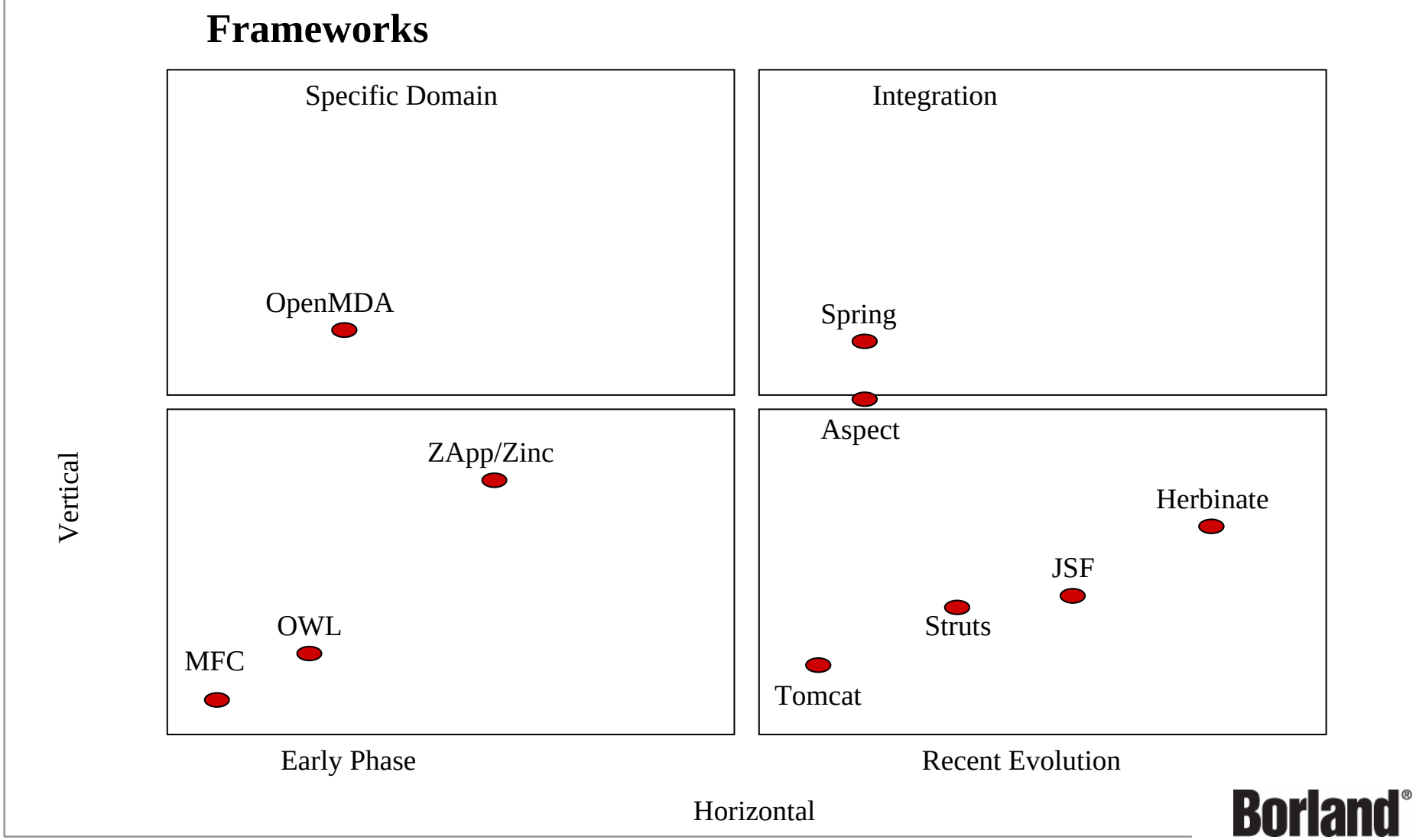
MAXIMIZE THE BUSINESS VALUE OF SOFTWARE

UML China/Together 2006

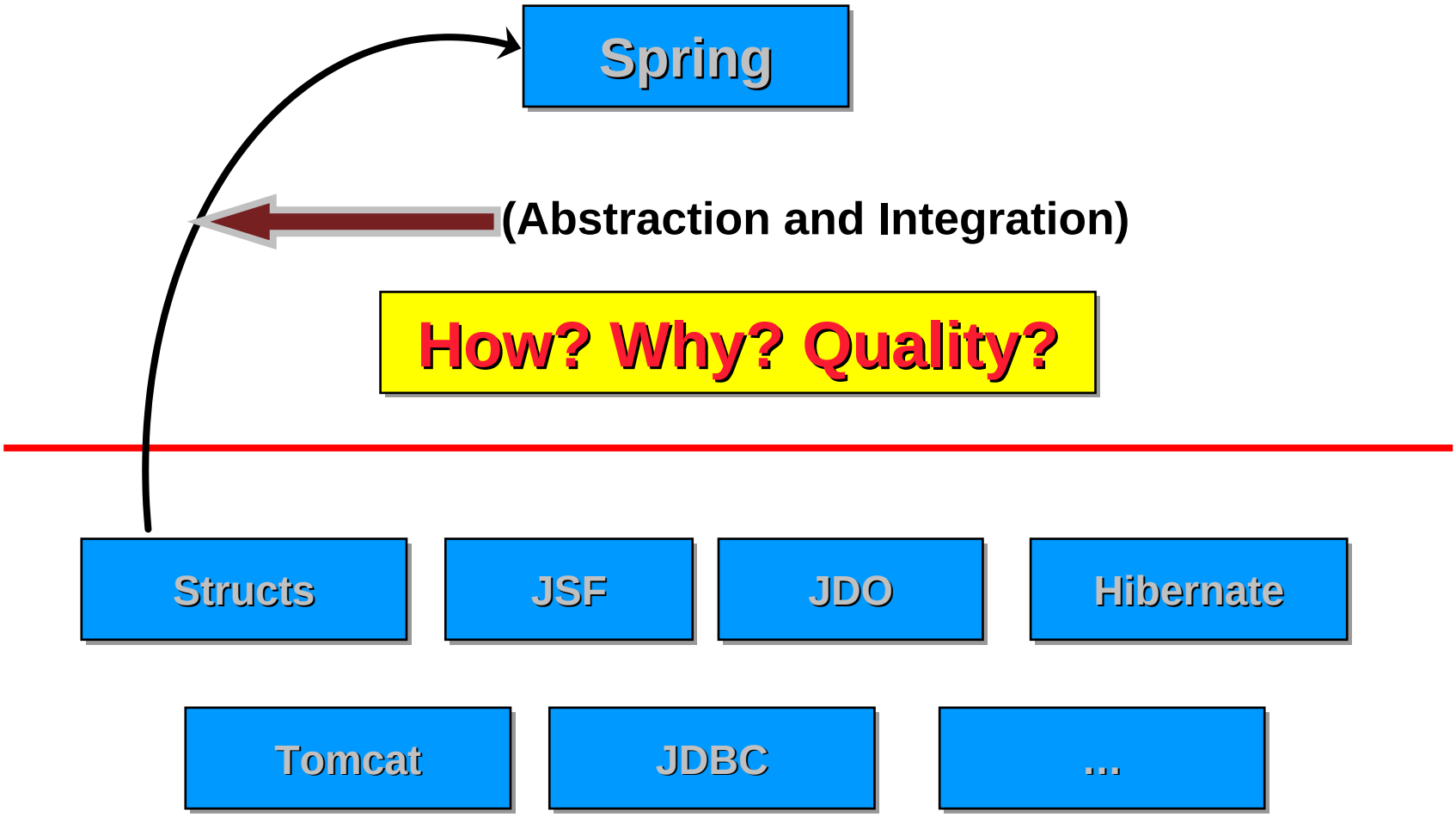
Borland®

掌握Open Source即掌握軟體發展趨勢



Borland®

掌握Open Source即掌握軟體發展趨勢



Borland®

Programming Quality Of Service

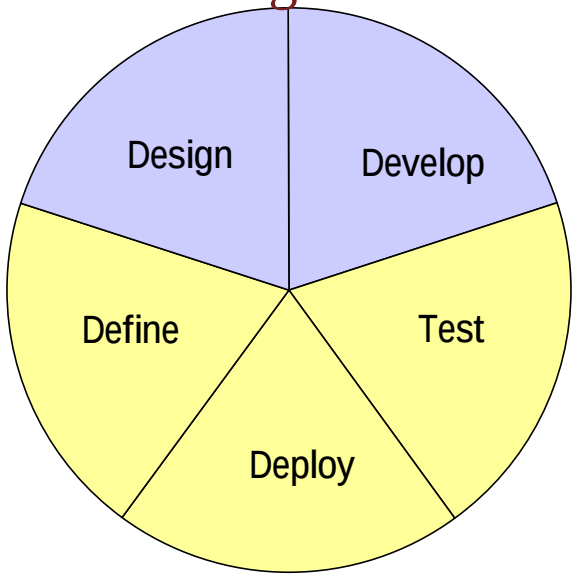
MDA Coverage

Business Process Modeling

Model Transformation

Application modeling

Source code generation



ALM Coverage

MDA

Requirements Management

Application Profiling

Documentation

Source code audits

System Metrics

- Source and model

Patterns management

- Optimal reuse

Change management

Source code management

Borland®

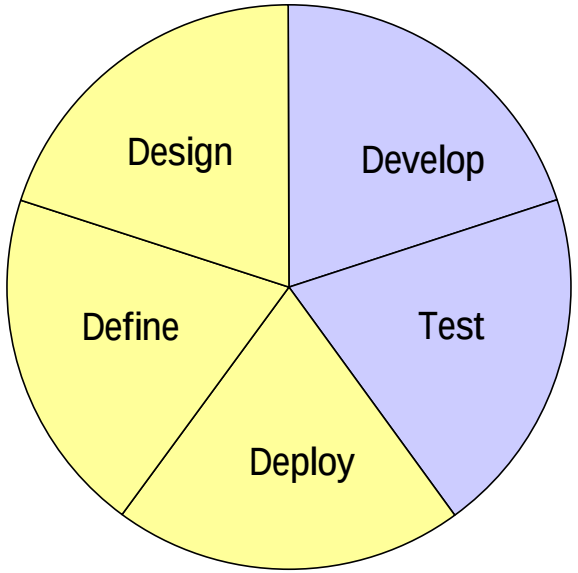
Programming Quality Of Service

XP Coverage

Code-Centric

Embrace Change

Testing-Oriented



XP Coverage

Lightweight

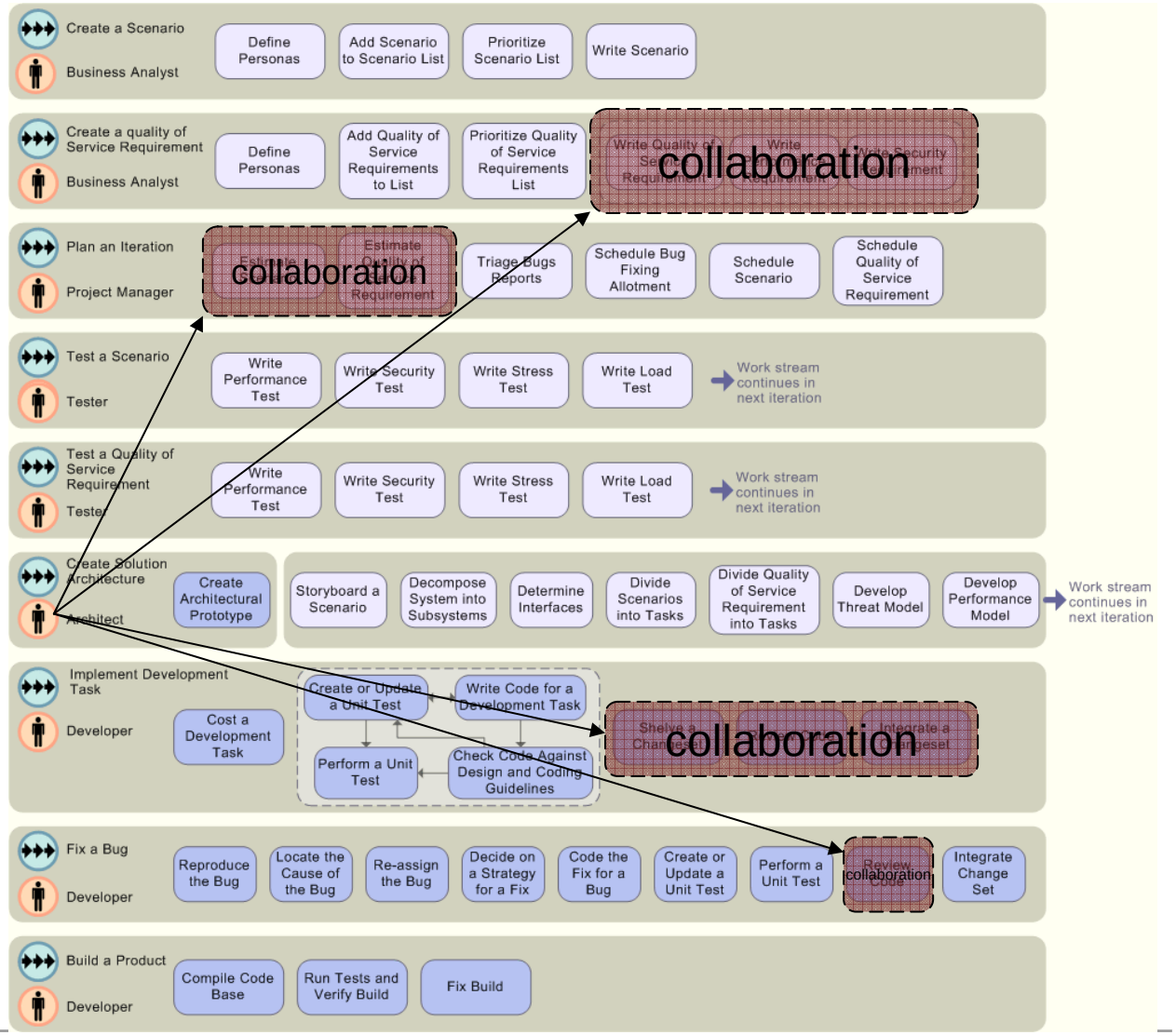
Embedded with IDEs

Peer Coding/Reviews

Unit Testing

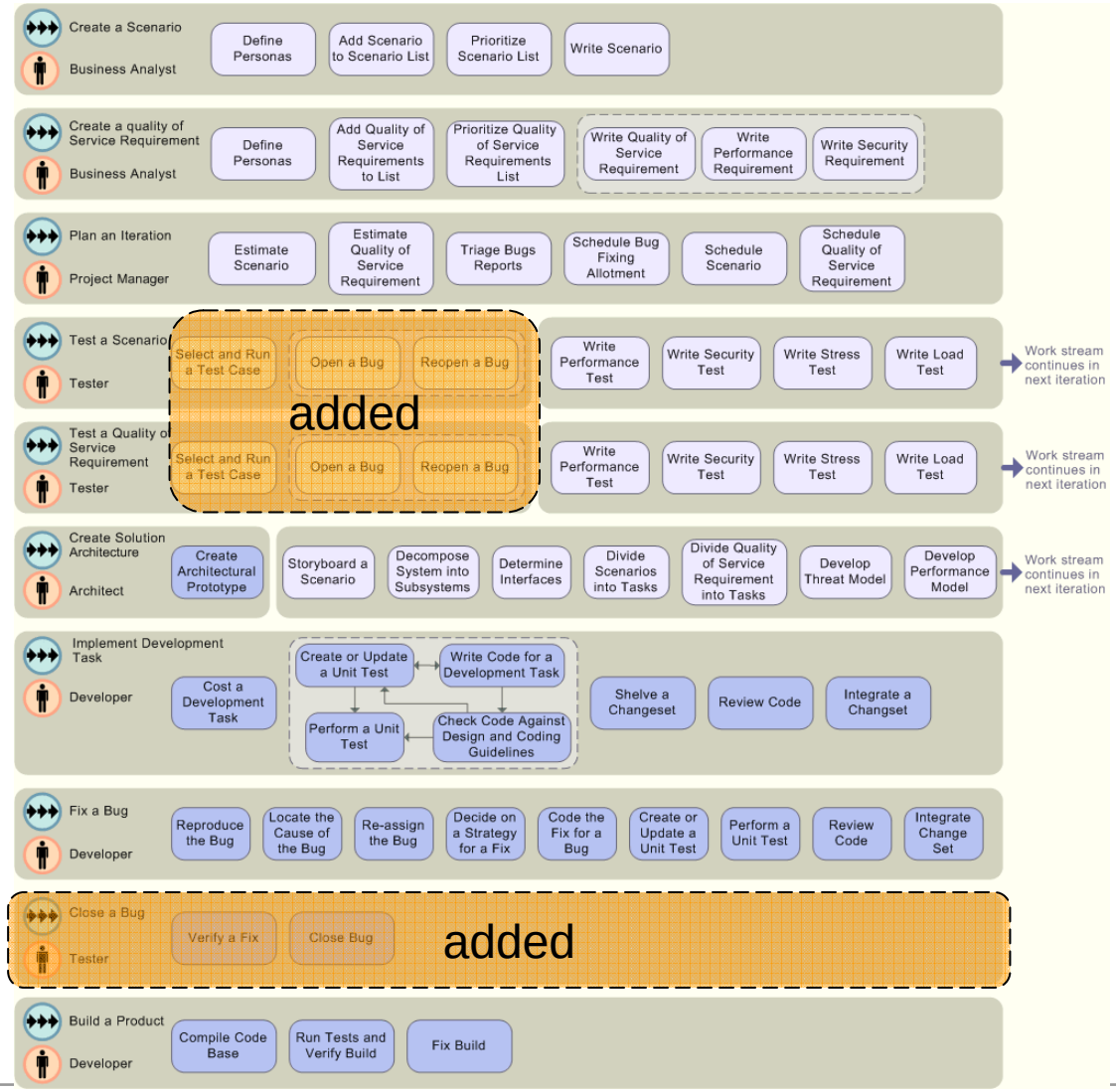
Borland®

Programming Quality Of Service



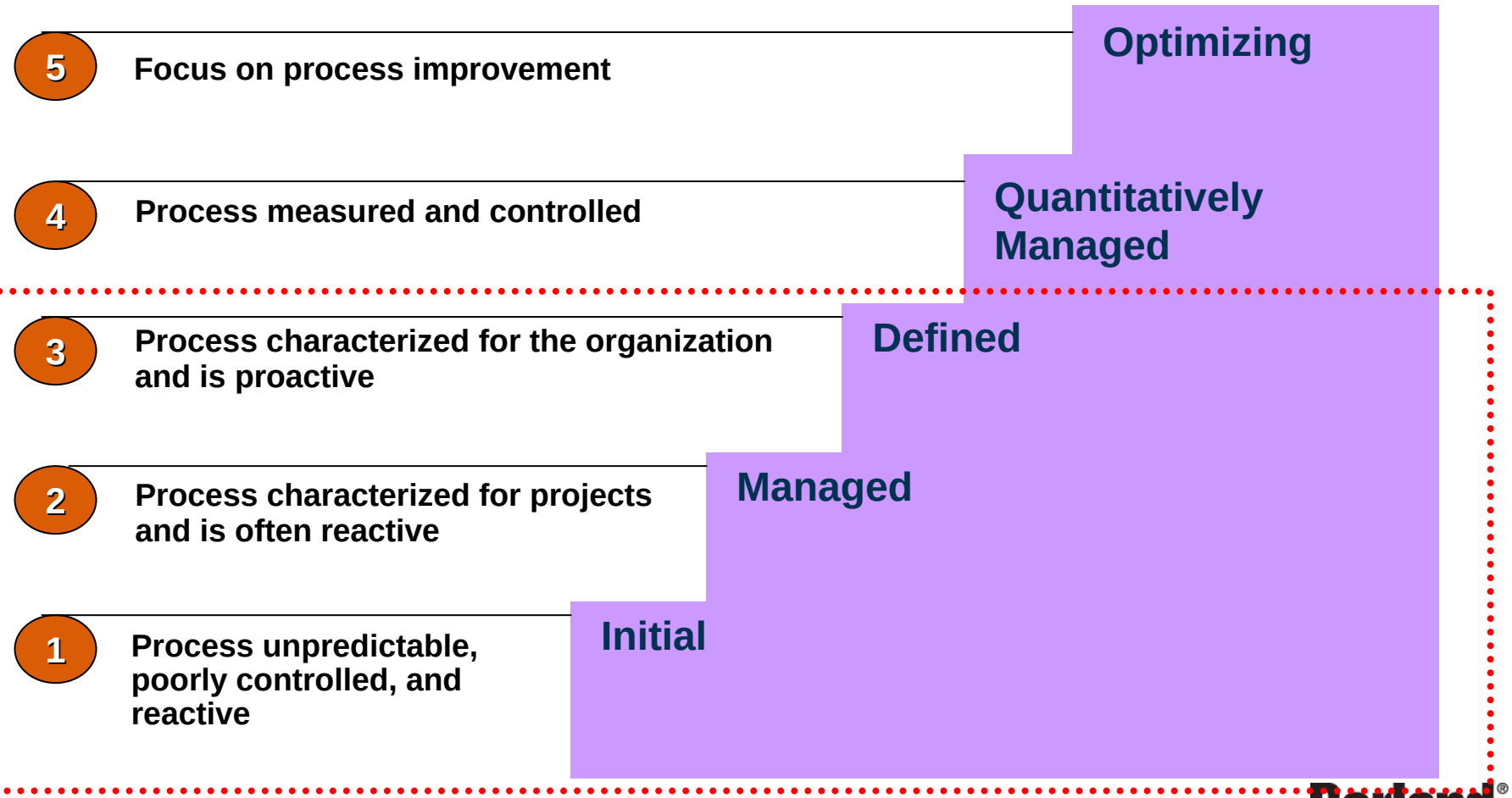
Borland®

Programming Quality Of Service



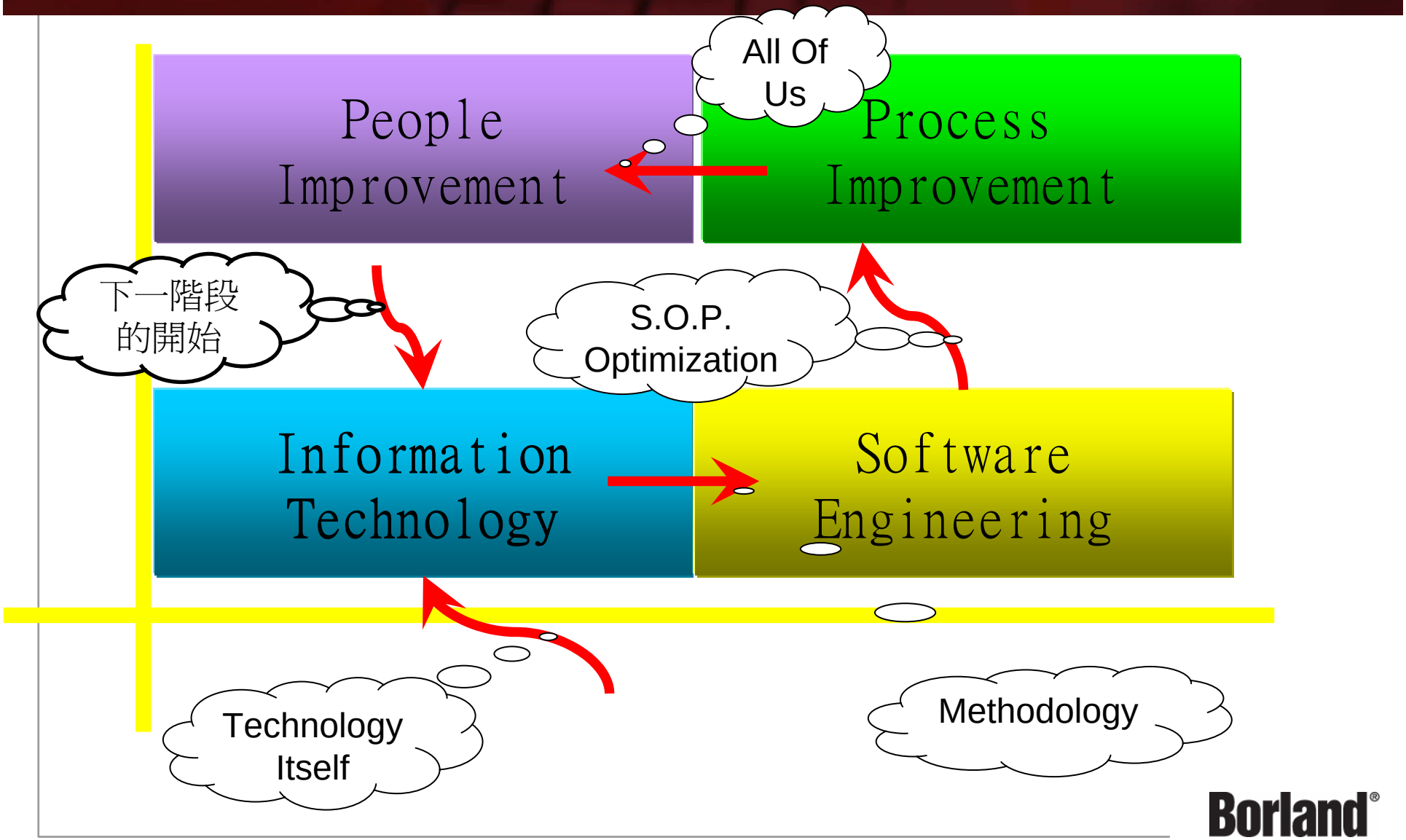
Borland®

Organizational Maturity Levels



Borland®

掌握Open Source即掌握軟體發展趨勢



Borland®

Programming Quality Of Service

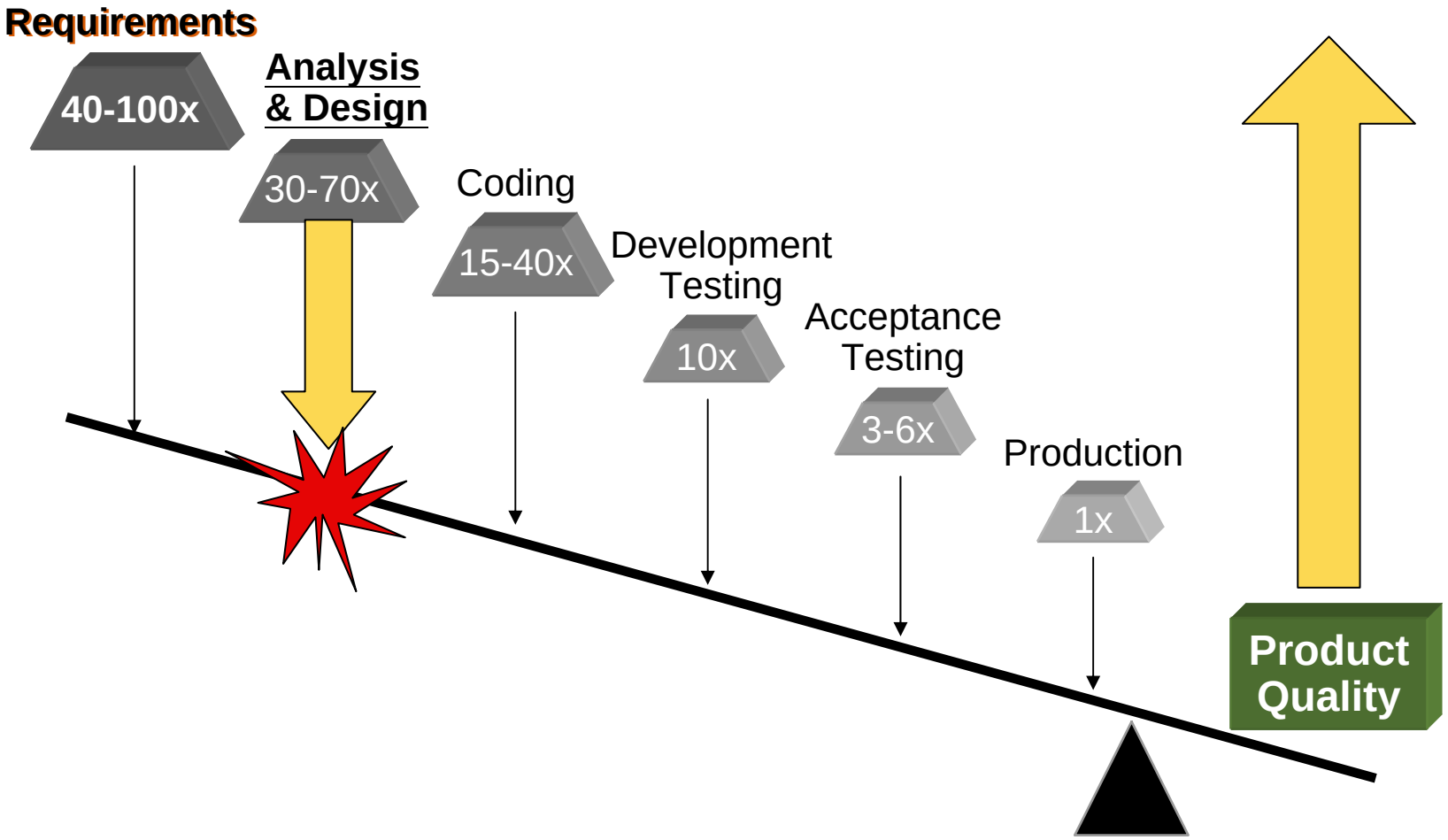
- Technologies, Programming Languages, Frameworks, Methodologies, Processes,这些都是做什么的?
  - Productivity?
  - Performance?
  - Stability?
  - Scalability?
  - Quality

Borland®

请试着回答下面的问题!

