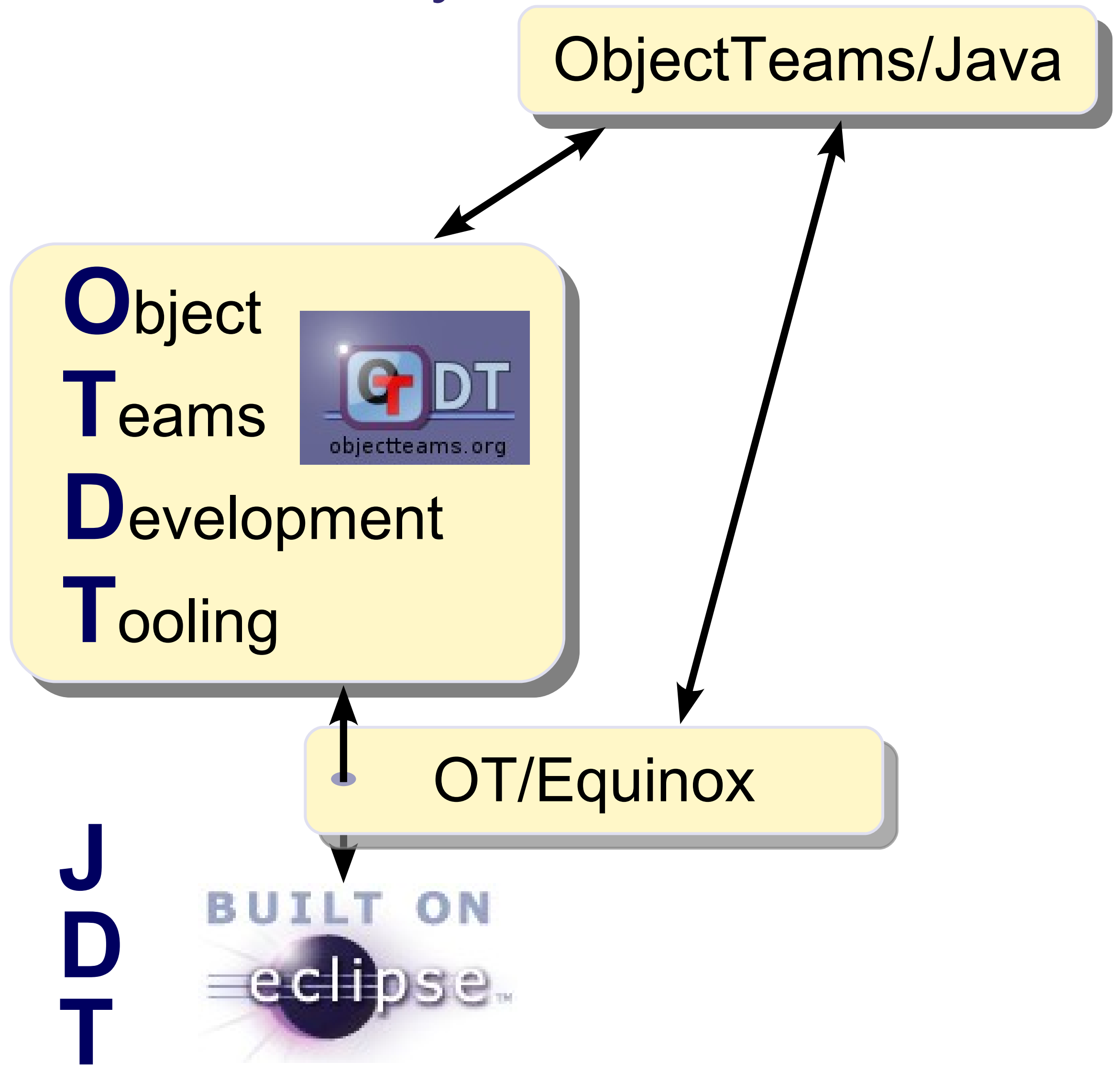
The Object Teams Development Tooling: A high fidelity extension of the JDT

Stephan Herrmann
Technische Universität Berlin

The Characters in this Play

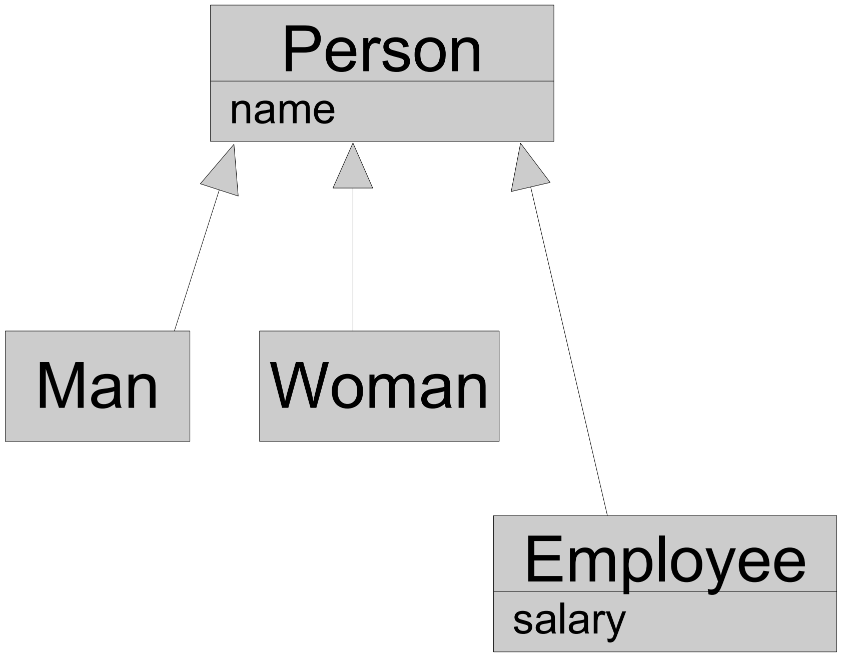


OO is not the end of language development

- E.g., **inheritance** is great, but ...

