systems
- **内建协同运作性及跨平台性**
- **降低开发初期成本及提高** ROI

- **可套用至你所使用的任何环境** :
  - Programming language
  - Operating system
  - Network
  - Middleware

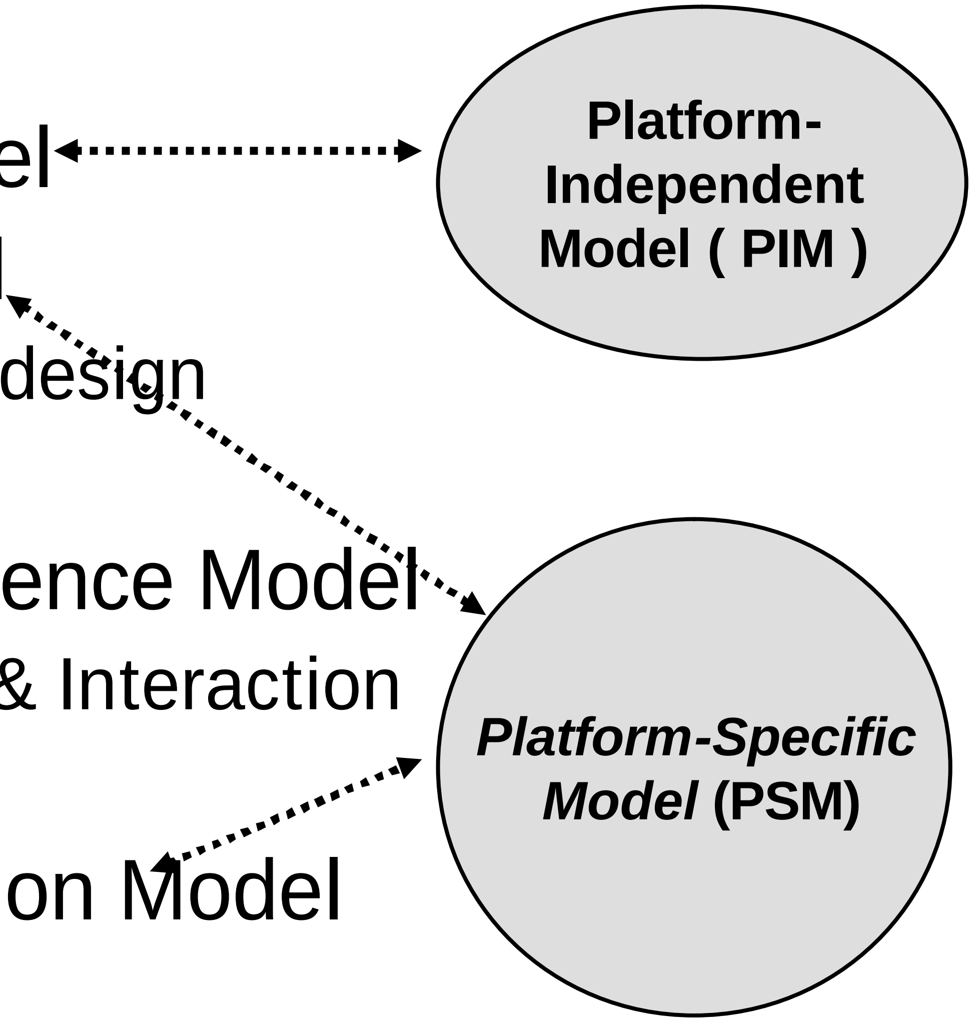
Model Driven Architecture


Platform-Independent Model (PIM)
Platform-Specific Model (PSM)

MDA tool applies an standard mapping to generate *Platform-Specific Model* (PSM) from the PIM. Code is partially automatic, partially hand-written.

开发流程

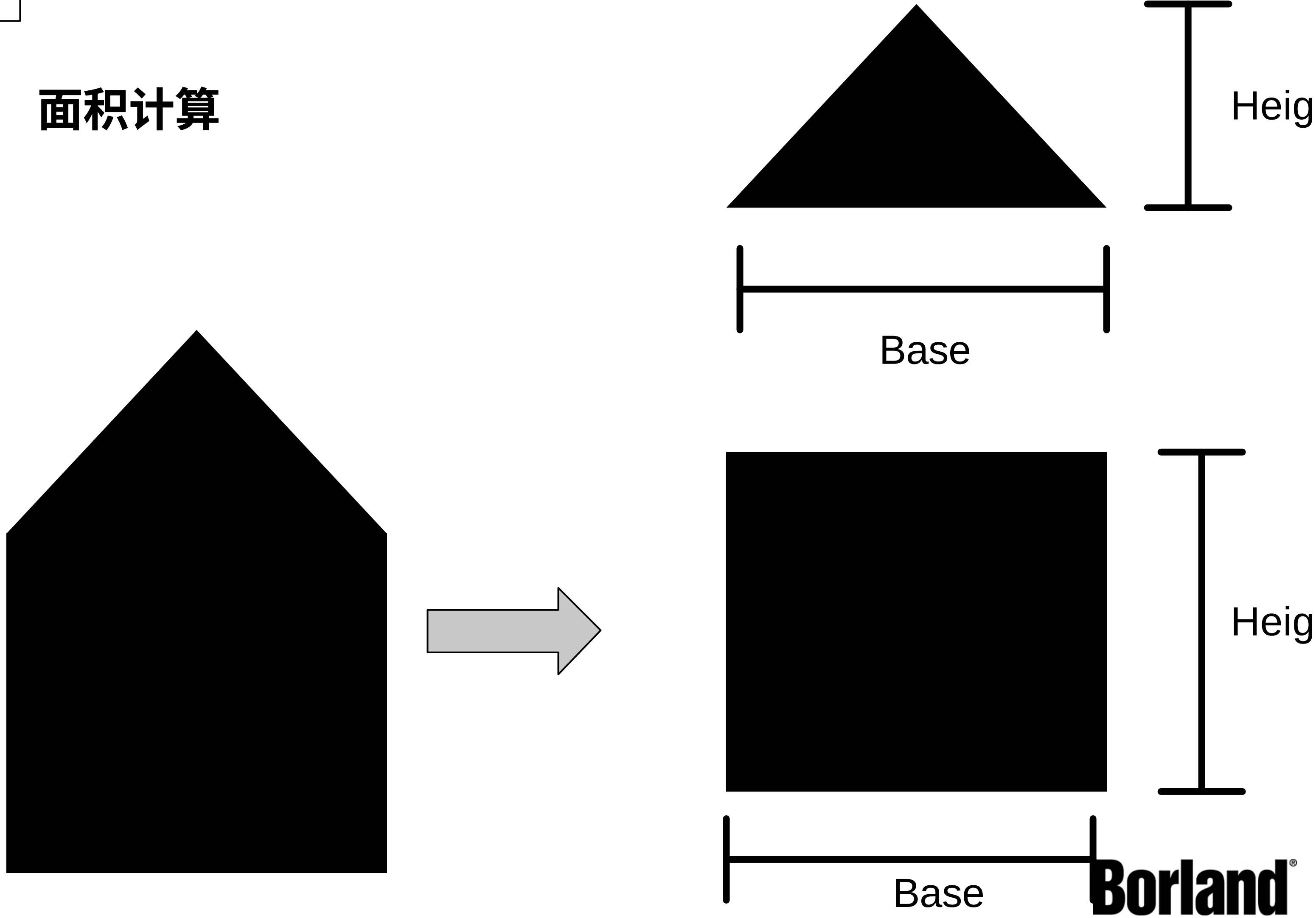
- Analysis Model
- Design Model
  - Architecture design
  - Data Model
  - User Experience Model
    - look-Feel & Interaction
- Implementation Model


Platform-Independent Model (PIM)
Platform-Specific Model (PSM)