- 设计是否根据需求而来? 哪一个需求?
- 改变需求会影响什么?
- 某某系统这次上线的程序代码是哪些版本?上一次呢?
- 设计模型一定是正确的吗?
  - 如何验证，测试设计模型?
- 不同设计模型如何相互转换?
- 开发出来的软件品质如何衡量呢?
- 现在的你/你的团队最有把握掌握品质的开发阶段是那一个?

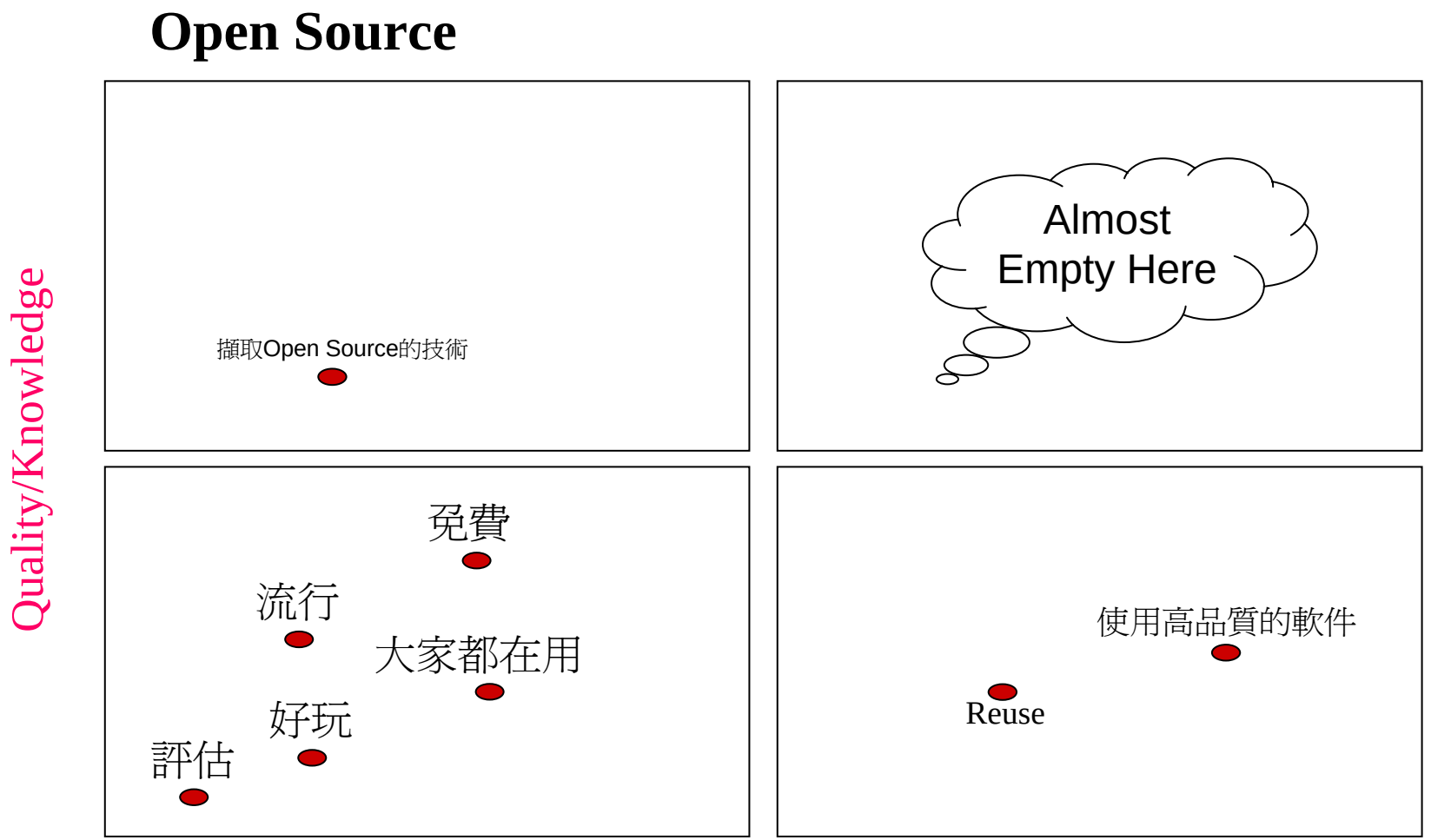
掌握Open Source即掌握軟體發展趨勢



- This data from Boehm: Software Engineering Economics

Borland®

Open Source的好處和陷阱



Borland®

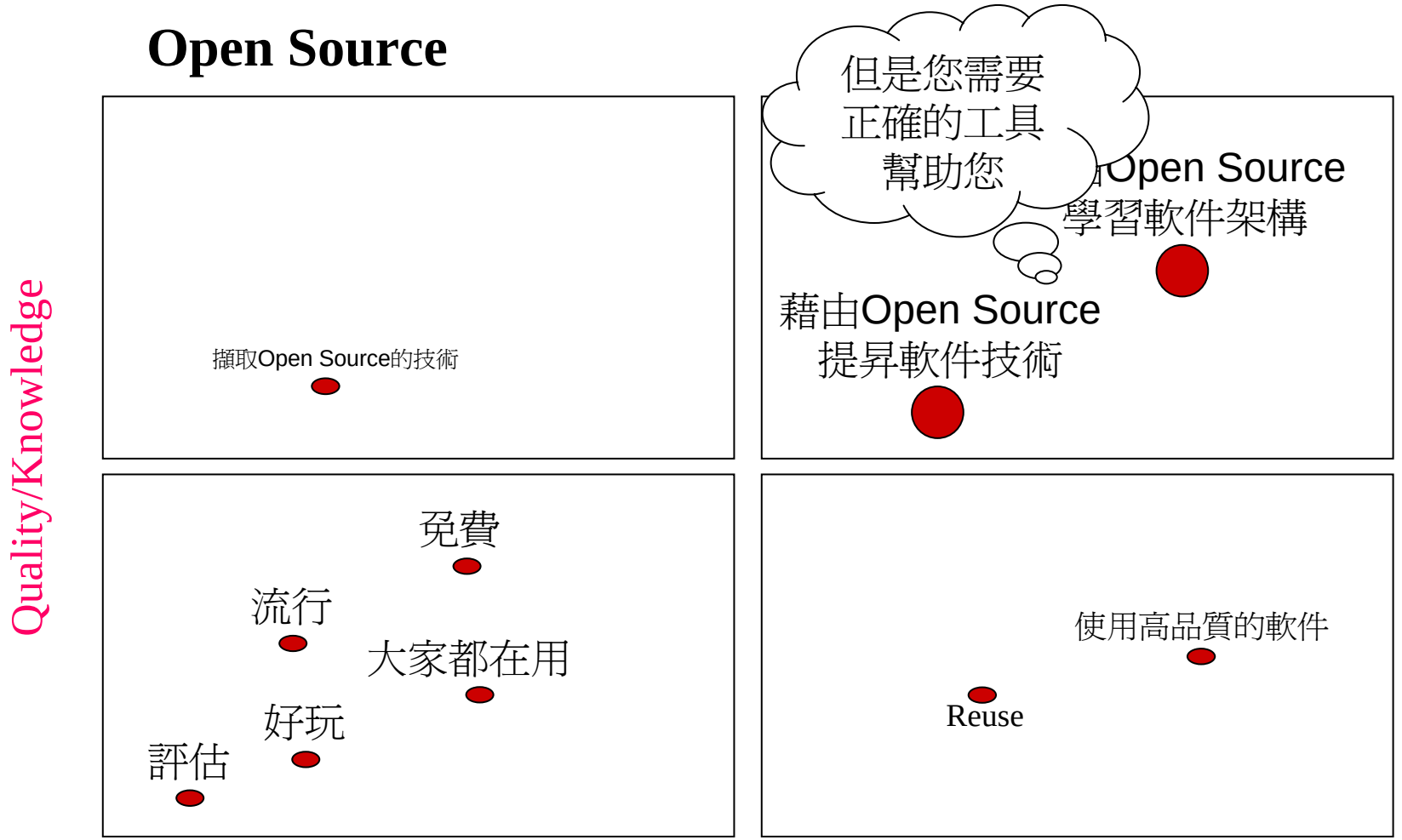
Open Source的好處和陷阱

請試著回答下面的問題?

- A：所有的Open Source都很好! Yes or No?
- B：我很瞭解我使用的Open Source的架構! Yes or No?
- C：Open Source的實作程式碼都沒問題! Yes or No?
- D：我的項目架構和Open Source的架構不會雞同鴨講! Yes or No?

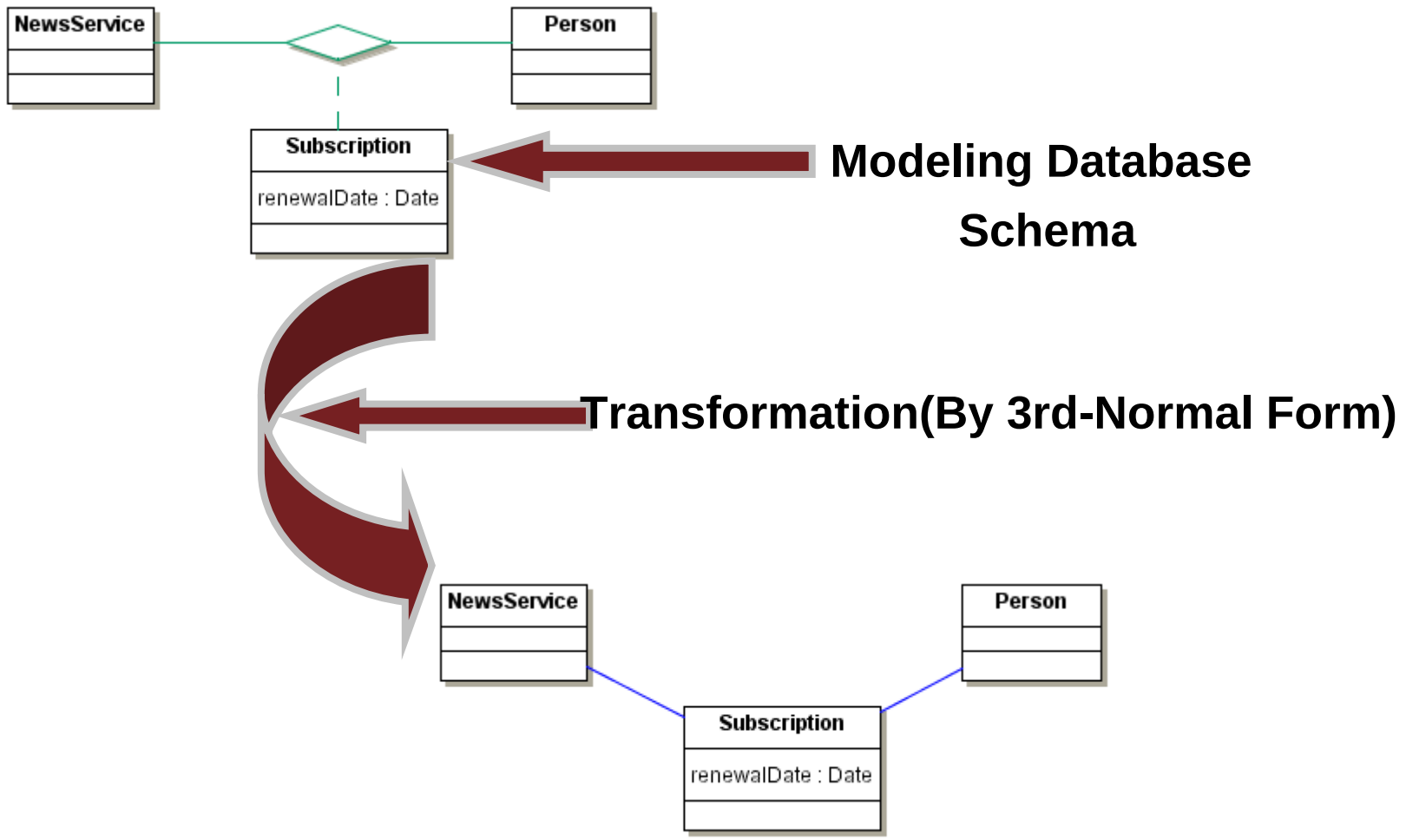
Borland®

Open Source的好處和陷阱



Borland®

Modeling Is Everywhere



Borland®

Modeling Is Everywhere

```
int countLetters(List<List<String>> doc) {
  int count = 0;
  for (Iterator<List<String>>i = doc.iterator();
       i.hasNext();) {
    List<String>line = i.next();
    for (Iterator<String>j = line.iterator();
         j.hasNext();) {
      String word = j.next();
      count += word.length();
    }
  }
  return count;
}
```

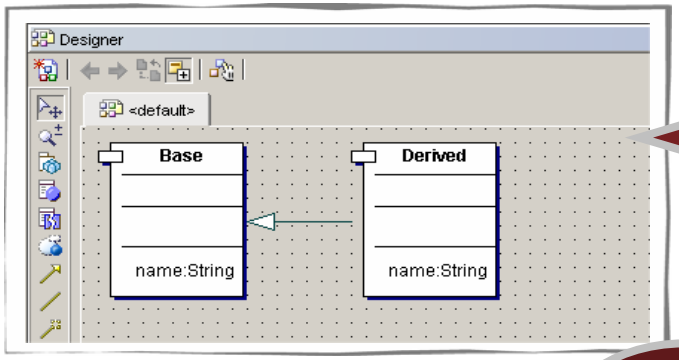
Modeling Your Code!

Transformation(By Refactoring)

```
int countLetters(List<List<String>> doc) {
  int cont = 0;
  for (List<String>> line : doc) {
    for (String word : line) {
      count += word.length();
    }
  }
}
```

Borland®

Modeling Is Everywhere



Modeling Your Code!