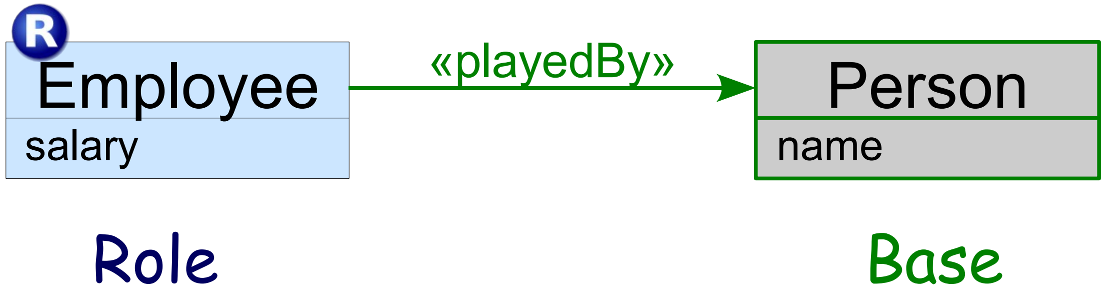
  <u>A text book example:</u>

  - A man/woman **is a** person, OK
  - An employee **is a** person, OK?
    - Born as an employee?
    - Dying when loosing the job?
    - Several jobs, yet only one salary?
- Whats wrong with inheritance?
  - Missing “become”, “quit” ☹
  - Can't duplicate fields ☹
  - Employee & Person = 1 instance ☹
- Can we do better?
  - Yes:
  - Employee is a Role played by a Person

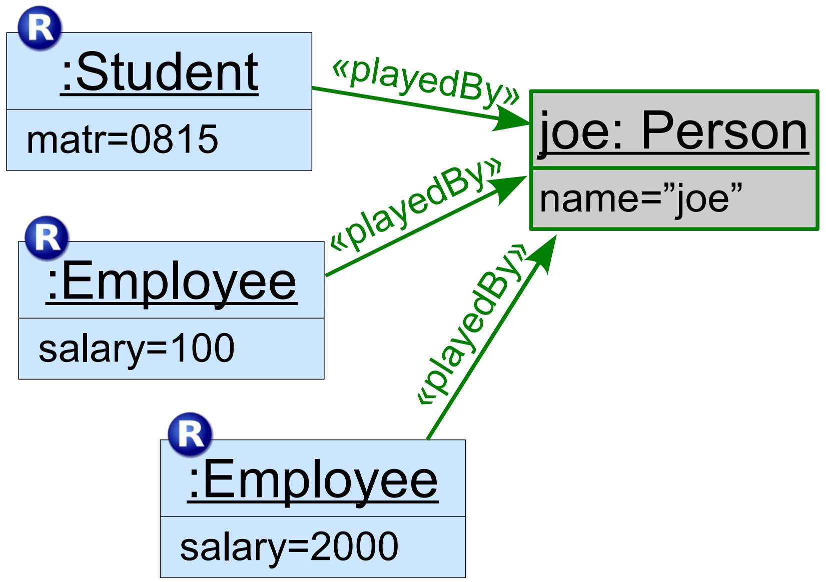


ObjectTeams/Java

playedBy relationship



- Advantages:
  - **Dynamism**:
    roles can come and go
    (same base object)
  - **Multiplicities:**
    one base can play several roles
    (different/same role types)
- Is this completely new?
  - No, has been around >15 years
  - **playedBy** is similar to **extends**
  - how similar?



Comparing at a closer look

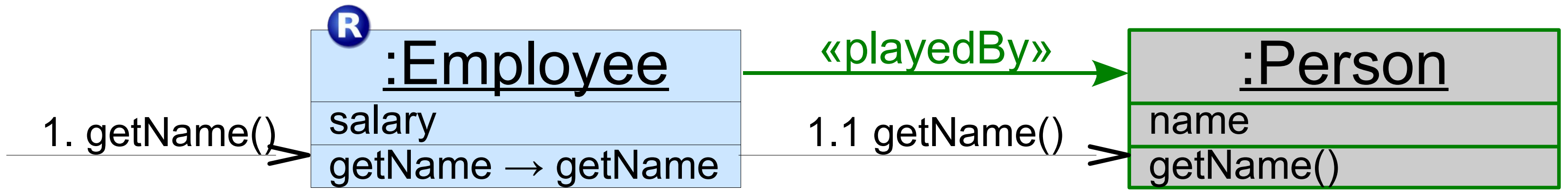
- Inheritance
  - Import / acquisition
    - **dispatch sub → super**
  - Overriding
    - **dispatch super → sub**
  - Subtype polymorphism
    - **substitutability**

?

- Role Playing

?

Import in OT/J:



- A **callout method binding** ...

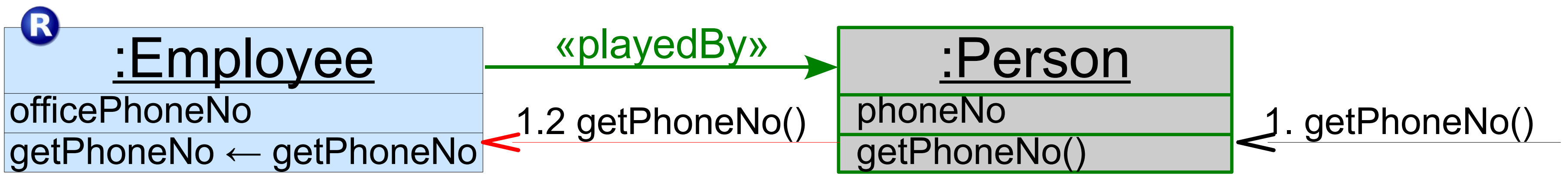
```
String getName() -> String getName();
```

... declares that calls to the role should be **forwarded** to its base

- ... can use different names on role / base sides
- ... can adjust signatures
  - implicitly: discard unused values
  - explicitly: parameter mappings

ObjectTeams/Java

Overriding in OT/J:



- A **callin method binding** ...

```
String getPhoneNo() <- replace String getPhoneNo();
```

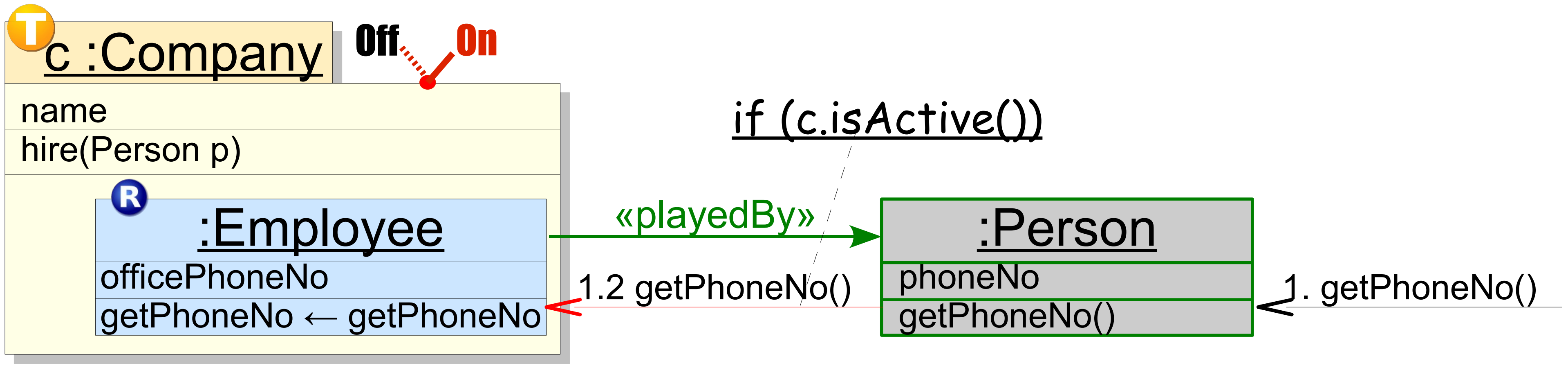
... declares that calls to the base should be **intercepted** by its role

- ... can use different names on role / base sides
- ... can adjust signatures
  - implicitly: discard unused values
  - explicitly: parameter mappings
- ... comes in one of three flavors: `before`, `replace` or `after`

ObjectTeams/Java

"... when (s)he is in the office ..."