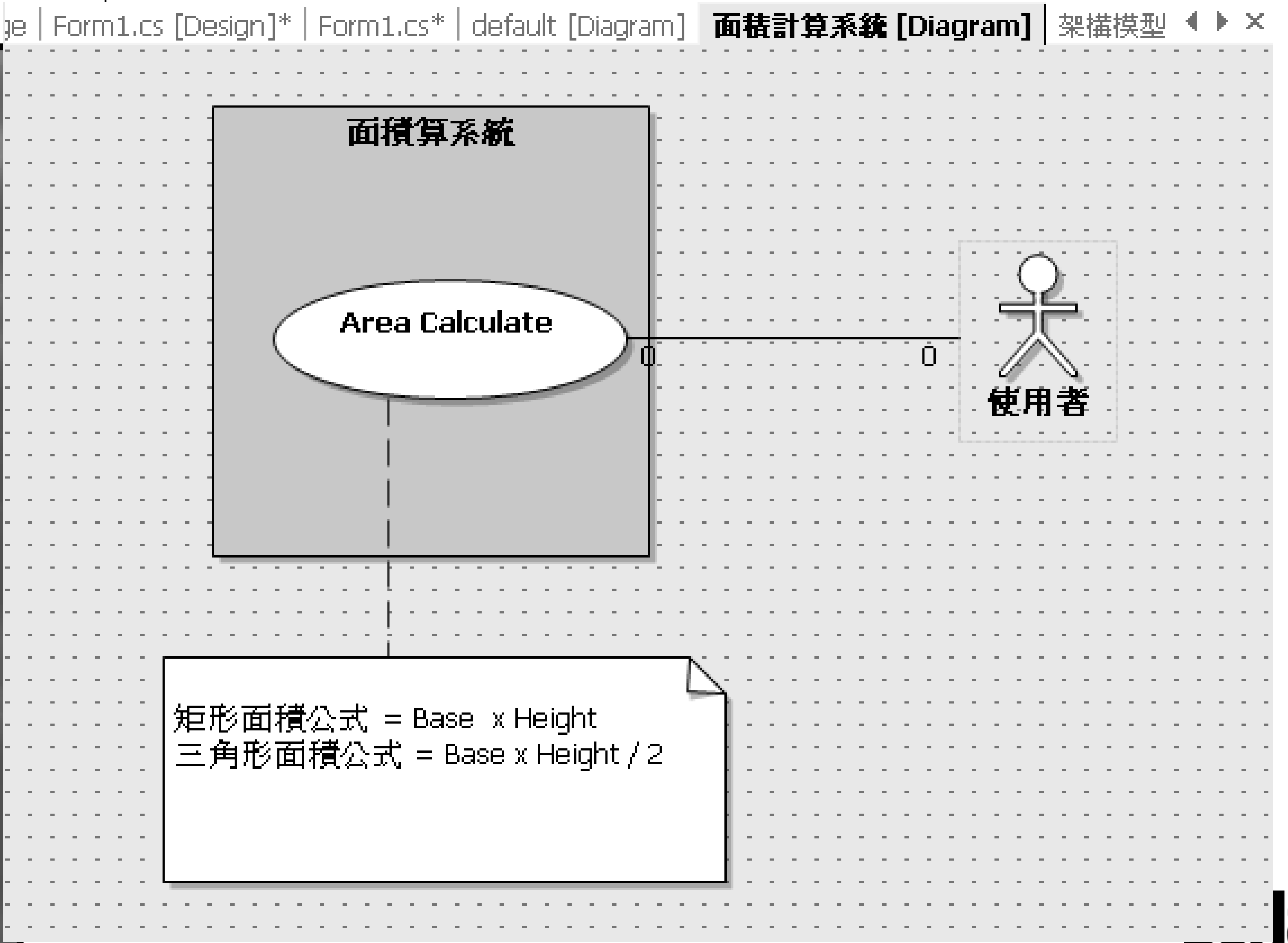
Borland

展示实例

面积计算

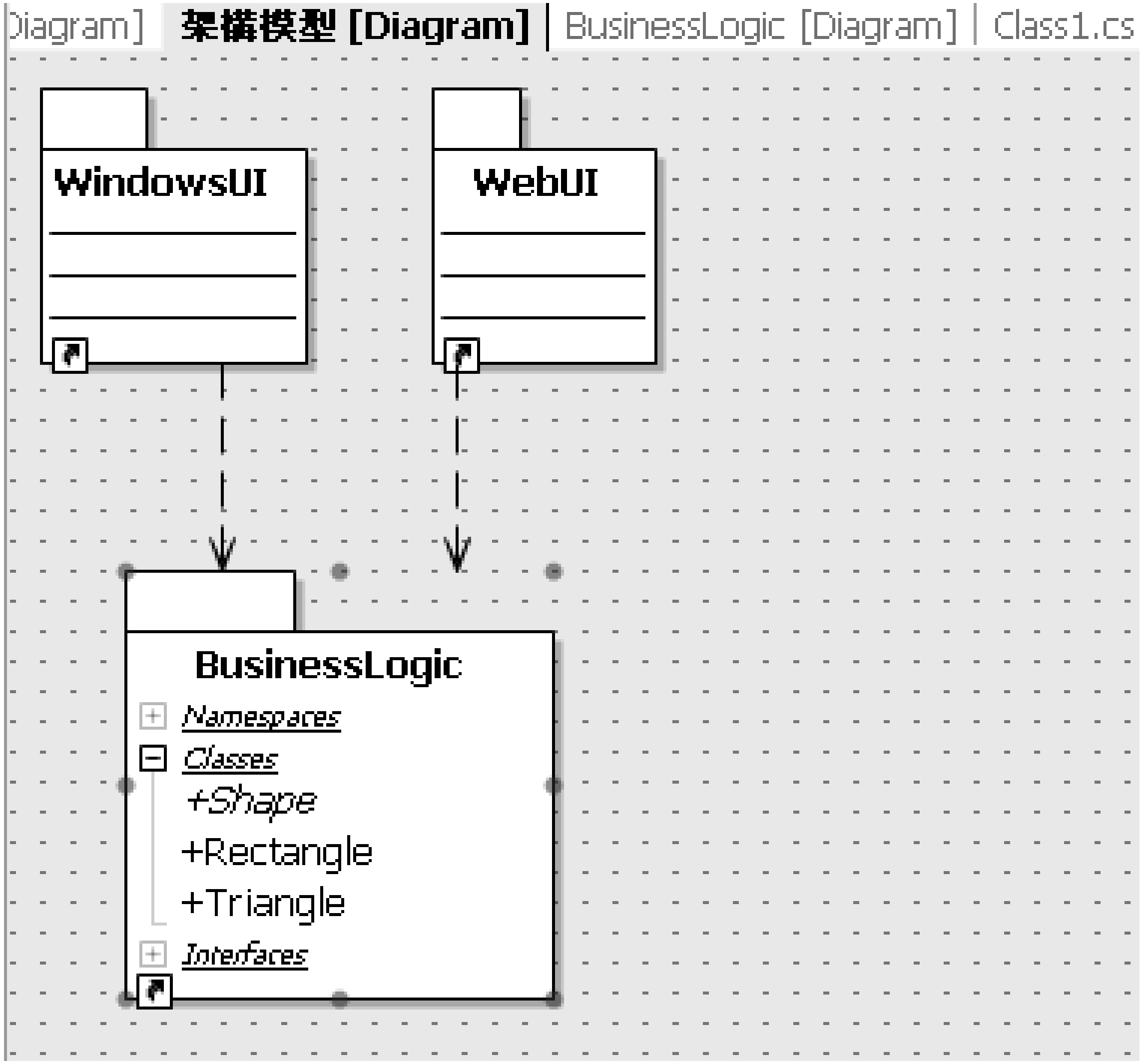


Use Case Model


ge | Form1.cs [Design]* | Form1.cs* | default [Diagram] | 面積計算系統 [Diagram] | 架構模型
面積算系統
Area Calculate
0
0
使用者

