Object Teams Creation Review

Submitters

Stephan Herrmann, Independent

Marco Mosconi, Fraunhofer FIRST

Olaf Otto, Unic AG

Review Date

January 27, 2010

Communication Channel

`eclipse.objectteams` **newsgroup**

d

- **Background and Description**
  - Motivation
  - Goal
  - Concepts
  - Realization
  - Benefits
- **Proposed Components**
- **Scope**
- **Relationship with other Eclipse Projects**
- **Organization**
  - Development
  - Participants
  - Code Contribution
  - Tentative Plan
- **References**

Background: Motivation

- Given these factors ...
  - software complexity
  - re-use of existing components
  - evolution over significant period of time
- … tensions are observed between
  - different requirements
  - different properties / qualities
  - different stakeholders, e.g.:
    - framework developer: focus on stability, based on encapsulation
    - framework user: focus on extending, based on openness
- State-of-the-Art
  - software engineering applies perspectives to capture the above
  - implementation has to create a single coherent O-O design

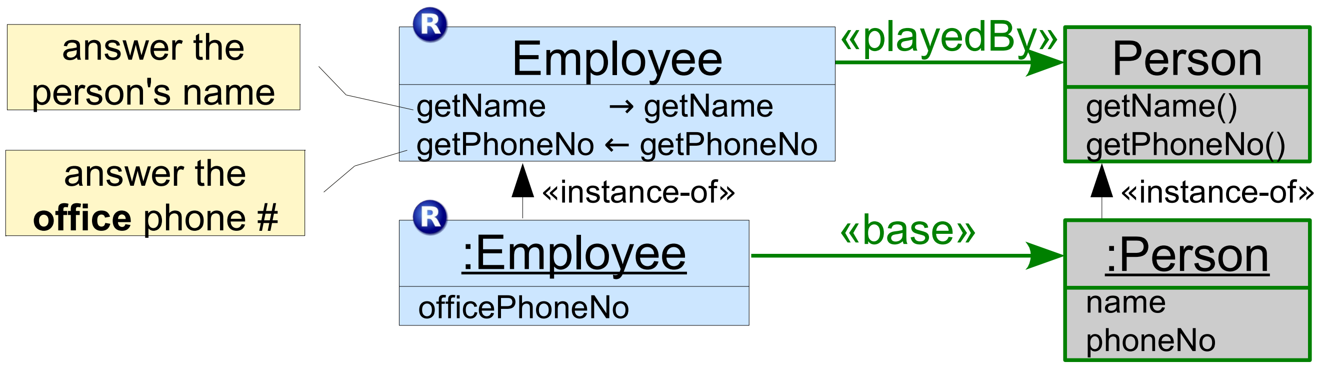
Background: Goal

- **Object Teams**
  - to explicitly support perspectives in O-O programming
    - one module per perspective
    - explicit connections/bindings between perspectives
  - core concepts
    - **role** objects are views on base objects
    - 3 kinds of **bindings** connect roles to their bases (see below)
    - **team** objects group collaborating role objects to a module
- **Properties**
  - adaptation of existing classes / objects
    - non-invasive adaptation
    - unanticipated adaptation
  - modules for complex cross-cutting concerns
  - optimal product of *reuse x maintainability*
  - seamless integration with core O-O concepts

Description: Concept of Roles (1)

