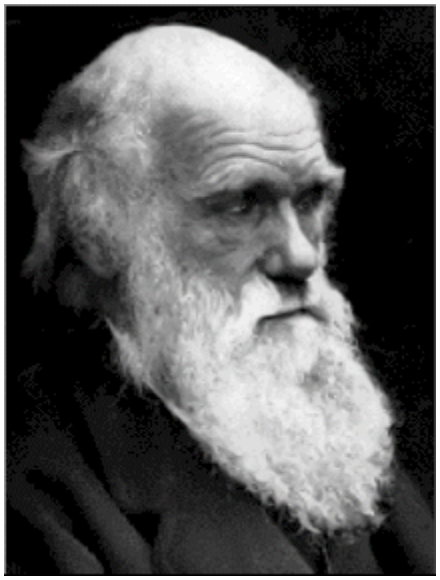
iDarwin

The Java™ Software

Architecture Evolution Tool



Reto Kramer, kramer@acm.org
http://www.reliable-systems.com

Definitions

Software Architecture:

the manner in which the components of a computer system are organised and integrated

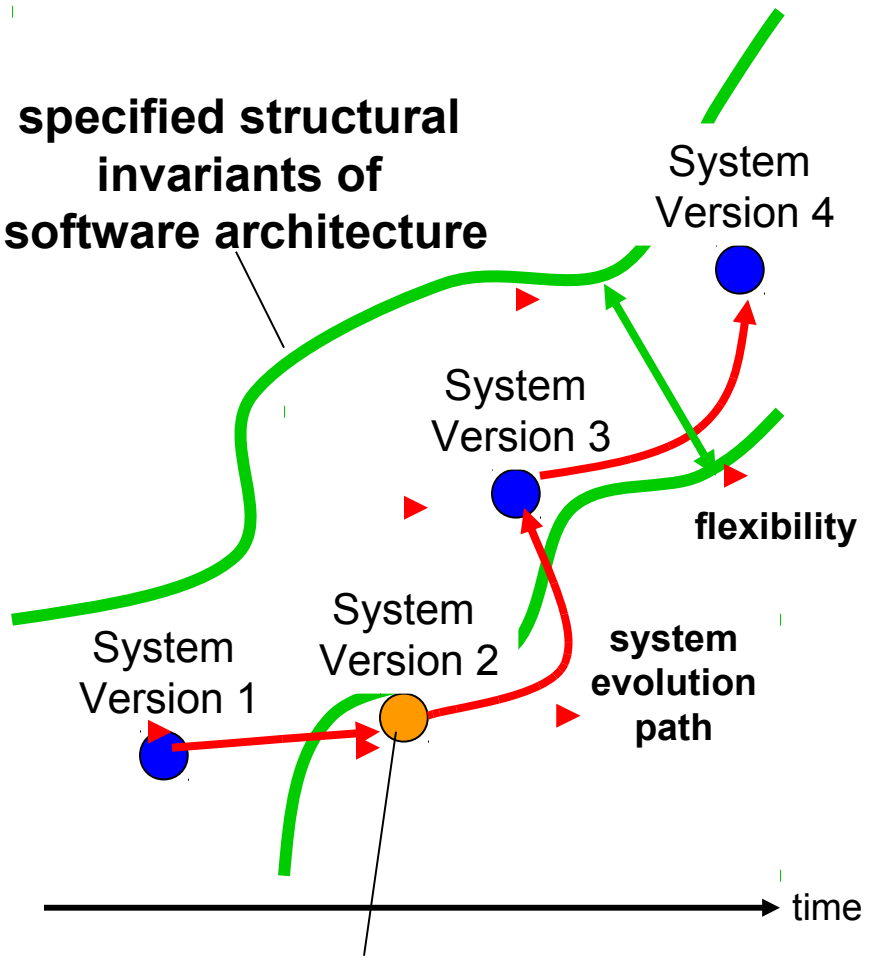
Evolution:

one of a set of prescribed movements, a process of change in a certain direction

Source: Merriam-Webster Dictionary, http://www.britannica.com

What Problem Does it Solve?

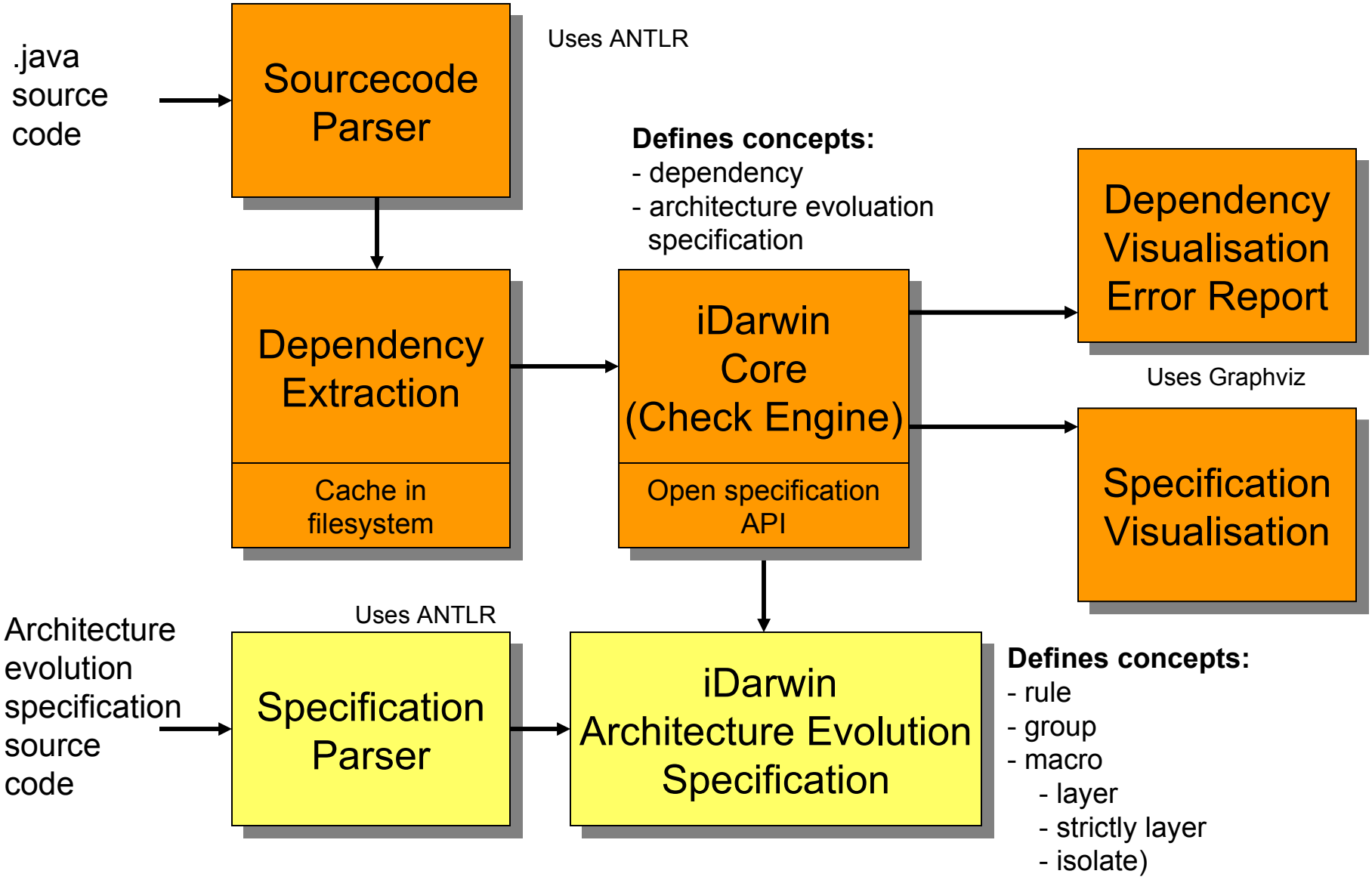
- iDarwin allows the specification and checking of structural invariants on a system's software architecture
- but it still leaves flexibility for change where deemed neccessary



