
W-Jax 2007

München



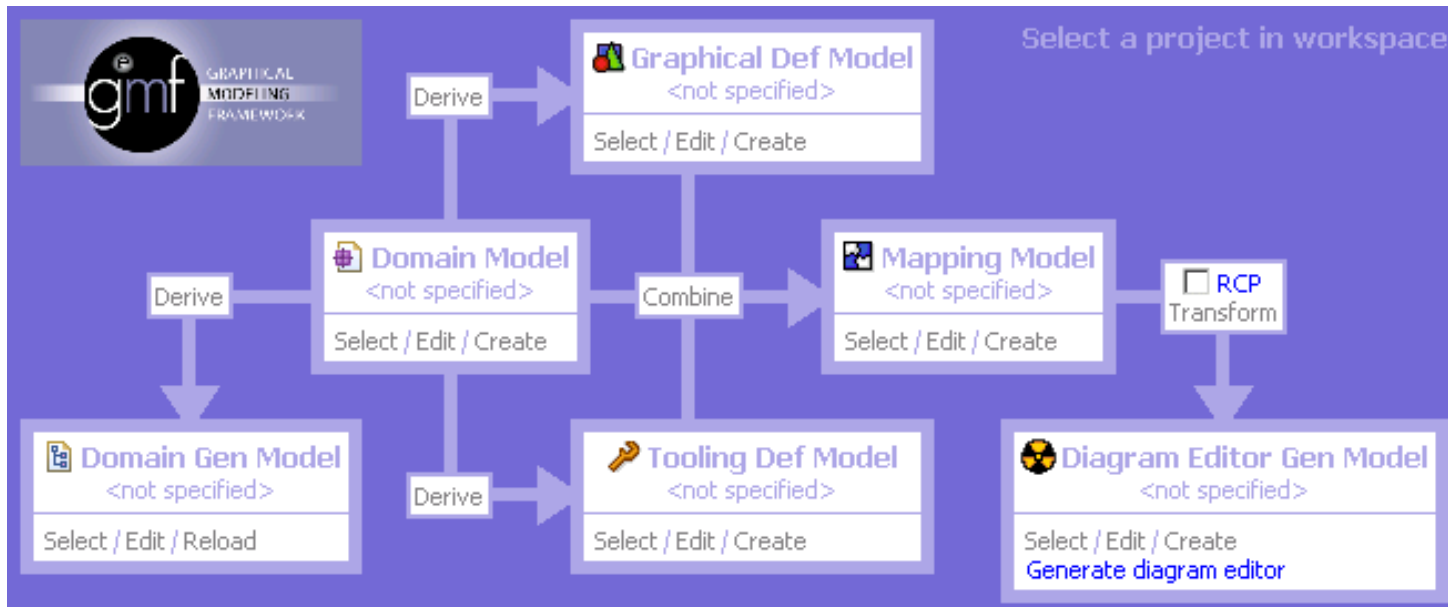
ex|Xcellent
solutions

MDSD in der Praxis

Customizing von graphischen DSL Editoren mit Eclipse GMF
von Andreas Schuster und Achim Demelt



- x| GMF Konzepte
- x| Wo sind die Stellschrauben im Code
- x| Motivation
- x| Erweiterungsbeispiele
- x| Best Practices für Customizing
- x| Fazit

