Adaption domänenspezifischer Modellierungswerkzeuge am Beispiel bflow* Toolbox

Seminar Modellgetriebene Softwareentwicklung
Endpräsentation

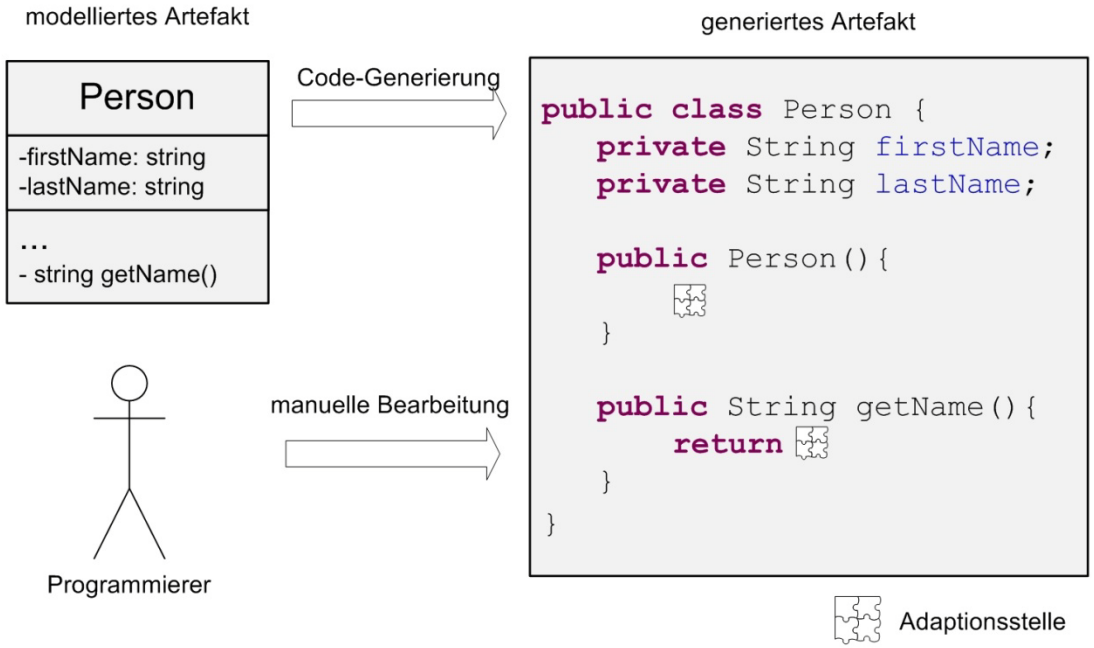
Jörg Hartmann

Gliederung

- **Wiederholung**
  - Problem
  - Zielsetzung
- **bflow* Toolbox**
  - Beispiele
- **Einführung in Adaptionstechniken**
  - Adaption am Code
  - Objektorientierung
  - Adaption am Template
  - Aspektorientierung
- **Vorgehen**
- **Bewertung**
- **Fazit**