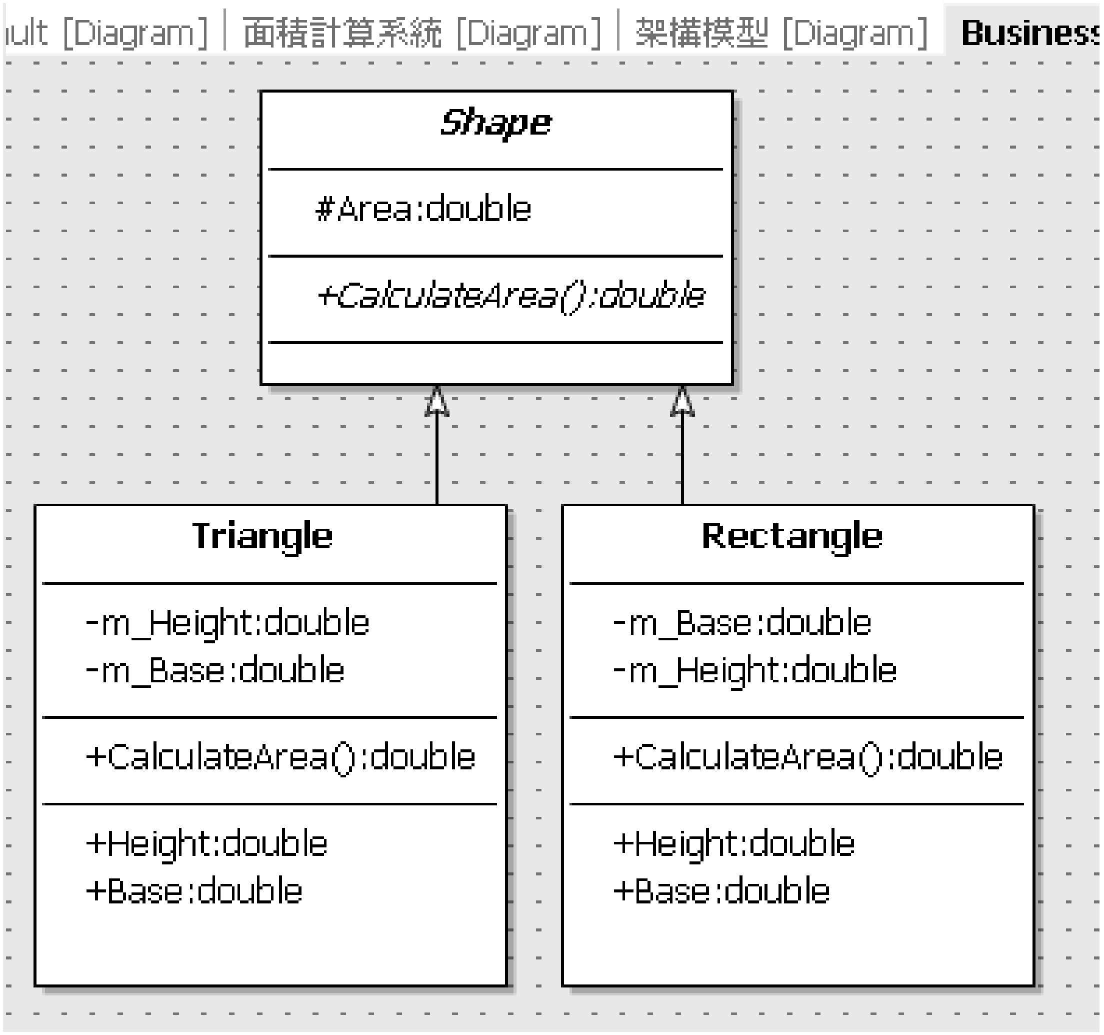
矩形面積公式 = Base x Height
三角形面積公式 = Base x Height / 2

架构设计



Borland

系统设计



Shape
#Area:double
+CalculateArea():double

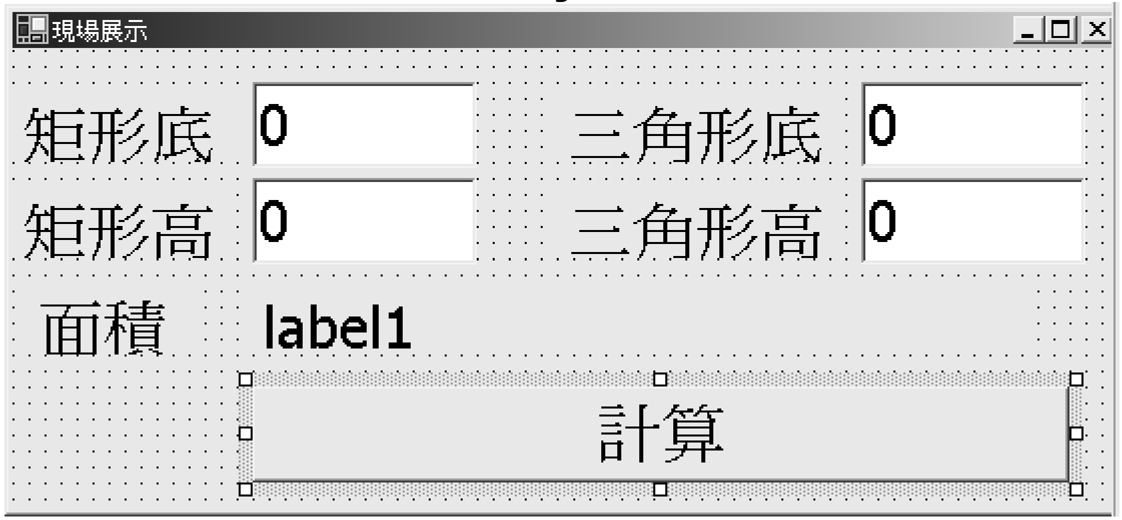
Triangle
-m_Height:double
-m_Base:double
+CalculateArea():double
+Height:double
+Base:double

Rectangle
-m_Base:double
-m_Height:double
+CalculateArea():double
+Height:double
+Base:double

Business Logic

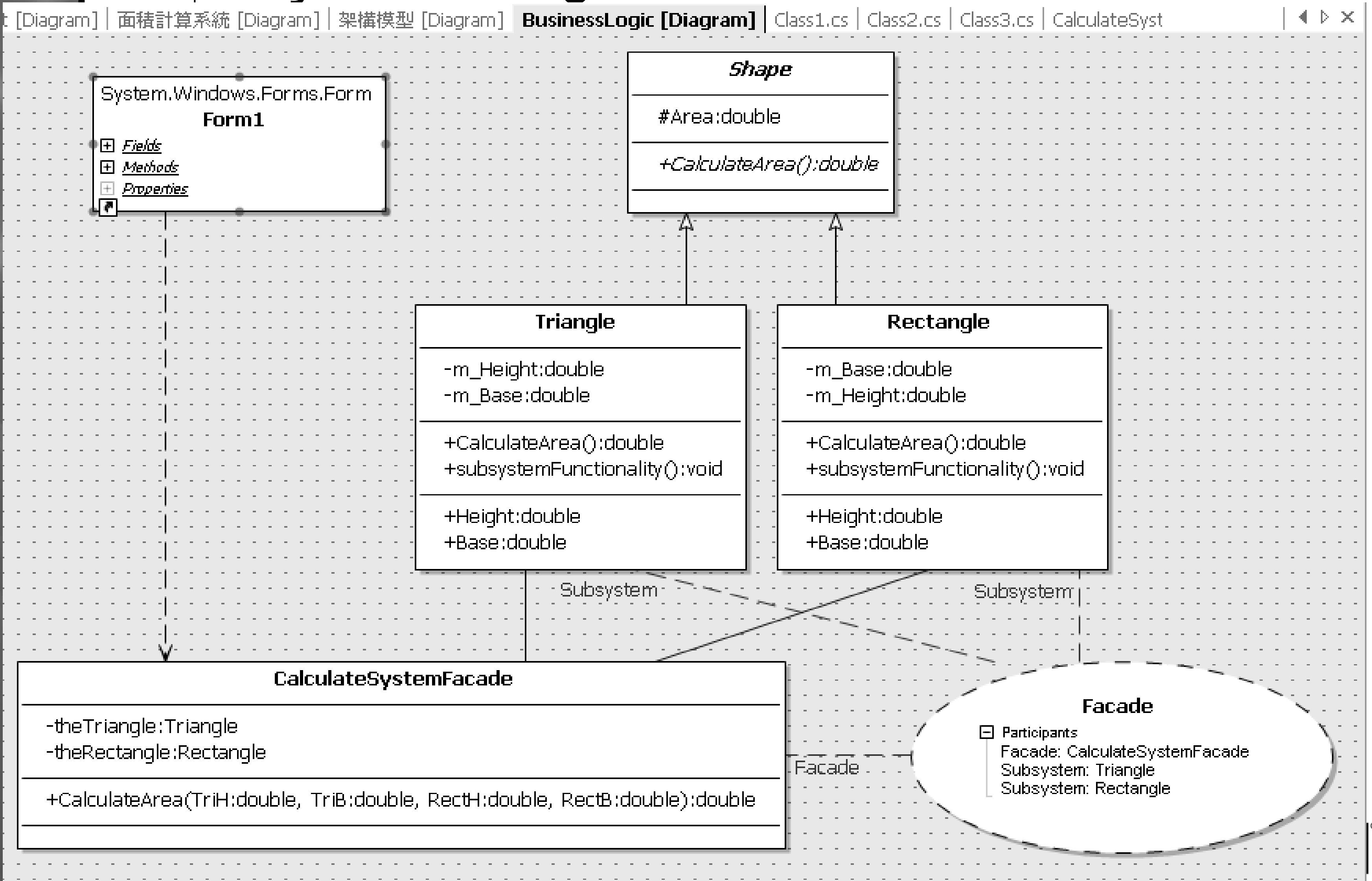
```
namespace BusinessLogic
{
  public class Triangle: Shape
{
      public override double CalculateArea()
      {
         this.Area = this.m_Base  * this.m_Height / 2;
         return (this.Area);
      }
     private double m_Height;
     private double m_Base;
     public double Height {
       get {
          return (m_Height);
       }
       set {
          m_Height = value;
       }
```

