Eclipse Demo Camp – Nov 09, Berlin

Adapting EclipseLink for Object Teams, of course using Object Teams

Stephan Herrmann
Independent

www.objectteams.org

Object Teams

- OT/J — since 2001
  - Java += roles, teams, aspect bindings
  - most seamless extension of **Java**, ...
- Object Teams Development Tooling — since 2003
  - Java Compiler += OT/J constructs
  - JDT for OT/J (code assist, ui, launch …)
  - ... of **the best existing IDE**
- OT/Equinox — since 2006
  - Equinox += aspect bindings
  - ... of **the most popular module system** for Java

Object Teams

- OT/J since 2001
  - Java += roles, teams, aspect bindings
  - most seamless extension of **Java**, ...
- Object Teams [illegible]pment Tooling since 2003
  - Java
  - JDT
  - ... of
- OT/Ec [illegible]ce 2006
  - Equino[illegible]
  - ... of **the most popular module system** for Java

Extreme modularization:

- every concern
  can be encapsulated in a module
- every relationship between concerns
  can be specified using explicit bindings

Object Teams

- OT/J since 2001
  - Java += roles, teams, aspect bindings
  - most seamless extension of **Java**, ...
- Object Teams Development Tooling since 2003
  - Java Compiler += OT/J constructs
  - JDT for OT/J (code assist, ui, launch …)
  - ... of **the best existing IDE**
- OT/Equi ice 2006
  - Equi
  - ... of

BUILT ON eclipse

eclipseCON
I'm speaking at EclipseCon 2008

High Fidelity extension:
- it is still the JDT
- all of the JDT is aware of Object Teams
- plus additional views / actions where needed

Object Teams

- OT/J since 2001
  - Java += roles, teams, aspect bindings
  - most
- Obje 2003
  - Jav
  - JD
  - ... of the b isting IDE
- OT/Equinox since 2006
  - Equinox += aspect bindings
  - ... of **the most popular module system** for Java

Extreme re-use, no compromise:

- find a plug-in/bundle that ~fits your needs?
- make it fit perfectly
- don't cheat, keep a crisp architecture

eclipseCON I'm speaking at EclipseCon 2009

Recent Development

- UML2Tools Case Study
  - non-invasive customization of GMF editors (M.Mosconi)
- OTDT 1.4.0 M1 (E 3.6 M3) available
  - previously 1 release / ~ 6 weeks – 1 migration / year
  - now fully aligned with Eclipse release train
- Object Teams is moving to Eclipse
  - shall be sub-project of „Tools“
  - http://eclipse.org/proposals/object-teams
  - news://news.eclipse.org/eclipse.objectteams
- Adoption of Student Projects
  - persistence
  - runtime weaver
  - extended refactoring

Application Development with OT/J

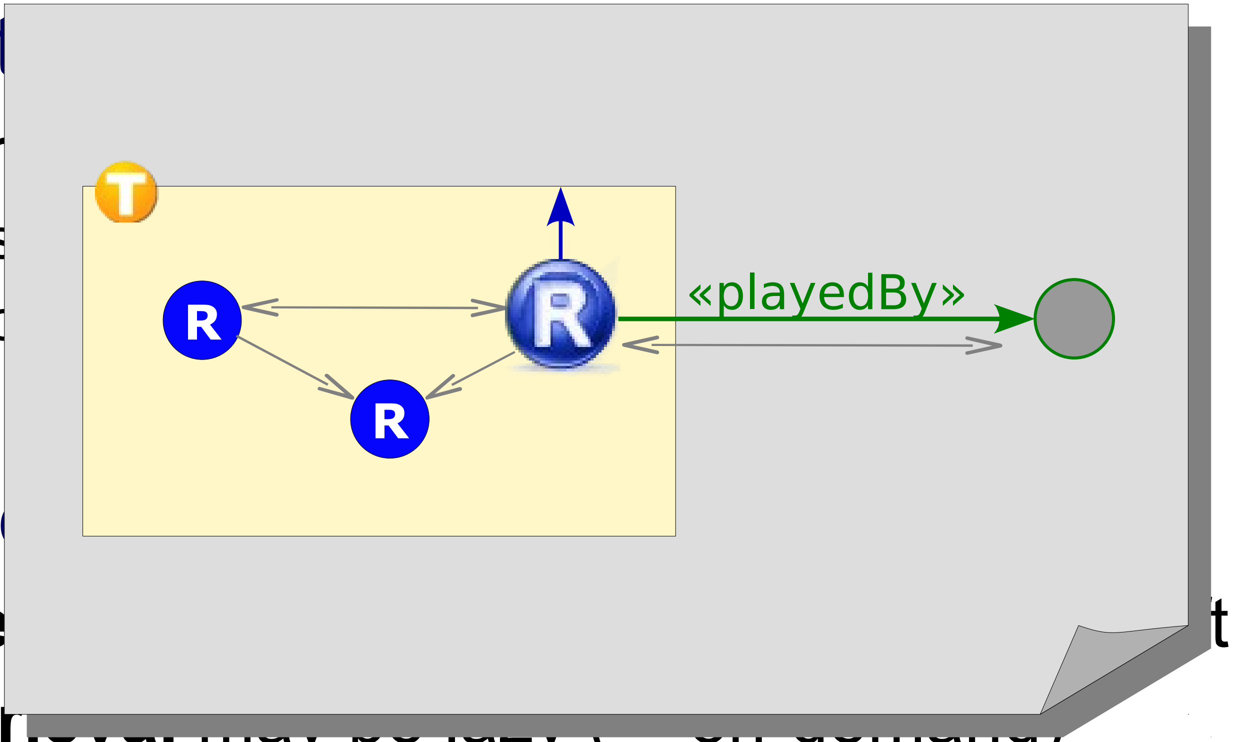
- Apply OT/J for ...
  - use case modules
    - use case = team
    - use case participants = roles
  - improved MVC
- First Case-Study 2003-06
  - performed at GEBIT Solutions
  - Nº 1 lesson: language + tools don't suffice
    - need integration w/ existing persistence framework!!
- First analysis
  - OT/J is ~ Java , roles are ~ inner classes
  - OR mapper don't support (non-static) inner classes
  - end of story!