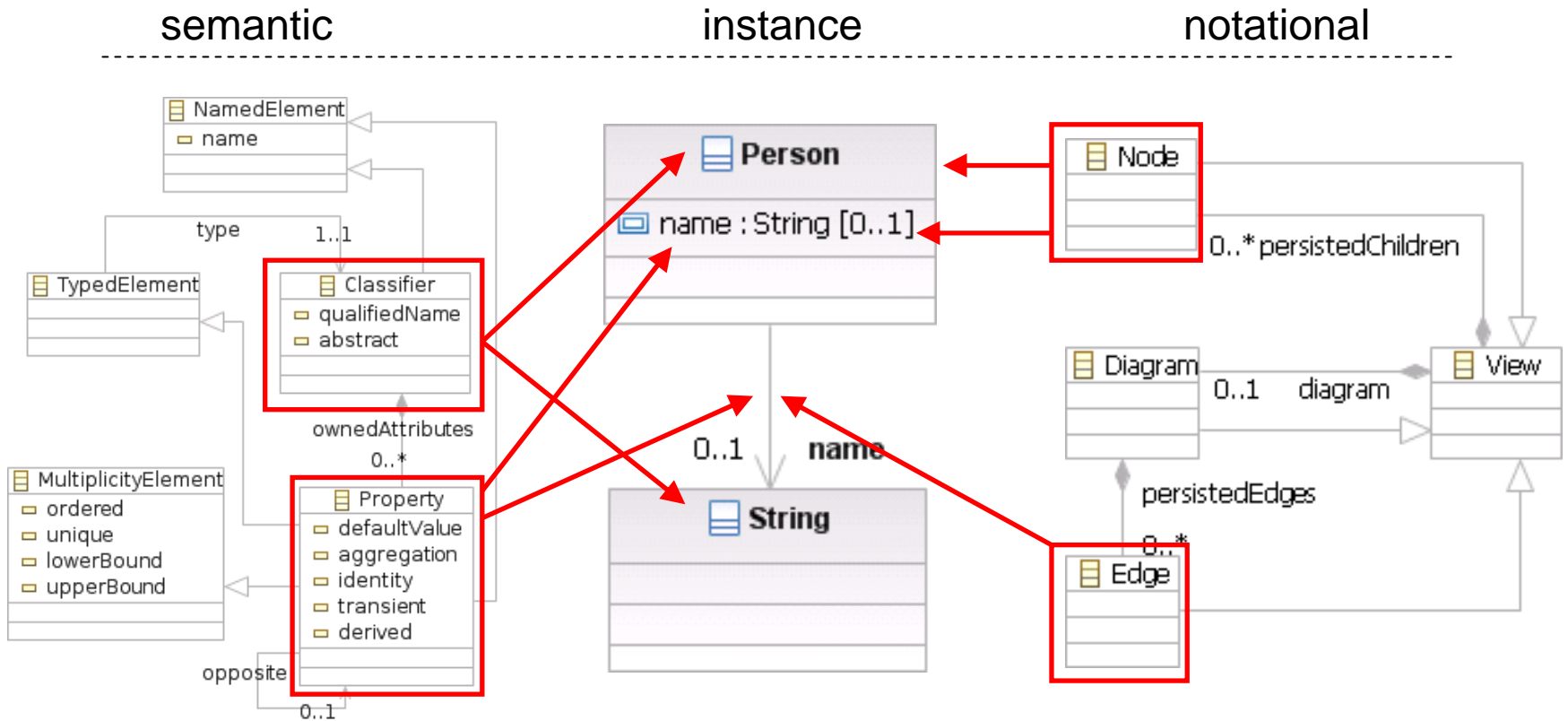


x| Voraussetzungen:

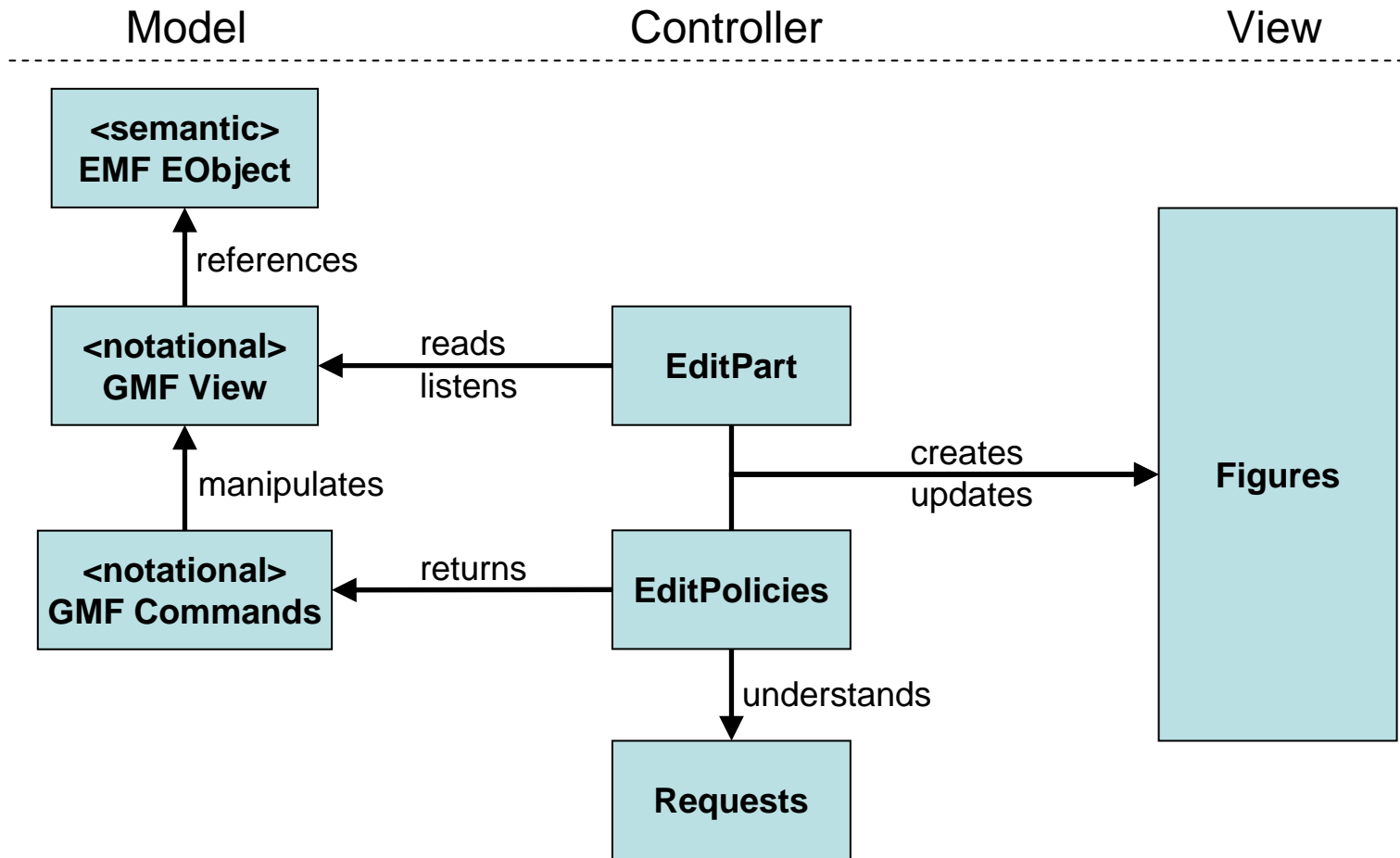
- | GMF
- | EMF
- | GEF

x| myeditor.gmfgen

x| GMF Runtime



x| GMF Runtime – MVC



x| Edit Parts

- | Modellinformationen:
 - semantic (EObject)
 - notational (View → Diagram, Node, Edge)
- | Registrieren von EditPolicies
- | Notification events (semantic und notational)
- | Zugang zu den Draw2D Figures
 - Position, Größe
 - Aktualisieren der Figures

x| Edit Policies

- | Kennt sein EditPart (Host)
- | Erhalten Anfragen (Requests)
- | Erzeugen Commands welche
 - das Modell ändern
 - die View ändern
- | Wichtige EditPolicies
 - CanonicalEditPolicy (synchronize semantic & notational)
 - ComponentEditPolicy (delete View / orphaned View)
 - SemanticEditPolicy (delete semantic)
 - DragDropEditPolicy
 - LayoutEditPolicy

x| Beispiel eines generierten GMF Editors für das Ecore Model

The screenshot shows the Eclipse IDE with a generated GMF editor for an Ecore model. The main window displays a class diagram with three classes: **Bibliothek**, **Buch**, and **Author**. The **Bibliothek** class has two associations: **bücher** (multiplicity 0..*) to the **Buch** class and **authoren** (multiplicity 0..*) to the **Author** class. The **Buch** class has two associations: **bücher** (multiplicity 0..*) to the **Author** class and **authoren** (multiplicity 0..*) to the **Author** class. The **Author** class has a **name** attribute. The interface includes a Project Explorer, Palette, Outline, and Properties view.

The Properties view shows the following table:

Property	Value
derived	false
editable	true
last modified	31. Oktober 2007 11:10:08
linked	false

x| Warum sind generierte Editoren nicht optimal?

- | Komfortfunktionen fehlen
- | Synchronisierung von Modell und Diagramm
- | Inplace-Editing nur für *EAttributes*
- | Mapping von Metamodell auf Diagrammelemente relativ starr
- | Grafische Darstellungsmöglichkeiten fehlen (Connection Router)