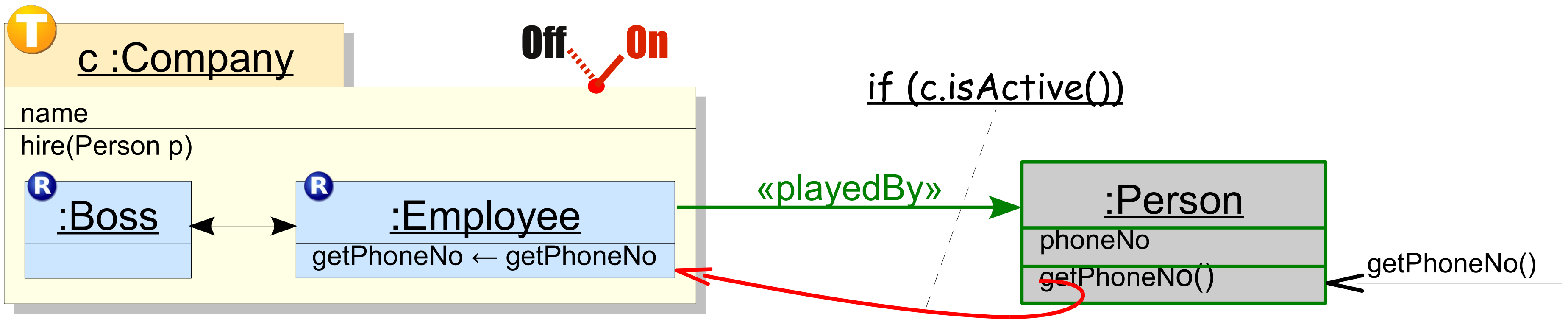
- **Concept of roles**
  - pervasive throughout different areas of software engineering
  - role modeling, e.g.:
    - "Object Oriented Role Analysis and Modeling" [Reenskaug 1996]
    - many scientific publications by different authors
    - UML has a notion of roles, too, influenced by Reenskaug
- **Similar to inheritance ...**
  - roles specialize their bases
- **… but different**
  - multiple independent specializations, e.g.,
    - the same person is perhaps an `Employee` *and* a `Father`
    - no need to create a class `EmployeeAndFather`
  - dynamic specialization, e.g.,
    - `Person` is not born as a `Student`, but may **become** a `Student`
    - after University, `Student` role is **removed**, `Person` **retained**

Description: Concept of Roles (2)



- **Binding classes & objects**
  - `playedBy` declares a binding between role/base classes
  - each role instance is bound to a base instance
- **Forwarding method calls**
  - callout method binding (→): role "inherits" base method
- **Overriding methods**
  - callin method binding (←): role "overrides" base method
- **Intercepting method calls**
  - callin method binding can be used to intercept control flow

Description: Concept of Teams



- **Encapsulate collaboration**
  - a team instance contains a set of collaborating roles
    - role instance cannot escape its enclosing team instance
    - no danger to mix roles from different teams, e.g.,
      - `Employee` will not inadvertently report to `Boss` of a different company
  - team instance can be dynamically (de)activated ( )
    - method call interception (callin) only if team is active
    - team ensures consistent (de)activation of all its contained roles
  - team automatically manages instance mapping *roles* ↔ *bases*

Description: Realization

- **Language: OT/J**
  - integrates Object Teams into Java™
  - thoroughly integrates with all of O-O / Java
    - inheritance, generics, static typing etc.
  - extended Java compiler
  - load-time byte-code weaving (for role-base binding)
- **IDE: Object Teams Development Tooling (OTDT)**
  - deeply integrated with JDT and many other plug-ins
    - full convenience of Eclipse / JDT available for OT/J development
    - IDE is fully aware of added concepts
- **Component Runtime: OT/Equinox**
  - declare "`aspectBinding`" at architecture level
  - a framework extension provides runtime support for OT/J

Description: Benefits

- **The code structure within each component …**

  … can be much simpler and easier to understand, because original concerns can faithfully be encapsulated as teams and roles, allowing for separate evolution of concerns over time.

  Any crosscutting interaction can concisely be defined using the mentioned bindings.

- **Re-use of existing components …**

  … is greatly improved, because bindings between different components can create client-defined adaptations of a re-usable component in a concise and disciplined way.

  With Object Teams many of the compromises that are necessary with traditional approaches become obsolete.

Proposed Components

The following components define the internal structure of the Object Teams Project, they do not define individual Eclipse projects:

- **OT/J.** Provides the compiler and runtime for OT/J programs. The compiler can be used as a batch compiler, via Apache Ant or Apache Maven 2 or as part of the OTDT (see below). The OTRE (Object Teams Runtime Environment) consists of a few foundation classes plus a set of byte code transformers that perform the weaving for any role-base bindings.
- **OT/Equinox.** Provides the runtime for running Equinox bundles written in OT/J. This includes providing an extension point by which aspect bindings are declared to the framework plus an integration of the OTRE byte code weavers into the Equinox framework.
- **OT/JPA.** Provides an integration of OT/J and the EclipseLink implementation of the JPA.
- **OTDT.** Provides comprehensive support for developing OT/J programs. This includes specific views, editors, wizards, content assist and debugging support. The OTDT specifically supports development for the OT/Equinox runtime.