- *How do you know?*
- **Roles depend on context**
- In OT/J contexts are reified as **Teams**
  - roles are inner classes of a **team class**
  - role instances are inner instances of a **team instance**
- Each team instance can be **(de)activated**
  - (several mechanisms: globally, per thread, implicitly, temporarily ...)



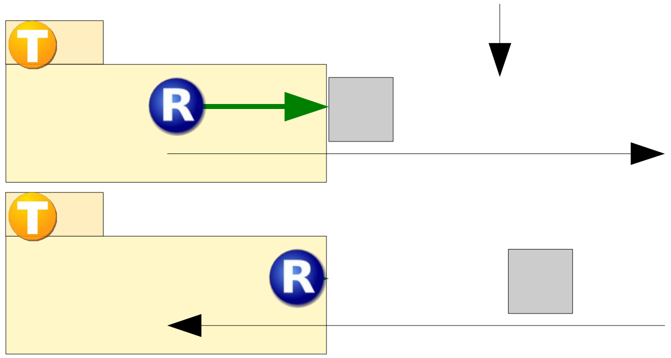
Substitutability?

- Are the following assignments legal?

```
   Employee emp = ...
   Person person = ...
1. person = emp;        // legal?
2. emp = person;        // legal?
```

Normally not, but...

- *Idea: roles live (usually) only within the team*
- When a role object **leaves** the team
  - it is **lowered** to its base
- When a base object **enters** a team
  - it can be **lifted** to a role



Comparison

• Inheritance	• Role Playing
◆ Import / acquisition ▪ **dispatch sub → super**	◆ Import / acquisition ▪ **dispatch role → base**
◆ Overriding ▪ **dispatch super → sub**	◆ Overriding ▪ **dispatch base → role**
◆ Subtype polymorphism ▪ **substitutability** ✦ **pass an instance of sub class where the super class is expected**	◆ Translation polymorphism ▪ lowering role → base ▪ lifting base → role ▪ **two-way substitutability!**

Summary ObjectTeams/Java

- Role playing combines the powers of inheritance with
  - **Dynamism**:
    roles can come and go (same base object)
  - **Multiplicities:**
    one base can play several roles (different/same role types)
- Teams
  - **encapsulate** a set of interacting roles
  - team **activation** controls the effect of all contained **callin** bindings
  - create **larger structures** *(stacking / nesting / layering)*

Tooling for ObjectTeams/Java

- Goal: “A **high-fidelity** extension of the JDT”
  - The same quality (code & user experience)
  - Unrestricted Java-development
  - Seamless support for specifics of OT/J
- Need to cover (minimum):
  - Compiler
  - Run/Debug
  - Help
  - Editor
  - Code Assist
  - Refactoring
  - Wizards
  - Structure Viewers
  - Search / Call Hierarchy
  - Integration with PDE
- Goal: “A high-fidelity **extension** of the JDT”
  - Maximum reuse
  - Maintainable
  - Evolvable

Working with the OTDT

<DEMO 1> Summary:

• Run	as Object Teams Application
◆ configuration	instantiate & activate a Team
• Wizard:	created role with `playedBy`
• Editor	seamlessly highlight etc.
• Compiling	seamlessly: incremental, eager ...
• Code assist	create OT/J elements apply std. assists to OT/J
• Help!	link problems to language definition

Working with the OTDT

<DEMO 1> Summary:

- Run
  - configuration
- Wizard:
- Editor
- Compiling
- Code assist
- Help!

- Seamless integration
  - Object Teams Runtime Environment
  - Run as Object Teams Application
- Additional configuration
  - Team Activation

Working with the OTDT

<DEMO 1> Summary:

- Run
  - configuration
- Wizard:
- Editor
- Compiling
- Code assist

- Help!

- Project creation
  - Object Teams Project
  - Object Teams Plugin Project
- Class creation
  - Team
  - Role playedBy Base
    - inline
    - role file

Working with the OTDT

<DEMO 1> Summary:

- Run
  - configuration
- Wizard:
- Editor
  - Syntax highlighting
    - includes other views like Compare
  - Semantic highlighting
  - Navigation
    - F3 everywhere!
    - Ctrl-O Ctrl-O
    - Callin markers
    - etc.
- Compiling
- Code assist
- Help!

Working with the OTDT

<DEMO 1> Summary:

- Run
  - configuration
- Wizard:
- Editor
- Compiling
  - Full Java 5 support
  - Incremental compilation
  - Eager & partial compilation
    - sophisticated error recovery (content assist)
  - Speed
  - .java → .class
  - Scoped keywords:
    - plain identifiers when used outside `team`
- Code assist
- Help!

Working with the OTDT

<DEMO 1> Summary:

- Run
  - configuration
- Wizard:
- Editor
- Compiling
- Code assist

- Help!

- Completion
  - OT/J `keywords`
  - create/complete method bindings (callout/callin)
  - support std. completions in OT/J code, too. *(work in progress)*

Working with the OTDT

<DEMO 1> Summary:

- Run
  - configuration
- Wizard:
- Editor
- Compiling
- Code assist

- Help!

Quick Fix
- modifier problems
  - class, method, method binding
- learn about special OT/J features
- type errors
- coding style
- adapt JDT quick fixes
  to handle OT/J elements/rules, too:
  - suppress warnings, create method,
    organize imports

Working with the OTDT

<DEMO 1> Summary:

- Run
  - configuration
- Wizard:
- Editor
- Compiling
- Code assist

- Help!