Purpose

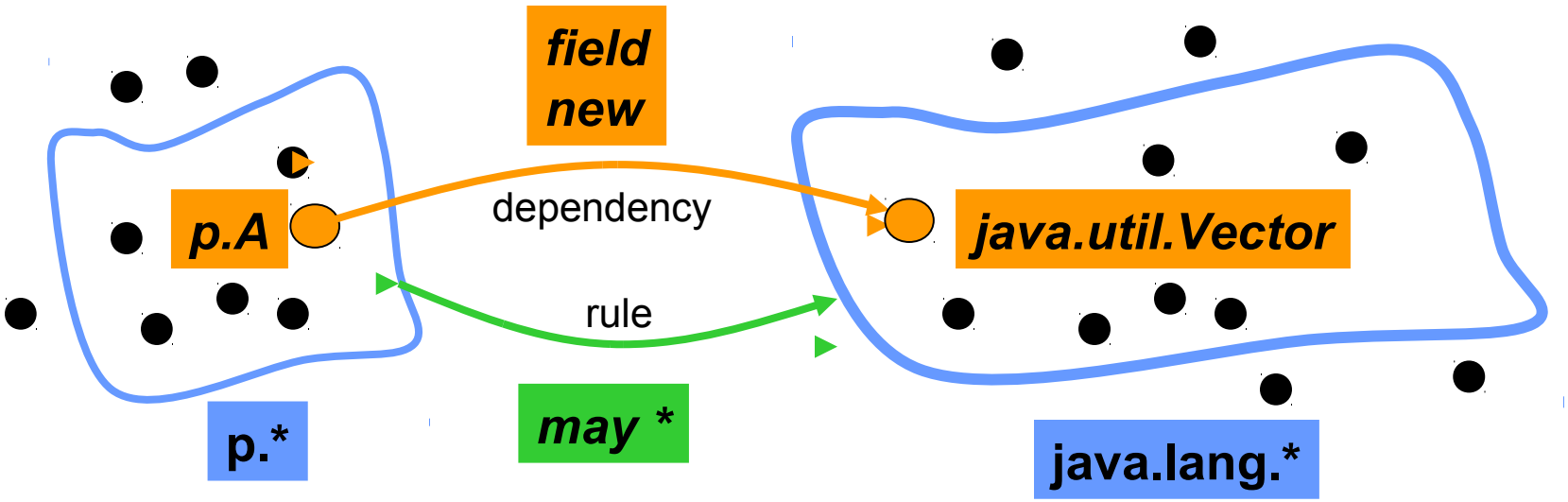
- Early warnings if system structure “deteriorates”
- Makes intend of particular package structures and names explicit
- Test refactoring hyposis
- Notifies architect if new libraries are used
- Controls access to critical libraries for embedded systems

Tool Architecture



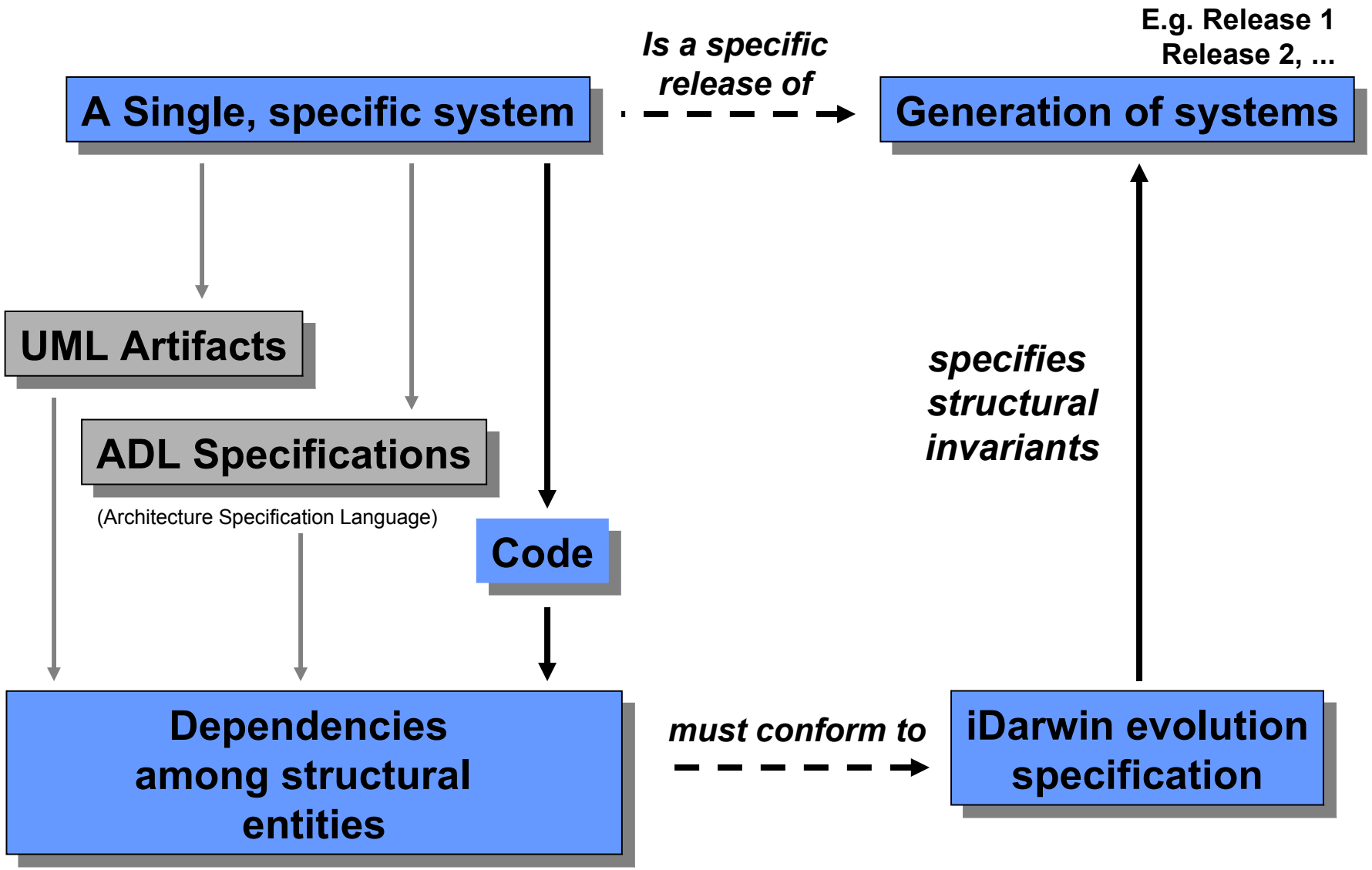
The Concept

```
package p;
Class A {
    private Vector v = new Vector();
}
```




```
<rule name=an example rule>
        p.*  may *  java.lang.*
</rule>
```

Specific Systems vs Generations of Systems



Evolution Specification Language

Available Language Constructs

- **rule**
- **group**
- **macro**
  - *layer*
  - *strictly-layer*
  - *isolate*

Language Construct: Rule

Syntax:

```
<rule name=aLabel>
        source may/mustnot dependency-type target
</rule>
```

Where:

source, target and dependency-type are either
simple patterns or
compound patterns that use set operators

Examples with simple patterns:

```
MyClass may * java.sql*

packageA* mustnot new java.util.Vector
```

Language Construct: Rule

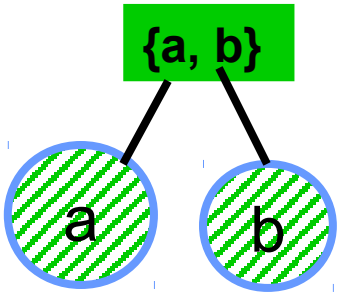
List of known dependency-type :

- extends
- implements
- return
- parameter
- nonstatic fields
- new
- local-var
- static
- cast

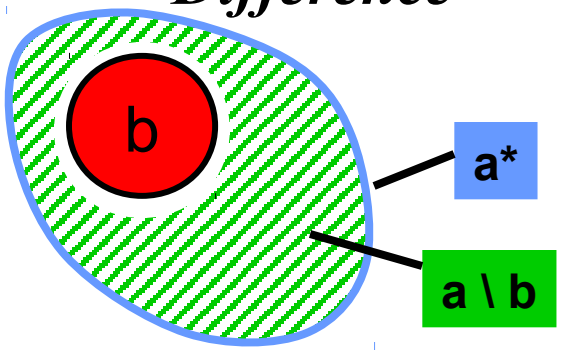
Available Set Operators

a and b are simple patterns (e.g. *.ejb.*, java.sql* or java.lang.Object)

