EclipeCon 2009 – Long Talk:

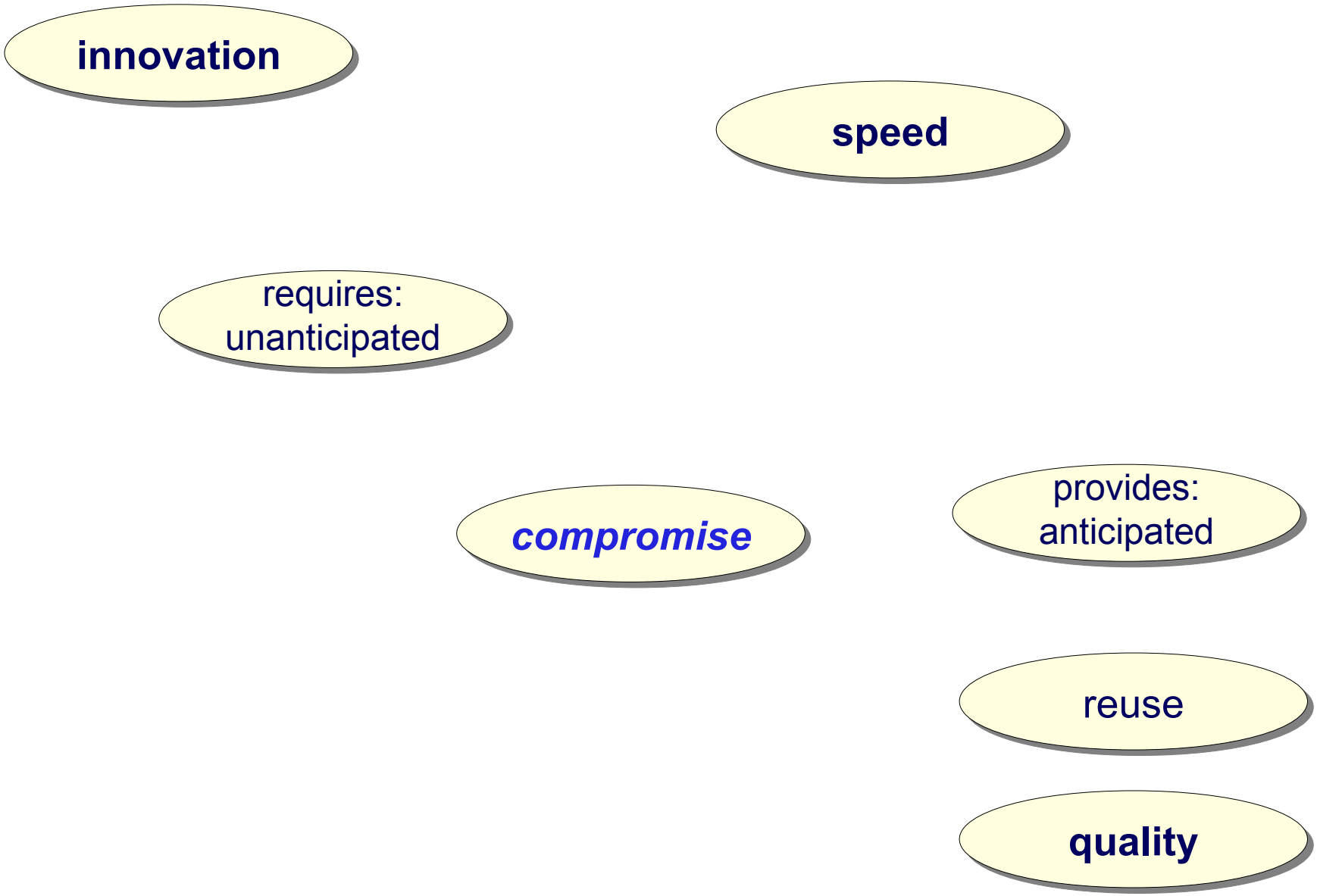
Plugin reuse and adaptation with Object Teams:

Don't settle for a compromise!

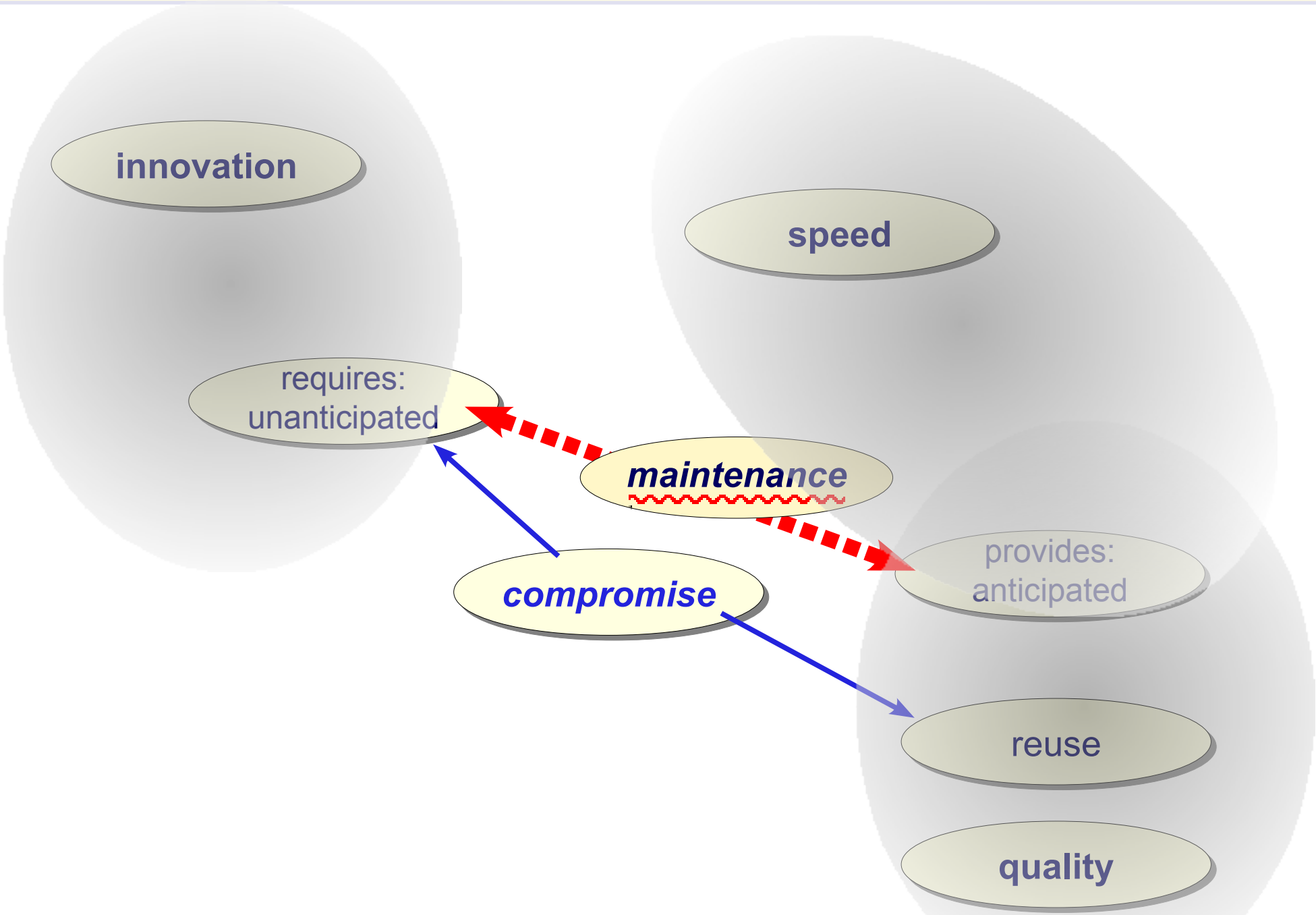
Stephan Herrmann
Technische Unversität Berlin

www.objectteams.org

The Game


innovation
speed
requires:
unanticipated
compromise
provides:
anticipated
reuse
quality