x| Synchronisierung mit dem Domänenmodell

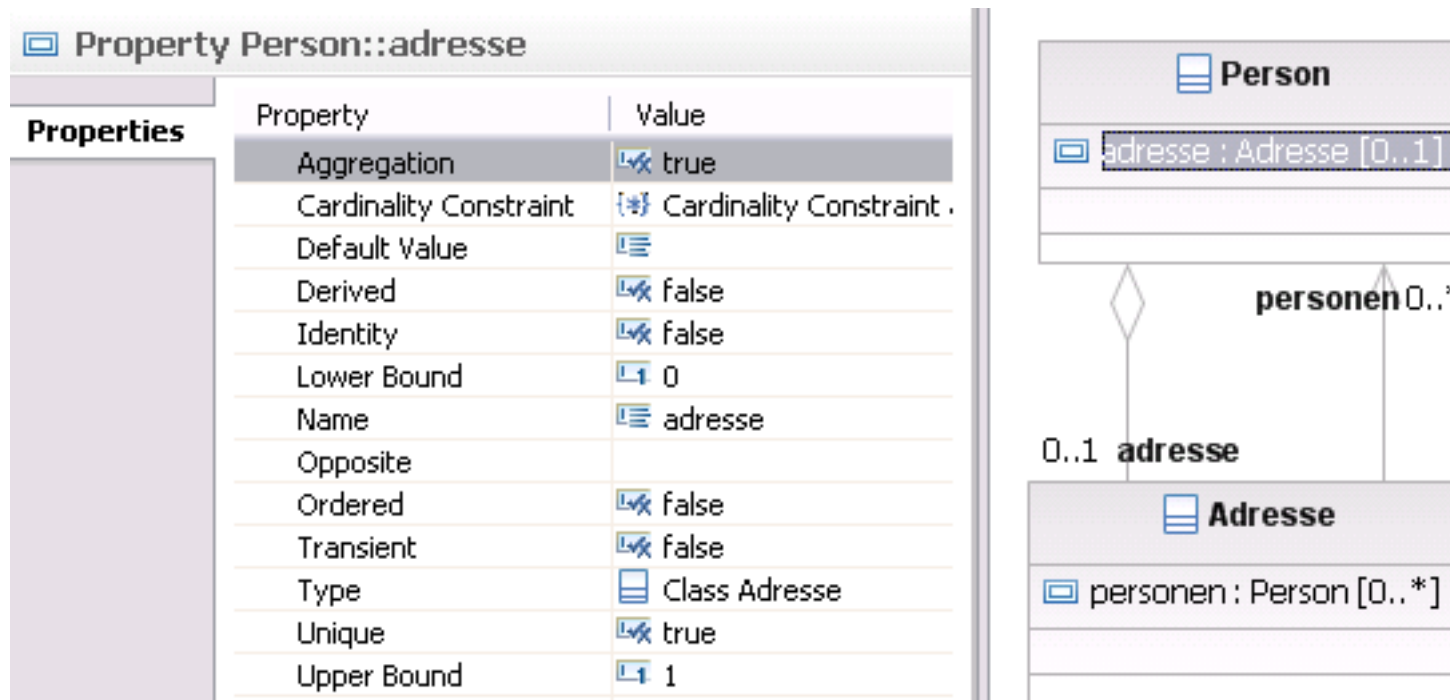
- | Triggern von Refresh-Mechanismen

x| Inplace-Editing und Content-Assist

- | Erweiterte String Parser
- | Field-Assist API

x| Synchronisierung mit dem Domänenmodell

- | **Ziel:** Modelländerungen führen auch zu Änderungen im Diagramm
- | **Ansatzpunkt:** CanonicalConnectionEditPolicy (und EditPart)
- | **Beispiel:** Update von Connections



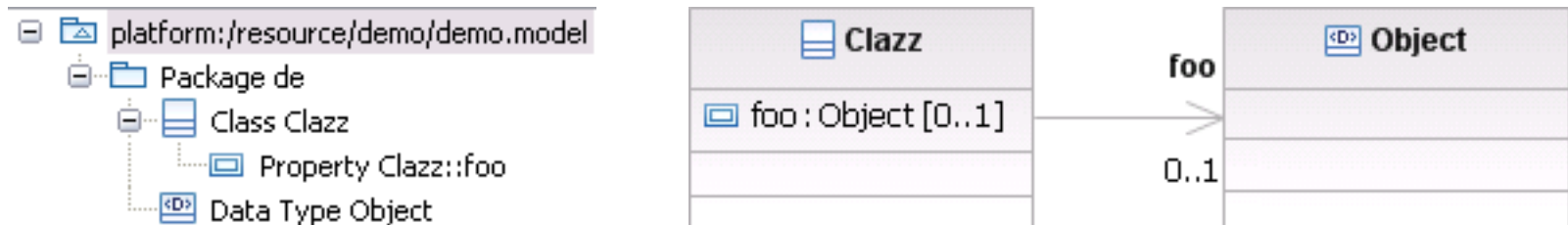
x| Synchronisierung mit dem Domänenmodell

| Synchronisiert

- Löschen eines Modellelements löscht auch View und umgekehrt
- Connection-Elemente sind *out of sync*, bei Mehrfach-Visualisierung
- Kein Drag&Drop vom Project Explorer

