ArgoUML Anwenderhandbuch

Eine Lernanleitung und eine Referenzbeschreibung

Alejandro Ramirez
Philippe Vanpeperstraete
Andreas Rueckert
Kunle Odutola
Jeremy Bennett
Linus Tolke
Michiel van der Wulp
Übersetzung: Harald Braun
Zusammenfassung

Diese Version des Handbuchs beschreibt die Version 0.33.2 von ArgoUML.

# Inhaltsverzeichnis

1. Vorwort ............................................................................................................ xviii
1. Einleitung .............................................................................................................. 1
   1.1. Die Anfänge und ein Überblick über ArgoUML ................................................. 1
      1.1.1. Objektorientierte Analyse und Design ..................................................... 1
      1.1.2. Die Entwicklung von ArgoUML ............................................................ 1
      1.1.3. Mehr über das ArgoUML-Projekt .......................................................... 2
   1.2. Der Bezugsbereich dieses Anwenderhandbuches ................................................. 2
      1.2.1. Der Leserkreis .................................................................................... 3
      1.2.2. Bezugsbereich .................................................................................... 3
   1.3. Überblick über das Anwenderhandbuch ............................................................. 3
      1.3.1. Die Struktur des Übungshandbuches ....................................................... 3
      1.3.2. Die Struktur des Referenzhandbuches ..................................................... 4
      1.3.3. Anwender-Feedback ............................................................................ 4
   1.4. Annahmen ................................................................................................... 4
1. Übungsanleitung ...................................................................................................... 5
2. Einleitung ...................................................................................................... 6
3. UML basierte OOA&D ..................................................................................... 7
   3.1. Hintergrundinformationen zu UML .......................................................... 7
   3.2. UML basierter Prozess für OOA&D ......................................................... 7
      3.2.1. Prozesstypen .............................................................................. 8
      3.2.2. Der Entwicklungsprozess für dieses Übungshandbuch ..................... 12
   3.3. Warum ist ArgoUML anders ................................................................. 13
      3.3.1. Kognitive Psychologie ............................................................... 13
      3.3.2. Offene Standards ...................................................................... 14
      3.3.3. 100% reines Java ...................................................................... 16
      3.3.4. Offener Quellcode (Open Source) .............................................. 16
   3.4. ArgoUML Grundlagen ......................................................................... 16
      3.4.1. Erste Schritte ........................................................................... 16
      3.4.2. Die ArgoUML-Anwenderschnittstelle ........................................... 20
      3.4.3. Ausgabe .................................................................................. 32
      3.4.4. Arbeiten mit Design-Kritiken ...................................................... 35
   3.5. Die Fallstudie (Noch zu schreiben) ........................................................... 38
4. Erfassen der Anforderungen ............................................................................. 40
   4.1. Einleitung .......................................................................................... 40
   4.2. Der Anforderungs-Erfassungs-Prozess .................................................... 40
      4.2.1. Prozess-Schritte ........................................................................ 41
   4.3. Ergebnis des Anforderungs-Erfassungs-Prozesses ...................................... 42
      4.3.1. Visions-Dokument .................................................................... 42
      4.3.2. Anwendungsfalldiagramm .......................................................... 42
      4.3.3. Die Anwendungsfall-Spezifikation ............................................. 47
      4.3.4. Ergänzende Anforderungsspezifikation ....................................... 50
   4.4. Anwendungsfälle in ArgoUML verwenden .............................................. 51
      4.4.1. Akteure ................................................................................... 51
      4.4.2. Anwendungsfälle ...................................................................... 52
      4.4.3. Assoziationen .......................................................................... 53
      4.4.4. Hierarchische Anwendungsfälle ............................................... 55
      4.4.5. Stereotypen ............................................................................. 56
      4.4.6. Dokumentation ......................................................................... 57
      4.4.7. Systemgrenzen ......................................................................... 57
   4.5. Fallstudie ........................................................................................... 58
      4.5.1. Das Dokument Vision ................................................................ 58
      4.5.2. Akteure und Anwendungsfälle identifizieren .................................. 60
      4.5.3. Assoziationen (Noch zu beschreiben) ........................................... 60
<table>
<thead>
<tr>
<th>Seite</th>
<th>Kapitel</th>
<th>Unterkapitel</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>4.5.4</td>
<td>Erweiterte Diagrammfunktionen</td>
<td>Noch zu beschreiben</td>
</tr>
<tr>
<td>60</td>
<td>4.5.5</td>
<td>Anwendungsfallspezifikationen</td>
<td>Noch zu beschreiben</td>
</tr>
<tr>
<td>60</td>
<td>4.5.6</td>
<td>Ergänzende Anforderungsspezifikation</td>
<td>Noch zu beschreiben</td>
</tr>
<tr>
<td>61</td>
<td>5.1</td>
<td>Der Analyseprozess</td>
<td>........................................</td>
</tr>
</tbody>
</table>
6.3.2. Beziehungen zwischen Paketen (Noch zu beschreiben) .................... 70
6.3.3. Erweiterte Paketfunktionen (Noch zu beschreiben) .......................... 70

6.4. Mehr über Klassendiagramme (Noch zu beschreiben) ........................ 70
  6.4.1. Das Klassendiagramm (Noch zu beschreiben) ............................... 70
  6.4.2. Erweiterte Klassendiagramme (Noch zu beschreiben) ..................... 71

6.5. Mehr über Klassendiagramme in ArgoUML (Noch zu beschreiben) ......... 71
  6.5.1. Klassen (Noch zu beschreiben) ................................................ 71
  6.5.2. Klassenattribute und -operationen (Noch zu beschreiben) ................ 71
  6.5.3. Erweiterte Klassenfunktionen .................................................. 71

6.6. Sequenz- und Kollaborationsdiagramme (Noch zu beschreiben) ............ 73
  6.6.1. Mehr über das Sequenzdiagramm (Noch zu beschreiben) ................. 74
  6.6.2. Das Kollaborationsdiagramm (Noch zu beschreiben) ....................... 74
  6.6.3. Erweiterte Kollaborationsdiagramme (Noch zu beschreiben) .......... 74

6.7. Kollaborationsdiagramme in ArgoUML erstellen (Noch zu beschreiben) ... 74
  6.7.1. Kollaborationsdiagramme (Noch zu beschreiben) ........................... 74
  6.7.2. Nachrichten (Noch zu beschreiben) .......................................... 74
  6.7.3. Erweiterte Kollaborationsdiagramme (Noch zu beschreiben) ........... 74

6.8. Zustandsdiagramme (Noch zu beschreiben) ....................................... 74
  6.8.1. Das Zustandsdiagramm (Noch zu beschreiben) ............................. 75
  6.8.2. Erweiterte Zustandsdiagramme (Noch zu beschreiben) .................... 75

6.9. Zustandsdiagramme in ArgoUML erstellen (Noch zu beschreiben) ........... 75
  6.9.1. Zustandsdiagramme (Noch zu beschreiben) .................................... 75
  6.9.2. Zustände (Noch zu beschreiben) .............................................. 75
  6.9.3. Transitionen (Noch zu beschreiben) ......................................... 75
  6.9.4. Aktionen (Noch zu beschreiben) ............................................. 76
  6.9.5. Erweiterte Zustandsdiagramme (Noch zu beschreiben) .................... 76

6.10. Aktivitätsdiagramme (Noch zu beschreiben) ..................................... 76
  6.10.1. Das Aktivitätsdiagramm (Noch zu beschreiben) .......................... 76

6.11. Aktivitätsdiagramme in ArgoUML erstellen (Noch zu beschreiben) ...... 76
  6.11.1. Aktivitätsdiagramme (Noch zu beschreiben) ................................ 76
  6.11.2. Aktionszustände (Noch zu beschreiben) ...................................... 77

6.12. Verteilungsdiagramme (Noch zu beschreiben) .................................. 77
  6.12.1. Das Verteilungsdiagramm (Noch zu beschreiben) ......................... 77

6.13. Verteilungsdiagramme in ArgoUML erstellen (Noch zu beschreiben) ...... 77
  6.13.1. Knoten (Noch zu beschreiben) ............................................... 77
  6.13.2. Komponenten (Noch zu beschreiben) ........................................ 77
  6.13.3. Beziehungen zwischen Knoten und Komponenten (Noch zu beschreiben) .. 77


6.15. Fallstudie (Noch zu beschreiben) ................................................ 78
  6.15.1. CRC-Karten (Noch zu beschreiben) .......................................... 78
  6.15.2. Pakete (Noch zu beschreiben) ................................................ 78
  6.15.3. Klassendiagramme (Noch zu beschreiben) .................................... 78
  6.15.4. Sequenzdiagramme (Noch zu beschreiben) ................................... 78
  6.15.5. Kollaborationsdiagramme (Noch zu beschreiben) .......................... 78
  6.15.6. Zustandsdiagramme (Noch zu beschreiben) .................................. 78
  6.15.7. Aktivitätsdiagramme (Noch zu beschreiben) ................................ 78
  6.15.8. Das Verteilungsdiagramm (Noch zu beschreiben) .......................... 78
  6.15.9. Die System-Architektur (Noch zu beschreiben) ............................ 79

7. Codegenerierung, Reverse Engineering und Round Trip Engineering ........... 80
  7.1. Einleitung ....................................................................................... 80
  7.2. Codegenerierung ............................................................................. 80
    7.2.1. Code aus der statischen Struktur generieren .............................. 80
    7.2.2. Code aus Interaktionen und Zustandsautomaten generieren .......... 81
  7.3. Codegenerierung in ArgoUML .......................................................... 82
    7.3.1. Statische Struktur ..................................................................... 82
    7.3.2. Interaktionen und Zustandsdiagramme ...................................... 82
  7.4. Reverse Engineering ...................................................................... 82
10.5.2. Suchen... ................................................................. 118
10.5.3. Zoom ........................................................................... 120
10.5.4. Gitter einstellen .......................................................... 121
10.5.5. Einrasten einrichten ....................................................... 121
10.5.6. Seitennumbrüche ......................................................... 121
10.5.7. Symbolleisten ............................................................. 122
10.5.8. XML-Quelltext ............................................................ 122
10.6. Das Menü "Neues Diagramm" ........................................... 122
10.6.1. Anwendungsfalldiagramm ............................................ 122
10.6.2. Klassendiagramm ........................................................ 122
10.6.3. Sequenzdiagramm ....................................................... 123
10.6.4. Kollaborationsdiagramm ............................................. 123
10.6.5. Zustandsübergangsdiagramm ....................................... 123
10.6.6. Aktivitätsdiagramm ..................................................... 123
10.6.7. Verteilungsdiagramm .................................................... 124
10.7. Das Menü Anordnen ........................................................ 124
10.7.1. Ausrichten ................................................................. 124
10.7.2. Anordnen ................................................................. 125
10.7.3. Reihenfolge ............................................................... 125
10.7.4. Grösse an Inhalt anpassen .......................................... 125
10.7.5. Layout ......................................................................... 126
10.8. Das Menü Generieren ...................................................... 126
10.8.1. Markierte Klassen generieren ...................................... 126
10.8.2. Alle Klassen generieren .............................................. 127
10.8.3. Gesamtes Projekt generieren... (Noch zu beschreiben) .. 128
10.8.4. Einstellungen zur Codegenerierung im Projekt... (Noch zu beschreiben) ................................................................. 128
10.9. Das Menü Kritiken .......................................................... 128
10.9.1. Kritiken ein-/ausschalten ............................................. 128
10.9.2. Design-Wichtungen... .................................................. 128
10.9.3. Design Ziele... ............................................................ 130
10.9.4. Kritiken anzeigen... ..................................................... 131
10.10. Das Menü Werkzeuge ..................................................... 133
11. Der Explorer .................................................................... 137
11.1. Einleitung ....................................................................... 137
11.2. Das Verhalten der Maus im Explorer ............................... 137
11.2.1. Taste 1-Klick .............................................................. 138
11.2.2. Taste 1-Doppelklick .................................................... 138
11.2.3. Taste 1-Bewegung ...................................................... 138
11.2.4. Taste 2-Aktionen ....................................................... 139
11.2.5. Taste 2-Doppelklick .................................................... 139
11.3. Verhalten der Tastatur im Explorer ................................... 139
11.4. Auswahl der Perspektiven .............................................. 139
11.5. Perspektiven konfigurieren ............................................. 140
11.5.1. Der Dialog Perspektiven konfigurieren ....................... 140
11.6. Das kontextsensitive Menü ............................................. 142
11.6.1. Erstelle neues ........................................................... 142
11.6.2. Erstelle Modellelement .............................................. 142
11.6.3. Kopiere das Diagramm als Bild in die Zwischenablage .. 143
11.6.4. Zum Diagramm hinzufügen ....................................... 143
13.7.1. Der Bedingungs-Editor .......................................................... 185
13.8. Das Register Stereotypen .......................................................... 186
13.9. Das Register Eigenschaftswerte ................................................ 187
13.10. Das Register Checkliste ........................................................... 188

14. Der Bereich Zu-Bearbeiten ......................................................... 189
14.1. Einleitung .................................................................................. 189
14.2. Das Verhalten der Maus im Bereich Zu-Bearbeiten ...................... 189
  14.2.1. Taste 1-Klick ..................................................................... 190
  14.2.2. Taste 1-Doppelklick ........................................................... 190
  14.2.3. Taste 2-Aktionen ............................................................... 190
  14.2.4. Taste 2-Doppelklick ........................................................... 190
14.3. Auswahl der Darstellung ........................................................... 190
14.4. Element-Zähler ........................................................................ 191

15. Die Kritiken .................................................................................. 192
15.1. Einleitung .................................................................................. 192
  15.1.1. Terminologie ..................................................................... 192
  15.1.2. Design-Mangel ................................................................. 192
15.2. Unkategorisiert ......................................................................... 192
15.3. Klassenauswahl ........................................................................ 192
  15.3.1. Datentyp verbergen ........................................................... 193
  15.3.2. Verringere die Anzahl der Klassen im Namensraum <Namensraum> 193
  15.3.3. Diagramm aufräumen ........................................................ 193
15.4. Benennung ................................................................................ 193
  15.4.1. Assoziations-Namenskonflikt auflösen ................................. 193
  15.4.2. Überarbeite die Attributnamen, um einen Konflikt zu vermeiden 194
  15.4.3. Ändere Namen oder Signaturen in einem Modellelement ......... 194
  15.4.4. Doppelter End- (Rollen-) Namen in einer Assoziation .......... 194
  15.4.5. Rollenname steht im Konflikt mit einem Element ..................... 195
  15.4.6. Einen Namen auswählen (Klassen und Schnittstellen) ............. 195
  15.4.7. Namenskonflikt in einem Namensraum ............................... 195
  15.4.8. Wählen Sie einen eindeutigen Namen für ein Modellelement aus (Klassen und Schnittstellen) .......................................................... 195
  15.4.9. Wählen Sie einen Namen aus (Attribute) ............................ 195
  15.4.10. Wählen Sie einen Namen aus (Operationen) ....................... 195
  15.4.11. Wählen Sie einen Namen aus (Zustände) ............................ 195
  15.4.12. Wählen Sie einen eindeutigen Namen für ein (zustandsbehaftetes) Modellelement aus. .......................................................... 195
  15.4.13. Ändern Sie den Namen, um eine Konfusion zu verhindern ....... 195
  15.4.14. Wählen Sie einen gültigen Namen aus ............................... 196
  15.4.15. Ändern Sie den Namen des Modellelementes in ein nicht-reserviertes Wort ............................................................... 196
  15.4.16. Wählen Sie einen besseren Namen für die Operation aus .......................................................... 196
  15.4.17. Wählen Sie einen besseren Attributnamen aus ....................... 196
  15.4.18. Klassenname groß schreiben ........................................... 196
  15.4.19. Paketname überarbeiten .................................................... 197

15.5. Speicher ................................................................................... 197
  15.5.1. Überarbeiten Sie die Attributnamen, um einen Konflikt zu vermeiden 197
  15.5.2. Fügen Sie Instanzvariablen zu einer Klasse hinzu .................... 197
  15.5.3. Fügen Sie der Klasse einen Konstruktor hinzu ....................... 197
  15.5.4. Reduzieren Sie die Zahl der Attribute in der Klasse ................ 198

15.6. Geplante Erweiterungen ............................................................ 198
  15.6.1. Operationen in Schnittstellen müssen public sein ................. 198
  15.6.2. Schnittstellen dürfen nur Operationen haben ........................ 198
  15.6.3. Entferne die Referenz auf die spezifische Subklasse ........................ 198

15.7. Zustandsautomaten ................................................................... 199
  15.7.1. Reduzieren Sie die Anzahl der Transitionen im <Zustand> ........ 199
  15.7.2. Reduzieren Sie die Anzahl der Zustände im Automaten <Automat> 199
  15.7.3. Fügen Sie dem <Zustand> Transitionen hinzu ........................... 199
15.7.4. Fügen Sie dem Modellelement <Modellelement> ankommende Transitionen hinzu ................................................................. 199
15.7.5. Fügen Sie dem Modellelement <Modellelement> abgehende Transitionen hinzu ............................................................... 200
15.7.6. Entfernen Sie den zusätzlichen Initialzustand ......................................................... 200
15.7.7. Fügen Sie einen initialen Zustand ein ............................................................. 200
15.7.8. Fügen Sie der Transition ein Signal oder einen Wächter hinzu ..................... 200
15.7.9. Ändere Vereinigungs-Transitionen ...................................................... 200
15.7.10. Andere Gabelungs-Transitionen ..................................................... 200
15.7.11. Fügen Sie Entscheidungs-/Kreuzungstransitionen hinzu ......................... 200
15.7.12. Fügen Sie der Transition einen Wächter hinzu ........................................... 200
15.7.13. Das Diagramm aufräumen ........................................................................ 200
15.7.14. Eine Kante sichtbarer machen ....................................................................... 201
15.7.15. Zusammengesetztes Assoziationsende mit der Kardinalität >1 ........ 201
15.8. Designmuster .................................................................................. 201
15.8.1. Ziehen Sie Nutzung des Singleton-Musters für eine <class> in Betracht. ................................................................. 201
15.8.2. Singleton Stereotyp-Verletzung in <Klasse> ........................................ 202
15.8.3. Knoten haben normalerweise keine Hülle .............................................. 202
15.8.4. Knoteninstanzen haben normalerweise keine Hülle ............................. 202
15.8.5. Komponenten befinden sich normalerweise innerhalb von Knoten .... 202
15.8.6. Komponenteninstanzen befinden sich normalerweise innerhalb von Knoten ........................................................................................................... 202
15.8.7. Klassen befinden sich normalerweise innerhalb von Komponenten .. 203
15.8.8. Schnittstellen befinden sich normalerweise innerhalb von Komponenten ........................................................................................................... 203
15.8.9. Objekte befinden sich normalerweise innerhalb von Komponenten . 203
15.8.10. Verknüpfungsenden haben nicht die gleiche Ebene .............................. 203
15.8.11. Klassifizierung einstellen (Verteilungsdiagramm) ...................................... 203
15.8.12. Return-Aktionen werden vermisst .................................................. 203
15.8.13. Vermisse Aufruf(Sende)-Aktion .................................................. 203
15.8.15. Klassifizierung einstellen (Sequenzdiagramm) ........................................ 204
15.8.16. Falsche Position dieses Auslöseimpulses .......................................... 204
15.9. Beziehungen ................................................................................... 204
15.9.1. Zirkuläre Assoziation ........................................................................... 204
15.9.2. <Assoziation> navigierbar machen .................................................. 204
15.9.3. Entferne die Navigation von der Schnittstelle via <Assoziation> .... 204
15.9.4. Fügen Sie dem <Modellelement> eine Assoziation hinzu ................. 205
15.9.5. Entfern Sie die Referenz auf die spezifische Subklasse ....................... 205
15.9.6. Reduzieren Sie die Assoziationen des <Modellelementes> ............... 205
15.9.7. Kante sichtbarer machen ....................................................................... 205
15.10. Instanzen bilden .............................................................................. 205
15.11. Modularität ................................................................................... 205
15.11.1. Der Klassifizierer befindet sich nicht im Namensraum seiner Assoziation. ................................................................................... 205
15.11.2. Fügen Sie Elemente zum Paket <Paket> hinzu. .................................. 206
15.12. Erwartete Verwendung ........................................................................ 206
15.12.1. Diagramm aufräumen ........................................................................ 206
15.13. Methoden ....................................................................................... 206
15.13.1. Ändere Namen oder Signaturen im <Modellelement> ...................... 207
15.13.2. Die Klasse muß abstrakt sein .......................................................... 207
15.13.3. Fügen Sie der <Klasse> Operationen hinzu ..................................... 207
15.13.4. Reduzieren Sie die Anzahl der Operationen im <Modellelement> .... 207
15.14.1. Ändern Sie die Mehrfachvererbung in Schnittstellen ...................... 207
15.15. Stereotypen .................................................................................... 207
15.16. Vererbung ..................................................................................... 207
15.16.1. Überprüfen Sie die Attributenamen, um einen Konflikt zu vermeiden. 208
15.16.2. Entfernen Sie die zirkuläre Vererbung der Klasse <Klasse> …… 208
15.16.3. Die Klasse muß abstrakt sein .............................................. 208
15.16.4. Entfernen Sie das Schlüsselwort final oder entfernen Sie Subklassen 208
15.16.5. Illegale Generalisierung ..................................................... 208
15.16.6. Entferne unnötige Realisierungen aus der Klasse <Klasse> …… 208
15.16.7. Definiere eine konkrete (Sub-)Klasse ................................. 209
15.16.8. Definieren Sie eine Klasse, um die Schnittstelle <Schnittstelle> zu implementieren .............................. 209
15.16.9. Andere Mehrfachvererbung in Schnittstellen ........................ 209
15.16.10. Machen Sie die Kante sichtbarer ........................................ 209

15.17. Containment ................................................................. 209
15.17.1. Entferne zirkuläre Komposition ...................................... 209
15.17.2. Duplicieren Sie den Parameternamen .............................. 209
15.17.3. Zwei Aggregatenden (Rollen) in binärer Assoziation .......... 209
15.17.4. Aggregatende (Rolle) in 3-Wege (oder mehr) Assoziation 210
15.17.5. Verbergen Sie den Datentyp ........................................... 210

3. Modellreferenz .......................................................... 211
16. Modellreferenz auf höchster Ebene ........................................ 212
16.1. Einleitung ........................................................................... 212
16.2. Das Modell ......................................................................... 212
16.2.1. Register Modelldetails ..................................................... 212
16.2.2. Symbolleiste Modelleigenschaften ................................... 213
16.2.3. Eigenschaftsfelder des Modelles ....................................... 213
16.3. Datatyp .............................................................................. 215
16.3.1. Register Datentypdetails .................................................. 216
16.3.2. Symbolleiste Datentypigenschaften ................................. 216
16.3.3. Eigenschaftsfelder für den Datentyp .................................. 217
16.4. Enumeration (Aufzählung) ................................................... 219
16.4.1. Die Detail-Register Enumeration ..................................... 219
16.4.2. Eigenschaftssymbolleiste Enumeration ............................. 220
16.4.3. Eigenschaftsfelder für Enumerationen ............................... 221
16.5. Enumeration Literal .......................................................... 222
16.6. Stereotyp .......................................................................... 222
16.6.1. Detail-Register Stereotyp .................................................. 222
16.6.2. Eigenschaftssymbolleiste Stereotyp ................................. 223
16.6.3. Eigenschaftsfelder für Stereotypen .................................... 224
16.7. Eigenschaftsdefinition ......................................................... 225
16.8. Diagramm ........................................................................ 225
16.8.1. Detail-Register Diagramm ............................................... 226
16.8.2. Eigenschaftssymbolleiste Diagramm ................................. 227
16.8.3. Eigenschaftsfelder für Diagramme .................................... 227

17. Referenz der Modellelemente für Anwendungsfalldiagramme 228
17.1. Einleitung ........................................................................... 228
17.1.1. ArgoUML-Einschränkungen, welche die Anwendungsfalldiagramme betreffen ........................... 228
17.2. Akteur .............................................................................. 229
17.2.1. Detail-Register Akteur ..................................................... 229
17.2.2. Eigenschaftssymbolleiste Akteur ...................................... 230
17.2.3. Eigenschaftsfelder für einen Akteur ................................. 231
17.3. Anwendungsfall ............................................................... 232
17.3.1. Detail-Register Anwendungsfall ...................................... 232
17.3.2. Eigenschaftssymbolleiste Anwendungsfall ....................... 233
17.3.3. Eigenschaftsfelder für einen Anwendungsfall .................... 234
17.4. Erweiterungspunkt ........................................................... 236
17.4.1. Detail-Register Erweiterungspunkt ................................. 237
17.4.2. Eigenschaftssymbolleiste Erweiterungspunkt ................. 237
17.4.3. Eigenschaftsfelder des Erweiterungpunktes ..................... 238
<table>
<thead>
<tr>
<th>Kapitel</th>
<th>Seitenzahl</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Ergänzendes Material für die Fallstudie</td>
<td>383</td>
</tr>
<tr>
<td>A.1. Einleitung</td>
<td>383</td>
</tr>
<tr>
<td>A.2. Anforderungsdokumente (Noch zu beschreiben)</td>
<td>383</td>
</tr>
<tr>
<td>A.2.1. Visionsdokument (Noch zu beschreiben)</td>
<td>383</td>
</tr>
<tr>
<td>A.2.2. Anwendungsfälle-Spezifikationen (Noch zu beschreiben)</td>
<td>383</td>
</tr>
<tr>
<td>A.2.3. Ergänzende Anforderungsspezifikation (Noch zu beschreiben)</td>
<td>383</td>
</tr>
<tr>
<td>B. UML-Ressourcen</td>
<td>384</td>
</tr>
<tr>
<td>B.1. Die UML-Spezifikationen (Noch zu beschreiben)</td>
<td>384</td>
</tr>
<tr>
<td>B.2. UML-Papiere (Noch zu beschreiben)</td>
<td>384</td>
</tr>
<tr>
<td>B.2.1. UML-Aktionsspezifikationen (Noch zu beschreiben)</td>
<td>384</td>
</tr>
<tr>
<td>B.3. UML-Webseiten (Noch zu beschreiben)</td>
<td>384</td>
</tr>
<tr>
<td>C. UML-konforme CASE Tools</td>
<td>385</td>
</tr>
<tr>
<td>C.1. Andere Open Source Projekte (Noch zu beschreiben)</td>
<td>385</td>
</tr>
<tr>
<td>C.2. Kommerzielle Werkzeuge (Noch zu beschreiben)</td>
<td>385</td>
</tr>
<tr>
<td>D. Das C++ Modul</td>
<td>386</td>
</tr>
<tr>
<td>D.1. Modellieren für C++</td>
<td>386</td>
</tr>
<tr>
<td>D.1.1. Eigenschaftswerte für eine Klasse</td>
<td>387</td>
</tr>
<tr>
<td>D.1.2. Eigenschaftswerte für Attribute</td>
<td>388</td>
</tr>
<tr>
<td>D.1.3. Parameter</td>
<td>388</td>
</tr>
<tr>
<td>D.1.4. Generalisierung</td>
<td>389</td>
</tr>
<tr>
<td>D.1.5. Realisierung</td>
<td>389</td>
</tr>
<tr>
<td>D.1.6. Geschützte Abschnitte</td>
<td>389</td>
</tr>
<tr>
<td>E. Beschränkungen und Mängel</td>
<td>391</td>
</tr>
<tr>
<td>E.1. Größe der Diagramm-Leinwand</td>
<td>391</td>
</tr>
<tr>
<td>E.2. Fehlende Funktionen</td>
<td>391</td>
</tr>
</tbody>
</table>
F. Open Publication Lizenz ....................................................................................... 392
  F.1. Einleitung ................................................................................................ 392
  F.2. Copyright ................................................................................................. 392
  F.3. Bezugsbereich der Lizenz ........................................................................ 392
  F.4. Anforderungen an modifizierte Arbeiten ................................................... 393
  F.5. Empfehlungen für die gute Praxis .............................................................. 393
G. Die CRC-Karten Methode .................................................................................... 394
  G.1. Die Karte ................................................................................................. 394
  G.2. Die Gruppe .............................................................................................. 395
  G.3. Die Sitzung .............................................................................................. 395
  G.4. Der Prozess .............................................................................................. 395
Stichwortverzeichnis ................................................................................................. 396
Vorwort

Softwareentwurf ist eine geistig herausfordernde Tätigkeit. Designer müssen Ihre Entwürfe manuell eingeben. Aber die grundlegende Schwierigkeit ist, entscheidungsorientiert statt dateneingabebasiert zu arbeiten. Wenn Designer ihre entscheidungsorientierten Fähigkeiten verbesserten, würden bessere Entwürfe dabei herauskommen.


ArgoUML ist eine domänenorientierte Designumgebung mit kognitiver Unterstützung des objektorientierten Entwurfes. ArgoUML enthält im Wesentlichen die gleichen Automatismen wie kommerzielle CASE-Tools. Sein Fokus liegt aber auf Funktionen, die die kognitiven Bedürfnisse von Designern befriedigen. Diese kognitiven Bedürfnisse werden durch 3 kognitive Theorien beschrieben:

1. Reflektion-während-Aktion;
2. Opportunistisches Design; und
3. Verständnis und Problemlösung.

ArgoUML basiert direkt auf der UML 1.4-Spezifikation. Das zentrale Modell-Repository ist eine Implementierung der Java Metadaten Schnittstelle (JMI=Java Metadata Interface), welche MOF direkt unterstützt und die maschinenlesbare Version der UML 1.4-Spezifikation der OMG verwendet.


Dieses Handbuch ist die kumulative Arbeit mehrerer Personen und entwickelte sich über mehrere Jahre.
Im Zusammenhang mit der ArgoUML-Release 0.10 schrieb Jeremy Bennett eine Menge neues Material, was dem in früheren Versionen von Alejandro Ramirez, Philippe Vanpeperstraete und Andreas Rueckert geschriebenen hinzugefügt wurde. Er fügte auch einige Dinge aus anderen Dokumenten ein, namentlich aus dem Programmierhandbuch von Markus Klink und Linus Tolke, der Kurzanleitung von Kunle Odutola, sowie die FAQ von Denny Daniels. Im Zusammenhang mit der Version 0.14 wurden Änderungen durch Linus Tolke und Michiel van der Wulp vorgenommen. Diese Änderungen passten das Handbuch an die neuen Funktionen und das neue Erscheinungsbild von ArgoUML, Version 1.4 an und führten einen Index ein. Es sind zu viele Anwender und Entwickler, die diese Arbeit durch Ihre wertvollen Hinweise, wie Review-Kommentare oder Beobachtungen während des Lesens und der Anwendung dieses Handbuchs unterstützt haben, um sie alle namentlich benennen zu können.


Das Handbuch ist in DocBook/XML geschrieben und sowohl als HTML als auch als PDF verfügbar.

Das ArgoUML-Projekt begrüßt all diejenigen, die tierig eingebunden werden wollen. Mehr finden Sie unter der Projekt-Webseite [http://argouml.tigris.org/].

Teilen Sie uns bitte mit, was Sie über das Anwenderhandbuch denken! Ihre Kommentare helfen uns, es weiter zu verbessern. Siehe Abschnitt 1.3.3, „Anwender-Feedback“. 
Kapitel 1. Einleitung

1.1. Die Anfänge und ein Überblick über ArgoUML

1.1.1. Objektorientierte Analyse und Design

Im letzten Jahrzehnt wurde die objektorientierte Analyse und Design (OOA&D) zu dem dominanten Softwareparadigma. Damit einhergehend fand ein Umdenken in allen, an dem Softwarelebenszyklus beteiligten Prozessen statt.


Es ist die Entwicklung von Tools während der 90'er Jahre, die die objektorientierte Analyse und das objektorientierte Design vorantrieben. Gekoppelt mit der Fähigkeit, Systeme auf einer sehr hohen Abstraktionsebene zu entwerfen, ermöglichte der toolbasierte OOA&D-Ansatz die Implementierung sehr viel komplexerer Systeme als es bisher möglich war.

Das letzte Element, das die OOA&D vorantrieb, war seine Fähigkeit, grafische Anwenderschnittstellen zu modellieren. Die Popularität der objektbasierten und objektorientierten grafischen Sprachen, wie Visual Basic und Java reflektieren die Effizienz dieses Ansatzes.

1.1.2. Die Entwicklung von ArgoUML


1. Es ist frei.

2. ArgoUML zeichnet sich durch die Erforschung kognitiver Psychologie aus, die ungewöhnliche Funktionen zur Folge hatten, die die Produktivität durch die Unterstützung der kognitiven Bedürfnisse objektorientiert arbeitender Softwaredesigner und -Architekten erhöhten.

4. ArgoUML ist eine 100%ige Java-Anwendung. Dadurch kann ArgoUML auf allen Plattformen laufen, auf denen die Laufzeitumgebung der Java-Plattform verfügbar ist.


1.1.3. Mehr über das ArgoUML-Projekt

1.1.3.1. Wie wurde ArgoUML entwickelt


Die Entwickler-Mails sind der Ort, in dem die Diskussionen über die aktuellen Aktivitäten stattfinden und Entwickler die Richtung diskutieren, die das Projekt entwickelt werden soll. Obwohl manchmal sehr kontrovers, blieben diese Diskussionen immer nett und freundlich (keine "flammenden Auseinandersetzungen" oder ähnliches), so dass neue Entwickler nicht zögern und einfach daran teilnehmen sollten. Sie werden immer willkommen sein.

Wenn Sie lernen wollen, wie das Projekt arbeitet und wie Sie sich einbringen können, rufen Sie die ArgoUML Webseite "Developer Zone" [http://argouml.tigris.org/dev.html] auf und lesen sich die Dokumentation durch. Das Programmierhandbuch (Cookbook) wurde speziell für diesen Zweck geschrieben.

1.1.3.2. Mehr über die Infrastruktur


1.2. Der Bezugsbereich dieses Anwenderhandbuches
1.2.1. Der Leserkreis

Die aktuelle Fassung dieses Dokumentes ist für erfahrende Anwender der UML in der OOA&D (vielleicht auch mit anderen Tools) gedacht, die zu ArgoUML wechseln möchten.

Künftige Fassungen werden Designer unterstützen, die die OOA&D kennen und die UML-Notation in Ihren Entwicklungsprozess einbinden möchten.

Ein langfristiges Ziel ist es, diejenigen zu unterstützen,

i. die das Design erlernen und mit einem OOA&D-Prozess beginnen wollen, der die UML-Notation unterstützt und

ii. Personen, die an einem modularisierten Design mit einer GUI interessiert sind.

1.2.2. Bezugsbereich

Die Absicht dieses Dokument ist es, eine verständliche Einführung zu geben, die die Designer in Lage versetzen, ArgoUML in vollem Umfang zu nutzen. Es ist in zwei Teile untergliedert.

• Ein Übungshandbuch, das die Arbeitsweise mit ArgoUML zeigt.

• Ein vollständiges Referenzhandbuch, das alles beschreibt, was Sie mit ArgoUML tun können.

Die Version 0.22 des Dokumentes bezieht sich auf den zweiten Teil.

In dieser Anleitung gibt es Einiges, das Sie nicht finden werden, weil es an anderen Stellen beschrieben ist.

• Beschreibungen, wie ArgoUML intern arbeitet.

• Wie ArgoUML mit neuen Eigenschaften und Funktionen erweitert wird.

• Eine Anleitung zur Fehlerbehebung.

• Eine zusammenfassende Kurzreferenz für ArgoUML.


1.3. Überblick über das Anwenderhandbuch

1.3.1. Die Struktur des Übungshandbuches

Das Kapitel 2, Einleitung enthält einen Überblick über UML-basierte OOA&D, einschliesslich einer Anleitung, wie man ArgoUML installiert und startet.

Kapitel 4, Erfassen der Anforderungen bis Kapitel 7, Codegenerierung, Reverse Engineering und Round Trip Engineering geht dann durch alle Teile des Designprozesses. Von der ersten Anforderung bis zum
abschliessenden Projekt-Build und der Verteilung.


1.3.2. Die Struktur des Referenzhandbuches


1.3.3. Anwender-Feedback


1.4. Annahmen


Die Fallstudie ist beschrieben, aber noch nicht vollständig in dieser Übungsanleitung realisiert. Dies wird erst in zukünftigen Versionen dieses Handbuches der Fall sein.
Teil 1. Übungsanleitung
Kapitel 2. Einleitung

Diese Übungsanleitung führt Sie in Modellierung eines Systems mit Hilfe von ArgoUML ein.


Jetzt sollten Sie ein Verzeichnis erzeugen, das Ihr Projekt aufnimmt. Benennen Sie das Verzeichnis so, dass es zum Rest Ihres Dateisystems passt. Die Inhalte und die Unterverzeichnisse sollten Sie so bezeichnen, wie nachfolgend beschrieben.


Wir werden natürlich keinen physikalisch existierenden Geldautomaten erstellen. Das Produkt, das wir als Fallstudie erzeugen werden, ist ein Geldautomatensimulator, der zum Testen und Entwerfen eines physikalisch vorhandenen Geldautomaten verwendet wird.

Wie Ihre Firma die Arbeit in Projekten organisiert, ist gewöhnlich mehr durch politische, als durch andere Einflüsse bestimmt. Aus diesem Grund steht dies nicht im Mittelpunkt dieses Dokumentes. Wir werden zeigen, wie Sie Ihr Projekt nach dessen Definition selbst strukturieren können.
Kapitel 3. UML basierte OOA&D

In diesem Kapitel sehen Sie, wie UML als Notation innerhalb der OOA&D verwendet wird.

3.1. Hintergrundinformationen zu UML


1. Das Erscheinen der OO-Programmiersprachen, wie SmallTalk und teilweise C++. C++ war eine pragmatische, von C abgeleitete OO-Sprache, die wegen ihrer Verknüpfung mit UNIX sehr weit verbreitet war.


3. Eine Vielzahl sehr bekannter Projektfehlschläge, die nahelegten, dass der aktuelle Ansatz nicht mehr tragfähig war.


Während der frühen 90'er wurde klar, dass diese Ansätze viele gute, oft sehr einfache Ideen enthielten. Ein Haupt-Stolperstein waren jedoch die unterschiedlichen Notationen, was bedeutete, dass die Ingenieure mehr mit einer bestimmten OOA&D-Methode als mit dem generellen Ansatz vertraut waren.


3.2. UML basierter Prozess für OOA&D

Sie müssen verstehen, dass UML eine Notation für die OOA&D ist. Sie schreibt keinen bestimmten Prozess vor. Welcher Prozess auch immer angewendet wird, ein System wird in mehreren Phasen konstruiert.

1. Erfassen der Anforderungen. In dieser Phase identifizieren wir die Anforderungen an das System. Dabei verwenden wir die Sprache des Problembereiches. Mit anderen Worten, wir beschreiben das Problem mit den Begrifflichkeiten des „Kunden“.


4. Build-Phase. Wir nehmen das aktuelle Design und setzen es in eine reale Programmiersprache um. Das schliesst nicht nur die Programmierung ein, sondern auch das Testen, ob das Programm den Anforderungen (Verifikation) entspricht, testen, ob das Programm das aktuelle Kundenproblem löst (Validierung) und das Schreiben der gesamten Anwenderdokumentation.

3.2.1. Prozesstypen

In diesem Abschnitt sehen wir uns die beiden Hauptprozesstypen an, die im Software-Engineering verwendet werden. Es gibt noch weitere, diese werden aber weniger häufig eingesetzt.

In den letzten Jahren gab es auch eine Bewegung, den für die Entwicklung von Software notwendigen Aufwand zu reduzieren. Dies führte in der Entwicklung zu einer Anzahl von kleineren Prozessvarianten (oft als agile computing oder extreme programming bekannt), die auf sehr kleine Ingenieurteams zugeschnitten waren.

3.2.1.1. Der Wasserfall-Prozess

In diesem Prozess wird jeder Prozessschritt - Erfassen der Anforderungen, Analyse, Design und Build (Code und Test) abgeschlossen, bevor der nächste beginnt. Dies ist in Abbildung 3.1, „Der Wasserfall-Prozess“ illustriert.

**Abbildung 3.1. Der Wasserfall-Prozess**
Dies ist ein sehr zufriedenstellender Prozess, in dem Anforderungen gut entworfen und keine Änderungen erwartet werden - zum Beispiel die Automatisierung eines bewährten, manuellen Systems.

Der Schwachpunkt dieses Ansatzes zeigt sich bei weniger gut definierten Problemen. Offene Punkte in den Anforderungen bleiben bis zur Phase Analyse und Design, oder auch bis zur Codephase ungeklärt. Dies erfordert dann ein Zurückgehen, um diese Arbeit nachzuholen.

Der schlechteste Aspekt davon ist, dass ausführbarer Code erst zum Ende des Projektes verfügbar wird. Sehr häufig ist es so, dass erst zu diesem Zeitpunkt Probleme mit den ursprünglichen Anforderungen (zum Beispiel mit der Anwenderschnittstelle) in Erscheinung treten.