The Game


innovation
speed
requires:
unanticipated
maintenance
provides:
anticipated
compromise
reuse
quality

Solution

- With **Object Teams** it is possible
  - to eat the cake
    apply unanticipated adaptations
  - and still have it
    sustain a well modularized, maintainable design
- **OT/Equinox** brings this power
  to the development of Eclipse plugins

Unanticipated Adaptation

- Object oriented adaptation: Inheritance
  - program by difference
  - choose at instantiation time
- This is good, but:
  - who controls instantiation?
  - behavioral changes after instantiation?
  - multiple (independent) adaptations?
- We need something similar to inheritance
  - apply to objects not classes:
  - adapt any time, any number

OT/J based Architecture

team
collaboration module

role
members of a team

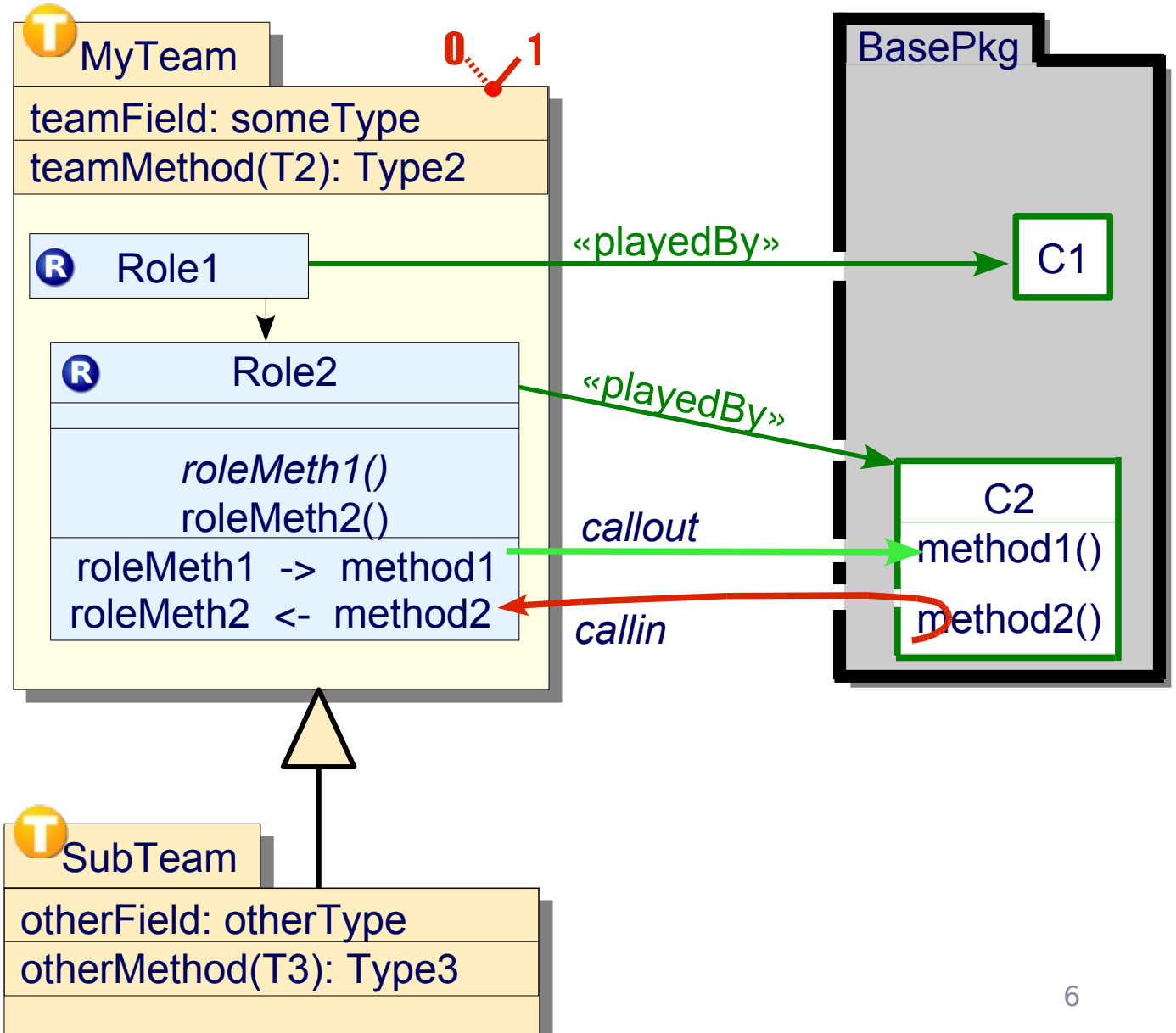
playedBy
connect role to base

callout
forward to base

callin
intercept base method

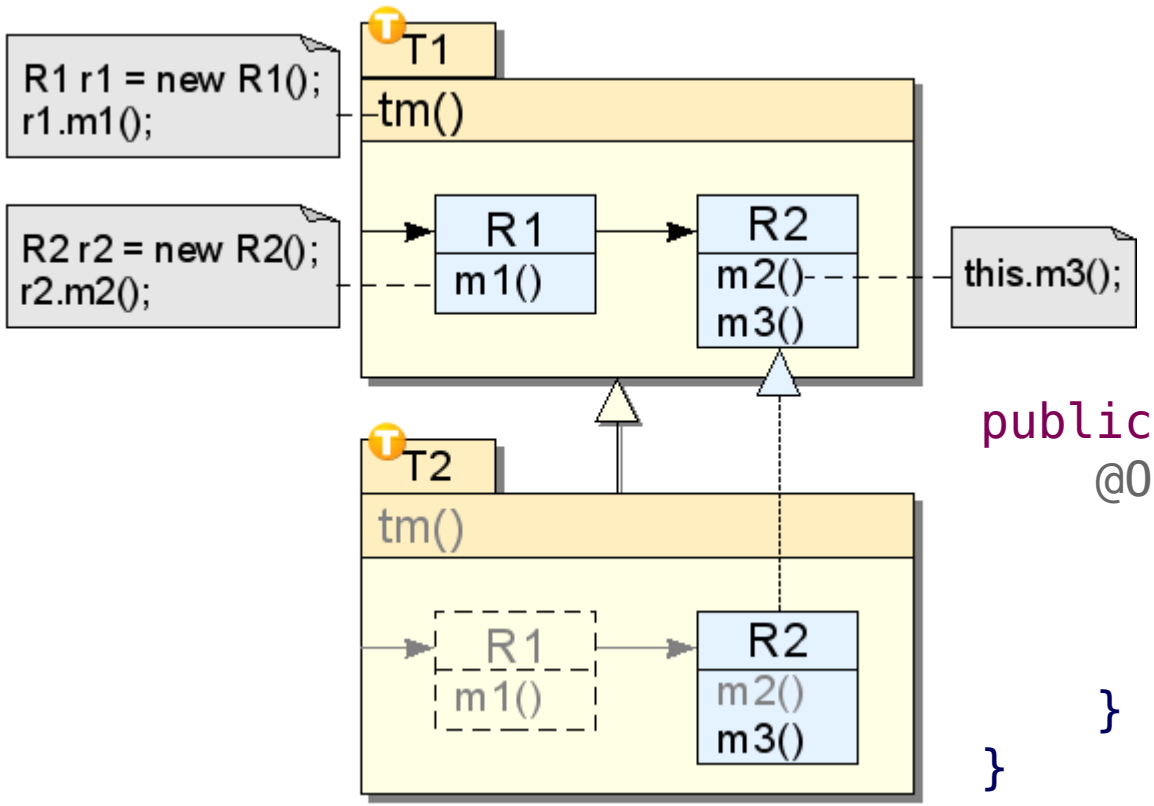
decapsulation
break base encapsulation

(de)activation
dis/enable all callins



OT/J based Architecture

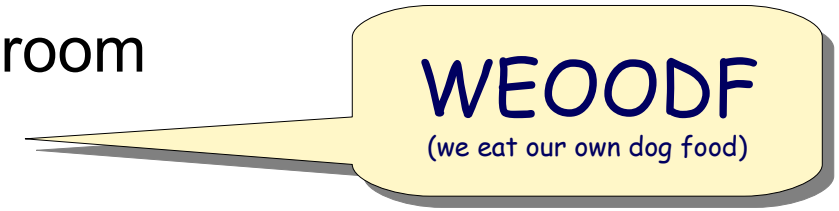
- Inheritance of complex structures
  - virtual methods and classes
  - propagating specialization