It will be checked whether the freely available **OTJLD** (OT/J Language Definition) shall also be maintained within the Object Teams Project.
More components may follow in the future

Scope (1)

Out of scope

- Since the OTJLD (OT/J **Language Definition**) version 1.2 is essentially fixed big changes to the language OT/J are outside the scope of this project

Within the scope of this project

- **Maintain and further improve the technology** for developing and executing programs written in OT/J.
    - editing (several views, content assist, refactoring etc.)
    - compiling (batch and incremental)
    - running and debugging
    - react to small changes in the OTJLD
- **Maturing a prototype** that replaces the load-type weaver with a **run-time weaver**
- Integration with **more tools** (see "Tentative Plan" below)

Within scope if sufficient interest is expressed

- Current technology includes integration with the runtimes of Equinox and EclipseLink
    - Create similar **integrations with other runtimes** (like, e.g., servlet engines)
- Further **development tools** like a graphic modeler for Object Teams

Scope (2)

Given the relative maturity of the code contribution ...

- … a significant goal behind moving this project to Eclipse lies in more effectively promoting Object Teams and **actively building a community**.
- These activities will aim at establishing a **long term perspective** for the Object Teams technology.

Planned reach-out to the community:

- **Articles**:
  - OT/J primer
    - simple hands-on introduction
  - Adapting plug-ins with OT/Equinox
    - starting from an OT/Equinox-Hello-World
    - include some patterns from real-world application
  - Customizing GMF editors with OT/Equinox
    - based on a case study (refactoring of the UML2Tools class diagram editor)
- **Presentations, screencasts, tutorials**
- Answering questions on **mailinglist / newsgroup**
- Further reach-out to potential adopters, open to suggestions

Relationship with other Eclipse Projects (1)

AspectJ and the AJDT

- Shared ...
  - motivation: support modularity where standard O-O fails to do so.
  - mechanism: support (some form of) method call interception.

⇒ Very generic commonalities, but differences prevail:

- **AspectJ**: focus on sophisticated pointcut language ...
  - to specify large sets of join points,
  - where join points are fine grained elements of the control flow
- **Object Teams**: focus on modules
  - role captures specialization for *one* base class / object
  - team encapsulates set of collaborating roles
  - roles and teams can be composed to create larger structures: nesting, layering & inheritance of complex modules.

⇒ Approaches are complementary, tools do not overlap

Relationship with other Eclipse Projects (2)

- **JDT/Core**
  - The core plug-in of the OTDT is a branch of JDT/Core
    - OT/J compiler, public AST etc.
    - compatible with the original JDT/Core
      - original JDT/Core tests find no significant difference
    - experience and discipline of maintaining the branch (see below)

  The parallel development of the JDT/Core and the OT/J branch has fostered quite some **interaction between developers** including numerous contributions to the JDT/Core

- **JDT/UI, JDT/Debug(core,ui), PDE(core,ui) etc.**
  - Many plug-ins are re-used and extended/adapted using OT/Equinox
    - While adapting these components in unanticipated ways, their source code is not altered
    - OT/Equinox and the OTDT are very explicit regarding the facts of bundle adaptation, so both developers and users can easily inspect the status of the system using the "About" dialogs, a specific OT/Equinox monitor and the package explorer (during development)
  - Prototyping of enhancements
    - Some contributions to Eclipse where first developed and deployed as OT/Equinox plug-ins, and only later refactored to a pure-Java inline solution that was contributed as a patch

Relationship with other Eclipse Projects (3)

- **Equinox Aspects**
  - Closely related to OT/Equinox.
    - Uses same Equinox Adaptor Hooks (co-op with Thomas Watson)
    - Ongoing discussions with Martin Lippert
      - may consider merging of both efforts
      - merging would facilitate combined use of OT/J with other weavers
      - some different technical requirements still to be discussed
  - OT/Equinox runtime
    - conceptually essential
    - technically very small part of the project (<3% of total kLOC)