Dies ist besonders schlimm, weil jeder folgende Schritt mehr Aufwand als der vorhergehende erfordert, so dass die Kosten spät entdeckter Probleme extrem teuer sind. Dies wird durch die Pyramide in Abbildung 3.2, „Zu erwartender Aufwand je Schritt des Wasserfallmodelles“ dargestellt.

Abbildung 3.2. Zu erwartender Aufwand je Schritt des Wasserfallmodelles

Der Wasserfallprozess ist wahrscheinlich der dominante Designprozess. Durch seine Beschränkungen wird er jedoch mehr und mehr durch iterative Prozesse ersetzt. Insbesondere bei Projekten, bei denen die Anforderungen schlecht definiert sind.

3.2.1.2. Iterative Entwicklungsprozesse

In den letzten Jahren wurde ein neuer Ansatz verwendet, dessen Ziel es war, einen kleinen Teil des Codes fertigzustellen und schnellstmöglichst auszuführen, um Probleme im Entwicklungszyklus sehr schnell aufzudecken.

Dieser Prozess wird dann für alle weiteren Anforderungen wiederholt, um ein Produkt mit wachsender Funktionalität zu konstruieren. Erneutes Feedback kann wieder in die Anforderungen einfließen.


**Abbildung 3.3. Zu erwartender Aufwand je Schritt des iterativen Prozesses**


### 3.2.1.2.1. Der Rational Unified Process


Dieser Prozess berücksichtigt, dass unsere Pyramidensicht der Wasserfallebenen nicht realistisch ist. In der Praxis neigen frühe Iterationen dazu, zu gross zu werden (sie müssen zu Beginn eine beträchtliche Menge definieren), während späterer Iterationen sehr viel Aufwand in der Design- und Buildphase benötigen.


- In der **Anfangsphase** neigen Iterationen dazu, mit Blick auf die Gesamtanforderung/ Analyse zu gross zu werden, während jede Build-Aktivität bezüglich der Emulation des Desgins durch das CASE-Tool eingeschränkt sein kann.
- In der **Ausarbeitungsphase** werden die Iterationen dazu benutzt, die Spezifikation der Anforderungen
zu vervollständigen. Ihr Schwerpunkt liegt auf Analyse und Design und dem wahrscheinlich ersten real erzeugten Code.

- In der *Konstruktionsphase* sind die Anforderungen und Analyse mehr oder weniger vollständig, der Aufwand liegt häufig im Design und in der Buildphase.

- In der abschliessenden *Deploymentphase* finden Iterationen weitestgehend in den Buildaktivitäten und teilweise beim Test der Software statt.

**Anmerkung**


Wir wollen in diesem Handbuch einen iterativen Prozess verwenden, der lose auf dem RUP basiert.

### 3.2.1.2.2. Iterationsgröße


### 3.2.1.3. Rekursive Entwicklungsprozesse

Sehr wenige Softwaresysteme sind als monolitische Werkzeuge geplant. Sie werden in Subsysteme, Module usw. unterteilt.

Die Softwareprozesse sind die gleichen. Mit frühen Abschnitten in denen der Prozess eine Top-Level-Struktur definiert und die wiederholte Anwendung des Prozesses auf Teile der Struktur, um jedes grössere Detail zu definieren.

Zum Beispiel könnte das grundlegende Design eines Telefonsystemes Objekte identifizieren um

i. Telefonleitungen zu belegen,
ii. Anrufe zu verarbeiten,

iii. das System zu verwalten und,

iv. den Kunden abzurechnen.

Der Softwareprozess kann dann auf jedes dieser vier Komponenten erneut angewendet werden, um deren Design zu identifizieren.


3.2.2. Der Entwicklungsprozess für dieses Übungshandbuch

In dieser Übungsanleitung wollen wir einen reduzierten, iterativen Prozess mit rekursiver Entwicklung verwenden, der mit dem RUP lose gekoppelt ist. Die Fallstudie bringt uns durch die erste Iteration, am Ende des Abschnittes Übungsanleitung werden wir sehen, wie das Projekt vollständig entwickelt wird.

Innerhalb der ersten Iteration werden wir jede Aktivität wie das Erfassen der Anforderungen, die Analyse, das Design und den Build angehen. Nicht alle Teile des Prozesses basieren auf UML oder ArgoUML. Wir werden sehen, welches andere Material hierfür benötigt wird.

Innerhalb dieses Prozesses werden wir die Möglichkeit haben, die verschiedenen UML-Diagramme in ihrer Anwendung zu sehen. Der vollständige Umfang der UML-Diagramme und wie sie angewendet werden ist im Referenzhandbuch beschrieben (siehe Abschnitt 16.8, „Diagramm“).

3.2.2.1. Erfassen der Anforderungen


3.2.2.2. Analyse


Wir werden in das UML Sequenzdiagramm und das Zustandsdiagramm einführen, um die Anforderungen für das Gesamtverhalten des Systems zu erfassen.

Abschliessend nehmen wir die Anwendungsfälle aus dem Abschnitt Anforderungsaufnahme und wandeln diese in die Sprache des Lösungsbereiches um. Dies wird die UML-Ideen von Stereotypen und Realisierung illustrieren.

3.2.2.3. Design

Wir verwenden das UML-Paketdiagramm, um die Komponenten des Projektes zu organisieren. Wir besuchen dann erneut das Klassendiagramm, das Sequenzdiagramm und das Zustandsdiagramm, um zu zeigen, wie diese rekursiv verwendet werden können, um die vollständige Lösung zu designen.
Während dieses Teils des Prozesses müssen wir unsere Systemarchitektur entwickeln, um zu definieren, wie alle Komponenten zusammenpassen und miteinander zusammenarbeiten.


Zum Schluß wollen wir das UML-Deploymentdiagramm verwenden, um zu spezifizieren, wie das System aktuell realisiert werden soll.

3.2.2.4. Build


Wir wollen uns auch ansehen, wie das UML-Anwendungsfalldiagramm und die Anwendungsfall-Spezifikation unschätzbare Werkzeuge für ein Testprogramm sind.

3.3. Warum ist ArgoUML anders

In der Einleitung listeten wir vier Dinge auf, die ArgoUML anders machen:

i. es macht Gebrauch von den Ideen der kognitiven Psychologie,
ii. es basiert auf offenen Standards;
iii. es ist 100% reines Java; und
iv. es ist ein Open-Source-Projekt.

3.3.1. Kognitive Psychologie

3.3.1.1. Theorie

ArgoUML ist teilweise durch die drei Theorien der kognitiven Psychologie inspiriert:

1. Reflektion-während-Aktion,
2. Opportunistisches Design und
3. Verständnis und Problemlösung.

• **Reflektion-während-Aktion**

Diese Theorie unterstellt, dass sich die Designer eines komplexen Systems das vollständig ausgeformte Design nicht vorstellen können. Stattdessen müssen Sie ein Teil-Design entwerfen, evaluiieren, es reflektieren und überarbeiten, bis Sie soweit sind, es weiter zu erweitern.

Da sich die Entwickler sehr intensiv mit dem Design beschäftigen, verbessert sich auch ihr mentales Modell der Problemsituation. Und dies führt zur Verbesserung Ihres Designs.
• Opportunistisches Design

Eine Theorie innerhalb der kognitiven Psychologie besagt, dass obwohl Designer ihre Arbeit in einer geordneten, hierarchischen Art und Weise beschreiben, sie in der Realität aufeinanderfolgende Aktivitäten auf der Basis kognitiver Kosten wählen.

Einfach ausgedrückt, Designer folgen nicht ihrer selbst geplanten Reihenfolge, sondern wählen Schritte, die mental weniger aufwändige Alternativen darstellen.

• Verständlichkeit und Problemlösung


Diese Theorie besagt, dass Programmierer Vorteile haben von:


2. Vertraute Aspekte eines Situationsmodelles, welche die Fähigkeiten des Designers verbessern, Lösungen zu formulieren.

3.3.1.2. Praktische Anwendung in ArgoUML

ArgoUML implementiert diese Theorien durch eine bestimmte Anzahl von Techniken.


3.3.2. Offene Standards


Der Hauptvorteil am Festhalten an offenen Standards ist, dass es die einfache Zusammenarbeit zwischen Anwendungen erlaubt und die Möglichkeit eröffnet, von einer Anwendung zu einer anderen wechseln, sofern dies notwendig ist.

3.3.2.1. XML Metadaten-Austausch (Metadata Interchange XMI)

XML Metadata Interchange (XMI) ist der Standard für das Speichern von Metadaten, die ein bestimmtes UML-Modell darstellen. Im Prinzip erlaubt er es, das in ArgoUML erstellte Modell in ein anderes Tool
zu importieren.

Dies hat klare Vorteile, die es der UML erlaubt, sein Ziel zu erreichen, der Standard für die Kommunikation zwischen Designern zu sein.

Die Realität sieht nicht so gut aus. Vor UML 2.0 enthielten die XMI-Dateien keine Informationen über die grafische Darstellung der Modelle, so dass das Diagrammlayout verloren ging. ArgoUML umgeht dies, indem es die grafische Information separat vom Modell speichert. (siehe Abschnitt 3.4.3.1, „Laden und Speichern“).

### 3.3.2.2. Grafik-Formate - EPS, GIF, PGML, PNG, PS, SVG


- Precision Graphics Markup Language (PGML) ist eine XML-basierte Sprache zur Darstellung von Vektorgrafiken. Sie war ein W3C-Entwurf, der aber nicht als Empfehlung übernommen wurde. PGML und VML, andere XML-basierte Vektor- Grafik-Sprachen, wurden später miteinander verknüpft und verbessert, um daraus SVG (siehe unten) zu erzeugen.


- PostScript (PS) ist eine Seitenbeschreibungs- und Programmiersprache, die primär in elektronischen und Desktop-Publishing-Bereichen eingesetzt wird.


### 3.3.2.3. Objekt-Bedingungs-Sprache (Object Constraint Language OCL)

Die Object Constraint Language (OCL) ist eine deklarative Sprache zur Beschreibung von Regeln, die auf UML-Modelle angewendet werden. Sie wurde durch IBM entwickelt, und ist jetzt Teil des UML-Standard. Ursprünglich war OCL nur als formale Spezifikations-Sprach-Erweiterung von UML gedacht. OCL kann jetzt mit jedem Meta-Object Facility (MOF) konformen Metamodell einschliesslich UML angewendet werden. Die Object Constraint Language ist eine präzise Textsprache, die Bedingungen und Objekt-Abfrageausdrücke für jedes MOF-
oder Meta-Modell enthält, die nicht auf andere Art durch eine Diagramm-Notation ausgedrückt werden können.

### 3.3.3. 100% reines Java


Das Schreiben eines Interpreters für eine JVM ist sehr viel leichter, als das Schreiben eines Compilers und, solche Maschinen sind jetzt in fast jedem Web Browser eingebaut. Im Ergebnis können die meisten Maschinen Java, ohne dass es weiterer Arbeiten bedarf.

(Für den Fall, dass Sie sich wundern, warum nicht alle Sprachen so sind wie diese, dann deshalb, weil interpretierende Sprachen langsamer sind als übersetzende Sprachen. Mit der hohen Leistung moderner PC’s und dem Handlungszwang nach Portabilität ist diese Einschränkung für viele Applikationen trotzdem lohnenswert. Darüber hinaus können moderne Multi-Cache-Systeme bedeuten, dass interpretierende Sprachen, die dichteren Code produzierenden, aktuell nicht mehr sehr viel langsamer sind.)

Durch die Entscheidung, ArgoUML in reinem Java zu schreiben, wird ArgoUML für sehr viele Anwender mit einem minimalem Aufwand unmittelbar verfügbar.

### 3.3.4. Offener Quellcode (Open Source)

ArgoUML ist ein *Open Source*-Projekt. Das bedeutet, jeder kann eine freie Kopie des Quellcodes haben, diesen ändern und für neue Zwecke einsetzen usw.. Die einzige (Haupt-) Bedingung ist, dass Sie Ihren Code auf die gleiche Weise anderen zur Verfügung stellen. Die genaue Ausprägung, was Sie tun können und was nicht, variiert von Projekt zu Projekt, aber das Prinzip ist das gleiche.


Es ist aber nicht nur der Geist reiner Uneigennützigkeit. Hier mitzuarbeiten ist ein Weg, „nebenbei“ mehr über führende Software zu lernen. Es ist auch ein Weg, viel Aufmerksamkeit (über 1.125.000 Personen haben ArgoUML Ende 2005 heruntergeladen) zu bekommen. In Ihrem Lebenslauf ist das eine sehr gute Erfahrung und eine Menge Arbeitgeber sehen Sie!

Und es ist hervorragend für Ihr Ego!


### 3.4. ArgoUML Grundlagen

Das Ziel dieses Abschnittes ist es, Sie in die Lage zu versetzen mit ArgoUML zu beginnen. Er zeigt Ihnen, wie Sie den Code bekommen und starten können.

#### 3.4.1. Erste Schritte

##### 3.4.1.1. Systemanforderungen


### 3.4.1.2. Download-Möglichkeiten

Es gibt drei Optionen, wie Sie ArgoUML bekommen können.

4. Sie laden sich den Quellcode mit Hilfe von Subversion herunter und erzeugen sich Ihre eigene Version. Wählen Sie diese Option, wenn Sie sich ansehen wollen wie ArgoUML intern arbeitet oder wenn Sie sich als Entwickler einbringen wollen. Diese Option erfordert das vollständige JDK (siehe Abschnitt 3.4.1.1, „Systemanforderungen“).


### 3.4.1.3. ArgoUML mit Java Web Start aufrufen

Hierfür sind zwei Schritte auszuführen.


### 3.4.1.4. Das Windows-Installationsprogramm verwenden

Wenn Sie sich dazu entscheiden, ArgoUML mit Hilfe des Windows- Installationsprogrammes herunterzuladen und zu installieren, haben Sie die Wahl, ob Sie die letzte stabile Version (die zuverlässiger ist, aber nicht die letzten Eigenschaften beinhaltet) oder die aktuelle Version (die unzuverlässiger ist, aber mehr Eigenschaften beinhaltet) herunterladen. Entscheiden Sie sich auf Basis
Ihrer eigenen Situation.

Der Installationsassistent gibt Ihnen die Möglichkeit, die letzte "JRE" (Java Runtime Environment) zu installieren. Es gibt keine Notwendigkeit diese Möglichkeit auszuwählen, wenn Sie bereits Sun Java, Version 5 oder höher installiert haben.

3.4.1.5. Die binäre, ausführbare Datei herunterladen

Wenn Sie sich für das Herunterladen der binären, ausführbaren Datei entscheiden, können sie wählen, ob Sie die letzte stabile Version (die zuverlässiger ist, aber nicht alle Eigenschaften aufweist) oder die aktuelle Version (die unzuverlässiger ist, aber mehr Eigenschaften aufweist) herunterladen wollen. Entscheiden Sie dies unter Berücksichtigung Ihrer eigenen Situation.


- Unter Unix. Nutzen Sie GNU tar, um die Dateien in ein Verzeichnis Ihrer Wahl zu entpacken.

```bash
tar zxfv <file>.tar.gz
```

Wenn Sie eine ältere Version von tar haben, ist die z-Option nicht verfügbar. Benutzen Sie in diesem Fall

```bash
gunzip < file.tar.gz | tar xv -.
```

Sie sollten jetzt in dem Verzeichnis eine Reihe von .jar -Dateien und die Datei README.txt vorfinden.

3.4.1.6. Probleme beim Herunterladen und Installieren


Sie können sich in die Mailingliste auf der Projektseite argouml.tigris.org [http://argouml.tigris.org] registrieren, oder eine leere Nachricht an users@argouml.org [mailto:users@argouml.org] mit dem Betreff subscribe senden.

Sie können dann Ihr Problem an users@argouml.org [mailto:users@argouml.org] senden und abwarten, ob andere Anwender Ihnen helfen können.

Die Anwender-Mailingliste ist eine exzellente Einführung in die Live-Aktivitäten des Projektes. Wenn Sie noch stärker eingebunden werden wollen, gibt es zusätzliche Mailinglisten, welche die Entwicklung des Produktes und Fragen zu aktuellen und künftigen Releases beinhalten.

3.4.1.7. ArgoUML starten

Wie Sie ArgoUML starten, hängt davon ab, ob Sie Microsoft Windows oder eine Unix-Variante verwenden.
• Bei Windows. Rufen Sie die MSDOS-Eingabeaufforderung auf, z.B. durch Start/Ausführen mit „command“. In dem Fenster wechseln Sie in das Verzeichnis, in dem sich die ArgoUML-Dateien befinden und geben Sie

```
java -jar argouml
```


• Unter Unix. Öffnen Sie ein Shell-Fenster und geben Sie

```
java -jar argouml
```

ein.

### 3.4.1.8. Probleme beim Starten von ArgoUML

Wenn Sie ArgoUML erfolgreich heruntergeladen haben, ist es ungewöhnlich, wenn Probleme auftreten. Wenn Sie das Problem nicht lösen können, versuchen Sie es über die Anwender-Mailingliste (siehe Abschnitt 3.4.1.6, „Probleme beim Herunterladen und Installieren“).

• Falsche JRE. Ein häufiger Grund ist eine veraltete Java Runtime Umgebung (es muss die Version 5 oder höher sein).


**Abbildung 3.4. Den Einstellungs-Assistenten finden**
Abbildung 3.5. Die Sprache unter Erscheinungsbild einstellen

3.4.2. Die ArgoUML-Anwenderschnittstelle


Abbildung 3.6, „Das erste ArgoUML-Fenster“, zeigt das ArgoUML-Fenster, wie es beim ersten Aufruf angezeigt wird.

Abbildung 3.6. Das erste ArgoUML-Fenster

Ziehen Sie die vertikalen Fensterteiler vorwärts und rückwärts. Ziehen Sie die horizontalen Fensterteiler hoch und runter. Spielen Sie etwas mit den kleinen Pfeilen links oder oben in den Fensterteilern. Siehe Abschnitt 8.3, „Generelle Informationen über Fenster“.

3.4.2.1. Die Menü- und die Symbolleisten

Die Menüleiste und die Symbolleisten ermöglichen Ihnen den Zugriff auf alle Features von ArgoUML. Menü- und Werkzeugoptionen sind per Konvention ausgeblendet, wenn sie nicht verfügbar (ausgeschaltet) sind und auf Menüelemente, die eine Dialogbox aufrufen, weisen drei Punkte hin (...). Zum jetzigen Zeitpunkt sollten Sie Kapitel 9, Die Symbolleiste und Kapitel 10, Die Menüzeile lesen.


1. Datei=>Projekt speichern rückgängig machen. Dies hat den gleichen Effekt wie Datei=>Projekt öffnen... und dann das aktuelle Projekt auswählen.

2. Export/Import. Markieren Sie die Projektzeile oben im Explorer. Wenn Sie diese nicht geändert
haben, sollte sie "unbenanntes Modell" lauten. Führen Sie Datei=>Exportiere XMI aus und geben Sie als Dateiname "DeleteThis" im Dateiauswahl-Fenster ein. Markieren Sie den Reiter "Eigenschaften" im Detail-Fenster und ändern Sie den Namen des Modelles. Führen Sie die Datei=>Import XMI- Aktion aus. Sie wird sie fragen, ob Sie Ihre vorher gemachten Änderungen speichern möchten. Klicken Sie auf "Nein" und markieren Sie dann die "DeleteThis.xml"-Datei, die Sie gerade erzeugt haben. Überprüfen Sie den Namen des importierten Modell es mit dem von Ihnen gespeicherten.

3. Datei=>Dateien importieren... . Wir werden dies später behandeln. Sie können dies jetzt nicht testen, es sei denn, Sie haben Java-Quellcode zur Hand.


6. Datei=>Projekt-Einstellungen. Im Dialog Projekt-Einstellungen ist es möglich, die projektspezifischen Einstellungen zu konfigurieren. Er enthält Register für den Anwender, die Profile, die Notationen und die Darstellung der Diagramme. Um zum Beispiel die Sprach-Notation im Dialog Projekt-Einstellungen zu ändern klicken Sie auf Datei=>Projekt-Einstellungen... und wählen Sie den Register Notationen aus. Stellen Sie die Notation auf UML1.4 ein. Markieren Sie alle Optionen und klicken Sie auf Übernehmen und beobachten Sie die Änderungen im Diagramm. Stellen Sie die Standard-Schattenbreite auf 8 ein und klicken Sie auf Übernehmen. Beachten Sie, es passiert nichts. Das ist so, weil Sie nicht die Schattenbreite veränderten, sondern den Standardwert. Wenn Sie beim nächsten Mal eine Klasse in einem Diagram erstellen, wird dieser neue Schatten erscheinen.


1. Bearbeiten=>Markieren.
   • Markieren Sie ein Klassendiagramm im Explorer. Ist keines vorhanden, erstellen Sie eines mit Hilfe von Neues Diagramm=>Klassendiagramm. Erstellen Sie mit Hilfe des Klassenwerkzeuges drei Klassen, wie in der Referenz zur Anwenderschnittstelle unter
klassendiagrammspezifische Werkzeuge beschrieben. Führen Sie einen Doppelklick aus und klicken Sie dann im Editierfenster für Klassendiagramme auf die drei unterschiedlichen Klassen.

• Durch klicken auf das "Markieren"-Werkzeug machen Sie den aktuellen Modus rückgängig. Siehe "Durch klicken auf das "Markieren"-Werkzeug den aktuellen Modus rückgängig machen". Abschnitt 12.4.1, „Layout-Symbole“. Dies erlaubt es Ihnen auch andere Dinge, wie das Erstellen von Klassen im Editierfenster auszuführen.

• Öffnen Sie den gesamten Explorer-Baum, so dass alle Elemente sichtbar sind. Dann aktivieren Sie den ersten Menüeintrag: Bearbeiten=>Alles markieren . Dies markiert natürlich alle Elemente, aber nur die Elemente des aktuellen Diagrammes. Elemente die nur in anderen Diagrammen vorhanden sind, werden auf diesem Weg nicht markiert. Diese Funktion ist z.B. hilfreich, wenn Sie ein grosses Diagramm haben und alles verschieben müssen, um einige zusätzliche Elemente auf der linken oder oberen Seite des Diagrammes einzufügen.


• Führen Sie den Befehl Bearbeiten=>Markieren=>Aus Diagramm entfernen aus. Beachten Sie, dass die Klasse B aus dem Diagramm verschwunden ist, aber im Explorer immer noch existiert.


2. Bearbeiten=>Konfiguriere Perspektiven... Lesen Sie Abschnitt 11.5, „Perspektiven konfigurieren“. Wir werden dies zum jetzigen Zeitpunkt nicht vertiefen, da es sehr viel grösserer darzustellender Projekte bedarf, als wir Sie jetzt zur Verfügung haben.

3. Abbildung 3.7. Das erste ArgoUML-Fenster


Das einzige Verzeichnis ist standardmäßig das UML 1.4-Profil. Gehen Sie jetzt in 
Bearbeiten=>Einstellungen=>Profile und betrachten Sie das Feld "Standardprofile". Es enthält nur das gleiche UML 1.4-Profil. Fügen Sie das Java-Profil hinzu und drücken Sie dann OK.

Wenn Sie jetzt im Explorer nachsehen, hat sich nichts verändert, da Sie die Standardeinstellung verändert haben und nicht die Projekteinstellung. Drücken Sie Datei=>Neu und sehen sich die Profile im Explorer an: er zeigt jetzt sowohl das UML 1.4-Profil als auch das Java-Profil an.

Viele Beispiele in diesem Handbuch gehen davon aus, dass das Java-Profil verfügbar ist. Sie lassen es am Besten gleich aktiviert.


9. Bearbeiten=>Einstellungen=>Notationen . Bei den Menüelementen Datei=>Notation und Datei=>Eigenschaften... spielten wir bereits damit herum. Starten Sie eine weitere Instanz von ArgoUML und stellen Sie die Grösse jeder Instanz so ein, dass Sie sie gleichzeitig ansehen können. In einer Instanz stellen Sie die Notation auf UML (die aktuelle Einstellung zeigt auch noch die Versionsnummer an), in der anderen stellen Sie die Notation auf Java ein.


1. Ansicht=>Gehe zu Diagramm... blendet das Gehezu-Diagramm-Dialogfester auf. Markieren Sie den Eintrag Klassendiagramm und klicken Sie auf die Schaltfläche "Gehe zur Auswahl". In der Spalte am rechten Rand sollten 0 Knoten und 0 Kanten angezeigt werden. Klicken Sie auf die Schaltfläche "Schliessen". Im Fenster Details (Register Eigenschaften) geben Sie den Namen "Blort" ein. Erstellen Sie im Klassendiagramm zwei Klassen (Siehe ...) und geben Sie wieder in Ansicht=>Gehe zu Diagramm... Sie sollten jetzt 2 Knoten und 0 Kanten angezeigt bekommen. Klicken Sie erneut auf die Schaltfläche "Schliessen" und verbinden Sie die Klassen mit einem der "Linien"-Elemente wie Assoziation oder Vererbung. Gehen Sie erneut in Ansicht=>Gehe zu Diagramm... und Sie sollten 2 Knoten und 1 Kante(n) sehen. Klicken Sie erneut auf die Schaltfläche "Schliessen" und erstellen Sie eine dritte Klasse. Gehen Sie mit der Maus in die Symbolleiste zur Schaltfläche "Neue Beziehungsklasse". Klicken Sie dieses Symbol an und verbinden Sie die neue Klasse mit einer der anderen. Nachdem Sie auf "Neue Beziehungsklasse" geklickt haben, bewegen Sie die Maus über die neue Klasse. Halten Sie die Taste 1 gedrückt. Bewegen Sie die Maus zur anderen Klasse und lassen Sie die Taste 1 los. Gehen Sie erneut in Ansicht=>Gehe zu Diagramm... und Sie sollten 3 Knoten und 2 Kante(n) sehen. Obwohl es eine
Klasse ist und eine zweidimensionale Darstellung aufweist, zählt sie als Kante und nicht als Knoten. Markieren Sie andere Einträge in diesem Fenster und klicken Sie auf die Schaltfläche "Gehe zur Auswahl" im "Gehe zu Diagramm"-Fenster. Beobachten Sie die Änderungen im Explorer.


4. Ansicht=>Raster einstellen.
5. Ansicht=>Einrasten einstellen.
6. Ansicht=>Seitenumbrüche.
7. Ansicht=>XML Dump.


Zustands- und Aktivitätsdiagramme können nur erstellt werden, wenn eine Klasse oder ein Akteur markiert ist und die relevanten Menüeinträge nicht deaktiviert sind (unter diesen Umständen passiert nichts).

Menü Anordnen. Das Menü Anordnen erlaubt es Ihnen, Modellelemente im Diagramm auszurichten, zu verteilen und neu anzuordnen und die Layout-Strategie für das Diagramm einzustellen.

Menü Generieren. Das Menü Generieren erlaubt es Ihnen, Java Code für die markierten oder alle Klassen zu generieren.

Menü: Kriterien. Das Menü Kriterien erlaubt es Ihnen, die Prüfkriterien ein- und auszuschalten, die Wichtung der Designvorgaben und Designziele festzulegen und die verfügbaren Kriterien anzusehen.


26
Menü Hilfe. Dieses Menü gibt Zugriff auf die Details, wer das System erdachte und wo zusätzliche Hilfe gefunden werden kann.


3.4.2.2. Der Explorer

Jetzt sollten Sie sich die Zeit nehmen Kapitel 11, Der Explorer zu lesen. Der Explorer ist für alles was Sie tun fundamental. Wir werden im Folgenden immer wieder auf ihn zurückkommen.

Im Explorer gibt es vor dem Paketsymbol „unbenanntes Modell“ ein Steuerelement zum Expandieren und Reduzieren und im "zu bearbeiten"-Fenster vor dem Paketsymbol „Mittel“. Klicken Sie auf diese Steuerelemente und beachten Sie, dass diese Fenster Baumstrukturen sind, die sich besser benehmen als Sie es erwarten würden. Das Steuerelement zum Expandieren oder Reduzieren ist entweder ein Plus- (+)/Minus- (-) Zeichen oder ein Knauf mit einem rechten oder nach unten gerichteten Zeiger, je nach dem, welches Aussehen (Look and Feel) von Ihnen eingestellt wurde.


3.4.2.3. Der Editor

Jetzt sollten Sie sich die Zeit nehmen, das Kapitel 12, Das Editierfenster zu lesen.

Wenn wir durch das Editierfenster gehen, werden manchmal Änderungen im Detail- und im "zu bearbeiten"-Fenster auftreten. Beachten Sie diese im Moment nicht. Wir werden sie betrachten, wenn wir diese Fenster durchgehen.


Markieren Sie im Diagramm die andere Klasse. Beachten Sie, dass sich im Explorer der markierte Knoten entsprechend verändert. Greifen Sie diese Klasse und verschieben Sie sie aus dem Paket und beobachten Sie, was im Explorer passiert.
3.4.2.4. Der Bereich Details

Jetzt sollten Sie sich die Zeit nehmen Kapitel 13, Der Bereich Details zu lesen.

Anmerkung

- Zu bearbeiten. Diskutiert die Unterschiede zu den anderen Registern je markiertem Element. Beinhaltet die Einzelheiten für die Diskussion im "Zu bearbeiten"-Fenster.
- Eigenschaften,
- Dokumentation,
- Darstellung,
- Quellcode,
- Bedingungen,
- Stereotypen,
- Eigenschaftswerte,
- Checkliste.

3.4.2.5. Der Bereich Zu-Bearbeiten

Jetzt sollten Sie sich die Zeit nehmen Kapitel 14, Der Bereich Zu-Bearbeiten zu lesen.

Anmerkung

- Beschreibt Prioritäten.
- Lööst Elemente auf.
- Beziehung zum "Zu bearbeiten"-Register im Detail- Fenster.

3.4.2.6. Diagramme zeichnen

Grundsätzlich werden Diagramme mit Hilfe der Symbolleiste des Editors gezeichnet. Das gewünschte Modellelement wird markiert und durch anklicken der gewünschten Position im Diagramm positioniert.

Modellelemente, die sich bereits im Modell, aber nicht in einem Diagramm befinden, können dem Diagramm hinzugefügt werden, indem man das Modellelement im Explorer markiert, aus dem Drop-Down- Menü (Taste 2) Zum Diagramm hinzufügen auswählt und dann mit der Taste 1 auf die gewünschte Position im Diagramm klickt.

Genau so wie für UML-Modellelemente liefert die Werkzeugleiste des Editierfensters Zeichenobjekte (Rechtecke, Kreise, Linien, Polygone, Kurven, Text), um ergänzende Informationen in Diagramme
3.4.2.6.1. Diagrammelemente bewegen

Es gibt verschiedene Wege Diagrammelemente zu bewegen.

3.4.2.6.1.1. Mit Hilfe der Maustasten

Markieren Sie die Elemente, die Sie bewegen wollen. Durch drücken der Taste Strg können Sie mehrere Elemente markieren und zeitgleich bewegen.

Nun betätigen Sie Ihre Pfeiltasten. Ihre Elemente bewegen sich bei jedem Tastendruck ein wenig.

Wenn Sie zusätzlich die Umschalttaste betätigen, bewegen Sie sich ein wenig weiter.

3.4.2.6.1.2. Mit Hilfe der Symbolleiste des Editors

Klicken Sie auf die Schaltfläche "Besen" in der Symbolleiste. Bewegen Sie Ihre Maus in das Diagrammfenster, klicken rechts und halten sie gedrückt. Nun wird das Bewegen der Maus die Elemente ausrichten.

3.4.2.6.2. Elemente anordnen

Das Menü Anordnen erlaubt es Ihnen, Elemente auszurichten oder zu gruppieren.

3.4.2.7. Arbeiten mit Projekten

3.4.2.7.1. Das Startfenster

Abbildung 3.6, „Das erste ArgoUML-Fenster“ zeigt, wie ArgoUML nach dem erstmaligen Start erscheint.


Das Modell und die beiden leeren Diagramme können Sie im Explorer sehen, der für Sie das Hauptwerkzeug darstellt, um durch Ihr Modell zu navigieren.

Lassen Sie uns für einen Moment annehmen, dass dies jetzt der Zeitpunkt ist, wo Sie mit der Modellierung eines neuen Einkaufs- systems beginnen wollen. Sie wollen ihm den Namen einkaufsmoell geben und Sie wollen es in einer Datei mit dem Namen ErstesProjekt speichern.

3.4.2.7.2. Ein Projekt speichern - Das Menü: Datei

Im Moment speichert ArgoUML Diagramme in einem früh veröffentlichtem Standard, der Precision Graphics Markup Language (PGML). Für diejenigen, die davon Gebrauch machen wollen, gibt es allerdings die Option, die grafischen Daten als SVG zu exportieren. Wenn ArgoUML UML 2.0
unterstützt, werden die Diagramme im UML 2.0 Diagramm Austausch Format gespeichert.

Zuerst lassen Sie uns das Modell in seinem aktuellen (leeren und unbenannten) Zustand speichern. Klicken Sie in der Menüzeile auf Datei, dann auf Projekt speichern unter.... Sie auch Abbildung 3.8 „Projekt speichern unter... aufrufen“.

**Abbildung 3.8. Projekt speichern unter... aufrufen**

Bitte beachten Sie, dass das Menü Datei die üblichen Optionen für das Erzeugen eines neuen Projektes, für das Öffnen eines existierenden Projektes, für das Speichern eines Projektes unter einem neuen Namen, für das Drucken des aktuell angezeigten Diagrammes, für das Speichern des aktuell angezeigten Diagrammes als Datei und für das Beenden des Programmes enthält.


**3.4.2.7.3. Der Dateiauswahl-Dialog**

Aber lassen Sie uns zum Speichern unseres Projektes zurückgehen. Nachdem wir das Menü-Kommando Projekt speichern unter... angeklickt haben, öffnet sich der Dateiauswahl-Dialog, in dem wir den von uns gewünschten Dateinamen eingeben können. Siehe auch Abbildung 3.9 „Der Dateiauswahl-Dialog“.

**Abbildung 3.9. Der Dateiauswahl-Dialog**
Dies ist eine Standard-Java-Dateiauswahl. Lassen Sie uns in einige Details einsteigen.


Doppelklicken auf eine der angezeigten Verzeichnisse navigiert Sie in dieses Verzeichnis und erlaubt es Ihnen, schnell in der Verzeichnishierarchie Ihrer Festplatte hinunterzunavigieren.


Beachten Sie auch, dass der aktuell markierte Verzeichnismname im oberen Drop-Down-Steuerelement mit der Bezeichnung Speichern in: angezeigt wird. Ein einziger Klick auf ein Verzeichnis innerhalb des Scrollbereiches markiert das Verzeichnis am Bildschirm, markiert das Verzeichnis allerdings nicht zum speichern.

Im oberen Bereich des Dialoges, über dem scrollbaren Verzeichnis- Auswahllbereich, gibt es einige weitere Verzeichnis-Navigations- Werkzeuge.

•

Das Verzeichnis-Drop-Down-Steuerelement. Klicken auf den Pfeil zeigt eine Baumansicht der Verzeichnis-Hierarchie an, erlaubt es Ihnen schnell in der Hierarchie nach oben und unten zu
navigieren und gleichzeitig schnell zu bestimmen, wo Sie sich aktuell in der Hierarchie befinden.


- Das Symbol: Darstellung der Verzeichnisse.


Sie haben nun ein aktives Projekt, ErstesProjekt genannt, das mit der Datei ErstesProjekt.zargo verbunden ist.

### 3.4.3. Ausgabe

#### 3.4.3.1. Laden und Speichern

##### 3.4.3.1.1. XMI-Dateien in ArgoUML speichern

ArgoUML speichert die Diagramminformationen in einer PGML-Datei (mit der Erweiterung .pgml, die Modellinformation in einer XMI-Datei (mit der Erweiterung .xmi und die Information über das Projekt in einer Datei mit der Erweiterung .argo. Mehr über PGML und XMI siehe unter Abschnitt 3.4.3.2.2, „Precision Graphics Markup Language (PGML)“ und Abschnitt 3.4.3.3, „XMI“.


**Warnung**

Beachten Sie, sofern ein ZIP-Dienstprogramm installiert ist, dass ein Doppelklick das
ZIP-Dienstprogramm starten wird und NICHT Argo.

3.4.3.2. Grafiken und Drucken

3.4.3.2.1. Das Graph Editing Framework (GEF)


3.4.3.2.2. Precision Graphics Markup Language (PGML)

PGML ist das aktuelle Speicherformat für in ArgoUML verwendete Diagramminformationen. In Zukunft wird PGML durch das UML 2.0 Diagramm-Austausch-Format ersetzt.

3.4.3.2.3. Anwendungen, die PGML öffnen

PGML ist ein Vorläufer von SVG (siehe Abschnitt 3.4.3.2.5, „Scalable Vector Graphics (SVG)“). Er wurde durch das W3C-Konsortium verworfen.

Aktuell kennen wir keine anderen Tools, die mit PGML arbeiten.

3.4.3.2.4. Diagramme drucken

Markieren Sie ein Diagramm und gehen Sie dann in Datei=>Grafik exportieren... Sie können GIF-, PostScript-, Encapsulated PostScript- oder SVG-Format generieren.

3.4.3.2.5. Scalable Vector Graphics (SVG)

Ein Standard-Vektor-Grafik-Format des weltweiten Web-Konsortiums (W3C) (http://www.w3.org/TR/SVG/).


3.4.3.2.6. Diagramme als SVG speichern

1. Wählen Sie .svg als Dateityp aus.

Et viola! SVG! Probieren Sie es aus und erforschen Sie es etwas... Sie sind nicht schön genug? Wenn Sie etwas über das Rendern schöner SVG-Formate wissen, lassen Sie es uns wissen.


Anmerkung

Sie wollen keine Scrollleisten für Ihre SVG haben, es sei denn Sie sind in HTML eingebettet. Viel Glück und lassen Sie es uns wissen, was Sie das finden!
3.4.3.3. XMI

ArgoUML unterstützt XMI 1.0, 1.1 und 1.2 Dateien, die UML 1.3 und UML 1.4-Modelle enthalten. Um die beste Kompatibilität zu erhalten, exportieren Sie Ihre Modelle in UML 1.4 und XMI 1.1 oder 1.2. Stellen Sie sicher, dass all proprietären Erweiterungen abgeschaltet sind (wie z.B. Poseidon's Diagrammdaten).

Mit UML-Versionen vor UML 2.0 ist es nicht möglich Diagramm- Informationen zu speichern, sodass keine Diagramme transferiert werden.


3.4.3.3.1. XMI von Rational Rose benutzen

...

3.4.3.3.2. Von Poseidon erzeugte Modelle benutzen

Im Exportiere als XMI..., aber stellen Sie sicher, das Speichere mit Diagrammdaten nicht markiert ist.

3.4.3.3.3. Von MagicDraw erzeugte Modelle benutzen

...

3.4.3.3.4. XMI Kompatibilität mit anderen ArgoUML-Versionen


Wenn Sie zusätzlich XMI-Dateien speichern, die durch andere Werkzeuge gelesen werden sollen, sollten Sie die unterschiedlichen Versionen unterscheiden. Die meisten modernen UML-Modellierungs-Werkzeuge sollten UML 1.4 lesen, aber Sie haben vielleicht In-Haus-Codegeneratoren oder andere Werkzeuge, die UML 1.3 benötigen.

3.4.3.3.5. Andere XMI-Formate in ArgoUML importieren

Die XMI-Kompatibilität zwischen den UML-Modellierungs-Werkzeugen wurde über die Jahre verbessert, aber Sie werden wahrscheinlich Probleme bekommen.

ArgoUML wird keine XMI-Dateien lesen, die UML 1.5 oder UML 2.0- Modelle enthalten, aber es sollte möglich sein, die meisten UML 1.4 und UML 1.3-Modelle zu öffnen. Wenn Sie eines finden, dass Sie nicht öffnen können, erstellen Sie bitte einen Bug- Bericht, so dass ein Entwickler dies erforschen kann.

3.4.3.3.6. XMI Format generieren

Wählen Sie das Kommando Datei=> Exportiere als XMI... und wählen Sie einen Dateinamen.

3.4.3.4. Code-Generierung

3.4.3.4.1. Durch ArgoUML generierter Code
Es ist möglich, Ihren generierten Code mit ArgoUML zu übersetzen. Sie müssen lediglich die Methodenrumpfe implementieren, um verwendbare Ergebnisse zu erzielen.

3.4.3.4.2. Code für Methoden generieren


Sie können uns hier helfen, wenn Sie wollen!

3.4.4. Arbeiten mit Design-Kritiken

Designkritiken sind Teil der praktischen Anwendung der Theorie der Kognitiven Psychologie, die in ArgoUML implementiert ist. Siehe Abschnitt 3.3.1, „Kognitive Psychologie“

3.4.4.1. Meldungen von den Design-Kritiken


Abbildung 3.10. ArgoUML-Fenster mit gespeichertem Projekt ErstesProjekt.zargo
Das Projekt enthält ein Paket auf oberster Ebene, genannt unbeanntesModell, welches ein Klassendiagramm und ein Anwendungsfalldiagramm enthält.