- New language – new error messages
  - *"What did I do wrong??"*
  - Let us tell you.
  - §§ Go to Language Definition
    - ✦precise links
    - ✦comprehensive
    - ✦single source: XML → {PDF, XHTML (x2)}
- Tutorial
- Code Samples
- Developers' Guide

Object Teams Development Tooling

Debug

<DEMO 2> Summary:

• Stepping	showing the right code
• Debug View	tell the user what's happening
• Team Activation	show and manipulate program mode

Object Teams Development Tooling

Debug

<DEMO 2> Summary:

- Stepping
- Debug View
- Team Activation

- Source mapping
  - uses JSR 45 / SMAP
- Filter runtime library code

Debug

<DEMO 2> Summary:

- Stepping
- Debug View
- Team Activation

- Beautify special stack frames
  - revert generated names
  - use custom syntax & coloring
    - dispatch code
      {{Dispatch method foo}}
    - lifting invocations
      {{Lifting to MyRole}}
    - executing declarative bindings
      [foo <- bar]
    - base calls: almost normal method calls
      base.foo()

Debug

<DEMO 2> Summary:

- Stepping
- Debug View
- Team Activation

Team Monitor View

- derived from Variables view
- all known team instances:
  - active
  - implicitly active
  - inactive

interactively change activation

✦ switch program modes

Correspondence:
runtime stack traces ↔ static call hierarchies

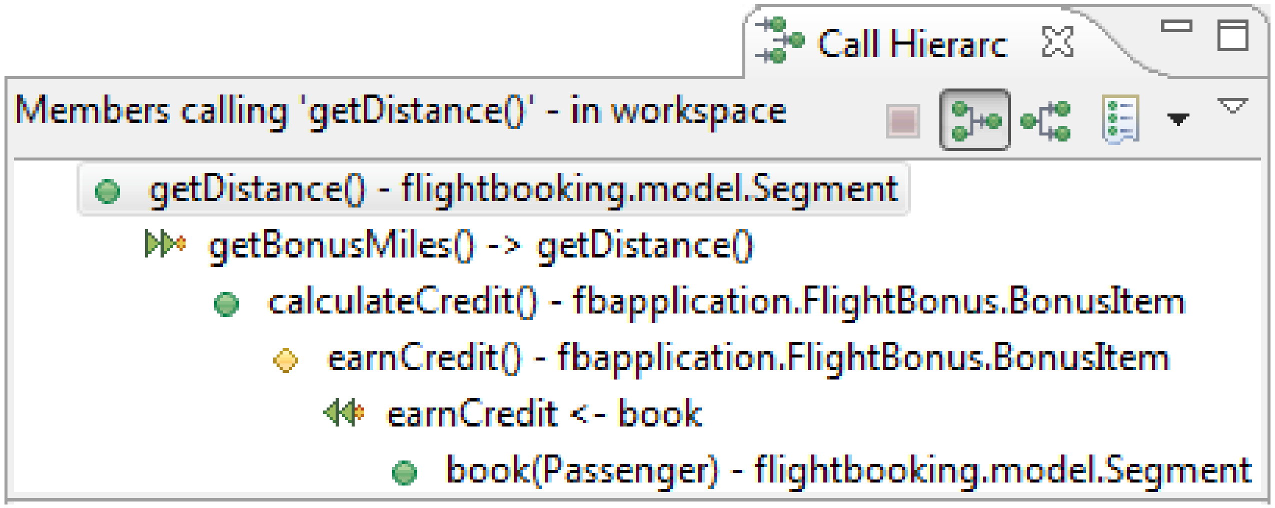
Object Teams Development Tooling

Search

- Core
  - find references within OT/J constructs
- UI
  - display found OT/J elements
  - hide generated elements
  - beautify mangled names

Call Hierarchy

- OT/J specific control flows
  - method invocations due to callout/callin method bindings
- While we're at it ...
  - control flows resulting in assignment to a given field (see https://bugs.eclipse.org/75800)



Members calling 'getDistance()' - in workspace

```
getDistance() - flightbooking.model.Segment
  getBonusMiles() -> getDistance()
    calculateCredit() - fbapplication.FlightBonus.BonusItem
      earnCredit() - fbapplication.FlightBonus.BonusItem
        earnCredit <- book
          book(Passenger) - flightbooking.model.Segment
```

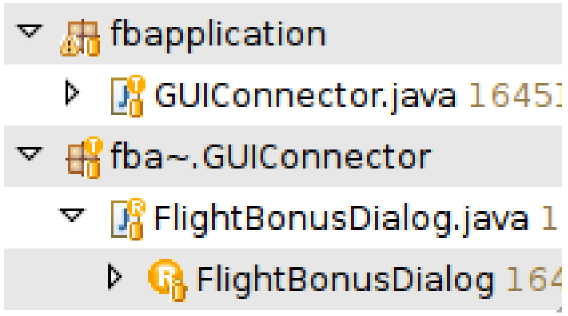
Structure Viewers

- Package explorer
- Type Hierarchy

Object Teams Development Tooling

Structure Viewers

- Package explorer
  - overlays for packages (package = team, containing role files)
  - switch physical/logical presentation of role files ()



fbapplication
GUIConnector.java 1645
fba~.GUIConnector
FlightBonusDialog.java 1
FlightBonusDialog 164

team = class
team = package

role = separate compilation unit
role = inner class

role itself is a team, containing nested roles

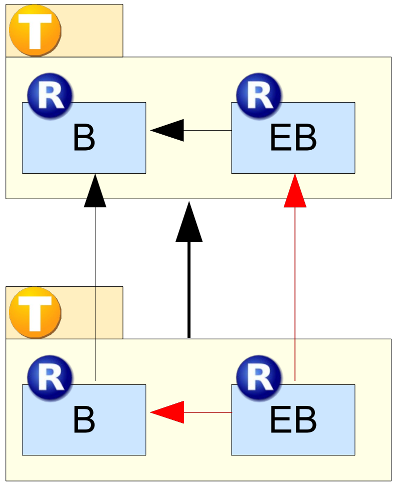
- Type Hierarchy

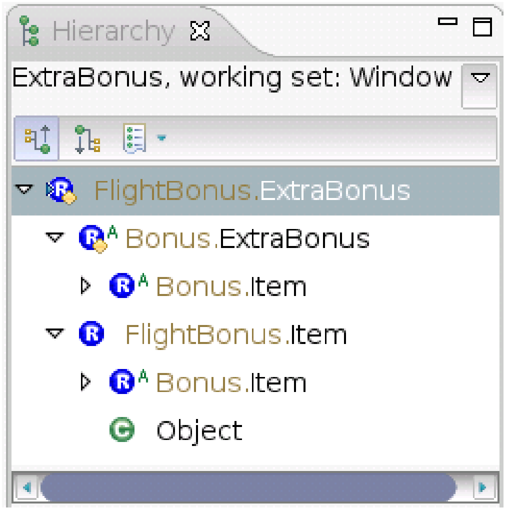
Structure Viewers

- Package explorer
- Type Hierarchy

Structure Viewers

- Package explorer
- Type Hierarchy
  - Team inheritance induces new structure
    - role classes can be overridden (virtual classes)
    - overriding role implicitly inherits from previous version
    - a role may have multiple supers