Relationship with other Eclipse Projects (4)

- **Potential benefits for other Eclipse Projects**
  - E.g., generated graphical editors based on EMF/GMF
    - well know problem in Modeling community:
      - how to apply fine-tuning to a generated application?
      - need to integrate hand coded parts with generated code
      - workaround using "generation gap pattern" or the like
      - maintainability is very difficult to achieve
    - **Real-world case study** with UML2Tools
      - refactored class diagram editor to use OT/Equinox
      - demonstrate 100% extraction of hand-coded customizations to OT/J
      - OT plug-in can be bound to a pristine version of the generated editor
      - regain maintainability & added feature-oriented modularization (generators create technology-oriented modularization)
  - Similar benefits expected for several other Eclipse projects

Organization: Development

- **Development infrastructure used**
  - Subversion source code repository (migrated from an initial CVS repository)
  - Testing
    - Original Eclipse JUnit plug-in test suites (with slight modifications) plus additional 2150 JUnit plug-in tests specific for OTDT features
    - A specific test suite for compiler and runtime: 2300+ test programs
    - Total number of test cases used in daily builds is currently 44,560
  - Build automation based on PDE-build
  - Issue tracking and wiki using Trac (trac.objectteams.org/ot)
- **Maintainability**
  - JDT/Core branch: strict discipline of marking code changes
  - All other plug-ins: adapted using API, extension points & OT/Equinox
  - Experience of 5 migrations to new versions of Eclipse

Integrating the Object Teams development with the Eclipse coordinated releases will require only minimal changes in our development process

Organization: Participants

- **Initial Committers**
  - Stephan Herrmann (Independent): Proposed project lead
  - Marco Mosconi (Fraunhofer FIRST): Proposed committer
  - Olaf Otto (Unic AG): Proposed committer OT/JPA
- **Mentors**
  - Chris Aniszczyk – *EclipseSource* – PDE, ECF
  - Darin Wright – *IBM* – Platform, JDT (debug), PDE (ui, API tools)
  - Ed Merks – *Macro Modeling* – Modeling (EMF, EMFT, e4)
- **Interested Parties**
  - itemis: Steffen Stundzig
  - GEF3D: Jens von Pilgrim
  - Intershop Communications AG: Peter Hänsgen
  - GEBIT Solutions: Tom Krauss
  - BeagleSoft: Tim Ehlers
  - Martin Lippert

Organization: Participants

- **Stephan Herrmann** (proposed project lead)
  - PhD in computer science at Technische Universität Berlin, 2002. Thesis on building a multi-view software engineering environment
  - Since 2002 focus on developing Object Teams
  - Project lead of a publicly funded project (“TOPPrax”, 2003-'06), where significant parts of the OTDT have been developed
  - Active participation in the development of the OTDT
  - Supervisor of 25+ diploma theses (comparable to master's)
    - several of these related to Eclipse and the OTDT
  - Active contributor for various Eclipse projects
    - reported 100+ bugzillas
    - contributed patches to 20+ bugzillas
    - patches cover JDT (core/ui), PDE (build,ui), Platform (compare,ui), Equinox(p2)

Organization: Participants

- **Marco Mosconi** (proposed committer)
  - PhD candidate (late phase) at Technische Universität Berlin.
  - Researcher at Fraunhofer FIRST, Berlin.
  - Diploma (~master's) from Technische Universität Berlin, 2003. Thesis on integrating the Object Teams approach with the CORBA Component Model
  - Focus on combining aspect-oriented, component-based and model-driven approaches.
  - Teaching of JEE and model driven technologies in advanced university courses.
  - Supervisor of 5 diploma theses (comparable to master's)
    - covering EMF, GMF and OT/Equinox
  - Supervisor of case study regarding UML2Tools (see slide #16)

Organization: Participants

- Olaf Otto (proposed committer, focus on OT/JPA)
  - Application architect at Unic AG, Switzerland
  - Diploma (~master's) from Technische Universität Berlin, 2009. Thesis on JPA-based persistence for Object Teams
  - 10+ years of experience developing JEE applications for international customers, with focus on:
    - OOA / OOD
    - AOP
    - TDD
    - Spring Framework