Wenn wir auf den Bildschirm blicken, sehen wir, dass das Verzeichnis „Mittel“ im "zu bearbeiten"-Fenster (das linke untere Fenster) einige Elemente enthalten muss, da sein Aktivierungs-Symbol angezeigt wird.

Klicken auf dieses Symbol öffnet das Verzeichnis „Mittel“. Ein geöffnetes Verzeichnis wird durch das Symbol dargestellt.

Aber, was ist dieses „zu bearbeiten“-Fenster überhaupt? Sie haben bis jetzt noch nichts aufgezeichnet, was zu tun wäre. Wo kommen diese "zu bearbeiten"-Elemente jetzt her?

Die Antwort ist simpel und ist gleichzeitig eines der wichtigsten Punkte von ArgoUML. Während Sie mit Ihrem UML-Modell arbeiten, wird Ihre Arbeit kontinuierlich durch ein Stück Code, Design-Kritik genannt, gemonitort. Das ist ähnlich einem persönlichen Mentor, der Ihnen über die Schulter sieht und Sie jedesmal darauf hinweist, wenn er irgendetwas fragwürdiges in Ihrem Design sieht.

Kritiken sind etwas völlig unaufdringliches. Sie geben Ihnen eine freundliche Warnung, aber Sie zwingen Sie nicht in Design- Prinzipien, denen Sie nicht folgen wollen. Lassen Sie uns einen Blick darauf werfen, was die Kritiken uns mitteilen. Klicken Sie auf das Symbol in der Nähe des Verzeichnisses Mittel und klicken Sie auf das Element Überprüfen Sie den Paketnamen unbeanntesModell.

Abbildung 3.11. „ ArgoUML-Fenster zeigt das Kritik-Element Überprüfen Sie den Paketnamen unbeanntesModell“ zeigt, wie Ihr Bildschirm nun aussieht.

Abbildung 3.11. ArgoUML-Fenster zeigt das Kritik-Element Überprüfen Sie den Paketnamen unbeanntesModell
Beachten Sie, dass Ihre Markierung im "zu bearbeiten"-Fenster in rot markiert ist und dass jetzt eine vollständige Erläuterung im Detail-Fenster (das untere rechte Fenster) erscheint. Sie müssen vielleicht Ihr Detail-Fenster in der Größe anpassen oder herunterscrollen, um die angezeigten Nachricht in Ihrem Beispiel vollständig sehen zu können.


An diesem Punkt können Sie wählen, ob Sie den Paketnamen manuell ändern oder durch die Design-Kritik stillschweigend dieses eine Mal oder immer ändern wollen.

Wir werden nichts davon tun (wir kommen darauf zurück, wenn wir über die Design-Kritiken detaillierter sprechen), aber wir werden eine andere einfache Funktion von ArgoUML nutzen - die Autokorrektur-Funktion.

Um dies zu tun, klicken Sie auf die Schaltfläche Weiter im Detailfenster. Dadurch wird ein Umbenennungs-Assistent innerhalb des Eigenschaftsfensters angezeigt, der vorschlägt, den Namen unbenanntesmodell (alles in Kleinbuchstaben) zu verwenden.

3.4.4.2. Design-Kritiken bei der Arbeit: Der Paket-Umbenennungs-Assistent

Abbildung 3.12. Das ArgoUML-Fenster zeigt den Kritik-Assistenten, zur Umbenennung des Paketes

Beobachten Sie nun, wie die Design-Kritik im "zu bearbeiten"-Fenster verschwindet und es verbleibt nur der Hinweis Fügen Sie dem Paket einkaufsmodell Elemente hinzu in der "zu bearbeiten"-Liste.


Die Änderung des Paketnamens sollte sich auch im Explorer, in der linken oberen Ecke Ihres ArgoUML-Fensters, wiederspiegeln.

Wir sind jetzt so weit, unser erstes UML-Diagramm, ein Anwendungsfalldiagramm, zu erstellen. Aber lassen Sie uns zuerst speichern, was wir bis jetzt getan haben.

Klicken Sie auf das Menüelement Datei und wählen Sie Projekt speichern... aus. Sie können jetzt ArgoUML sicher beenden, ohne Ihre bisherige Arbeit zu verlieren, oder mit dem Erstellen Ihres ersten Diagrammes fortfahren.

3.5. Die Fallstudie (Noch zu schreiben)

Hier wollen wir mit der Fallstudie beginnen. Es ist der Zeitpunkt, wo Sie Ihr Projekt definieren; nicht Ihr Produkt, aber Ihr Projekt. Es kann argumentiert werden, dass die Modellierungskonzepte hier auch erscheinen sollten, aber das wurde so nicht aufgebaut. Wenn Sie sich die Zeit nehmen, sich das

...
Kapitel 4. Erfassen der Anforderungen

4.1. Einleitung

Das Erfassen der Anforderungen ist der Identifizierungsprozess, was der „Kunde“ von dem vorgeschlagenen System will.


Das grösste Risiko, dass wir beim Erfassen der Anforderungen haben ist, dass wir in Begriffen der möglichen Lösung anfangen zu denken. Dies muss bis zur Analyse-Phase warten (siehe Kapitel 5, Analyse). Einer der Schritte der Analyse-Phase nimmt das Ergebnis der Anforderungsphase und übersetzt es in die Sprache einer gedachten Lösung.

Erinnern Sie sich, dass wir beides einsetzen, einen inkrementalen und einen iterativen Prozess.

Wir kommen im Anforderungsprozess darauf zurück, wenn wir das Problem in kleinere Teile unterteilen. Für jedes davon müssen seine Anforderungen erfasst sein.

Wir werden wahrscheinlich während der Anforderungsphase bei jeder Iteration darauf zurückkommen, wenn wir versuchen, die Anforderungen des Systems immer weiter zu definieren.

Anmerkung


4.2. Der Anforderungs-Erfassungs-Prozess

Wir beginnen mit einer Top-Level-Sicht auf das von uns zu lösende Problem und die Schlüsselbereiche der Funktionalität, die wir für die Lösung adressieren müssen. Dies ist unser Visions- Dokument und es sollte nur einige Seiten lang sein.

Die Top-Level-Sicht eines Geldautomaten (automated teller machine (ATM)) zum Beispiel, sollte folgendes unterstützen.

1. Bargeld lagern, Bargeld abheben und Kontoabfragen durch Kunden.
2. Warten des Equipments durch Bank-Ingenieure und leeren der Kassen und Bargeld nachfüllen durch die lokale Bankfiliale.

Aus dieser Top-Level-Sicht können wir die prinzipiellen Aktivitäten des Systems und die extern Handelnden (Menschen, Equipment) die in diese Aktivitäten eingebunden sind extrahieren. Diese Aktivitäten sind als Anwendungsfälle und die extern Handelnden als Akteure bekannt.

Akteure können Menschen oder Maschinen sein. Vom praktischen Standpunkt aus gesehen besteht deren Wert darin, die Nutzer hinter einer Maschine zu kennen, da nur diese in der Lage sind, sich mit
dem Anforderungs-Erfassungs-Prozess zu beschäftigen.


Es gibt keine harte und schnelle Regel. Einige Architekten präferieren wenige relativ grosse Anwendungsfälle, andere präferieren eine grössere Anzahl kleinerer Anwendungsfälle. Eine nützliche Faustregel ist, dass jedes praktikable Projekt nicht mehr als 30 Anwendungsfälle erfordern sollte (wenn es mehr benötigt, sollte es in separate Projekte aufgeteilt werden).


Wir müssen dann eine detailliertere Spezifikation für jeden Anwendungsfall erstellen. Dies beinhaltet sein normales Verhalten, alternatives Verhalten und alle Vor- und Nachbedingungen. Dies erfolgt in einem Dokument, dass häufig als Anwendungsfalldokumentation oder Anwendungsfallsszenario bekannt ist.

Da Anwendungsfälle natürlicherweise funktional sind, benötigen wir ein Dokument, um die nicht-funktionalen Anforderungen (Kapazitäts-, Leistungs-, Umgebungsanforderungen usw.) aufzunehmen. Diese Anforderungen werden in einem Dokument Ergänzende Anforderungsspezifikation festgehalten.

4.2.1. Prozess-Schritte

Die Schritte im Anforderungs-Erfassungs-Prozess können wie folgt zusammengefasst werden.

1. Das Visions-Dokument beinhaltet eine Gesamtsicht auf das Problem und die gewünschten Charakteristika seiner Lösung.

2. Identifizieren Sie die Anwendungsfälle und Akteure aus dem Visions-Dokument und stellen deren Beziehungen zueinander in einem oder mehreren Anwendungsfalldiagrammen dar.

3. Erstellen Sie für jeden Anwendungsfall eine detaillierte Anwendungsfall-Spezifikation, die das normale und alternative Verhalten, die Vor- und Nachbedingungen beinhaltet.


In jedem iterativen Entwicklungsprozess werden wir priorisieren und frühe Iterationen werden sich darauf fokussieren, das hauptsächliche Verhalten der wichtigsten Anwendungsfälle aufzunehmen.

Die meisten modernen Anforderungs-Erfassungs-Prozesse unterstellen, dass es wichtig ist, dass der maßgebliche Repräsentant des Kunden während dieses Prozesses vollständig involviert ist.

4.3. Ergebnis des Anforderungs-
Erfassungs-Prozesses

Fast alle Ergebnisse des Anforderungs-Erfassungs-Prozesses sind dokumentarisch. Das einzige Diagramm ist das Anwendungsfalldiagramm, das die Beziehungen zwischen den Anwendungsfällen und den Akteuren darstellt.

4.3.1. Visions-Dokument

Typische Kapitel dieses Dokumentes könnten die folgenden sein.

- **Zusammenfassung.** Eine Aussage zum Umfeld, zum Problem und zu den Zielen der Lösung.

- **Ziele.** Was wir erreichen wollen (und wie wir es erreichen wollen).

- **Marktumfeld oder vertragliche Vereinbarungen.** Bei einer Entwicklung zur Marktführerschaft, sollte es die Zielmärkte, die Alleinstellungsmerkmale, die zwingenden Ereignisse und so weiter aufzeigen. Bei einer vertraglich vereinbarten Entwicklung sollte es die Hauptelemente der vertraglichen Vereinbarung erläutern.


- **Haupteigenschaften.** Auf einer sehr hohen Abstraktionsebene: Was sind die hauptsächlichen Aspekte des Problems/der gewünschten Lösung. Es ist hilfreich, hier bereits zu priorisieren.


- **Anhang.** Eine Liste der Akteure und Anwendungsfälle die zur Erfüllung der Vision notwendig sind. Es ist hilfreich, diese mit den vorher beschriebenen Abschnitten zu verlinken um eine in sich geschlossene Darstellung sicherzustellen.

4.3.2. Anwendungsfalldiagramm


Abbildung 4.1, „Grundlegendes Anwendungsfalldiagramm für einen Geldautomaten“ zeigt, wie das in einem Anwendungsfalldiagramm dargestellt werden kann. Die Anwendungsfälle werden als Ovale dargestellt, die Akteure als Strichmännchen (auch wenn es Maschinen sind), mit Linien (als Assoziationen bekannt), die die Anwendungsfälle mit den damit involvierten Akteuren verbinden. Ein Rahmen um die Anwendungsfälle stellt die Abgrenzung zwischen dem System (durch die Anwendungsfälle definiert) und den Akteuren dar, die extern sind.

Anmerkung
Nicht alle Analytiker möchten einen Rahmen um die Anwendungsfälle verwenden. Dies ist ein Fall persönlicher Vorlieben.

Abbildung 4.1. Grundlegendes Anwendungsfalldiagramm für einen Geldautomaten

Die folgenden Abschnitte zeigen, wie das grundlegene Anwendungsfalldiagramm erweitert werden kann, um zusätzliche Informationen über das zu designende System darzustellen.

4.3.2.1. Aktive und passive Akteure


Dies ist ein gutes Beispiel, wo die Pfeile helfen, da es uns erlaubt, ein ereignisgetriebenes System (der Geldautomat initiiert die Interaktion mit dem Zentralcomputer) von einem zyklisch abfragenden System (der Zentralcomputer fragt den Geldautomaten von Zeit zu Zeit ab) zu unterscheiden.

Dort wo ein Akteur entweder aktiv oder passiv sein kann, je nach den Umständen, kann der Pfeil weggelassen werden. Im Geldautomaten-Beispiel fällt der Bankingenieur in diese Kategorie. Normalerweise ist er aktiv, indem er regelmässig auftaucht um die Maschine zu warten. Wenn jedoch der Geldautomat einen Fehler entdeckt, kann er den Ingenieur auffordern diesen zu beheben.


4.3.2.2. Kardinalität


Im Geldautomaten-Beispiel gibt es nur einen Zentralcomputer, aber es können beliebig viele Nutzungen des Geldautomaten aufgezeichnet werden. Daher plazieren wir das Kennzeichen 0..* am Ende des Anwendungsfalles. Es gibt keine Notwendigkeit für eine Kennzeichnung am anderen Ende, da der Standard 1 ist.

Eine lokale Bank kann bis zu drei Beamte haben, die autorisiert sind, die Geldautomaten zu leeren und zu befüllen. Daher plazieren wir bei der Beziehung zwischen dem Akteur und dem Anwendungssfall Wartung Geldautomat am Ende zum Akteur die Kennzeichnung 1..3. Sie können sich mit beliebig vielen Geldautomaten befassen, so dass wir auf dem anderen Ende das Kennzeichen 0..* plazieren.

Es gibt beliebig viele Kunden und beliebig viele Geldautomaten die diese nutzen dürfen. Daher plazieren wir an jedem Ende der Assoziation das Kennzeichen 0..*.

Abbildung 4.3, „Anwendungsfalldiagramm für einen Geldautomaten und dargestellter Kardinalität.“ zeigt das Geldautomaten-Anwendungsfalldiagramm mit dargestellter Kardinalität.
4.3. Hierarchien von Anwendungsfällen

In unserem Geldautomatenbeispiel haben wir jetzt drei Anwendungsfälle, um das Verhalten des Systems zu beschreiben. Auch wenn Anwendungsfälle immer einen signifikanten Teil des Systemverhaltens beschreiben sollten, wenn diese zu generell sind, können sie schwer zu beschreiben sein.


Die Aufteilung eines Anwendungsfalles in einfacherere Unteranwendungsfälle wird in UML durch eine include Beziehung, einem gestrichelten Pfeil vom Hauptanwendungsfall zum Unteranwendungsfall mit der Kennzeichnung «include» bezeichnet.

Abbildung 4.4. Anwendungsfalldiagramm für einen Geldautomaten mit Erfassen der Anforderungen
Include-Beziehungen.

Include-Beziehungen sind ideal für das Herunterbrechen des Anwendungsfallverhaltens in Hierarchien. Wir wollen jedoch auch einen Anwendungsfall, der eine Erweiterung (extension) eines existierenden Anwendungsfalles darstellt, in bestimmten Umständen darstellen.

Im Geldautomatenbeispiel haben wir einen Anwendungsfall „Equipment warten“, der die Routinewartung des Geldautomaten beinhaltet. Wir wollen aber auch den Spezialfall einer ungeplanten Reparatur abdecken, die durch den vom Geldautomaten erkannten internen Fehlers ausgelöst wurde.

Dies wird in UML durch die extend-Beziehung dargestellt. Im Hauptanwendungsfall spezifizieren wir einen Namen für einen Ort in der Beschreibung, an den die Erweiterung des Verhaltens hinzugefügt werden kann. Der Name und der Ort werden in einem separaten Abschnitt innerhalb des Anwendungsfall-Ovales dargestellt. Die Darstellung der extend-Beziehung entspricht der include-Beziehung, aber mit dem Kennzeichen «extend». An der extend-Beziehung spezifizieren wird die Bedingung, unter der das Verhalten hinzugefügt wird.


### 4.3.3. Die Anwendungsfall-Spezifikation


Eine typische Anwendungsfall-Spezifikation wird folgende Kapitel enthalten.

- **Name.** Der Name des Anwendungsfalles.
- **Ziel.** Eine ein oder zweizeilige Zusammenfassung darüber, was dieser Anwendungsfall für seine Akteure ausführt.
- **Akteure.** Die in diesen Anwendungsfall involvierten Akteure und jeden Umstand bezüglich Ihrer Einbindung.
Anmerkung

Dies sollte keine Beschreibung des Akteurs sein. Diese sollte mit dem Akteur im Anwendungsfalldiagramm verknüpft sein.

- **Vorbedingung.** Diese würden besser als „Voraussetzungen“ bezeichnet, aber der überall verwendete Begriff ist Vorbedingungen. Dies ist eine Beschreibung jeder vereinfachenden Voraussetzung, die wir zum Start des Anwendungsfalles machen können.

Im Geldautomatenbeispiel könnten wir beim Anwendungsfall „Equipment warten“ die Voraussetzung haben, das immer ein Ingenieur verfügbar ist und wir uns nicht vor dem Fall fürchten müssen, in dem ein Routinewartungs-Besuch ausgelassen wurde.

Achtung

Vermeiden Sie Vorbedingungen wo immer das möglich ist. Sie müssen sich absolut sicher sein, dass die Vorbedingung unter allen möglichen Umständen eingehalten wird. Wenn nicht, ist ihr System zu wenig spezifiziert und wird daher fehl schlagen, wenn die Vorbedingung nicht wahr ist. Wenn Sie nicht sicher sein können, dass die Vorbedingung immer wahr ist, müssen Sie einen zweiten Anwendungsfall spezifizieren, der den Fall handhabt, wenn die Vorbedingung falsch ist. Im ersten Fall sind die Vorbedingungen die Ursache der Probleme, im zweiten Fall die Ursache für mehr Arbeit.

- **Standardablauf.** Die aufeinander folgenden Schritte, die das Verhalten des Anwendungsfalles im „Normalfall“ beschreiben. Wo ein Anwendungsfall mehrere normale Szenarien aufweist, wird einer davon willkürlich ausgewählt. Die Spezifizierung des Standardablaufes wird nachfolgend detaillierter beschrieben Abschnitt 4.3.3.1, „“.

- **Alternative Abläufe.** Eine Reihe von linearen Sequenzen beschreiben jede der alternativen, gegenüber dem Standardablauf abweichenden Verhaltensweisen. Die Spezifizierung alternativer Abläufe ist detaillierter in Abschnitt 4.3.3.2, „“ beschrieben.

- **Nachbedingungen.** Dies ist der Zustand jeder Nachbedingung, den wir am Ende des Anwendungsfalles feststellen können. Sehr hilfreich, wo der Anwendungsfall einer von einer Serie von in den Hauptanwendungsfall eingebundenen Unteranwendungsfällen ist, wo sie die Vorbedingung für den nächsten einzubindenden Anwendungsfall bilden.

Achtung

Nachbedingungen sind wie Vorbedingungen am Besten zu vermeiden. Sie bilden eine Belastung für die Spezifikation des Anwendungsfallablaufes, das sichergestellt sein muss, dass die Nachbedingung immer eingehalten wird. Daher sind sie auch eine Ursache von Problemen und zusätzlicher Arbeit.

- **Anforderungen.** In einer idealen Welt würden das Visionsdokument, die Anwendungsfalldiagramme, die Anwendungsfallbeschreibungen und die Spezifikation der zusätzlichen Anforderungen die Anforderungen für ein Projekt bilden.

Bei den meisten Marktführer-Entwicklungen, ist es gewöhnlich der Fall, dass sich die
Eigentumsrechte der Anforderungen im gleichen Business befinden, wie das Team, das die Entwicklung durchführen wird. Die Marketingabteilung kann das Erfassen und die Analyse der Anwendungsfällenanforderungen erlernen, um ihre kundenspezifischen Aktivitäten damit zu verbinden.

Bei externen Vertragsentwicklungen jedoch, kann der Kunde auf einer traditionellen „Liste von Features“ als Basis des Vertrages bestehen. Wo dies der Fall ist, sollte dieser Abschnitt der Anwendungsfallspezifikation mit den vertraglich festgelegten Features übereinstimmen, die durch den Anwendungsfall abgedeckt werden.

Das wird oft mit Hilfe eines Third-Party-Werkzeuges ausgeführt, das Dokumente verknüpfen kann und eine automatische Prüfung des Geltungsbereiches enthält. In diesem Fall wird dieser Abschnitt nicht benötigt oder kann automatisch generiert werden.

Die abschließende Größe der Anwendungsfallspezifikation hängt von der Komplexität der Anwendungsfälle ab. Als Faustregel gilt, dass die meisten Anwendungsfälle ca. 10-15 Seiten für die Spezifikation benötigen, das meiste davon für die alternativen Abläufe. Wenn sie sehr viel größer sind, sollten Sie die Anwendungsfälle aufteilen. Wenn sie sehr viel kleiner sind, betrachten Sie, ob der Anwendungsfall einen zu kleinen Bereich des Verhaltens beschreibt.

4.3.3.1. Alle Abläufe in einer Anwendungsfallspezifikation sind linear - d.h. es gibt keine bedingte Verzweigung. Jede Auswahl im Ablauf wird durch einen anderen, nach dem Auswahlpunkt kommenden alternativen Ablauf behandelt. Es ist wichtig, sich daran zu erinnern, dass wir hier das Verhalten und nicht die Programmierung spezifizieren.


In unserem Geldautomatenbeispiel könnten wir im Anwendungsfall „Bargeld ausgeben“ die folgende Sequenz von Schritten im Standardablauf haben:

1. Der Kunde gibt an, dass eine Quittung erforderlich ist.
2. Der Kunde gibt die gewünschte Bargeldmenge ein.
3. Der Geldautomat überprüft mit dem Zentralcomputer, dass der Kunde diese Auszahlung durchführen kann.
4. Der Geldautomat gibt das Bargeld an den Kunden.
5. Der Geldautomat gibt die Quittung an den Kunden aus.


Anmerkung

Der erste Schritt ist keine Bedingung. Wir nehmen als unseren Standardablauf den Fall, wo
4.3.3.2.

Dies erfasst die alternativen Szenarien, wie lineare Abläufe, die durch den Standardablauf referenziert werden. Zu Beginn erzeugen wir eine Liste der alternativen Abläufe.

A.

A.1. Der Kunde benötigt keine Quittung.
A.2. Das Kundenkonto unterstützt keine Auszahlung.
A.4. Der Kunde unterbricht die Transaktion.
A.5. Der Kunde macht beim Annehmen des Bargeldes Fehler.

Nachfolgend arbeiten wir jeden alternativen Ablauf als Referenz zum Standardablauf aus. Der erste alternative Ablauf könnte zum Beispiel wie folgt aussehen:

A.

A.1. Der Kunde benötigt keine Quittung.

A.1. In Schritt 1 des Standardablaufes sagen die Kunden, dass sie keine Quittung benötigen.
A.1. Der Standardablauf geht von Schritt 2 in Schritt 4, Schritt 5 wird nicht benötigt.


4.3.3.3. Iterative Entwicklung der Anwendungsfallspezifikationen

Die Iterative Entwicklung wird die die Anwendungsfälle priorisieren und die erste Iteration wird die wichtigsten adressieren.

Frühe Iterationen werden den Standardablauf der wichtigsten Anwendungsfälle nur mit den grundlegenden Details erfassen und die Überschriften der hauptsächlichen alternativen Abläufe auflisten.

Spätere Iterationen werden die verbleibenden Anwendungsfälle adressieren, die Schritte der individuellen alternativen Abläufe ausformulieren und wahrscheinlich mehr Details über die individuellen Schritte enthalten.

4.3.4. Ergänzende Anforderungsspezifikation

Dies erfasst die nicht-funktionalen Anforderungen oder Randbedingungen, die für das System gelten. Da Anwendungsfälle von Natur aus inhärent funktional sind, können sie diese Art von Information nicht aufnehmen.
Einige Analytiker plazieren nicht-funktionale Anforderungen in einem Abschnitt am Ende einer jeden Anwendungsfallspecifikation, der die nicht-funktionalen, anwendungsfallbezogenen Anforderungen enthält.


- **Geschwindigkeit.** Prozessorleistung, Anwender-/Ereignis-Antwortzeiten, Bildauffrischungszeiten.
- **Größe.** Hauptspeicher (und mögliche Zwischenspeicher), Plattenkapazität.
- **Leichte Anwendbarkeit.** Ausbildungszeit, Stil und Details des Hilfesystems.
- **Zuverlässigkeit.** Durchschnittliche Fehlerauftretenszeit, die Nichtverfügbarkeitswahrscheinlichkeit, die Fehlerrate, die Verfügbarkeit.
- **Widerstandsfähigkeit.** Wiederanlaufzeit nach einem Fehler, Prozentsatz der Ereignisse, die Fehler verursachen, die Wahrscheinlichkeit von Datenverlust bei einem Fehler.
- **Portabilität.** Prozentsatz von zielabhängigem Code/Klassen, Anzahl der Zielsysteme.

Dazu sollten wir Abschnitte über die Umgebung (Temperatur, Feuchtigkeit, Blitzschutz) und Übereinstimmung mit Standards hinzufügen.

### 4.4. Anwendungsfälle in ArgoUML verwenden

ArgoUML erlaubt es Ihnen Anwendungsfalldiagramme zu zeichnen. Wenn Sie ein neues Projekt erstellen, ist ein Anwendungsfalldiagramm standardmäßig mit dem Namen Anwendungsfalldiagramm 1 angelegt. Wählen Sie diesen Diagrammnamen im Explorer durch einen Klick mit der Taste 1 aus (der obere linke Quadrant des ArgoUML-Fensters).

Neue Anwendungsfalldiagramme können, wenn benötigt, über Erzeuge Diagramm in der Menüzeile oder über die Werkzeugleiste erstellt werden. Sie werden im Bearbeitungfenster (der obere rechte Quadrant des ArgoUML-Fensters) bearbeitet.

#### 4.4.1. Akteure

Um einen Akteur dem Diagramm hinzuzufügen, klicken Sie mit der Taste 1 auf das Akteur-Symbol in der Werkzeugleiste des Bearbeitfensters ( ) und dann klicken Sie mit der Taste 1 an die Stelle, wo Sie ihn plazieren wollen. Der Akteur kann nachträglich durch eine Taste 1-Bewegung bewegt werden (z.B. über dem Akteur die Taste 1 herunterdrücken, um diesen zu markieren, ihn an die neue Position bewegen und die Taste 1 loslassen, um den Akteur an diese Stelle zu bringen).
Mehrere Akteure können in einem Schritt durch einen Doppelklick mit der Taste 1 auf das Akteursymbol hinzugefügt werden. Jeder nachträgliche Taste 1-Klick wird einen Akteur in das Diagramm bringen. Ein Klick mit der Taste 1 auf das Auswahlsymbol ( ) wird das Hinzufügen der Akteure stoppen.

Der Name des Akteurs wird in seinem Eigenschaftsfenster vergeben. Zuerst markieren Sie den Akteur (wenn nicht bereits markiert) im Bearbeitungsfenster mit Hilfe des Taste 1-Klick. Dann klicken Sie auf das Eigenschaften-Register im Detailfenster. Der Name wird im Feld Name eingegeben und wird am Bildschirm erscheinen.

Als eine Abkürzung, die Ihnen erlaubt, den Namen direkt einzugeben, führen Sie mit der Taste 1 einen Doppelklick auf den Namen des Akteurs im Bearbeitenfenster aus (oder geben Sie ihn einfach über die Tastatur ein, wenn der Akteur markiert ist). Dies ist ein komfortabler Weg, einen Namen für einen neuen Akteur einzugeben.


4.4.2. Anwendungsfälle

Das Vorgehen, um Anwendungsfälle hinzuzufügen, ist das gleiche wie für das Hinzufügen von Akteuren, allerdings indem das Anwendungsfall-Symbol im Bearbeitenfenster verwendet wird ( ).

Standardmäßig zeigen Anwendungsfälle in ArgoUML nicht ihre Erweiterungspunkte an (wird verwendet, um Beziehungen anzulegen). Sie können die Erweiterungspunkte auf einen von zwei Wegen anzeigen:

1. Markieren Sie den Anwendungsfall im Bearbeitenfenster mit einem klick der Taste 1 aus, dann markieren Sie das Register Darstellung und klicken auf die Checkbox Erweiterungspunkte.


Der gleiche Ansatz kann für das verstecken des Erweiterungspunktbereiches verwendet werden.

4.4.2.1. Einem Anwendungsfall einen Erweiterungspunkt hinzufügen

Es gibt zwei Wege, einem Anwendungsfall einen Erweiterungspunkt hinzuzufügen.


Anmerkung
Das Symbol Erweiterungspunkt hinzufügen ist inaktiv, bis ein Anwendungsfall markiert ist.


Welche Methode auch immer verwendet wird, der neue Erweiterungspunkt wird markiert und sein Register Eigenschaften kann im Detailfenster angezeigt werden. Der Name und der Ort des Erweiterungspunktes sind Freitext, der die entsprechenden Felder des Registers Eigenschaften setzt.

Ein existierender Erweiterungspunkt kann in seinem Register Eigenschaften bearbeitet werden. Das Register Eigenschaften kann auf zwei Wegen erreicht werden.


2. Im anderen Fall markieren Sie den Anwendungsfall und sein Register Eigenschaften im Detailfenster. Ein Taste 1-Klick auf den gewünschten Eintrag im Feld Erweiterungspunkte wird das Register Eigenschaften für den Erweiterungspunkt im Detailfenster zur Anzeige bringen.

Die Felder Name und Ort des Erweiterungspunktes können bearbeitet werden.


Nachdem Sie einen Erweiterungspunkt erstellt haben, wird er im Explorer (oberer linker Quadrant des ArgoUML-Fensters) erscheinen. Erweiterungspunkte werden immer als Sub-Baum ihres eigenen Anwendungsfalles dargestellt.

### 4.4.3. Assoziationen

Um im Diagramm einen Anwendungsfall mit einem Akteur zu verbinden, klicken Sie der Taste 1 auf das Assoziationssymbol in der Editierwerkzeugleiste ( ). Auf dem Anwendungsfall halten Sie die Taste 1 gedrückt, gehen zum Akteur und lassen die Taste 1 los (oder beginnen Sie beim Akteur und enden am Anwendungsfall).

Dadurch wird eine gerade Linie zwischen dem Akteur und dem Anwendungsfall erzeugt. Sie können die


Es ist auch möglich, Assoziationen mit Hilfe der kleinen „Griffe“ hinzuzufügen, die links und rechts eines Anwendungsfall oder eines Akteurs erscheinen, wenn dieser markiert ist und sich die Maus darüber befindet. Das Ziehen des Grifffes von einem Anwendungsfall zu einem Akteur wird eine Assoziation zu diesem Akteur erzeugen (und umgekehrt beim Ziehen eines Grifffes von einem Akteur zu einem Anwendungsfall).


4.4.3.1. Navigation einstellen

Es gibt zwei Wege, die Navigation einer Assoziation einzustellen.


**Anmerkung**


Sie werden sehen, dass es möglich ist, einem Assoziationsende im Eigenschaftsregister einen Namen zu geben. Dieser Name wird an dem Ende der Assoziation erscheinen und kann dazu verwendet werden die Rolle zu beschreiben, die ein Akteur oder ein Anwendungsfall in dieser Assoziation spielt.

Ein Zeitmanagementsystem für ein Geschäft kann zum Beispiel Anwendungsfälle für die
Vervollständigung der Zeittabellen und für das Abzeichnen der Zeittabellen aufweisen. Ein Akteur Mitarbeiter kann in beides involviert sein. Einmal als Mitarbeiter und zum anderen in der Rolle als Manager.

### 4.4.3.2. Die Kardinalität einstellen

Es gibt zwei Wege, die Kardinalität am Ende einer Assoziation einzustellen.

1. Der Taste 2-Klick über dem Ende einer Assoziation verursacht das Erscheinen eines kontextsensitiven Popup-Menüs mit einem Untermenü Kardinalität. Dieses erlaubt es Ihnen, auszuwählen aus 1 (dem Standard), 0..1, 0..* und 1..*.

2. Bringen Sie das Eigenschaftsregister für das Assoziationsende wie in "Navigation einstellen" nach oben (siehe zweite Option in Abschnitt 4.4.3.1, „Navigation einstellen“). Ein Dropdown-Menü gibt Ihnen die Kardinalitätsoptionen aus, die ausgewählt werden können.

Der zweite dieser beiden Ansätze hat mehr Optionen, obgleich ArgoUML es dem Anwender aktuell nicht erlaubt eine beliebige Kardinalität einzustellen.

### 4.4.4. Hierarchische Anwendungsfälle


#### 4.4.4.1. Includes

Die Prozedur zum Hinzufügen einer Include-Beziehung ist die gleiche, wie für das Hinzufügen einer Assoziation. Allerdings verwenden Sie das Include-Symbol aus der Editier-Werkzeugleiste ( ) um die beiden Anwendungsfälle zu verbinden.

Seit Include-Beziehungen richtungsgebunden sind, ist die Reihenfolge der die beiden Enden markiert werden wichtig. Der einbindende (Haupt-) Anwendungsfall sollte zuerst (Taste 1 drücken) und der eingebundene (sekundäre) Anwendungsfall als zweiter (Taste 1 loslassen) markiert werden.

Es ist möglich die Include-Beziehungen mit Hilfe des Eigenschaftsregisters zu benennen. Dies wird aber selten getan und wird im Anwendungsfalldiagramm nicht dargestellt.

#### 4.4.4.2. Extends

Die Prozedur zum Hinzufügen einer Extend-Beziehung ist die gleiche, wie für das Hinzufügen einer Include-Beziehung. Allerdings verwenden Sie das Extend-Symbol aus der Editier-Werkzeugleiste ( ) um die beiden Anwendungsfälle zu verbinden.

Wie bei den Include-Beziehungen ist die Reihenfolge des Markierens wichtig. In diesem Fall sollte der erweiterte (sekundäre) Anwendungsfall zuerst (Taste 1 drücken) und dann der erweiterte (Haupt-) zweite Anwendungsfall markiert werden (Taste 1 loslassen).

**Anmerkung**

Dies ist gegenüber der Include-Beziehung genau umgekehrt, reflektiert aber die Art und
Weise, wie Designer denken. Die Tatsache, dass der Pfeil des Extend-Symboles nach oben zeigt (im Gegensatz zum Include-Symbol) sollte Ihnen helfen, sich daran zu erinnern.

Um eine Bedingung für die Extend-Beziehung einzugeben, markieren Sie die Extend-Beziehung im Editierfenster (Taste 1-Klick) und öffnen Sie das Eigenschaftsregister im Detailfenster (Taste 1-Klick auf das Register). Der Text der Bedingung kann in das Bedingung-Feld eingefügt werden. Lange Bedingungen sollten - sofern gewünscht - über mehrere Zeilen aufgeteilt werden. Die Bedingung wird unterhalb der «extend»-Kennzeichnung im Diagramm dargestellt.

Es ist möglich die Extend-Beziehungen mit Hilfe des Eigenschaftsregisters zu benennen. Dies wird aber selten getan und wird im Anwendungsfalldiagramm nicht dargestellt.

4.4.4.3. Generalisierung


Seit die Generalisierung eine gerichtete Beziehung ist, ist die Reihenfolge des Markierens wichtig. Der spezialisierte Anwendungsfall sollte zuerst (Taste 1 drücken) und der generalisierte als zweites markiert werden (Taste 1 loslassen).


Ähnlich ist es beim Ziehen des unteren Grifffes. Dies erzeugt eine Generalisierung bei der der Original-Anwendungsfall das generalisierte Ende darstellt.


Es ist möglich die Generalisierungs-Beziehungen mit Hilfe des Eigenschaftsregisters zu benennen. Wird ein Name eingegeben, wird dieser im Anwendungsfalldiagramm dargestellt.

4.4.5. Stereotypen

Die UML enthält das Konzept der Schablonen, um die Basisnotation zu erweitern. Es mag zum Beispiel nützlich erscheinen, ein Problem sowohl auf Geschäftsebene als auch auf der Ingenieur-Ebene zu modellieren. In diesem Fall unterscheidet die OMG zwischen einem PIM und einem PSM. Für beide werden wir Anwendungsfälle benötigen, aber die Anwendungsfälle auf der Geschäftsebene enthalten eine andere Art von Informationen als die auf der Ingenieurs-Ebene. Sie nutzen sehr wahrscheinlich eine andere Sprache und Notation in ihren darunter liegenden Anwendungsfallspezifikationen.

Sie werden sehen, dass ArgoUML in jedem Eigenschaftsregister eine Drop-Down-Auswahl **Stereotypen** aufweist. Dieses ist mit den Standard-Stereotypen gefüllt, zu denen Sie Ihre eigenen hinzufügen können.

Die Details der Schablonen liegt ausserhalb des Bezugsbereiches dieses Tutorials. Das Referenzhandbuch (siehe Abschnitt 16.6, „Stereotyp“) dokumentiert die von ArgoUML bereitgestellte Unterstützung.

**Warnung**


### 4.4.6. Dokumentation

ArgoUML enthält einige einfache Dokumentationsmöglichkeiten, die mit den Modellelementen im Diagramm verknüpft sind. Hauptsächlich sollten diese nur zum Aufzeichnen der Ablageorte der Dokumente verwendet werden, nicht für die aktuelle Dokumentation selbst.

Die Dokumentation für ein bestimmtes Modellelement wird im Dokumentationsregister im Detailfenster aufgezeichnet (der unten rechts befindliche Quadrant des Bildschirms).

Zusätzlich können in den Diagrammen Kommentare mit Hilfe des Text-Symboles der Editier-Werkzeugleiste hinzugefügt werden (A).

Es wird empfohlen, dass ein Anwendungsfalldiagramm das Dokumentationsregister des Akteurs nutzen sollte, um die Informationen über den Akteur aufzuzeichnen. Oder, sollte der Akteur sehr komplex sein, sich auf ein separates Dokument beziehen, welches die Information über den Akteur beinhaltet.

Das Dokumentationsregister des Anwendungsfalles sollte den Ablageort der Anwendungsfalldokumentation aufzeichnen. Die Information in der Anwendungsfallspezifikation (für alle, auch den einfachsten Anwendungsfällen) ist zu komplex, als dass sie direkt in das Register eingegeben werden könnte.

Das Projekt sollte auch ein separates Visionsdokument haben und eine ergänzende Anforderungsspezifikation. Ein Kommentar in Diagrammen kann dazu verwendet werden, sich auf diese zu beziehen, wenn der Anwender dies nützlich findet.

**Warnung**

Das Dokumentationsregister enthält ein Kästchen **Veraltet**. Der Status dieses Kennzeichens bleibt in der aktuellen Release von ArgoUML beim Speichern und Laden nicht erhalten.

### 4.4.7. Systemgrenzen

ArgoUML enthält eine Reihe von Werkzeugen, um beliebige grafische Kommentare in Diagramme einzufügen (wir lernten das Text-Tool bereits kennen). Diese finden Sie auf der rechten Seite der Editier-Werkzeugleiste und sind im Referenzhandbuch vollständig dokumentiert (siehe Kapitel 12, *Das Editierfenster*).

Es gibt jedoch keinen Weg, die Füllfarbe zu ändern. Sie werden es daher vorziehen, die Systemgrenzen aus vier einzelnen Linien zu zeichnen. Dies ist die Methode, die für die Diagramme in diesem Kapitel verwendet wurde.

**Anmerkung**


### 4.5. Fallstudie

#### 4.5.1. Das Dokument Vision

Ein Visionsdokument enthält mehr als nur die Dinge, die für die Modellierung erforderlich sind. Es enthält auch finanzielle und verwaltungsrelevante Informationen. Die folgenden Abschnitte sind solche Teile eines Visionsdokumentes Abschnitt 4.3.1, „Visions-Dokument“. In der Praxis muss dieses Format nicht sklavisch eingehalten werden, aber es ist hier für die Konsistenz erforderlich.

#### 4.5.1.1. Zusammenfassung

Die Firma möchte eine Serie von Geldautomaten produzieren und vermarkten. Der Zweck dieses Projektes ist es, die Hardware und die Software zu produzieren und zuverlässig und robust zu machen.

#### 4.5.1.2. Ziele

Bessere entworfene Produkte auf Basis neuer Technologien produzieren. Wir folgen der MDA-Philosophie der OMG indem wir zuerst ein plattformunabhängiges Modell (Platform Independent Model (PIM)) erzeugen. Da die aktuelle Modellierungstechnologie die Verwaltung der Integrität der Verbindung zwischen der PIM und den plattformspezifischen Modellen (Platform Specific Model (PSM)) nicht zulassen, wird die PIM vergleichsweise stabil werden bevor die erste Iteration der PSM produziert wird. Die Softwareplattform wird die Java-Technologie sein. Das System wird einen einfachen Userid (von der Geldautomatenkarte) und Kennwort (oder PIN)-Mechanismus verwenden.