Transformation(QVT)

```
/* Generated by Together */

#ifndef DERIVED_H
#define DERIVED_H
#include "Base.h"
class Derived : public Base {
public:

        /**
         * @mod__returnType String
         */
        String getName();
};
#endif //DERIVED_H
```

Modeling Is Everywhere

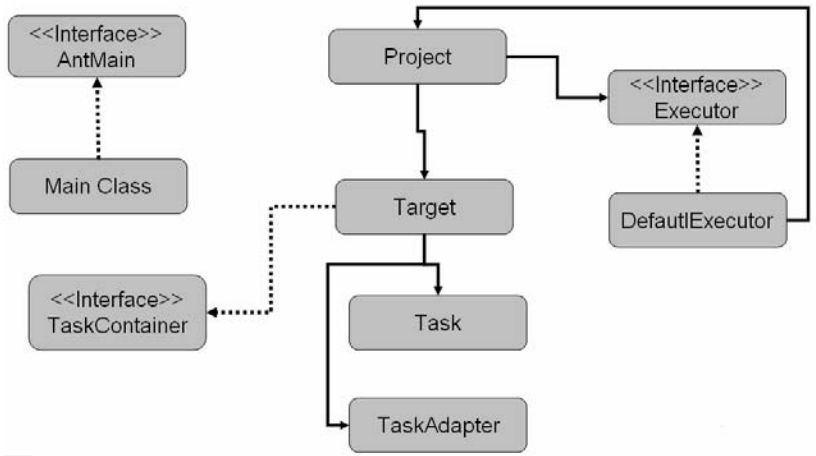
```
public static void main(String[] args) {
        try {
            Launcher launcher = new
Launcher();
            launcher.run(args);
        } catch (LaunchException e) {

System.err.println(e.getMessage());
        } catch (Throwable t) {
            t.printStackTrace();
        }
    }
```