Presentation Layer



```
theRectangle.Base = RectB;
theRectangle.Height = RectH;
theTriangle.Base = TriB;
theTriangle.Height = TriH;
result = theTriangle.CalculateArea()+ theRectangle.CalculateArea();
```

Façade Design Pattern


[Diagram] | 面積計算系統 [Diagram] | 架構模型 [Diagram] | BusinessLogic [Diagram] | Class1.cs | Class2.cs | Class3.cs | CalculateSyst

System.Windows.Forms.Form
Form1
⊞ *Fields*
⊞ *Methods*
⊞ *Properties*

Shape
#Area:double
+CalculateArea():double

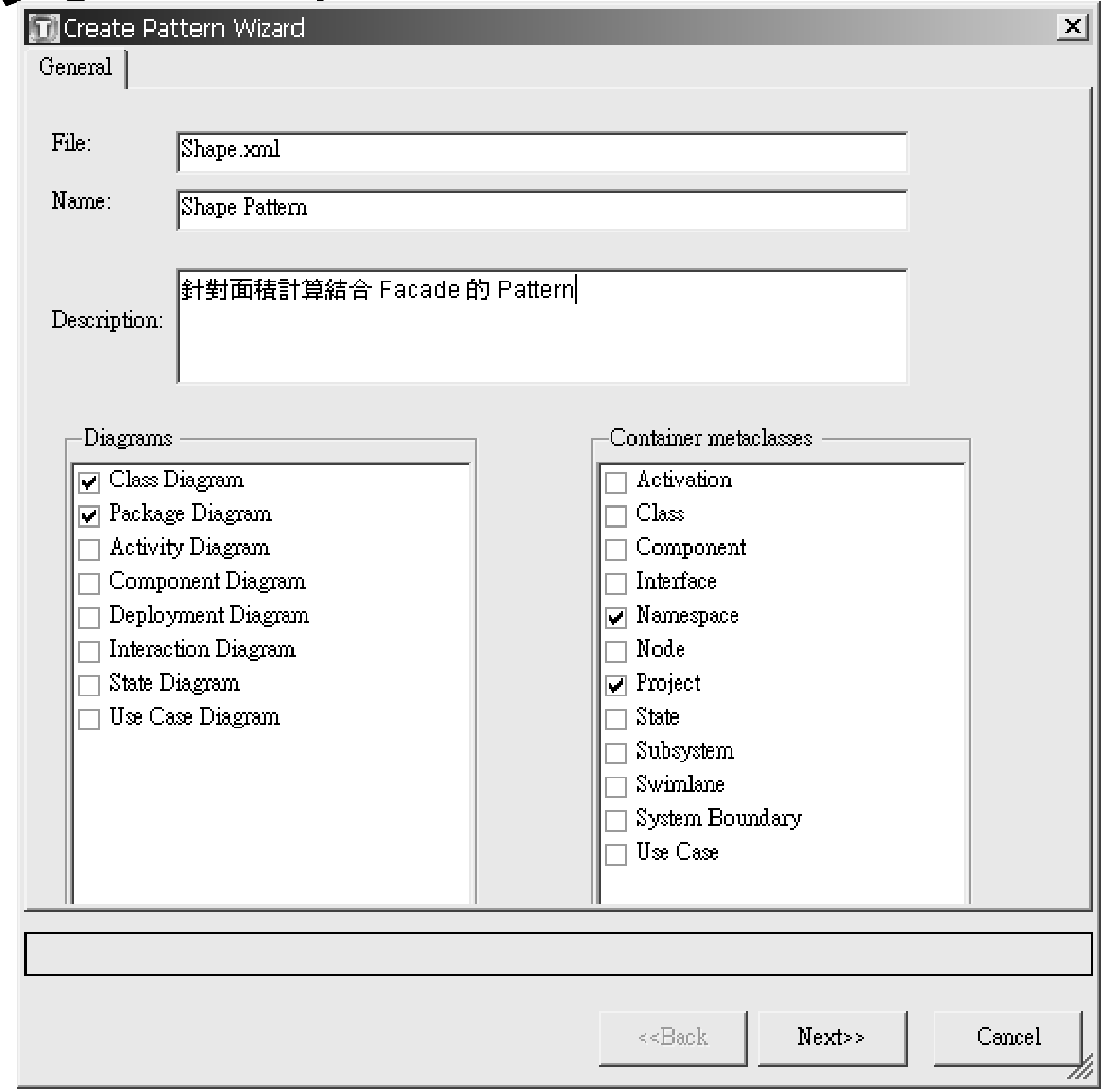
Triangle
-m_Height:double
-m_Base:double
+CalculateArea():double
+subsystemFunctionality():void
+Height:double
+Base:double

Rectangle
-m_Base:double
-m_Height:double
+CalculateArea():double
+subsystemFunctionality():void
+Height:double
+Base:double

CalculateSystemFacade
-theTriangle:Triangle
-theRectangle:Rectangle
+CalculateArea(TriH:double, TriB:double, RectH:double, RectB:double):double

Facade
⊟ Participants
Facade: CalculateSystemFacade
Subsystem: Triangle
Subsystem: Rectangle