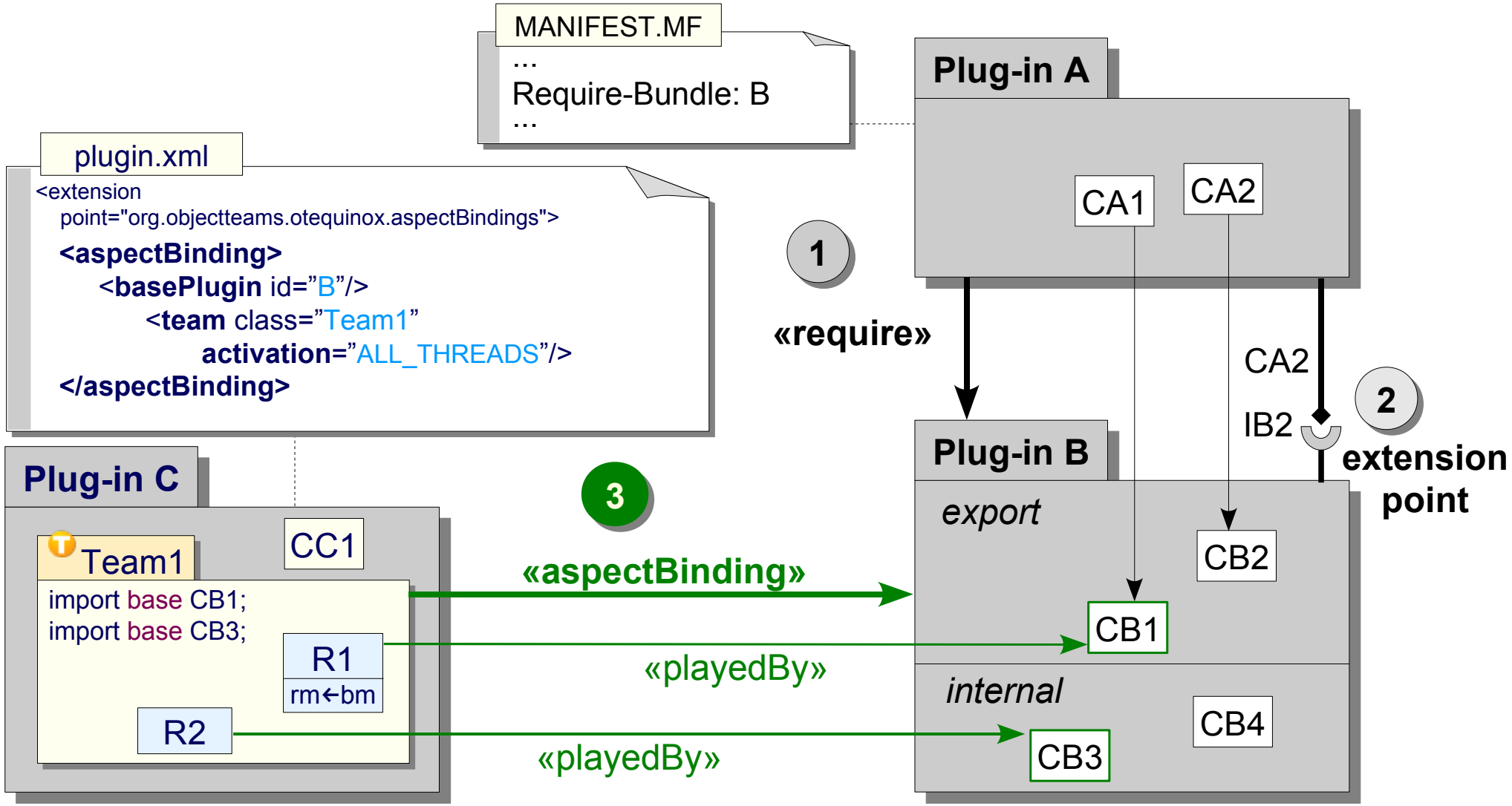

```
public team class T2 extends T1 {
    @Override protected class R2 {
        @Override void m3() {
            doMyStuff();
        }
    }
}
```

Object Teams

- ObjectTeams/Java (OT/J) — since 2001
  - Java += roles, teams, aspect bindings
- Object Teams Development Tooling — since 2003
  - Java Compiler += OT/J constructs
  - JDT for OT/J (code assist, ui, launch ...)
- OT/Equinox — since 2006
  - Equinox += aspect bindings
- Application
  - Case studies (project TOPPrax)
  - Class room
  - OTDT



Plug-in relationships



Demo 1

OT/Equinox Hello World

- Create a plugin that
  - makes “Foo” an illegal Java type name
    - can no longer do demos
    - “AntiDemo”
- Prerequisite
  - Have identified a join point in
  - `org.eclipse.jdt.core.JavaConventions.validate`*`XYZ`*`()`

Demo 1 – Summary

- Development of OT Plug-ins
  - Wizards, validation, content assist, ...
  - Minimal: 1 aspect binding, 1 team, 1 role, 1 callin
- Running Eclipse with OT/Equinox
  - ☑ `Enable OT/Equinox`
- Inside Eclipse
  - About Plug-ins
    - inspect which plug-ins are adapted
  - OT/Equinox Monitor
    - inspect active team instances
    - dynamically (de)activate

one more lesson we learned ...

Demo 1 – Summary

- Development of OT Plug-ins
  - Wizards, validation, content assist, ...
  - Minim… 1 callin
- Running I…
  - Enable…
- Inside Ec…
  - About Pl…
    - inspec…
  - OT/Equi…
    - inspec…
    - dynamically (de)activate…

Adapting the core

– not the symptoms –

improves consistency

one more lesson we learned ...