Code Contribution (1)

- **Current release is 1.4.0 Milestone 2**
  - 50+ releases prior to 1.0.0 in March '07
  - 24 releases since 1.0.0
- **All code is licensed under the EPL v1.0**
  - Contributors (former partners in the TOPPrax project)
    - Technische Universität Berlin
    - Fraunhofer FIRST
  - Individual contributors:
    - Stephan Herrmann (Independent since April 2009)
    - Olaf Otto (Results of Diploma project)

Code Contribution (2)

- **Components to be contributed**[1]
  - Object Teams Runtime Environment (OTRE)
  - branch of `org.eclipse.jdt.core`
  - 12 regular plug-ins: **`org.objectteams...`**

`...runtime`	– basic support for launching OT/J programs
`...otdt`	– OTDT branding plugin
`...ui`	– dialogs, wizards, markers, actions, preferences ...
`...ui.help`	– developer guide, linked language definition
`...debug`	– special breakpoints
`...debug.ui`	– new debug view “Team Monitor”
`...metrics`	– code metrics for OT/J, computation and browsing
`...otequinox`	– client visible part of OT/Equinox
`...otequinox.hook`	– framework extension hooking into Equinox
`...otequinox.runtime`	– provide OTRE as a framework extension
`...eclipse.monitor`	– new view for viewing/changing OT/Equinox state

[1] for full list of features see http://www.objectteams.org/distrib/features.html

Code Contribution (3)

- **Components to be contributed (continued)**
  - 8 OT plug-ins: `org.objectteams...`

`...jdt.ui.adaptor`	– adaptations of numerous parts of the JDT/UI
`...otdt...`	
`...apt.adaptor`	– make APT aware of OT extensions of the AST
`...compiler.adaptor`	– help the incremental builder to handle OT/J, – add OT/Equinox support to compilation
`...debug.adaptor`	– JSR-045 support – add OT support to launch configurations – hide OT/J internals during debugging
`...pde.ui`	– project natures for OT/Equinox development (wizard) – add OT/Equinox elements to the package explorer – anticipate pending patches against PDE/UI
`...refactoring.adaptor`	– adapt Java refactorings for OT/J – add OT-specific refactorings (pending)
`...samples`	– provide OT/J sample projects – fix assumed bugs in org.eclipse.pde.internal.ui.samples
`...otequinox.branding`	– show OT/Equinox info in "About plug-ins" dialog

  - 6 Maven modules:
    - OT/JPA runtime; awaiting packaging as OSGi bundles.

Tentative Plan

- **Releases**
  - A new version can be released any time.
  - Project graduation planned for summer of 2010
    - official 1.0 release of Object Teams as an Eclipse project shall hopefully be published in conjunction with the Helios coordinated release.
- **Future plan items**
  - Maturing the JPA integration.
  - Potentially integrating with Equinox Aspects.
  - Integrating with more tools like Mylyn, EclEmma, TOD, etc.
  - Continuous improvement of all existing components
    - currently have 44 open issues in Trac
    - long term average of 10+ issues resolved per month
    - always keep the number of open issues low

References

- **Current Home Page**
  - www.objectteams.org
- **Selected Presentations**
  (available at www.objectteams.org/publications/presentations.html)
  - *The Object Teams Development Tooling: A high fidelity extension of the JDT*
    Stephan Herrmann at EclipseCon 2008
  - *Plugin reuse and adaptation with Object Teams: Don't settle for a compromise!*
    Stephan Herrmann at EclipseCon 2009
  - *Modular EMF/GMF customization with OT/J – case study: UML2 Tools*
    Marco Mosconi at Eclipse Demo Camps May/June 2009.
- **Key Publication**
  (available at www.objectteams.org/publications)
  - *Object Teams: Improving Modularity for Crosscutting Collaborations.*
    Stephan Herrmann in Proceedings of Net.ObjectDays, Erfurt, 2002.
  - *Integrating Object Teams and OSGi: Joint Efforts for Superior Modularity*
    Stephan Herrmann and Marco Mosconi in Journal of Object Technology, vol. 6, no. 9, October 2007, pages 105—125.