定义 Design Pattern



File: Shape.xml

Name: Shape Pattern

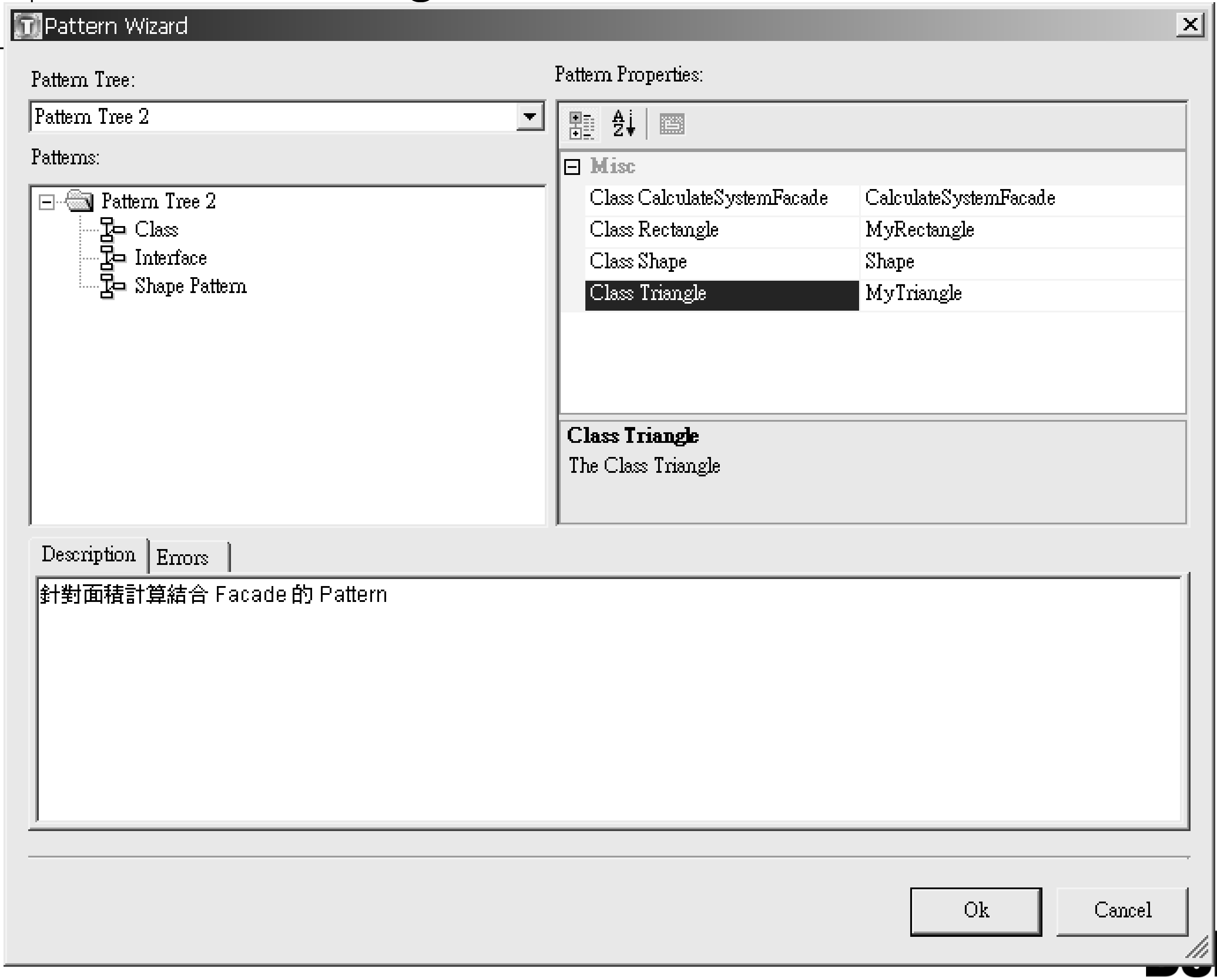
Description: 針對面積計算結合 Facade 的 Pattern

Diagrams
- [✓] Class Diagram
- [✓] Package Diagram
- [ ] Activity Diagram
- [ ] Component Diagram
- [ ] Deployment Diagram
- [ ] Interaction Diagram
- [ ] State Diagram
- [ ] Use Case Diagram

Container metaclasses
- [ ] Activation
- [ ] Class
- [ ] Component
- [ ] Interface
- [✓] Namespace
- [ ] Node
- [✓] Project
- [ ] State
- [ ] Subsystem
- [ ] Swimlane
- [ ] System Boundary
- [ ] Use Case

Borland®

Reuse Design Pattern



Class CalculateSystemFacade	CalculateSystemFacade
Class Rectangle	MyRectangle
Class Shape	Shape
Class Triangle	MyTriangle

Class Triangle
The Class Triangle

針對面積計算結合 Facade 的 Pattern

Together Edition for Microsoft Visual Studio .NET

UML Diagram **绘制功能**

LiveSource™ **技术实时同步** Model **与程序代码**

深度整合 Microsoft Visual Studio .NET **自动化文件产生**

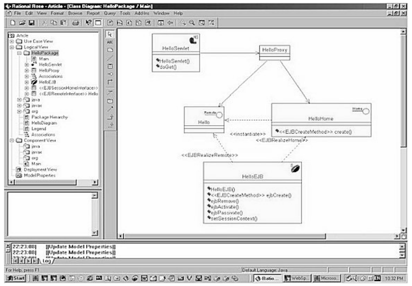
支援 Pattern

支持 XMI **汇入与汇出**

LiveSource™技术

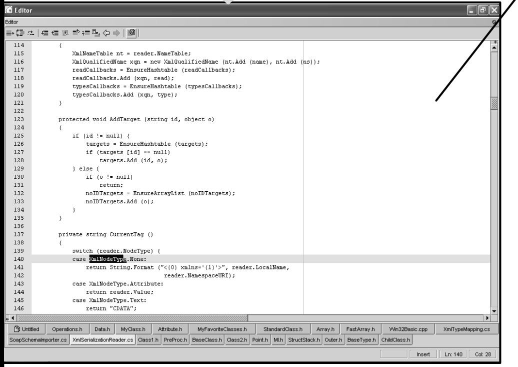
Reverse/Forward Engineer

IBM/Rational



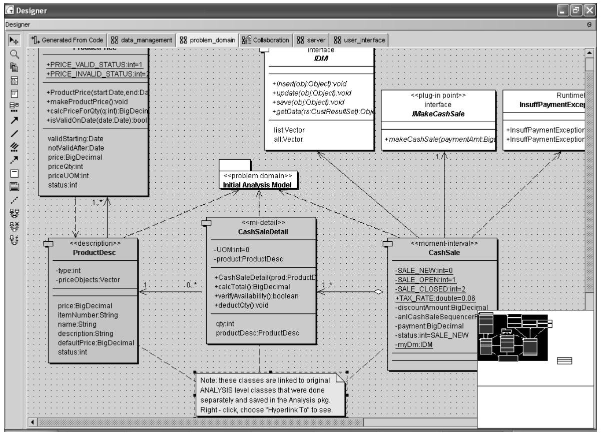
Binary Repository

？ ？ ？



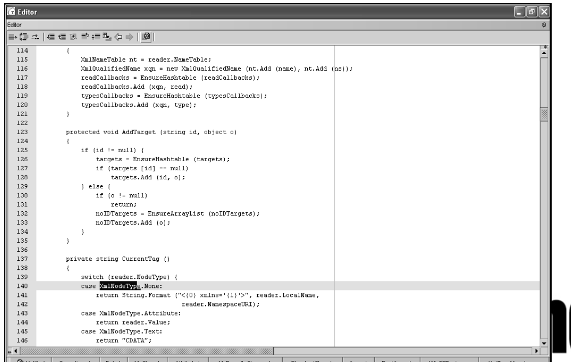
```
{
    XmlNameTable nt = reader.NameTable;
    XmlQualifiedName xqn = new XmlQualifiedName (nt.Add (name), nt.Add (ns));
    readCallbacks = EnsureHashtable (readCallbacks);
    readCallbacks.Add (xqn, read);
    typesCallbacks = EnsureHashtable (typesCallbacks);
    typesCallbacks.Add (xqn, type);
}

protected void AddTarget (string id, object o)
{
    if (id != null) {
        targets = EnsureHashtable (targets);
        if (targets [id] == null)
            targets.Add (id, o);
    } else {
        if (o != null)
            return;
        noIDTargets = EnsureArrayList (noIDTargets);
        noIDTargets.Add (o);
    }
}

private string CurrentTag ()
{
    switch (reader.NodeType) {
    case XmlNodeType.None:
        return String.Format ("<{0} xmlns='{1}'>", reader.LocalName,
                              reader.NamespaceURI);
    case XmlNodeType.Attribute:
        return reader.Value;
    case XmlNodeType.Text:
        return "CDATA";
```