Ant Framework

２０５９４９行程式碼

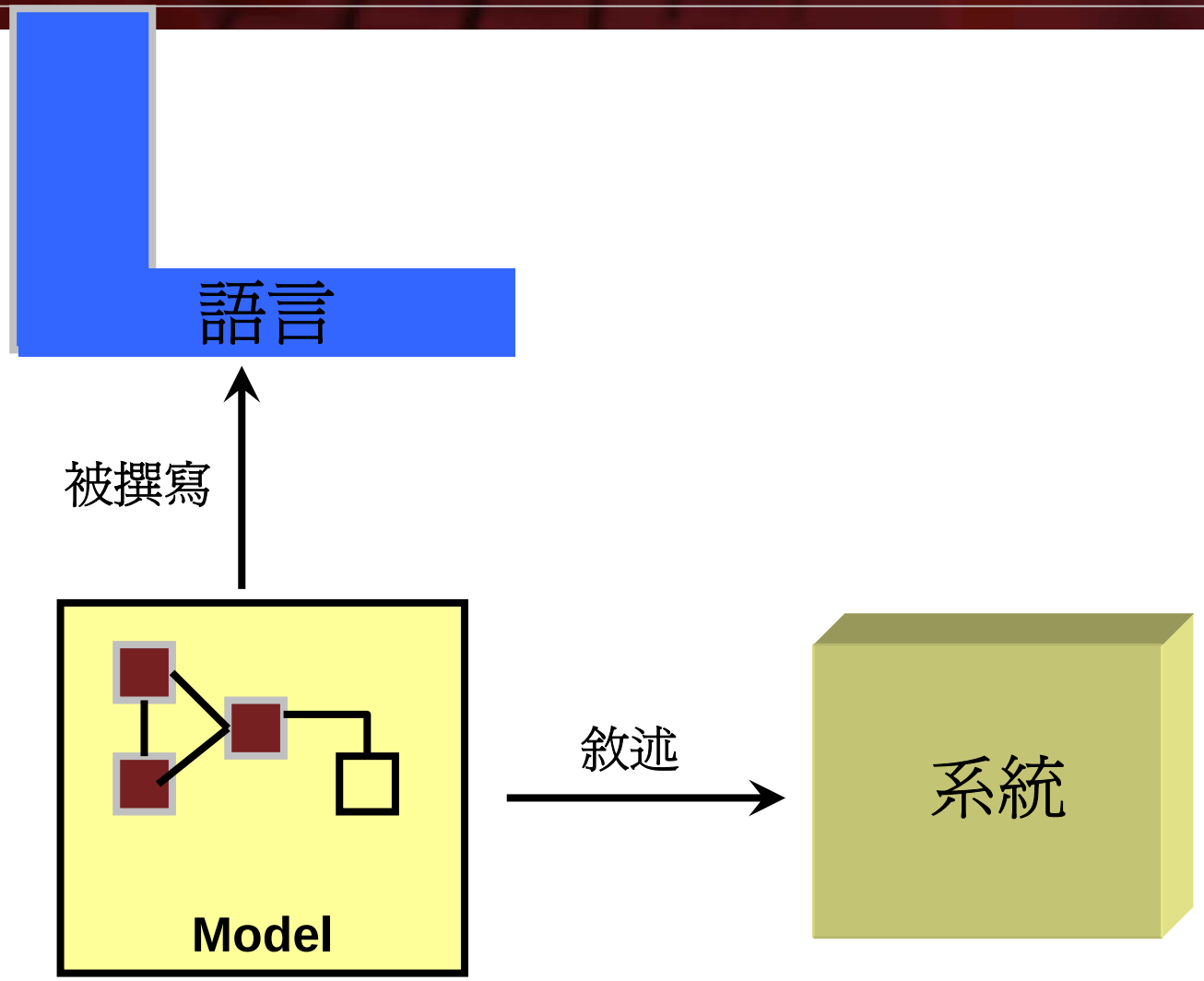
Together

Reverse Engineering



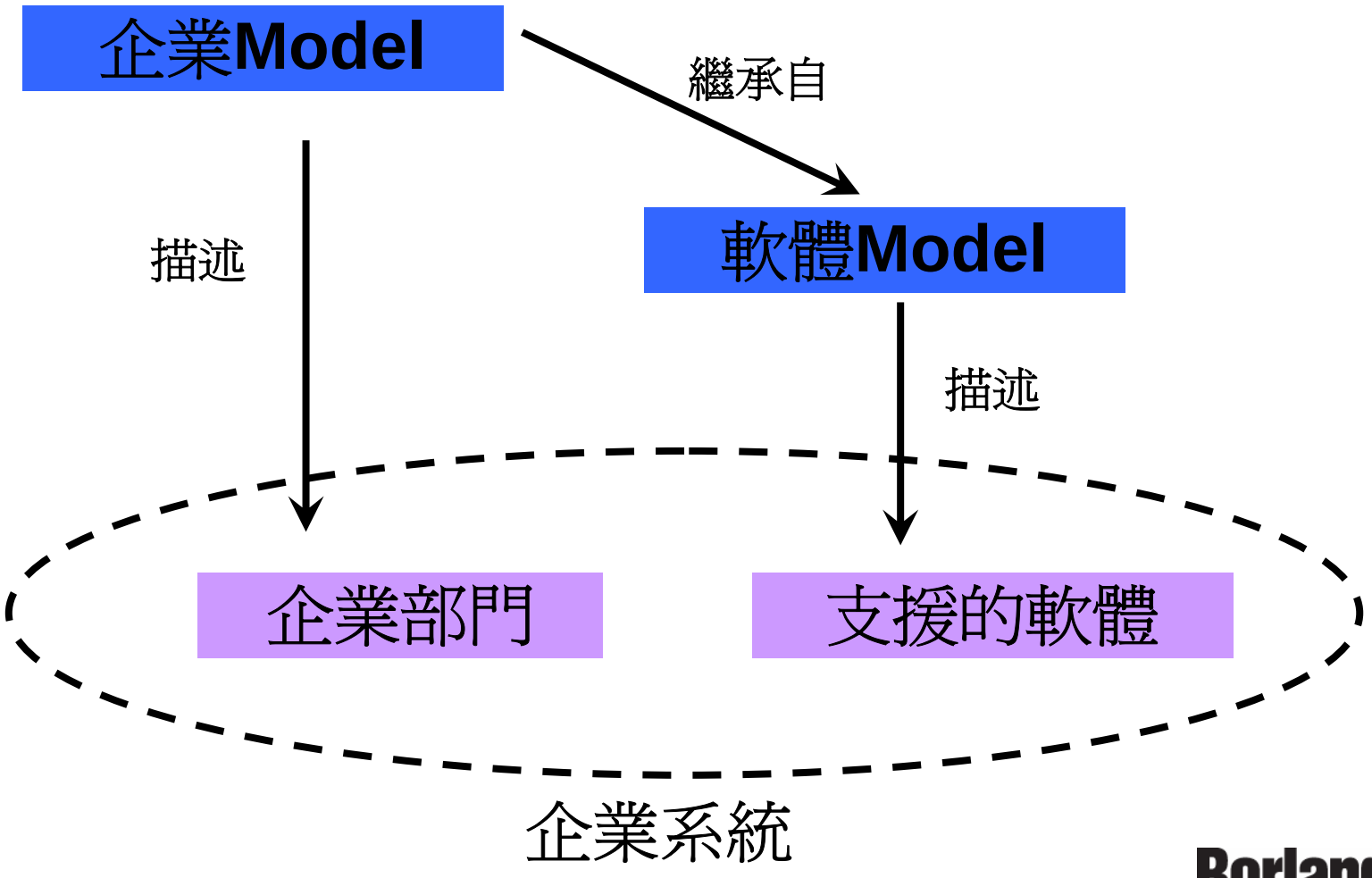
Borland®

Models和語言

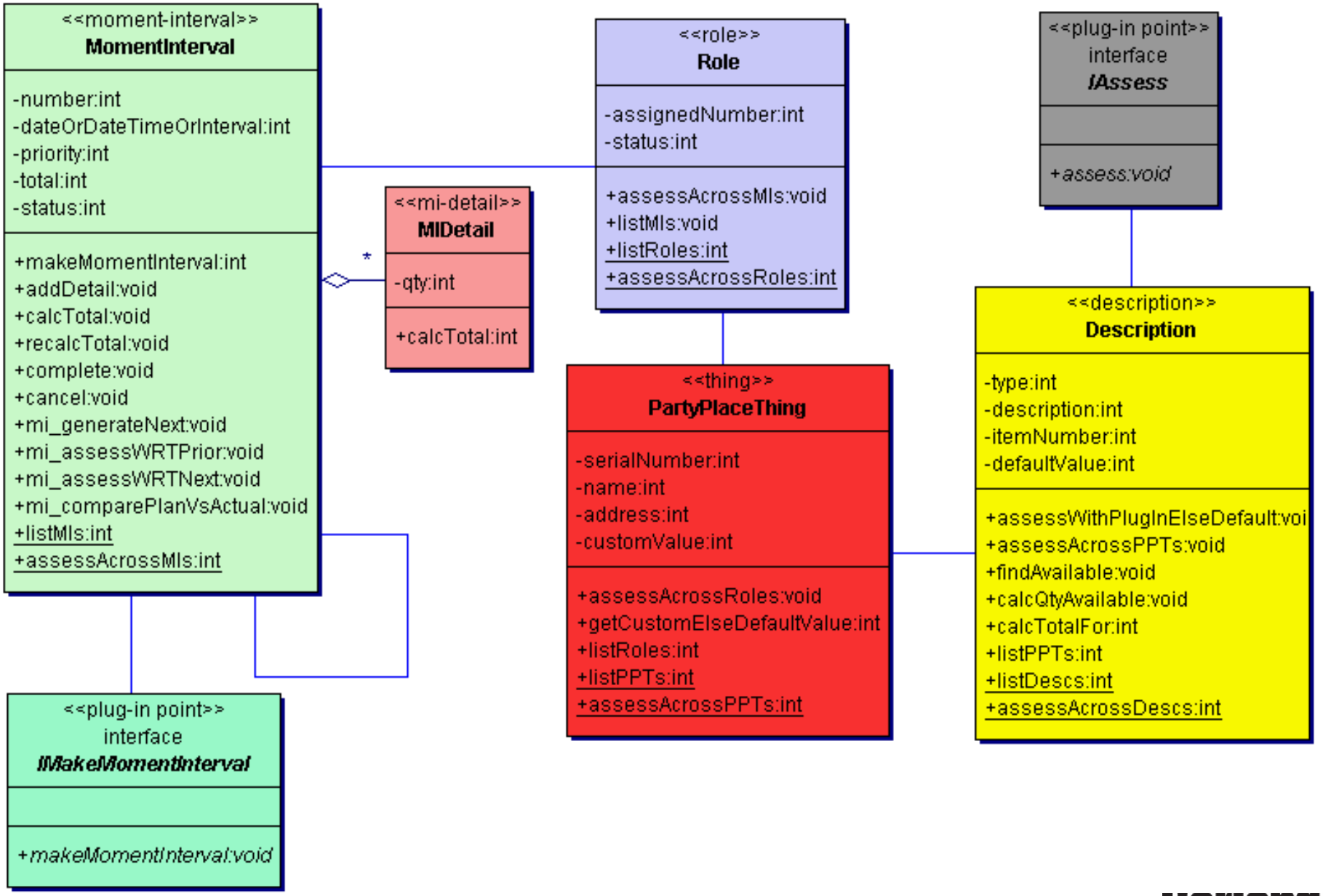


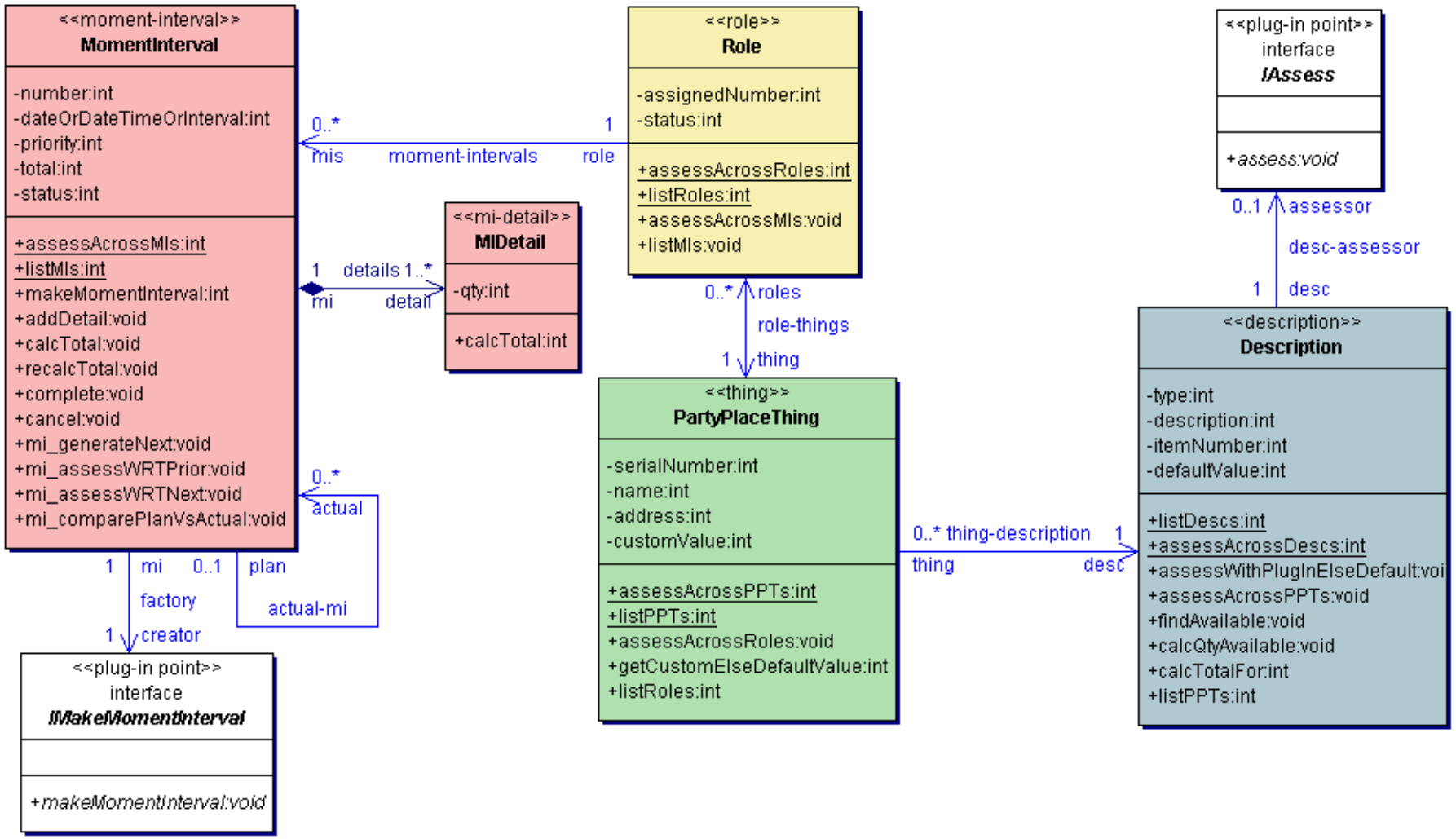
Borland®

各種Models

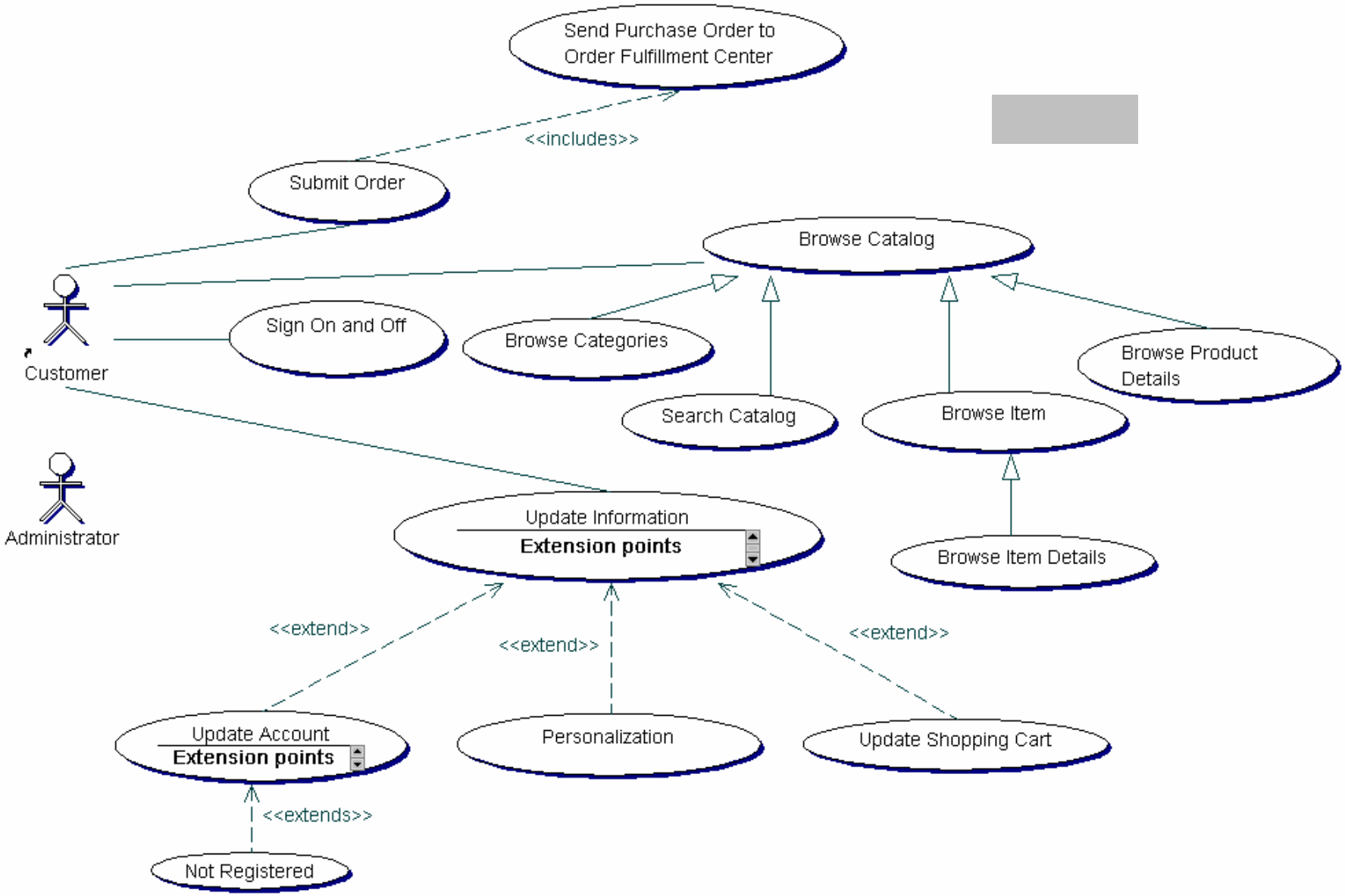


Borland®

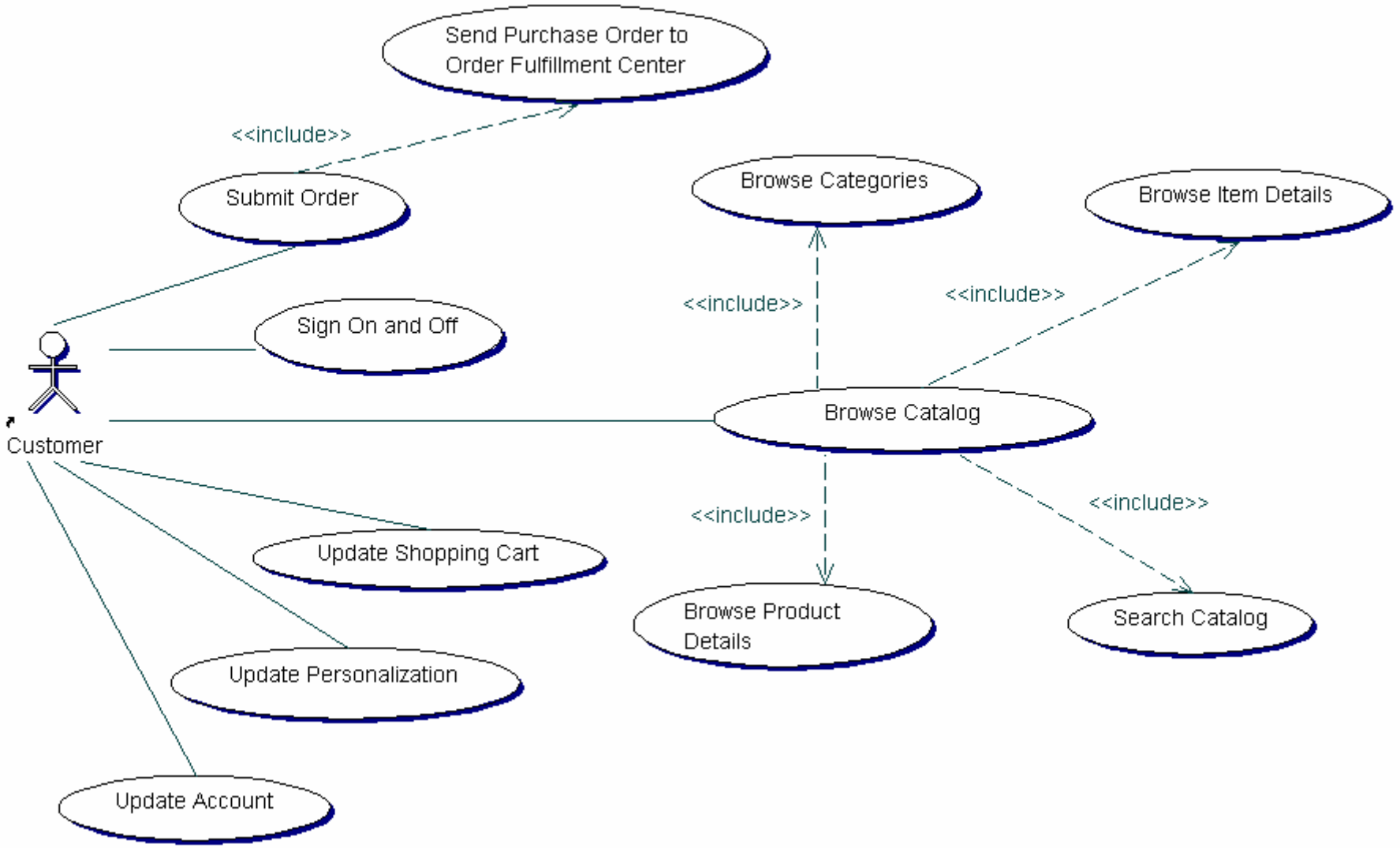

<<moment-interval>>
MomentInterval
-number:int
-dateOrDateTimeOrInterval:int
-priority:int
-total:int
-status:int
+makeMomentInterval:int
+addDetail:void
+calcTotal:void
+recalcTotal:void
+complete:void
+cancel:void
+mi_generateNext:void
+mi_assessWRTPrior:void
+mi_assessWRTNext:void
+mi_comparePlanVsActual:void
+listMIs:int
+assessAcrossMIs:int
<<mi-detail>>
MIDetail
-qty:int
+calcTotal:int
*
<<role>>
Role
-assignedNumber:int
-status:int
+assessAcrossMIs:void
+listMIs:void
+listRoles:int
+assessAcrossRoles:int
<<plug-in point>>
interface
IAssess
+assess:void
<<thing>>
PartyPlaceThing
-serialNumber:int
-name:int
-address:int
-customValue:int
+assessAcrossRoles:void
+getCustomElseDefaultValue:int
+listRoles:int
+listPPTs:int
+assessAcrossPPTs:int
<<description>>
Description
-type:int
-description:int
-itemNumber:int
-defaultValue:int
+assessWithPlugInElseDefault:voi
+assessAcrossPPTs:void
+findAvailable:void
+calcQtyAvailable:void
+calcTotalFor:int
+listPPTs:int
+listDescs:int
+assessAcrossDescs:int
<<plug-in point>>
interface
IMakeMomentInterval
+makeMomentInterval:void


<<moment-interval>>
MomentInterval
-number:int
-dateOrDateTimeOrInterval:int
-priority:int
-total:int
-status:int
+assessAcrossMIs:int
+listMIs:int
+makeMomentInterval:int
+addDetail:void
+calcTotal:void
+recalcTotal:void
+complete:void
+cancel:void
+mi_generateNext:void
+mi_assessWRTPrior:void
+mi_assessWRTNext:void
+mi_comparePlanVsActual:void
<<role>>
Role
-assignedNumber:int
-status:int
+assessAcrossRoles:int
+listRoles:int
+assessAcrossMIs:void
+listMIs:void
<<plug-in point>>
interface
IAssess
+assess:void
<<mi-detail>>
MIDetail
-qty:int
+calcTotal:int
<<thing>>
PartyPlaceThing
-serialNumber:int
-name:int
-address:int
-customValue:int
+assessAcrossPPTs:int
+listPPTs:int
+assessAcrossRoles:void
+getCustomElseDefaultValue:int
+listRoles:int
<<description>>
Description
-type:int
-description:int
-itemNumber:int
-defaultValue:int
+listDescs:int
+assessAcrossDescs:int
+assessWithPlugInElseDefault:voi
+assessAcrossPPTs:void
+findAvailable:void
+calcQtyAvailable:void
+calcTotalFor:int
+listPPTs:int
<<plug-in point>>
interface
IMakeMomentInterval
+makeMomentInterval:void
0..*
mis
moment-intervals
1
role
1
details
1..*
mi
detail
0..*
roles
role-things
1
thing
0..*
actual
1
mi
0..1
plan
factory
actual-mi
1
creator
0..* thing-description
thing
1
desc
0..1
assessor
desc-assessor
1
desc

Borland®