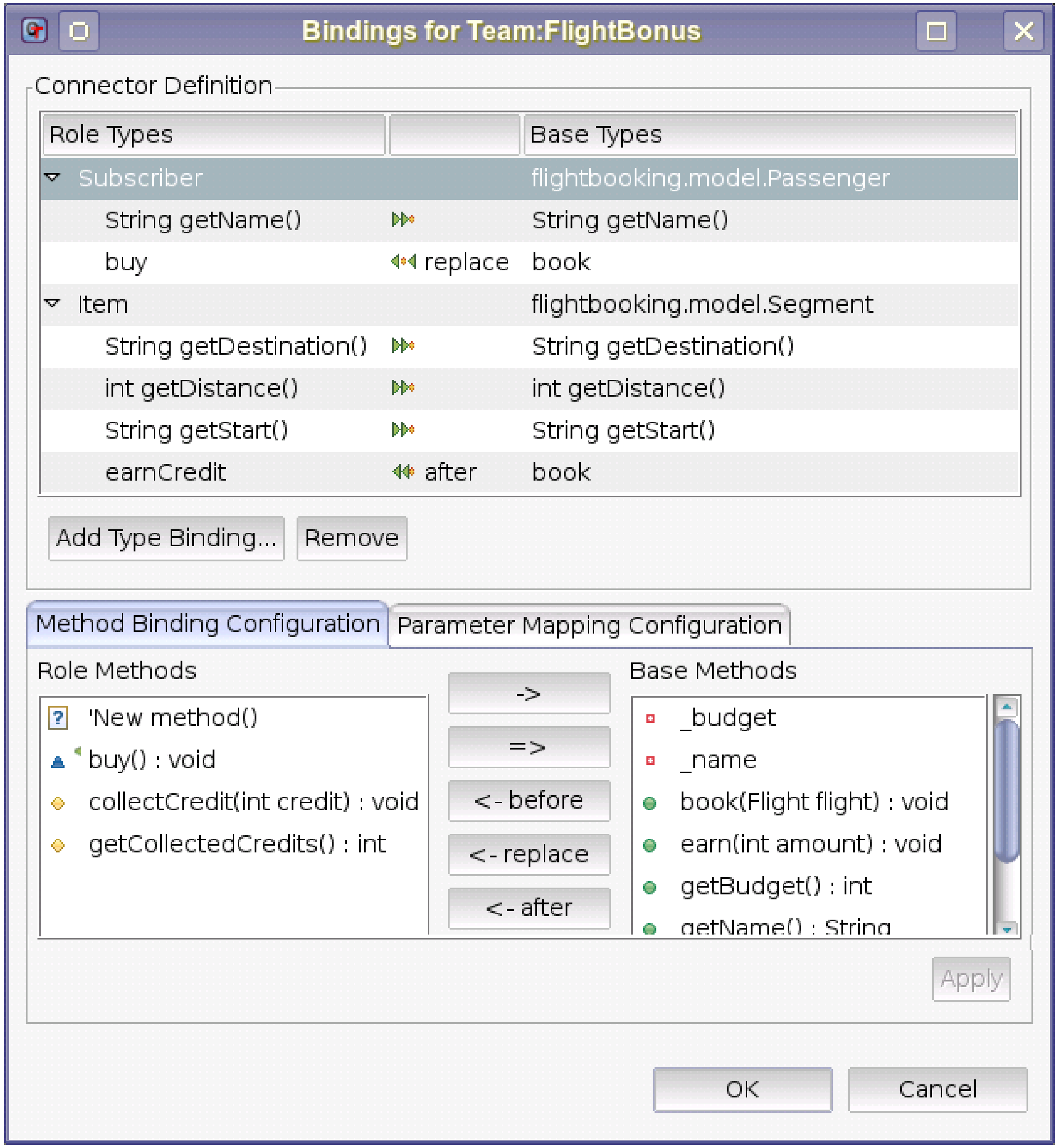
ExtraBonus, working set: Window

```
▽ FlightBonus.ExtraBonus
  ▽ Bonus.ExtraBonus
    ▷ Bonus.Item
  ▽ FlightBonus.Item
    ▷ Bonus.Item
      Object
```

Object Teams Development Tooling

Binding Editor

- Genuinely new view
- Idea: table based editing of these bindings:
  - playedBy
  - callout
  - callin
- Implemented
  - as a dialog
  - using AST rewriting



Bindings for Team:FlightBonus

Connector Definition

Role Types		Base Types
Subscriber		flightbooking.model.Passenger
String getName()		String getName()
buy	replace	book
Item		flightbooking.model.Segment
String getDestination()		String getDestination()
int getDistance()		int getDistance()
String getStart()		String getStart()
earnCredit	after	book

Role Methods		Base Methods
'New method()	->	_budget
buy() : void	=>	_name
collectCredit(int credit) : void	<- before	book(Flight flight) : void
getCollectedCredits() : int	<- replace	earn(int amount) : void
	<- after	getBudget() : int
		getName() : String

Refactoring

- Ensure soundness of existing refactorings
  - Java refactorings must not break OT/J code
  - done for some fundamental refactorings
    - implementation was based on JDT 3.0
    - currently only partly supported
    - *migration is work in progress*
- New refactorings
  - want to create OT/J structures by refactoring
  - *current stage: planning*