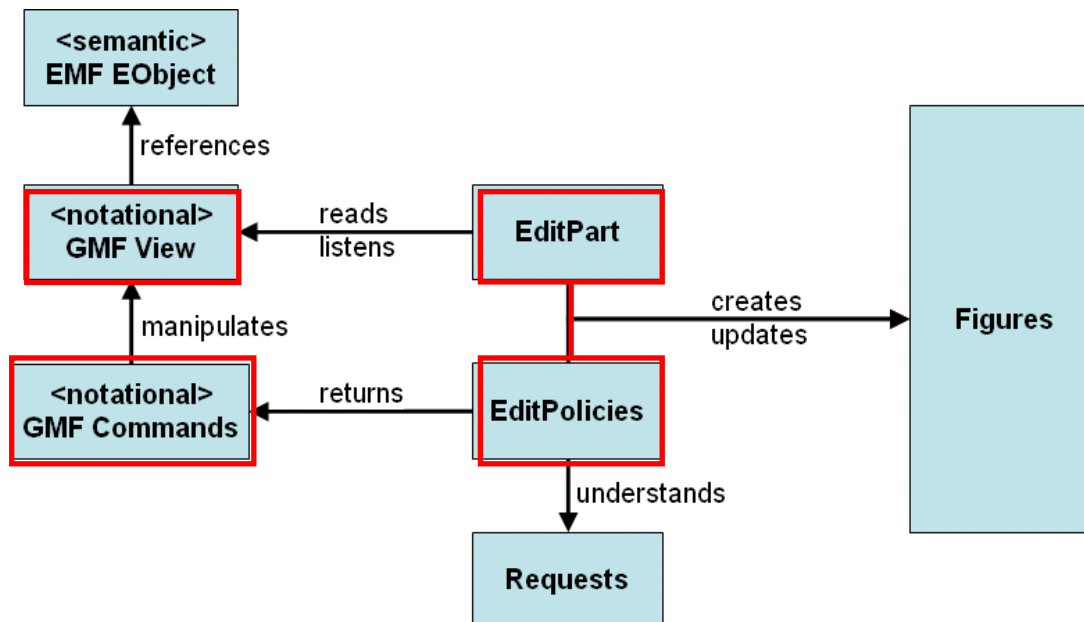
| Nicht-Synchronisiert

- Löschen einer View löscht nicht automatisch das Modellelement



x| Synchronisierung mit dem Domänenmodell

1. Events abfangen
2. Modelländerungen feststellen (vorher – nachher)
3. Aktualisierungen durchführen



x| Synchronisierung mit dem Domänenmodell

- | *CanonicalConnectionEditPolicy* spezialisieren
- | **Filtern** relevanter Events (Notification Mechanismus)
 - notational und semantic features

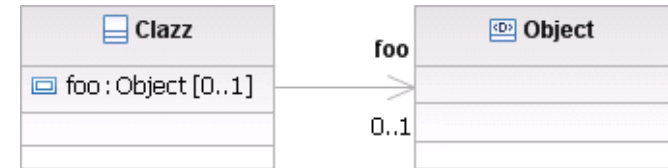
```
protected boolean shouldHandleNotificationEvent(Notification event) {
    Object notifier = event.getNotifier();
    Object feature = event.getFeature();
    // only trigger the relevant events
    if (notifier instanceof Property) {
        if (ModelPackage.eINSTANCE.getProperty_Aggregation().equals(feature) ||
            ModelPackage.eINSTANCE.getProperty_Opposite().equals(feature) ||
            ModelPackage.eINSTANCE.getTypeElement_Type().equals(feature)) {
            return true;
        }
    }
    return false;
}
```

x| Synchronisierung mit dem Domänenmodell

| Modeländerungen mittels Descriptoren feststellen

- generated *XYZLinkDescriptor*

| Aktualisierungen durchführen



```
private XYZLinkDescriptor oldDescriptor = null;
private XYZLinkDescriptor newDescriptor = null;

protected void refreshSemantic() {
    EObject modelElement = ((View) getHost().getModel()).getElement();
    oldDescriptor = newDescriptor;
    newDescriptor = XYZLinkDescriptor.createDescriptor(modelElement);
    [...]
    if(!newDescriptor.equals(oldDescriptor)) {
        [...] // remove obsolete Views, create new Views
    }
}
```

x| Synchronisierung mit dem Domänenmodell

| XYZCanonicalEditPolicy beim EditPart registrieren

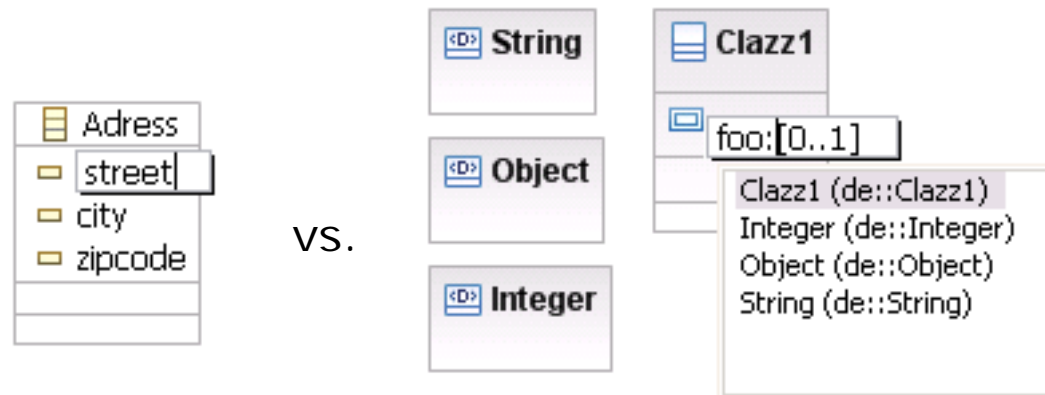
```
protected void createDefaultEditPolicies() {  
    super.createDefaultEditPolicies();  
    [...]  
    installEditPolicy(EditPolicyRoles.CANONICAL_ROLE,  
        new PropertyConnectionCanonicalEditPolicy());  
    [...]  
}
```

Synchronisierung mit dem Domänenmodell

Live Demo !