Send Purchase Order to
Order Fulfillment Center
<<includes>>
Submit Order
Browse Catalog
Customer
Sign On and Off
Browse Categories
Search Catalog
Browse Item
Browse Product
Details
Administrator
Update Information
Extension points
Browse Item Details
<<extend>>
<<extend>>
<<extend>>
Update Account
Extension points
Personalization
Update Shopping Cart
<<extends>>
Not Registered

Borland®


Send Purchase Order to
Order Fulfillment Center
<<include>>
Submit Order
Browse Categories
Browse Item Details
<<include>>
<<include>>
Sign On and Off
Customer
Browse Catalog
<<include>>
<<include>>
Update Shopping Cart
Browse Product
Details
Search Catalog
Update Personalization
Update Account

Sample Audits

Avoid Aggregation, Favor Composition
Avoid Dangling Model Elements
Always Indicate Multiplicity
Always Indicate Navigability
Avoid Multiplicities Involving Max and Mins
Avoid * Multiplicity
Always Name Associations
Avoid Using Dependencies
Do not Overlap Guards
Do not Use Disjoint Guards
Identifier Conflicts with Keyword
Indicate Role Name on Association Ends
Indicate Role Names on Recursive Associations
Lines Should Not Cross
Naming Conventions
Never Place Guard on Initial Transition
Provide Comment for OCL Constraints
Use Plural Names on Association Ends with Multiplicity > 1
Avoid Generalization Between Use Cases
Avoid Unassociated Actors
Avoid <<uses>>, <<includes>>, and <<extends>>
Avoid Weak Verbs at Beginning of Use Case
Avoid Association Classes
Abstract Class Declaration
Avoid Cyclic Dependencies Between Packages
Avoid N-ary Associations
Avoid Qualifiers
Always Specify Type on Attributes and Parameters
Class Should be Interface

Sample Audits

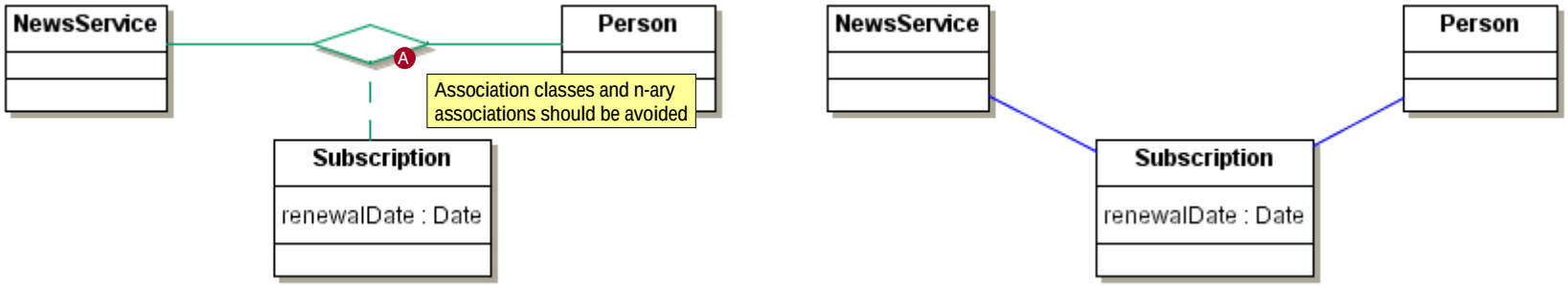
- Conflict With System Class
- Do not Model Elements of Implemented Interfaces
- Do not Model Scaffolding Code
- Do not Name Associations that have Association Classes
- Hiding Inherited Attribute
- Hiding Inherited Static Method
- List Static Operations/Attributes Before Instance Operations/Attributes
- Overriding Non-abstract Method with Abstract Method
- Subclasses have the Same Member
- Use Singular Names for Classes
- Avoid Modeling Destruction
- Avoid Modeling Return Arrows
- Avoid “Black Hole” States
- Avoid “Miracle” States
- Avoid Recursive Transitions With no Entry or Exit Actions
- Avoid “Black Hole” Activites
- Avoid “Miracle” Activities
- All Transitions Existing a Decision Must Have Guards
- Forks Should Have Only One Entry Transition
- Joins Should Have Only One Exit Transition
- Components Should only Depend on Interfaces

Borland®

Class Diagram Audits

Avoid Association Classes (AAC)

Association Classes can be decomposed into a separate class that associates two others. These may confuse generators, or be decomposed anyway.