Under the hood

Facing real problems

- OTDT should be similar to JDT, just different

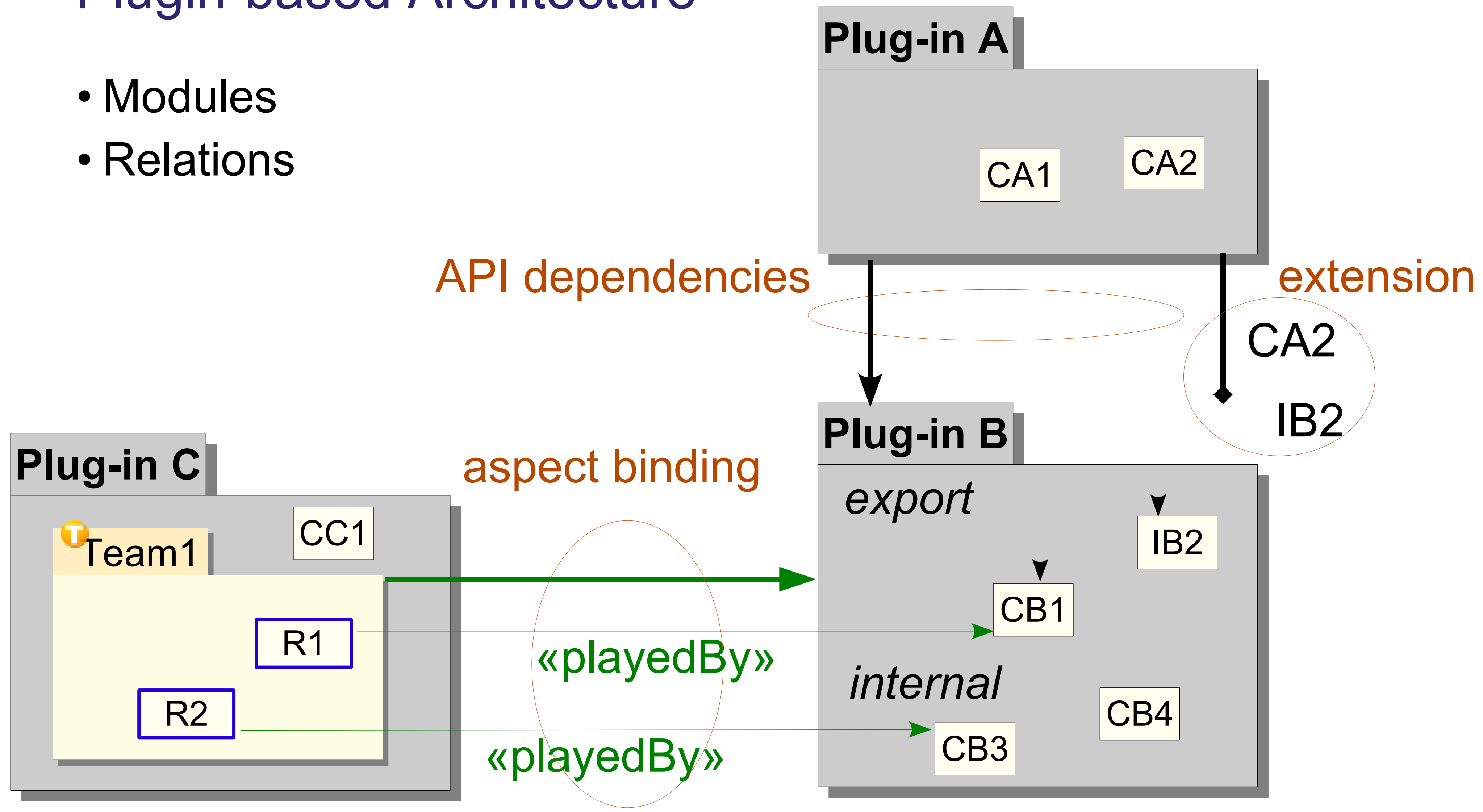
  Difference is

  - not anticipated by JDT developers
  - nor likely to be supported by future version of the JDT
    - too many hooks needed,
    - scenarios are too specific
- "I need this hook now."
  - Copy & paste? Other ways of hacking the base?
- "My adaptation must be maintainable." – *"clean"*
  - one module per feature (and vice versa)
  - well-defined / narrow interface to existing plugins.
- That's what OT/J has been developed for!
- Our "trick":
  - Plugins written in ObjectTeams/Java

OT/Equinox

Plugin-based Architecture

- Modules
- Relations

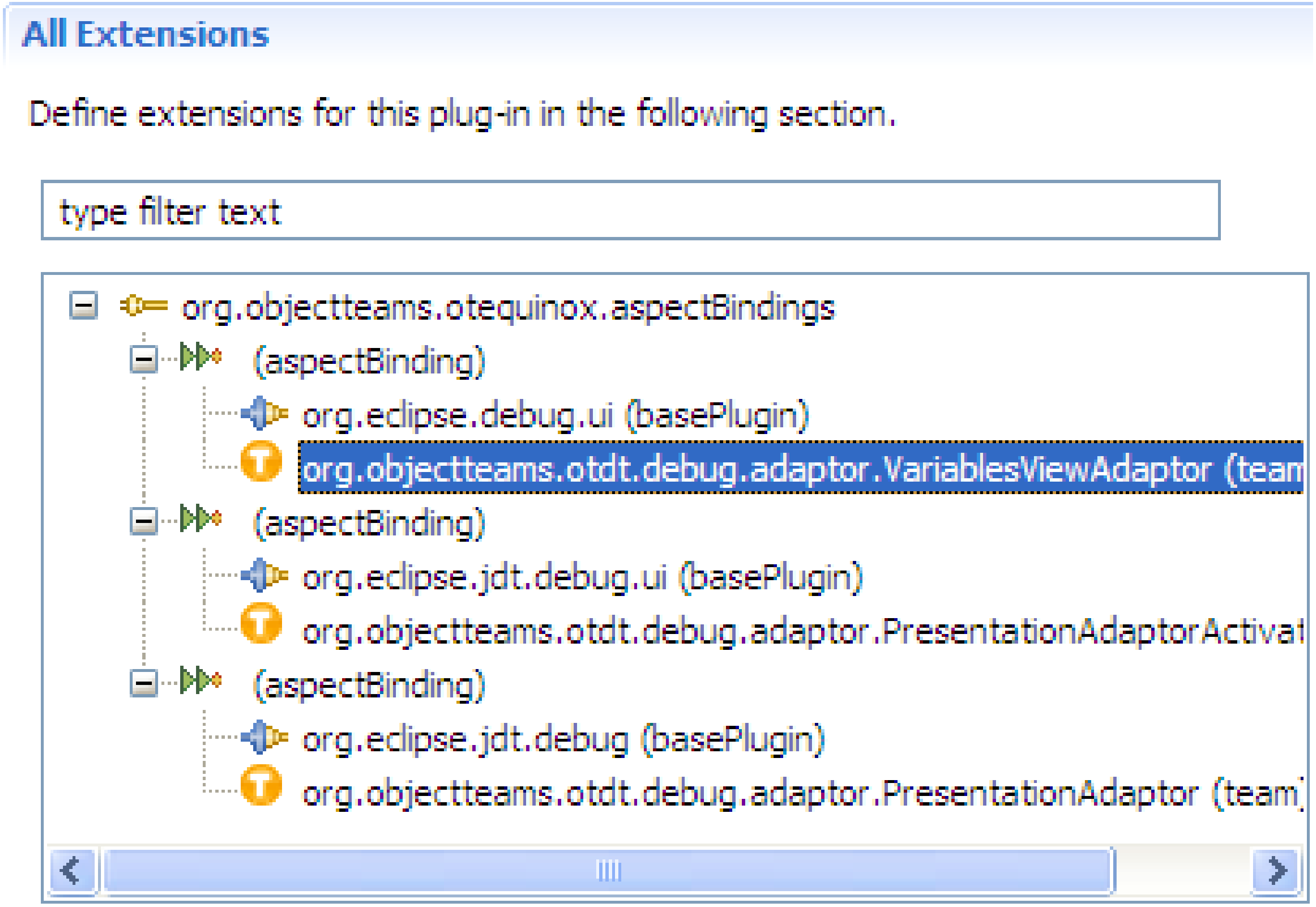


Declaring Aspect Bindings

Extendsion point `org.objectteams.otequinox.aspectBindings`

- basePlugin
- team

- Limited privileges
- Deployment
  - Instantiation
  - Activation



All Extensions

Define extensions for this plug-in in the following section.