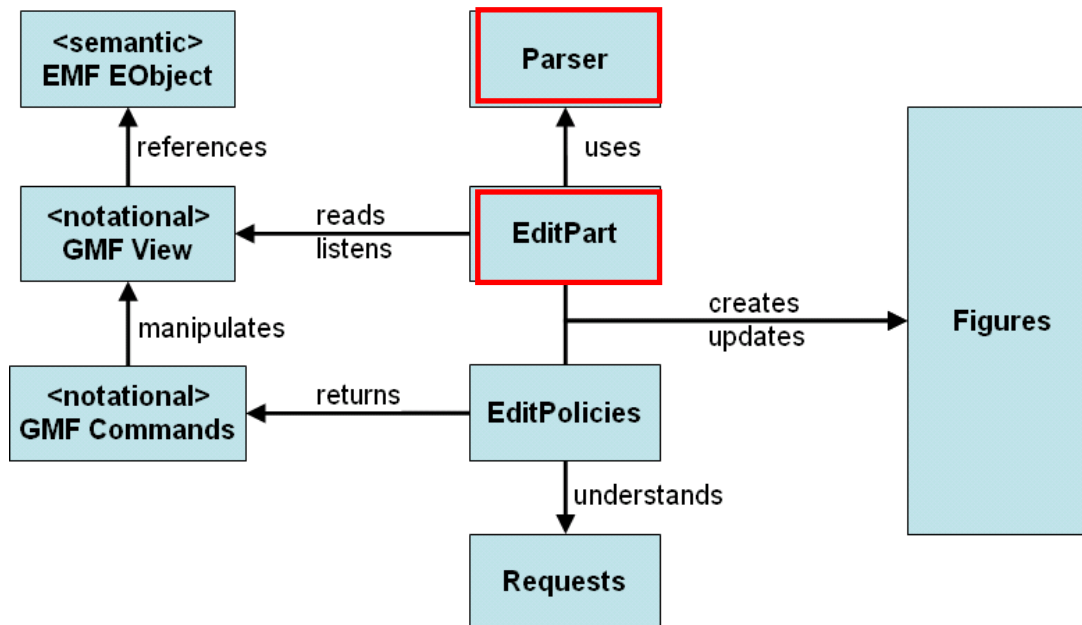
x| Inplace-Editing und Content-Assist

- | **Ziel:** komfortableres, effizienteres Editieren im Diagram
- | **Ansatzpunkt:**
 - EditPart, ParserProvider, MessageFormatParser, AbstractParser (IParser, ISemanticParser)
- | **Beispiel:** Abbildung von mehreren EAttributes/EReferences auf ein Label inklusive Content-Assist