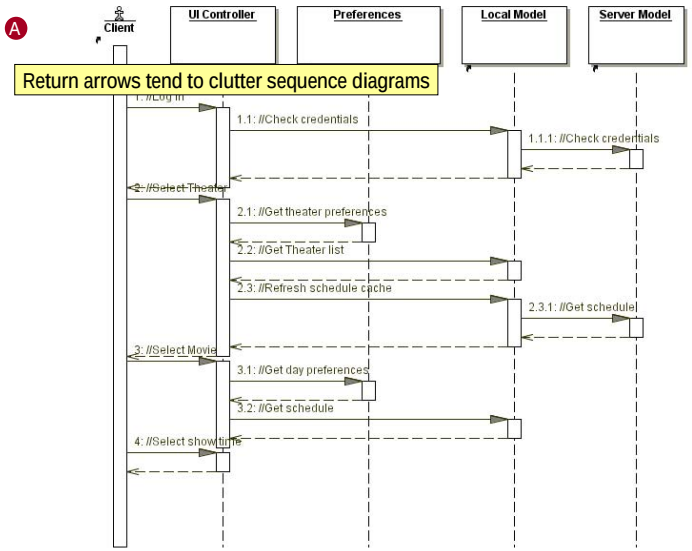


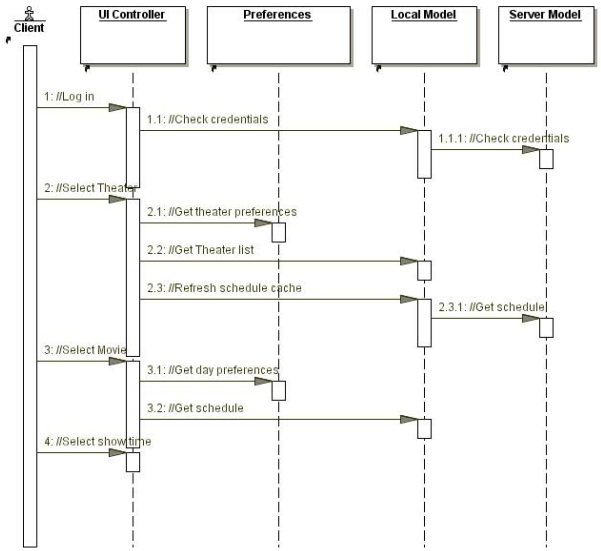
Borland®

Sequence Diagram Audits

Avoid Modeling Return Arrows (AMRA)

To reduce clutter on diagrams, the explicit modeling of return arrows is discouraged.



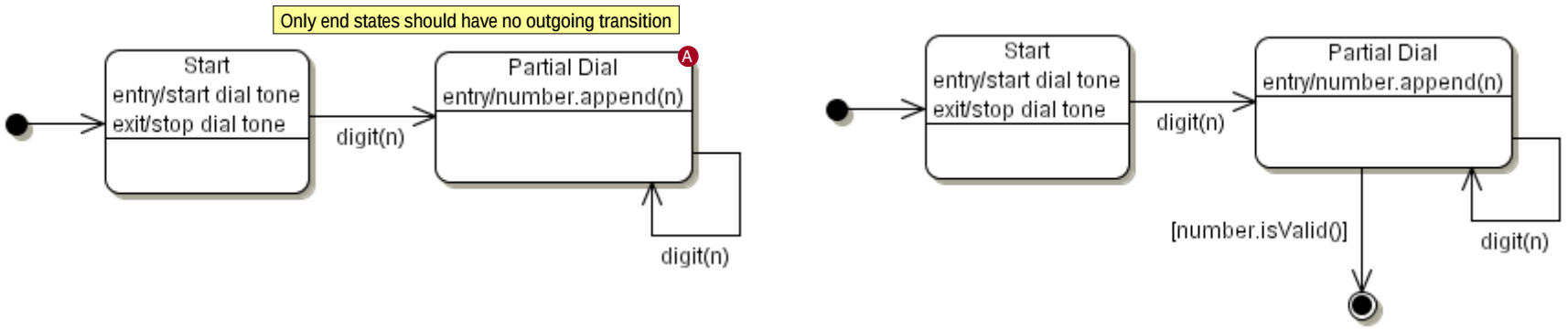


Borland®

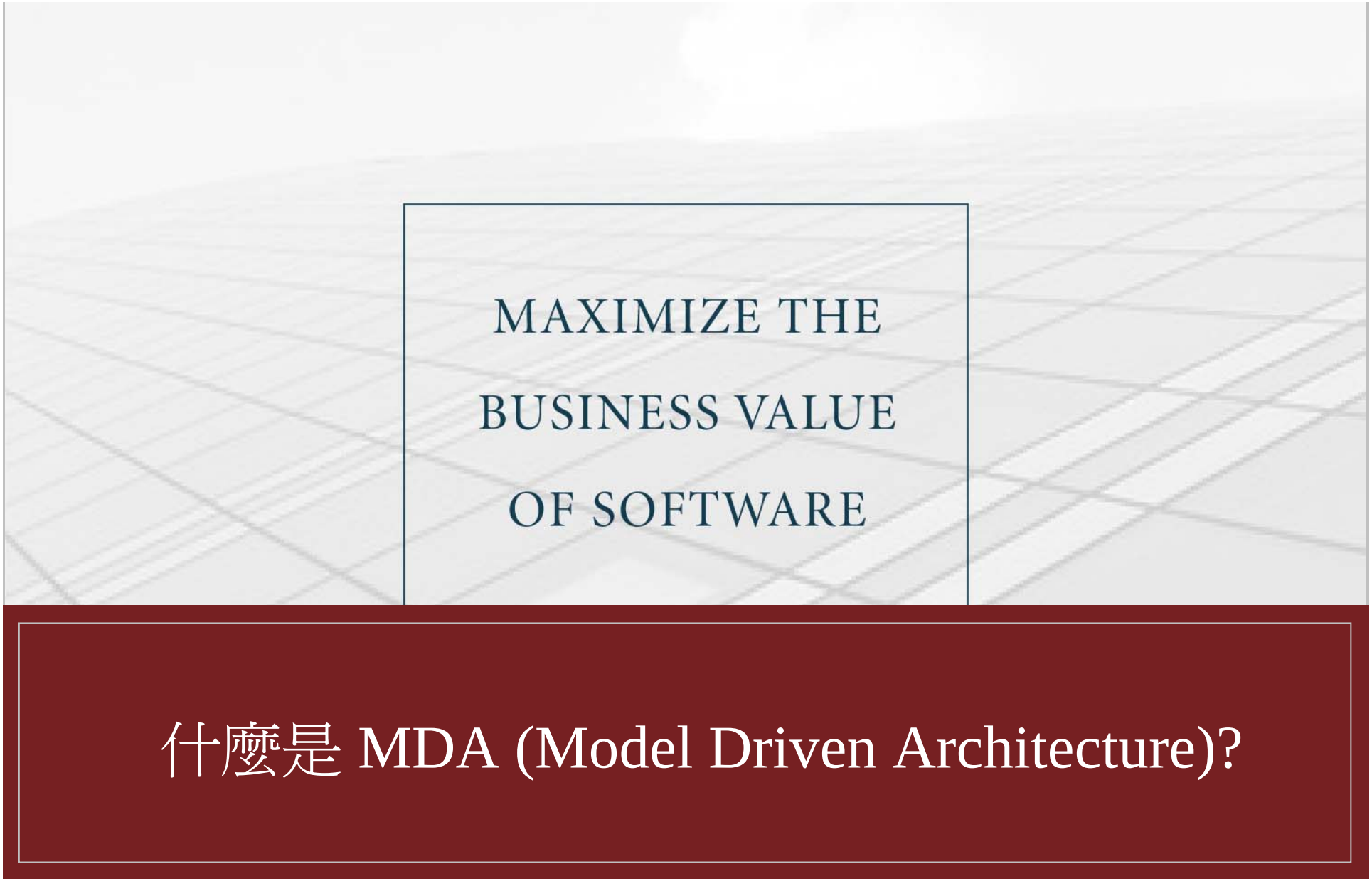
State Diagram Audits

Avoid “Black Hole” States (ABHS)

Only End states should have an incoming transition with no outgoing transition.



Borland®



什麼是 MDA (Model Driven Architecture)?

Borland®

Model Driven Architecture

MDA provides an approach for, and enables tools to be provided for:

- specifying a system independently of the platform that supports it,
- specifying platforms,
- choosing a particular platform for the system, and
- transforming the system specification into one for a particular platform.

Three primary goals of MDA

- Portability
- interoperability
- reusability

through architectural separation of concerns.

What Comprises MDA?

MDA is not a single specification, but a collection of related OMG specifications:

- Unified Modeling Language (UML™) 2.0
  - Infrastructure
  - Superstructure
  - Diagram Interchange
  - Profiles
- Object Constraint Language (OCL)
- Meta-Object Facility (MOF)
- XML Meta-Data Interchange (XMI)
- Common Warehouse Meta-model (CWM)
- Query View Transformation (QVT)

Borland®

MDA is not a Standard…yet

The MDA adopted Standards include:

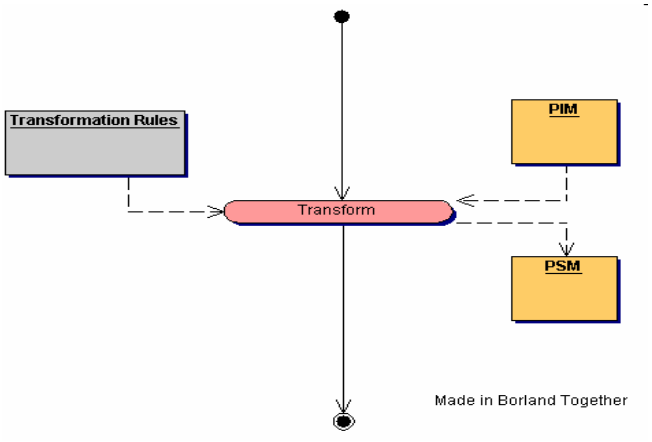
- UML, including OCL
- MOF, including JMI and XMI
- **QVT, which doesn't exist yet**
- And more: CWM, Diagram Interchange, and various domain specific models which play a role

No OMG test for MDA compliance

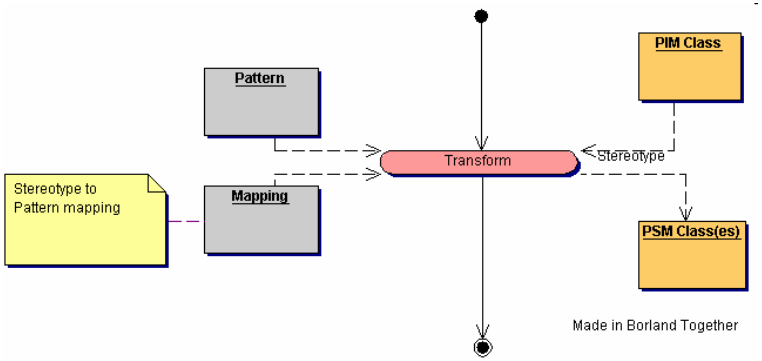
- This allows many to make loose claims

Borland®

MDA Transformation



Transformation of diagram



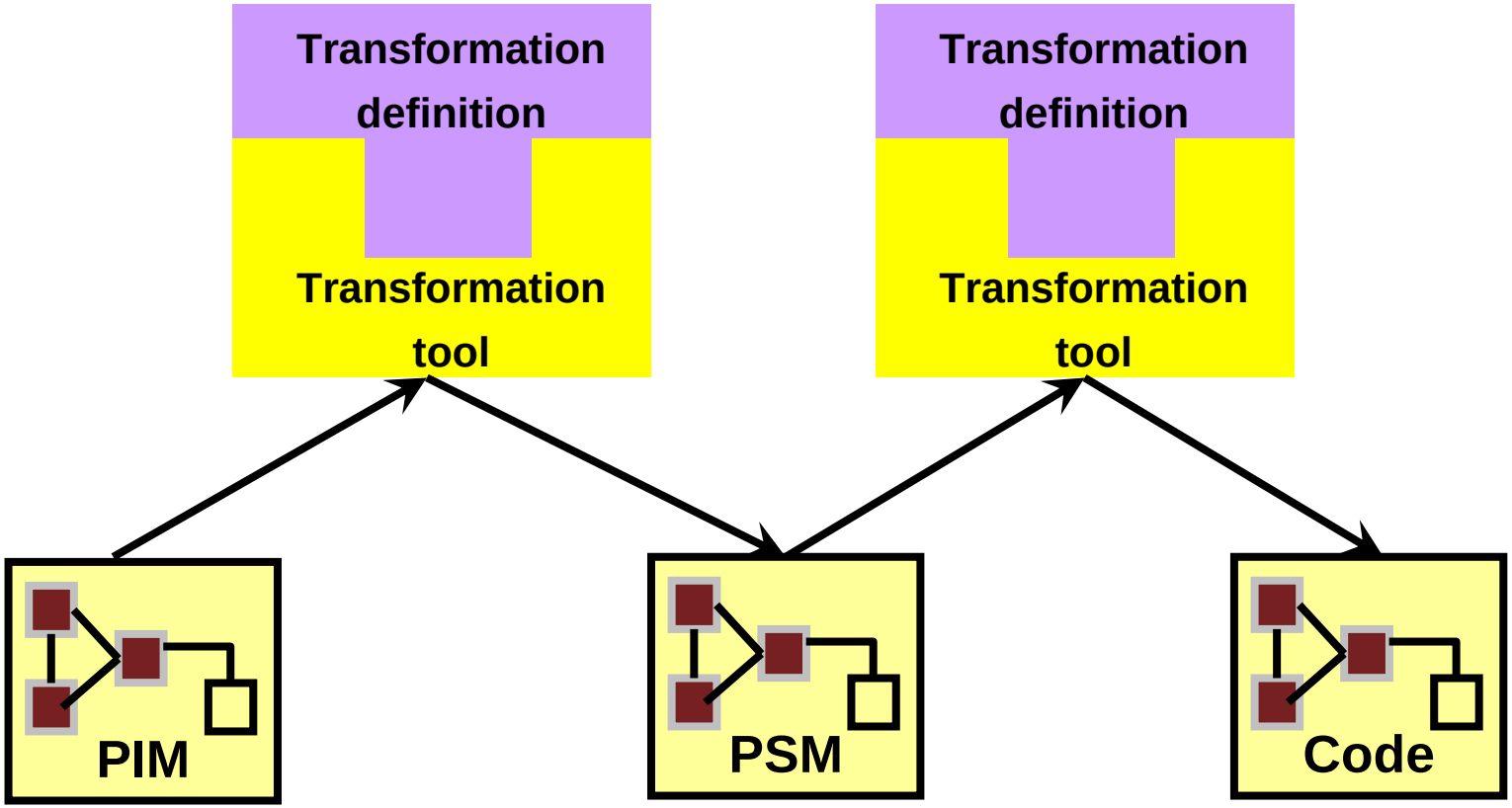
Example of transformation of a PIM class using pattern based transformation