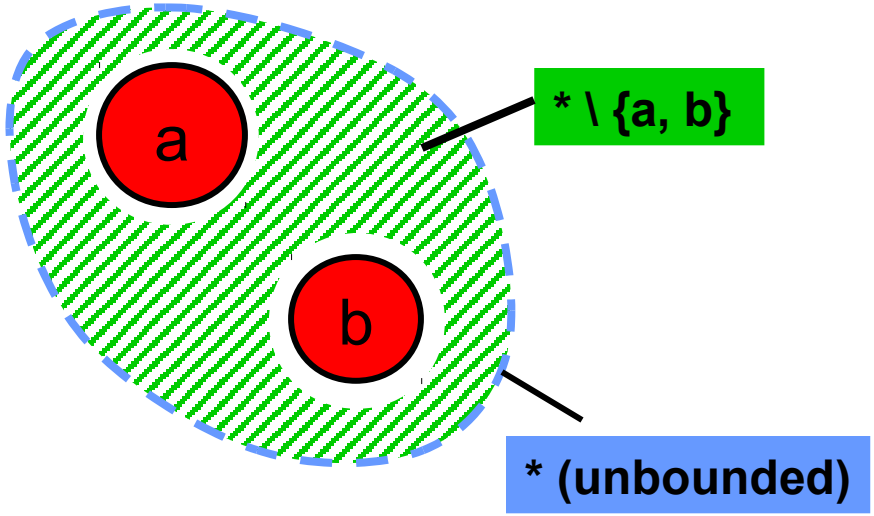
Union



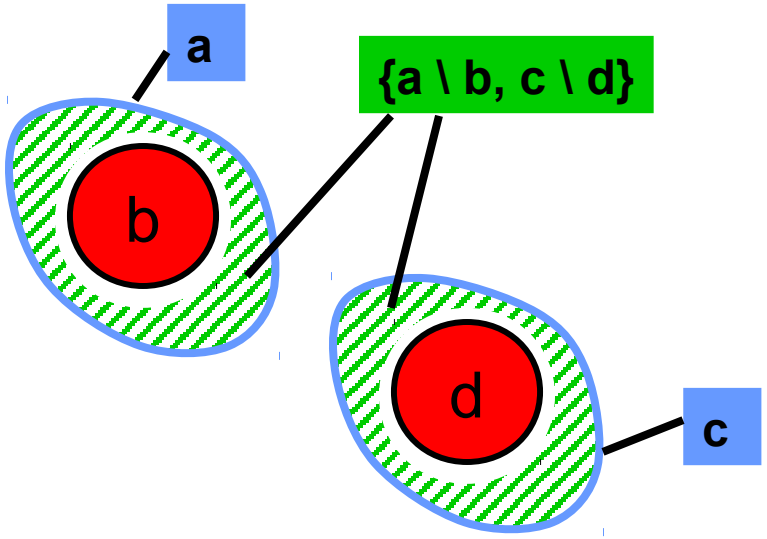
Difference



Example Combination



Example Combination



Language Construct: Rule (con't)

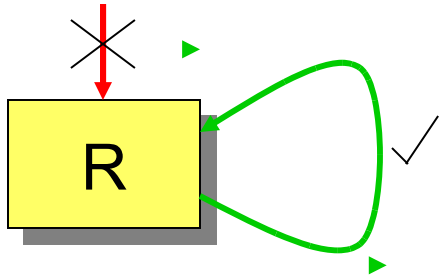
Examples with set operator patterns:

```
<rule name=taglib only knows itself and util>
      com.sun.estore.taglib*
    mustnot *
       *\{com.sun.estore.taglib*, com.sun.estore.util*}
</rule>

<rule name=catalog-exception-1>
      {com.sun.estore.catalog.model.CatalogImpl,
       com.sun.estore.catalog.model.CatalogModel}
    may return
       com.sun.estore.inventory.ejb.Inventory
</rule>
```

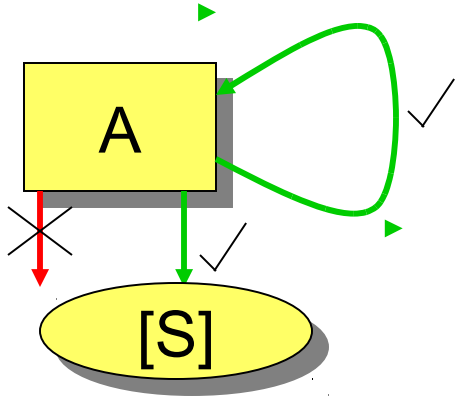
Rule Templates (dependency structures)

The following rules templates are often used in practice:

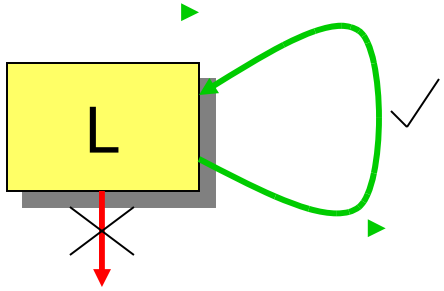
<rule name=root R>
*\R mustnot * R
</rule>



<rule name=A uses group [S] and A>
A mustnot * *\{A,[S]}
</rule>



<rule name=leaf L>
L mustnot * *\L
</rule>



Language Construct: Group

Syntax:

```
<group name=aLabel>
    {a,b,c}
</group>
```

Where:

a,b and c are either simple patterns or compound patterns that use set operators

Reference:

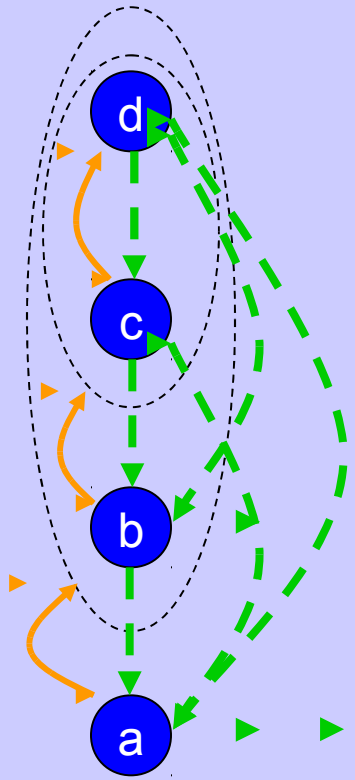
in rules, macros and groups, [group-name] will be substituted with set {a,b,c}

Examples:

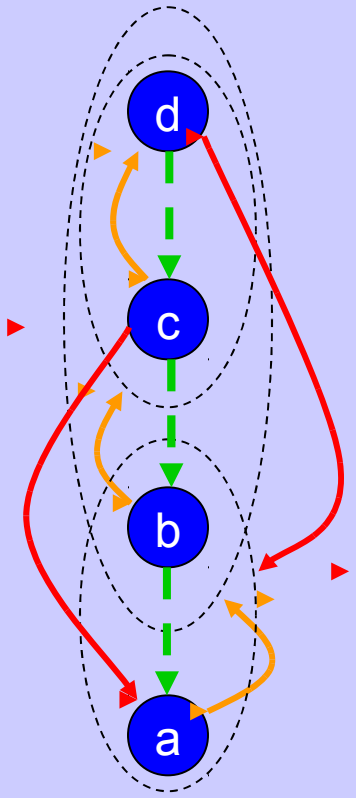
```
<group name=technical-layers>
        {*.model*, *.ejb*, *.web*}
</group>
```

Language Construct: Macros

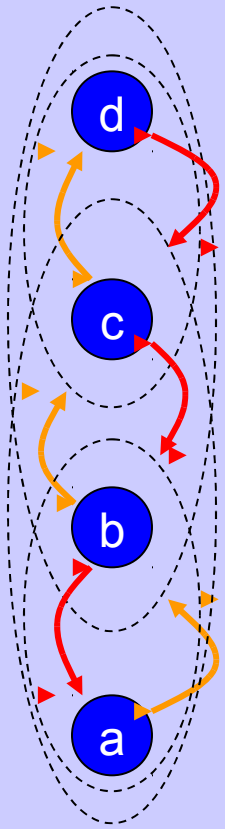
<macro name=**layer**>
{a,b,c,d}
</macro>