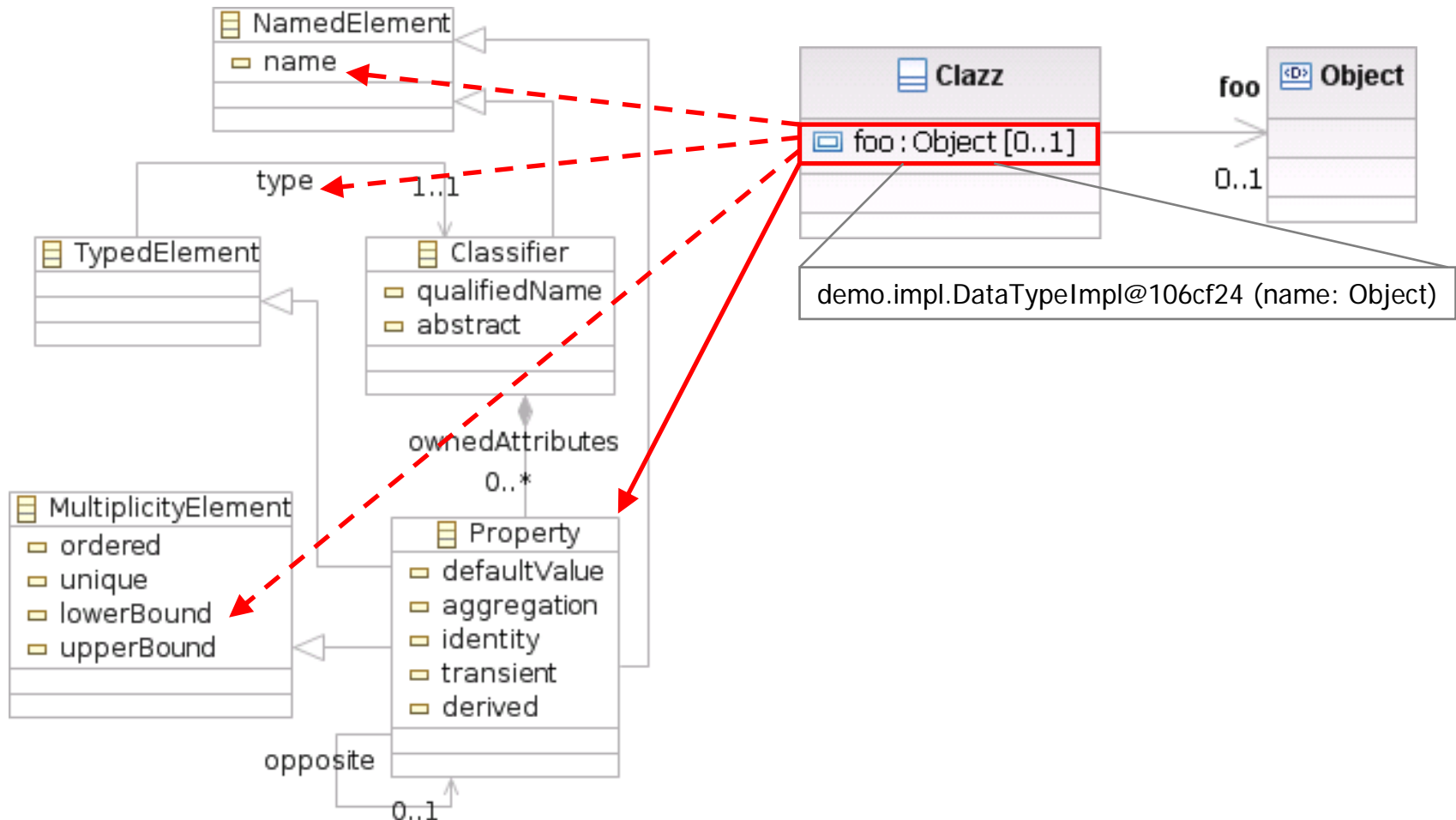


x| Inplace-Editing

1. String Parser modifizieren
2. ParserProvider anpassen



x| Inplace-Editing



x| Inplace-Editing

| *MessageFormatParser, AbstractParser*

- Suchen & Ersetzen: EAttribute → EStructuralFeature

| **EObject → String** parsen

- *AbstractParser.getValue()* modifizieren

```
protected Object getValue(EObject element, EStructuralFeature feature) {
    Object value = element.eGet(feature);
    Class iClass = feature.getEType().getInstanceClass();
    if(value instanceof Classifier) {
        value = ((Classifier) value).getName();
    }
    if (String.class.equals(iClass)) {
        if (value == null) {
            value = "";
        }
    }
    return value;
}
```

x| Inplace-Editing

| String → EObject parsen

- *AbstractParser.isValidNewValue()*
- z. B. Hashmap mit *Name:EObject*

```
protected Object getValidNewValue(EStructuralFeature feature, Object value) {
    EClassifier type = feature.getEType();
    [...]
    if (type instanceof EClass) {
        Class iClass = type.getInstanceClass();
        if (Classifier.class.equals(iClass)) {
            if (value instanceof String) {
                value = classifierProvider.getClassifierMap().get(value) != null
                    ? classifierProvider.getClassifierMap().get(value)
                    : new InvalidValue("invalid value");
            } else {
                value = new InvalidValue("invalid value");
            }
        }
    }
    return value;
}
```

x| Inplace-Editing

| *<Model>ParserProvider*

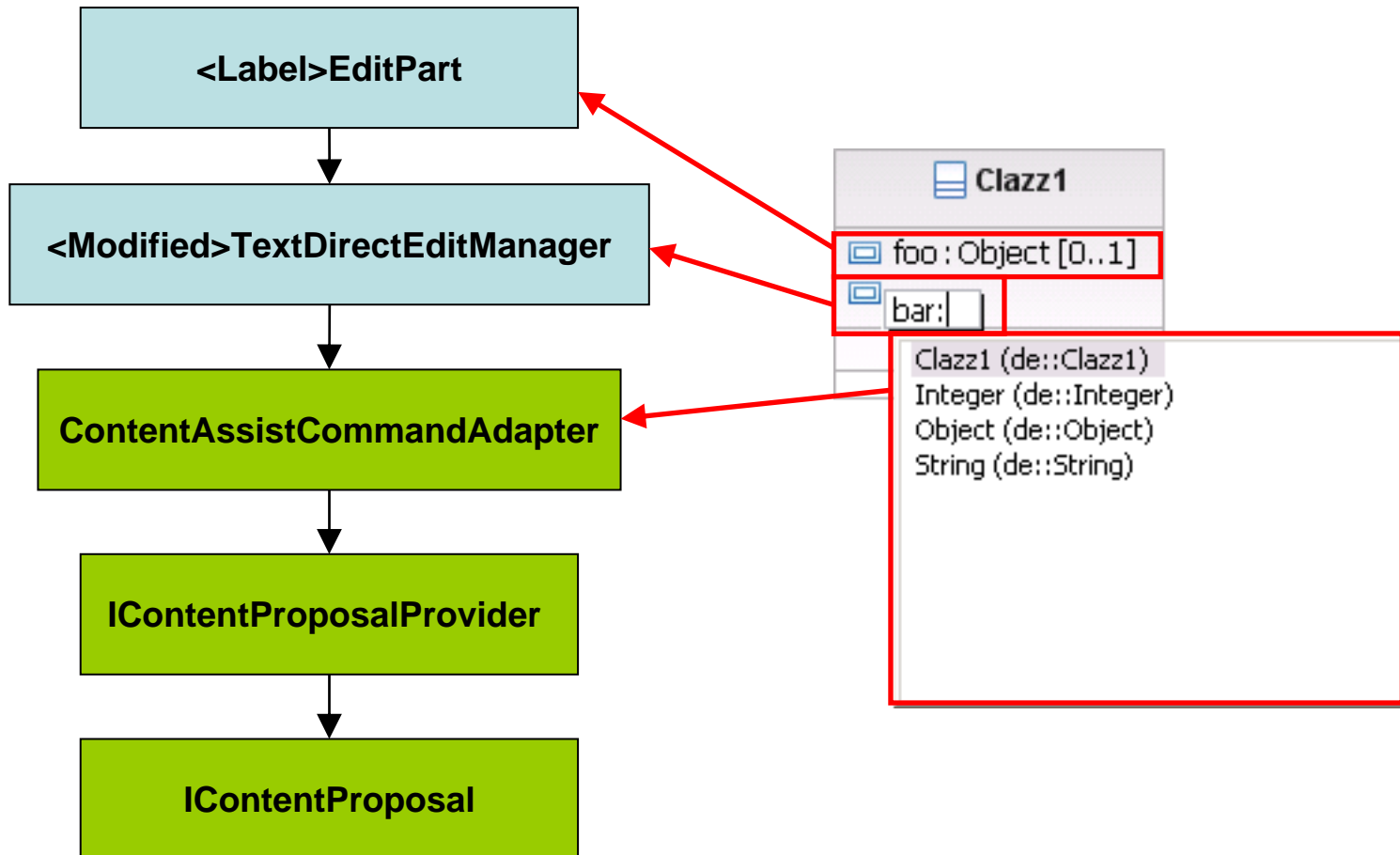
- Methode für entsprechendes Label (EditPart) modifizieren