Issues with Persisting OT/J Entities

- Inner classes – constructor arguments
  - `new Inner()` → `<init>(Enclosing this$0)`
  - OR mappers rely on empty constructor (reflection)
- Referential integrity
  - bound role always has (synthetic) reference
    - to base instance
    - to enclosing team instance
  - iff team and base are deleted role is garbage
- Life cycle
  - role **creation** may be implicit & on-demand („lifting“)
  - role **retrieval** may be lazy (= on-demand)
- Enhanced inheritance
  - team inheritance → implicit inheritance of contained roles

Issues with Persisting OT/J Entities

- Inner classes – constructor arguments
  - `new Inner() → <init>(Enclosing this$0)`
  - OR mappers rely on empty constructor (reflection)
- Referent
  - bound r
    - to bas
    - to enc
  - iff team
- Life cycl
  - role **cre** ting")
  - role **retr**ieval may be lazy (on demand)



- Enhanced inheritance
  - team inheritance → implicit inheritance of contained roles

Student Project Olaf Otto

- Select EclipseLink
  - supports JPA
  - open source, mature, Eclipse based
- Define requirements
- Setup development environment
  - OT/J ↔ Maven2 integration
- Adapt EclipseLink ...
  - ... using OT/J
- Test, develop, document, create demo
- ⇒ Diplom

Requirements

- „A1 Transparenz.
  Anwender der JPA für Object Teams sollen in die Lage versetzt werden, Teams, Rollen und Basisklassen so auszuzeichnen,
  als würde es sich um einfache Java-Klassen (POJOs) handeln.
  Die automatisch generierte, sprachinterne Infrastruktur muss dabei vor dem Anwender verborgen bleiben.“
- „A2 Schutz der Invarianten.
  Die **Wahrung der Invarianten** der Object Teams Applikation, insbesondere die Herstellung und Wahrung der Referentiellen Integrität zwischen den Object Teams-spezifischen Typen
  ist Aufgabe der Persistenzlösung.“
- „A3 Konformität.
  Die Verwendung der JPA sollte keinen Beschränkungen unterliegen. Ebenso soll die Einführung neuer, nicht in der Spezifikation definierter Auszeichnungsformen vermieden werden.
  Nur auf diese Weise kann gewährleistet werden,
  dass die entstehende Lösung von der Verbreitung des **JPA-Standards** und dessen Akzeptanz in der Java-Community profitieren kann.“

Requirements (2/2)

- „A4 Datenunabhängigkeit.
  Während die Kopplung an eine konkrete JPA-Implementierung unvermeidbar ist,
  muss die **Unabhängigkeit von der verwendeten relationalen Datenbank** und dem eingesetzten SQL-Dialekt erhalten werden.“
- „A5 Explizite Persistenz.
  Die Persistenzeigenschaften aller Typen und ihrer Attribute
  müssen entsprechend der JPA Spezifikation sichtbar und kontrollierbar sein.“

- Result:
  - Implementation fulfills requirements
  - Single restriction:
    - if implicit inheritance is used → InheritanceStrategy.TABLE_PER_CLASS

Implementation Statistics

- EclipseLink adaptation
  - 2 team classes
  - 14 role classes
  - 17 callin method bindings (interception)
  - 19 callout method bindings (forwarding)

  1349 physical lines of code (540 OT/J + 809 Java)
  106 kByte .jar

- Additional: Integration with Spring
  - Support for container managed persistence