Borland Together



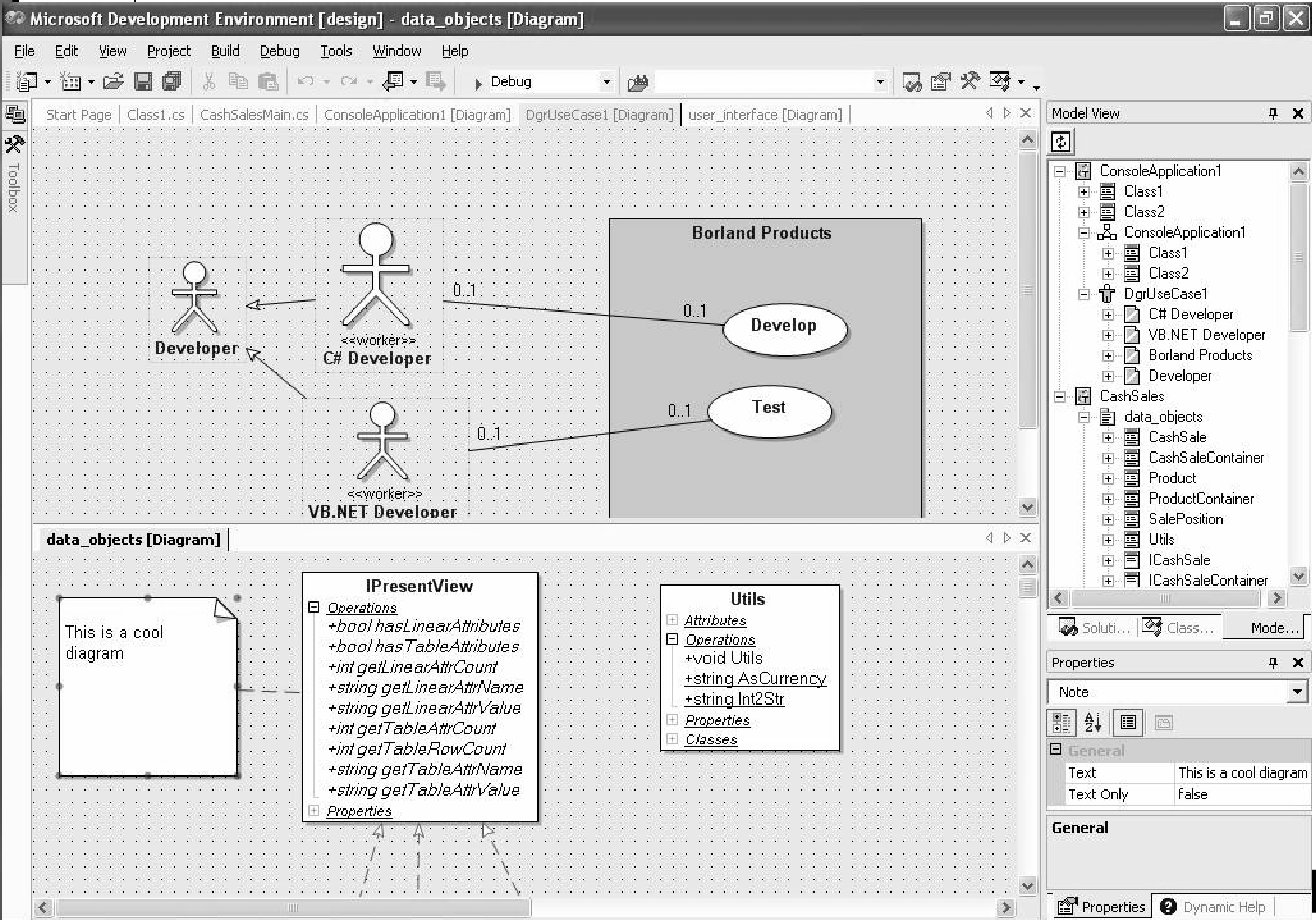
Designer
Generated From Code
data_management
problem_domain
Collaboration
server
user_interface
IDM
<<plug-in point>> interface IMakeCashSale
InsuffPaymentException
<<problem domain>> Initial Analysis Model
<<description>> ProductDesc
<<mi-detail>> CashSaleDetail
<<moment-interval>> CashSale
Note: these classes are linked to original ANALYSIS level classes that were done separately and saved in the Analysis pkg. Right - click, choose "Hyperlink To" to see.

？ ？ ？ ？



```
{
    XmlNameTable nt = reader.NameTable;
    XmlQualifiedName xqn = new XmlQualifiedName (nt.Add (name), nt.Add (ns));
    readCallbacks = EnsureHashtable (readCallbacks);
    readCallbacks.Add (xqn, read);
    typesCallbacks = EnsureHashtable (typesCallbacks);
    typesCallbacks.Add (xqn, type);
}

protected void AddTarget (string id, object o)
{
    if (id != null) {
        targets = EnsureHashtable (targets);
        if (targets [id] == null)
            targets.Add (id, o);
    } else {
        if (o != null)
            return;
        noIDTargets = EnsureArrayList (noIDTargets);
        noIDTargets.Add (o);
    }
}

private string CurrentTag ()
{
    switch (reader.NodeType) {
    case XmlNodeType.None:
        return String.Format ("<{0} xmlns='{1}'>", reader.LocalName,
                              reader.NamespaceURI);
    case XmlNodeType.Attribute:
        return reader.Value;
    case XmlNodeType.Text:
        return "CDATA";
```

深度整合 Microsoft Visual Studio .NET



IPresentView
Operations
+bool hasLinearAttributes
+bool hasTableAttributes
+int getLinearAttrCount
+string getLinearAttrName
+string getLinearAttrValue
+int getTableAttrCount
+int getTableRowCount
+string getTableAttrName
+string getTableAttrValue
Properties

Utils
Attributes
Operations
+void Utils
+string AsCurrency
+string Int2Str
Properties
Classes

Model View:

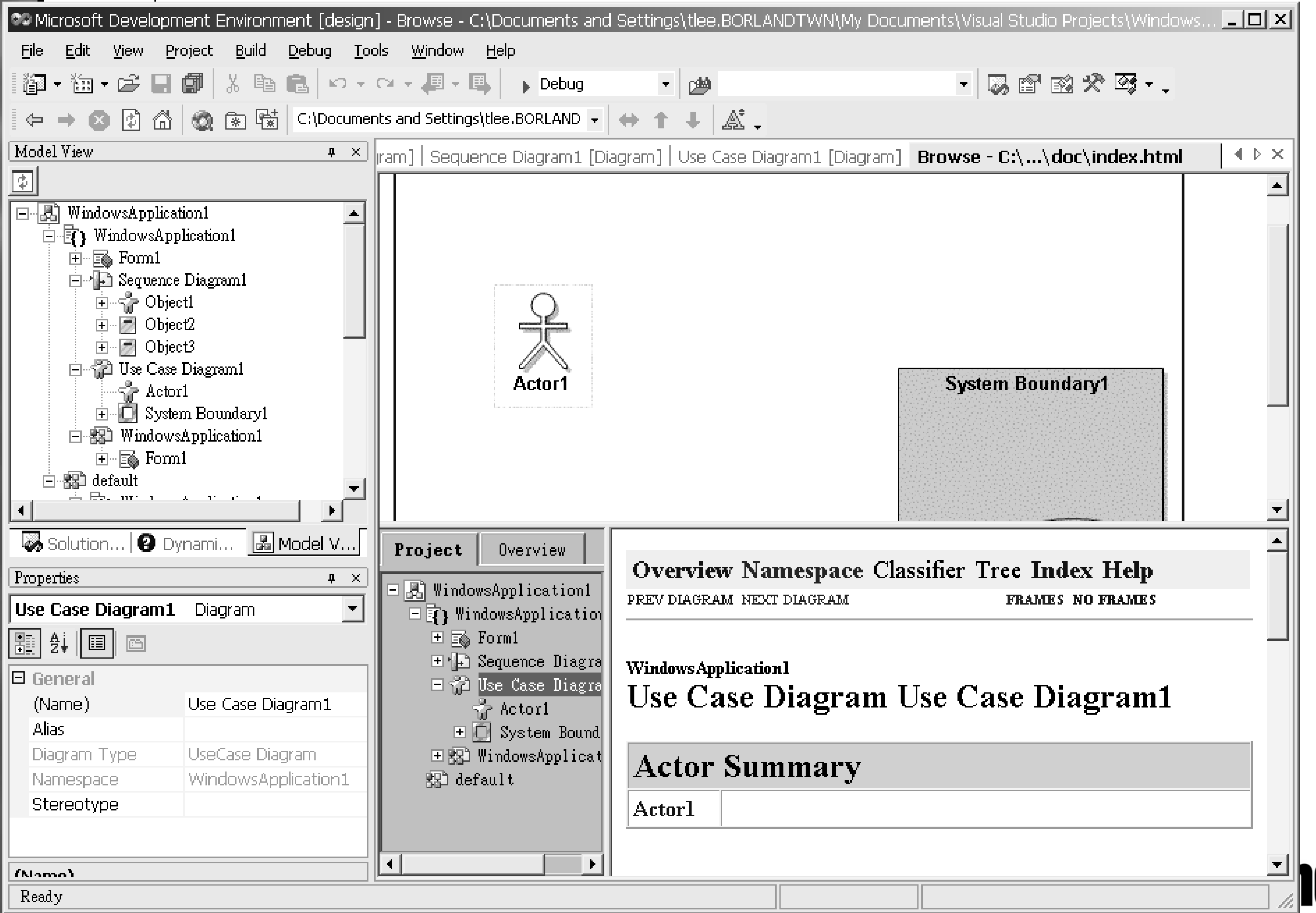
- ConsoleApplication1
  - Class1
  - Class2
  - ConsoleApplication1
    - Class1
    - Class2
  - DgrUseCase1
    - C# Developer
    - VB.NET Developer
    - Borland Products
    - Developer
- CashSales
  - data_objects
    - CashSale
    - CashSaleContainer
    - Product
    - ProductContainer
    - SalePosition
    - Utils
    - ICashSale
    - ICashSaleContainer