| EStructuralFeature statt EAttribute

```
protected IParser createProperty_2001Parser() {
    EStructuralFeature[] features = new EStructuralFeature[] {
        ModelPackage.eINSTANCE.getNamedElement_Name(),
        ModelPackage.eINSTANCE.getTypedElement_Type(),
        ModelPackage.eINSTANCE.getMultiplicityElement_LowerBound(),
        ModelPackage.eINSTANCE.getMultiplicityElement_UpperBound()
    };
    MessageFormatParser parser = new MessageFormatParser(features);
    parser.setEditPattern("{0}:{1}[{2}..{3}]");
    parser.setEditorPattern("{0}:{1}[{2}..{3}]");
    parser.setViewPattern("{0}:{1}[{2}..{3}]");
    return parser;
}
```

x| Content-Assist

| org.eclipse.jface.fieldassist API einbinden



x| Content-Assist

- | *<Mod>TextDirectEditManager* → override *initCellEditor()*
- | *commit()* Bug (.jface.text.contentassist API!)
 - Focus geht verloren, wenn mit Maus selektiert wird

```
@Override
protected void initCellEditor() {
    super.initCellEditor();
    Text text = (Text) getCellEditor().getControl();
    char[] autoActivationCharacters = new char[] { ':' };
    String activationCommandID =
        "org.eclipse.ui.edit.text.contentAssist.proposals"; // Ctrl+Space

    ContentAssistCommandAdapter contentAdapter =
        new ContentAssistCommandAdapter(
            text, new TextContentAdapter(), getProposalProvider(),
            activationCommandID, autoActivationCharacters);
    [...]
}
```

x| Content-Assist

- | *XYZNameEditPart.getManager()* modifizieren
 - Erweiterte Version des *TextDirectEditManager* instanzieren

```
protected DirectEditManager getManager() {
    if (manager == null) {
        setManager(new ModifiedTextDirectEditManager(
            this,
            TextDirectEditManager.getTextCellEditorClass(this),
            XYZEditPartFactory.getTextCellEditorLocator(this)
        ));
    }
    return manager;
}
```

Inplace-Editing und Content-Assist

Live Demo !

- | Zusätzliche Klassen in *src-custom* Verzeichnis
- | Vererbung anstatt generierten Code ändern
- | Geänderte Methoden und Klassen markieren
 - "@generated" vs. "@generated not"
 - Achtung bei Neu-Generierung!
- | Code Formatter verwenden
 - Blank vs. Tab (SVN!)
- | Mapping Hierarchie (*.gmfmap) nicht verändern
 - Visual IDs der EditParts leiten sich daraus ab

- x| Alles ist möglich
- x| Es gibt mehrere Wege, um ans Ziel zu kommen
- x| Einarbeitungszeit nicht zu unterschätzen
 - | Erste Ergebnisse werden schnell erzielt
 - | Detailarbeiten benötigen Zeit
- x| Erfahrungen gewinnt man nur am konkreten Projekt
- x| Lohnt sich der Aufwand, einen generierten Editor zu erweitern, oder entwickelt man gegen die API?
 - generierter Code weist teilweise hohe Redundanz auf
 - Abstrahierungsgrad bei generierten EditParts eher gering
 - trotzdem enormer Mehrwert durch generierten Code (EditPolicies, Factories, Providers, sonstige Utilities)

x| Artikel zu GMF: „KOMPLETT EFFIZIENT“

- | „Customizing eines mit GMF generierten Diagrammeditors“
- | Im kommenden „Eclipse Magazin“
- | In der englischen Online Ausgabe des “Eclipse magazine” (Termin noch nicht bekannt)

x| eXXcellent solutions GmbH Ulm

- | Beim Alten Fritz 2
- | 89075 Ulm
- | <http://www.exxcellent.de>

**x| Haben Sie weitere Fragen zum Thema?
Dann besuchen Sie uns auch am Stand.**

Diskussion!