<macro name=**strictly-layer**>
{a,b,c,d}
</macro>



<macro name=**isolate**>
{a,b,c,d}
</macro>



no dependencies among a,b,c,d accepted

solid lines denote "mustnot *" relations

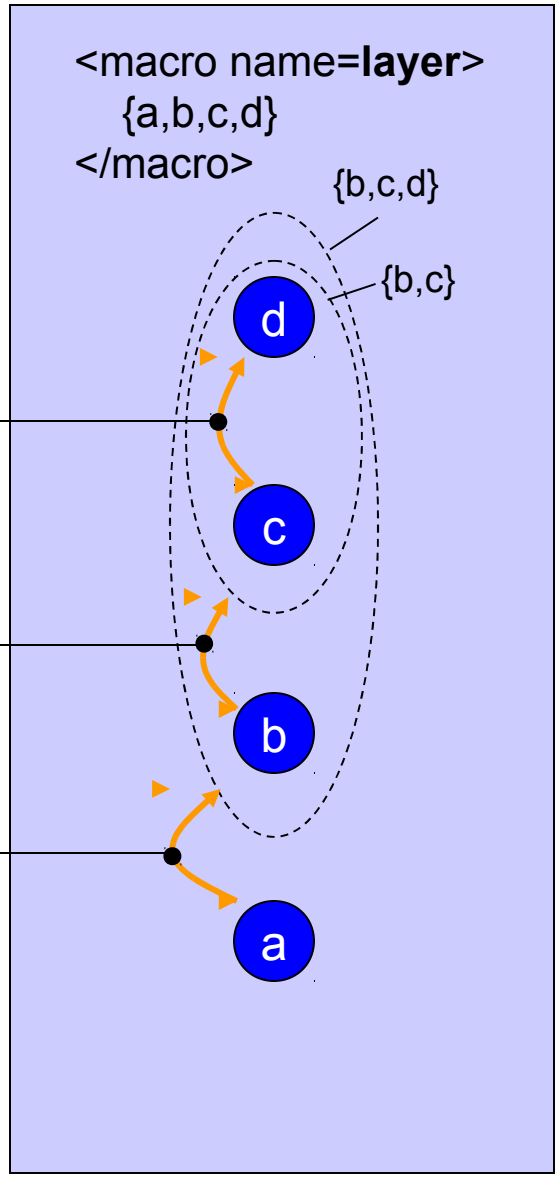
dashed lines denote "accepted" dependencies

Expansion of macros into rules: “layer”

```
<macro name=layer>
   {a,b,c,d}
</macro>
```

expands into the following specification fragment

```
<idarwin language=XMLRuleLanguage>

   <rule name=rule-1>
      c mustnot * d
   </rule>

   <rule name=rule-2>
      b mustnot * {c,d}
   </rule>

   <rule name=rule-3>
      a mustnot * {b,c,d}
   </rule>

</idarwin>
```



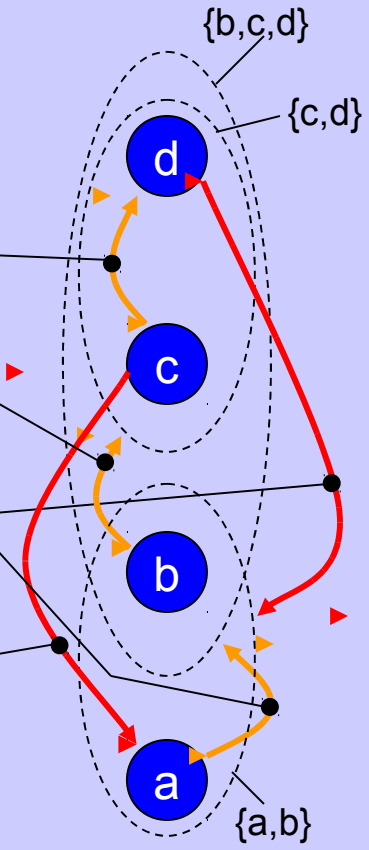
Expansion of macros into rules: “strictly-layer”

```
<macro name=strictly-layer>
   {a,b,c,d}
</macro>
```

expands into the following specification fragment

```
<idarwin language=XMLRuleLanguage>

   expand as shown previously
   <macro name=layer>
      {a,b,c,d}
   </macro>

   &

   <rule name=rule-4>
      d mustnot * {a,b}
   </rule>

   <rule name=rule-5>
      c mustnot * a
   </rule>

</idarwin>
```

```
<macro name=stricly-layer>
   {a,b,c,d}
</macro>
```

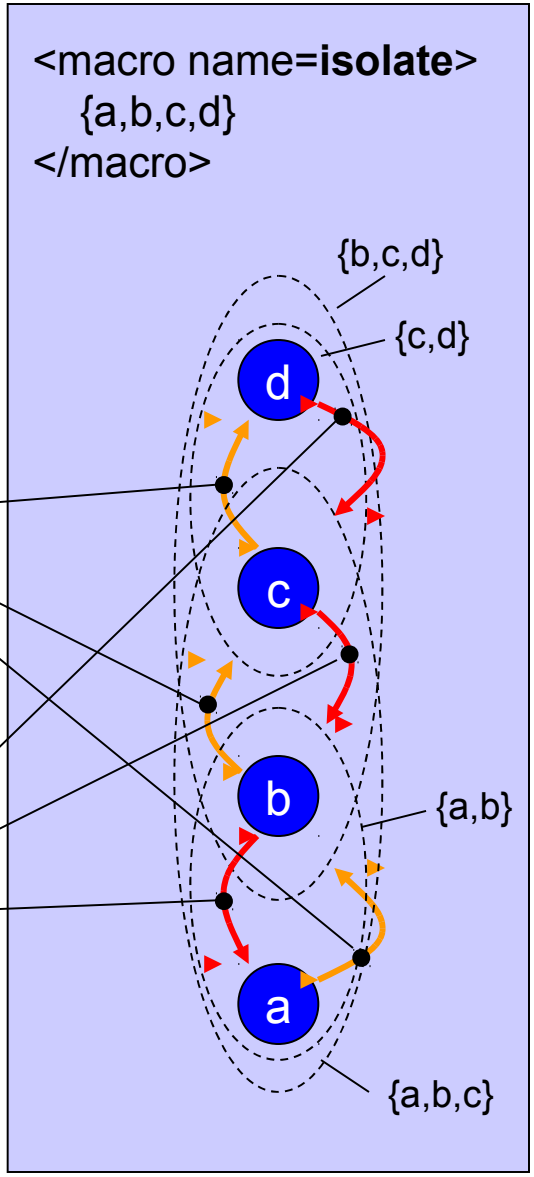


Expansion of macros into rules: “isolate”

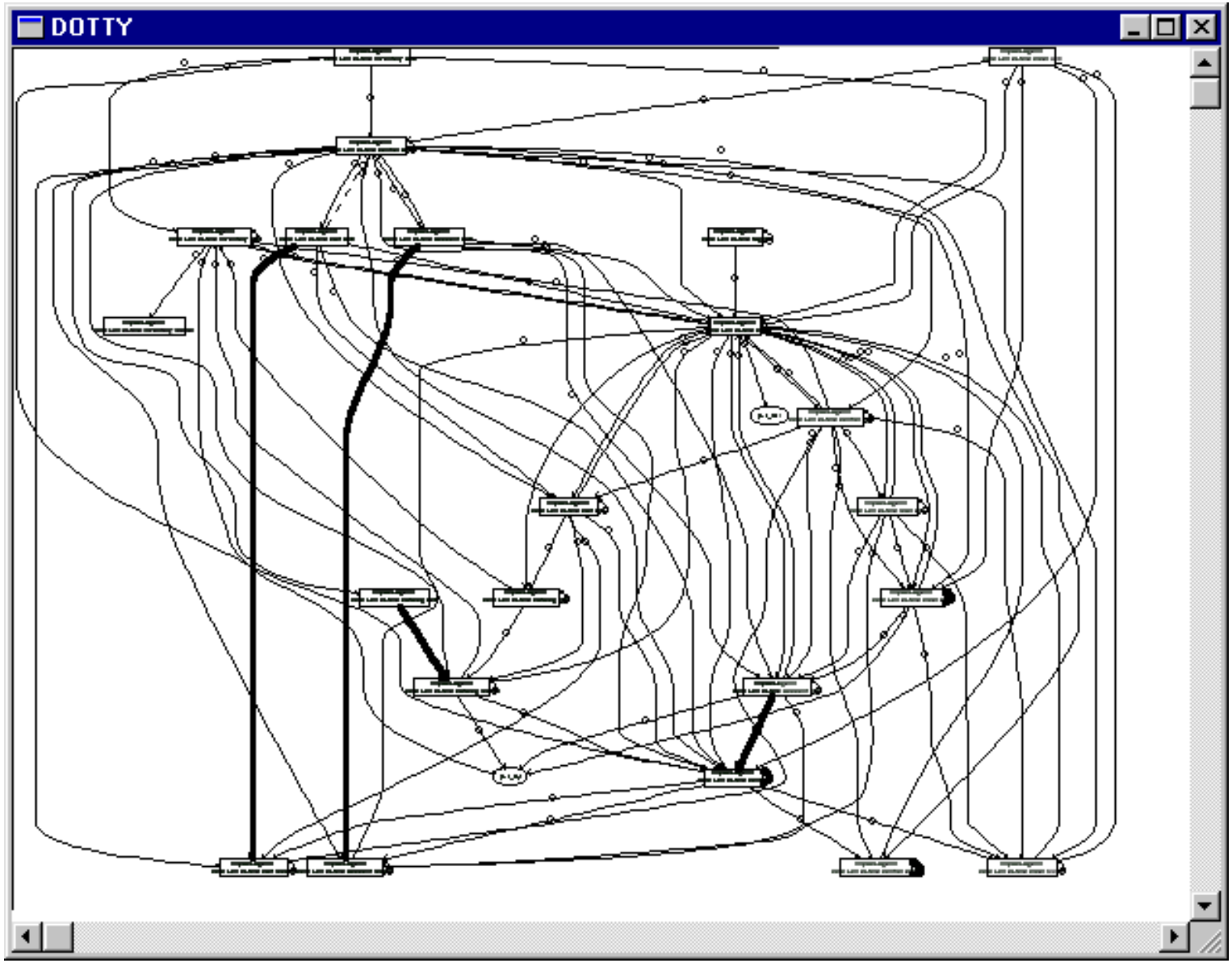
```
<macro name=isolate>
   {a,b,c,d}
</macro>
```

expands into the following specification fragment

```
<idarwin language=XMLRuleLanguage>

   expand as shown previously
      <macro name=layer>
         {a,b,c,d}
      </macro>

   &

   expand as shown previously
      <macro name=layer>
         {d,c,b,a}  // reverse order
      </macro>

</idarwin>
```