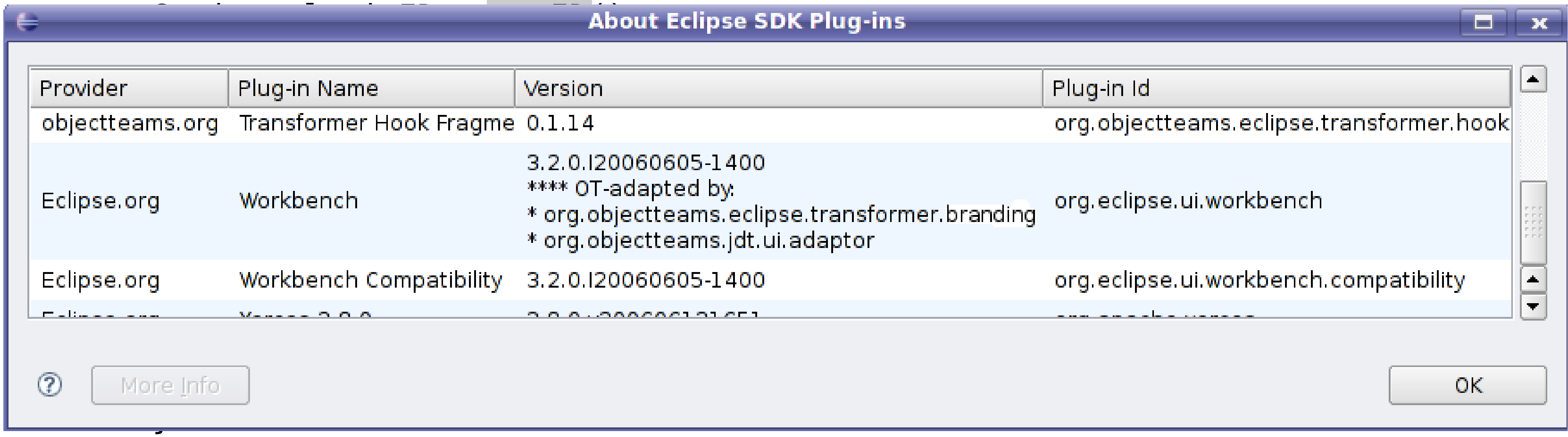
type filter text

```
org.objectteams.otequinox.aspectBindings
  (aspectBinding)
    org.eclipse.debug.ui (basePlugin)
    org.objectteams.otdt.debug.adaptor.VariablesViewAdaptor (team
  (aspectBinding)
    org.eclipse.jdt.debug.ui (basePlugin)
    org.objectteams.otdt.debug.adaptor.PresentationAdaptorActivat
  (aspectBinding)
    org.eclipse.jdt.debug (basePlugin)
    org.objectteams.otdt.debug.adaptor.PresentationAdaptor (team)
```

OT/Equinox

E.g., adapting “About Bundles”

```
public team class BrandingAdaptor {
  protected class AboutBundleAdaptor playedBy AboutBundleData
  {
    callin String getVersion() {
      String adaptationString = ""; //$NON-NLS-1$
```


About Eclipse SDK Plug-ins

Provider	Plug-in Name	Version	Plug-in Id
objectteams.org	Transformer Hook Fragme	0.1.14	org.objectteams.eclipse.transformer.hook
Eclipse.org	Workbench	3.2.0.I20060605-1400 **** OT-adapted by: * org.objectteams.eclipse.transformer.branding * org.objectteams.jdt.ui.adaptor	org.eclipse.ui.workbench
Eclipse.org	Workbench Compatibility	3.2.0.I20060605-1400	org.eclipse.ui.workbench.compatibility

```
      return base.getVersion() + adaptationString;
    }
    String getID() -> String getId();
    getVersion <- replace getVersion;
  }
}
```