#### 4.5.1.3. Der Markt

Das aktuell auf dem Markt vorhandene Equipement basiert auf älterer Technologie bei der Hard- und Software. Diese Technologie hat noch nicht das Ende seiner Lebensdauer erreicht, was es unwahrscheinlich macht, dass die Hersteller dieser Produkte diese in der nahen Zukunft austauschen werden. Auf der anderen Seite ist neuere Technologie verfügbar, die uns einen nennenswerten Vorteil verschafft, wenn wir sie jetzt implementieren.

#### 4.5.1.4. Beteiligte

Zwischen den Beteiligten dieses Systemes befinden sich die Entwicklungsabteilung, die Wartung und der zentrale Computer- Betrieb. Die vollständige Liste der Projektbeteiligten und die spezifischen Personen, die diese repräsentieren sind:

- **Entwicklung**: Bunny, Bugs
- **Wartung**: Hardy, Oliver
4.5.1.5. Die Hauptfunktionen

Geld aufbewahren, Geld abheben und Kontostandsabfragen der Kunden. Kunden sind Personen, die Konten bei ihrer Bank haben aber auch Personen, die Abhebungen von Konten anderer Banken oder von Kreditkarten vornehmen wollen.

Wartung des Equipements durch die Bankingenieure. Diese Aktion kann durch den Ingenieur initiiert auf Basis eines Wartungsplanes werden. Sie kann aber auch durch das Equipement initiiert werden, die den Ingenieur ruft, wenn es einen internen Fehler entdeckt.

Das Herausnehmen der Einlagen und das Aufladen von Geld erfolgt durch Beamte der lokalen Bankfiliale. Diese Aktionen werden entweder auf Basis eines Plans ausgeführt, oder wenn der Zentralcomputer feststellt, dass der Geldvorrat zu gering oder die Einlagenkassette fast voll ist.


4.5.1.6. Randbedingungen


4.5.1.7. Anhang


- Zentralcomputer
- Kunde
- Lokaler Bankbeamter
- Wartungsingenieur

Im Folgenden finden Sie Anwendungsfälle, die diese Vision direkt unterstützen. Zusätzliche Anwendungsfälle können später identifiziert werden, wenn Sie zur Unterstützung dieser Vision oder der Technologie erforderlich sind oder die hier aufgelisteten Anwendungsfälle unterstützen. Sie sollten dieser Liste nicht hinzugefügt werden, es sei denn, sie sind für die direkte Unterstützung der Vision, wie
in diesem Dokument beschrieben, erforderlich.

- Prüfen
- Kunden nutzt den Automaten
- Warten des Automaten

4.5.2. Akteure und Anwendungsfälle identifizieren


In Abbildung 4.4, „Anwendungsfalldiagramm für einen Geldautomaten mit include-Beziehungen.„ sehen wir ein Anwendungsfalldiagramm für einen Geldautomaten, bestehend aus «include»-Beziehungen für die Anwendungsfälle Geldautomat warten und Geldautomat nutzen. Geldautomat warten wird darüber hinaus durch zwei Anwendungsfälle definiert: ”Equipement warten“ und ”Geldautomat neu starten“. Geldautomat nutzen wurde darüber hinaus durch drei einfachere Anwendungsfälle definiert: ”Geld einnehmen“, ”Geld abheben“ und ”Kontostand abfragen“. 

Noch zu beschreiben...

4.5.3. Assoziationen (Noch zu beschreiben)
Noch zu beschreiben...

4.5.4. Erweiterte Diagrammfunktionen (Noch zu beschreiben)
Noch zu beschreiben...

4.5.5. Anwendungsfallspezifikationen (Noch zu beschreiben)
Noch zu beschreiben...

4.5.6. Ergänzende Anforderungsspezifikation (Noch zu beschreiben)
Noch zu beschreiben...
Kapitel 5. Analyse

Die Analyse ist der Prozess, die „Kunden“-Anforderungen zu nehmen und in die Sprache und Perspektive der künftigen Lösung umzuwandeln.

Wir sollten zu diesem Zeitpunkt nicht versuchen die detaillierte Lösung auszuarbeiten. Dies passiert in der Design-Phase (siehe Kapitel 6, Design).

Wie die Grenze zwischen den Anforderungs- und der Analyse-Phase ist die Grenze zwischen Analyse und Design inhärent unscharf. Der Schlüssel dazu ist, dass die Analyse die Lösung nicht weiter als notwendig definieren sollte, um die Anforderungen in der Sprache der Lösung zu spezifizieren. Die Modellelemente in der Analyse repräsentieren generell eine höhere Abstraktionsebene.

Noch einmal, die rekursive, und iterative Natur unserer Prozesse bedeutet, dass wir in der Zukunft noch häufig auf die Analysephase zurückkommen werden.

5.1. Der Analyseprozess

Es gibt drei gedankliche Ausrichtungen darüber, wie man sich der Analyse annähern sollte. Die Ontologen definieren die Daten (aktuell die Metadaten) zuerst und plagen sich später mit den Prozessen. Der wahre Ontologe würde es vorziehen, überhaupt nicht an Prozesse denken zu müssen. Der Phänomenalismus kehrt dies um und stellt die Prozesse über die Daten. Der Paradigmatiker betrachtet die Prozesse und Daten gleichwertig wichtig und adressiert beide von Beginn an.

Wenn es dazu kommt, Purist zu sein, dann hat der Ontologe die Oberhand. Es ist möglich, eine Datenbank zu definieren und zu erzeugen, in die Daten eingegeben und geholt werden kann, ohne Rücksicht darauf, was mit ihnen getan wird oder nicht. Auf der anderen Seite ist es nicht besonders nützlich einen Prozess zu implementieren ohne irgendwelche Datenstrukturen zu haben auf denen der Prozess arbeitet.

5.1.1. Klasse-, Verantwortlichkeits- und Zusammenarbeits-Karten (CRC)

Die CRC-Methode (CRC = Class, Responsibility, Collaborators) favorisiert die Vorliebe der Phänomenologen für die Analyse. Es ist äquivalent mit den Anwendungsfällen zu beginnen, die Aspekte der Prozesse (Operationen) in Klassendiagrammen und Szenarien darzustellen aus denen Sequenzdiagramme initiert werden können.


Die Stärke der CRC-Karten während der Analyse.

- Gemeinsames Projektvokabular -
- Breites Bereichswissen -
- Führt Paradigmenwechsel durch -
- Live-Prototyping -
- Indentifiziert Lücken in den Anforderungen -
In dieser Phase sollte die Gruppe aus zwei oder drei Fachexperten bestehen. Einer als Vermittler der objektorientierten Technologie und der Rest der Gruppe sollte aus Personen bestehen, die für das Ausrollen des Systems verantwortlich sind.


Nachdem ein vernünftiger Satz von Klassen durch die Gruppe definiert wurde, können die Verantwortlichkeiten hinzugefügt werden. Fügen Sie Verantwortlichkeiten hinzu, die aus den Anforderungen oder den Namen der Klassen deutlich hervorgehen. Sie müssen nicht alle finden müssen (oder in jedem Fall alle). Die Szenarien werden dies deutlicher machen. Der Vorteil, einige bereits am Anfang zu finden ist, dass es hilft, einen Anfangspunkt zu finden.

Wählen Sie die ersten Szenarios aus dem Anforderungsdokument aus, indem Sie dessen Verben auf die gleiche Weise prüfen, wie wir vorher die Hauptwörter durchgingen. Dann führen Sie diesen Prozess so oft als notwendig durch, um die begonnene Analysephase zu vervollständigen.

Wann ist genug Analyse durchgeführt worden und wann kann das Design beginnen? Wenn alle unterschiedlichen Verantwortlichkeiten zugeordnet wurden und das System stabil geworden ist. Nachdem das gesamte normale Verhalten abgedeckt wurde, ist es notwendig aussergewöhnliches Verhalten zu simulieren. Wenn Sie feststellen, dass die Verantwortlichkeiten alle an der richtigen Stelle sind, um die neuen Szenarien zu unterstützen und nur noch wenige Änderungen an den Karten vorzunehmen sind, dann ist das ein Anzeichen dafür, dass Sie mit dem Design beginnen können.

5.1.2. Konzeptdiagramm (Noch zu beschreiben)

5.1.3. System-Sequenzdiagramm (Noch zu beschreiben)

5.1.4. System-Zustandsdiagramm (Noch zu beschreiben)

5.1.5. Anwendungsfalldiagramm realisieren (Noch zu beschreiben)

5.1.6. Dokumente (Noch zu beschreiben)

5.2. Klassendiagramme (Noch zu beschreiben)

5.2.1. Das Klassendiagramm (Noch zu beschreiben)
5.2.2. Erweiterte Klassendiagramme (Noch zu beschreiben)

5.2.2.1. Assoziationsklassen (Noch zu beschreiben)

5.3. Klassendiagramme in ArgoUML erzeugen

5.3.1. Klassen

Klassendiagramme aus existierendem Material (Vision, Anwendungsfälle, usw.) identifizieren. Noch zu beschreiben...

5.3.1.1. Das Kommentarsymbol in der Symbolleiste verwenden

Klicken Sie auf Ihre Zielklasse. Dann klicken Sie auf das Kommentarsymbol. ArgoUML wird die Verknüpfung automatisch generieren.

Sie können auch einen Rechtsklick ausführen, um einen Kommentar hinzuzufügen! Beachten Sie, dass Sie eine unbegrenzte Anzahl von Kommentaren zu jeder Klasse hinzufügen können!

**Warnung**

Beachten Sie, dass Ihr Kommentar nicht im Sourcecoderegister erscheint.

5.3.2. Assoziationen (Noch zu beschreiben)

5.3.2.1. Aggregation (Noch zu beschreiben)

5.3.3. Klassenattribute und Operationen (Noch zu beschreiben)

5.3.3.1. Daten in Attribut- und Methodenfenster eingeben

Klicken Sie direkt in das Klassenelement und beginnen Sie mit der Eingabe. Verwenden Sie nicht die Dialogfenster des Eigenschaftsregisters; sie sind noch nicht vollständig implementiert und führen nur zu ein bisschen Frustration.

Natürlich wäre es von Interessse, wenn Sie Stereotypen rechts in den Bereich für Klassenattribute schreiben könnten, um XML-Diagramme zu generieren.

5.3.3.2. Klassenattribute (Noch zu beschreiben)

5.3.3.3. Klassenoperationen (Noch zu beschreiben)
5.3.4. Erweiterte Klasseneigenschaften (Noch zu beschreiben)

5.3.4.1. Assoziationsklassen (Noch zu beschreiben)

5.3.4.2. Stereotypen (Noch zu beschreiben)

5.4. Sequenzdiagramme (Noch zu beschreiben)

5.4.1. Das Sequenzdiagramm (Noch zu beschreiben)

5.4.2. Aktionen identifizieren (Noch zu beschreiben)

5.4.3. Erweiterte Sequenzdiagramme (Noch zu beschreiben)

5.5. Sequenzdiagramme in ArgoUML erzeugen

5.5.1. Sequenzdiagramme

5.5.1.1. Ein Sequenzdiagramm erzeugen

Normalerweise können Sie mit einem Sequenzdiagramm sofort beginnen. Im Menü Neues Diagramm wählen Sie aus Sequenzdiagramm.

5.5.2. Aktionen (Noch zu beschreiben)

5.5.3. Erweiterte Sequenzdiagramme (Noch zu beschreiben)

5.6. Zustandsdiagramme (Noch zu beschreiben)

5.6.1. Das Zustandsdiagramm (Noch zu beschreiben)
Die Zustandsdiagrammarten (Moore, Mealy); Hierarchische Diagramme. Noch zu beschreiben...

5.6.2. Erweiterte Zustandsdiagramme (Noch zu beschreiben)
   Noch zu beschreiben...

5.6.2.1. Hierarchical Statechart Diagrams (To be written)
   Noch zu beschreiben...

5.7. Zustandsdiagramme in ArgoUML erstellen

5.7.1. Zustandsdiagramme (Noch zu beschreiben)
   Noch zu beschreiben...

5.7.1.1. Ein Zustandsdiagramm erstellen
   Markieren Sie eine Klasse, dann können Sie ein Zustandsdiagramm erstellen.

5.7.2. Zustände (Noch zu beschreiben)
   Noch zu beschreiben...

5.7.2.1. Einen zusammengesetzten Zustand editieren
   Wenn Sie einen zusammengesetzten Zustands editieren, wie erhalten Sie Zugriff auf den zusammengesetzten Zustand?
   Die Antwort ist, die Klasse markieren und dann das Zustandsdiagramm erstellen.

5.7.3. Transitionen (Noch zu beschreiben)
   Noch zu beschreiben...

5.7.4. Aktionen (Noch zu beschreiben)
   Noch zu beschreiben...

5.7.5. Erweiterte Zustandsdiagramme (Noch zu beschreiben)
   Noch zu beschreiben...

5.7.5.1. Hierarchische Zustandsdiagramme (Noch zu beschreiben)
   Noch zu beschreiben...

5.8. Anwendungsfälle realisieren (Noch zu beschreiben)
   Noch zu beschreiben...

5.9. Realisierungs-Anwendungsfälle in
ArgoUML erstellen (Noch zu beschreiben)

5.10. Fallstudie (Noch zu beschreiben)

Abhängig davon, welche Methode Sie verwenden, ist es an der Zeit, dass Sie ohne jeden Zweifel die Problemstellung aus Abschnitt 4.5, „Fallstudie“ nehmen und die Hauptwörter extrahieren. Diese Liste sollte verdichtet werden, so dass nur noch die Hauptwörter enthalten sind, die als Klasse erwartet werden. Dieser Ansatz hat folgendes Ergebnis:

- Konto
- Nachweislog
- Bank
- Geld
- Kunde

5.10.1. CRC Karten

Der Projektmanager beruft eine CRC-Sitzung ein, in der die ersten Klassen definiert werden. Der Multiplikator erinnert die Teilnehmer daran, dass wir uns in der Analysephase befinden und nur an den Dingen interessiert sind, welche Bedürfnisse erfüllen müssen (auf Geschäftsebene) und alles weglassen müssen, was nach "wie müssen wir es tun" aussieht. Als generelle Regel dieses Ansatzes bedeutet das, eine Teilmenge der Hauptwörter aus dem Problembereich (siehe oben). Die Gruppe beginnt mit einer vollständigen Liste aller Hauptwörter der Beschreibung, prüft jedes und entscheidet, welche unpassend sind und aus der Liste gestrichen werden. Jede Klasse wird dann einem der Teilnehmer zugewiesen.

ist fortzusetzen......

5.10.2. Klassendiagramme konzipieren (Noch zu beschreiben)

5.10.2.1. Klassen identifizieren (Noch zu beschreiben)

5.10.2.2. Assoziationen identifizieren (Noch zu beschreiben)

5.10.3. System-Sequenzdiagramme (Noch zu beschreiben)

5.10.3.1. Aktionen identifizieren (Noch zu beschreiben)
5.10.4. System-Zustandsdiagramme (Noch zu beschreiben)
Noch zu beschreiben...

5.10.5. Die Realisierung von Anwendungsfällen (Noch zu beschreiben)
Noch zu beschreiben...
Kapitel 6. Design

Wir haben jetzt das Problem, das wir in der Sprache der vermeintlichen Lösung zu lösen versuchen. In der Designphase konstruieren wir alle Details dieser Lösung.


Erneut bedeutet die rekursive und iterative Natur unseres Prozesses, dass wir in Zukunft viele Male zurück in die Designphase kommen.

6.1. Der Designprozess (Noch zu beschreiben)


6.1.1. Klasse, Verantwortlichkeits- und Zusammenhänge-(CRC) Karten

Die Stärken der CRC-Karten während des Design

• Verbreitern der objektorientierten Design-Expertise
• Design Reviews
• Framework die Implementierung
• Informelle Notation
• Auswahl der unterstützenden Softwarekomponenten
• Performance-Anforderungens

In dieser Phase ersetzen die Entwickler einige der fachlichen Experten in der Gruppe. Es sollte aber immer mindestens ein fachlicher Experte in der Gruppe verbleiben.


Während der Designphase wird der Unterschied zwischen Klasse und Objekt wichtig. Denken Sie über die Objekte in Ihren Szenarien nach. Wer erzeugt die Objekte? Was passiert, wenn sie erzeugt und
gelöscht werden? Wie ist die Lebensdauer des Objektes im Gegensatz zur Lebensdauer der Information, die durch das Objekt gehalten wird.

Jetzt ist es an der Zeit sich anzusehen, welche Informationen die Objekte halten, verglichen mit den von anderen Klassen angeforderten oder berechneten Informationen. Benutzen Sie die Rückseite der Karte, um die gefundenen Attribute für diese Klassen aufzuschreiben. Teilen Sie die Verantwortlichkeiten in Sub-Verantwortlichkeiten auf und listen Sie die Sub-Verantwortlichkeiten unter der Hauptverantwortlichkeit eingerückt auf. Verschieben Sie die dafür notwendigen Klassen zu den Verantwortlichkeiten die sie nutzen.

Hinter der Kollaboratorklasse listen Sie auf Ihrer Karte die Verantwortlichkeit der in dieser Zusammenarbeit verwendeten Klasse auf. Hinter den kollaborierenden Verantwortlichkeiten Ihrer Karte listen Sie die durch die kollaborierenden Objekte zurückgelieferten Daten in Klammern auf.

Spielen Sie die Szenarien der Analysephase erneut durch, beachten Sie dabei aber alle Erwägungen der diskutierten Design-Heuristiken. Nehmen Sie Ihre eigenen Szenarien und versuchen Sie es.

6.1.2. Paketdiagramm (Noch zu beschreiben)

6.1.3. Klassendiagramme realisieren (Noch zu beschreiben)

6.1.4. Sequenz- und Kollaborationsdiagramme (Noch zu beschreiben)

6.1.5. Zustands- und Aktivitätsdiagramme (Noch zu beschreiben)

6.1.6. Verteilungsdiagramme (Noch zu beschreiben)

6.1.7. Dokumente (Noch zu beschreiben)

6.2. Paketdiagramme (Noch zu beschreiben)

6.2.1. Das Paketdiagramm (Noch zu beschreiben)

6.2.2. Erweiterte Paketdiagramme (Noch zu beschreiben)

6.2.2.1. Subpakete (Noch zu beschreiben)
6.2.2.2. Datentypen hinzufügen (Noch zu beschreiben)
  Noch zu beschreiben...

6.2.2.3. Stereotypen hinzufügen (Noch zu beschreiben)
  Noch zu beschreiben...

6.3. Paketdiagramme in ArgoUML erstellen

6.3.1. Pakete
  Ausarbeiten, was in Pakete kommt. Noch zu beschreiben...

6.3.1.1. Subpakete (Noch zu beschreiben)
  Noch zu beschreiben...

6.3.2. Beziehungen zwischen Paketen (Noch zu beschreiben)
  Noch zu beschreiben...

6.3.2.1. Abhängigkeit (Noch zu beschreiben)
  Noch zu beschreiben...

6.3.2.2. Generalisierung (Noch zu beschreiben)
  Noch zu beschreiben...

6.3.2.3. Realisierung und Abstraktion (Noch zu beschreiben)
  Noch zu beschreiben...

6.3.3. Erweiterte Paketfunktionen (Noch zu beschreiben)
  Noch zu beschreiben...

6.3.3.1. Neue Datentypen erstellen (Noch zu beschreiben)
  Noch zu beschreiben...

6.3.3.2. Neue Stereotypen erstellen (Noch zu beschreiben)
  Noch zu beschreiben...

6.4. Mehr über Klassendiagramme (Noch zu beschreiben)
  Noch zu beschreiben...

6.4.1. Das Klassendiagramm (Noch zu beschreiben)
  Noch zu beschreiben...

6.4.1.1. Klassenattribute (Noch zu beschreiben)
  Noch zu beschreiben...

6.4.1.2. Klassenoperationen (Noch zu beschreiben)
6.4.2. Erweiterte Klassendiagramme (Noch zu beschreiben)

Noch zu beschreiben...

6.4.2.1. Realisierung und Abstraktion (Noch zu beschreiben)

Noch zu beschreiben...

6.5. Mehr über Klassendiagramme in ArgoUML (Noch zu beschreiben)

6.5.1. Klassen (Noch zu beschreiben)

Mehr über das Identifizieren von Klassen aus existierendem Material und der Gebrauch von Stereotypen. Noch zu beschreiben...

6.5.2. Klassenattribute und -operationen (Noch zu beschreiben)

Noch zu beschreiben...

6.5.2.1. Klassenattribute (Noch zu beschreiben)

Noch zu beschreiben...

6.5.2.2. Klassenoperationen (Noch zu beschreiben)

Noch zu beschreiben...

6.5.3. Erweiterte Klassenfunktionen

6.5.3.1. Operationen bei Schnittstellen

6.5.3.1.1. Schnittstellen, die Schnittstellen erweitern

Durch einfaches Klicken auf das Schnittstellensymbol in der Werkzeugleiste und anschliessendem klicken in das Diagrammfenster fügen Sie dem aktuellen Klassendiagramm eine unbenannte Schnittstelle hinzu (siehe Abbildung 6.1, „Auswählen des Werkzeuges Schnittstelle“).

Abbildung 6.1. Auswählen des Werkzeuges Schnittstelle
Dann führen Sie einen Doppelklick auf das Namensfeld der Schnittstelle aus, um dessen Namen wie im Bild Abbildung 6.2, „Modellelement Schnittstelle im Klassendiagramm“ gezeigt zu ändern.

Abbildung 6.2. Modellelement Schnittstelle im Klassendiagramm

Geben Sie den Namen ein (z.B. TestSchnittstelle in diesem Fall). Drücken Sie „Enter“, wenn der Name vollständig ist. (Sie können den Namen auch ändern, indem Sie in das Eigenschaftsregister im Detailfenster nach dem Hinzufügen der Schnittstelle gehen.)


Abbildung 6.3. Generalisierung über die Symbolleiste des Klassendiagrammes
Bewegen Sie den Mauszeiger auf die Subschnittstelle, drücken Sie die linke Maustaste und ziehen Sie die Generalisierung auf die Superschnittstelle, indem Sie die Maustaste loslassen. Bild Abbildung 6.4, „Generalisierung zwischen zwei Schnittstellen.“ zeigt, wie Ihr Diagramm jetzt aussehen sollte.

**Abbildung 6.4. Generalisierung zwischen zwei Schnittstellen.**

Durch klicken auf die Subschnittstellen und das Sourceregister und anschliessende Auswahl der Javanotation für das Sourceregister können Sie sehen, dass die Schnittstelle nun seine Superschnittstelle erweitert.

6.5.3.2. Stereotypen (Noch zu beschreiben)

Noch zu beschreiben...

6.6. Sequenz- und Kollaborationsdiagramme
(Noch zu beschreiben)
Anmerkung
Sequenzdiagramme funktionieren in der ArgoUML-Version 0.14 nicht.

Noch zu beschreiben...

6.6.1. Mehr über das Sequenzdiagramm (Noch zu beschreiben)
Noch zu beschreiben...

6.6.2. Das Kollaborationsdiagramm (Noch zu beschreiben)
Noch zu beschreiben...

6.6.2.1. Nachrichten (Noch zu beschreiben)
Noch zu beschreiben...

6.6.2.2. Aktionen (Noch zu beschreiben)
Noch zu beschreiben...

6.6.3. Erweiterte Kollaborationsdiagramme (Noch zu beschreiben)
Noch zu beschreiben...

6.7. Kollaborationsdiagramme in ArgoUML erstellen (Noch zu beschreiben)

6.7.1. Kollaborationsdiagramme (Noch zu beschreiben)
Noch zu beschreiben...

6.7.2. Nachrichten (Noch zu beschreiben)
Noch zu beschreiben...

6.7.2.1. Aktionen (Noch zu beschreiben)
Noch zu beschreiben...

6.7.3. Erweiterte Kollaborationsdiagramme (Noch zu beschreiben)
Noch zu beschreiben...

6.8. Zustandsdiagramme (Noch zu beschreiben)
Noch zu beschreiben...
6.8.1. Das Zustandsdiagramm (Noch zu beschreiben)
Mehr darüber. Noch zu beschreiben...

6.8.2. Erweiterte Zustandsdiagramme (Noch zu beschreiben)
Noch zu beschreiben...

6.8.2.1. Aktionen (Noch zu beschreiben)
Noch zu beschreiben...

6.8.2.2. Transitionen (Noch zu beschreiben)
Noch zu beschreiben...

6.8.2.2.1. Trigger (Noch zu beschreiben)
Noch zu beschreiben...

6.8.2.2.2. Wächter (Noch zu beschreiben)
Noch zu beschreiben...

6.8.2.2.3. Effekte (Noch zu beschreiben)
Noch zu beschreiben...

6.8.2.3. Pseudo-Zustände (Noch zu beschreiben)
Noch zu beschreiben...

6.8.2.3.1. Junction and Choice (To be written)
Noch zu beschreiben...

6.8.2.3.2. Verzweigen und Verknüpfen (Noch zu beschreiben)
Noch zu beschreiben...

6.8.2.4. Hierarchische Zustandsautomaten (Noch zu beschreiben)
Noch zu beschreiben...

6.8.2.5. Modelle für die Zustandshistorie (Noch zu beschreiben)
Breite versus Tiefe. Noch zu beschreiben...

6.9. Zustandsdiagramme in ArgoUML erstellen (Noch zu beschreiben)

6.9.1. Zustandsdiagramme (Noch zu beschreiben)
Noch zu beschreiben...

6.9.2. Zustände (Noch zu beschreiben)
Noch zu beschreiben...

6.9.3. Transitionen (Noch zu beschreiben)
Noch zu beschreiben...
6.9.4. Aktionen (Noch zu beschreiben)
   Noch zu beschreiben...

6.9.5. Erweiterte Zustandsdiagramme (Noch zu beschreiben)
   Noch zu beschreiben...

6.9.5.1. Transitionen (Noch zu beschreiben)
   Noch zu beschreiben...

6.9.5.1.1. Trigger (Noch zu beschreiben)
   Noch zu beschreiben...

6.9.5.1.2. Wächter (Noch zu beschreiben)
   Noch zu beschreiben...

6.9.5.1.3. Effekte (Noch zu beschreiben)
   Noch zu beschreiben...

6.9.5.2. Pseudozustände (Noch zu beschreiben)
   Noch zu beschreiben...

6.9.5.2.1. Junction and Choice (To be written)
   Noch zu beschreiben...

6.9.5.2.2. Verzweigen und Verknüpfen (Noch zu beschreiben)
   Noch zu beschreiben...

6.9.5.3. Hierarchische Zustandsautomaten (Noch zu beschreiben)
   Noch zu beschreiben...

6.9.5.4. Historie (Noch zu beschreiben)
   Breite versus Tiefe. Noch zu beschreiben...

6.10. Aktivitätsdiagramme (Noch zu beschreiben)
   Noch zu beschreiben...

6.10.1. Das Aktivitätsdiagramm (Noch zu beschreiben)
   Mehr darüber. Noch zu beschreiben...

6.10.1.1. Aktionszustände (Noch zu beschreiben)
   Noch zu beschreiben...

6.11. Aktivitätsdiagramme in ArgoUML erstellen (Noch zu beschreiben)

6.11.1. Aktivitätsdiagramme (Noch zu beschreiben)
Noch zu beschreiben...

6.11.1.1. Ein Aktivitätsdiagramm erstellen
Markieren Sie einen Anwendungsfall oder eine Klasse, dann können Sie ein Aktivitätsdiagramm erstellen.

6.11.2. Aktionszustände (Noch zu beschreiben)
Noch zu beschreiben...

6.12. Verteilungsdiagramme (Noch zu beschreiben)
Noch zu beschreiben...

6.12.1. Das Verteilungsdiagramm (Noch zu beschreiben)
Noch zu beschreiben...

6.13. Verteilungsdiagramme in ArgoUML erstellen (Noch zu beschreiben)

6.13.1. Knoten (Noch zu beschreiben)
Noch zu beschreiben...

6.13.1.1. Knoten-Instanzen (Noch zu beschreiben)
Noch zu beschreiben...

6.13.2. Komponenten (Noch zu beschreiben)
Noch zu beschreiben...

6.13.2.1. Komponenten-Instanzen (Noch zu beschreiben)
Noch zu beschreiben...

6.13.3. Beziehungen zwischen Knoten und Komponenten (Noch zu beschreiben)
Noch zu beschreiben...

6.13.3.1. Abhängigkeit (Noch zu beschreiben)
Noch zu beschreiben...

6.13.3.2. Assoziationen (Noch zu beschreiben)
Noch zu beschreiben...

6.13.3.3. Verknüpfungen (Noch zu beschreiben)
Noch zu beschreiben...

beschreiben)
   Noch zu beschreiben...

6.15. Fallstudie (Noch zu beschreiben)

6.15.1. CRC-Karten (Noch zu beschreiben)
   Noch zu beschreiben...

6.15.2. Pakete (Noch zu beschreiben)
   Noch zu beschreiben...
   6.15.2.1. Pakete identifzieren (Noch zu beschreiben)
      Noch zu beschreiben...
   6.15.2.2. Datentypen und Stereotypen (Noch zu beschreiben)
      Noch zu beschreiben...

6.15.3. Klassendiagramme (Noch zu beschreiben)
   Noch zu beschreiben...
   6.15.3.1. Klassen identifizieren (Noch zu beschreiben)
      Noch zu beschreiben...
   6.15.3.2. Assoziationen identifizieren (Noch zu beschreiben)
      Noch zu beschreiben...
   6.15.3.3. Attribute und Operationen spezifizieren (Noch zu beschreiben)
      Noch zu beschreiben...

6.15.4. Sequenzdiagramme (Noch zu beschreiben)
   Noch zu beschreiben...
   6.15.4.1. Aktionen identifzieren (Noch zu beschreiben)
      Noch zu beschreiben...

6.15.5. Kollaborationsdiagramme (Noch zu beschreiben)
   Noch zu beschreiben...
   6.15.5.1. Nachrichten identifizieren (Noch zu beschreiben)
      Noch zu beschreiben...

6.15.6. Zustandsdiagramme (Noch zu beschreiben)
   Noch zu beschreiben...

6.15.7. Aktivitätsdiagramme (Noch zu beschreiben)
   Noch zu beschreiben...
6.15.8. Das Verteilungsdiagramm (Noch zu beschreiben)
   Noch zu beschreiben...

6.15.9. Die System-Architektur (Noch zu beschreiben)
   Noch zu beschreiben...
Kapitel 7. Codegenerierung, Reverse Engineering und Round Trip Engineering

7.1. Einleitung


Das Ergebnis dieses Prozesses ist ein Satz von Dateien, der das Programm bildet, welches das Problem löst.

Nochmals, die rekursive und iterative Natur unseres Prozesses bedeutet, dass wir in der Zukunft noch viele Male in die Buildphase zurückkehren werden.

Es gibt dazu auch noch eine andere Seite und das ist die Reverse Engineering-Seite. Wenn wir ein altes Programm haben, dass wir vielleicht überprüfen wollen, dann können wir die Dateien nehmen und daraus mit Hilfe des Reverse Engineering ein Design erstellen. Das kann eingesetzt werden, wenn wir versuchen wollen, ein nicht so gut dokumentiertes Programm verstehen zu wollen oder als Schnelleinstieg in die Designarbeit.


7.2. Codegenerierung

Das Ergebnis der Codegenerierung ist das vollständige Programm. Je nach Inhalt des Designs, können wir auch Unit-Testfälle generieren.

Um dies tun zu können, benötigen wir das Designmodell, welches die statischen und dynamischen Aspekte des Programmes beinhaltet.

7.2.1. Code aus der statischen Struktur generieren

Es ist ziemlich unkompliziert, diese Generierung durchzuführen. Zumindestens so lange wir es für eine objektorientierte Sprache tun. Dies sind einige der grundlegenden Regeln:

- Eine Klasse wird eine Klasse.
  In einigen Zielsprachen (wie Java, C++) werden sie auch Dateien und Übersetzungseinheiten.
- Eine Generalisierung wird zu einer Vererbung.
  Wenn die Zielsprache keine Vererbung unterstützt und wir dies während des Design's nicht
adressieren, sind einige spezielle Konvertierungen erforderlich, um dieses Problem zu lösen.

- Ein Attribut wird zu einer Membervariable.
- Eine navigierbare Assoziation wird eine Membervariable.
  Abhängig von der Zielsprache, der Zielplattform und der Kardinalitäten der Assoziation wird dies ein Zeiger, eine Referenz, eine Collection-Klasse, ein Eintrag in einigen Tabellen oder Bitabbildungen (map) sein.
- Eine nicht abstrakte Operation in einer Klasse wird zu einer Methode.
- Eine abstrakte Operation in einer Klasse wird zu einer abstrakten Methode.
- Ein Parameter in einer Operation wird zu einem Parameter in der Methode.
  Bei einfachen Typen (int, boolean) ist dies der Normalfall. In C++ werden diese wahrscheinlich Const-Klassen. In Java kann dies für Klassen nicht eingefordert werden.
- Ein out- oder in/out-Parameter in einer Operation wird zu einem referenzierenden Parameter in der Methode.
- Die Sichtbarkeit von Attributen, Assoziationen und Operationen werden zu Sichtbarkeit von Membervariablen oder Methoden.
- Pakete werden Verzeichnisse, Namensräume oder beides.

### 7.2.2. Code aus Interaktionen und Zustandsautomaten generieren

Diese Konvertierung ist nicht so unkompliziert, wie die Konvertierung der statischen Struktur. Sie hängt sehr viel mehr von der Zielsprache und der Zielplattform ab.

Generell ist es nur möglich, das Folgende zu Interaktionen zu sagen:

- Eine Nachricht wird in einen Funktionsaufruf konvertiert.
  Die Empfängerklasse wird eine Funktion mit dem richtigen Namen und der richtigen Signatur haben.
  Die sendende Funktion in der Klasse eines Senders wird einen Funktionsaufruf auf eine Funktion des Empfängers aufweisen.
- Eine asynchrone Nachricht wird entweder eine Nachricht senden, die durch einige andere Thread- oder Funktionsaufrufe verarbeitet werden, die einen neuen Thread starten.

Das Folgende beschreibt einen möglichen Weg, Zustandsautomaten zu generieren:

- Ein Zustandsautomat wird in einen Satz von Membervariablen generiert, bei dem sich jede Methode
in dieser Klasse auf das entscheidende Verhalten bezieht.

• Ein Zustand wird zu einem zusammengehörenden Satz von Werten dieser Membervariablen generiert.

• Ein Ereignis wird zu einem Aufruf auf eine Membermethode generiert, der den Zustand ändern kann.

Diese Methoden würden dann typischerweise eine grosse switch-Anweisung haben, aufgesplittet je nach aktuellem Zustand.

• Ein Wächter wird, in dem Zweig des richtigen Zustandes, zu einer if-Anweisung in der Ereignis-Membermethode generiert.

• Eine Transition wird als Zuordnung auf einige Zustandsvariablen generiert.

• Eine Aktion wird als Funktionsaufruf generiert.

7.3. Codegenerierung in ArgoUML

7.3.1. Statische Struktur


7.3.2. Interaktionen und Zustandsdiagramme

Es gibt aktuell keine Unterstützung dafür in ArgoUML, für keine Sprache.

7.4. Reverse Engineering

Reverse Engineering wird in zwei Fällen verwendet:

1. Um die vorher entwickelte Klasse in das aufzubauende Modell zu bekommen.

2. Um eine UML-Darstellung der vorher entwickelten Klassen zu erhalten, damit man verstehen kann, wie diese arbeiten.

Grundsätzlich führt dies eine umgekehrte Codegenerierung aus.

7.5. Round-Trip Engineering

Round-Trip Engineering macht es möglich, die Perspektive während des Entwurfes zu wechseln. Erstellen Sie einige Klassen in einem Klassendiagramm. Schreiben Sie mit Hilfe Ihres favorisierten Editors etwas Code für einige Operationen oder Funktionen. Verschieben Sie die Operationen im Klassendiagramm von einer Klasse in eine andere...

Aktuell wird dies durch ArgoUML für keine Sprache unterstützt.
Teil 2. Referenz
Anwenderschnittstelle
Kapitel 8. Einleitung

Dieses Kapitel beschreibt das gesamte Verhalten der Anwenderschnittstelle. Die Beschreibung der verschiedenen Teile der Komponenten, die Menüzeile, Fenster und verschiedene Diagramme befinden sich in separaten Kapiteln.

8.1. Überblick über das Fenster

Abbildung 8.1, „Überblick über die ArgoUML-Fenster“ zeigt das Hauptfenster von ArgoUML.

Die Titelzeile des Fensters zeigt die folgenden 4 Informationen, jeweils getrennt voneinander durch einen Bindestrich.

- Der Name des aktuell aktiven Diagrammes.
- Der Name „ArgoUML“.

Abbildung 8.1. Überblick über die ArgoUML-Fenster
Oben am Bildschirm befindet sich eine Menüzeile, die in Kapitel 9, *Die Symbolleiste* beschrieben wird. Darunter ist die Symbolleiste, die in Kapitel 9, *Die Symbolleiste* beschrieben wird.

Der Fensterrumpf teilt sich in vier Subfenster oder Felder auf. Von oben links im Uhrzeigersinn sind dies der Explorer (siehe Kapitel 11, Der Explorer), das Editierfenster (siehe Kapitel 12, Das Editierfenster), das Detailfenster (siehe Kapitel 13, Der Bereich Details) und das „Zu bearbeiten“-Fenster (siehe Kapitel 14, Der Bereich Zu-Bearbeiten). Alle 4 Fenster haben oben eine Werkzeugleiste (im Detailfenster befindet sie sich unter dem Eigenschafts-Register. Einen Überblick über die Fenster finden Sie in Abschnitt 8.3, „Generelle Informationen über Fenster“). Im unteren Bereich des Fensters befindet sich eine Statuszeile, die in Abschnitt 8.4, „Die Statuszeile“ beschrieben ist.

### 8.2. Generelles Verhalten der Maus in ArgoUML

Das in verschiedenen Fenstern von ArgoUML (siehe Abschnitt 8.3, „Generelle Informationen über Fenster“) oder der Menüzeile vorhandene, spezifische Verhalten der Maus wird in den Kapiteln diskutiert, die diese Fenster und die Menüzeile beschreiben. In diesem Abschnitt behandeln wir das Verhalten, das in ArgoUML generell anzutreffen ist.

An verschiedenen Stellen in ArgoUML muss Text direkt editiert werden (zum Beispiel im Randbedingungs-Editor; siehe Abschnitt 13.7.1, „Der Bedingungs-Editor“). Das Verhalten der Maus beim Bearbeiten von Text wird in den nachfolgenden Abschnitten diskutiert.

#### 8.2.1. Maustasten-Terminologie


Ein einfaches Drücken und Loslassen der Maustaste wird als Klick bezeichnet. Zwei Klicks in schneller Abfolge wird als Doppelklick bezeichnet. Das Bewegen der Maus während eine Taste gedrückt ist, wird als *Tastenbewegung* bezeichnet, mit einem Startpunkt bei *Taste gedrückt* und einem Endpunkt bei *Taste losgelassen*.

#### 8.2.2. Taste 1 Klick


Noch detaillierter: der Taste 1-Klick kann die folgenden Ergebnisse verursachen:

#### 8.2.2.1. Auswahl

Hier wird die Taste 1 dazu verwendet, ein Modellelement (aus einer Liste oder einer Baumstruktur oder aus einem Diagramm) auszuwählen (zu markieren), auf das die darauf folgenden Operationen

In einem Diagramm werden die markierten Modellelemente durch farbige „Vierecke“ an den Ecken/Enden des Objektes dargestellt. Modellelemente können auf unterschiedlichen Wegen markiert oder dessen Markierung aufgehoben werden:

- Taste 1-Klick. Entfernt die Markierung aller Modellelemente und markiert das angeklickte Element.
- Taste 1-Bewegung. Das Bewegen der Maus mit gedrückter Taste erlaubt in einem Diagramm, nicht mit einem Modellelement, das Zeichnen eines Rechteckes um die Modellelemente, die markiert werden, wenn die Taste 1 losgelassen wird.

8.2.2.2. Aktivierung

Hier wird die Taste 1 dazu verwendet, die Komponente der Anwenderschnittstelle zu aktivieren. Z.B. eine Schaltfläche. Das Objekt wird gewöhnlich hervorgehoben, wenn die Maustaste gedrückt und dann aktiviert, wenn die Maustaste losgelassen wird. Ein Objekt der Anwenderschnittstelle aktivieren bedeutet, dass dessen Funktion ausgeführt wird.

8.2.2.3. Navigation

Hier wird die Taste 1 dazu verwendet, den Fokus von einer Komponente der Anwenderschnittstelle oder eines Modellelementes eines Diagrammes auf ein anderes zu verändern. Dies ist unter dem Begriff Tastaturfokus besser bekannt, weil Tastaturkommandos gewöhnlich auf Modellelemente wirken, die den Fokus haben. Der Fokus wird durch einen (harten sichtbaren) Rahmen um das Modellelement, oder bei einem Texteingabefeld durch einen blinkenden Cursor dargestellt.