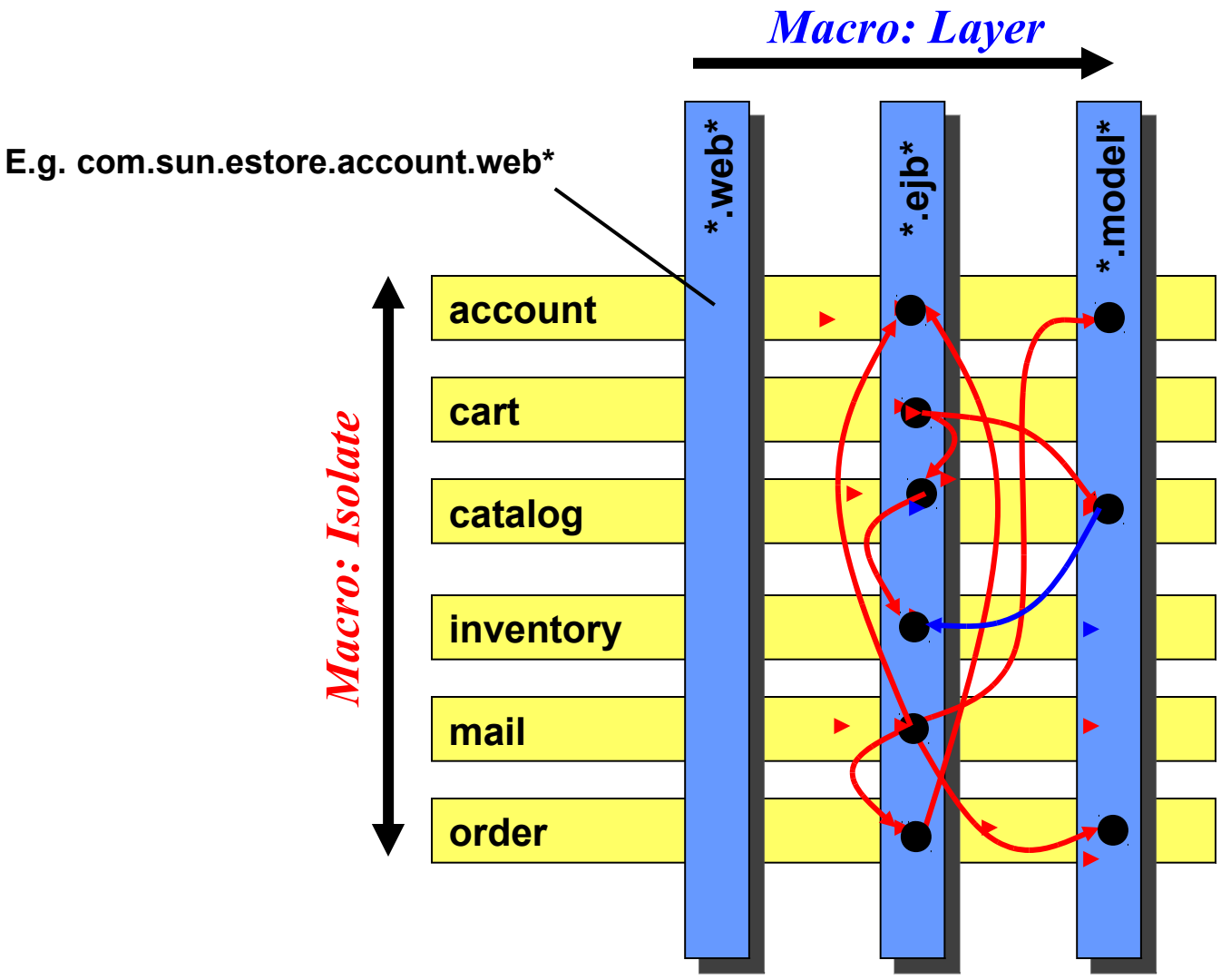


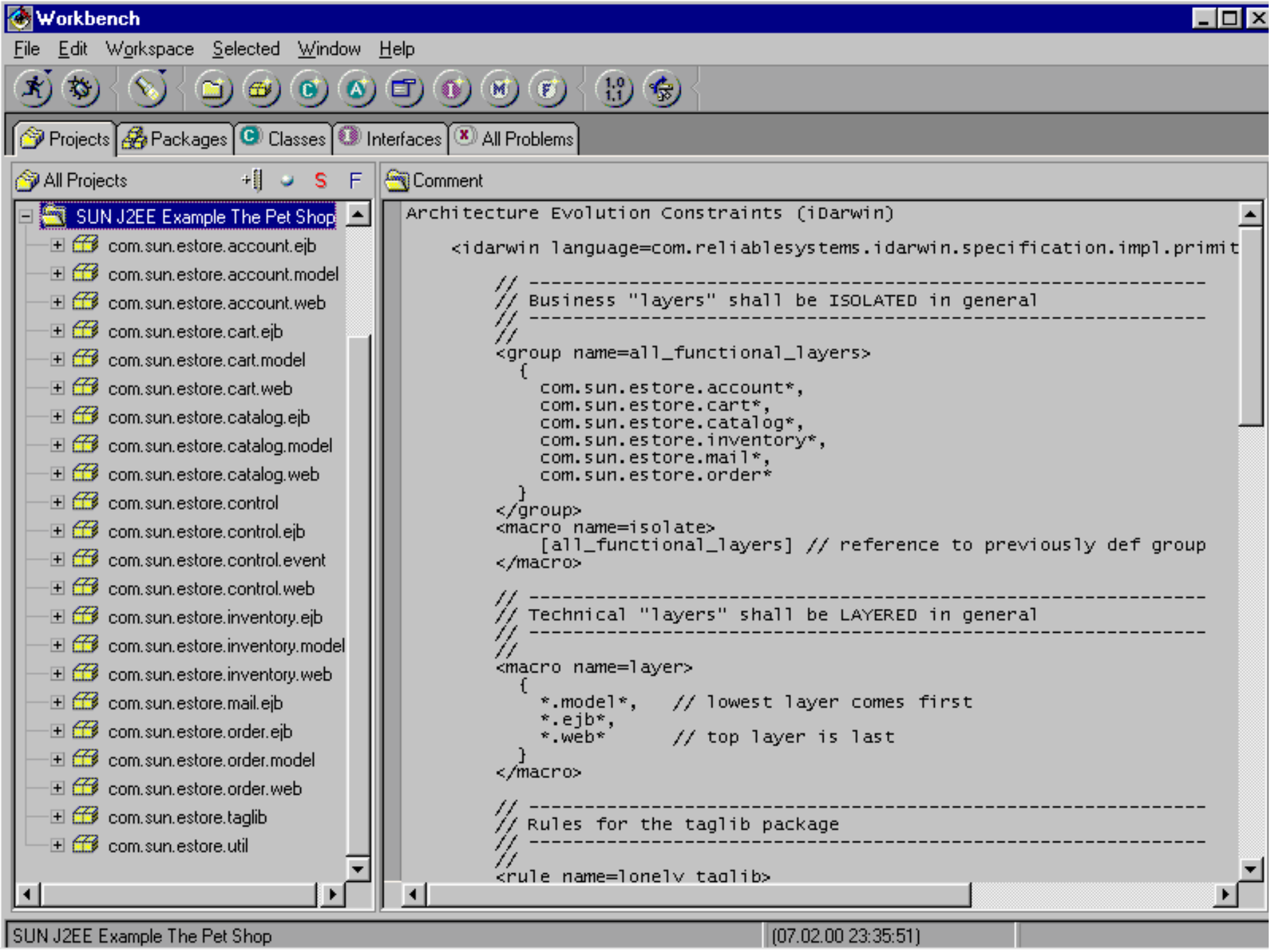
SUN J2EE Pet Shop Example (22 packages)