Demo Application

- „OrderSystem“
  - plain domain entities:
    - Storage, StockItem, Customer, Address
  - domain teams/roles:
    - StockOrder
      - Item, Customer, Address are views of StockItem, Customer, Address
    - Reservations
  - MVC teams (swing)

ModelAdapterTeam	map domain entities to GUI tables
ControllerAdapterTeam	dispatch application events
GUIAdapterTeam	center windows on screen ;-)

  - data
    - hard coded dummy values

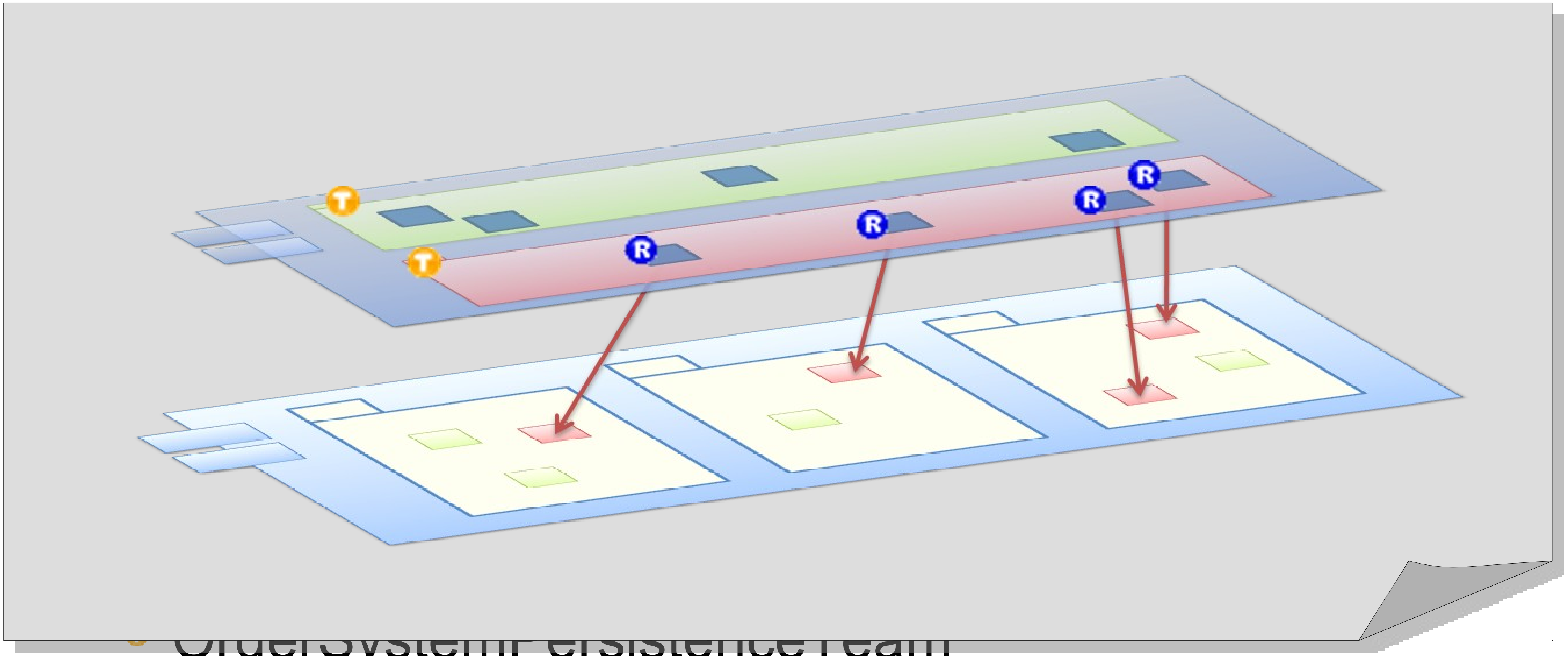
Demo:
Adding Persistence to the OrderSystem

Demo Summary

- 100% non-invasive
- XML descriptors
  - `persistence.xml`: list persistent teams & roles
  - `orm.xml`: specify mapping details
    - 100% standard JPA
    - some roles need to use TABLE_PER_CLASS
- Programmatical integration
  - AbstractPersistenceTeam (re-usable)
    - provide functions for transactioned CRUD
  - OrderSystemPersistenceTeam
    - connect application life-cycle events
    - intercept data initialization (dummy vs. from DB)
    - dispatch events for CRUD

Demo Summary

- 100% non-invasive



- OrderSystemPersistenceTeam
  - connect application life-cycle events
  - intercept data initialization (dummy vs. from DB)
  - dispatch events for CRUD

Summary

- Missing persistence for OT/J **was** a barrier
  - **No longer is ;-)**
- Whatever you do with Java ...
  - for example:
    - writing IDEs
    - customizing generated applications
    - extending existing frameworks
    - application development
      - domain
      - MVC
      - persistence
      - ...
  - it'll be easier with OT/J
  - the result will (can) be better maintainable