8.2.2.4. Generelles Verhalten beim Editieren von Text

Hier wird Taste 1 dazu verwendet, die Stelle innerhalb des Textes zu markieren, an dem die Operation (Text hinzufügen und löschen) ausgeführt werden soll.

8.2.3. Taste 1-Doppelklick

Das Verhalten des Taste 1-Doppelklicks variiert in den einzelnen Fenstern und wird in den zugehörigen Kapiteln behandelt.

8.2.3.1. Generelles Verhalten beim Editieren von Text

Hier wird der Taste 1-Doppelklick dazu verwendet, ein vollständiges Wort oder eine andere syntaktische Einheit innerhalb des Textes zu markieren. Nachfolgende Operationen (Text einfügen und löschen) werden den markierten Text ersetzen.

8.2.4. Taste 1-Bewegung

8.2.4.1. Generelles Verhalten beim Editieren von Text
Hier wird die Taste 1-Bewegung verwendet, um einen Textbereich zu markieren. Nachfolgende Operationen (Text einfügen und löschen) werden den markierten Text ersetzen.

8.2.5. Umschalt- und Strg- und die Taste 1

8.2.5.1. Innerhalb von Listen

Dieses Verhalten tritt auf, wo es Listen mit Dingen gibt, die markiert werden dürfen. Dies schließt verschiedene Dialogfenster und das „Zu bearbeiten“-Fenster ein, wo es eine Liste von „zu bearbeitenden“ Elementen gibt, die zu markieren sind.

Dort, wo markiert werden muss wird die Umschalttaste mit der Taste 1 verwendet, um die Markierung von der ursprünglichen Taste 1-Markierung zur aktuellen Position zu erweitern.

Die Strg-Taste und die Taste 1 wird ähnlich verwendet, um individuelle Elemente der aktuellen Markierung hinzuzufügen. Wird Strg-Taste 1 auf einem bereits markierten Element verwendet, wird das Element aus der Markierung entfernt.

Achtung


8.2.5.2. Generelles Verhalten beim Editieren von Text


8.2.6. Alt mit Taste 1: Verschieben

Wenn Sie die Alt Gr-Taste gedrückt halten, während Sie die Taste 1 in einem Diagramm drücken, wird die Bewegung der Maus den Zeichenbereich verschoben. Diese Funktion wird durch den Mauszeiger angezeigt, der zu einer Kreuz mit Pfeilen-Darstellung wechselt.

8.2.7. Strg mit Taste 1: Bedingtes ziehen

Wenn Sie die Strg-Taste herunterdrücken, während Sie mit der gedrückten Maustaste 1 auf ein Diagramm ziehen, wird die Bewegung des gezogenen Elementes bedingt in eine der acht grundsätzlichen Richtungen erfolgen: Norden, Süden, Osten, Westen, Nordosten, Südosten, Südwesten, Nordwesten.

8.2.8. Taste 2-Aktionen

Taste 2-Aktionen sind alle vom Fenster oder der Menüzeile abhängig und werden in den jeweiligen Kapiteln behandelt.

8.2.9. Taste 2-Doppelklick

Einleitung
Taste 2-Aktionen sind alle vom Fenster oder der Menüzeile abhängig und werden in den jeweiligen Kapiteln behandelt.

8.2.10. Taste 2-Bewegung

Taste 2-Aktionen sind alle vom Fenster oder der Menüzeile abhängig und werden in den jeweiligen Kapiteln behandelt.

8.3. Generelle Informationen über Fenster

Die vier Sub-Fenster des ArgoUML-Hauptfensters werden Fenster genannt. Von oben links im Uhrzeigersinn sind dies der Explorer (siehe Kapitel 11, Der Explorer), das Editierfenster (siehe Kapitel 12, Das Editierfenster), das Detailfenster (siehe Kapitel 13, Der Bereich Details) und das „Zu bearbeiten“-Fenster (siehe Kapitel 14, Der Bereich Zu-Bearbeiten). Oben im Editierfenster gibt es eine Werkzeugleiste.

8.3.1. Fenstergrösse verändern

Sie können die Fenstergrösse verändern, indem Sie die Trennbalken zwischen den Fenstern verschieben. Um diese Möglichkeit anzuzeigen, wechselt die Mausdarstellung, wenn sie sich über den Trennbalken befindet.


Es gibt ebenso kleine nach unten gerichtete Pfeile in den horizontalen Trennbalken, jeweils am linken Ende. Klicken auf diese Pfeile wird den Explorer und das Editierfenster auf die volle Höhe des Fensters erweitern.

Durch die Verwendung des oberen Pfeiles der vertikalen Trennbalken und des Pfeiles des horizontalen Trennbalkens ist es möglich, das Editierfenster auf das gesamte Fenster zu erweitern.

Die ursprüngliche Konfiguration kann durch erneutes Klicken auf diese Pfeile wiederhergestellt werden, die sich jetzt am Rand des Fensters befinden.

8.4. Die Statuszeile

Kapitel 9. Die Symbolleiste

9.1. Dateioperationen

Diese Schaltflächen haben wie ihre Gegenstücke im Menü Datei identische Funktionen.

- Neu
  Siehe vollständige Beschreibung unter Abschnitt 10.3.1, „ Neu “.

- Projekt öffnen...
  Siehe vollständige Beschreibung unter Abschnitt 10.3.2, „ Projekt öffnen... “.

- Projekt speichern
  Siehe vollständige Beschreibung unter Abschnitt 10.3.3, „ Projekt speichern “.

- Projekt-Eigenschaften
  Siehe vollständige Beschreibung unter Abschnitt 10.3.14, „ Projekteneinstellungen... “.

- Drucken
  Siehe vollständige Beschreibung unter Abschnitt 10.3.10, „ Drucken... “.

9.2. Editieroperationen

Diese Schaltflächen haben mit Ihren Gegenstücken im Menü Bearbeiten identische Funktionen.

- Aus Diagramm entfernen
  Siehe vollständige Beschreibung unter Abschnitt 10.4.2, „ Aus Diagramm entfernen “.

- Aus Modell entfernen
  Siehe vollständige Beschreibung unter Abschnitt 10.4.3, „ Aus Modell entfernen “.

- Perspektiven konfigurieren
  Siehe vollständige Beschreibung unter Abschnitt 10.4.4, „ Perspektiven konfigurieren... “.

- Einstellungen
  Siehe vollständige Beschreibung unter Abschnitt 10.4.5, „ Einstellungen... “.

9.3. Ansicht-Operationen

Die Schaltfläche Suche... hat das identische Verhalten wie das Gegenstück im Menü Ansicht. Die Schaltfläche Zoom ist die luxeröser wie die Version im Menü Ansicht.

- Suche...
  Siehe vollständige Beschreibung unter Abschnitt 10.5.2, „ Suchen... “.

- }
Dies ist eine andere Version als die im Menü Ansicht in Abschnitt 10.5.3, „Zoom“ beschriebene Fassung. Das Klicken mit der Taste 1 auf das Zoom-Symbol öffnet ein Fenster wie im Bild unten dargestellt.

Abbildung 9.1. Der Zoom-Schieberegler in der Symbolleiste

Wenn das Fenster geöffnet ist, sind folgende Aktionen möglich:

- Das Klicken mit der Taste 1 auf den „Schieberegler“ gefolgt von der Bewegung der Taste 1 wird den Zoomfaktor einstellen.

- Das Klicken mit der Taste 1 auf die angezeigte Prozentzahl erlaubt das direkte Editieren des vorgegebenen Zoomfaktors (in Prozent) mit der Tastatur. Ein Doppelklick auf den angezeigten Wert markiert den vollständigen Eintrag für ein leichtes überschreiben.

- Das Klicken mit der Taste 1 ober- oder unterhalb des Schiebereglers erhöht oder verringert den Zoomfaktor um 1%. Verwenden Sie diese Funktion, um die Feinjustierung des Prozentsatzes vorzunehmen.

- Das Klicken mit der Taste 1 oder mit der Taste 2 auf das Zoom-Werkzeug oder ausserhalb des Schiebereglerfensters schliesst das Fenster.

- Mit Hilfe der Tastatur kann der Zoom-Schieberegler wie folgt bedient werden: Wenn das Zoom-Symbol in der Werkzeugleiste den Fokus hat (wird durch einen dünnen blauen Rahmen dargestellt) und drücken der Leertaste öffnet sich das Zoom- Schiebereglerfenster. Verwenden Sie die Pfeil-Tasten, um den Prozentsatz 1 durch 1 zu erhöhen bzw. zu verringern. Verwenden

90
Sie **Umschalt-Tab**, um den Fokus auf das Eingabefeld Prozentsatz zu bringen, in dem sie den vorgegebenen Wert direkt editieren können. Drücken der Taste **Return** aktiviert den geänderten Wert. Wenn der „Schieberegler“ den Fokus hat, drücken der Taste **Bild nach oben/ Bild nach unten** erhöht/verringert den Prozentsatz um 50. Drücken der Taste **Pos 1** setzt den Prozentsatz auf 500% und **Ende** auf 0%.

### 9.4. Neues Diagramm

Diese Schaltflächen haben wie Ihre Gegenstücke im Menü **Neues Diagramm** identische Funktionen.

- **Anwendungsfalldiagramm** Siehe vollständige Beschreibung unter Abschnitt 10.6.1, „Anwendungsfalldiagramm“.
- **Klassendiagramm** Siehe vollständige Beschreibung unter Abschnitt 10.6.2, „Klassendiagramm“.
- **Sequenzdiagramm** Siehe vollständige Beschreibung unter Abschnitt 10.6.3, „Sequenzdiagramm“.
- **Kollaborationsdiagramm** Siehe vollständige Beschreibung unter Abschnitt 10.6.4, „Kollaborationsdiagramm“.
- **Zustandsdiagramm** Siehe vollständige Beschreibung unter Abschnitt 10.6.5, „Zustandsübergangsdiagramm“.
- **Aktivitätsdiagramm** Siehe vollständige Beschreibung unter Abschnitt 10.6.6, „Aktivitätsdiagramm“.
- **Verteilungsdiagramm** Siehe vollständige Beschreibung unter Abschnitt 10.6.7, „Verteilungsdiagramm“.
Kapitel 10. Die Menüzeile

10.1. Einleitung

Ein wichtiges Prinzip hinter ArgoUML ist es, dass Aktionen auf jedem Weg, den der Anwender für günstig hält, aufgerufen werden können. Im Ergebnis können viele (aber nicht alle) Aktionen über das Menü aber auch über andere Wege in ArgoUML ausgeführt werden.

Eine Anzahl gemeinsamer Menüeinträge ist auch über Tastaturkürzel verfügbar.


Das Folgende ist eine Erläuterung, wie die Menüelemente angeordnet sind.

- Das Menü **Datei** enthält Operationen, die auf das ganze Projekt/Datei wirken. Alle Elemente in diesem Menü können so begründet werden.
- Das Menü **Ansicht** ist für Funktionen, die weder das Modell noch das Diagrammlayout verändern, sondern nur die Art und Weise wie das Diagramm angezeigt wird. Ein gutes Beispiel ist „Zoom“. Auch navigierende Funktionen befinden sich hier, z.B. „Suchen“ und „Gehezu Diagramm...“. Alle Änderungen der Einstellungen dieses Menüs wirken sich auf alle Diagramme aus (z.B. Zoom).
- Das Menü **Neues Diagramm** enthält alle möglichen Diagramme, die erstellt werden können. Diese Funktionen sind kontextabhängig, da mit dem markierten Modellelement arbeiten.
- Das Menü **Anordnen** erlaubt Layoutänderungen im aktuellen Diagramm, was nicht das selbe ist wie die Elemente im Menü Ansicht. Die Funktionen können nicht das UML-Modell ändern.
- Das Menü **Generieren** ist für die Codegenerierung. Die Funktionen hier wirken entweder auf das markierte Modellelement oder auf das ganze Projekt.
- Das Menü **Kritiken** ist speziell für die Einstellungen der Kritiken, die für alle Projekte gelten.
- Das Menü **Werkzeuge** ist aktuell leer. Wurden Plugins installiert, erscheinen deren Funktionen an dieser Stelle.
- Das Menü **Hilfe** enthält die üblichen „Systeminformationen“ und „Über ArgoUML“.

10.2. Das Mausverhalten in der Menüzeile

Das generelle Verhalten der Maus und die Bezeichnung der Tasten ist umfassend im Kapitel
Anwenderschnittstelle ausgeführt (siehe Abschnitt 8.2, „Generelles Verhalten der Maus in ArgoUML“). Es gibt kein ArgoUML-spezifisches Verhalten für Menüs.

10.3. Das Menü Datei

Dieses sind Aktionen, welche die Ein- und Ausgabe betreffen und das umfassende Management von Projekten sowie das ArgoUML-System.

10.3.1. Neu

Tastenkürzel Strg-N.

Dies initialisiert ein neues Projekt in ArgoUML. Das Projekt wird ohne Name erstellt. Es enthält ein (oberste Ebene) unbenanntes Modell bezeichnetes Modell und zwei leere Diagramme: Ein Klassendiagramm und ein Anwendungsfalldiagramm.

Achtung


Wurde das Modell geändert (was durch den „*“ in der Titelzeile des ArgoUML-Fensters angezeigt wird), ist die Aktivierung der Funktion „Neu“ potentiell nicht die Absicht des Anwenders, da sie die Änderungen löschen würde. Aus diesem Grund erscheint ein Bestätigungsdialog, der es dem Anwender erlaubt, zuerst seine Arbeit zu speichern oder die Operation vollständig rückgängig zu machen.

Abbildung 10.1. Der Bestätigungsdialog für Neu.

10.3.2. Projekt öffnen...

Tastenkürzel Strg-O.

Diese Funktion öffnet ein existierendes Projekt aus einer Datei. Die Auswahl dieser Menüoption wird einen Dateiauswahldialog öffnen (siehe Abbildung 10.2, „Der Dateiauswahldialog für Projekt öffnen...“).

Abbildung 10.2. Der Dateiauswahldialog für Projekt öffnen...
Der Hauptbereich des Dialoges ist ein Textbereich mit einer Liste aller Verzeichnisse und Dateien im aktuell ausgewählten Verzeichnis, welches mit dem aktuellen Filter übereinstimmt (siehe nachfolgend).

Das Navigieren im Verzeichnisbaum ist durch das Markieren eines Verzeichnisses in der Dropdown-Auswahl oben im Dialog möglich. Im Baum tiefer navigieren kann durch Doppelklicken auf das im Hauptbereich angezeigte Verzeichnis mit der Taste 1 bewirkt werden.

Im unteren Teil des Dialoges befindet sich ein Textfeld mit der Bezeichnung Dateiname: für den Namen der Datei, die geöffnet werden soll. Der Dateiname kann hier direkt eingegeben oder aus der obigen Verzeichnisliste mit Hilfe des Taste 1-Klick ausgewählt werden.


- ArgoUML-komprimierte Projektdatei (*.zargo)
- ArgoUML-Projektdatei (*.uml)
- XML Metadata Interchange (*.xmi)
- XML Metadata Interchange (*.xml)
- XMI komprimierte Projektdatei (*.zip)

10.3.3. Projekt speichern

Tastenkürzel Strg-S.

Speichert das Projekt unter seinem aktuellen Dateinamen. Benutzen Sie Projekt speichern unter..., um das Projekt in einer anderen Datei zu speichern. Wurde kein Dateiname vergeben (z.B. nach Neu), dann arbeitet diese Funktion genau wie Projekt speichern unter....
**Anmerkung**


**10.3.4. Projekt speichern unter...**

Öffnet einen Dialog, der es Ihnen erlaubt, das Projekt unter einem anderen Dateinamen zu speichern (oder das erste Mal einen Dateinamen zu spezifizieren, wenn das Projekt ein neues Projekt ist).

Der Dialog ist weitgehend identisch mit dem für Projekt öffnen (siehe Abbildung 10.2, „Der Dateiauswahldialog für Projekt öffnen...“). Die Dateinamenerweiterung wird automatisch gesetzt.

**10.3.5. Projekt speichern rückgängig machen**

Diese Menüoption erlaubt es Ihnen, alle vorher vorgenommen Änderungen rückgängig zu machen und die zuletzt gespeicherte Version des aktuellen Projektes zurückzuladen. Sie arbeitet ein bisschen wie die Rückgängig-Funktion, speichert aber nur die Änderungen zurück, die seit dem letzten Speichern der Datei vorgenommen wurden.

Diese Menüoption ist deaktiviert, bis das aktuelle Projekt gespeichert oder vorher geladen und geändert wurde.

Wenn diese Menüoption aktiviert ist, öffnet sich ein kleiner Bestätigungsdialog, wie im nachfolgenden Bild gezeigt. Diese Warnung, dass alle vorher vorgenommenen Änderungen rückgängig gemacht werden ist notwendig, da diese Aktion nicht mehr rückgängig gemacht werden kann. Die Auswahl Nein bricht die ganze Aktion ab. Das ist dann so, als hätten Sie die Menüoption niemals ausgewählt. Die Auswahl Ja lädt die zuletzt gespeicherte Datei zurück.

**Abbildung 10.3. Der Warndialog für Projekt speichern rückgängig machen.**

**10.3.6. XMI importieren...**

Diese Menüoption erlaubt es Ihnen, ein UML 1.3 oder 1.4-Modell zu laden, welches durch ein anderes Tool als XMI-Datei, entsprechend dem XML-V1.0-, V1.1- oder V1.2-Standard exportiert wurde. Die Erweiterung einer solchen Datei sollte .xmi lauten.
Wenn das Modell geändert wurde (angezeigt durch einen „*“ in der Titelzeile von ArgoUML), dann ist die Aktivierung der „XMI importieren...“-Funktion wahrscheinlich nicht die Absicht des Anwenders, da dies die Änderungen löscht. Aus diesem Grund erscheint die Dialog, der es dem Anwender erlaubt, zuerst seine Arbeit zu speichern oder die Operation vollständig abzubrechen.

**Abbildung 10.4. Der Bestätigungsdialog für XMI importieren....**

Ist dieses Menü aktiviert erscheint die Standarddateiauswahl, siehe Abbildung 10.5. „Der Dialog für XMI importieren...“. Beachten Sie die Tatsache, dass diese Datei nur das Modell beinhaltet, nicht irgendein Diagrammlayout. Aus diesem Grund wird das neue Projekt keinerlei Diagramme enthalten.

**Abbildung 10.5. Der Dialog für XMI importieren....**

10.3.7. Exportiere als XMI...

---

96
Dieses Menü erlaubt es Ihnen, die vollständige Struktur eines UML 1.4-Modelles als XMI-Datei entsprechend dem XMI V2.1-Standard zu speichern. Beachten Sie die Tatsache, dass diese Datei nur das Modell enthält, nicht irgendein Diagrammlayout. Aus diesem Grund gehen die Diagramme verloren, wenn die XMI-Datei über das Menü Datei - Projekt öffnen... erneut geladen wird.

Ist das Menü aktiviert, erscheint die Standarddateiauswahl, siehe Abbildung 10.6, „Der Dialog für Exportiere als XMI...“.

**Abbildung 10.6. Der Dialog für Exportiere als XMI...**

![Export XML Dialog](image)

10.3.8. 🍃 Quellcode importieren...


Der Dialog ist ähnlich dem für Projekt öffnen... (siehe Abbildung 10.2, „ Der Dateiauswahldialog für Projekt öffnen...,“), allerdings mit zwei zusätzlichen Registern neben der Verzeichnisliste, wie in Abbildung 10.7, „ Der Dateiauswahldialog für Quellcode importieren...,“ gezeigt).

**Abbildung 10.7. Der Dateiauswahldialog für Quellcode importieren...**
Diese Felder sind im Verhalten die gleichen wie in Projekt öffnen... (siehe Abschnitt 10.3.2, „Projekt öffnen...“).

In der Nähe des Dateifilters „Alle Dateien“ gibt es den Standardfilter „Java-Quelldatei (*.java).“

Das erste der beiden Register ist mit Allgemeines bezeichnet und durch einen Taste 1-Klick auf das Register ausgewählt. Es enthält eine Auswahlbox für die Auswahl der Sprache (in V0.18 von ArgoUML kann nur Java ausgewählt werden) und der folgenden Wahlmöglichkeiten:

• Verzeichnisse rekursiv absteigend. Wenn markiert (Standard), dann wird das Reengineering auch die Unterverzeichnisse nach Javadateien durchsuchen. Wenn nicht, wird die Suche auf das aktuelle Verzeichnis eingeschränkt.

• Nur geänderte/neue Dateien. Wenn markiert (Standard), dann werden nur geänderte oder neue Dateien importiert. Wenn nicht, werden alle Klassen ersetzt.

• Erzeuge Diagramme aus dem importierten Code. Wenn Sie dies deaktivieren, werden keine Diagramme erzeugt. Z.B., alle Daten werden nur im Explorer sichtbar sein.


• Automatisches Diagramm-Layout ausführen. Wenn markiert, dann wird ArgoUML sein Bestes tun, um die generierten Diagramme automatisch zu formatieren. Wenn nicht, dann werden alle Elemente in die linke obere Ecke des Diagrammes plaziert.

• Importdetails: Nur Klassifizierungen / Klassifizierungen und Eigenschaften / Vollständiger Import. Das letztere ist der Standard.


Das zweite der beiden Register ist mit Java bezeichnet und wird durch einen Taste 1-Klick auf das
Register ausgewählt. Es enthält zwei Paar Optionsauswahlfelder.

- Die erste Optionsauswahl erlaubt die Auswahl zwischen der Modellierung der Attribute von Javaklassen als UML-Attribute (der Standard) oder als UML-Assoziationen auf die Klasse.
- Die zweite Optionsauswahl erlaubt die Auswahl zwischen der Modellierung von arrays als eigenständige neue Datentypen (der Standard) oder als Basisdatentyp mit Kardinalität.

10.3.9. Seite einrichten...

Diese Option öffnet den Standarddialog des Betriebssystems, um die Drucker-Papiergröße, die Ausrichtung und andere Optionen einzustellen.

10.3.10. Drucken...

Tastenkürzel Strg-P.

Diese Option öffnet den Standarddialog des Betriebssystems, der es Ihnen erlaubt, das aktuelle Diagramm auszudrucken.


Abbildung 10.8. Der Dialog: Das Diagramm überschreitet die Seitengröße.

Warnung

Wenn das aktuelle Diagramm keine markierten Modellelemente enthält, dann wird das gesamte Diagramm gedruckt. Jedoch, wenn ein oder mehrere Modellelemente markiert sind, wird nur der Bereich ausgedruckt den diese umfassen! Wenn die Skalierung ausgewählt ist (durch die Wahl der Option „An Seite anpassen“ im oben beschriebenen Dialog), dann wird die Skalierung nur auf Basis der markierten Modellelemente ausgeführt. Ist keine Skalierung ausgewählt (oder nicht notwendig), dann werden alle Seiten ausgedruckt, die ein markiertes Modellelement beinhalten.
10.3.11. Grafik exportieren...

Diese Menüoption öffnet einen Dialog, der es erlaubt, das aktuell markierte Diagramm (im Editierfenster) in einem von einer Vielzahl von Grafikformaten zu speichern.


Die verfügbaren Grafiktypen sind:

- GIF-Bild (*.gif)
- Encapsulated Postscript-Datei (*.eps)
- PNG-Bild (*.png)
- Postscript-Datei (*.ps)
- Scalable Vector Graphics-Datei (*.svg)

Das standardmäßig ausgewählte Grafikformat wird im Dialog unter der Menüoption Bearbeiten - Einstellungen... eingestellt.

10.3.12. Alle Grafiken exportieren...

Diese Menüoption öffnet einen Dialog, um ein Verzeichnis auszuwählen. In dieses Verzeichnis wird für alle Diagramme des aktuellen Projektes je eine Grafikdatei generiert.

Die Namen der Dateien werden aus den Diagrammnamen gebildet. Das verwendete Grafikformat wurde im Bearbeiten-Menü (siehe Abschnitt 10.4.5, „Register Einstellungen...“) eingestellt.

10.3.13. Notation

Dieses Untermenü präsentiert eine Auswahl von Optionsschaltflächen für die Notation, z.B. die Sprache, in der alle textuellen Erläuterungen in den Diagrammen dargestellt werden.

Diese Eigenschaft definiert die Notation des Projektes.

Es gibt 2 Wege, die Notation einzustellen:

- Im Bearbeiten-Menü, siehe Abschnitt 10.4.5.7, „Register Notation“ im Notationsregister des Dialoges Einstellungen, der die Standard- Notation für neue Projekte definiert. Diese Einstellung wird in der Datei argouml.user.properties im Verzeichnis argouml des User-Home-Verzeichnisses gespeichert.

- Im Datei-Menü, Menüelement Notation. Dieses bestimmt, wie alle textuellen Erläuterungen in den Diagrammen des aktuellen Projektes dargestellt werden. Diese Einstellung wird in der Projektdatei gespeichert.

Die folgenden 2 Notationen werden in ArgoUML erzeugt:
Die Menüzeile

- UML 1.4. Verwendet die UML-Notation als Standardnotation für jedes Modellelement in jedem Diagramm.
- Java. Verwendet die Java-Notation als Standardnotation für jedes Modellelement in jedem Diagramm.

Die folgenden Optionen sind nur verfügbar, wenn das entsprechende Plugin installiert wurde.

- Cpp.
- CSharp.
- PHP.

*Neben UML ist in V0.22 von ArgoUML nur Java teilweise implementiert.*

10.3.14. Projekteinstellungen...

In dieser Menüoption erscheint ein Dialog, der es dem Anwender erlaubt, verschiedene Optionen des aktuell geladenen Projektes einzustellen.

Alle Einstellungen in diesem Dialog werden in der Projektdatei, zusammen mit dem Modell, gespeichert.

Abbildung 10.9. Der Dialog Projekteinstellungen: Das Register Benutzer.

Im Register Benutzer können Sie folgende Felder einstellen:

- Das erste Feld enthält den Namen des Autors oder den Verantwortlichen für das aktuelle Projekt.
Standardmäßig wird der Name und die E-Mailadresse des Erstellers eingefügt, so dass Sie dies wahrscheinlich niemals bearbeiten müssen. Aber, es ist möglich.

- Das Feld Projektbeschreibung kann einen beliebigen Text enthalten, den Sie zur Beschreibung des Projektes benötigen. Standardmäßig ist dieses Feld leer.


**Abbildung 10.10. Der Dialog Projekteinstellungen - Das Register Profile.**

Im Register Profile können Sie die folgenden Einstellungen ändern:

- Den Typ der “Stereotype Darstellung” für das Projekt; diese kann als Text, mit kleinen oder großen Symbolen erscheinen.

- Die im Projekt konfigurierten UML-Profile - die Modellelemente dieser UML-Profile können im Projekt referenziert werden.

**Abbildung 10.11. Der Dialog Projekteinstellungen: - Das Register Notationen.**
Im Register Notationen können Sie die folgenden Felder einstellen:

- Das erste Feld ist ein Dialogfeld, das die Auswahl der im Projekt verwendeten Notation erlaubt. Standardmäßig werden UML und Java aufgelistet. Es können aber auch andere Sprachen per Plugins hinzugefügt werden. Weitere Erläuterungen entnehmen Sie dem Kapitel über die Notation: Abschnitt 12.11. „Notation“.

- Französische Anführungszeichen (« ») für Stereotypen (standardmäßig leer). Standardmäßig verwendet ArgoUML Paare von kleiner als und größer als (<< >>) Zeichen für Stereotypen. Wird dieses Kästchen markiert, werden die Stereotypen in den Diagrammen in richtigen französische Anführungszeichen gestellt (« »).

Diese Eigenschaft wurde ArgoUML vermutlich hinzugefügt, weil französische Anführungszeichen durch die verschiedensten Schriften nur sehr schlecht unterstützt werden und wenn Sie vorhanden sind, dann sind sie sehr klein und schlecht sichtbar.


• Eigenschaften anzeigen (standardmäßig leer). Ist dies markiert, wird ArgoUML die Eigenschaften zwischen geschweiften Klammern {} anzeigen. Für ein Attribut könnte die Anzeige wie folgt aussehen: +neuesAttr : int { eingefroren }.

• Typen und Parameter anzeigen (standardmäßig markiert). Wenn dieses Markierfeld nicht markiert ist, werden die Attribute in Klassen ohne Typ dargestellt und Operationen ohne Parameter angezeigt. Diese Eigenschaft kann während der Analysephase Ihres Projektes nützlich sein. Sind alle Markierfelder im Register Notation nicht markiert, dann zeigt ArgoUML z.B. für ein Attribut folgendes an: neueOperation().

• Stereotypen im Explorer anzeigen (standardmäßig leer). Ist dies markiert, dann wird ArgoUML die Stereotypen in der Nähe des Modellelementsymbols im Explorer anzeigen. Z.B. in der Baumstruktur auf der linken Seite.


Im Register Diagrammdarstellung können Sie die im Diagramm verwendete Schriftart ändern.

10.3.15. Am häufigsten verwendete Dateien

ArgoUML erinnert sich an die am häufigsten gespeicherten Dateien und listet sie an dieser Stelle auf.
um Sie in die Lage zu versetzen, diese auf einfache Weise zu laden.

Die maximale Anzahl von Dateien, die hier gelistet werden können, kann im Menü Bearbeiten -> Einstellungen... eingestellt werden. Die Liste der Dateien wird in der Datei argo.user.properties im Homeverzeichnis des Benutzers gespeichert.

10.3.16. Beenden

Tastenkürzel Alt-F4.

Diese Menüoption schliesst ArgoUML. Wenn Sie ein Projekt mit ungesicherten Änderungen haben, erscheint eine Warnmeldung, die Sie fragt, ob Sie diese speichern wollen. Siehe Abbildung 10.13, „Der Dialog Änderungen speichern.“. Die Optionen sind:

- **Ja** (das Projekt speichern und ArgoUML beenden);
- **Nein** (das Projekt nicht speichern, aber ArgoUML beenden); und
- **Abbrechen** (das Projekt nicht speichern und ArgoUML nicht beenden).

- Der Dialog kann auch durch Klicken auf die Schliessen-Schaltfläche im Fensterrand geschlossen werden. Dies hat den gleichen Effekt wie die Auswahl "Abbrechen".


10.4. Das Menü Bearbeiten

Dieses Menü enthält die Unterstützung für das Markieren von Modellelemente im Editierfenster; das Entfernen von Modellelementen aus Diagrammen und dem Modell und die Steuerung der Benutzereinstellungen.

10.4.1. Markieren

Dieses Untermenü unterstützt das Markieren von Elementen im Menü Bearbeiten. Es hat folgende Einträge:

- **Alle Elemente** (Tastenkürzel Strg-A). Markiert alle Modellelemente im aktuellen Fenster oder im aktuellen Feld. Das genaue Verhalten hängt vom aktuellen Fenster ab (z.B. das Letzte, in das Sie hineingeklickt haben); Explorerfenster, Editierfenster, "Zu Bearbeiten"-Fenster, Detailfenster. Eine Regel ist auf alle Fälle anwendbar: die Markierung im Diagramm (Editierfenster)
und im Explorer sind immer synchronisiert.

Wenn das Editierfenster das aktuelle Fenster ist: Zuerst werden die Markierungen im Explorer und im aktuellen Diagramm entfernt und dann wird alles im Diagramm befindliche markiert (und wenn die gleichen Elemente im Explorer erscheinen, werden diese ebenfalls markiert, weil dies immer synchronisiert ist).

Wenn der Explorer das aktuelle Fenster ist: Alle sichtbaren Elemente im Explorer sind markiert und unsichtbare Elemente sind nicht markiert.

Wenn das "Zu Bearbeiten"-Fenster das aktuelle Fenster ist: Alle sichtbaren Elemente im "Zu Bearbeiten"-Fenster sind markiert, alle unsichtbaren Elemente sind nicht markiert. Tatsächlich funktioniert dies genauso wie im Explorerfenster, weil beides Baumstrukturen sind.

Wenn das Detailfenster das aktuelle Fenster ist: Die Funktion arbeitet nur, wenn sich der Cursor in einem bestimmten Feld befindet, in dem das Markieren möglich ist. Z.B. in einem Namensfeld. In so einem Fall erweitert die Funktion Alle Elemente markieren die aktuelle Markierung auf den gesamten Feldinhalt.


- Umkehren. Diese Menüoption kehrt die aktuelle Markierung im aktuellen Fenster um. Genauer: Alles, was markiert ist wird demarkiert und alles was innerhalb des aktuellen Fensters nicht markiert ist wird markiert.

10.4.2. Aus Diagramm entfernen

Tastenkürzel Entf.

Dies entfernt die aktuell markierten Elemente aus dem Diagramm, aber nicht aus dem Modell.

Das Modellelement kann durch einen Taste 2-Klick auf das Modellelement im Explorer, oder durch ziehen des markierten Elementes in das Diagramm wieder in das Diagramm eingefügt werden.

10.4.3. Aus Modell entfernen

Tastenkürzel Strg-Entf.

Diese Funktion löscht die markierten Elemente vollständig aus dem Modell.

Wenn das zu löschen Element nicht nur im aktuellen Diagramm sondern auch in einem anderen Diagramm vorhanden ist, erscheint der Dialog x.

10.4.4. Perspektiven konfigurieren...

Diese Menüoption ruft den gleichen Dialog auf, wie die Schaltfläche oben im Explorer. Die vollständige Beschreibung entnehmen Sie bitte Abschnitt 11.5, „Perspektiven konfigurieren“.

10.4.5. Einstellungen...

Diese Menüoption öffnet einen Dialog, der es dem Benutzer erlaubt, verschiedene Optionen, die das Verhalten von ArgoUML bestimmen einzustellen (siehe Abbildung 10.15, „Der Dialog Einstellungen - Voreinstellungen.“).

Diese Einstellungen werden persistent für die Nutzung durch nachfolgende ArgoUML-Sitzungen gespeichert.


**Tipp**

Dies ist eine Textdatei, die Sie zum Konfigurieren von ArgoUML bearbeiten können.

**Abbildung 10.15. Der Dialog Einstellungen – Voreinstellungen.**
Die Optionen können in verschiedenen Registern eingestellt werden, die in den folgenden Abschnitten beschrieben werden. Für jedes Register gibt es drei Schaltflächen im unteren Bereich des Dialoges.

• **OK**. Die Aktivierung dieser Schaltfläche (Taste 1-Klick) übernimmt die gewählten Einstellungen und beendet den Dialog.

• **Abbrechen**. Die Auswahl dieser Schaltfläche (Taste 1-Klick) beendet den Dialog ohne irgendeine, seit dem letzten **Übernehmen** geänderte Einstellung anzuwenden (oder seit dem der Dialog gestartet wurde, wenn **Übernehmen** noch nicht verwendet wurde).

• **Übernehmen**. Die Auswahl dieser Schaltfläche (Taste 1-Klick) übernimmt die gewählten Einstellungen und verbleibt im Dialog.

Das Schliessen des Dialoges (mit der Schliessen-Schaltfläche in der oberen Ecke des Fensterrandes) hat den gleichen Effekt, wie **Abbrechen**.

### 10.4.5.1. Register Voreinstellungen

Die Auswahl des Registers Voreinstellungen (Taste 1-Klick auf das Register) enthält die folgenden Optionen als Markierfelder.

• **Start-Fenster anzeigen** (standardmäßig markiert). Wenn markiert, wird ArgoUML ein kleines Fenster mit einem Bild während des Startvorganges anzeigen.

**Tipp**

Das Start-Fenster kann auch im Hilfe-Menü angesehen werden (siehe Abschnitt 10.11.2, „Über ArgoUML“).

• **Beim Starten: Letztes Projekt laden** (standardmäßig leer). Prüfen Sie diesen Eintrag, wenn Sie immer im gleichen Projekt arbeiten und wollen, dass dieses automatisch geladen
wird, wenn Sie ArgoUML starten.

- **Entferne (Nicht-Standard)-Diagramme während des Importes (standardmäßig leer).** Das Markieren dieses Elementes weist ArgoUML an, die "Diagrammelemente" während des importieren der XMI-Dateien zu ignorieren.


### 10.4.5.2. Register Umgebung

Das Auswählen des Registers Umgebung (Taste 1-Klick auf das Register) listet verschiedene Umgebungselemente auf. Beachten Sie, dass keiner der Pfade geändert werden kann - diese sind nur Gegenstand einer Aufzeichnung.


- **Standard-Grafikformat.** Hier können Sie das gleiche Grafikformat auswählen, wie im Menü Abschnitt 10.3.11, „Grafik exportieren...“. Das ausgewählte Format wird standardmäßig in den Menüoptionen "Grafik exportieren..." und "Alle Grafiken exportieren..." verwendet.
- **Auflösung Grafikexport.** Dies erlaubt es Ihnen, die Auflösung der erzeugten Grafiken

- `{argo.ext.dir}`. Das Verzeichnis, welches die ArgoUML-Erweiterungen beinhaltet; standardmäßig ist dies das Unterverzeichnis `ext` im Build-Verzeichnis von ArgoUML.

- `{java.home}`. Das Home-Verzeichnis der Java Laufzeitumgebung (Java Runtime Environment = JRE).

- `{user.home}`. Das Homeverzeichnis des Benutzers. Wird zum Speichern der Datei `argo.user.properties` im Verzeichnis `.argouml` verwendet.

- `{user.dir}`. Das Verzeichnis, von dem aus ArgoUML gestartet wurde.

- `Startverzeichnis`. Das Verzeichnis, in dem ArgoUML seine Dateisuche startet usw.

10.4.5.3. Register Benutzer

Dieses Register erlaubt es dem Benutzer zusätzliche Informationen zu erfassen, die im System genutzt werden. Es werden zwei Textfelder angeboten.

Abbildung 10.17. Der Dialog für Einstellungen – Benutzer.

- **Vollständiger Name**. Erlaubt es dem Benutzer seinen vollständigen Namen einzugeben.
• **Email-Adresse.** Erlaubt es dem Benutzer, seine EMail-Adresse einzugeben. Diese Informationen werden benötigt, wenn Sie Hilfe per EMail anfordern.

### 10.4.5.4. Register Erscheinungsbild

Dieses Register erlaubt es dem Benutzer, das Aussehen (Look and Feel) und das Thema einzustellen. Z.B. wie die gesamte Anwenderschnittstelle von ArgoUML aussehen soll. Es bietet die folgenden Einstellungen an.

**Abbildung 10.18. Der Dialog für Einstellungen – Erscheinungsbild.**

- **Look and Feel.** Die hier gemachte Auswahl beeinflusst die gesamte Anwenderschnittstelle. Die Änderung wird nur wirksam, wenn ArgoUML beendet und neu gestartet wird.

- **Metall-Thema.** Dieses Element ist deaktiviert, wenn das Aussehen Metall nicht ausgewählt wurde. Die Auswahl hier beeinflusst die gesamte Anwenderschnittstelle. Die Änderung wird nur wirksam, wenn ArgoUML beendet und neu gestartet wird.


### 10.4.5.5. Das Register Profile

In diesen Register kann der Anwender die Einstellungen der ArgoUML-Anwendung bezüglich der Profile ändern.

**Abbildung 10.19. Der Dialog Einstellungen – Profile.**
• **Stereotyp-Darstellung** - Auswahl, um Stereotypen als Text, kleine oder große Symbole darzustellen.

• **Standard-XMI-Verzeichnisse** - erlaubt dem Anwender, die Verzeichnisse zu konfigurieren, in denen ArgoUML die benutzerdefinierten Profile finden kann.

• **Standard-Profile** - Auswahl, welches Profil von den verfügbaren Profilen als Standard für neue Projekte festgelegt wurde.

10.4.5.6. **Das Register Tastenkombinationen konfigurieren**

(Noch zu beschreiben)

**Abbildung 10.20. Der Dialog für Einstellungen - Tastenkombinationen konfigurieren.**
10.4.5.7. Register Notation

Dieses Register erlaubt es dem Nutzer, bestimmte Notationseinstellungen zu spezifizieren. Z. B. wie Dinge in Diagrammen dargestellt werden. Es bietet folgende Markierfelder an.

Alle Einstellungen definieren nur die Standards, die in neuen Projekten verwendet werden. Wenn Sie die Art und Weise wie Diagramme in Ihrem aktuellen Projekt aussehen sollen ändern wollen, dann siehe Menü Datei - Eigenschaften.

• Notationssprache (standardmäßig UML 1.4). Diese Funktion erlaubt die Änderung der Standardnotation (z.B. Sprache: UML, Java, ...), die in Diagrammen neuer Projekte verwendet wird. Nehmen Sie an, dass ein Designer fordert, dass die Standardnotation des Projektes Java sein soll. Wenn er das Projekt speichert, wird die Auswahl Java innerhalb der Projektdatei gespeichert. Wenn jemand anderes das Diagramm anzeigt, wird es die Java-Notation ebenfalls vorfinden. Diese Person kann die UML-Notation im Menü Datei - Notation auswählen und wird alle Diagramme in UML sehen. Siehe Abschnitt 10.3.13, „Notation“.