Wiederholung: Problem

- **Warum muss generierter Code adaptiert werden?**



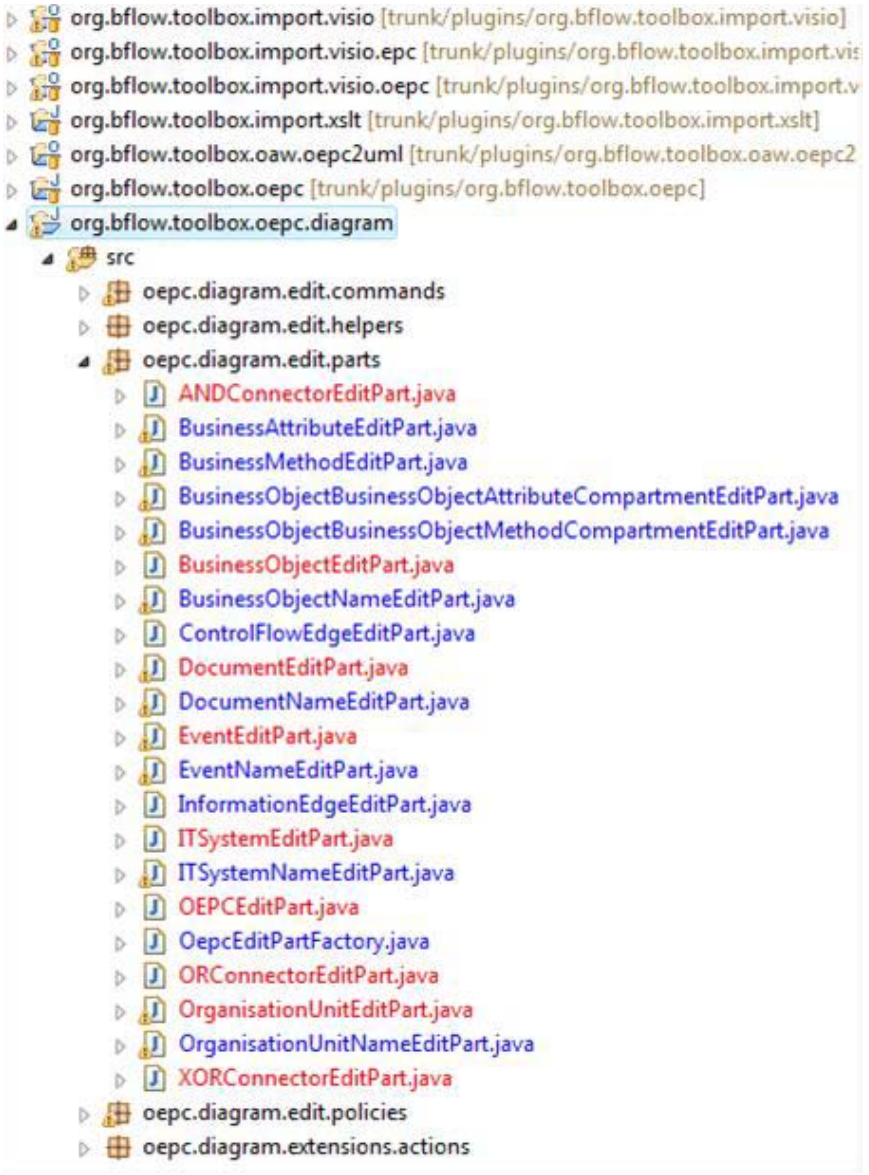
- **Mischform birgt Probleme**
  - Redundanz
  - höhere Aufwand
  - höhere Fehlerwahrscheinlichkeit
- **Was passiert mit individuellen Anteilen zur Neugenerierung?**

Wiederholung: Zielsetzung

- **Wie lassen sich Modellierungswerkzeuge effektiv adaptieren?**
  - Wie lässt sich generierter und individueller Code trennen?
  - Wie kann man Code-Redundanz verringern?
  - Welche Techniken gibt es?
  - Nach welchen Kriterien werden diese bewertet?
  - Wie kann der Aufwand zum Plattformwechsel gering gehalten werden?

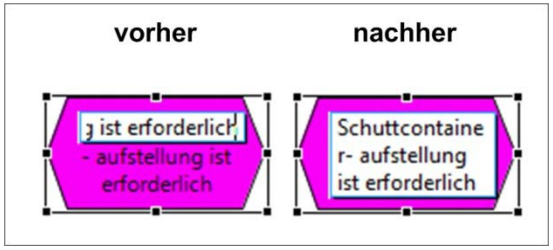
bflow* Toolbox

- **generiertes Modellierungswerkzeug**
  - EPK, oEPK, VC
- **Funktionalität durch Adaption hinzugefügt**
- **Plattform:**
  - EMF 2.4.2
  - GEF 3.4.2,
  - GMF 2.1.3
- **Neugenerierung?**
- **neue Elemente in bestehenden Editoren?**
- **Plattformwechsel?**