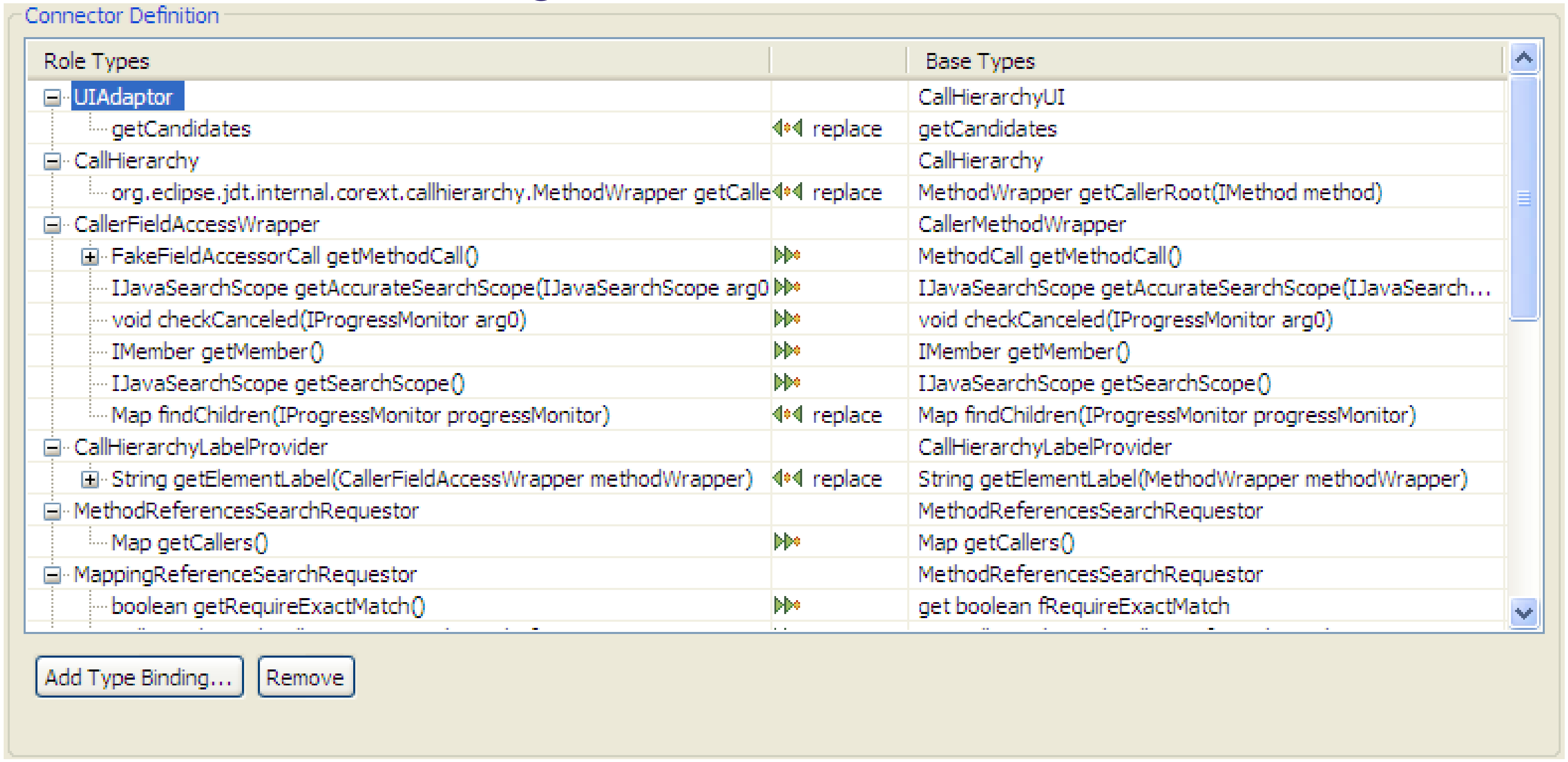
Overriding 1 single method

OT/Equinox

Typical adaptation: extend switch-case

```
protected class SuppressWarningsAdaptor playedBy SuppressWarningsSubProcessor {
  static callin void addSuppressWarningsProposal(ICompilationUnit cu, ASTNode node,
    String warningToken, int relevance, Collection<ASTRewriteCorrectionProposal> proposals)
  {
    // adding one case block to the front of the original method:
    ChildListPropertyDescriptor property= null;
    String name;
    Object baseElement;
    switch (node.getNodeType()) {
    case ASTNode.CALLIN_MAPPING_DECLARATION:
      property= CallinMappingDeclaration.MODIFIERS2_PROPERTY;
      baseElement=                               // otj specific code here
      name=                                      // otj specific code here
      break;
    // other similar cases omitted
    default:
      // other cases are already handled by the original method.
      base.addSuppressWarningsProposal(cu, node, warningToken, relevance, proposals);
      return;
    }
    String label=                                // otj specific code here
    ASTRewriteCorrectionProposal proposal=       // otj specific code here
    proposals.add(proposal);
  }
  addSuppressWarningsProposal <- replace addSuppressWarningsProposal;
}
```

Advanced OT-Plugin


Connector Definition

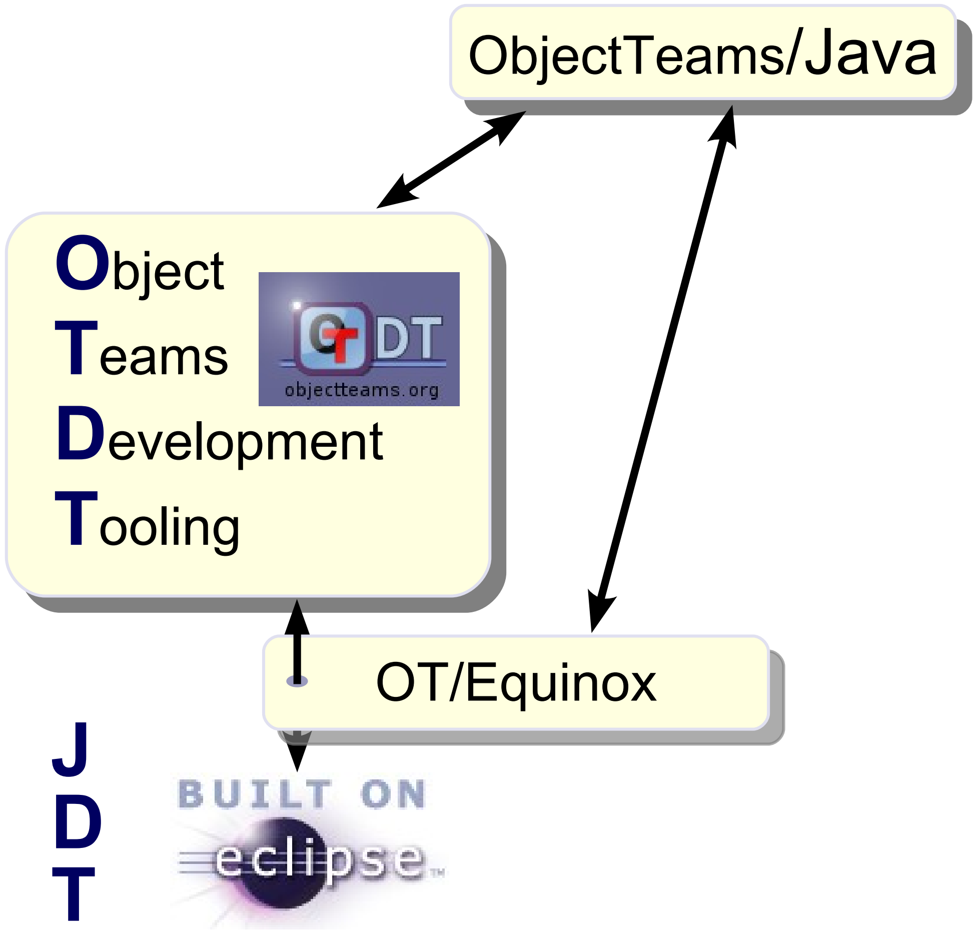
Role Types		Base Types
⊟ UIAdaptor		CallHierarchyUI
getCandidates	◀◀ replace	getCandidates
⊟ CallHierarchy		CallHierarchy
org.eclipse.jdt.internal.corext.callhierarchy.MethodWrapper getCalle	◀◀ replace	MethodWrapper getCallerRoot(IMethod method)
⊟ CallerFieldAccessWrapper		CallerMethodWrapper
⊞ FakeFieldAccessorCall getMethodCall()	▶▶	MethodCall getMethodCall()
IJavaSearchScope getAccurateSearchScope(IJavaSearchScope arg0	▶▶	IJavaSearchScope getAccurateSearchScope(IJavaSearch...
void checkCanceled(IProgressMonitor arg0)	▶▶	void checkCanceled(IProgressMonitor arg0)
IMember getMember()	▶▶	IMember getMember()
IJavaSearchScope getSearchScope()	▶▶	IJavaSearchScope getSearchScope()
Map findChildren(IProgressMonitor progressMonitor)	◀◀ replace	Map findChildren(IProgressMonitor progressMonitor)
⊟ CallHierarchyLabelProvider		CallHierarchyLabelProvider
⊞ String getElementLabel(CallerFieldAccessWrapper methodWrapper)	◀◀ replace	String getElementLabel(MethodWrapper methodWrapper)
⊟ MethodReferencesSearchRequestor		MethodReferencesSearchRequestor
Map getCallers()	▶▶	Map getCallers()
⊟ MappingReferenceSearchRequestor		MethodReferencesSearchRequestor
boolean getRequireExactMatch()	▶▶	get boolean fRequireExactMatch

Add Type Binding... | Remove

- One team encapsulates many individual adaptations.
- Adapting several plugins involves several teams.

Trailer

The way we've come



- ObjectTeams/Java
  - started in late 2001
- OTDT
  - started in 2003
  - 1.0.0 release with Callisto
- OTDT on OT/Equinox
  - since mid 2006
  - current version is 1.1.8

- OT/Equinox for non-IDE use

see you on http://www.objectteams.org/