• Die Namen der Knoten fettgedruckt darstellen.

Diese Eigenschaft veranlasst, dass die Namen jedes Knotens (z.B. etwas mit einem geschlossenen Polygon gezeichnet) fettgedruckt dargestellt werden. Es gibt keine Semantik, fettgedruckte Namen darzustellen, aber ihr Diagramm wird schöner aussehen.

• Französische Anführungszeichen (« ») für Stereotypen (standardmäßig leer). Standardmäßig verwendet ArgoUML die Zeichenpaare kleiner als und größer als (<< >>) für Stereotypen. Ist dieses Feld markiert, werden die Stereotypen in Diagrammen zwischen französische Anführungszeichen gestellt (« »).

Diese Funktion wird in ArgoUML selten genutzt, da französische Anführungszeichen in diversen Schriften schlecht unterstützt werden und wenn sie vorhanden sind, sind sie sehr klein und schlecht sichtbar.

Unabhängig von der Art und Weise in der sie dargestellt werden, können Sie immer reale französische Anführungszeichen (wenn Ihre Tastatur dies unterstützt) oder deren << >>-Äquivalente eingeben.
• Zeige Assoziationsnamen.

Diese Eigenschaft veranlasst, dass die Namen jeder Assoziation versteckt werden, sofern sie nicht markiert wurden.

• Sichtbarkeit anzeigen (standardmäßig leer). Ist dies markiert, dann wird ArgoUML die Sichtbarkeitsmarkierungen vor z.B. jedem Attribut in den Diagrammen anzeigen. In UML steht die Notation "+" für public, "-" für private, "#" für protected und "~" für Paket. Es könnte für ein Attribut z.B. wie folgt aussehen: +neuesAttr : int.


• Eigenschaften anzeigen (standardmäßig leer). Wenn dies markiert ist, dann wird ArgoUML die verschiedenen Eigenschaften zwischen geschweifte Klammern {} darstellen. Für ein Attribut z.B. könnte dies wie folgt aussehen: +neuesAttr : int { eingefroren }.

• Typen und Parameter anzeigen (standardmäßig markiert). Wenn das Feld nicht markiert ist, werden die Attribute in Klassen ohne Typ und Operationen ohne Parameter dargestellt. Diese Funktion kann während der Analysephase Ihres Projektes nützlich sein. Sind alle Felder im Register Notation nicht markiert, dann könnte ArgoUML ein Attribut wie folgt anzeigen: neuesAttr. Und für eine Operation: neueOperation().

• Stereotypen im Explorer anzeigen (standardmäßig leer). Wenn dies markiert ist, dann wird ArgoUML die Stereotypen in der Nähe der Symbole der Modellelemente im Explorer anzeigen. Z.B. in der Baumstruktur auf der linken Seite.

• Zeige "1"-Kardinalitäten.

Diese Eigenschaft erlaubt es dem Anwender auszuwählen, ob er alle Kardinalitäten, die "1" sind darstellen will oder nicht...

Manche Menschen betrachten eine nicht dargestellte Kardinalität als "undefiniert", so dass es der einzige Weg ist, zwischen einer Kardinalität von 1 und einer undefinierten Kardinalität zu unterscheiden, indem man dieses Markierfeld markiert.

• Verstecke Pfeilspitzen bei bi-direktionalen Assoziationen..

Der UML-Standard definiert unterschiedliche Arten, die Navigierbarkeit von Assoziationen in Diagrammen darzustellen. Darstellungsoption 1 ist es, alle Pfeile anzuzeigen (z.B. sie können nur in eine bestimmte Richtung navigieren, wenn ein Pfeil dargestellt wird). Darstellungsoption 2 ist, keine Pfeile anzuzeigen und Darstellungsoption 3 ist, nur dann einen Pfeil anzuzeigen, wenn die Assoziation gerichtet ( unidirektional) ist.

Vor der Version 0.26, konnte ArgoUML nur die Darstellungsoption 3 verwenden. Aktuell kann der Anwender zwischen der Option 1 und 3 auswählen. Die Option 2 wird nicht unterstützt.

In der Vergangenheit wurde die Option 3 sehr häufig in anderen UML-Werkzeugen verwendet, aber neuerdings wird die Option 1 häufiger eingesetzt.

Das Register "Darstellung" im Detailfenster erlaubt das Einstellen des Schattens je Modellelement, nachdem diese erzeugt wurden.

10.4.5.8. Das Register Diagramm-Darstellung

(Noch zu beschreiben)


10.4.5.9. Das Register Module

Dieses Register zeigt eine Liste der installierten Module an, die aktiviert oder deaktiviert werden können. Seitdem dies ein neues Konzept in ArgoUML ist, enthält es derzeit eine Liste von Modulen, die nicht entfernt werden können und eine Schaltfläche, um das Konzept zu testen. Das Drücken dieser Schaltfläche fügt dem Menü Werkzeuge eine nutzlose Menüoption hinzu.

Beachten Sie auch, dass es sich um ein Konzept für "neue " Module handelt, so dass alte einbindbare Module nicht auf diese Weise arbeiten und daher nicht aufgelistet sind.

10.4.5.10. Durch Plugins zusätzlich hinzugefügte Register

Ein Plugin-Modul hat die Möglichkeit zusätzliche Register hinzufügen zu können. Ein Beispiel ist C++, welches folgendes Register hinzufügt.
10.5. Das Menü Ansicht

Dieses Menü wird für Aktionen verwendet, die Auswirkungen darauf haben, wie die verschiedenen Fenster dargestellt werden.

10.5.1. Gehe zu Diagramm...

Dieses Menü öffnet einen Dialog, der alle im aktuellen Projekt befindlichen Diagramme aufführt.

Abbildung 10.24. Der Dialog für Gehe zu Diagramm....
Der Dialog enthält eine Tabelle aus drei Spalten und einer Zeile für jedes Diagramm im aktuellen Projekt. Ein Schieberegler erlaubt den Zugriff auf die gesamte Tabelle, wenn diese für das Fenster zu gross ist. Ein Doppelklick mit der Taste 1 auf eine beliebige Zeile wird das Diagramm im Editierfenster markieren. Die drei Spalten sehen wie nachfolgend beschrieben aus:

- Typ. Auflistung des Diagrammtyps.
- Name. Auflistung der für die Diagramme vergebenen Namen.

Wenn der Dialog nicht modal ist, kann der Dialog während des Editierens des Modelles zur leichteren Navigation geöffnet bleiben.

**Warnung**


### 10.5.2. 📔 Suchen...

Diese Menü öffnet einen nicht-modalen Dialog mit der ArgoUML- Suchmaschine.

**Abbildung 10.25. Der Dialog für Suchen . . .**
Das Register Name und Ort definiert die durchzuführende Suche. Es enthält folgendes:


- Rechts von den beiden Textfeldern erlaubt ein Auswahlelement mit der Bezeichnung Elementtyp: die Spezifikation der zu suchenden UML-Metaklasse.

- Die Auswahl Suche in: erlaubt die Suche über das gesamte Projekt (Standard) oder eine Teilsuche über die Ergebnisse der vorhergehenden Suche. Wenn sie geöffnet ist, erscheint eine Liste mit Registern, welche die Suchergebnisse enthalten.

- Unterhalb dieser Felder befindet sich die Schaltfläche Lösche Register. Diese löscht die Darstellung der Register aus den vorangegangenen Suchläufen (siehe unten). Diese Schaltfläche ist deaktiviert, wenn keine Register, ausser dem Register Hilfe, vorhanden sind.


Die beiden unteren Drittel des Dialoges beinhalten ein grundlegendes Register (mit Hilfe bezeichnet), das eine zusammenfassende Hilfestellung bereitstellt und weitere Register, welche die Ergebnisse der
Suchläufe anzeigen. Diese Bezeichnungen der Suchregister werden aus dem gesuchten Element im Diagramm zusammengesetzt und sind horizontal in zwei Hälften unterteilt.

Der Taste 1-Doppelklick auf dieses Register entfernt dieses und öffnet ein neues Fenster, welches den Registerinhalt enthält. Z. B. die Suchergebnisse. Diese Fenster kann beliebig verschoben und in seiner Größe verändert werden. Beim Register Hilfe funktioniert dies nicht.


- Typ. Enthält den Typ des Modellelementes.
- Name. Enthält den Namen des Modellelementes.
- Im Diagramm. Wo Modellelemente in einem Diagramm sichtbar sind, werden hier die Namen der Diagramme aufgelistet. Im anderen Fall wird N/A angezeigt.
- Beschreibung. Enthält die Beschreibung des Modellelementes. In ArgoUML V0.18 scheint dies auf einen Dokumenteintrag begrenzt zu sein.

Der Taste 1-Klick auf eine Zeile wird mehr Informationen über das Modellelement preisgeben, indem es die zugehörigen Modellelemente in der unteren Hälfte (siehe unten) darstellt. Der Doppelklick auf eine Zeile beschreibt ein Modellelement im Diagramm und das Element und das Diagramm werden markiert.

Die untere Hälfte des Registers ist eine Tabelle mit der Bezeichnung Zugehörige Elemente: mit den gleichen Spalten wie in der oberen Hälfte. Wenn ein Modellelement in der oberen Hälfte markiert wurde, zeigt diese Tabelle die Details eines jeden zugehörigen Elementes.

**Tipp**


**Warnung**

Dieser Dialog ist nicht modal, was es erlaubt, dass er während des editierens des Modellges gesöffnet bleiben kann. Aber die Implementierung der Version 0.26 von ArgoUML aktualisiert diesen Dialog nicht sofort, wenn Änderungen an den gefundenen Modellelementen vorgenommen werden: Änderungen des Namens des Modellelementes, Änderung des Diagrammnamens. Das Löschen eines Diagrammes stoppt nicht die Möglichkeit danach zu suchen.

### 10.5.3. Zoom

Diese Menüoption öffnet ein Untermenü, welches das Skalieren aller Diagramme um einen Faktor erlaubt. Diese Einstellung wird nicht persistent gespeichert.

Im Untermenü kann folgendes ausgewählt werden:
• Verkleinern. Tastenkürzel (Strg-Minus). Verbessert den Überblick über die Zeichnung.
• Rückgängig. Kehrt zur Standard-Zoomrate zurück (z.B. 100%).
• Vergrößern. Tastenkürzel (Strg-Plus). Vergrößert die Elemente in den Zeichnungen.

10.5.4. Gitter einstellen

Dieses Menü erlaubt folgende Auswahl der Bildschirm-Gitterdarstellungen:

• Zeilen 8: vollständiges Gitter, mit einem Zwischenraum von 8 Pixeln.
• Punkte 16: Punkte, mit einem Zwischenraum von 16 Pixeln (der Standard).
• Kein Gitter: Kein irgendwie geartetes Gitter.

10.5.5. Einrasten einrichten

Dieses Menü erlaubt die Auswahl zwischen folgenden Gitter-Einrast-Schwellwerten:

• Einrasten 4: Rastet innerhalb eines Bereiches von 4 Pixeln ein.
• Einrasten 8: Rastet innerhalb eines Bereiches von 8 Pixeln ein.
• Einrasten 16: Rastet innerhalb eines Bereiches von 16 Pixeln ein.
• Einrasten 32: Rastet innerhalb eines Bereiches von 32 Pixeln ein.

Anmerkung

Es gibt keine Option das Einrasten auf das Gitter abzustellen.

Anmerkung

Wenn Sie existierende Elemente auf geänderte Einrastbereiche ausrichten wollen, können Sie das Menü Anordnen > Am Gitter ausrichten verwenden (siehe Abschnitt 10.7.1, „Ausrichten“).

10.5.6. Seitenumbrüche

Mit dieser Menüoption werden die Seitenumbrüche im Diagramm dargestellt oder nicht (durch weiss gepunktete Linien).
Warnung

Diese Menüoption funktioniert in ArgoUML V0.26 nicht.

10.5.7. Symbolleisten

Dieses Menü erlaubt es dem Anwender beliebige Symbolleisten zu verbergen oder anzuzeigen. Standardmäßig werden alle angezeigt.

10.5.8. XML-Quelltext

Aktiviert ein Fenster, das den gesamten Inhalt des aktuellen Projektes im XML-Format darstellt.

Obwohl sehr nützlich zum Debuggen von ArgoUML, ist diese Menüoption für den normalen Benutzer wenig interessant.

10.6. Das Menü "Neues Diagramm"

Dieses Menü ist dafür gedacht, die verschiedenen, von ArgoUML unterstützten Typen von UML-Diagrammen zu erzeugen.

10.6.1. Anwendungsfalldiagramm


Tipp

Das verhindert nicht, dass Modellelemente aus anderen Namensräumen/Paketen im Diagramm erscheinen. Sie können im Explorer mit Hilfe des Popup-Menüs Zum Diagramm hinzufügen hinzugefügt werden.

10.6.2. Klassendiagramm


Tipp

Das verhindert nicht, dass Modellelemente aus anderen Namensräumen/Paketen im Diagramm erscheinen. Sie können im Explorer mit Hilfe des Popup-Menüs Zum Diagramm hinzufügen hinzugefügt werden.
10.6.3. Sequenzdiagramm


10.6.4. Kollaborationsdiagramm


Tipp

Das verhindert nicht, dass Modellelemente aus anderen Namensräumen/Paketen im Diagram erscheinen. Sie können im Explorer mit Hilfe des Popup-Menüs Zum Diagramm hinzufügen hinzugefügt werden.

10.6.5. Zustandsübergangsdiagramm

Dieser Menüeintrag erstellt ein, mit der aktuellen Klasse verknüpftes, leeres Zustandsübergangsdiagramm und markiert das Diagramm im Editierenfenster. Er erstellt auch ein UML-Element Zustandsautomat, der ein Container für die im neuen Diagramm dargestellten Elemente ist.


10.6.6. Aktivitätsdiagramm

Dieser Menüeintrag erstellt ein, mit der aktuell markierten Klasse verknüpftes, leeres Aktivitätsdiagramm und markiert das Diagramm im Editierenfenster. Er erzeugt auch ein UML-Element Aktivitätsgraph, der einen Container für die im neuen Diagramm dargestellten Elemente ist.

markiert ist, wird ein ungebundener Aktivitätsgraph erstellt. Um ein korrektes UML-Modell zu erhalten, müssen Sie den Kontext des Aktivitätsgraphen im Detailfenster angeben.

10.6.7. Verteilungsdiagramm

Dieser Menüeintrag erstellt ein leeres Verteilungsdigramm und markiert das Diagramm im Editierfenster.

**Tipp**

Modellelemente aus anderen Namensräumen/Paketen können vom Explorer aus durch ziehen oder durch das Popup-Menü Zum Diagramm hinzufügen hinzugefügt werden.

10.7. Das Menü Anordnen

Dieses Menü enthält Funktionen, die dabei helfen, die Modellelemente in den Diagrammen des Editierfensters auszurichten. Die aufgerufenen Menüfunktionen werden hauptsächlich auf jedes Modellelement oder auf die aktuell im Editierfenster markierten Modellelemente angewendet.

10.7.1. Ausrichten

Dieses Untermenü richtet die markierten Elemente aus. Es enthält sieben Ausrichtungsoptionen.

- **Oben bündig.** Richtet die markierten Modellelemente entlang ihren oberen Kanten aus.

- **Unten bündig.** Richtet die markierten Modellelemente entlang ihrer unteren Kanten aus.

- **Rechtsbündig (Tastenkürzel Strg-R).** Richtet die markierten Modellelemente entlang ihrer rechten Kanten aus.

- **Linksbündig (Tastenkürzel Strg-L).** Richtet die markierten Modellelemente entlang ihrer linken Kanten aus.

- **Horizontal mittig.** Richtet die markierten Modellelemente entlang der horizontalen Mitten in einer vertikalen Linie aus.

- **Vertikal mittig.** Richtet die markierten Modellelemente entlang ihrer vertikalen Mitten in einer horizontalen Linie aus.

- **Am Gitter ausrichten.** Richtet die markierten Modellelemente entlang ihrer oberen und rechten Kanten auf die Gittereintrastgrenzen aus (siehe Abschnitt 10.5.5, „Einrasten einrichten“).

**Tipp**

Die Ausrichtung erfolgt bezogen auf die aktuelle Gittereintrast-Einstellung. Diese kann kleiner, größer oder identisch mit dem dargestellten Gitter sein. Seitdem die elemente an den Gittereintrastgrenzen ausgerichtet werden hat dieser Menüeintrag solange keine
10.7.2. Anordnen

Dieses Untermenü ordnet die markierten Elemente an. Es enthält vier Optionen zum Anordnen.

- **Horizontal gleiche Zwischenräume.** Die am weitesten rechts und links befindlichen Modellelemente werden nicht bewegt. Die anderen werden horizontal justiert, bis der horizontale Zwischenraum (z.B. von der rechten Kante des linken Modelllementes zur linken Kante des rechten Modelllementes) zwischen allen markierten Elementen gleich ist.

- **Horizontal gleiche Abstände.** Die am weitesten rechts und links befindlichen Modellelemente werden nicht bewegt. Die anderen werden horizontal justiert, bis der Abstand zwischen den horizontalen Mitten aller markierten Elemente gleich ist.

- **Vertikal gleiche Zwischenräume.** Die oberen und die unteren Modellelemente werden nicht bewegt. Die anderen werden vertikal justiert, bis der vertikale Zwischenraum (z.B. von der unteren Kante des oberen Modelllementes zur oberen Kante des unteren Modelllementes) für alle markierten Elemente gleich ist.

- **Vertikal gleiche Abstände.** Die oberen und die unteren Modellelemente werden nicht bewegt. Die anderen werden vertikal justiert, bis der Abstand zwischen den vertikalen Mitten aller markierten Elemente gleich ist.

10.7.3. Reihenfolge

Dieses Untermenü bestimmt die Reihenfolge überlappender Elemente. Es enthält vier Optionen.

- **Nach vorne.** Die markierten Modellelemente werden in der Reihenfolge einen Schritt nach vorne geholt.

- **Nach hinten.** Die markierten Modellelemente werden in der Reihenfolge einen Schritt nach hinten gesetzt.

- **In den Vordergrund.** Die markierten Modellelemente werden in der Reihenfolge vor alle anderen Modellelemente gebracht.

- **In den Hintergrund.** Die markierten Modellelemente werden in der Reihenfolge hinter alle anderen Modellelemente gebracht.

10.7.4. Grösse an Inhalt anpassen

Dieses Menüelement wirkt auf alle markierten Elemente im aktuellen Diagramm. Es setzt die Grösse
aller Modellelemente auf die minimale Größe, in welcher der gesamte Text gerade noch in das Element passt.

10.7.5. Layout

Dieses Menüelement enthält eine automatische Diagramm-Layoutfunktion. Wenn Sie z.B. dieses Menüelement aktivieren, werden alle Elemente des aktuellen Klassendiagrammes entsprechend bestimmter Layoutalgorithmen neu angeordnet.

Diese Funktion arbeitet aktuell nur bei Klassendiagrammen. Bei allen anderen Diagrammtypen führt dieses Menüelement nichts aus.

10.8. Das Menü Generieren

Dieses Menü enthält die Funktionen für die Codegenerierung aus UML-Diagrammen. Diese Funktionalität baut auf den strukturellen Informationen der Klassendiagramme auf.

**Anmerkung**

Ohne installierte Plugin-Module unterstützt ArgoUML nur die Codegenerierung mit Java. ArgoUML V0.20 unterstützt die folgenden Sprachen per Plugin: C#, C++, php4, php5.

**Warnung**


10.8.1. Markierte Klassen generieren ...

Dieser Menüeintrag öffnet den Dialog für den ArgoUML-Codegenerator (siehe Abbildung 10.26, „Der Dialog für Markierte Klassen generieren ....“).

**Abbildung 10.26. Der Dialog für Markierte Klassen generieren ....**
Neben der Beschriftung Verfügbare Klassen listet der Dialog für jede installierte Sprache alle markierten Klassen namentlich auf, mit einem Markierfeld auf der linken Seite. Alle Markierfelder sind beim ersten Mal nicht markiert. Das Markieren eines Markierfeldes veranlasst die Codegenerierung für diese Klasse. Das Markieren mehrerer Sprachen für eine Klasse veranlasst, dass die Klasse in all diesen Sprachen generiert wird.

Die Schaltflächen Alles markieren und Nichts markieren kann helfen, wenn sehr viele Elemente markiert oder deren Markierung entfernt werden sollen.


Am Ende des Dialoges befinden sich zwei Schaltflächen, die mit Generieren und Abbrechen beschriftet sind. Ein Taste 1-Klick auf die erstgenannte wird die Codegenerierung auslösen, ein Taste 1-Klick auf die zuletzt genannte wird die Codegenerierung abbrechen.

### 10.8.2. Alle Klassen generieren...

Tastenkürzel F7.

Diese Funktion verhält sich wie Markierte Klassen generieren... (siehe Abschnitt 10.8.1, „Markierte Klassen generieren ...“) als wären alle Klassen im aktuellen Diagramm markiert.

### 10.8.3. Gesamtes Projekt generieren... (Noch zu
beschreiben)

10.8.4. Einstellungen zur Codegenerierung im Projekt...
(Noch zu beschreiben)

10.9. Das Menü Kritiken


Anmerkung

Ein Wort zur Terminologie: Die Kritiken sind Hintergrundprozesse, die das aktuelle Modell anhand verschiedener „guter“ Designkriterien überprüfen. Es gibt jeweils eine Kritik für jedes Designkriterium.

Die Ausgabe einer Kritik ist eine kritische Beschreibung - eine Ausführung zu einigen Aspekten des Modells, die nicht der guten Designpraxis zu folgen scheinen.

Zum Schluss wird die kritische Beschreibung generell durch ein hochgestelltes "zu bearbeiten"-Element empfehlen, wie der identifizierte, schlechte Designansatz berichtigt werden kann.

Anmerkung

Die Kritiken sind asynchrone Prozesse die parallel zum Hauptprozess von ArgoUML ablaufen. Änderungen benötigen üblicherweise eine oder zwei Sekunden bis die Kritiken verfügbar sind.

10.9.1. Kritiken ein-/ausschalten

Dies ist ein Markierfeld, welches steuert, ob die Kritiken eingeschaltet sind. Standardmäßig ist das Feld markiert. Ist es nicht markiert, sind alle Kritiken ausgeschaltet und jedes, durch die Kritiken generierte "zu bearbeiten"-Element (alle, außer den vom Designer von Hand erstellten) wird im "Zu bearbeiten"-Fenster versteckt.

10.9.2. Design-Wichtungen...

Dieser Menüeintrag öffnet einen Dialog, der steuert, wie die mit einem bestimmten Designbereich verknüpften Kritiken angewendet werden (siehe Abbildung 10.27, „Der Dialog für Design-Wichtungen....“).

Abbildung 10.27. Der Dialog für Design-Wichtungen....
ArgoUML kategorisiert Kritiken je nachdem wie die Designwichtungen diese adressieren. Es gibt 16 solcher Kategorien. Die Kritiken in jeder Kategorie werden detailliert im Kapitel über Kritiken (Kapitel 15, Die Kritiken) diskutiert.

Die Schieberegler können für jede Kategorie eingestellt werden, um die Kritiken zu steuern, die in dieser Kategorie ausgelöst werden. Das Verschieben eines Reglers auf Aus schaltet alle Kritiken dieser Kategorie ab und entfernt alle damit verbundenen "zu bearbeiten"-Elemente aus dem "Zu bearbeiten"-Fenster.

Das Einstellen des Reglers auf einen höher priorisierten Wert, wird alle auf oder über dieser Priorität befindlichen Kritiken innerhalb der Design...Kategorie freischalten (Aus ist die niedrigste Priorität)
Anmerkung

Die Regler sind für alle Designkategorien standardmäßig auf Hoch eingestellt.

10.9.3. Design Ziele...

Dieser Menüeintrag öffnet einen Dialog, der steuert, wie Designziele behandelt werden (siehe Abbildung 10.28, „Der Dialog für Design Ziele....“).

Abbildung 10.28. Der Dialog für Design Ziele....

ArgoUML verfolgt das Konzept, dass Designer eine Anzahl von Designzielen haben, die sie erreichen wollen (zum Beispiel eine gute strukturelle Darstellung, eine detaillierte Verhaltensdarstellung usw.). Kritiken sind mit einem oder mehreren Zielen verknüpft.

Dieser Dialog erlaubt es dem Benutzer, die Priorität eines jeden Designzieles zu spezifizieren.

Um die Kritiken, die das jeweilige Ziel beeinflussen zu steuern, können die Schieberegler für jedes Designziel eingestellt werden. Das Einstellen des Reglers auf Null, wird alle Kritiken dieses Ziels ausschalten und alle damit verknüpften "Zu bearbeiten"- Elemente aus dem "Zu bearbeiten"-Fenster entfernen.

Die Einstellung eines Reglers auf einen höheren Wert wird alle Kritiken auf oder über der Priorität innerhalb der Designwichtungskategorie freigeben (1 ist die höchste und 5 die niedrigste Priorität).
**Tipp**

Es kann nützlich sein, über diese Funktion ähnlich zu denken wie bei Design-Wichtungen (siehe Abschnitt 10.9.2, „Design-Wichtungen...“), aber mit der Gruppierung der Kritiken gemäß dem Ergebnis der OOA&D und nicht mit der Gruppierung gemäß der Struktur der UML.

**Warnung**

Die Version 0.20 von ArgoUML enthält ein einziges Designziel mit der Bezeichnung *Nicht spezifiziert*. Der Regler ist standardmäßig auf die Priorität 1 eingestellt. Jedoch enthält es keine Kritiken und hat somit keine Auswirkungen.

### 10.9.4. Kritiken anzeigen...

Dieser Menüeintrag öffnet einen Dialog, der die individuellen Kritiken steuert (siehe Abbildung 10.29, „Der Dialog für Kritiken anzeigen....“).

**Abbildung 10.29. Der Dialog für Kritiken anzeigen....**


Jede Kritik, bei der das Markierfeld in der ersten Spalte nicht markiert ist, ist inaktiv und wird nicht ausgelöst. Zusätzlich wird jedes, mit dieser Kritik verknüpfte "Zu bearbeiten"-Element aus dem "Zu bearbeiten"-Fenster entfernt.
Die Version 0.26 von ArgoUML umfasst 90 Kritiken, einige davon sind unvollständig implementiert. Sie sind je Designwichtungskategorie im Kapitel Kritiken detailliert beschrieben (siehe Kapitel 15, *Die Kritiken*).

Rechts von der Liste gibt es eine Reihe von Feldern, mit Details Kritik bezeichnet, die eine detaillierte Kontrolle über die einzelnen Kritiken gibt. Das Markieren einer Kritik in der linken Liste wird die Felder für diese Kritik befüllen.


Das erste Feld danach ist ein Textfeld mit der Beschriftung Titel:. Dieses Textfeld beinhaltet den vollständigen Titel der Kritik (der in der linken Liste abgeschnitten sein kann).

### Anmerkung

Im Titel können Sie den Text `<ocl>self</ocl>` sehen, der durch den Namen des in Frage kommenden Modellelementes ersetzt wird, wenn die Kritik ausgelöst wird.


Das nächste Feld ist mit Mehr Informationen: beschriftet und enthält eine URL, die auf weitergehende Informationen zeigt. Mit der rechts befindlichen Schaltfläche können Sie zu dieser URL springen.

### Warnung

In der Version 0.26 von ArgoUML sind keine weitergehende Informationen verfügbar und die Schaltfläche Gehe zu ist deaktiviert.

Das nächste Textfeld ist mit Beschreibung: bezeichnet und ist ein Textbereich mit einer detaillierten Beschreibung dessen, was die Kritik bedeutet. Ist der Text zu gross für den Bereich, erscheint auf der rechten Seite ein Schieberegler.

### Anmerkung

In diesem Textbereich können Sie den Text `<ocl>self</ocl>` vorfinden, der durch den Namen des in Frage kommenden Modellelementes ersetzt wird, wenn die Kritik ausgelöst wird.

Das letzte Feld ist ein mit Verwende Kennzeichen: beschriftetes Pull-down-Menü, mit drei Optionen: Immer, Wenn, nur eines und Nie.


Ein Benutzer möchte z.B. die Kritik Fehlender Name mit einem roten Unterstrich angezeigt
bekommen, ein anderer Benutzer möchte die Kennzeichen ausschalten oder mit einer grünen Wellenlinie oder einem blauen Fragezeichen bezeichnet haben. Kritiken, bei denen die Kennzeichen ausgeschaltet sind, würden immer noch im "Zu bearbeiten"-Fenster aufgelistetes Feedback erzeugen.

**Achtung**

In der Release V0.26 von ArgoUML hat diese Auswahl keine Funktion. Sie ist für die künftige Entwicklung.

Unterhalb der Felder befinden sich zwei Schaltflächen in einer horizontalen Reihe.

- **Aktivieren.** Es ist möglich, eine Kritik im "Zu bearbeiten"-Fenster zu deaktivieren (siehe Kapitel 14, *Der Bereich Zu-Bearbeiten*), was die Kritik für eine bestimmte Zeit ausschaltet. Wenn die Kritik deaktiviert wurde, wird diese Schaltfläche aktiviert und wird die Kritik wieder aktivieren. Ansonsten ist sie deaktiviert.

- **Erweitert.** Diese Schaltfläche veranlasst ArgoUML einige zusätzliche Spalten in der Tabelle der Kritiken anzuzeigen. Sie erlauben eine detailliertere Untersuchung der Eigenschaften einer Kritik.

Die untere rechte Schaltfläche des Dialoges ist mit *Schliessen* beschriftet. Ein Taste 1-Klick schliesst den Dialog.

### 10.10. Das Menü Werkzeuge

Dieses Menü enthält einen generischen Menüerweiterungspunkt für die in ArgoUML enthaltenen Plugins. Das Standardsystem hat kein Plugin und dieser Menüeintrag ist standardmäßig leer.

### 10.11. Das Menü Hilfe

Diese Menü enthält Hilfen für den Gebrauch von ArgoUML. Es hat zwei Einträge.

#### 10.11.1. Systeminformation

Dieses Menü öffnet den Dialog Systeminformation, siehe Abbildung 10.30, "Der Dialog für Systeminformation."

**Abbildung 10.30. Der Dialog für Systeminformation.**

10.11.2. Über ArgoUML

Dieser Menüeintrag öffnet das Hilfefenster von ArgoUML (siehe Abbildung 10.31, „Das Hilfefenster von ArgoUML“).

Abbildung 10.31. Das Hilfefenster von ArgoUML
Das Fenster weist sechs Register auf, die durch einen Taste-1-Klick ausgewählt werden können. Standardmäßig wird das erste Register (Startfenster) angezeigt.

- **Startfenster.** Dies zeigt das Bild und die aktuelle Versionsnummer, welche angezeigt werden, wenn ArgoUML hochfährt.

- **Version.** Dieses Register enthält die Versionsinformationen von den verschiedenen Paketen aus denen ArgoUML besteht, sowie einigen Betriebssystem- und Umgebungsinformationen.

- **Anerkennung.** Dieses Register führt alle auf, die ArgoUML erstellt haben, einschliesslich der Kontaktdaten für die verschiedenen Modul-Eigentümer.

- **Kontakt.** Dieses Register enthält die Haupt-Kontaktdaten für ArgoUML- Projekt-Webseite und der Entwickler-Mail-Listen.

- **Fehler mitteilen.** Dieses Register gibt Ihnen Informationen, wie Sie mit Fehlern in ArgoUML umgehen sollen. Es ist wichtig, dass alle Fehler erfasst und jede Kooperation gewürdigt wird.

- **Recht.** Ein Auszug der Lizenz, der die gesamte ArgoUML-Software unterliegt.

**Achtung**

Die verschiedenen Projektdokumentationen unterliegen nicht alle der Lizenz (wie es
für die Software der Fall ist). Im speziellen unterliegt dieses Handbuch der OpenPub-Lizenz (siehe Anhang F, *Open Publication Lizenz*).
Kapitel 11. Der Explorer

Der Explorer wurde vorher Navigationsfenster/-baum genannt oder machmal Navigatorfester/-baum.

11.1. Einleitung


Abbildung 11.1. Überblick über den Explorer


Eine wichtige Eigenschaft, bezogen auf die Idee der kognitive Psychologie in ArgoUML ist, dass nicht alle Modellelemente notwendigerweise in allen Perspektiven dargestellt werden. Im Gegenteil, die Perspektiven werden dazu verwendet, uninteressante Teile des Modells zu verstecken.

11.2. Das Verhalten der Maus im Explorer

Das generelle Verhalten der Maus und die Benennung der Schaltflächen ist im Kapitel mit dem Überblick über die Anwenderschnittstelle ausgeführt (siehe Kapitel 8, Einleitung ).
11.2.1. Taste 1-Klick

Elemente, die Subhierarchien haben, werden innerhalb der hierarchischen Darstellung durch angezeigt, wenn die Hierarchie verborgen ist und wenn die Hierarchie geöffnet ist.


**Anmerkung**

Wenn das Modellelement Teil eines, vom aktuell im Editierfenster angezeigten, abweichenden Diagrammes ist, gibt es keine Änderung des Diagrammes im Editierfenster.

Wo der Taste 2-Klick verwendet wurde, um ein kontextsensitives Popup-Menü zu öffnen (siehe nachfolgend), wird der Taste 1-Klick dazu verwendet, den gewünschten Menüeintrag auszuwählen. Ein Taste 1-Klick ausserhalb des Menübereiches wird diesen entfernen.

11.2.2. Taste 1-Doppelklick

Dies hat den gleichen Effekt wie ein einziger Taste 1-Klick. Wenn das Baumelement kein Blatt ist, wird es zwischen dem Öffnen und Schliessen der Hierarchie hin- und herwechseln.

11.2.3. Taste 1-Bewegung

Die Taste 1-Bewegung bedeutet, dass Sie ein oder mehrere Modellelemente nehmen und an eine neue Stelle ziehen. Das Loslassen des Modellelementes bewirkt in ArgoUML die Ausführung einiger Funktionen. Je nachdem, wo Sie das Modellelement loslassen.

11.2.3.1. Von Explorer zu Explorer

Das Loslassen der Maustaste über einem Namensraum bewirkt, dass das Modellelement Teil dieses Namensraumes wird. In der paketorientierten Explorerperspektive bedeutet dies eine einfach ziehen- und-loslassen-Funktion.

Verwenden Sie diese ziehen-und-loslassen-Eigenschaft, um z.B. Klassen leicht von einem Paket zu einem anderen zu bewegen.

11.2.3.2. Vom Explorer zum Diagramm

Das Loslassen eines Modellelementes in einem Diagramm entspricht der Funktion "Zum Diagramm hinzufügen". Aus diesem Grund wird es, wenn das Diagramm dieses Modellelement noch nicht darstellt, hinzugefügt.

Verwenden Sie diese ziehen-und-loslassen-Eigenschaft, um z.B. ein Diagramm aus importierten XMI-Dateien zu erstellen. Dieses tun Sie, weil XMI-Dateien zwar alle Modellelemente, aber keine Diagramminformation enthalten.
11.2.4. Taste 2-Aktionen

Wenn sie im Explorer verwendet werden, werden sie ein auswahlabhängiges Popup-Menü anzeigen. Die Menüeinträge sind hervorgehoben (aber nicht markiert) und Untermenüs werden durch nachfolgende Mausbewegungen geöffnet (ohne irgendwelche Tasten). Die Auswahl der Menüeinträge erfolgt mit der Taste 1 oder der Taste 2.

11.2.5. Taste 2-Doppelklick

Dies hat keinen anderen Effekt als ein einfacher Taste 2-Klick.

11.3. Verhalten der Tastatur im Explorer

Alle in einer Baumverzweigung aktiven Tasten weisen ihr normales Verhalten auf.

Wenn ein Diagramm ausgewählt ist, wird das Drücken von Strg-C das Diagramm im GIF-Format in die Zwischenablage kopieren.

11.4. Auswahl der Perspektiven


Die folgenden Explorerperspektiven können in dem obigen Pull-down-Menu ausgewählt werden:


  Unterhalb eines jeden Paketes befinden sich alle Modellelemente, die sich innerhalb des Namensraumes dieses Paketes befinden, einschließlich aller weiteren Sub-Pakete (die widerum ihre eigenen Subhierarchien haben können).

- **Klassenorientiert**. Zeigt Klassen in deren Pakethierarchie, genauso wie Datentypen und Elemente von Anwendungsfalldiagrammen. Sie ist der paketorientierten Sicht sehr ähnlich, aber sie zeigt keine verbundenen oder verkümmerten Elemente.

- **Diagrammorientiert**. In dieser Sicht umfasst die oberste Ebene alle Diagramme des Modelles. Unterhalb eines jeden Diagrammes befindet sich eine flache Liste aller Modellelemente des Diagrammes. Modellelemente, die Sub-Modellelemente haben, die nicht im Diagramm erscheinen, haben ihre eigene Hierarchie (zum Beispiel Attribute und Operationen von Klassen).

- **Vererbungsorientiert**. In dieser Sicht zeigt die oberste Ebene das Modell. Unterhalb dieser Ebene befinden sich alle Modellelemente, die im Modell keine Generalisierung aufweisen. Modellelemente, die eine Spezialisierung aufweisen haben eine Sub-Hierarchie, welche die Spezialisierungen anzeigt.

- **Klassenassoziationenorientiert**. In dieser Sicht zeigt die oberste Ebene das Modell. Darunter befinden sich alle Diagramme und alle Klassen. Alle Klassen mit Assoziationen zeigen die
Hierarchie zu den assoziierten Klassen.

- **Hierarchieorientiert.** In dieser Sicht wird das Modell auf der obersten Ebene dargestellt, darunter nur Knoten und unter diesen nur Komponenten, die auf den Knoten basieren. Und unter diesen Komponenten alle Elemente die auf den Komponenten basieren.

- **Zustandsorientiert.** In dieser Sicht zeigt die oberste Ebene alle Zustandsautomaten und alle, mit Klassen verknüpften Aktivitätsgrafiken.
  Unterhalb jedes Zustandsautomaten befindet sich eine Hierarchie, die alle Zustandsübergangsdiagramme und all deren Zustände anzeigt.
  Unterhalb jeder Aktivitätsgrafik befindet sich eine Hierarchie, die das Aktivitätsdiagramm und alle seine Aktionszustände anzeigt. Unterhalb jedes Aktionszustandes befindet sich eine Liste von in den Aktionszustand ein- und ausgehenden Übergängen.

- **Übergangsorientiert.** Dies ist sehr ähnlich der Zustandsorientierten Sicht, aber unter jedem Zustandsautomaten sind die Diagramme und alle im Diagramm befindlichen Übergange aufgelistet, wobei die Zustände als Subhierarchien unter ihren verknüpften Übergängen dargestellt werden.
  Unter jedem Aktivitätsgrafen sind die Diagramme und alle Übergänge in den Diagrammen aufgelistet, wobei die Aktionszustände als Subhierarchien unter ihren verknüpften Übergängen dargestellt werden.

- **Anordnungsorientiert.** In dieser Sicht werden alle Modelllemente entsprechend ihrer Anordnung im UML-Metamodell dargestellt.
  Diese Perspektive zeigt weit mehr Modellelemente als alle anderen - sie versteckt nichts. Aus diesem Grund ist diese Sicht nicht so anwenderfreundlich, aber sehr nützlich für den UML-Spezialisten.

### 11.5. Perspektiven konfigurieren

Der Explorer ist anwenderkonfigurierbar entworfen worden, um es dem Designer zu erlauben, alles auf seine oder ihre präferierte Art und Weise anzeigen zu können.

#### 11.5.1. Der Dialog Perspektiven konfigurieren

Ein Taste 1-Klick oben links im Explorer auf das Symbol "Perspektiven konfigurieren" öffnet den Dialog Explorer Perspektiven (siehe Abbildung 11.2, „Der Dialog Perspektiven konfigurieren“).

**Abbildung 11.2. Der Dialog Perspektiven konfigurieren**
Die obere Hälfte des Dialoges enthält eine Liste aller aktuell definierten Perspektiven und rechts davon eine Reihe von untereinander angeordneten Schaltflächen. Der Taste 1-Klick kann zum Auswählen einer Perspektive verwendet werden. Sie können gleichzeitig nur jeweils eine Perspektive auswählen.

Das Markieren einer Perspektive füllt ein Textfeld oberhalb der Liste, in dem der Name der Perspektive editiert werden kann.


Als Trennung der beiden Bereiche in der unteren Hälfte des Dialoges befinden sich Schaltflächen, die mit >> und << beschrifft sind. Die erste davon überträgt die in der Bibliothek markierte Regel von der linken Liste in die rechte - z.B. sie fügt der Perspektive eine Regel hinzu. Die zweite überträgt die rechts markierte Regel in die Bibliotheksliste auf der Linken - z.B. sie entfernt aus der Perspektive eine Regel.