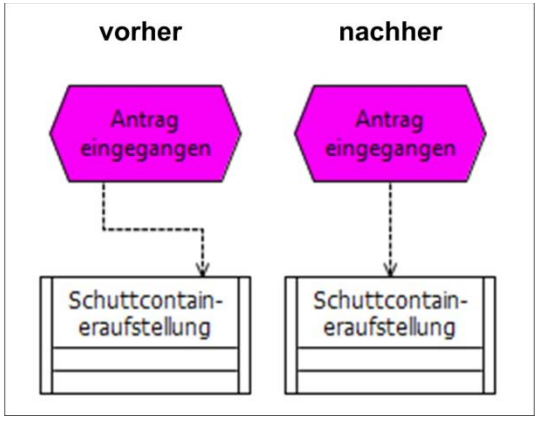


bflow* Toolbox – Beispiele indiviudellen Codes

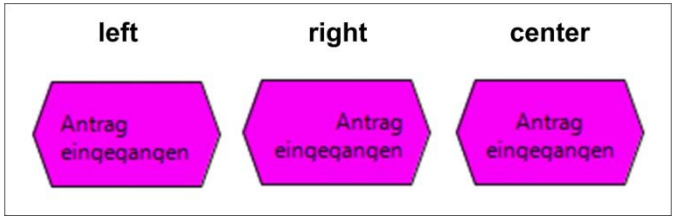
- **Celleditor**



- **Ankerpunkt**



- **Textalignment**



Einführung in Adaptionstechniken

- **Adaption am Code**
  - individuelle Anteile im generierten Code
  - Problemverursacher!
  - erhöht Redundanz

  - Code kann zur Neugenerierung geschützt werden
  - Protected Regions

```
/**
 * @generated NOT
 */
protected DirectEditManager getManager(){
    // method stub
}
```

- **Adaption am Generator**
  - GMF beschreibt Struktur des Codes durch Templates
  - Ansatz zur Adaption im Template, statt im Code selbst
  - erzeugt kontrollierte Redundanz
  - aber, Template gehört zur Plattform

TextAware.xpt

```
13
14«IMPORT 'http://www.eclipse.org/gmf/2009/GenModel'»
15«EXTENSION xpt::diagram::ViewmapAttributesUtils»
16
17«DEFINE fields FOR gmfgen::GenCommonBase-»
18    «EXPAND xpt::Common::generatedMemberComment»
19    private org.eclipse.gef.tools.
20        DirectEditManager manager;
21
22    «EXPAND xpt::Common::generatedMemberComment»
23    private org.eclipse.gmf.runtime.common.
24        ui.services.parser.IParser parser;
25
26    «EXPAND xpt::Common::generatedMemberComment»
27    private java.util.List parserElements;
28
29    «EXPAND xpt::Common::generatedMemberComment»
30    private String defaultText;
31«ENDDEFINE»
```

Einführung in Adaptionstechniken

- **Adaption durch Code Seperation**
  - Versuch Funktionalität zu kapseln
  - redundante Anteile werden wiederverwendet und externalisiert

- **Zwei Ansätze:**
  - objektorientierte Adaption
  - aspektorientierte Adaption

- **aspektorientierte Adaption**
  - Joinpoint – ein Punkt im Program, wird über Advices erweitert wird
  - Advice – Aktion, die beim erreichen eines Joinpoints durchgeführt wird – before, after, around
  - Pointcut – ein oder mehrere Joinpoints

- **Umsetzung durch Object Teams in Eclipse**
  - (OT/Equinox)
  - Plugin-Klassen durch Aspekte erweitert
  - aspect bindings

```
<extension>
   <aspectBinding>
      <basePlugin id="org.eclipse.gmf.runtime.diagram.ui">
      </basePlugin>
   <team activation="ALL_THREADS"
      class="org.bflow.toolbox.diagram.extensions.ot.BflowVisualTeam">
   </team>
</extension>
```