OTDT Example 1

- Check this consistency constraint

> *”A plug-in that defines aspect bindings with team activation must have an activation policy (lazy).“*

- Could write a complete own validator
  - parse plugin.xml and MANIFEST.MF
  - report errors
  - offer quickfix
  - get triggered on file changes
- Easier: reuse PDE – piggy-back implementation
  - find join points
  - (just) implement the above rule

Bundle Validation – Join points

- Found two interesting methods in `org.eclipse.pde.internal.core.builders`
  - `ExtensionsErrorReporter.validateExtension(Element)`
    - invoked for each extension in `plugin.xml`
    - can be used as trigger for detected aspect bindings
  - `BundleErrorReporter.validateBundleActivatorPolicy()`
    - specifically checks the activation policy in `MANIFEST.MF`
    - can be used to add our validation

Bundle Validation – Join points

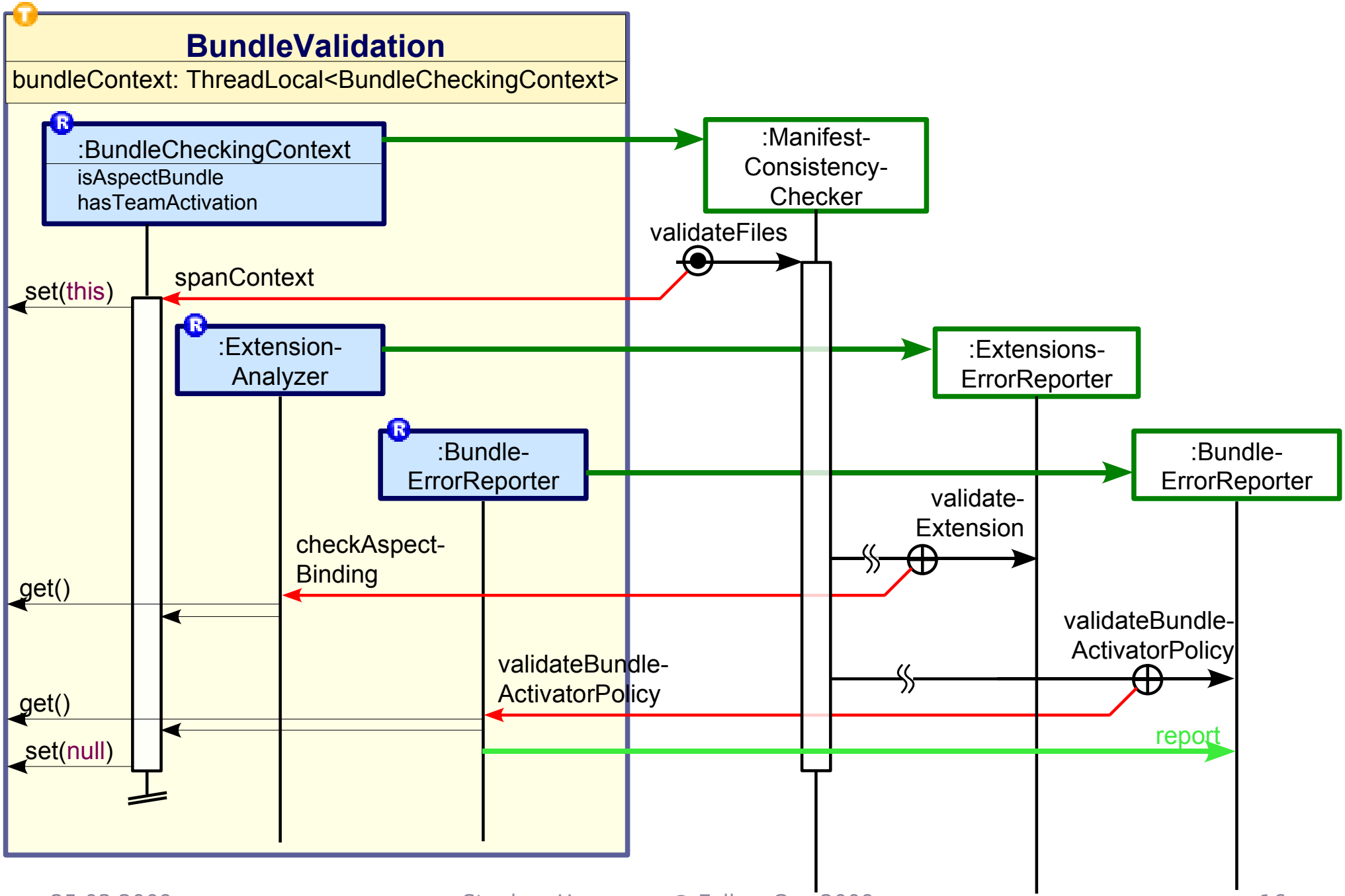
- Found two interesting methods in `org.eclipse.pde.internal.core.builders`
  - ExtensionsErrorReporter

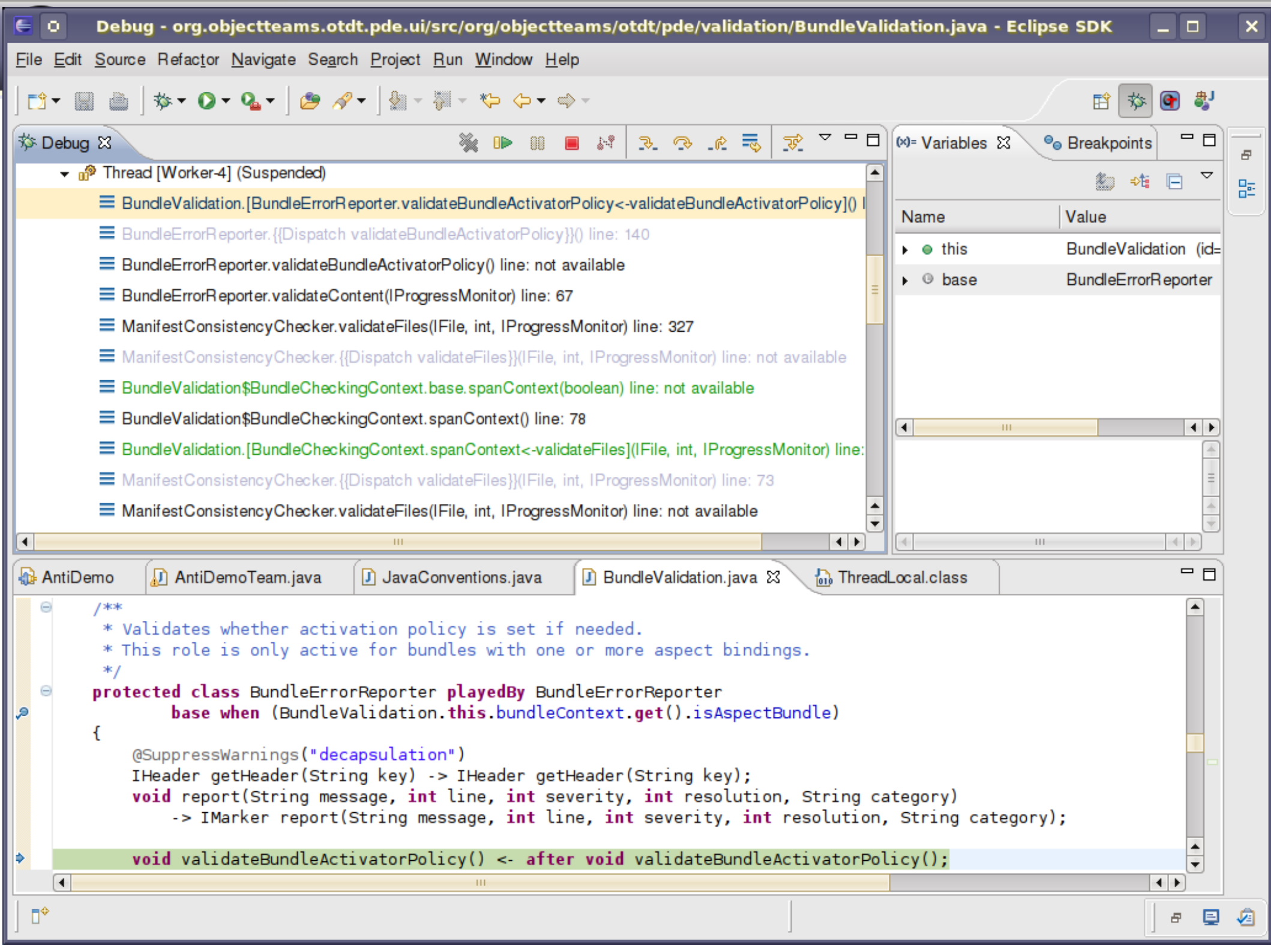
```
protected class ExtensionAnalyzer playedBy ExtensionsErrorReporter
{
    void checkAspectBinding(Element element) <- after void validateExtension(Element element);
    …
}
protected class BundleErrorReporter playedBy BundleErrorReporter
{

    void validateBundleActivatorPolicy() <- after void validateBundleActivatorPolicy();
    …
}
```

  - can be used to add our validation

- But: those two objects don't know each other!
  - what is the commen context of both join points?

