EMF Dos and Don´ts

Maximilian Koegel
EMF is very powerful but…

“The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code accounts for the other 90% of the development time.”

Tom Cargill
Example Model and Application

- Application to manage bowling games and players
- Model with players and games
Example #1

**Problem:** Every game in a resource is changed to professional on resource load iff it points to a professional player

**Cause:** Custom code is triggered during resource load and sets game to professional

```java
public void setPlayer(Player newPlayer) {
    Player oldPlayer = player;
    player = newPlayer;
    //custom code begin
    if (player!=null && player.isProfessional()) {
        this.setProfessional(true);
    }
    //custom code end
    if (eNotificationRequired())
        eNotify(new ENotificationImpl(this, Notification.SET, BowlingPackage.GA
```
Don´t #1: Change generated getters/setters

Reason:
• Generated getters and setters are widely used by:
  • EMF Core Framework, e.g. Resources, EcoreUtil.copy()
  • EMF-based Frameworks, e.g. Teneo, CDO, EMFStore, EEF
• Changing it may have unexpected side effects

Mitigation:
Do not change the generated getters/setters but move this code into controller classes (MVC pattern), e.g. Commands
Example #2

**Problem:** Players are lost in old game if assigned to new game

**Cause:**
- Reference containment property changed
- Generated getter is not updated
- Players can only be contained in at most one game
Do #2: Use pattern for replacing gen. code

```java
/**
 * @generated
 */

public void setPlayerGen(Player newPlayer) {
    Player oldPlayer = player;
    player = newPlayer;
    if (eNotificationRequired())
        eNotify(new ENotificationImpl(this, Notification.SET, BowlingF

/**
 * @generated NOT
 */

public void setPlayer(Player newPlayer) {
    setPlayerGen(newPlayer);
    //custom code
```
Example #3

**Problem:** Non-professional Players are reset to professional players when copied

**Cause:**
- Changed factory method in EMF factory to init default values
- Values will be set to init value on copy

```java
public Player createPlayer() {
    PlayerImpl player = new PlayerImpl();
    player.setProfessional(true);
    return player;
}
```
Don´t #3: Default initialization in EMF factory

Reason:
• Generated factories is widely used by:
  • EMF Core Framework, e.g. Resources, EcoreUtil.copy()
  • EMF-based Frameworks, e.g. Teneo, CDO, EMFStore
• Changing it may have unexpected side effects

Mitigations:
• Default values can be set in Ecore
• More complicated default initialization, e.g. creating containment children, should go into controller code
Example #4

**Problem:** CDO-Server throws NPE at startup

**Cause:** Bug in CDO 4.0 with sub-packages

---

**[CDO/EMF] Ecore SubPackages & CDOPackage reading** [message #1021715]

**ben ben**

Messages: 19

Registered: June 2012

Hi everyone.

I have an Ecore model with root & subpackage as:

```
xxx.yyy.model
xxx.yyy.model.subpackage
```

When CDO server starts, it reads PackageUnits from database doing:

```
SELECT unit, uri, parent FROM cdo_package_infos
```

It seems that my both packages (root & sub) have the same unit: `xxx.yyy.model`.

The problem is that read packages are put in a HashMap, so I obtains only one entry for both packages.

So it causes a NPE in `MetaDataManager.loadPackageUnit()` when trying to resolve package with id="xxx.yyy.model".
Don’t #4: Use of sub-packages

Reasons:
• Do not use sub-packages
• Ed Merks (EMF): “Subpackages are evil!”
• Many bugs and problems in frameworks result from subpackage support

Mitigation: Use multiple Ecores
Examples #5: Containment structure

Model:
• Car
• Motor is contained in Car?
• Tire is contained Car?
Dos #5: Design Containment Structure

- Design a containment structure following your application domain:
  - Which objects belongs to which other object?
  - Should this object be copied/deleted if the other object is copied/deleted?
- Avoid flat collections of EObjects:
  - Might be slow if there are many objects
  - Difficult to visualize
- Allow the end-users to structure their data from an application domain point of view: e.g. Folder-like Elements
Example #6

Problems:
• Application slow-down over time
• Application uses exceeding amounts of memory
• “Widget is disposed” – Exceptions

Cause: Missing Adapter Disposals
Dos #6: Carefully use Adapters

Reasons:
• Adapters will use more and more memory
• If UI Widgets are disposed without adapters Exceptions result

Mitigations:
• Make sure adapters are disposed properly
• Use content and cross reference adapter wisely
• Establish tests for proper adapter removal
  • Adapter count before and after test is same
Example #7

Problems:

• Undo sometimes does not work
• Implementing Undo is a lot of work

Cause: Manual Command Implementation
Dos #7: Use EMF Commands

Reasons:
• EMF Commands support undo/redo out of the box
• Allow integration with Transactional Command Stack
• Defines granularity of changes for other frameworks

Implementation:

```java
Command command = AddCommand.create(editingDomain, eObject, eStructuralFeature, referencedObjects);
editingDomain.getCommandStack().execute(command);
editingDomain.getCommandStack().undo();
```
Example #8

Problems:
- UI does not update to new values in the model
- “Widget is disposed” – Exceptions

Cause: Data binding buggy
Dos #8: Use Databinding

Reasons:

• Avoid update problems
• Databinding uses commands with undo/redo support
• Adapters are properly disposed off

Implementation:

```java
@override
public void bindValue() {
    IObserverableValue value = SWTObservables.observeText(text, SWT.FocusOut);
    IObserverableValue model = EMFEditObservables.observeValue(editingDomain, eObject, eStructualFeature);
    getDataBindingContext().bindValue(value, model);
}
```
Example #9

**Problem:** Deleting elements takes too much time

**Cause:** EcoreUtil.delete() is used for deletion
Dos #9: Carefully use EcoreUtil.delete()

Cause:
• Considers whole ResourceSet
• Slow if many Eobjects in ResourceSet

Mitigations:
• Make sure there are no or only known incoming cross-references
• Use Cross-Reference Adapter to speed up cross reference look-up
Don´ts #10: Build on generated editor

Reasons:
• Editor meant as an example only
• No defined extensibility, code is generated
• No update strategy to new versions of EMF

Mitigations:
• Use EMF reflection to build custom editor
• Use EEF
• Use EMF Client Platform
More information on EMF

- EMF Book
- EMF Newsgroup
- The EMF Source Code
- EMF Dos and Don´ts Blog Series
  - Google “EMF Dos and Don´ts
  - More blogs coming