Properties: Note

General	
Text	This is a cool diagram
Text Only	false

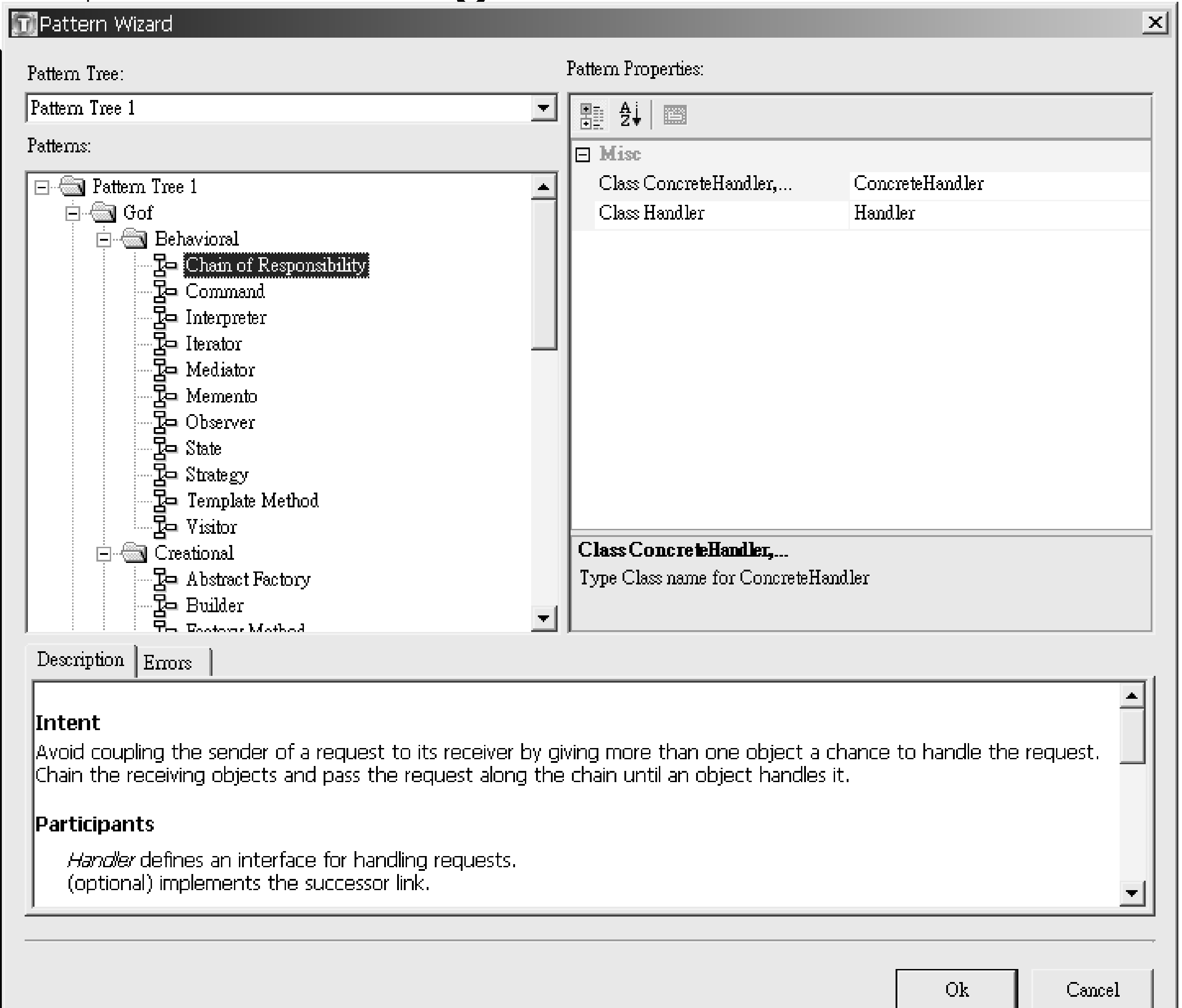
General

自动化文件产生



Microsoft Development Environment [design] - Browse - C:\Documents and Settings\tlee.BORLANDTWN\My Documents\Visual Studio Projects\Windows...
File Edit View Project Build Debug Tools Window Help
Debug
C:\Documents and Settings\tlee.BORLAND
Model View
WindowsApplication1
WindowsApplication1
Form1
Sequence Diagram1
Object1
Object2
Object3
Use Case Diagram1
Actor1
System Boundary1
WindowsApplication1
Form1
default
ram] | Sequence Diagram1 [Diagram] | Use Case Diagram1 [Diagram] | Browse - C:\...\doc\index.html
Actor1
System Boundary1
Solution... Dynami... Model V...
Properties
Use Case Diagram1 Diagram
General
(Name) Use Case Diagram1
Alias
Diagram Type UseCase Diagram
Namespace WindowsApplication1
Stereotype
Project Overview
WindowsApplication1
WindowsApplicatio
Form1
Sequence Diagra
Use Case Diagra
Actor1
System Bound
WindowsApplicat
default
Overview Namespace Classifier Tree Index Help
PREV DIAGRAM NEXT DIAGRAM
FRAMES NO FRAMES
WindowsApplication1
Use Case Diagram Use Case Diagram1
Actor Summary
Actor1
Ready

支援 Design Pattern



- Pattern Tree 1
  - Gof
    - Behavioral
      - Chain of Responsibility
      - Command
      - Interpreter
      - Iterator
      - Mediator
      - Memento
      - Observer
      - State
      - Strategy
      - Template Method
      - Visitor
    - Creational
      - Abstract Factory
      - Builder

Misc	
Class ConcreteHandler,...	ConcreteHandler
Class Handler	Handler

Class ConcreteHandler,...
Type Class name for ConcreteHandler

Intent
Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.

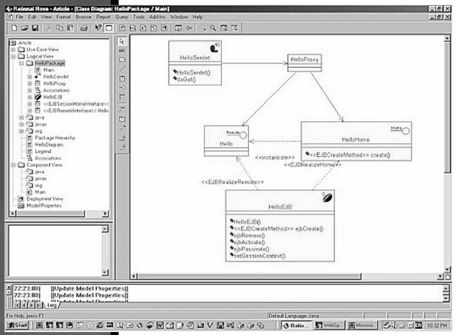
Participants

Handler defines an interface for handling requests.
(optional) implements the successor link.