Bundle Validation – Context




Debug - org.objectteams.otdt.pde.ui/src/org/objectteams/otdt/pde/validation/BundleValidation.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Thread [Worker-4] (Suspended)
BundleValidation.[BundleErrorReporter.validateBundleActivatorPolicy<-validateBundleActivatorPolicy]() l
BundleErrorReporter.{{Dispatch validateBundleActivatorPolicy}}() line: 140
BundleErrorReporter.validateBundleActivatorPolicy() line: not available
BundleErrorReporter.validateContent(IProgressMonitor) line: 67
ManifestConsistencyChecker.validateFiles(IFile, int, IProgressMonitor) line: 327
ManifestConsistencyChecker.{{Dispatch validateFiles}}(IFile, int, IProgressMonitor) line: not available
BundleValidation$BundleCheckingContext.base.spanContext(boolean) line: not available
BundleValidation$BundleCheckingContext.spanContext() line: 78
BundleValidation.[BundleCheckingContext.spanContext<-validateFiles](IFile, int, IProgressMonitor) line:
ManifestConsistencyChecker.{{Dispatch validateFiles}}(IFile, int, IProgressMonitor) line: 73
ManifestConsistencyChecker.validateFiles(IFile, int, IProgressMonitor) line: not available
Variables
Breakpoints
Name
Value
this
BundleValidation (id=
base
BundleErrorReporter
AntiDemo
AntiDemoTeam.java
JavaConventions.java
BundleValidation.java
ThreadLocal.class
/**
* Validates whether activation policy is set if needed.
* This role is only active for bundles with one or more aspect bindings.
*/
protected class BundleErrorReporter playedBy BundleErrorReporter
base when (BundleValidation.this.bundleContext.get().isAspectBundle)
{
@SuppressWarnings("decapsulation")
IHeader getHeader(String key) -> IHeader getHeader(String key);
void report(String message, int line, int severity, int resolution, String category)
-> IMarker report(String message, int line, int severity, int resolution, String category);
void validateBundleActivatorPolicy() <- after void validateBundleActivatorPolicy();

Team – Context – View

- A **team** defines a **view**
  - map only relevant elements
    - classes using playedBy
    - events using callin
    - provided actions using callout
  - team & roles is a self-contained world
- A **team** and its roles define **context**
  - store context specific state
  - context can be defined ...
    - by a team instance
    - by a role instance / a graph of ...
    - per thread
    - per control flow
    - ...

feature specific

roles may
- start disconnected
- discover each other later
- superimpose structure

Large-Scale Control-Flow Context

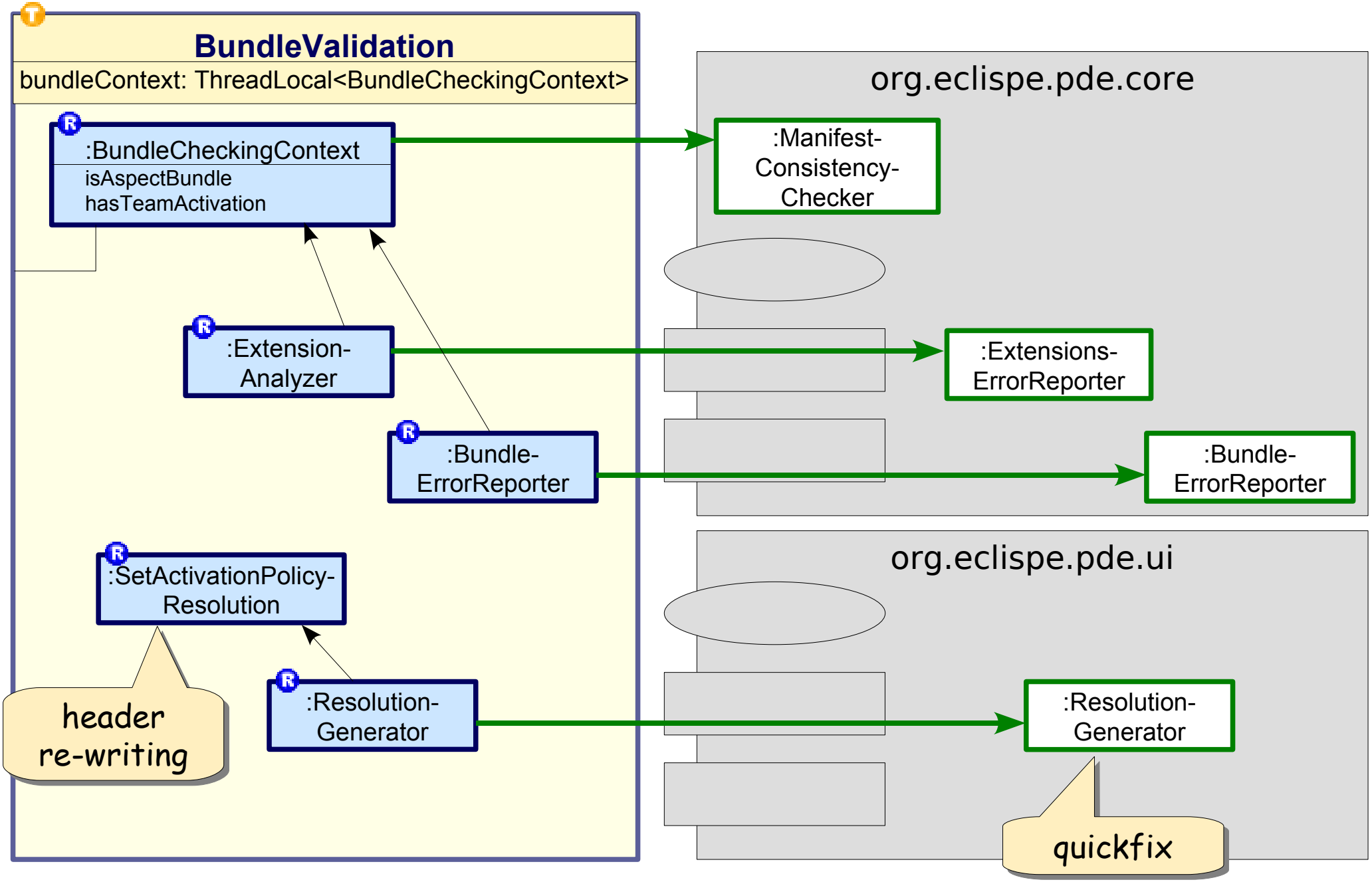
- A complex features applies only conditionally
  - identify the relevant situation with a control-flow
  - define two teams
    - **feature team** is inactive by default
    - **guard team**
      - observes initial trigger
      - instantiates and activates feature team
    - use this pattern:

```
trigger ← after someEvent;
callin void trigger() {
    within (new FeatureTeam()) {
        base.trigger();
    }
}
```