Wenn Sie die Maus über die horizontale Line bewegen, die die zwei Hälften des Dialoges voneinander trennen, dann sehen Sie den Änderungscursor, der Ihnen anzeigt, dass Sie diese Linie nehmen und nach oben oder unten ziehen können.

Alle drei Titel der Listen zeigen die Anzahl der in der Liste befindlichen Elemente an. ArgoUML Version 0.26 hat 9 Standardperspektiven und 72 Regeln in der Bibliothek, aus den Perspektiven gebildet werden können.
Die Schaltflächen oben rechts sind nachfolgend erklärt:

- **Neu.** Sie erstellt eine grundlegend neue Perspektive ohne Regeln und einem automatisch generierten Namen.
- **Entfernen.** Sie entfernt die markierte Perspektive.
- **Duplizieren.** Sie erstellt eine Kopie der markierten Perspektive, so dass diese als Basis für eine neue Perspektive genutzt werden kann. Die neue wird mit "Kopie von" gefolgt vom Originalnamen bezeichnet.
- **Nach oben.** Sie bewegt die markierte Perspektive um einen Platz in der Liste nach oben. Bei der obersten Perspektive ist diese Schaltfläche deaktiviert.
- **Nach unten.** Sie bewegt die markierte Perspektive um einen Platz in der Liste nach unten. Bei der letzten Perspektive ist diese Schaltfläche deaktiviert.
- **Standards wiederherstellen.** Sie stellt alle Perspektiven und deren Regeln auf die eingebauten Standards von ArgoUML ein.


### 11.6. Das kontextsensitive Menü

Ein Taste 2-Klick über irgendeinem markierten Modellelement im Hauptbereich des Explorers veranlasst, dass ein Popup-Menü erscheint.

Die Präsenz all dieser Menüeinträge hängt von den markierten Element(en) und anderen Umständen ab. Zum Beispiel das Untermenü "Erstelle Modellelement" ist bei Modellelementen, die Teil eines nicht editierbaren Profilelementes sind, nicht vorhanden.

#### 11.6.1. Erstelle neues

Dieser Eintrag im Popup-Menü öffnet ein Untermenü mit Einträgen für jeden Diagrammtyp.

Der Namensraum des neuen Diagrammes, wird auf dem markierten Modellelement basieren.

#### 11.6.2. Erstelle Modellelement


Das neue Modellelement wird aus dem markierten Modellelement zusammengesetzt.

Das Markieren mehrerer Elemente führt zu unterschiedlichen Menüs, z.B. das Markieren von 2 Klassen, erlaubt das Erzeugen von mehreren Beziehungsarten.

#### 11.6.3. Kopiere das Diagramm als Bild in die
Zwischenablage


Das Grafikformat und seine Auflösung werden durch die Standardeinstellungen von ArgoUML bestimmt: Wählen Sie das Menü Bearbeiten, dann Einstellungen... und dann das Register Umgebung aus. Die PNG und GIF-Formate und die Auflösung Standard werden empfohlen.

Tipp

Einige Anwendungen (wie z.B. Doors von Telelogic) erfordern es, dass die Hintergrundfarbe der generierten Grafik angepasst wird (es sei denn, das Bild ist leer). Dies kann mit einem Tool wie IrfanView durchgeführt werden; es ist genauso leicht wie das Klicken auf die Schaltfläche Einfügen und dann auf die Schaltfläche Kopieren.

11.6.4. Zum Diagramm hinzufügen

Dieser Eintrag im Popup-Menü erscheint bei jedem Modellelement, welches dem Diagramm im Editierfenster hinzugefügt werden kann.

Das Element kann in ein Diagramm durch Bewegen des Cursors in das Editierfenster oder ein gedoppeltes Editierfenster (wobei es als Kreuz erscheint) und klicken mit der Taste 1 plaziert werden.

Achtung

Dieser Menüeintrag erscheint nur dann als nicht deaktiviert, wenn es das Diagramm im Editierfenster erlaubt, das dieses Modellelement enthalten sein darf und das Modellelement sich nicht bereits in dem Diagramm befindet. ArgoUML läßt es nicht zu, dass Sie mehr als eine Kopie eines bestimmten Modellelementes in ein Diagramm plazieren.

11.6.5. Aus Modell entfernen

Dieser Eintrag im Popup-Menü erscheint bei jedem Modellelement, welches aus dem Modell gelöscht werden kann.

Warnung

Dies löschts das Modellelement vollständig aus dem Modell, nicht nur aus dem Diagramm. Um das Modellelement nur aus dem Diagramm zu entfernen, benutzen Sie das Bearbeiten-Menü (siehe Abschnitt 10.4.2, „ Aus Diagramm entfernen “).

Achtung

Sie können ein Diagramm aus dem Modell entfernen. Je nach Typ des Diagrammes, kann das alle Modellelemente lösen, die in dem Diagram angezeigt werden. Um die Unterschiede zu illustrieren, betrachten Sie die folgenden Beispiele:

• Das Löschen eines Klassendiagrammes lösenicht jedes Modellelement, das darin angezeigt wird. Alle Modellelemente die im Diagram angezeigt werden, bleiben im...
Modell erhalten. Dies ist so, weil ein Klassendiagramm nicht auf jedes Modelllement entsprechend dem UML-Standard V1.4 "abgebildet" wird.


11.6.6. Einstellen des Quellpfades... (Noch zu beschreiben)

Dieser Eintrag im Popup-Menü ...

11.6.7. Paket hinzufügen


11.6.8. Neuer Stereotyp

Dieser Eintrag im Popup-Menü ist verfügbar ... (Noch zu beschreiben)

11.6.9. Alle Klassen im Namensraum hinzufügen

Dieser Eintrag im Popup-Menü ist nur bei Klassendiagrammen verfügbar. Das Aktivieren dieses Menüeintrages wird alle Klassen des aktuellen Namensraumes zum Diagramm hinzufügen. Sie werden in der oberen linken Ecke angeordnet; offensichtlich eine günstige Gelegenheit, die im Menü befindliche „Anordnen->Layout“-Funktion zu verwenden.
Kapitel 12. Das Editierfenster

12.1. Einleitung

Abbildung 12.1, „Überblick über das Editierfenster“ zeigt das ArgoUML-Fenster mit dem hervorgehobenen Editierfenster.

Abbildung 12.1. Überblick über das Editierfenster


Das Fenster hat oben eine Werkzeugeiste und unten ein einziges mit Als Diagramm beschriftetes Register, das in Version 0.20 von ArgoUML keine Funktion hat. Der Hauptbereich zeigt das aktuell ausgewählte Diagramm, dessen Name in der Titelzeile des Fensters angezeigt wird.

12.2. Das Verhalten der Maus im Editierfenster

Das generelle Verhalten der Maus und die Benennung der Tasten ist im Kapitel Überblick über die Anwenderschnittstelle ausgeführt (siehe Kapitel 8, Einleitung).
12.2.1. Taste 1-Klick

In der Symbolleiste des Editierfensters wird der Taste 1-Klick dazu verwendet, ein Werkzeug für das Erstellen eines neuen Modellelementes auszuwählen und dieses dem Diagramm hinzuzufügen (siehe Doppelklicken zum Erstellen mehrerer Modellelemente). Das Hinzufügen eines neuen Modellelementes zum Diagramm wird bei den meisten Werkzeugen durch bewegen der Maus in den Editierbereich und erneutes klicken bewerkstelligt.

Im Haupteditierbereich wird der Taste 1-Klick dazu verwendet, ein individuelles Modellelement zu markieren.


Wo der Taste 2-Klick verwendet wurde, um ein kontextsensitives Popup-Menü zu öffnen (siehe unten), wird der Taste 1-Klick dazu verwendet, den gewünschten Menüeintrag auszuwählen. Das Popup-Menü wird durch einen beliebigen Taste 1-Klick ausserhalb des Menübereiches entfernt.

Es gibt verschiedene noch detailliertere Effekte, die in den Beschreibungen der verschiedenen Werkzeuge diskutiert werden (siehe Abschnitt 12.4, „Die Symbolleiste“).

12.2.2. Taste 1-Doppelklick

Wenn dies in der Werkzeugleiste mit einem Werkzeug zum Hinzufügen eines Modellelementes verwendet wird, wird das markierte Modellelement dem Zeichenbereich mehrmals hinzugefügt. Einmal für jeden weiteren Tastenklick, bis das Werkzeug erneut markiert oder ein anderes Werkzeug ausgewählt wird.

Wenn er innerhalb des Zeichenbereiches auf einem Modellelement mit Subkomponenten verwendet wird, wird der Doppelklick die Subkomponente zum Editieren auswählen (wenn notwendig, wird eine erstellt).

Das Doppelklicken über einem Operationsbereich einer Klasse wird die Operation auswählen. Oder eine erstellen, sofern noch keine vorhanden ist.

Eine spezielle Anwendung gibt es mit Paket-Modellelementen im Klassendiagramm. Ein Doppelklick auf ein Paket bringt Sie zu dem mit einem Paket verknüpften Klassendiagramm (das erste wird erstellt, wenn es mehr als eines gibt) oder bietet Ihnen an, eines für Sie zu erstellen, wenn keines vorhanden ist. Siehe Abbildung 12.2, „Der Dialog für das Hinzufügen eines neuen Klassendiagrammes“

Abbildung 12.2. Der Dialog für das Hinzufügen eines neuen Klassendiagrammes
12.2.3. Taste 1-Bewegung

Wo das Modellelement hinzugefügt wurde, wird mit dem Taste 1-Klick über dem abschließenden Modellelement eine besondere Form eines Verbinders mit seinem Endpunkt angezeigt. Der Taste 1-Klick darf auch im Raum zwischen den Modellelementen verwendet werden, um Verbindungspunkte an einem VerBinder zu erstellen. Dies ist immer dann nützlich, wo Verbinder auf sich selbst erstellt werden müssen.

Über grafischen Modellenelementen wird die Taste 1-Bewegung das Modellenelement an eine neue Position bewegen.


Einige Modellelemente (z.B. Akteur, Klasse) zeigen spezielle Griffe („Auswahl-Aktionsschaltflächen“, siehe Abschnitt 12.6, „Auswahl-Aktionsschaltflächen“) an den Seiten, oben und unten an, die gezogen werden können, um Beziehungstypen zwischen anderen Modellelementen zu bilden.


Es gibt verschiedene noch detailliertere Effekt, die in den Beschreibungen der verschiedenen Werkzeuge diskutiert werden (siehe Abschnitt 12.4, „Die Symbolleiste“).

12.2.4. Umschalt- und Strg-Veränderungen mit Taste 1

Wo mehrere Markierungen zu erstellen sind, wird die Strg-Taste mit der Taste 1 verwendet, um unmarkierte Modellelemente den aktuell markierten hinzuzufügen. Wo ein Modellelement bereits markiert ist, wird es aus der aktuellen Markierung entfernt.

Klicken auf die Taste 1 während die ALT-GR taste gedrückt ist, aktiviert das Werkzeug Besen. Dieses veranlasst, dass die markierten Modellelemente (und alles andere mitbewegt wird) durch den Besen verschoben wird (siehe Abschnitt 12.4.1, „Layout-Symbole“).

12.2.5. Taste 2-Aktionen

Wenn sie über Modellelementen im Editierenfenster verwendet wird, wird ein kontextabhängiges Pop-up-Menü erscheinen. Die Menüeinträge sind aktiviert (aber nicht ausgewählt) und die Untermenüs werden durch die fortgesetzte Mausbewegung aufgeblendet (ohne irgend eine Taste). Die Menüeinträge werden mit Taste 1 oder Taste 2 ausgewählt. Details über die spezifischen Pop-up-Menüs siehe Abschnitt 12.10, „Pop-Up Menüs“. 
Für den Fall, dass mehrere Elemente markiert sind, erscheint das Popup-Menü nur, wenn alle Elemente von der gleichen Art sind. In diesem Fall, wirken die Funktionen auf alle markierten Elemente.

12.2.6. Taste 2-Doppelklick

Dies hat keinen anderen Effekt als der einfache Taste 2-Klick.

12.2.7. Taste 2-Bewegung

Sie wird verwendet, um Elemente in einem mit Hilfe eines Taste 2-Klicks geöffneten kontextsensitiven Menü zu markieren.

12.2.8. Alt Gr mit Taste 1-Bewegung

Das Drücken der mittlere Taste im Diagramm, während die "ALT Gr"-Taste gedrückt ist, erlaubt es, die Leinwand mit der Taste- Bewegung in alle Richtungen zu scrollen.

12.3. Das Verhalten der Tastatur im Editierfenster

Viele Tastenkürzel können verwendet werden, wenn das Editierfenster aktiv ist. Hauptsächlich, um die Markierung zu ändern oder sich durch die Modellelemente zu bewegen.

12.3.1. Schrittweises Bewegen eines Modellelementes

Sie können ein Gebilde durch das markieren eines Elementes und die Verwendung der Pfeiltasten schrittweise bewegen. Das gedrückt halten der Umschalt- oder Alt-Taste wird eine grössere Bewegung produzieren.

12.3.2. Durch die Modellelemente bewegen

Sie können das dem markierten am nächsten stehende Modellelement markieren, indem Sie die Pfeiltasten verwenden, während Sie die rechte Maustaste anklicken. Sie können auch das nächste Modellelement mit Hilfe der Tab-Taste markieren oder das vorhergehende mit Hilfe der Strg+Tab-Tasten.

12.4. Die Symbolleiste


Die Symbole fallen in vier Kategorien.

- layout-Symbole. Geben Unterstützung bei der Anordnung der Modellelemente im Diagramm.
- kommentierungssymbole. Werden zum Kommentieren von Modellelementen im Diagramm verwendet.

12.4.1. Layout-Symbole

Die folgenden beiden Symbole werden in allen Diagrammen dieser Kategorie vorausgesetzt.

• **Auswählen.** Dieses Symbol sorgt für die generelle Auswahl von Modellelementen im Diagramm. Der Taste 1-Klick wird ein Modellelement markieren. Strg und die Taste 1 kann dazu verwendet werden, mehrere Modellelemente zu markieren (oder die Markierung aufzuheben). Die Taste 1-Bewegung wird markierte 2D-Elemente bewegen oder hinzufügen und bewegen eines neuen Grifles auf einer Verknüpfung. Die Taste 1-Bewegung auf einem markierten Griff einer Komponente wird die Form der Komponente dehnen.

• **Besen.** Die Taste 1-Bewegung mit diesem Symbol liefert einen „Besen“, der alle Modellelemente mitnimmt. Dies ist ein sehr schneller Weg, Dinge auszurichten.

Der Besen kann auch durch die ALT oder ALT-GR taste mit der Taste 1-Bewegung ausgelöst werden, wenn das Werkzeug **Auswählen** aktiv ist.

Der Besen wird ausführlich in seinem eigenen Kapitel diskutiert, siehe Abschnitt 12.5, „Der Besen“

**Tipp**

Zusätzliche Beeinflussungsmöglichkeiten des Modellelementlayout sind über das Menü **Anordnen** verfügbar (siehe Abschnitt 10.7, „Das Menü Anordnen“).

12.4.2. Kommentierungs-Symbole

Das Kommentierungs-Symbol **Kommentar** wird dazu verwendet, einen Kommentar zu einem markierten UML-Modellelement hinzuzufügen.

**Achtung**

Wie bei den meisten anderen Werkzeugen verwenden Sie das Werkzeug **Auswählen**, um ein Modellelement zu markieren und dann den Taste 1-Klick auf **Kommentar**, um einen Kommentar zu erstellen. Wenn kein Element markiert ist, wenn das Werkzeug Kommentar angeklickt wird, dann wird der Kommetar erstellt und in die linke obere Ecke plaziert.

Der Kommetar wird neben dem markierten Modellelement erstellt und ist standardmäßig leer. Die
Texteingabe kann mit einem Taste 1- Doppelklick aktiviert und der Text anschliessend mit Hilfe der Tastatur eingegeben werden.

Der UML-Standard erlaubt es, Kommentare zu jedem Modellelement hinzuzufügen.

Sie können jeden Kommentar mit weiteren Elementen mit Hilfe des Symboles Neue Kommentar-Verknüpfung (,...) verbinden.

12.4.3. Zeichen-Symbole


**Tipp**


Abbildung 12.3. Die Symbolauswahl Zeichen.

* Rechteck. Erzeugt ein Rechteck.

Das Editierfenster

- Gerade. Erzeugt eine Gerade.

12.4.4. Anwendungsfalldiagrammspezifische Symbole


- Akteur. Für dem Diagramm einen Akteur hinzu. Wenn sich die Maus über einem markierten Akteur befindet, werden aus Gründen des Komforts links und rechts zwei Griffe angezeigt, die gezogen werden können, um Assoziationsbeziehungen herzustellen.

**Abbildung 12.4. Die Symbolauswahl Assoziation.**
• Abhängigkeit. Fügt eine Abhängigkeit zwischen zwei Modellelementen mit Hilfe der Taste 1- Bewegung ein (vom abhängigen Modellelement).

• Generalisierung. Fügt eine Generalisierung zwischen zwei Modellelementen mit Hilfe der Taste 1- Bewegung ein (vom Kind zu den Eltern).

• Extend. Fügt eine Extend-Beziehung zwischen zwei Modellelementen mit Hilfe der Taste 1- Bewegung ein (vom erweiterten zum zu erweiternden Anwendungsfall).

• Include. Fügt eine Include-Beziehung zwischen zwei Modellelementen mit Hilfe der Taste 1- Bewegung ein (vom einschliessenden zum eingeschlossenen Anwendungsfall).


Anmerkung
Dieses Werkzeug ist deaktiviert, es sei denn, ein Anwendungsfall ist markiert.

12.4.5. Klassendiagrammspezifische Symbole


• Paket. Fügt dem Diagramm ein Paket hinzu.

• Klasse. Fügt dem Diagramm eine Klasse hinzu. Wenn sich die Maus über einer markierten Klasse befindet, werden links und rechts zwei Griffe angezeigt, die angeklickt oder gezogen werden.
können, um eine Assoziations-Beziehung herzustellen (oder eine Komposition, sofern die Umschalt-Taste gedrückt wurde) und oben und unten zwei Griffe, die angeklickt oder gezogen werden können, um eine Generalisierungs- und Spezialisierungs-Beziehung herzustellen.

- **Assoziation.** Fügt zwischen zwei Modellelemente mit Hilfe einer Taste 1-Bewegung eine Assoziation ein (vom ersten Modellelement zum zweiten). Es werden hier 2 Typen von Assoziationen angeboten, die bidirektionale oder die unidirektionale.

- **Aggregation.** Fügt zwischen zwei Modellelemente mit Hilfe einer Taste 1-Bewegung eine Aggregation ein (vom ersten Modellelement zum zweiten). Es werden hier 2 Typen von Aggregationen angeboten, die bidirektionale oder die unidirektionale.

- **Komposition.** Fügt zwischen zwei Modellelemente mit Hilfe einer Taste 1-Bewegung eine Komposition ein (vom ersten Modellelement zum zweiten). Es werden hier 2 Typen von Kompositionen angeboten, die bidirektionale oder die unidirektionale.

- **Assoziationsende.** Fügt mit Hilfe der Taste 1 ein anderes Ende zu einer bereits existierenden Assoziation ein (von der Assoziation einer Klasse, oder umgekehrt). Auf diesem Weg werden sogenannte N-wertige Assoziationen erstellt.

- **Generalisierung.** Fügt mit Hilfe der Taste 1 zwischen zwei Modellelementen eine Generalisierung ein (vom Kind zum Vater).

- **Schnittstelle.** Fügt in das Diagramm eine Schnittstelle ein. Wenn sich die Maus über einer markierten Schnittstelle befindet, wird unten ein Griff angezeigt, der gezogen werden kann, um eine Realisierungs-Beziehung herzustellen (das Ziel wird die zu realisierende Klasse).

- **Realisierung.** Fügt eine Realisierung mit Hilfe einer Taste 1-Bewegung zwischen eine Klasse und einer Schnittstelle ein (von der realisierenden Klasse zur realisierten Schnittstelle).

- **Abhängigkeit.** Fügt eine Abhängigkeit mit Hilfe einer Taste 1-Bewegung zwischen zwei Modellelementen ein (vom abhängigen Modellelement). Es werden hier auch 2 spezielle Typen von Abhängigkeiten angeboten, Erlaubnis Add a dependency between two model elements selected using button 1 motion (from the dependent model element). There are also 2 special types of dependency offered here, Permission (↑) und Verwendung (↑). Eine Erlaubnis wird standardmäßig mit dem Stereotyp Import erstellt und wird verwendet, um Elemente von einem Paket in ein anderes zu importieren.

- **Attribut.** Fügt ein Attribut zu der aktuell markierten Klasse hinzu. Das Attribut erhält den Standardnamen neuesAttr vom Typ int und kann durch einen Taste 1-Doppelklick und der Tastatur editiert werden oder durch auswählen mit einem Taste 1-Klick (nachdem die Klasse markiert wurde) und anschliessender Nutzung des Registers Eigenschaften.

**Anmerkung**
Dieses Werkzeug ist deaktiviert, außer wenn eine Klasse markiert ist.

- **Operation.** Fügt der aktuell markierten Klasse oder Schnittstelle eine Operation hinzu. Die
Operation erhält den Standardnamen neueOperation ohne Argumente und dem Rückgabewert void und kann mit Hilfe eines Taste 1-Doppelklicks und der Tastatur editiert werden, oder durch auswählen mit einem Taste 1-Klick (nachdem die Klasse markiert wurde) und der Nutzung des Registers Eigenschaften.

**Anmerkung**

Dieses Werkzeug ist deaktiviert, es sei denn, eine Klasse oder eine Schnittstelle ist markiert.

- **Assoziationsklasse.** Fügt eine neue Assoziationsklasse mit Hilfe einer Taste 1-Bewegung zwischen zwei Modellelemente ein (vom ersten Modellelement zum zweiten).

### 12.4.6. Sequenzdiagrammspezifische Symbole


- □ Klassifizierte Rolle. Fügt dem Diagramm eine klassifizierte Rolle hinzu.
- □ Vertikalen Zwischenraum in Diagramm einfügen. Fügen Sie vertikalen


12.4.7. Kollaborationsdiagrammspezifische Symbole


- Klassifizierte Rolle. Fügt dem Diagramm eine klassifizierte Rolle hinzu.


- Generalisierung. Fügt eine Generalisierung zwischen zwei Modellelementen ein, die mit der Taste 1 markiert wurden (von Kind zum Vater).


- Nachricht hinzufügen. Fügt zum markierten Assoziatationstyp eine Nachricht hinzu.

**Anmerkung**

Dieses Werkzeug ist deaktiviert, es sei denn, eine Assoziation ist markiert.

12.4.8. Zustandsdiagrammspezifische Symbole


- Zustand. Fügt dem Diagramm einen Zustand hinzu.

- Zusammengesetzter Zustand. Fügt dem Diagramm einen zusammengesetzten Zustand
hinzu. Alle Modellelemente die nachfolgend im Diagramm auf dem zusammengesetzten Zustand
plaziert werden, werden Teil des zusammengesetzten Zustandes.

- **Transition (Übergang).** Fügt eine Transition (einen Übergang) mit Hilfe einer Taste
1-Bewegung zwischen zwei Zuständen ein (von dem verursachenden Zustand zu dem empfangenden
Zustand).

- **Synchronistations-Zustand.** Fügt dem Diagramm einen Synchronisations-Zustand
hinzu.

- **Teilautomatenzustand.** Fügt dem Diagramm einen Teilautomatenzustand hinzu.

- **Teilzustand.** Fügt dem Diagramm einen Teilzustand hinzu.

- **Startzustand.** Fügt in das Diagramm einen Pseudo-Startzustand ein.

  **Achtung**

  Es gibt nichts zu stoppen, wenn Sie dem Diagramm mehr als einen Startzustand oder
zusammengesetzten Zustand hinzufügen. So etwas zu tun ist bedeutungslos und es
wird eine der Kritiken erscheinen.

- **Endzustand.** Fügt in das Diagramm einen Endzustand ein.

- **Kreuzung.** Fügt in das Diagramm einen Pseudo-Kreuzungszustand ein.

  **Achtung**

  Eine wohlgeformte Kreuzung sollte mindestens eine kommende und mindestens ein
ausgehende Transition haben. ArgoUML erzwingt dies nicht, aber es erscheint eine
Kritik bei jeder Kreuzung, die dieser Regel nicht folgt.

- **Entscheidung.** Fügt in das Diagramm einen Entscheidungs-Pseudozustand ein.

  **Achtung**

  Eine wohlgeformte Entscheidung sollte mindestens eine kommende und mindestens
ein ausgehende Transition haben. ArgoUML erzwingt dies nicht, aber es erscheint eine
Kritik bei jeder Entscheidung, die dieser Regel nicht folgt.

- **Gabelung.** Fügt in das Diagramm einen Gabelungs-Pseudozustand ein.

  **Achtung**

  Eine wohlgeformte Gabelung sollte genau eine kommende und und zwei oder mehr
ausgehende Transitionen haben. ArgoUML erzwingt dies nicht, aber es erscheint eine
Kritik bei jeder Gabelung, die dieser Regel nicht folgt.
• Vereinigung. Fügt in das Diagramm einen Vereinigungs-Pseudozustand ein.

Achtung

Eine wohlgeformte Vereinigung sollte genau eine kommende und zwei oder mehr ausgehende Transitionen haben. ArgoUML erzwingt dies nicht, aber es erscheint eine Kritik bei jeder Vereinigung, die dieser Regel nicht folgt.

• Flache Historie. Fügt in ein Diagramm eine flache Historie ein.

• Tiefgehende Historie. Fügt in das Diagramm eine tiefgehende Historie ein.


• Wächter. Fügt einer Transition einen Wächter hinzu.


12.4.9. Aktivitätsdiagrammspezifische Symbole

Es sind sieben Symbole vorhanden, die speziell für UML- Modellelemente in Aktivitätsdiagrammen geschaffen wurden. Die detaillierten Eigenschaften dieser Modellelemente werden im Abschnitt Modellelemente in Aktivitätsdiagrammen beschrieben (siehe Kapitel 22, Modellelement-Referenz Aktivitätsdiagramm).

• Aktion. Sie fügen dem Diagramm eine Aktion hinzu.

• Transition. Sie fügen mit Hilfe einer Taste 1-Bewegung eine Transition zwischen zwei markierten Aktionen ein (von der verursachenden Aktion zur empfangenden Aktion).

• Startknoten. Sie fügen dem Diagramm einen Startknoten hinzu.

Achtung

Es gibt nichts, was Sie daran hindert, dem Diagramm mehr als einen Startknoten hinzuzufügen. Wenn Sie es doch tun, ist es bedeutungslos und es wird eine der Kritiken ausgelöst.

• Endknoten. Sie fügen dem Diagramm einen Endknoten hinzu.
Entscheidungs-/Verbindungsknoten. Sie fügen dem Diagramm einen Entscheidungs-/Verbindungsknoten (Entscheidung) hinzu.

**Achtung**

Ein wohlgeformter Entscheidungs-/Verbindungsknoten sollte eine eingehende Transition und zwei oder mehrere ausgehende Transitionen aufweisen. ArgoUML erzwingt dies nicht, aber es wird eine ArgoUML-Kritik bei jedem Entscheidungs-/Verbindungsknoten ausgelöst, der dieser Regel nicht entspricht.

Gabelung. Sie fügen dem Diagramm eine Gabelung hinzu.

**Achtung**

Ein wohlgeformte Gabelung sollte eine eingehende Transition und zwei oder mehrere ausgehende Transitionen aufweisen. ArgoUML erzwingt dies nicht, aber es wird eine ArgoUML-Kritik bei jeder Gabelung ausgelöst, die dieser Regel nicht entspricht.

Vereinigung. Sie fügen dem Diagramm eine Vereinigung hinzu.

**Achtung**

Ein wohlgeformte Vereinigung sollte eine eingehende Transition und zwei oder mehrere ausgehende Transitionen aufweisen. ArgoUML erzwingt dies nicht, aber es wird eine ArgoUML-Kritik bei jeder Vereinigung ausgelöst, die dieser Regel nicht entspricht.


Objektknoten. Sie fügen dem Diagramm einen Objektknoten hinzu. Ein Objektknoten ist ein Objekt, das Eingabe oder Ausgabe einer Aktion ist.

12.4.10. Verteilungsdiagrammspezifische Symbole


**Anmerkung**

Erinnern Sie sich daran, dass ArgoUML’s Verteilungsdiagramme auch als Komponentendiagramme verwendet werden.
Das Editierfenster


- Generalisierung. Sie fügen eine Generalisierung zwischen zwei, mit der Taste 1 markierten Modellelementen ein (vom Kind zur Mutter).

- Realisierung. Sie fügen eine Realisierung zwischen einer Klasse und einer Schnittstelle ein (von der realisierenden Klasse zur realisierten Schnittstelle).

- Abhängigkeit. Sie fügen eine Abhängigkeit zwischen zwei Modellelementen ein (vom abhängigen Modellelement).


**Achtung**

Die Randbedingung, dass Assoziationen zwischen Klassen und Schnittstellen von der Schnittstelle nicht navigierbar sein dürfen, gilt auch in Verteilungsdiagrammen.


- Verknüpfung. Sie fügen eine Verknüpfung zwischen zwei Modellelementen (Knoteninstanz, Komponenteninstanz oder Objekt) ein.

**12.5. Der Besen**


Abbildung 12.5. Der Besen.


Das Betätigen der Taste Tab arbeitet genauso wie die Taste Enter.

12.6. Auswahl-Aktionsschaltflächen


Abbildung 12.6. Einige Beispiele für „Auswahl-Aktionsschaltflächen“.
Auswahl-Aktionsschaltflächen bieten häufig erforderliche Operationen auf das markierte Objekt an. Zum Beispiel: Eine Klasse hat eine Schaltfläche bei 12 Uhr, um eine Superklasse hinzuzufügen zu können; eine bei 6 Uhr, um eine Subklasse hinzuzufügen und Schaltflächen bei der 3 Uhr- und 9 Uhr-Position, um Assoziationen hinzuzufügen zu können. Diese Schaltflächen unterstützen eine "Klick und Ziehen"-Interaktion: Ein einziger Klick erzeugt eine neue verknüpfte Klasse an der Standardposition, relativ zur Originalklasse und erzeugt eine Vererbung oder eine Assoziation; das Ziehen von der Schaltfläche zu einer existierenden Klasse erzeugt nur eine Vererbung oder Assoziation; und das Ziehen in einen leeren Raum des Diagrammes erzeugt eine neue Klasse an der Mausposition mit der Vererbung oder Assoziation. ArgoUML enthält eine automatische Layout-Unterstützung, so daß das Klicken auf die Schaltfläche Subklasse die neue Klasse an einer Stelle positioniert, so dass sie sich nicht überlappen.


12.7. Erläuterungen (Clarifiers)


12.8. Das Zeichengitter

Das Editierfenster ist mit einem Hintergrundgitter ausgestattet, das auf verschiedene Arten über das Menü (siehe Abschnitt 10.5.4, „Gitter einstellen“) eingestellt oder auch ausgeschaltet werden kann.

Welches Gitter auch immer aktuell eingestellt ist, das Plazieren der Elemente im Diagramm wird immer durch die Einstellungen der Gitterrastung bestimmt, die sich im Bereich zwischen 4 und 32 Pixeln bewegt. (siehe Abschnitt 10.5.5, „Einrasten einrichten“).

12.9. Das Register Diagramm


Frühere Versionen von ArgoUML implementierten eine tabellarische Darstellung, das aktuelle Release aber unterstützt nur die Diagramm-Darstellung, so daß dieser Reiter keinerlei Funktion hat.

12.10. Pop-Up Menü’s

Ein Taste 2-Klick über einem Modellelement im Editierfenster öffnet ein Pop-up-Menü mit Menüelementen, viele davon mit einem Untermenü.

12.10.1. Kritiken


12.10.2. Reihenfolge

Dieses Menü steuert die Reihenfolge der sich überlappenden Modellelemente im Diagramm. Es entspricht dem Untermenü Reihenfolge des Menüs Anordnen (siehe Abschnitt 10.7.3, „Reihenfolge“). Es enthält vier Einträge.

  * Nach vorne. Das markierte Modellelement wird hinsichtlich der es überlappenden Modellelemente in der Reihenfolgenhierarchie eine Ebene nach oben bewegt.
12.10.3. Hinzufügen

Dieses Untermenü erscheint nur bei Modellelementen, denen Erläuterungen hinzugefügt werden können (Klassen, Schnittstellen, Objekte, Zustände, Pseudozustände) oder denen Methoden oder Attribute hinzugefügt wurden (Klassen, Schnittstellen). Es hat meistens drei Einträge.

• Neues Attribut. Erscheint nur, wenn das markierte Modellelement eine Klasse ist. Es erzeugt ein neues Attribut im Modellelement.

• Neue Methode. Erscheint nur, wenn das markierte Modellelement eine Klasse oder eine Schnittstelle ist. Erzeugt eine neue Methode im Modellelement.

• Neuer Kommentar. Fügt zu dem markierten Modellelement einen Kommentar hinzu.

• Alle Assoziationen hinzufügen. Erscheint nur, wenn das markierte Element eine Klasse oder eine Schnittstelle ist. Macht alle im Modell existierenden Beziehungen sichtbar, die mit dem markierten Modellelement verknüpft sind.

• Alle Assoziationen entfernen. Erscheint nur, wenn das markierte Modellelement eine Klasse oder eine Schnittstelle ist. Entfernt alle verknüpften Beziehungen aus dem Diagramm (ohne sie aus dem Modell zu entfernen).

12.10.4. Darstellung

Dieses Untermenü erscheint nur bei bestimmten Modellelementen. Es ist vollständig kontextabhängig. Es gibt viele mögliche Einträge, je nach markiertem Modellelement und dessen Zustand.


• Attribute ausblenden. Erscheint nur, wenn die Attribute einer Klasse oder eines Objektes eingeblendet sind. Blendet die Attribute aus.
• **Attribute einblenden.** Erscheint nur, wenn die Attribute einer Klasse oder eines Objektes ausgeblendet sind. Blendet die Attribute ein.

• **Operationen ausblenden.** Erscheint nur, wenn die Operationen einer Klasse oder eines Objektes eingeblendet sind. Blendet die Operationen aus.

• **Operationen einblenden.** Erscheint nur, wenn die Operationen einer Klasse oder eines Objektes ausgeblendet sind. Blendet die Operationen ein.

• **Aufzählung ausblenden.** Erscheint nur, wenn die Aufzählung eingeblendet ist. Blendet die Aufzählung aus.

• **Aufzählung einblenden.** Erscheint nur, wenn die Aufzählung ausgeblendet ist. Blendet die Aufzählung ein.

• **Alle Kanten einblenden.** Erscheint nur bei Klassen. Blendet alle Assoziationen ein (zu angezeigten Modellelementen) die aktuell nicht eingeblendet sind. Dies ist die selbe Funktion wie "Zum Diagramm hinzufügen" einer Assoziation im Explorer-Kontextmenü.

• **Alle Kanten ausblenden.** Erscheint nur bei Klassen. Blendet alle Assoziationen aus. Dies ist die gleiche Funktion wie die Funktion „Aus Diagramm entfernen“ auf alle Assoziationen dieser Klasse.

• **Stereotypen ausblenden.** Erscheint nur, wenn die Stereotypen eines Paketes eingeblendet sind. Blendet die Stereotypen aus.

• **Stereotypen einblenden.** Erscheint nur, wenn die Stereotypen eines Paketes ausgeblendet sind. Blendet die Stereotypen ein.

• **Sichtbarkeit ausblenden.** Erscheint nur, wenn die Sichtbarkeit eines Paketes eingeblendet ist. Blendet die Sichtbarkeit aus.

• **Sichtbarkeit einblenden.** Erscheint nur, wenn die Sichtbarkeit eines Paketes ausgeblendet ist. Blendet die Sichtbarkeit ein.

### 12.10.5. Modifikatoren

Dieses Untermenü erscheint nur bei Klassen, Schnittstellen, Paketen und Anwendungsfall-Modellelementen. Es wird verwendet, um die Werte verschiedener verfügbarer Modifikatoren einzustellen oder zu löschen.

• **Abstrakt.** Wird bei einem abstrakten Modellelement eingestellt.

• **Blatt.** Wird bei einem abschliessendem Modellelement gesetzt. Zum Beispiel eines ohne Sub-Modellelemente.

• **Wurzel.** Wird bei einem Wurzel-Modellelement gesetzt. Zum Beispiel eines ohne übergeordnetes Modellelement.

• **Aktiv.** Wird bei einem Modellelement mit dynamischem Verhalten gesetzt.

**Anmerkung**

Dies sollte natürlich automatisch bei Modellelementen mit Zustandsautomaten oder Aktivitätsdiagrammen gesetzt werden.
12.10.6. Kardinalität

Dieses Untermenü erscheint nur bei Assoziations-Modellelementen, beim Anklicken eines Assoziationsendes. Es wird dazu verwendet, die Kardinalität an dem Assoziationsende zu steuern, das dem Mausklick am nächsten liegt. Es gibt nur vier Einträge, eine Untermenge einer Menge von Kardinalitäten, die über die Eigenschaftstabelle eines Assoziationsendes verfügbar sind (siehe Abschnitt 17.6, „Assoziationsenden“).

- 1
- 0..1
- 1..*
- 0..*

12.10.7. Aggregation

Dieses Untermenü erscheint nur bei Assoziations-Modellelementen, beim Anklicken eines Assoziationsendes. Es wird dazu verwendet, die Aggregation an dem Assoziationsende zu steuern, das dem Mausklick am nächsten liegt. Es gibt drei Einträge.

- keine. Entfernt alle Aggregationen.
- Aggregation. Macht dieses Ende zu einer Aggregation (gewöhnlich als „Aggregation“ bekannt).

**Achtung**

UML fordert, dass ein Ende einer Komposition die Kardinalität 1 aufweisen muss (der Standard).

12.10.8. Navigierbarkeit

Dieses Untermenü erscheint nur bei Assoziations-Modellelementen beim Anklicken auf ein Assoziationsende. Es wird verwendet, um die Navigierbarkeit der Assoziation zu steuern. Es gibt drei Einträge.

- bidirektional. Macht die Assoziation in beide Richtungen navigierbar.
- <Klasse1> nach <Klasse2>. Macht die Assoziation nur von <Klasse1> nach <Klasse2> navigierbar. Mit anderen Worten, <Klasse1> kann die <Klasse2> referenzieren, aber nicht umgekehrt.
- <Klasse2> nach <Klasse1>. Macht die Assoziation nur von <Klasse2> nach <Klasse1> navigierbar. Mit anderen Worten, <Klasse2> kann die <Klasse1> referenzieren, aber nicht umgekehrt.
Anmerkung

UML erlaubt keine nicht-navigierbaren Assoziation in beide Richtungen. ArgoUML wird dies zulassen, aber Sie müssen die Navigationseigenschaft aller über den Reiter Eigenschaften der Assoziation erreichbaren Assoziationsenden einstellen - und das Diagramm wird in diesem Fall keine Pfeile anzeigen.

Dies wird als schlechte Design-Praxis betrachtet (es wird in ArgoUML eine Kritik auslösen), so dass dies nur von theoretischem Interesse ist.

Anmerkung


12.11. Notation

Eine Notation ist die textuelle Darstellung in einem Diagramm eines Modellelementes oder seiner Eigenschaften.

12.11.1. Notation Sprachen

ArgoUML unterstützt die Darstellung der Notation in unterschiedlichen Sprachen. Standardmäßig wird jeder Text in UML-Notation dargestellt, Menüs erhalten jedoch einen Eintrag um zwischen UML und Java auswählen zu können. Über Plugin-Module ist es auch möglich andere Sprachen auszuwählen; wie zum Beispiel C++.

Abbildung 12.7, „Eine Klasse in UML-Notation“ zeigt eine Klasse in UML-Notation, während Abbildung 12.8, „Eine Klasse in Java-Notation“ die gleiche Klasse in Java-Notation darstellt.

Abbildung 12.7. Eine Klasse in UML-Notation

![UML-Notation](image1)

Abbildung 12.8. Eine Klasse in Java-Notation

![Java-Notation](image2)
12.11.2. Notation Editieren im Diagramm

Der in einem Diagramm gezeigte Text kann durch einen Doppelklick auf den Text editiert werden. Dies blendet ein Editierfenster mit dem zuvor markierten Text auf, in dem der Text geändert werden kann.


Das Editieren im Diagramm ist eine sehr leistungsfähige Art und Weise eine Menge Modellinformationen auf kompakte Art einzugeben. Es ist zum Beispiel möglich, eine Operation, den Stereotypen, alle Parameter und deren Typen, sowie die Eigenschaften der Operationen (Sichtbarkeit, Gleichzeitig) auf einmal einzugeben:

```
+ auftrag(kundenID : Integer, positionen : List) : Boolean {sequential}
```

Eine Assoziation (z.B. zwischen zwei Klassen) zeigt viele Texte in der Nähe seiner Mitte und der Enden an, die zusätzliche Erläuterungen liefern. Abbildung 12.9, „Eine Menge von Assoziationen mit Eingabefeldern“ zeigt zwei Assoziationen, um folgendes zu erläutern:

**Abbildung 12.9. Eine Menge von Assoziationen mit Eingabefeldern**

Die rechte Assoziation zeigt, dass unsichtbare Felder in die Text eingegeben werden kann sichtbar werden, wenn das Modellelement markiert wird. Die Felder sind als blaue Rechtecke gekennzeichnet - ein Doppelklick auf diese Felder mit der rechten Maustaste ermöglicht das Editieren.