Borland®

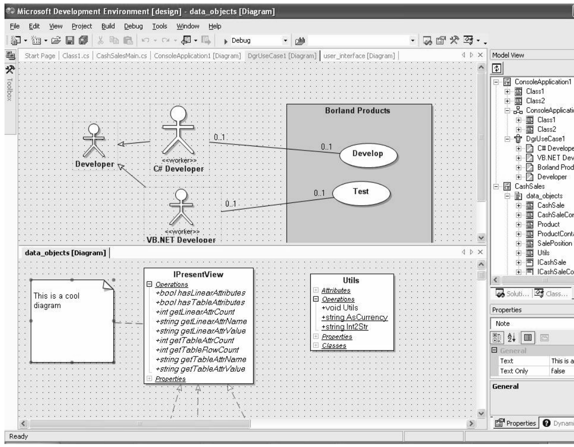
支持 XMI 汇入与汇出

IBM/Rational

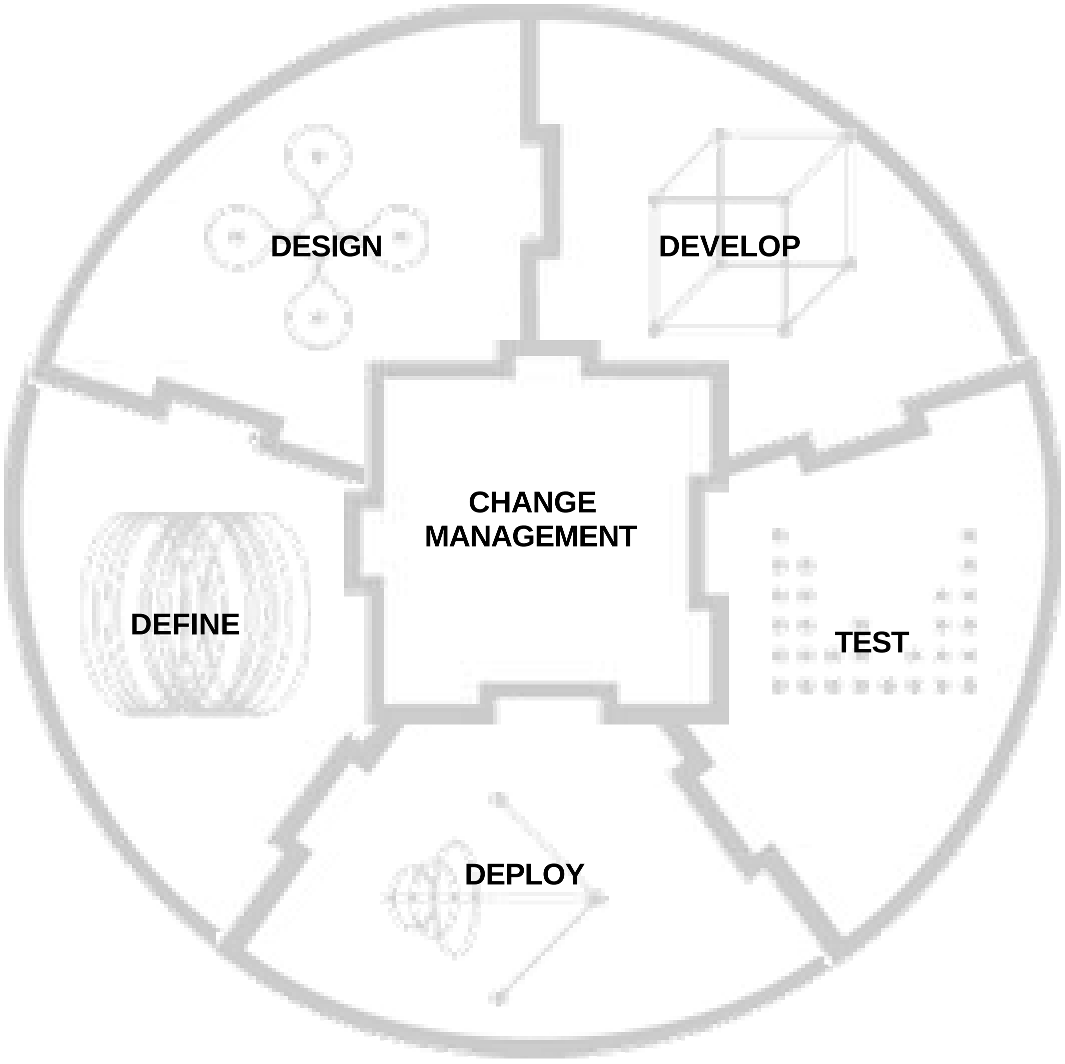

Rational Rose - Article - [Class Diagram: HelloPackage / Main]
HelloServlet
HelloProxy
Hello
HelloHome
HelloEJB
<<instantiate>>
<<EJBRealizeHome>>
<<EJBRealizeRemote>>
<<EJBCreateMethod>> create()
[Update Model Properties]

XMI？？？？

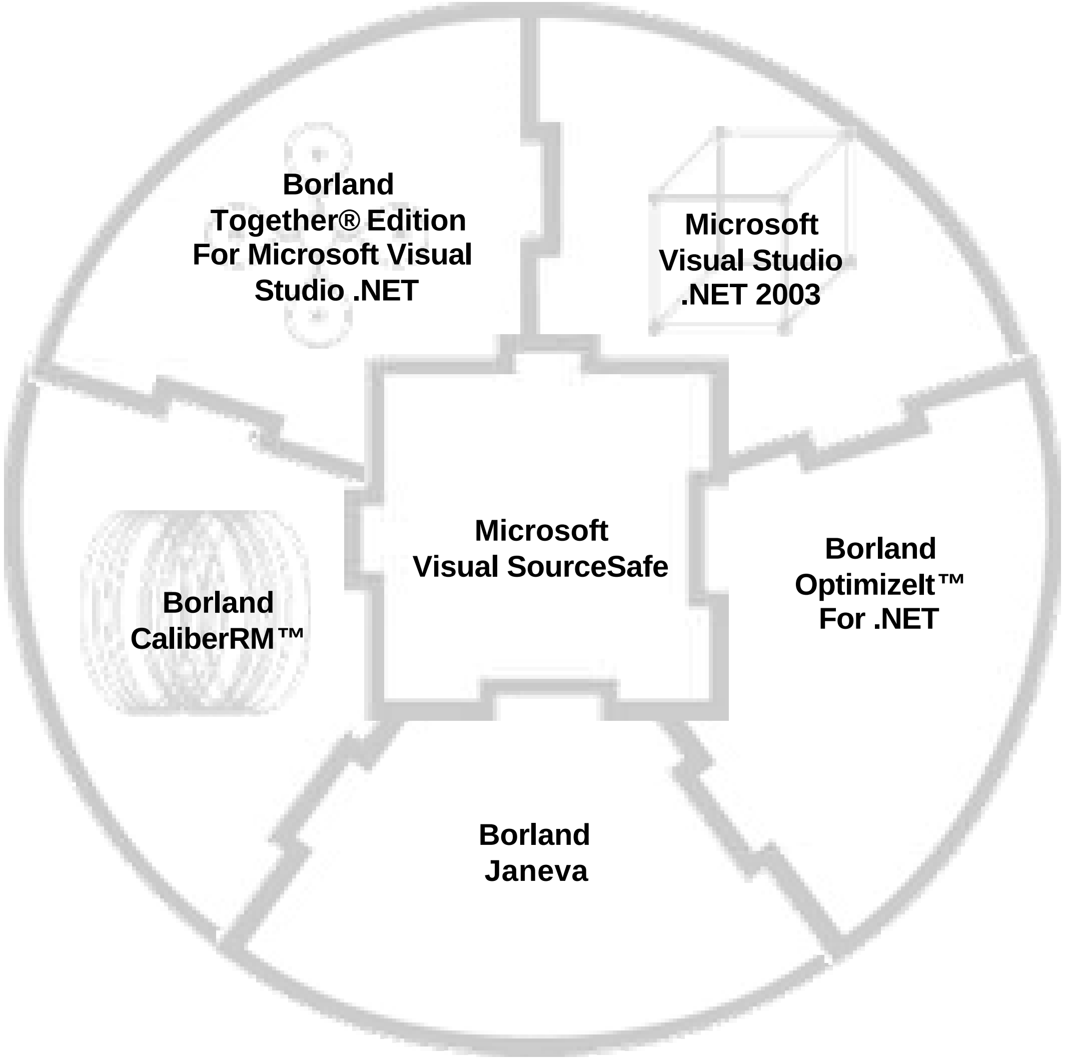
Borland Together



Borland ALM ? ? ? ? - .NET



Borland ALM ? ? ? ? - .NET



总结

Learn UML

Follow a process

Try a modeling tools

Find supports

- Time,
- Budget,
- Mentoring

May The Power

Be With You !!

Resources

需搭配 Microsoft Visual Studio .NET 2003

试用版下载网址

http://www.borland.com/products/downloads/download_together.html

Thank You

Q&A