temporary per-thread activation

    - thread safe
    - exception safe

Multi Plugin Coordination



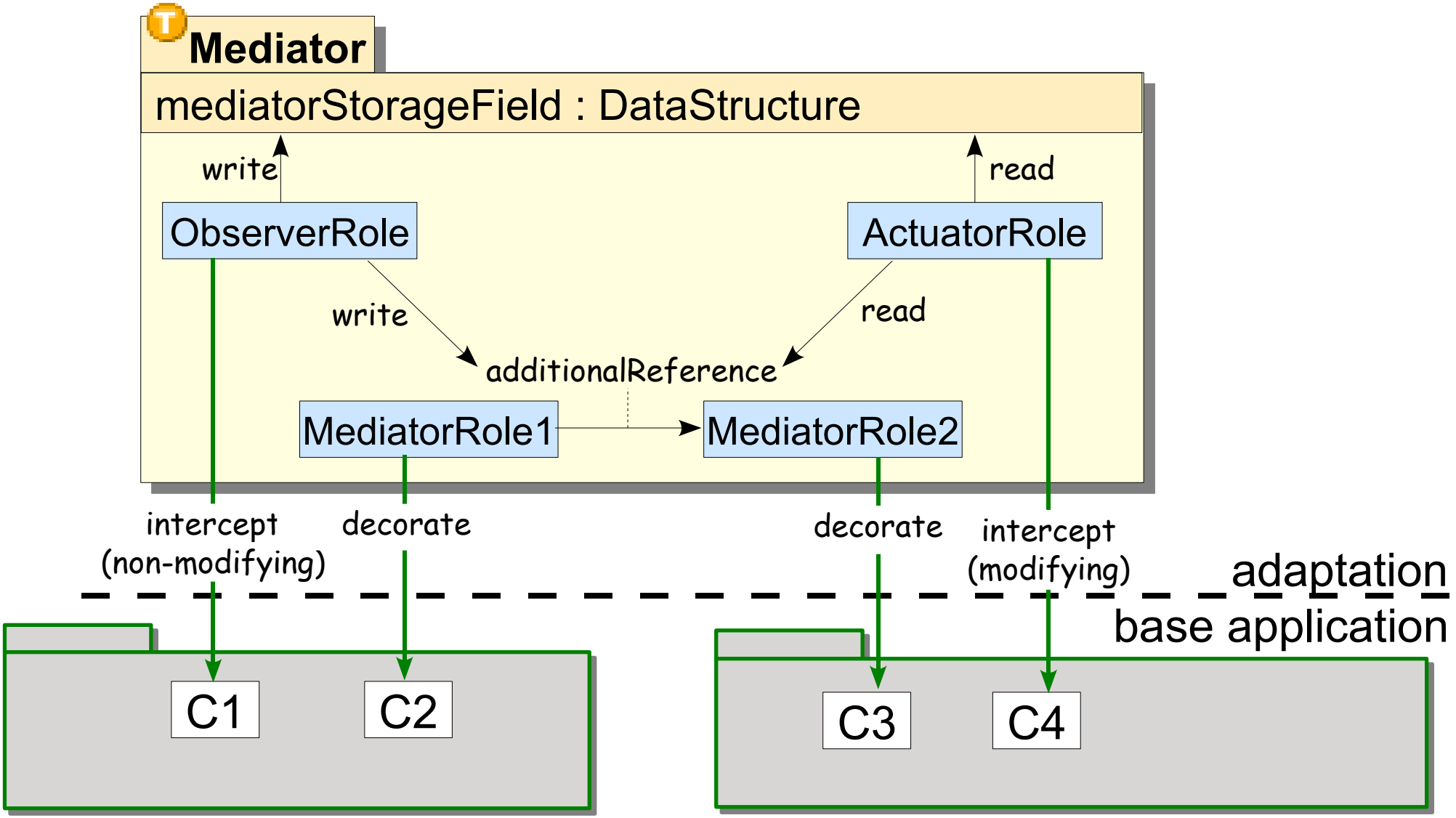
From-Scratch vs. Piggy-Back

- Don't bother with ...
  - locating files (plugin.xml and MANIFEST.MF)
  - parsing XML and manifest syntax
  - mechanics of re-writing the manifest
  - receiving triggers on file changes

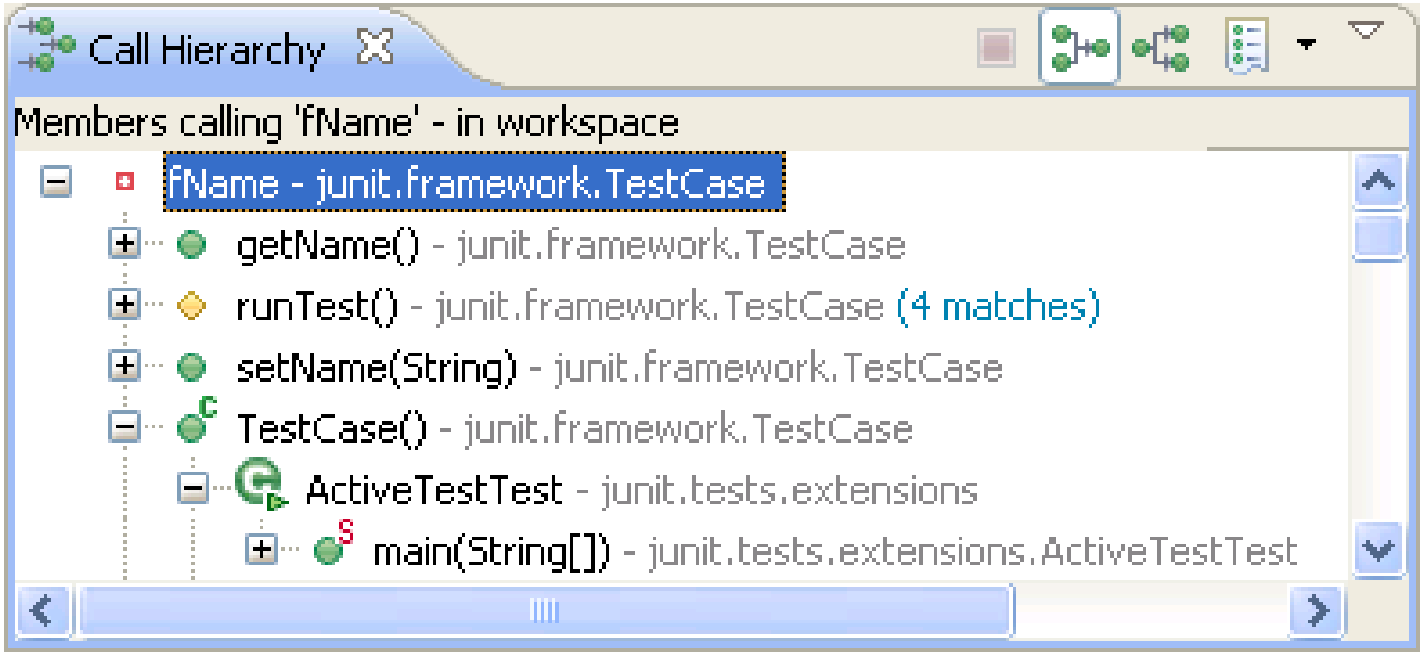
  All this is already implemented => re-use it!!
- Only implement the net value
  - less code to write
  - less code to read
  - it's all in one place
- Find join points to hook into
  - this task is new