Examples

- MOF and QVT based transformation, transforms based upon PIM metamodel and PSM metamodel
- XSLT based transformation, transforms XMI (or other XML format) of PIM and transforms into source code
- Apply PSM Patterns based on stereotypes defined in the PIM
- Apply patterns interactively, using Borland's Together products achieving many-many transformation

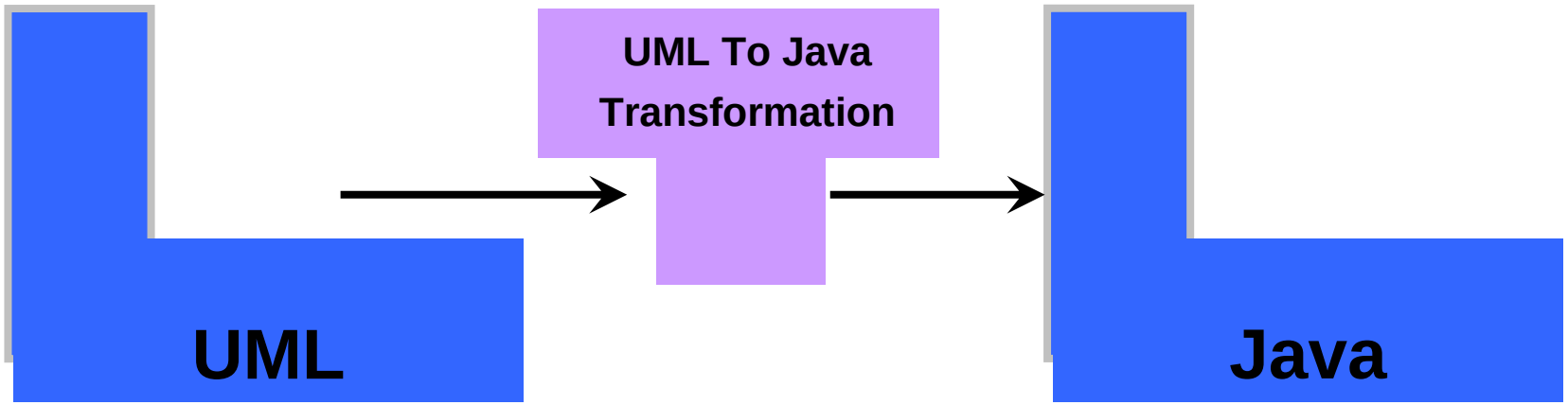
Borland®

MDA Transformation

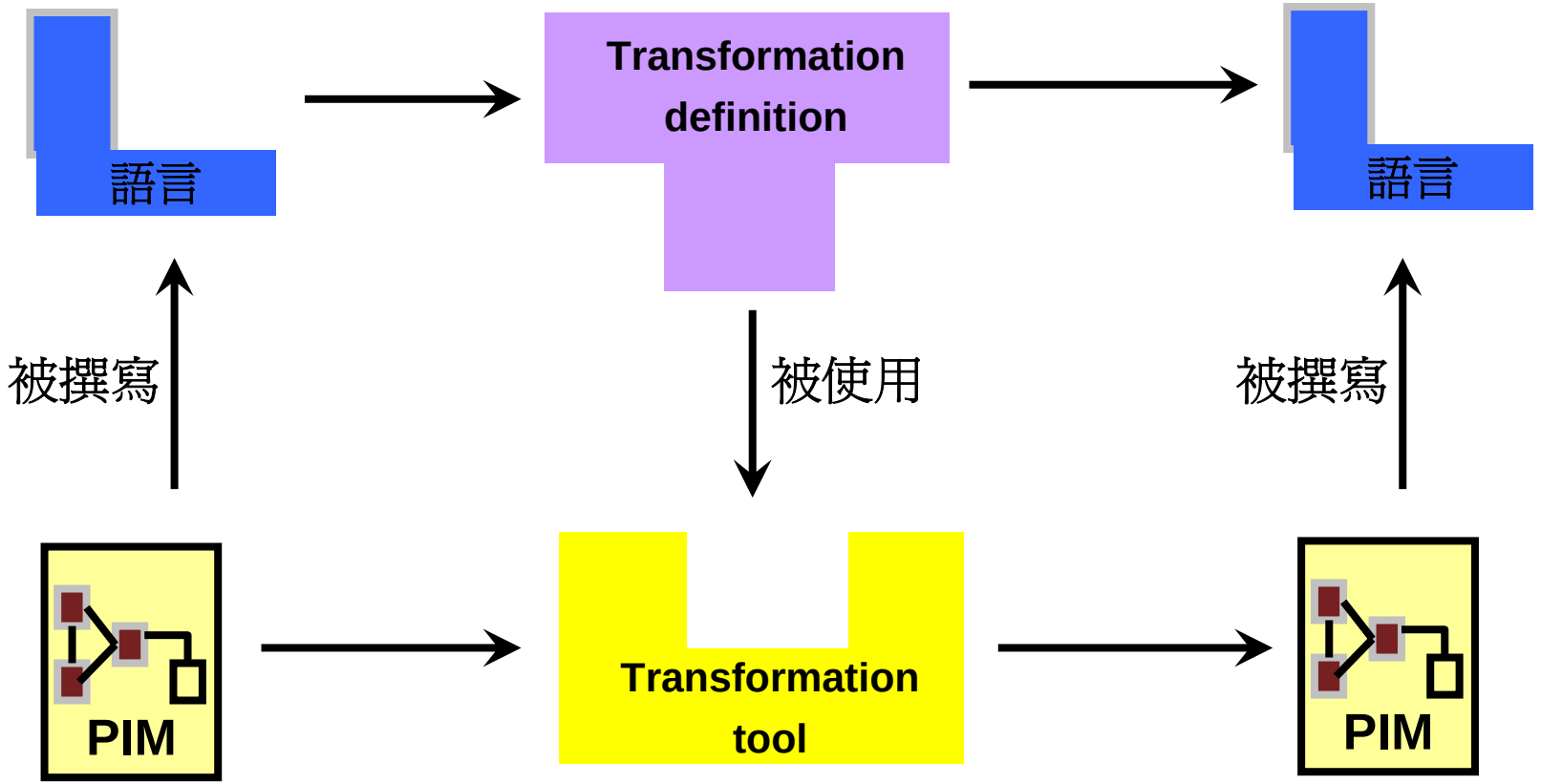


Borland®

MDA Transformation



MDA Transformation



Borland®

So, the focus is on

Languages adequate to express what is required.

These languages need not even be UML

They need not even be “modeling” languages

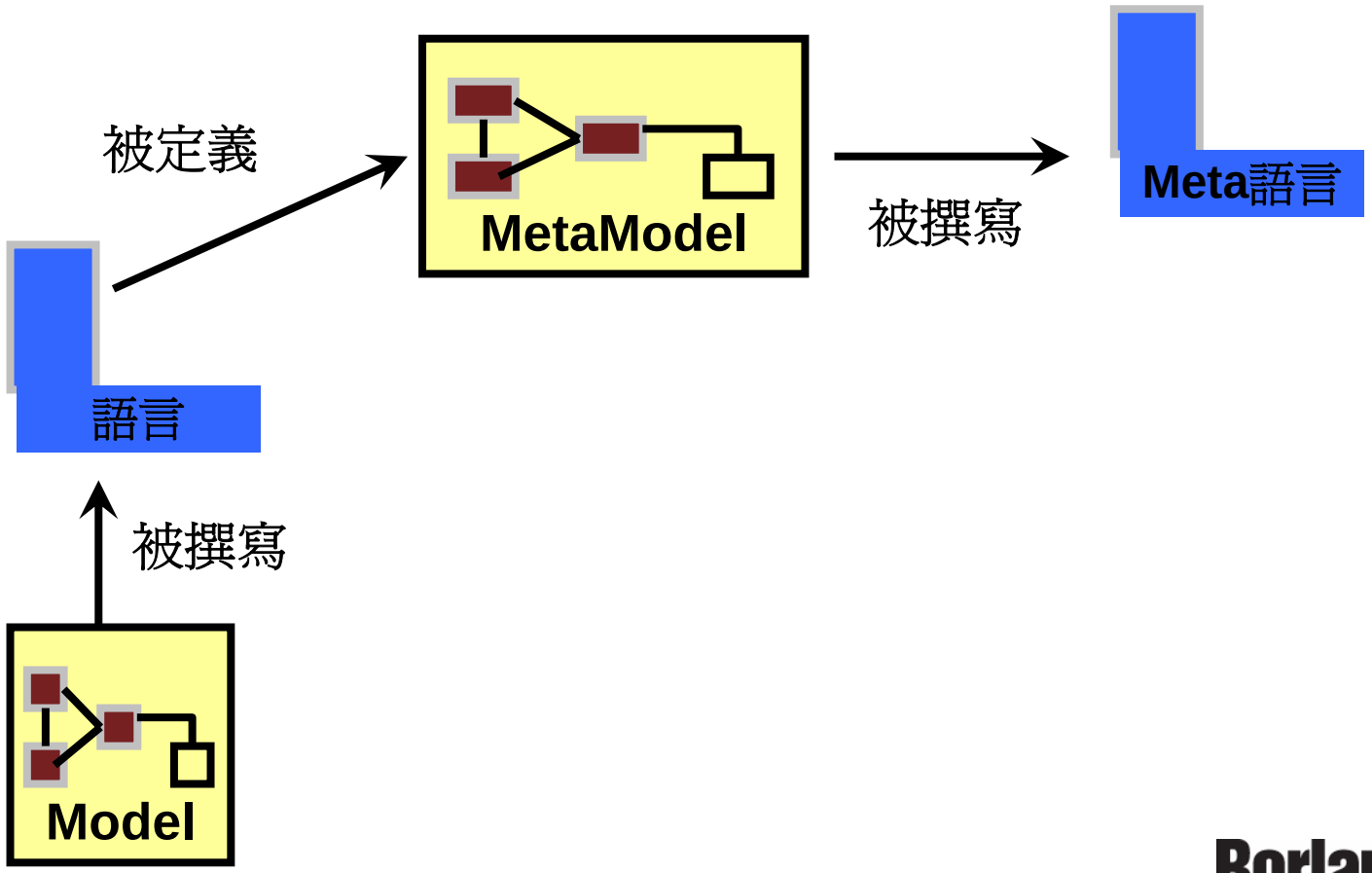
Example: OCL

The point is that the languages need to be well defined so that transformations can be applied to models expressed in those languages.

Ultimately we need Metadata to define the language in which the model is expressed.