Reference: Sun BluePrints™, Design Guidelines for Java 2 SDK, Enterprise Edition
http://developer.java.sun.com/developer/earlyAccess/j2sdkee/download-docs.html

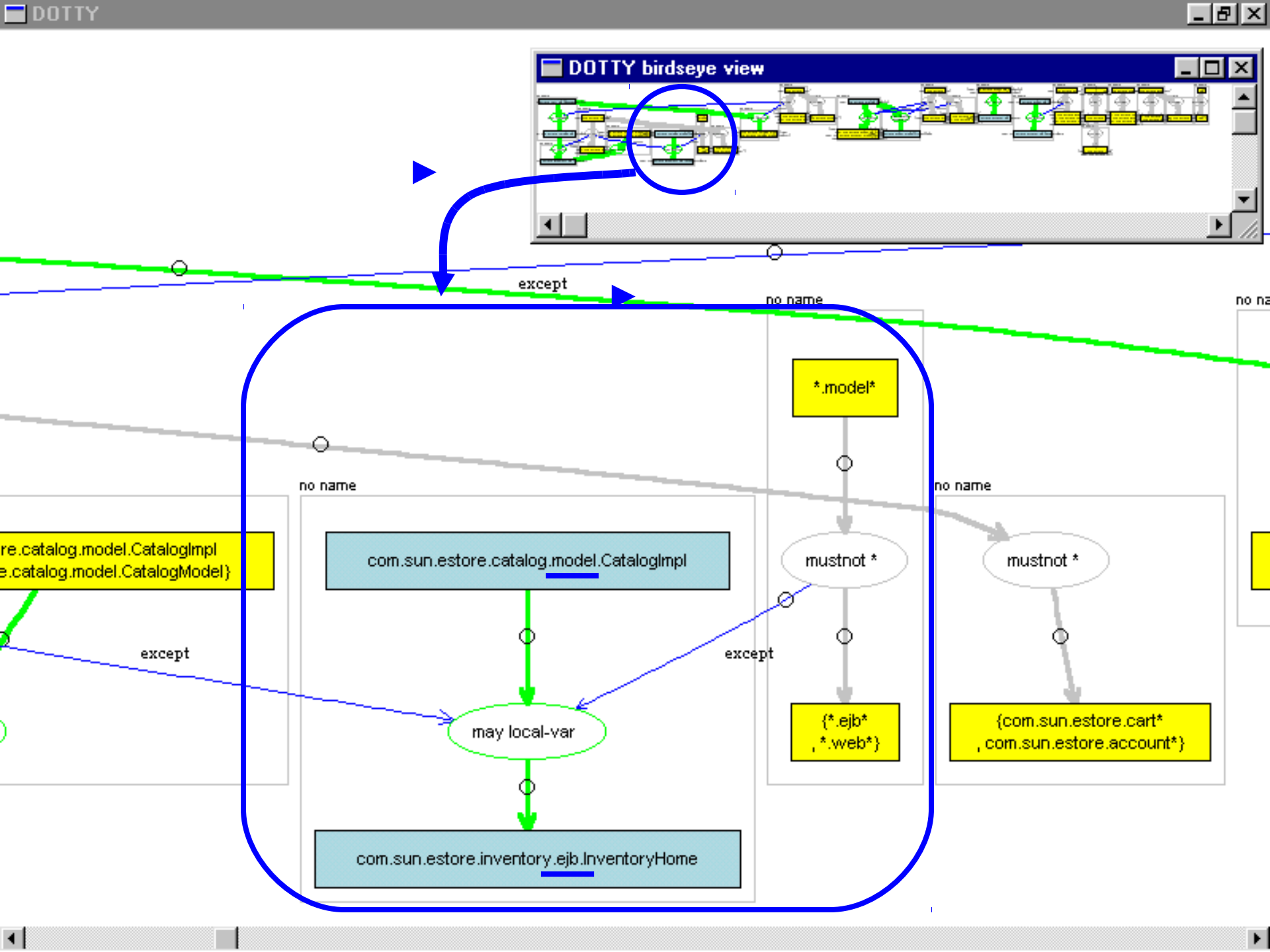
Pet Shop Architecture Evolution Specification