Observer-Mediator-Actuator

Abstracting to a general pattern



Example Call Hierarchy

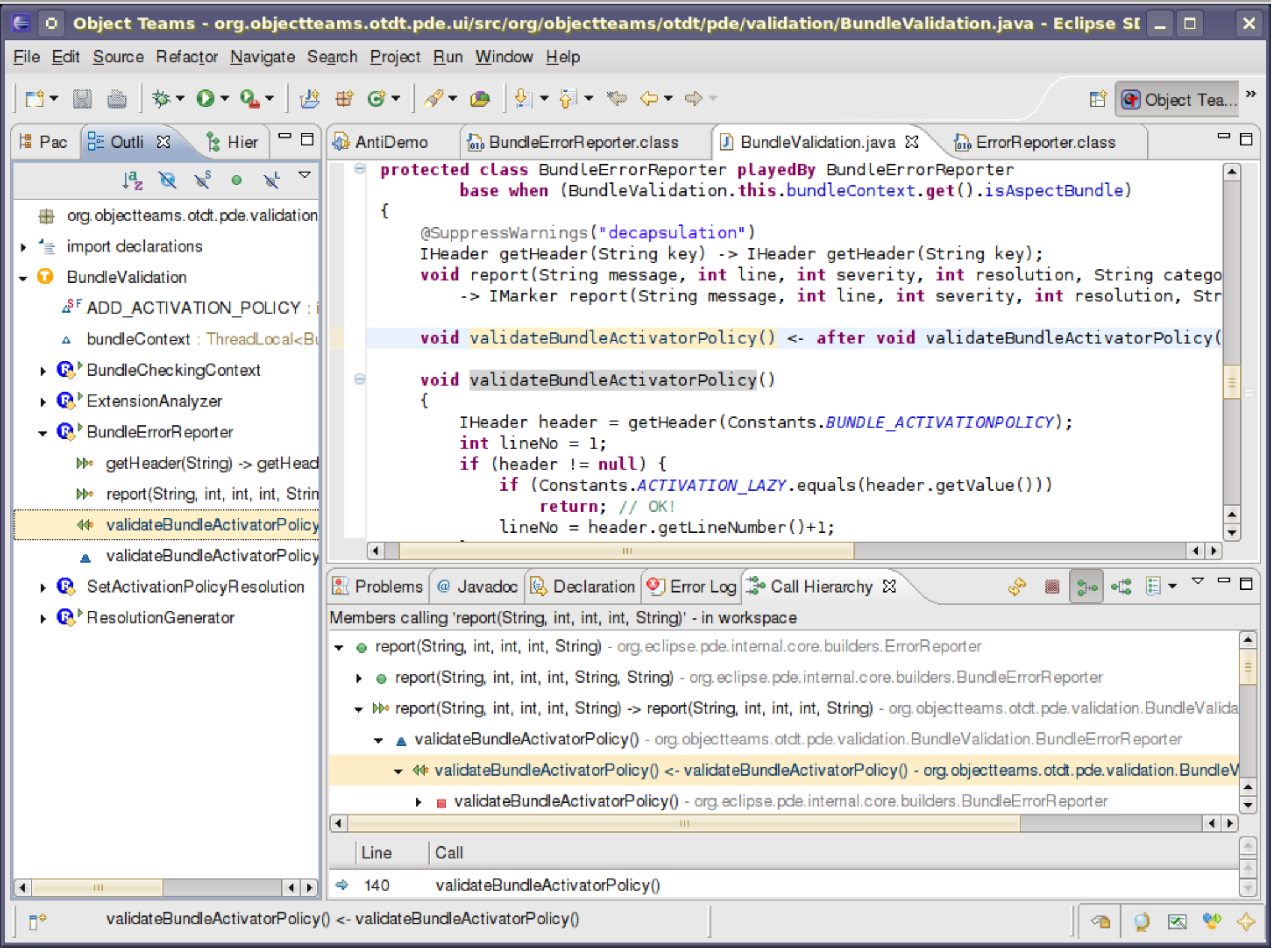
- From 3.4 New&Noteworthy
  - call hierarchy view works with fields ...
  - 

Example Call Hierarchy

- From 3.4 New&Noteworthy
  - call hierarchy view works with fields ...
  - first released in the OTDT
  - later refactored and contributed to Eclipse
- More control flows: callin & callout
  - prerequisite: can already search for callin & callout bindings

  ui part: adaptation using ...
  - 1 team adapting 1 base plugin (org.eclipse.jdt.ui)
  - 6 roles adapting 6 base classes
  - 4 callin bindings, 12 callout bindings

  365 LOC


Object Teams - org.objectteams.otdt.pde.ui/src/org/objectteams/otdt/pde/validation/BundleValidation.java - Eclipse SI
File Edit Source Refactor Navigate Search Project Run Window Help
Object Tea...
Pac
Outli
Hier
org.objectteams.otdt.pde.validation
import declarations
BundleValidation
ADD_ACTIVATION_POLICY : i
bundleContext : ThreadLocal<Bu
BundleCheckingContext
ExtensionAnalyzer
BundleErrorReporter
getHeader(String) -> getHead
report(String, int, int, int, Strin
validateBundleActivatorPolicy
validateBundleActivatorPolicy
SetActivationPolicyResolution
ResolutionGenerator
AntiDemo
BundleErrorReporter.class
BundleValidation.java
ErrorReporter.class
protected class BundleErrorReporter playedBy BundleErrorReporter
base when (BundleValidation.this.bundleContext.get().isAspectBundle)
{
@SuppressWarnings("decapsulation")
IHeader getHeader(String key) -> IHeader getHeader(String key);
void report(String message, int line, int severity, int resolution, String catego
-> IMarker report(String message, int line, int severity, int resolution, Str
void validateBundleActivatorPolicy() <- after void validateBundleActivatorPolicy(
void validateBundleActivatorPolicy()
{
IHeader header = getHeader(Constants.BUNDLE_ACTIVATIONPOLICY);
int lineNo = 1;
if (header != null) {
if (Constants.ACTIVATION_LAZY.equals(header.getValue()))
return; // OK!
lineNo = header.getLineNumber()+1;
Problems
Javadoc
Declaration
Error Log
Call Hierarchy
Members calling 'report(String, int, int, int, String)' - in workspace
report(String, int, int, int, String) - org.eclipse.pde.internal.core.builders.ErrorReporter
report(String, int, int, int, String, String) - org.eclipse.pde.internal.core.builders.BundleErrorReporter
report(String, int, int, int, String) -> report(String, int, int, int, String) - org.objectteams.otdt.pde.validation.BundleValida
validateBundleActivatorPolicy() - org.objectteams.otdt.pde.validation.BundleValidation.BundleErrorReporter
validateBundleActivatorPolicy() <- validateBundleActivatorPolicy() - org.objectteams.otdt.pde.validation.BundleV
validateBundleActivatorPolicy() - org.eclipse.pde.internal.core.builders.BundleErrorReporter
Line
Call
140
validateBundleActivatorPolicy()
validateBundleActivatorPolicy() <- validateBundleActivatorPolicy()

Example Call Hierarchy

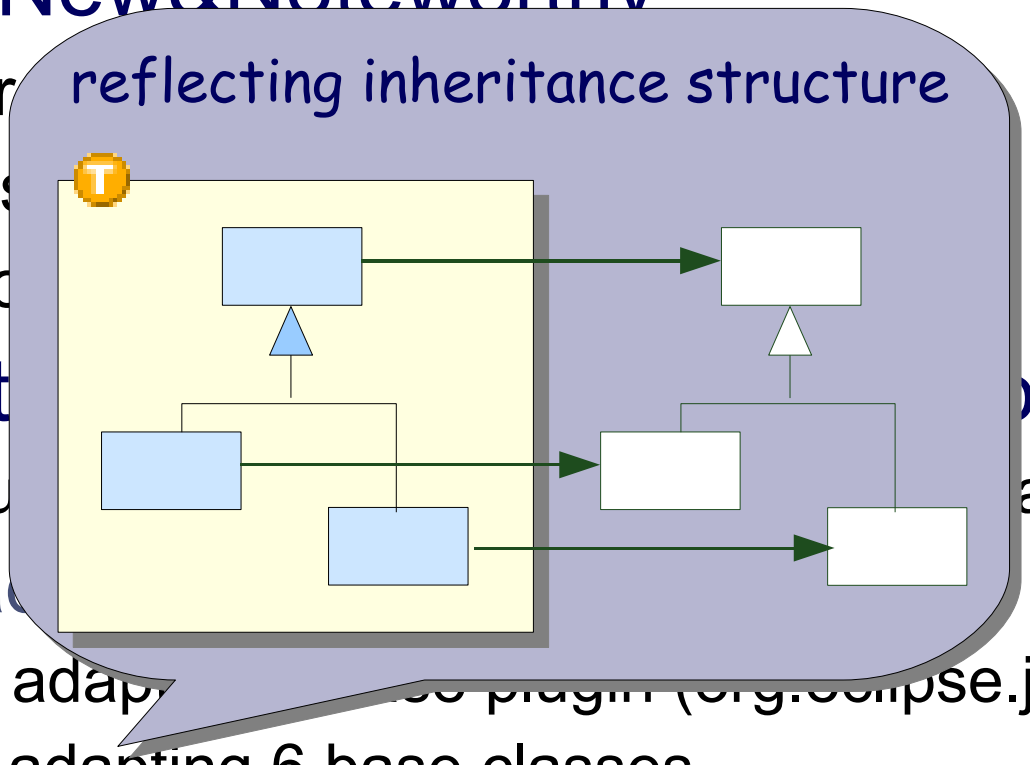
- From 3.4 New&Noteworthy
  - call hierarchy view works with fields ...
  - first released in the OTDT
  - later refactored and contributed to Eclipse
- More control flows: callin & callout
  - prerequisite: can already search for callin & callout bindings