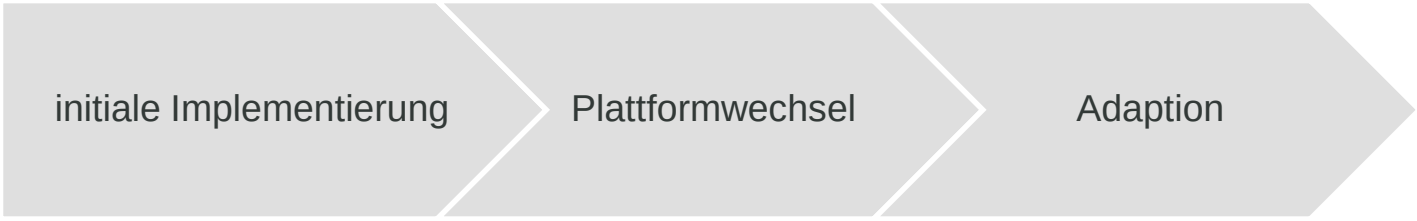
- **Objekt Teams – Umsetzung**
- **Basiskonzepte**
  - Teams und Rollen

```
public team class BflowVisualTeam {

   public class TextAlignmentRole playedBy ShapeNodeEditPart
   base when (BflowOTNegotiator.requestPermission(base)){

        public void handleNotificationEvent(Notification event){
                if(event.getNotifier() instanceof TextStyle){
                        refreshTextAlignment(TransactionContext.Write);
                }
        }
        handleNotificationEvent <- after handleNotificationEvent;

        public void refreshTextAlignment(final TransactionContext context){
                final EClass eClass = NotationPackage.Literals.TEXT_STYLE;

                final TextStyle textStyle = (TextStyle) new LoadStyleAction(
                                getPrimaryView(), eClass, context).getStyle();

                if(textStyle != null){
                        IFigure figure = getPrimaryShape();
                        applyTextAlignment(figure, textStyle.getTextAlignment());
                }
        }

        public IFigure getPrimaryShape() -> IFigure getNodeFigure();
        public View getPrimaryView() -> View getPrimaryView();
...
```

joinpoint
advice
callout
guard

Vorgehen

- **Implementierung und Adaption**
- **Festhalten des Aufwandes**
- **späterer Vergleich**



Bewertung

- **Kriterien**
  - Lernaufwand

Technik	Lernaufwand
direkte Adation	kein
OO	gering
Template	mittel
AO	hoch

  - Anzahl der Adaptionsstellen a
  - Anzahl der adaptierten Artefakte A
  - Adaptionsverhältnis r=a/A
  - Lines of Code

Bewertung - Textalignment

- **Grundlagen**
  - refreshVisuals() und handleNotificationEvent(Notification) überschreiben
  - dort ist die neue Ausrichtung abzufangen und an das Label zu leiten
  - 23 Klassen, alle Instanzen von ShapeNodeEditPart, die ein Label verwalten
  - 4 Hilfsmethoden

```
public void handleNotificationEvent(Notification event) {
        super.handleNotificationEvent(event);
        if (event.getNotifier() instanceof TextStyle) {
                refreshTextAlignment();
        }
}

public void refreshVisuals() {
        super.refreshVisuals();
        refreshTextAlignment();
}
```

Bewertung - Textalignment

- **Adaption am Code**
  - in jeder der 23 Subklassen sind die 6 Methoden hinzuzufügen
- **Implementierung**
  - pro Klasse 6 Adaptionsstellen,
  - 37 Zeilen Code
  - 23 Klassen
  - A = 23, a = 138, r = 6, LoC = 851
- **Adaption**
  - Funktion erneut hinzuzufügen
  - Analog Implementierung
  - kein Gewinn!

Bewertung - Textalignment

- **Adaption am Generator**
  - 2 Templates; diagram\editparts\NodeEditPart und impl\diagram\editparts\NodeEditPart
  - ersteres übernimmt Aufruf der Define Blöcke

- **naiver Ansatz**
  - Code in impl Template schreiben und aufrufen
  - A = 2, a = 4, r = 2, LoC = 57

```
«DEFINE handleTextAlignment FOR gmfgen::GenNode-»
public void handleNotificationEvent(Notification event){
        super.handleNotificationEvent(event);
        if(event.getNotifier() instanceof TextStyle){
                refreshTextAlignment();
        }
}

public void refreshVisuals(){
        super.refreshVisuals();
        refreshTextAlignment();
}
«EXPAND refreshTextAlignment»
«ENDDEFINE»
```

Bewertung - Textalignment

- **besser:**
  - hohe Anzahl Code im Template
  - benutze eigenes Template!