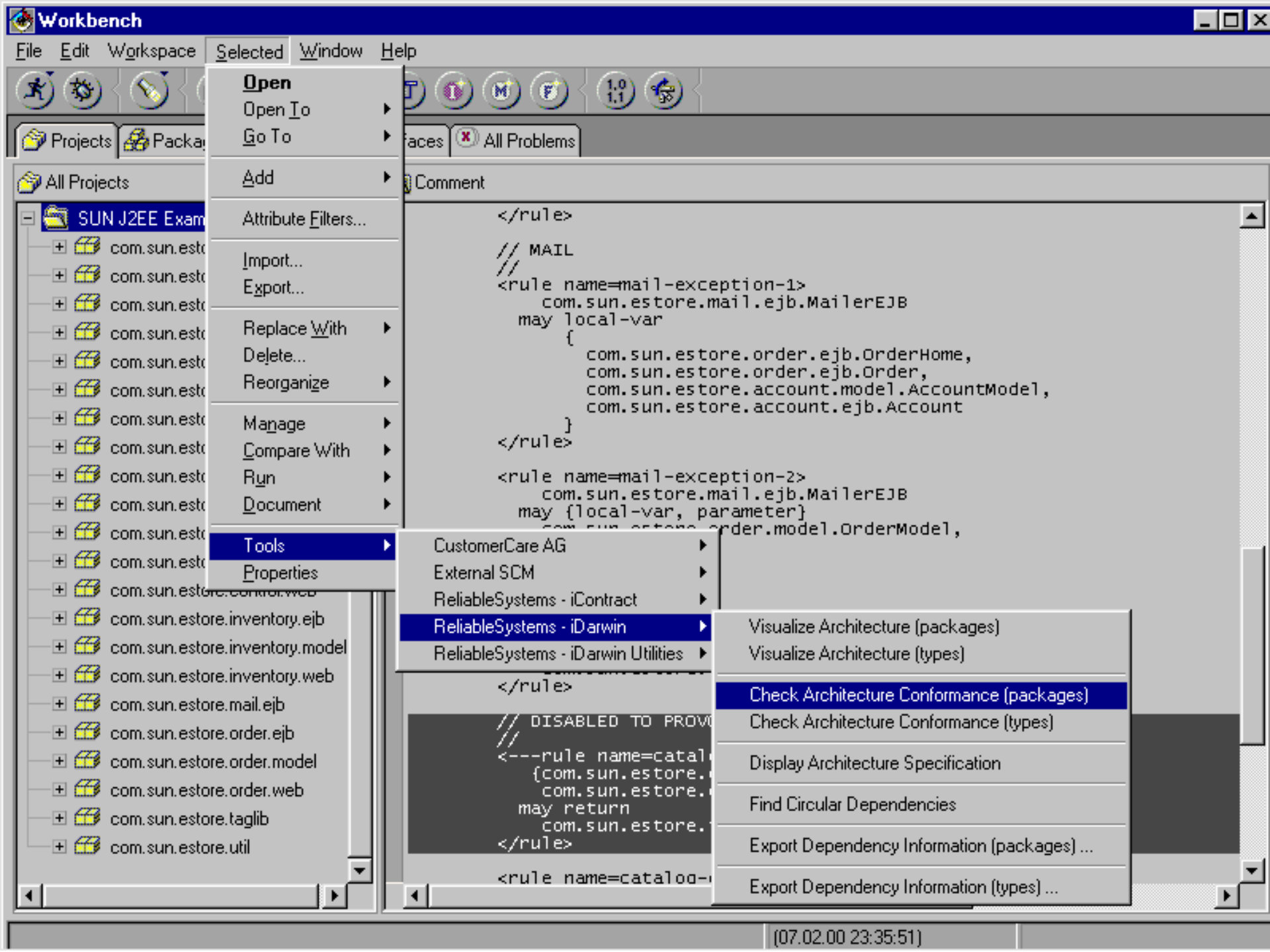


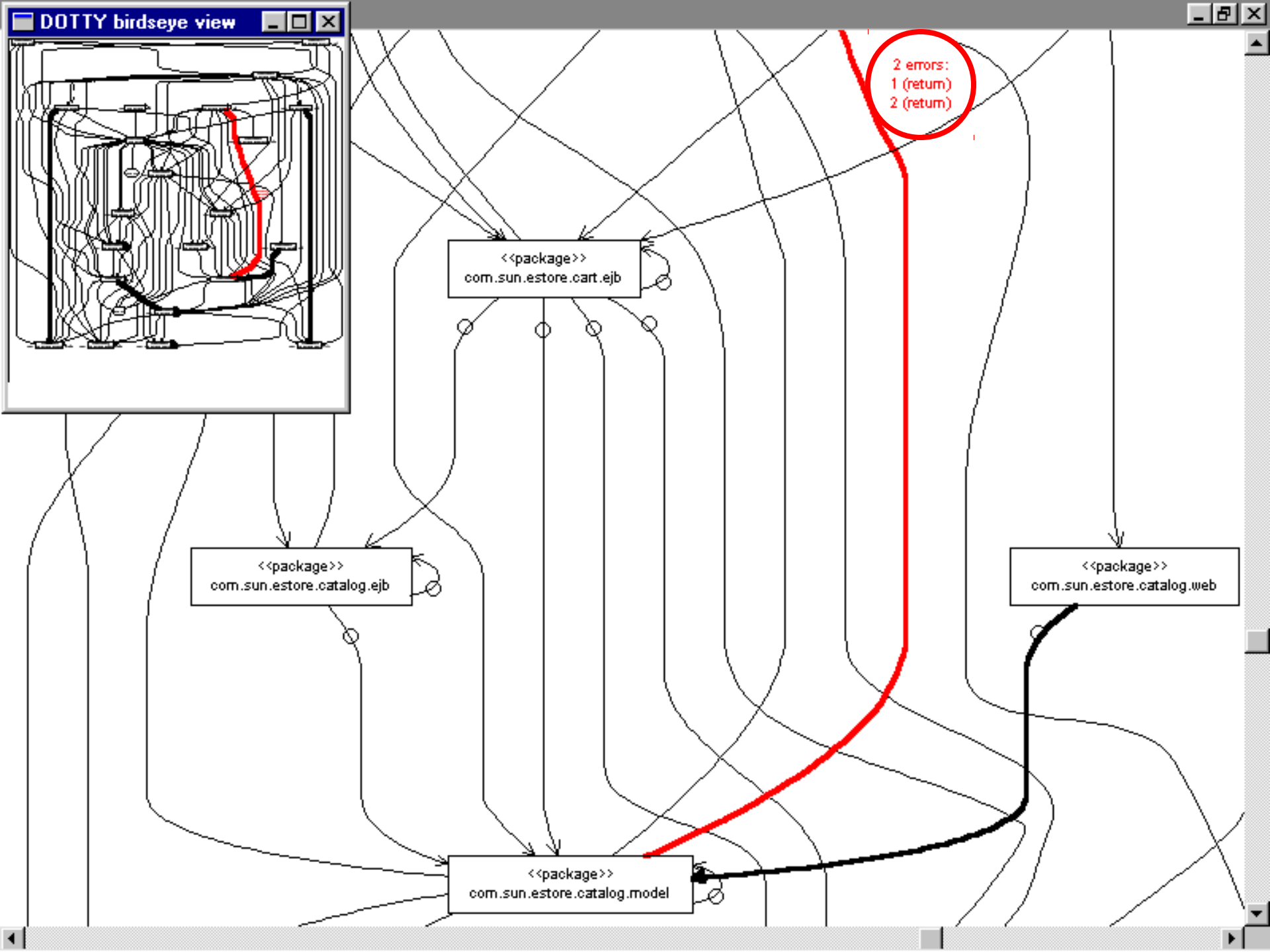

Workbench
File Edit Workspace Selected Window Help
Projects Packages Classes Interfaces All Problems
All Projects
SUN J2EE Example The Pet Shop
com.sun.estore.account.ejb
com.sun.estore.account.model
com.sun.estore.account.web
com.sun.estore.cart.ejb
com.sun.estore.cart.model
com.sun.estore.cart.web
com.sun.estore.catalog.ejb
com.sun.estore.catalog.model
com.sun.estore.catalog.web
com.sun.estore.control
com.sun.estore.control.ejb
com.sun.estore.control.event
com.sun.estore.control.web
com.sun.estore.inventory.ejb
com.sun.estore.inventory.model
com.sun.estore.inventory.web
com.sun.estore.mail.ejb
com.sun.estore.order.ejb
com.sun.estore.order.model
com.sun.estore.order.web
com.sun.estore.taglib
com.sun.estore.util
Comment
Architecture Evolution Constraints (iDarwin)
<idarwin language=com.reliablesystems.idarwin.specification.impl.primit
// ----------
// Business "layers" shall be ISOLATED in general
// ----------
//
<group name=all_functional_layers>
{
com.sun.estore.account*,
com.sun.estore.cart*,
com.sun.estore.catalog*,
com.sun.estore.inventory*,
com.sun.estore.mail*,
com.sun.estore.order*
}
</group>
<macro name=isolate>
[all_functional_layers] // reference to previously def group
</macro>
// ----------
// Technical "layers" shall be LAYERED in general
// ----------
//
<macro name=layer>
{
.model, // lowest layer comes first
.ejb,
.web // top layer is last
}
</macro>
// ----------
// Rules for the taglib package
// ----------
//
<rule name=lonely taglib>
SUN J2EE Example The Pet Shop
(07.02.00 23:35:51)

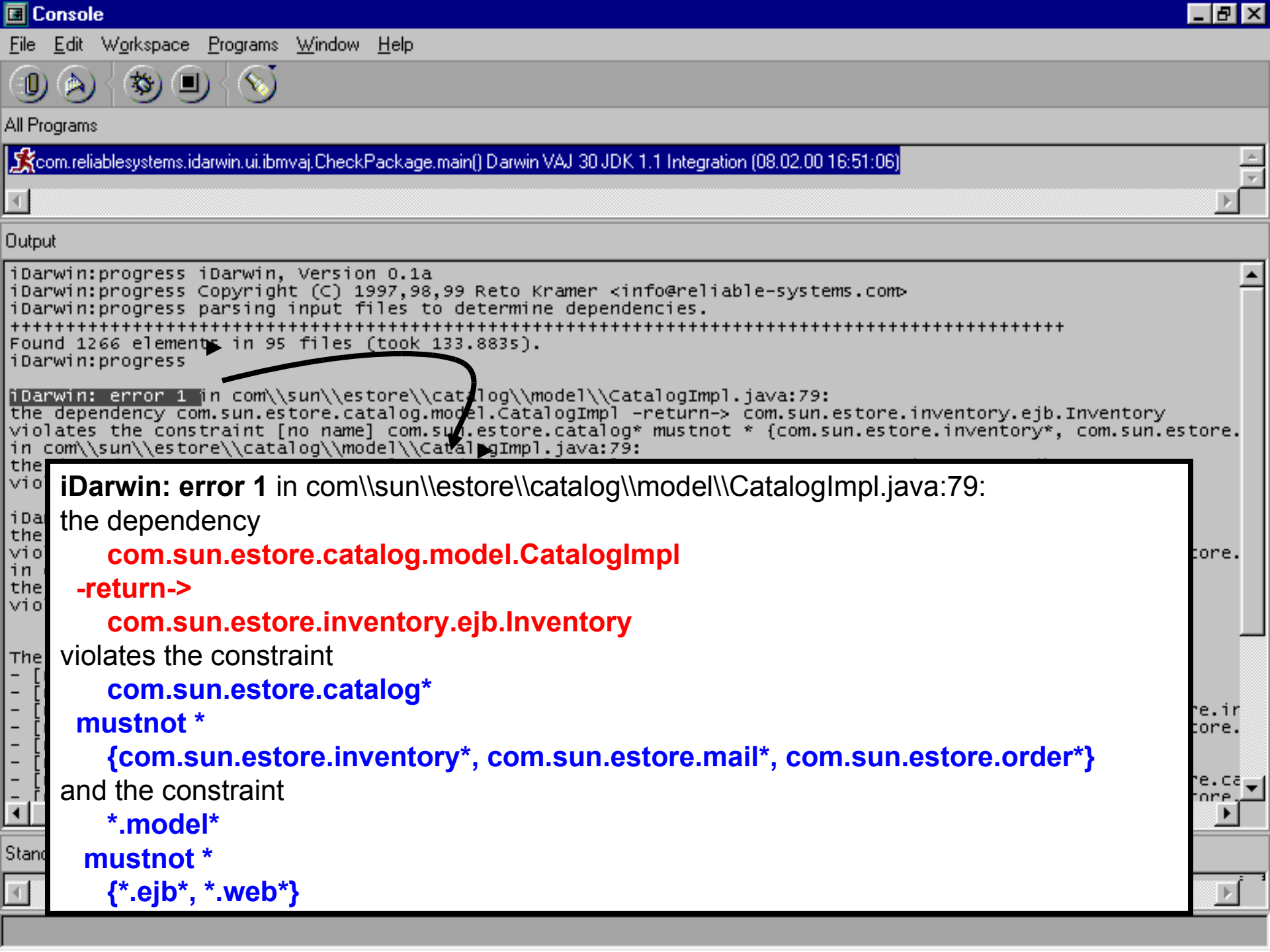
Pet Shop, Exception to Technical Layering

```
// ------------------------------------------------------------
// Technical "layers" shall be LAYERED in general
// ------------------------------------------------------------
//
<macro name=layer>
  {
    *.model*,         // lowest layer comes first
    *.ejb*,
    *.web*            // top layer is last
  }
</macro>

...

<rule name=catalog-exception-2>
      com.sun.estore.catalog.model.CatalogImpl
  may local-var
      com.sun.estore.inventory.ejb.InventoryHome
</rule>
```