  ui part: adaptation using ...
  - 1 team adapting 1 base plugin (org.eclipse.jdt.ui)
  - 6 roles adapting 6 base classes
  - 4 callin bindings, 12 callout bindings

  365 LOC
- Future: declaratively induced control flows
  - plugin activator: → `MyActivator.start()`
  - aspect binding: → `new MyTeam()`

Example Call Hierarchy

- From 3.4 New&Noteworthy
  - call hierar
  - first releas
  - later refac
- More cont ut
  - prerequ allout
  - ui part: a
    - 1 team adap se.jdt.ui)
    - 6 roles adapting 6 base classes
    - 4 callin bindings, 12 callout bindings



365 LOC

- Future: declaratively induced control flows
  - plugin activator: → `MyActivator.start()`
  - aspect binding: → `new MyTeam()`

Example: Launching with OT/J

- First approach:
  new launch configuration type
  - replace Main
  - augment classpath
  - change program arguments
- Duplicate for each of
  - JUnit Test launches
  - Eclipse Application launches
  - OSGi Framework launches
  - JUnit Plugin Test launches
  - ...

Example: Launching wih OT/J

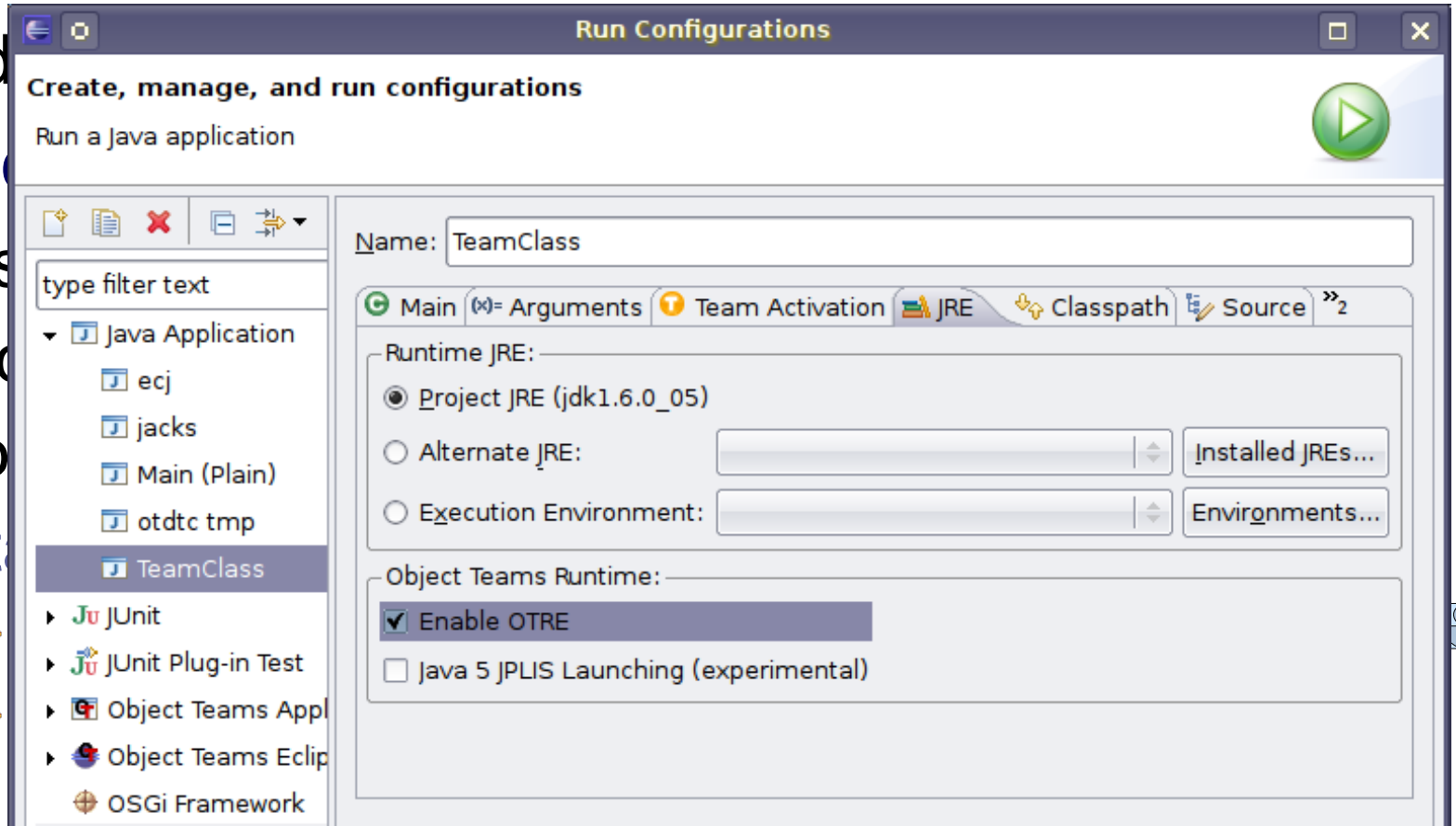
- Second approach:
  add OT-capability to existing launches
  - new checkboxes next to JRE selection
    - defaults from the project context
  - add team activation tab
  - adapt the command line behind the scenes
- Benefits
  - less code
  - more consistency
  - composable launches

  Stats:
    - adapting 7 plugins
    - 8 teams, 16 roles, 660 LOC

Adapting the core
- not the symptoms -
improves consistency

Example: Launching wih OT/J

- Second approach:
  add OT-capability to existing launches
  - new checkboxes next to JRE selection
    - defaults from the project context
  - add team activation tab
  - ad
- Ben
  - les
  - mo
  - co
    St
    -
    -



Evolution / Maintenance

- Meaningful moduls
  - comprehensable: only one feature
  - comprehensive: all of one feature
- Join points may change – seldomly
  - mostly “syntactical” changes: ~ refactoring
  - very few “semantical” changes
    - join point called differently
    - need to refine detection of relevant situation
    - OT provides sufficient means
  - embrace change with agility
- OTDT built on OT/Equinox since 2006
  - migrating the OT-plugins: minimal effort

Conclusion

- Find something similar to what you need?
  - use it! adapt it to exactly your requirements!
  - near miss is no excuse!
  - make the adaptation a meaningful module!
  - piggy-back adaptation where suitable!
  - feature = module = context = team!
  - quickly innovate!

No Compromise!

- Coming next
  - combining OT with generative techniques
    - modeling (e.g., UML 2 tools)
    - IMP?