- **Implementierung**
  - neues Template entsteht, erhöht Aufwand
  - A = 2, a = 5, r = 2.5, LoC = 60

- **Adaption**
  - vorher implementiertes Template hinzufügen
  - A = 2, a = 1, r = 0.5, LoC = 1

Bewertung - Textalignment

- **objektorientierte Adaption**
  - Einführung einer neuen Superklasse
  - A = 1, a = 8 (6 Methoden + Konstruktor + Klassendeklaration), Loc = 39

- **Implementierung**
  - hinzufügen der Superklasse
  - Adaptierung der Vererbungshierarchie
  - addiert sich: a = A = LoC = 23
  - A = 24, a = 31, r = 2.21, LoC = 62

- **Adaption**
  - nur Superklasse hinzufügen und Vererbungshierarchie adaptieren
  - A = 24, a = 23, r = 0.95, LoC = 23

Bewertung - Textalignment

- **aspektorientierte Adaption**
  - neues Plugin
  - eine Rolle, die von ShapeNodeEditPart gespielt wird
  - siehe Abbildung

- **Implementierung**
  - A = 4, a = 12, r = 3,
  - LoC = 70

- **Adaption**
  - Team wiederverwenden
  - A = 1!
  - a = r = LoC = 0!

```
public team class BflowVisualTeam {

   public class TextAlignmentRole playedBy ShapeNodeEditPart
   base when (BflowOTNegotiator.requestPermission(base)){

        public void handleNotificationEvent(Notification event){
               if(event.getNotifier() instanceof TextStyle){
                      refreshTextAlignment(TransactionContext.Write);
               }
        }
        handleNotificationEvent <- after handleNotificationEvent;

        public void refreshTextAlignment(final TransactionContext context){
               final EClass eClass = NotationPackage.Literals.TEXT_STYLE;

               final TextStyle textStyle = (TextStyle) new LoadStyleAction(
                             getPrimaryView(), eClass, context).getStyle();

               if(textStyle != null){
                      IFigure figure = getPrimaryShape();
                      applyTextAlignment(figure, textStyle.getTextAlignment());
               }
        }

        public IFigure getPrimaryShape() -> IFigure getNodeFigure();
        public View getPrimaryView() -> View getPrimaryView();
...
```

Bewertung - Überblick

Celleditor

Technik	*a*	*A*	*r*	*LoC*
Adaption am Code	23 | 23	23 | 23	1 | 1	23 | 23
Adaption am Template	1 | 1	1 | 1	1 | 1	1 | 1
Objektorientierte Adaption	72 | 69	24 | 24	3 | 2.875	41 | 23
Aspektorientierte Adaption	73 | 0	5 | 1	14.6 | 0	303 | 0

Ankerpunkt

Technik	*a*	*A*	*r*	*LoC*
Adaption am Code	118 | 116	30 | 30	3.93 | 3.86	406 | 348
Adaption am Template	5 | 4	3 | 3	1.67 | 1.33	89 | 21
Objektorientierte Adaption	34 | 29	30 | 30	1.13 | 0.96	47 | 29
Aspektorientierte Adaption	9 | 0	4 | 1	2.25 | 0	60 | 0

Textalignment

Technik	*a*	*A*	*r*	*LoC*
Adaption am Code	138 | 138	23 | 23	6 | 6	851 | 851
Adaption am Template	5 | 1	3 | 2	1.66 | 0.5	60 | 1
Objektorientierte Adaption	29 | 23	24 | 23	1.2 | 1	62 | 23
Aspektorientierte Adaption	12 | 0	4 | 1	3 | 0	70 | 0

Fazit

- **Adaption am Code**
  - uneffektiv, erzeugt Redundanz
  - aber, keine Protected Regions genutzt
- **objektorientierte Adaption**
  - verhindert Redundanz
  - hoher Aufwand zur Adaption der Vererbungshierarchie
- **Adaption am Template**
  - Plattformwechsel erhöhte Aufwand!
  - insgesamt geringer Aufwand für Implementierung und Adaption
  - für Adaptionen am Generat selbst
- **aspektorientierte Adaption**
  - effiziente Trennung vom generierten Code
  - hoher Implementierungsaufwand für Adaption am Generat
  - besser für Adaptionen am Framework
  - Ermöglich getrennte Entwicklung von Editor - Funktion