DOTTY
DOTTY birdseye view
except
no name
.model
no name
no name
re.catalog.model.CatalogImpl
e.catalog.model.CatalogModel}
com.sun.estore.catalog.model.CatalogImpl
mustnot *
mustnot *
except
except
may local-var
{*.ejb*
, *.web*}
{com.sun.estore.cart*
, com.sun.estore.account*}
com.sun.estore.inventory.ejb.InventoryHome


Workbench
File Edit Workspace Selected Window Help
Open
Open To
Go To
Add
Attribute Filters...
Import...
Export...
Replace With
Delete...
Reorganize
Manage
Compare With
Run
Document
Tools
Properties
CustomerCare AG
External SCM
ReliableSystems - iContract
ReliableSystems - iDarwin
ReliableSystems - iDarwin Utilities
Visualize Architecture (packages)
Visualize Architecture (types)
Check Architecture Conformance (packages)
Check Architecture Conformance (types)
Display Architecture Specification
Find Circular Dependencies
Export Dependency Information (packages) ...
Export Dependency Information (types) ...
Projects
All Problems
All Projects
Comment
SUN J2EE Exam
com.sun.estore.inventory.ejb
com.sun.estore.inventory.model
com.sun.estore.inventory.web
com.sun.estore.mail.ejb
com.sun.estore.order.ejb
com.sun.estore.order.model
com.sun.estore.order.web
com.sun.estore.taglib
com.sun.estore.util
(07.02.00 23:35:51)


DOTTY birdseye view
2 errors:
1 (return)
2 (return)
<<package>>
com.sun.estore.cart.ejb
<<package>>
com.sun.estore.catalog.ejb
<<package>>
com.sun.estore.catalog.web
<<package>>
com.sun.estore.catalog.model


Console
File Edit Workspace Programs Window Help
All Programs
com.reliablesystems.idarwin.ui.ibmvaj.CheckPackage.main() Darwin VAJ 30 JDK 1.1 Integration (08.02.00 16:51:06)
Output
iDarwin:progress iDarwin, Version 0.1a
iDarwin:progress Copyright (C) 1997,98,99 Reto Kramer <info@reliable-systems.com>
iDarwin:progress parsing input files to determine dependencies.
Found 1266 elements in 95 files (took 133.883s).
iDarwin:progress
iDarwin: error 1 in com\\sun\\estore\\catalog\\model\\CatalogImpl.java:79:
the dependency com.sun.estore.catalog.model.CatalogImpl -return-> com.sun.estore.inventory.ejb.Inventory
violates the constraint [no name] com.sun.estore.catalog* mustnot * {com.sun.estore.inventory*, com.sun.estore.
in com\\sun\\estore\\catalog\\model\\CatalogImpl.java:79:
iDarwin: error 1 in com\\sun\\estore\\catalog\\model\\CatalogImpl.java:79:
the dependency
com.sun.estore.catalog.model.CatalogImpl
-return->
com.sun.estore.inventory.ejb.Inventory
violates the constraint
com.sun.estore.catalog*
mustnot *
{com.sun.estore.inventory*, com.sun.estore.mail*, com.sun.estore.order*}
and the constraint
.model
mustnot *
{*.ejb*, *.web*}

Summary

- Low entrance barrier, easy to learn
  - simple concept
  - only 3 language constructs
    - rule
    - group
    - macro
- Scalable to real-world projects
  - so far used up to the order of 100k lines of code

- Easy to use, immediate benefits
  - uses graphical displays
  - seamless VisualAge for Java integration
- Non invasive
  - code is not affected
  - no tool lock in

Next Steps

- Best practices
  - maintenance of "large" evolution specifications
  - specification styles, e.g.
    - p* mustnot * q*
    - p.* may * p.A
      OR
    - p.* mustnot * q*\{p.A}
- Use for UML & ADL
  - extract dependencies from UML models
    (e.g. as Rational Rose add on)
- Use for UML & ADL
  - extract dependencies from ADL (components, connectors)
- Research
  - extend UML with notion of "evolution" and "structural invariant"
  - (define semantics in UML metamodel using OCL)
  - graphical editor for evolution specs

Appendix A
Pet Shop, Complete Evolution Specification

```
<idarwin language=com.reliablesystems.idarwin.specification.impl.primitive_language.xml.XMLRuleLanguage>
// ------------------------------------------------------------
// Business "layers" shall be ISOLATED in general
// ------------------------------------------------------------
//
<group name=all_functional_layers>
  {
    com.sun.estore.account*,
    com.sun.estore.cart*,
    com.sun.estore.catalog*,
    com.sun.estore.inventory*,
    com.sun.estore.mail*,
    com.sun.estore.order*
  }
</group>
<macro name=isolate>
    [all_functional_layers] // reference to previously def group
</macro>
// ------------------------------------------------------------
// Technical "layers" shall be LAYERED in general
// ------------------------------------------------------------
//
<macro name=layer>
  {
    *.model*,  // lowest layer comes first
    *.ejb*,
    *.web*       // top layer is last
  }
</macro>
// ------------------------------------------------------------
// Rules for the taglib package
// ------------------------------------------------------------
//
<rule name=lonely taglib>
     *\com.sun.estore.taglib* mustnot * com.sun.estore.taglib*
</rule>
<rule name=taglib only knows itself and util>
     com.sun.estore.taglib*
  mustnot *
    *\{com.sun.estore.taglib*, com.sun.estore.util*}
</rule>
// ------------------------------------------------------------
// Exceptions to the layer and isolation structure
// ------------------------------------------------------------
// CART
//
<rule name=cart-exception-1>
    com.sun.estore.cart.ejb.ShoppingCartEJB
  may {variable}
    com.sun.estore.catalog.ejb.Catalog
</rule>
<rule name=cart-exception-2>
    com.sun.estore.cart.ejb.ShoppingCartEJB
  may {local-var}
    {com.sun.estore.catalog.model.Item,
     com.sun.estore.catalog.model.Product}
</rule>
```

Appendix A
Pet Shop, Complete Evolution Specification

```
// MAIL
//
<rule name=mail-exception-1>
   com.sun.estore.mail.ejb.MailerEJB
  may local-var
  {
   com.sun.estore.order.ejb.OrderHome,
   com.sun.estore.order.ejb.Order,
   com.sun.estore.account.model.AccountModel,
   com.sun.estore.account.ejb.Account
  }
</rule>
<rule name=mail-exception-2>
   com.sun.estore.mail.ejb.MailerEJB
  may {local-var, parameter}
   com.sun.estore.order.model.OrderModel,
</rule>
// CATALOG
//
<rule name=catalog-exception-1>
   com.sun.estore.catalog.ejb.Catalog
  may return
   com.sun.estore.inventory.ejb.Inventory
</rule>
<rule name=catalog-exception-1>
  {com.sun.estore.catalog.model.CatalogImpl,
   com.sun.estore.catalog.model.CatalogModel}
  may return
   com.sun.estore.inventory.ejb.Inventory
</rule>
<rule name=catalog-exception-2>
   com.sun.estore.catalog.model.CatalogImpl
  may local-var
   com.sun.estore.inventory.ejb.InventoryHome
</rule>
// ORDER
//
<rule name=order-exception-1>
   {com.sun.estore.order.ejb.OrderEJB,
    com.sun.estore.order.ejb.Order}
  may return
   com.sun.estore.account.ejb.Account
</rule>

<rule name=order-exception-1>
   com.sun.estore.order.ejb.OrderEJB
  may local-var
   com.sun.estore.account.ejb.AccountHome
</rule>

</idarwin>
```

More on iDarwin ...

Visit the iDarwin homepage at:
http://www.reliable-systems.com