Quoted from ORMSC minutes

MetaModeling

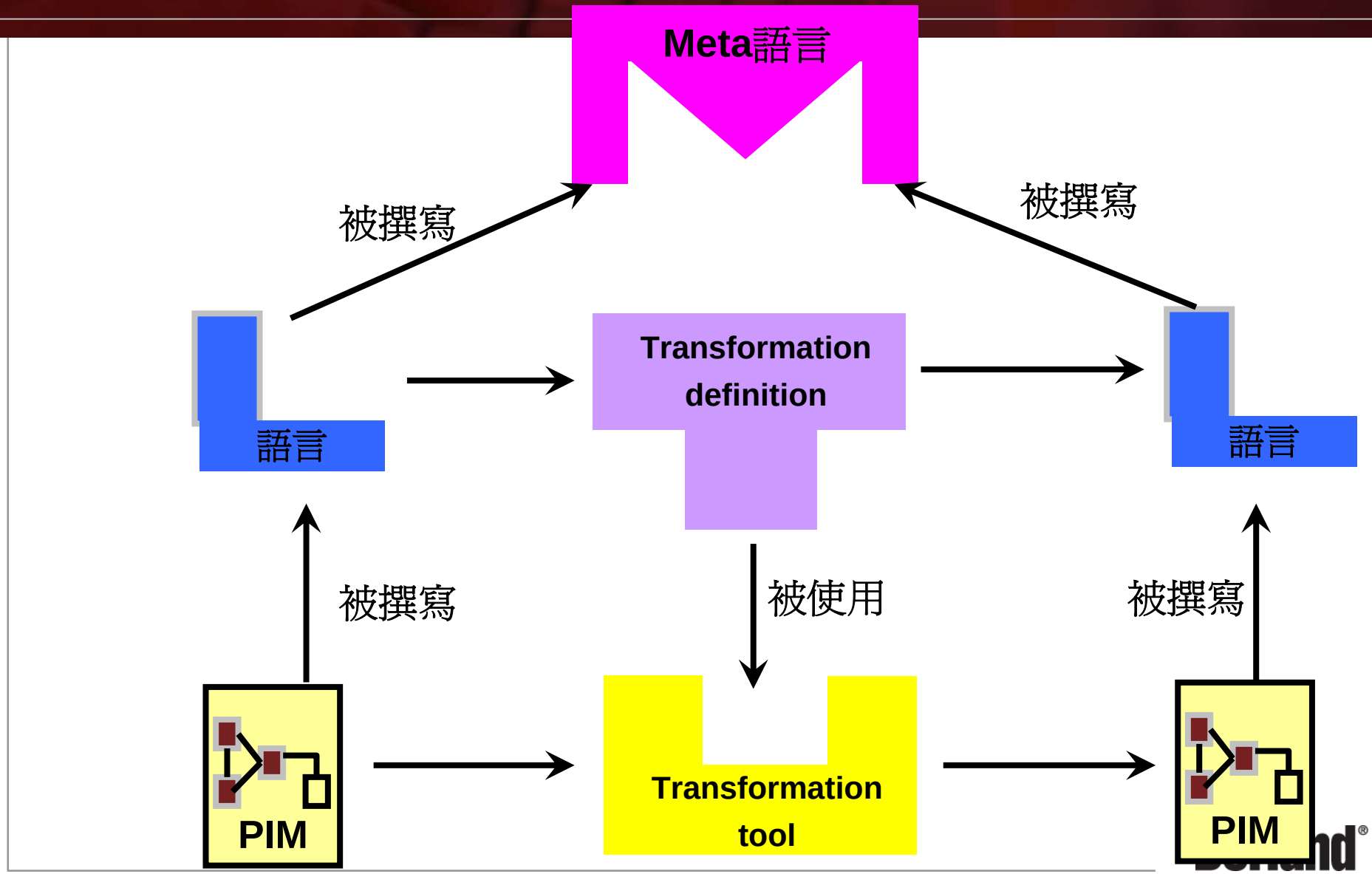


Borland®

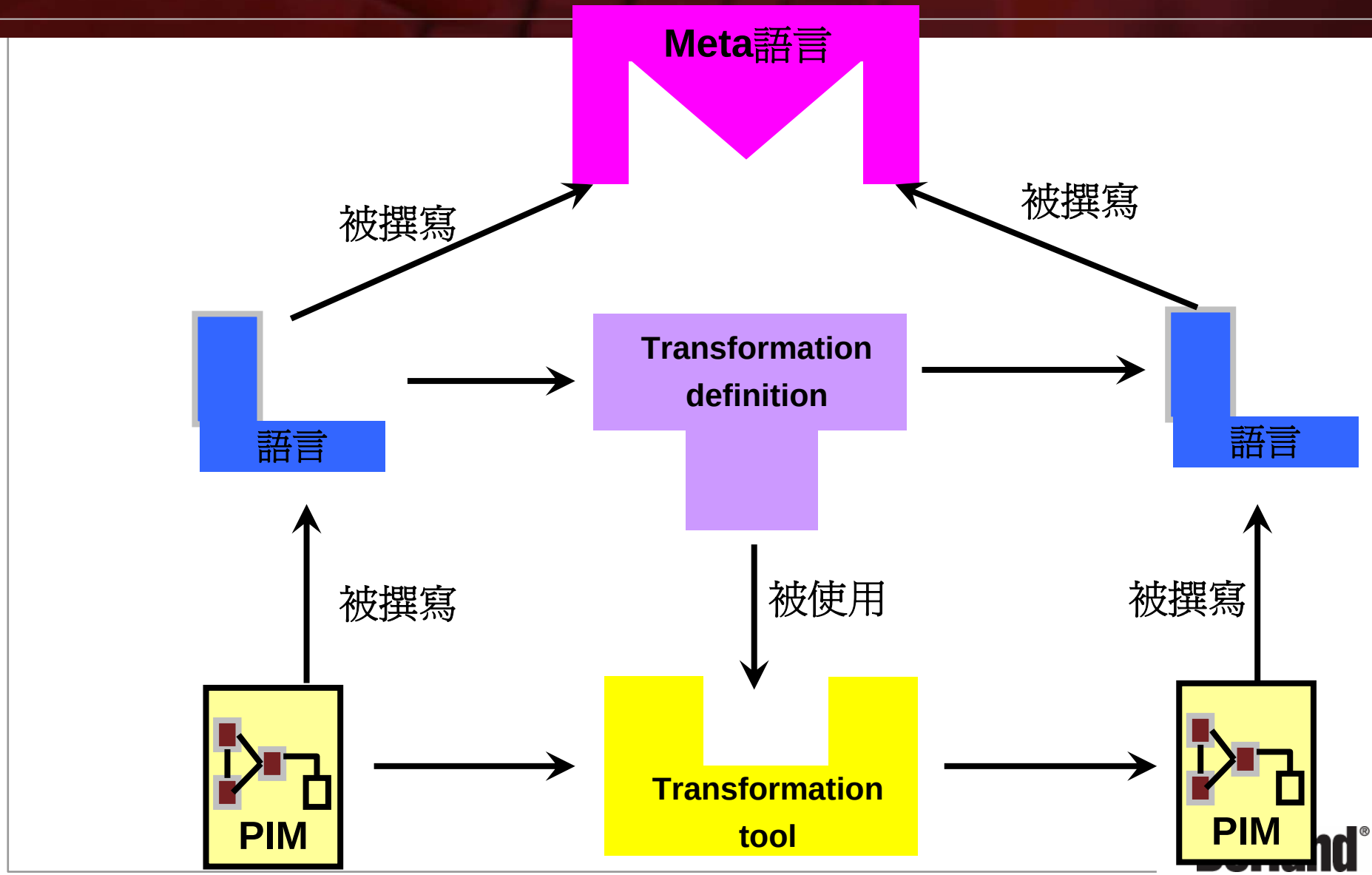
Models, 語言和Meta語言

Borland®

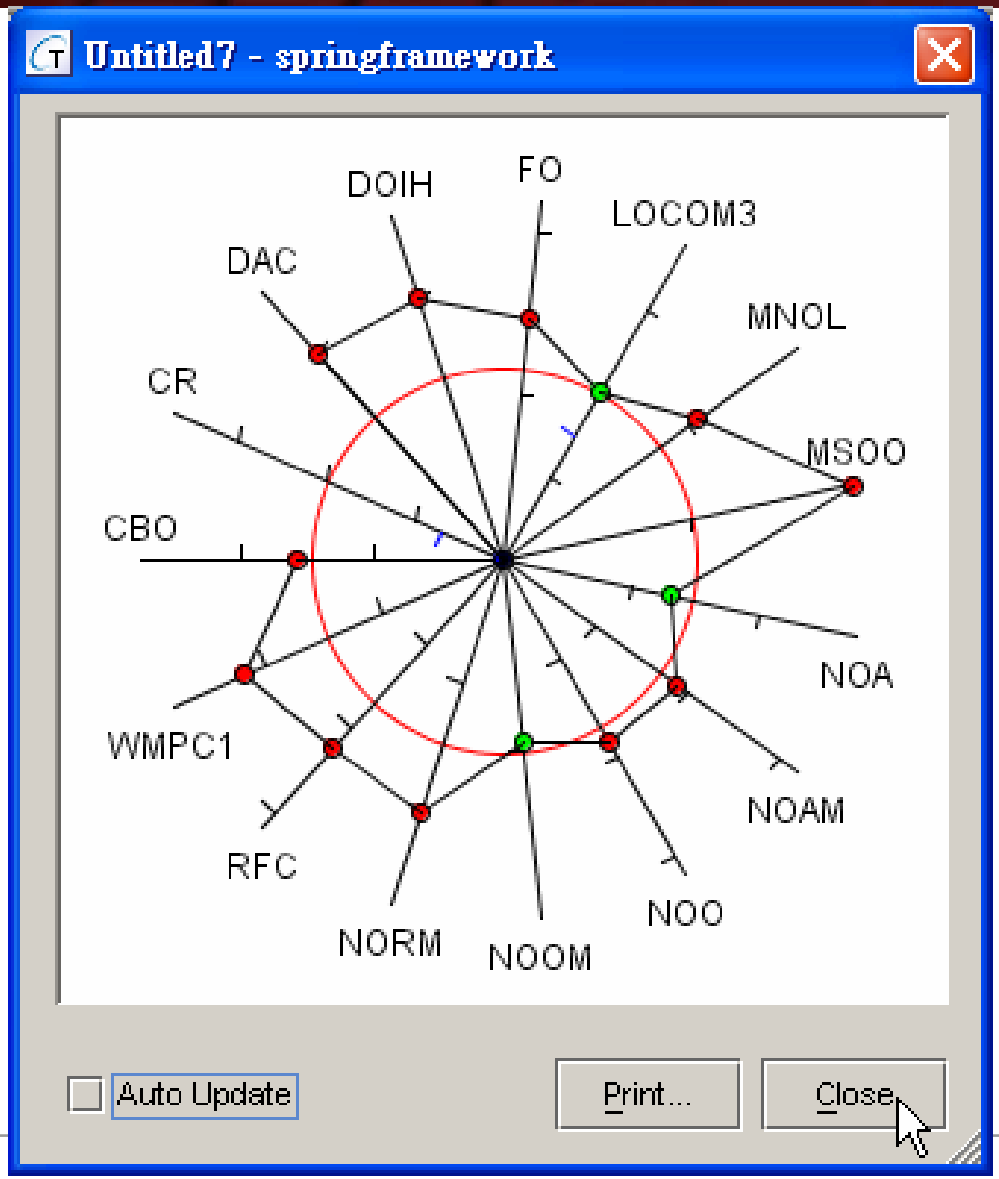
MDA Framework



Programming Quality Of Service

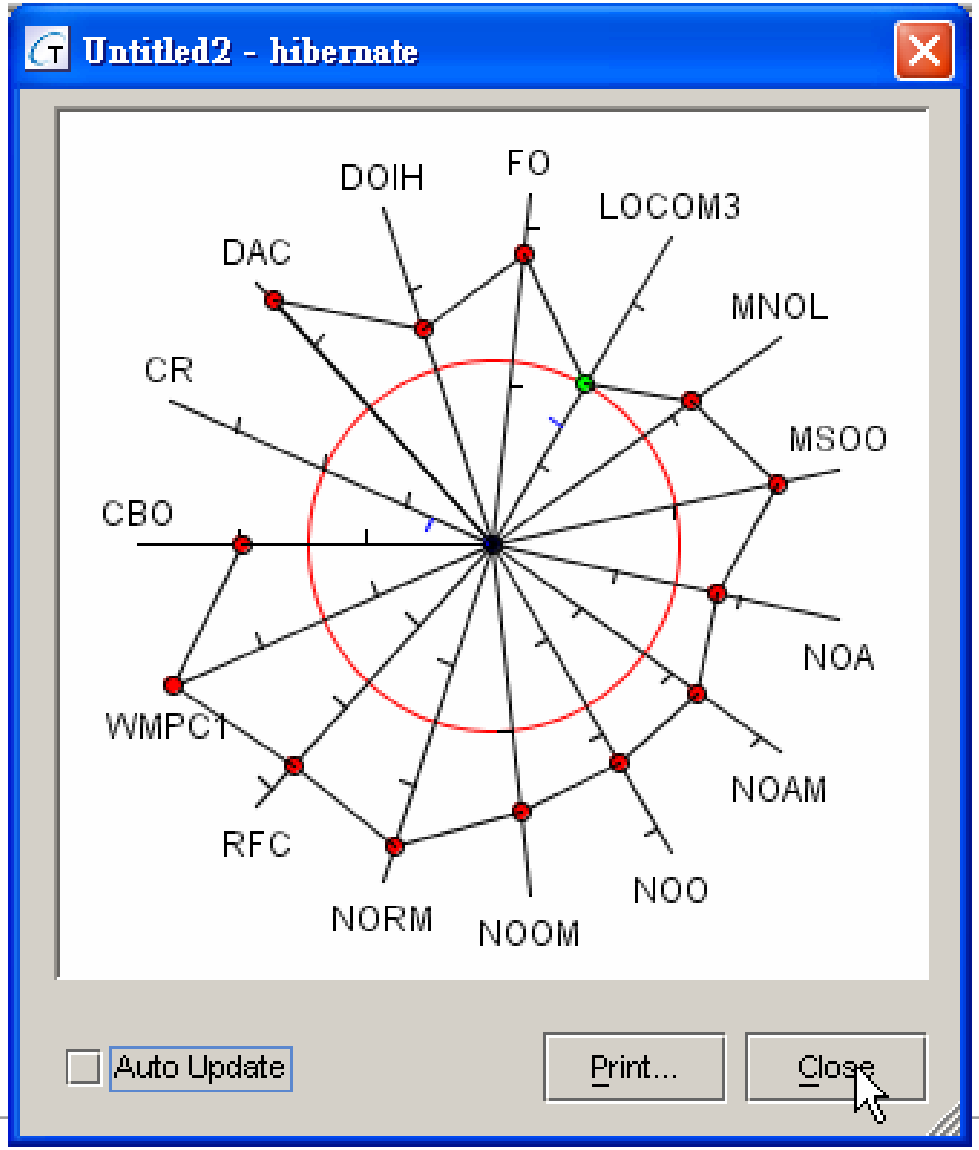


Spring



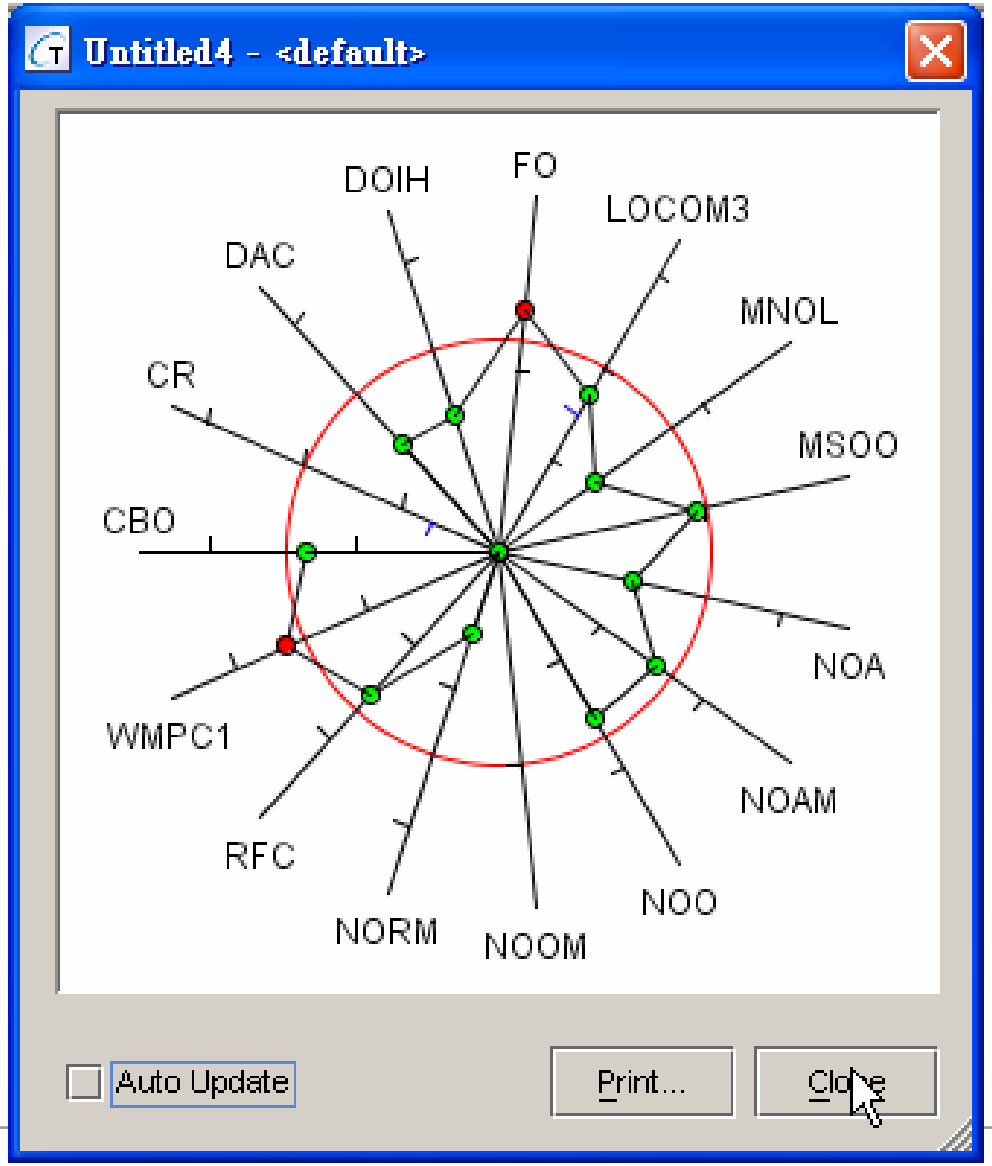
Borland®

Hibernate!



Borland®

Loki



Borland®

MAXIMIZE THE
BUSINESS VALUE
OF SOFTWARE

Q&A

Borland®