Die Sichtbarkeit (das +, -, # oder ~) wird im Zusammenhang mit dem Namen des Assoziationsendes angezeigt. Bei unbenannten Assoziationsendenden wird sie nicht angezeigt.

Die Kardinalität wird nicht angezeigt, wenn sie 1 ist, es sei denn, die Einstellung "1 Kardinalitäten anzeigen" im Menü Datei => Projekteigenschaften ist markiert.


12.11.3. Notation Parsen

(noch zu beschreiben)
Kapitel 13. Der Bereich Details

13.1. Einleitung

Abbildung 13.1, „Überblick über den Bereich Details“ zeigt das ArgoUML-Fenster mit dem hervorgehobenen Bereich Details.

Abbildung 13.1. Überblick über den Bereich Details

Für jedes Modelllement innerhalb des Systems, werden alle mit ihm verknüpften Daten innerhalb dieses Bereiches angezeigt und eingegeben.

Der Bereich enthält eine Reihe von Registern, die mit einem Taste 1-Klick ausgewählt werden. Der Rumpf des Reiters ist ein Menü von Einträgen, die je nach ausgewähltem Register markiert, ausgewählt oder eingegeben werden.

13.2. Das Register "Zu Bearbeiten"


Abbildung 13.2. Beispiel eines Zu-Bearbeiten-Elementes im Eigenschaftsbereich

Der Rumpf des Registers beschreibt das Problem und beschreibt, wie es gelöst werden kann. Links befinden sich vier Schaltflächen.

• Neue Kritik... Öffnet ein Dialogfenster (siehe Abbildung 13.3, "Dialogfenster für Neue Kritik"), das es Ihnen erlaubt, Ihre eigene Kritik zu schreiben, mit eigener Überschrift (die im Zu-Bearbeiten-Bereich erscheint), eigener Priorität für den Zu-Bearbeiten-Bereich, Referenz-URL und detaillierter Beschreibung für weitere Informationen.

Abbildung 13.3. Dialogfenster für Neue Kritik
• **Kritik auflösen...** Öffnet einen Dialog, der es dem Anwender ermöglicht, das markierte Zu-Bearbeiten-Element aufzulösen (siehe Abbildung 13.4, „Dialogfenster für Kritik auflösen“). Dies ist ein wichtiger Dialog, weil er es erlaubt, die Zu-Bearbeiten-Elemente auf andere Art und Weise zu behandeln, wie im Zu-Bearbeiten-Element empfohlen (was der Grund ihrer Existenz ist).


**Abbildung 13.4. Dialogfenster für Kritik auflösen**
Oben befinden sich drei Auswahlfelder, von denen das letzte standardmäßig markiert ist. Sie sind wie folgt bezeichnet: 1) es ist für meine Ziele nicht relevant, 2) es ist Momentan nicht von Belang und 3) die Gründe sind nachfolgend beschrieben. Wenn Sie die 3. Option auswählen, sollten Sie die Gründe in dem nachfolgendem Textbereich beschreiben.

**Tipp**

Wenn Sie ein Zu-Bearbeiten-Element auflösen wollen (das durch eine Kritik generiert wurde) indem Sie dessen Empfehlung folgen, dann führen Sie die empfohlenen Änderungen durch und das Zu-Bearbeiten-Element wird von selbst verschwinden. Es gibt dann keine Notwendigkeit, diesen Dialog zu verwenden.

**Warnung**


Wenn eine generiertes Zu-Bearbeiten-Element aufgelöst wird, dann gibt es keinen Weg dies ungeschehen zu machen (es sei denn, durch erneutes erzeugen des Kritik-auslösenden Objektes).

- **Kritik deaktivieren** Dies unterdrückt die Aktivität der Kritik, die das aktuelle Zu-

Die Kritik wird nach einer gewissen Zeit wieder aufleben. Zu Beginn ist diese Periode auf 10 Minuten eingestellt, aber sie wird bei jedem Anklicken der Schaltfläche verdoppelt. Die Kritik kann ausdrücklich durch Kritiken > Kritiken anzeigen... wieder zum Leben erweckt werden (siehe Abschnitt 10.9.4, „Kritiken anzeigen...“).

### Tipp

Einige Kritiken können während der gesamten Erstellungszeit eines grossen Diagrammes erscheinen. Einige Anwender finden es daher nützlich, diese Kritiken zu deaktivieren bis das Diagramm vollständig ist.

#### 13.2.1. Assistenten

Einige der am häufigsten vorkommenden Kritiken haben einen „Assistenten“, der dabei hilft, das Problem zu lösen. Der Assistent umfasst im Zu-Bearbeiten-Element eine Reihe von Seiten (eine oder mehrere), die Sie Schritt für Schritt durch die Änderungen führen. Sie starten den Assistenten durch Anklicken der Schaltfläche Weiter >>.

**Abbildung 13.5. Beispiel eines Assistenten**

Der Assistent wird durch die ersten drei Schaltflächen im unteren Bereich des Zu-Bearbeiten-Elementes gesteuert.


- **Weiter >>.** Dies bringt Sie zum nächsten Schritt des Assistenten. Wenn es sich um den letzten Schritt des Assistenten handelt, ist diese Schaltfläche deaktiviert.

- **Fertigstellen.** Dieser Schritt vollzieht die von Ihnen im Assistenten in den vorhergehenden Schritten vorgenommenen Änderungen und/oder verwendet die Standardwerte für alle folgenden Schritte.

**Anmerkung**

Nicht alle Zu-Bearbeiten-Elemente haben Assistenten. Wenn es keinen Assistenten gibt,
sind alle drei Schaltflächen deaktiviert.

Die ArgoUML-Assistenten sind nicht-model, d.h. einmal gestartet, können Sie andere Zu-Bearbeiten-Elemente auswählen oder andere Aktionen ausführen. Und dies alles, während sich der Assistent daran erinnert, wo er war. Sie können somit zu diesem Zu-Bearbeiten-Element zurückkehren und der Assistent wird mit dem gleichen Schritt fortfahren, in dem er war, bevor Sie ihn verlassen haben.

13.2.2. Die Schaltfläche Hilfe

Es gibt noch eine verbliebene Schaltfläche im unteren Bereich des Registers Zu-Bearbeiten-Element, mit Hilfe bezeichnet. Diese wird künftig einen Browser öffnen, mit einer URL für eine weitergehende Hilfe.

13.3. Das Register Eigenschaften

Über dieses Register werden die Eigenschaften eines im Explorer oder im Editierbereich markierten Modellelementes eingestellt. Die Eigenschaften eines Modellelementes können auf folgende Art und Weise angezeigt werden:

1. Markieren des Modellelementes im Explorer oder im Editierbereich, gefolgt von der Auswahl des Registers Eigenschaften im Bereich Details; oder
2. Über die Navigationsschaltflächen werden unterschiedliche Modellelemente markiert. Zum Beispiel, die Schaltfläche Nach oben im Register Eigenschaften, die Schaltflächen Zurück und Vorwärts in der Symbolleiste und die verschiedenen Menüeinträge unter Bearbeiten - Markieren.

Abbildung 13.6. „Ein typisches Register Eigenschaften im Bereich Details“ zeigt ein typisches Register Eigenschaften für ein Modellelement in ArgoUML (in diesem Fall eine Klasse).

Abbildung 13.6. Ein typisches Register Eigenschaften im Bereich Details

Oben links befindet sich das Symbol und der Name des Modellelement-Typs (z.B. die UML-Metaklasse, nicht der aktuelle Name dieses bestimmten Modellelementes). In diesem Beispiel ist es das Register Eigenschaften für eine Klasse.

Rechts davon befindet sich eine Symbolleiste mit Symbolen, die in diesem Register Eigenschaften relevant sind. Das Erste ist immer die Schaltfläche Nach oben. Das Letzte ist immer die Schaltfläche
Löschen, um das markierte Modelllement aus dem Modell zu löschen. Die Symbole dazwischen, hängen vom jeweiligen Modelllement ab.


Jedoch bei vielen Textfeldern und Textbereichen erfolgt die Dateneingabe über ein kontextsensitives Popup-Menü (mit Hilfe des Taste 2-Klicks), welches die Optionen anbietet, um neue Einträge hinzuzufügen, einen Eintrag zu löschen oder Einträge nach oben oder nach unten zu bewegen (in Textbereichen mit Mehrfach-Einträgen).

Das erste Feld ist meist immer ein Textfeld mit der Bezeichnung Name, wo der Name des spezifischen Modelllementes eingegeben werden kann. Die verbleibenden Felder variieren, je nach markiertem Modelllement.

Die detaillierten Eigenschaftstabellen für alle ArgoUML-Modelllemente werden in separaten Kapiteln für jeden Diagrammotyp diskutiert (Anwendungsfalldiagramm (Kapitel 17, Referenz der Modellelemente für Anwendungsfalldiagramme), Klasseendiagramm (Kapitel 18, Modelllement-Referenz Klasseendiagramm), Sequenzdiagramm (Kapitel 19, Modelllement-Referenz Sequenzdiagramm), Zustandsdiagramm (Kapitel 20, Modelllement-Referenz Zustandsdiagramm), Kollaborationsdiagramm (Kapitel 21, Modelllement-Referenz Kollaborationsdiagramm), Aktivitätsdiagramm (Kapitel 22, Modelllement-Referenz Aktivitätsdiagramm), Verteilungsdiagramm (Kapitel 23, Modelllement-Referenz Verteilungsdiagramm). Eigenschaftstabellen für Modelllemente, die sehr häufig in allen Diagrammytypen erscheinen, haben ihr eigenes Kapitel (Kapitel 16, Modellreferenz auf höchster Ebene).

**Achtung**


### 13.4. Das Register Dokumentation

Innerhalb des UML 1.4-Standards sind alle Modellelemente Subelemente der Metaklasse Element. Die Metaklasse Element definiert einen gekennzeichneten Wert Dokumentation für einen Kommentar, eine Beschreibung oder eine Erläuterung eines jeden Elementes. Da dieser gekennzeichnete Wert zu jedem Modelllement gehört, hat er sein eigenes Register im Bereich Details erhalten und nicht im Bereich Eigenschaftswerte.

Abbildung 13.7, „Ein typisches Register Dokumentation im Bereich Details“ zeigt ein typisches Register Dokumentation für ein Modelllement in ArgoUML.

**Abbildung 13.7. Ein typisches Register Dokumentation im Bereich Details**

Die Felder in diesem Register sind für alle Modellelemente die gleichen.

Da UML-Kommentare eine Art von Dokumentation sind, werden sie auch in diesem Register mit ihrem Namen und Erläuterungen angezeigt.

- **Autor:** Ein Textfeld für den Autor der Dokumentation.
- **Version:** Ein Textfeld für die Version dieser Dokumentation.
- **Seit:** Ein Textfeld, das angibt, seit wann die Dokumentation gültig ist.
- **Veraltet:** Ein Markierfeld, das angibt, ob das Modellelement veraltet ist (z.B. soll in künftigen Versionen des Designmodèles entfernt werden).
- **Siehe:** Referenzen auf Dokumentationen ausserhalb dieses Systems.
- **Dokumentation:** Der Text einer Dokumentation.
- **Kommentarbezeichnung:** Die Namen aller Kommentare, die diesem Modellelement hinzugefügt wurden.
- **Kommentar:** Die diesem Modellelement hinzugefügten Kommentare.

**Tipp**

ArgoUML ist primär kein Dokumentationssystem. Bei Modellelementen, die eine umfangreiche Dokumentation erfordern, zum Beispiel Anwendungsfälle, verwenden Sie das Feld Siehe:, um auf externe Dokumente zu verweisen. Dies ist praktischer.

### 13.5. Das Register Darstellung

Dieses Register enthält eine begrenzte Steuerung der grafischen Darstellung von Modellelementen im Diagramm des Editierbereiches.

Modellelemente, die keine spezifische grafische Repräsentation auf dem Bildschirm (neben ihrer Textbeschreibung) haben, weisen keine Register Darstellung auf. Die Darstellungs-Tabelle einer Operation einer Klasse, zum Beispiel, wird deaktiviert.
Darstellungstabellen variieren etwas von Modellelement zu Modellelement, aber Abbildung 13.8, „Eine typisches Register Darstellung im Bereich Details“ zeigt ein typisches Register Darstellung für ein Modellelement in ArgoUML (in diesem Fall eine Klasse).

**Abbildung 13.8. Eine typisches Register Darstellung im Bereich Details**

In einigen Fällen können weitere Felder vorhanden sein, z.B. für ein Paket. Aber die meisten Felder sind den Modellelementen gemeinsam.

- **Pfad** Dieses Markierfeld erlaubt es, den Pfad vor dem Namen des Modellelementes auszugeben oder nicht. Er wird in der UML-Notation durch "-"-Trenner dargestellt. Zum Beispiel würde die Klasse Main von ArgoUML wie folgt angezeigt: org::argouml::application::Main.

- **Attribute** Dieses Markierfeld erlaubt es, den Attributbereich einer Klasse ein- und auszublenden.

- **Operation** Dieses Markierfeld erlaubt es, den Operationsbereich einer Klasse oder einer Schnittstelle ein- und auszublenden.

- **Stereotyp** Dieses Markierfeld erlaubt es, die über dem Namen dargestellten Stereotypen eines Paketes ein- und auszublenden.

- **Sichtbarkeit** Dieses Markierfeld erlaubt es, die Sichtbarkeit eines Paketes ein- und auszublenden. Die Sichtbarkeit wird in der UML-Notation als +, -, # oder ~ dargestellt.

- **Erweiterungspunkte** Das Markierfeld erlaubt es Ihnen, die Erweiterungspunkte von Anwendungsfällen ein- und auszublenden.

- **Begrenzung:** Sie definiert die Ecken des 2D-Modellelementes. Es besteht aus vier, durch Kommas separierte Nummern. Diese vier Nummern sind: i) die X-Koordinate der linken oberen Ecke; ii) die Y-Koordinate der linken oberen Ecke; iii) die Breite des Feldes und iv) die Höhe des Feldes. Alle Einheiten sind Pixel im Editierbereich.

Dieses Feld hat bei 1D-Modellelementen, die andere Modellelemente verbinden (Assoziationen, Vererbungen, usw.) keine Auswirkung, da deren Position durch ihre Verbindungspartner bestimmt wird. In diesem Fall ist dieses Feld deaktiviert.

- **Füllfarbe:** Diese DropDown-Auswahl bestimmt die Füllfarbe der 2D-Modellelemente. Sie ist bei Linien-Modellelementen nicht vorhanden. Die Auswahl Keine Füllfarbe macht das Modellelement transparent. Die Auswahl Benutzerdefiniert erlaubt es andere Farben auszuwählen, als die derzeit aufgelisteten. Die Auswahl dieses Eintrages öffnet den Farbauswahl-Dialog, siehe Abbildung 13.9, „Der Dialog Benutzerdefinierte Füll-/Linienfarbe“.

- **Linienfarbe:** Diese DropDown-Auswahl bestimmt die Linienfarbe der Modellelemente. Die Auswahl Keine Linienfarbe macht das Modellelement transparent. Die Auswahl
Benutzerdefiniert erlaubt es andere Farben auszuwählen, als die derzeit aufgelisteten. Die Auswahl dieses Eintrages öffnet den Farbauswahl-Dialog, siehe Abbildung 13.9, „Der Dialog Benutzerdefinierte Füll-/Linienfarbe“.

Abbildung 13.9. Der Dialog Benutzerdefinierte Füll-/Linienfarbe

Abbildung 13.10. Der Dialog Benutzerdefinierte Füll-/Linienfarbe
Abbildung 13.11. Der Dialog Benutzerdefinierte Füll-/Linienfarbe
13.6. Das Register Quellcode


Jeder von Ihnen hinzugefügte Code geht verloren - das ist nicht die Absicht von ArgoUML - verwenden Sie stattdessen eine IDE.


13.7. Das Register Randbedingungen

Randbedingungen sind einer der in UML enthaltenen Erweiterungsmechanismen. ArgoUML ist mit einem leistungsfähigen Randbedingungsseditor ausgerüstet, der auf der im UML 1.4-Standard definierten Objekt Randbedingungsprache (Object Constraint Language; OCL) basiert.

Achtung

Die OCL-Editorimplementierung für ArgoUML V0.24 unterstützt keine OCL-Randbedingungen außer Klassen und Eigenschaften.

Dies ist manchmal eine sehr starke Einschränkung von OCL. Obwohl die UML-Spezifikation bestimmt, dass es für jedes Modellelement Randbedingungen geben kann, spezifiziert die OCL-Spezifikation diese nur für Klassen/Schnittstellen und Operationen im zulässigen Kontext.

Um die Geschichte abzukürzen, scheint es so, dass es im Moment die einfachste Lösung für ArgoUML ist, den OCL-Eigenschaftsbereich nur für solche Modellelemente freizugeben, für die es aktuell eine Definition der kontextsensitiven Klassifizierer in der OCL 1.4 gibt. Dies sind (siehe oben) Klassen/Schnittstellen und Eigenschaften.

Der Standard definierte eine kleine Anzahl von Randbedingungen vor (zum Beispiel die Bedingung \( \land \) über einen Satz von Assoziationen, was anzeigt, dass nur einer in einer bestimmten Instanz bekannt ist).

Der Standard hat auch eine Anzahl von Umständen im Auge, in denen generelle Randbedingungen hilfreich sein können:

- Um zu kennzeichnen, dass sich Klassen und Typen im Klassenmodell nicht ändern (Invarianz);
- Um zu kennzeichnen, dass sich Typen von Stereotypen nicht ändern (Invarianz);
- Um Vor- und Nachbedingungen in Operationen und Methoden zu beschreiben;
- Um Wächter zu beschreiben;
- Als Sprachnavigation; und
- um Bedingungen in Operationen zu spezifizieren.

Abbildung 13.14, „Ein typisches Register Randbedingungen im Bereich Details“ zeigt ein typisches Register Randbedingungen für ein Modellelement in ArgoUML (in diesem Fall eine Klasse).

Abbildung 13.14. Ein typisches Register Randbedingungen im Bereich Details
Im oberen Bereich des Registers befinden sich eine Reihe von Symbolen.

- **Neue Randbedingung.** Erzeugt eine neue Randbedingung und startet den Bedingungseditor im Register **Randbedingungen** für diese neue Randbedingung (siehe Abschnitt 13.7.1, „Der Bedingungs-Editor“). Die neue Randbedingung wird mit einer Kontextdeklaration für das aktuell markierte Modellelement erstellt.

  **Warnung**


- **Bedingung löschen.** Die aktuell im Feld **Name der Bedingung** markierte Bedingung (siehe unten) wird gelöscht.

  **Achtung**

  In der V0.26 von ArgoUML wird diese Schaltfläche nicht deaktiviert, wenn sie inaktiv ist, z.B., wenn keine Bedingung markiert ist.

- **Bedingung editieren.** Dies startet den Bedingungseditor im Register **Randbedingungen** (siehe Abschnitt 13.7.1, „Der Bedingungs-Editor“). Der Editor wird mit dem aktuell im Feld **Name der Bedingung** markierten Randbedingung aufgerufen.
Achtung

In der V0.26 von ArgoUML wird diese Schaltfläche nicht deaktiviert, wenn sie inaktiv ist, z.B., wenn keine Bedingung markiert ist.

• Bedingungseditor konfigurieren. Dies ist ein Dialog, um die Optionen des Bedingungseditors zu konfigurieren (siehe Abbildung 13.15, „Dialog zum Konfigurieren von Bedingungen“).

Abbildung 13.15. Dialog zum Konfigurieren von Bedingungen

Der Dialog hat ein Markierfeld für die folgende Option.

• Typkonformität von OCL-Bedingungen prüfen. OCL ist strikt typisiert. Zu frühen Designzeitpunkten kann es hilfreich sein, die Typprüfung auszuschalten. Dies ist besser, als allen benötigten, detaillierten Spezifikationen zu folgen, die erforderlich sind um Typkonsistenz zu erreichen.

Am unteren Ende befinden sich zwei Schaltflächen, die mit OK (um die Optionsänderungen zu akzeptieren) und Abbrechen (um die Änderungen rückgängig zu machen).

Der Rumpf des Registers Randbedingungen enthält zwei Felder, ein kleineres auf der linken Seite und ein größeres auf der rechten Seite. Die beiden sind durch zwei kleine Pfeil-Schaltflächen getrennt, welche die Grösse der Felder steuern.

• Linkes Feld verkleinern. Ein Taste 1-Klick auf dieses Symbol verkleinert das linke Feld. Dieser Effekt kann durch die Schaltfläche Rechtes Feld verkleinern aufgehoben werden (siehe unten).

• Rechtes Feld verkleinern. Ein Taste 1-Klick auf dieses Symbol verkleinert das rechte Feld. Dieser Effekt kann durch die Schaltfläche Linkes Feld verkleinern aufgehoben werden (siehe oben).
Eine feiner abgestimmte Steuerung kann durch eine Taste 1- Bewegung erfolgen, indem Sie die Trennungsleiste nach links und nach rechts verschieben.

Das linke Feld ist mit Name Bedingung bezeichnet und listet alle Bedingungen auf (wenn es welche gibt) so weit sie für das markierte Modellelement definiert wurden. Eine Bedingung kann durch einen Taste 1-Klick markiert werden.

Das rechte Feld ist mit Vorschau bezeichnet und enthält den Text der Bedingung. Dieses Feld zeigt nur einen Inhalt an, wenn eine Bedingung markiert wurde. Ist eine Bedingung für das Feld zu groß, erscheint rechts eine Scrollleiste.

13.7.1. Der Bedingungs-Editor

Dieser wird durch die Schaltfläche Editiere Bedingung im Register Bedingungen aufgerufen. Der Bedingungseditor nimmt das ganze Register ein (siehe Abbildung 13.16, „Dialog für das Konfigurieren der Bedingungen“).

Abbildung 13.16. Dialog für das Konfigurieren der Bedingungen

Im oberen Bereich des Registers befinden sich eine Reihe von Symbolen.

• Editieren abbrechen. Dies beendet den Bedingungseditor ohne die Änderungen abzuspeichern und kehrt zum Haupt-Bedingungen-Register zurück.

• Prüfe OCL-Syntax. Diese Schaltfläche ruft eine vollständige Syntaxprüfung der im Editor geschriebenen OCL auf. Ist die Syntax gültig, wird die Bedingung gespeichert und kehrt zum Haupt-Bedingungen-Register zurück. Wenn die Syntax ungültig ist, erläutert ein Dialogfenster das Problem.

Warnung

Ob die Typprüfung eingebunden wird, sollte mit der Schaltfläche Bedingungseditor konfigurieren einstellbar sein (siehe unten). ArgoUML V0.20 prüft aber immer und lehnt jede Bedingung mit dem kleinsten Fehler ab.

• Bedingungseditor konfigurieren. Dies ist ein Dialog, um die Optionen des Bedingungsseditors zu konfigurieren. Er ist auch im Haupt-Bedingungen-Register verfügbar und wird dort detailliert beschrieben (siehe Abschnitt 13.7, „Das Register Randbedingungen“).
Rechts von der Symbolleiste befindet sich ein Markierfeld, das als Syntaxunterstützung bezeichnet ist (standardmäßig nicht markiert). Dieses schaltet die Syntaxunterstützung im Bedingungseditor ein.

Wenn die Syntaxunterstützung eingeschaltet ist, erscheinen sechs DropDown-Menüs in einer Zeile unmittelbar unterhalb der Symbolleiste. Diese enthalten Standardtemplates für OCL, die in die editierte Bedingung eingefügt werden, sofern sie ausgewählt werden.

Die Syntaxunterstützung kann in ein separates Fenster gebracht werden, indem man mit einer Taste 1-Bewegung des kleinen, links befindlichen Trenners die Zeile mit den DropDown-Menüs aus dem Fenster bewegt.

- **Allgemein.** Allgemeine OCL Konstrukte. Einträge: `inv` (fügt eine Invarianz ein); `pre` (fügt eine Vorbedingung ein); `post` (fügt eine Nachbedingung ein); `self` (fügt eine Referenz auf sich selbst ein); `@pre` (fügt eine Referenz auf einen Wert beim Start einer Operation ein); und `result` (fügt eine Referenz auf ein vorhergehendes Ergebnis ein).

- **Operatoren.** Relationale Operatoren und Klammern. Einträge: `=;<>;<=;>=; and ()`.  

- **Nummern.** Arithmetische Operatoren und Funktionen. Einträge: `+; -; *; /; div; abs; max; min; round; und floor`. 

- **Zeichenketten.** Funktionen für Zeichenketten. Einträge: `concat; size; toLower; toUpper; and substring`. 

- **Booleans.** Logische Funktionen. Einträge: `or; and; xor; not; implies; und if then else`. 

- **Sammlungen.** Operatoren und Funktionen für Sammlungs-Mengen, Sätzen und Sequenzen. Die grosse Anzahl von Funktionen ist in Untergruppen unterteilt.

  - **Allgemein.** Funktionen, die auf alle Sammlungstypen anwendbar sind. Einträge: `Collection {};` (fügt eine neue Sammlung ein); `Set {};` (fügt einen neuen Satz ein); `Bag {};` (fügt eine neue Menge ein); `Sequence {};` (fügt eine neue Sequenz ein); `size; count; isEmpty; notEmpty; includes; includesAll; iterate; exists; forAll; collect; select; reject; union; intersection; including; excluding; und sum`. 

  - **Sätze.** Operatoren und Funktionen die nur auf Sätze anwendbar sind. Einträge: `− (set difference);` und `symmetricDifference`. 

  - **Sequenzen.** Funktionen, die nur auf Sequenzen anwendbar sind. Einträge: `first; last; at; append; prepend; und subSequence`. 

Der Rest des Registers beinhaltet ein beschreibbares Textfeld, welches den zu editierenden Text enthält. Die Maus-Schaltflächen weisen innerhalb des editierbaren Textfeldes ihr Standardverhalten auf (siehe Abschnitt 8.2, „Generelles Verhalten der Maus in ArgoUML“).


### 13.8. Das Register Stereotypen

Dieses Register zeigt die verfügbaren und anwendbaren Stereotypen für das aktuell markierte Modelllement. Es besteht aus zwei Feldern und zwei Schaltflächen. Die Schaltflächen erlauben es, die Stereotypen von einer Liste in die andere zu bewegen.

In den Listen, zwischen [], wird die Basisklasse der Stereotypen angezeigt. Zum Beispiel in dem obigen Bild kann der Stereotyp thread[Classifier] auf alle Typen von Klassifizierern wie Klassen, Anwendungsfälle, ... angewendet werden.

13.9. Das Register Eigenschaftswerte

Eigenschaftswerte sind ein anderer Erweiterungsmechanismus der in UML enthalten ist. Der Anwender kann Name-Wert-Paare definieren, die mit Modellelementen verknüpft sind und die Eigenschaften dieses Modells definieren. Die Namen sind als tags bekannt. UML definiert eine Anzahl von Tags vor, die für viele seiner Modellelemente nützlich sind.

Anmerkung

Die Tag-Dokumentation ist für die Top-UML- Metaklasse Element definiert und ist so in allen Modellelementen verfügbar. In ArgoUML sind Dokumentationswerte im Register Dokumentation enthalten, so dass diese nicht mit Hilfe der Eigenschaftswerte definiert werden müssen.

Das Register Eigenschaftswerte in ArgoUML umfasst eine zweispaltige Tabelle mit einem Kombinationsfeld links, um die Tag-Definition in einem editierbaren Feld auszuwählen und rechts für den damit verknüpften Wert. Es gibt immer mindestens eine leere Zeile, die für einen neuen Tag bereit steht.

Die Schaltfläche oben in diesem Register erlaubt das Erzeugen einer neuen Tag-Definition. Nach dem Anklicken dieser Schaltfläche gehen Sie zuerst in das Register Eigenschaften, um den Namen der neuen Tag-Definition festzulegen.

die Maus-Schaltflächen weisen ihr Standardverhalten innerhalb des editierbaren Wertebereiches auf (siehe Abschnitt 8.2, „Generelles Verhalten der Maus in ArgoUML“). Zusätzlich können Sie auch die
Tastenkürzel **Strg-X, Strg-C** und **Strg-V** in einem Wertefeld aufrufen.

## 13.10. Das Register Checkliste


![Checkliste Beispiel](image)

#### Achtung

In der Release V0.22 von ArgoUML ist dieses Register nicht vollständig implementiert. Zum Beispiel werden die Markierungen nicht gespeichert.
Kapitel 14. Der Bereich Zu-Bearbeiten

14.1. Einleitung


Abbildung 14.1. Überblick über den Bereich Zu-Bearbeiten

Dieser Bereich hat Zugriff auf die Ergebnisse des Kritikenprozesses, der innerhalb von ArgoUML abläuft.

Ein Auswahlfeld oben erlaubt die Einstellung, wie die Daten dargestellt werden, eine Schaltfläche erlaubt es, die Darstellung der Hierarchie zu ändern und es gibt eine Anzeige, wie viele Zu-Bearbeiten-Elemente identifiziert wurden.

Mehr Informationen über Kritiken können Sie in der Diskussion des Kritiken-Menüs (siehe Abschnitt 10.9, „Das Menü Kritiken“) finden.

14.2. Das Verhalten der Maus im Bereich Zu-Bearbeiten
Das generelle Verhalten der Maus und die Benennung der Tasten ist im Kapitel über die gesamte Anwenderschnittstelle ausgeführt (siehe Kapitel 8, Einleitung).

14.2.1. Taste 1-Klick

Diese Aktion wird generell dazu verwendet, ein Element für darauf folgende Operationen zu markieren.

Innerhalb der hierarchischen Anzeige, können Elemente mit Unter-Hierarchien durch angezeigt werden, wenn die Hierarchie versteckt und wenn die Hierarchie geöffnet ist.

Wenn diese Symbole angezeigt werden, wechselt die Anzeige der Hierarchie bei jedem Taste 1-Klick auf diese Symbole.

Ein Taste 1-Klick über der Überschrift eines Zu-Bearbeiten-Elementes wird dessen Details im Register Zu-Bearbeiten-Element des Bereiches Details anzeigen. Dieses Register wird automatisch ausgewählt, wenn es aktuell nicht sichtbar ist.

14.2.2. Taste 1-Doppelklick

Wenn dies auf ein Verzeichnissymbol in der Hierarchie ausgeführt wird, wird sich die Darstellung dieser Hierarchie ändern.

Wenn dies auf eine Überschrift ausgeführt wird, wird der Taste 1-Doppelklick das Diagramm für das Modellelement anzeigen, zu dem das Zu-Bearbeiten-Element gehört und das Modellelement im Diagramm mit Hilfe des entsprechenden Elementes markieren (das Modellelement kann hervorgehoben, mit einer Wellenlinie unterstrichen oder mit einem farbigen Rahmen umgeben sein).

14.2.3. Taste 2-Aktionen

Es gibt keine Taste 2-Funktionen im Bereich Zu-Bearbeiten.

14.2.4. Taste 2-Doppelklick

Es gibt keine Taste 2-Funktionen im Bereich Zu-Bearbeiten.

14.3. Auswahl der Darstellung

Oben im Bereich befindet sich ein Kombinationsfeld, was die Darstellung der Zu-Bearbeiten-Elemente steuert. Die Zu-Bearbeiten-Elemente können auf sechs unterschiedliche Arten dargestellt werden. Diese Einstellung wird nicht dauerhaft gespeichert. Zum Beispiel ist sie auf den Standardwert beim Start von ArgoUML eingestellt.

• Nach Priorität. Dies ist die Standardeinstellung. Die Zu-Bearbeiten-Elemente werden nach Priorität in drei Hierarchien organisiert: Hoch, Mittel und Niedrig. Die Priorität, die mit den durch eine bestimmte Kritik generierten Zu-Bearbeiten-Elementen verknüpft sind, können über das Menü Kritiken > Kritiken anzeigen... geändert werden (siehe Abschnitt 10.9.4, „Kritiken anzeigen...“).

Die Details der Kritiken in jeder Kategorie werden in Abschnitt 10.9.2, „Design-Wichtungen...“ diskutiert.

- **Nach Ziel.** ArgoUML verfolgt das Konzept, dass Kritiken entsprechend der sie beeinflussenden Benutzerziele gruppiert werden. Diese Darstellung gruppiert die Zu-Bearbeiten-Elemente in einer Hierarchie entsprechend der Ziele.

  **Achtung**


- **Nach Kurzbezeichnung.** Die Zu-Bearbeiten-Elemente werden danach hierarchisch organisiert, welche Kritik das Zu-Bearbeiten-Element generierte. Der Klassenname der Kritik wird anstelle seiner Überschrift aufgelistet.


### 14.4. Element-Zähler

Rechts von der Schaltfläche flach/hierarchisch befindet sich ein Zähler, der die Anzahl der aktuell gefundenen Zu-Bearbeiten-Elemente ausgibt. Er wird gelb hervorgehoben, wenn die Anzahl der Zu-Bearbeiten-Elemente auf über 50 Zu-Bearbeiten-Elemente anwächst und rot, wenn es über 100 sind.
Kapitel 15. Die Kritiken

15.1. Einleitung


Kritiken sind einer der Hauptarten mit denen diese Ideen implementiert wurden. Im Hintergrund laufend, bieten Sie dem Designer Ratschläge an, die akzeptiert oder ignoriert werden können. Der Schlüsselpunkt ist, dass sie keine Entscheidung des Designers ausschliessen.

Anmerkung

Die Kritiken sind asynchrone Prozesse, die parallel zu ArgoUML ablaufen. Änderungen benötigen eine oder zwei Sekunden, bis die Kritiken erneut erzeugt wurden.

15.1.1. Terminologie

Die Kritiken sind Hintergrundprozesse, die das aktuelle Modell anhand verschiedener „guter“ Designkriterien überprüfen. Es gibt eine Kritik für jedes Designkriterium.

Die Ausgabe einer Kritik ist eine Kritik - eine Anweisung über einige Aspekte des Modelles, die nicht der guten Designpraxis folgen.

Natürlich wird eine Kritik nur vorschlagen, wie der gefundene Designmangel behoben werden kann, in dem es ein Zu-Bearbeiten-Element erzeugt.

15.1.2. Design-Mangel


In diesem Handbuch sind die Beschreibungen der Kritiken entsprechend ihres Designmangels in Abschnitten gruppiert.

15.2. Unkategorisiert

Diese Kritiken passen zu keiner der anderen Kategorien.


15.3. Klassenauswahl

Dies sind Kritiken, die sich darauf beziehen, wie Klassen ausgewählt und verwendet wurden.

ArgoUML hat die folgenden Kritiken in dieser Kategorie.
15.3.1. Datentyp verbergen

Datentypen sind innerhalb von UML 1.4 keine vollständigen Klassen. Sie können nur Aufzählungsliterale als Werte haben und nur Abfrage-Operationen unterstützen (Das sind Operationen, die nicht den Zustand des Datentyps verändern).


Eine gute OOA&D hängt von der richtigen Auswahl der Entities ab, welche die vollständigen Objekte und welche die Attribute von Objekten repräsentieren.

Es gibt zwei Optionen, um dieses Problem zu lösen.

- Ersetze den Datentyp durch eine ganze Klasse.
- oder ändere die Aggregation in eine Komposition am Ende das Datentyps.

15.3.2. Verringere die Anzahl der Klassen im Namensraum <Namensraum>


Der Assistent dieser Kritik erlaubt das Setzen eines Schwellwertes, z.B. die maximale Anzahl von Klassen, ab der die Kritik ausgelöst wird.

**Achtung**


15.3.3. Diagramm aufräumen

Vorschlag, dass das Diagramm verbessert werden sollte, indem man sich überlappende Modellelemente auseinander zieht.

15.4. Benennung

Dies sind Kritiken, die sich auf die Benennung von Modellelementen beziehen. Die aktuelle Version von ArgoUML hat 18 Kritiken in dieser Kategorie.

15.4.1. Assoziations-Namenskonflikt auflösen

Hinweis, dass zwei Assoziationen im gleichen Namensraum den gleichen Namen haben. Dies ist in UML nicht erlaubt.
15.4.2. Überarbeite die Attributnamen, um einen Konflikt zu vermeiden

Hinweis, dass zwei Attribute einer Klasse den gleichen Namen haben. Dies ist in UML nicht erlaubt.

Anmerkung

Das Problem kann durch vererben eines Attributes über eine Vererbungsbeziehung verursacht werden.

15.4.3. Ändere Namen oder Signaturen in einem Modellelement

Zwei Operationen in einem <Modellelement> haben die gleiche Signatur. Das heißt, die Namen sind gleich und die Liste der Parameter hat den gleichen Typ.


Beim Vergleichen von Signaturen betrachtet die Kritik:

1. den Namen;
2. die Liste der Ein-, Aus- und Ein-Ausgabeparametertypen in ihrer Reihenfolge;

Nur wenn diese sowohl in Typ und Reihenfolge übereinstimmen, wird die Signatur als gleich betrachtet.


Anmerkung


15.4.4. Doppelte End- (Rollen-) Namen in einer Assoziation

Die angegebene Assoziation hat zwei (oder mehrere) Enden (Rollen) mit dem gleichen Namen. Eine der wohlgeformten Regeln für Assoziationen in UML 1.4 ist, dass alle Enden (Rollen)-Namen eindeutig sein müssen.

Dies stellt sicher, dass es keine mehrdeutige Referenz auf die Enden einer Assoziation geben kann.

Um dies zu beheben, markieren Sie die Assoziation manuell und ändern die Namen von einer oder mehreren der betroffenen Enden (Rollen) mit Hilfe des Taste 2-Popup-Menüs oder der Eigenschaftstabelle.
15.4.5. Rollenname steht im Konflikt mit einem Element

Ein Hinweis, dass ein gutes Design Rollenname für Assoziationen verhindert, die mit Attributen oder Operationen der Quellklasse kollidieren. Rollen können im Code als Attribute oder Operationen realisiert werden, die dann Codegenerierungsprobleme auslösen.

15.4.6. Einen Namen auswählen (Klassen und Schnittstellen)

Der betrachtete Klasse oder Schnittstelle wurde kein Name gegeben (sie wird im Modell als Unbenannt erscheinen). Hinweis, dass gutes Design fordert, dass alle Schnittstellen und Klassen benannt werden.

15.4.7. Namenskonflikt in einem Namensraum

Der betrachtete Namensraum (z.B. Paket) enthält eine Klasse oder eine Schnittstelle mit dem gleichen Namen wie eine andere (im gleichen Namensraum). Dies ist schlechtes Design und verhindert das generieren gültigen Codes.

15.4.8. Wählen Sie einen eindeutigen Namen für ein Modellelement aus (Klassen und Schnittstellen)

Hinweis, dass die angegebene Klasse oder Schnittstelle den gleichen Namen wie eine andere (im Namensraum) aufweist. Dies ist schlechtes Design und wird eine gültige Codegenerierung verhindern.

15.4.9. Wählen Sie einen Namen aus (Attribute)

Das betrachtete Attribut hat keinen Namen erhalten (es wird im Modell als (Unbenanntes Attribute) erscheinen). Hinweis, dass gutes Design fordert, dass alle Attribute benannt werden.

15.4.10. Wählen Sie einen Namen aus (Operationen)

Die betrachtete Operation hat keinen Namen erhalten (sie wird im Modell als (Unbenannte Operation) erscheinen). Hinweis, dass gutes Design fordert, dass alle Operationen benannt werden.

15.4.11. Wählen Sie einen Namen aus (Zustände)


15.4.12. Wählen Sie einen eindeutigen Namen für ein (zustandsbehaftetes) Modellelement aus.

Hinweis, dass der angegebene Zustand den gleichen Namen wie ein anderer aufweist (im aktuellen Zustandsdiagramm). Dies ist schlechtes Design und wird die gültige Codegenerierung verhindern.

15.4.13. Ändern Sie den Namen, um eine Konfusion zu verhindern

Zwei Namen im gleichen Namensraum haben sehr ähnliche Namen (sie unterscheiden sich nur durch ein
Die Kritiken


Achtung

Diese Kritik kann manchmal störend sein, da es manchmal nützlich und gutes Design ist, eine Serie von Modellelementen var1, var2 usw. zu haben.

Es ist wichtig sich daran zu erinnern, dass Kritiken Anleitungen anbieten, die nicht immer korrekt sein müssen. ArgoUML läßt es zu, dass Sie die entsprechenden Zu-Bearbeiten-Elemente über den Zu-Bearbeiten-Bereich verlassen (siehe Kapitel 14, Der Bereich Zu-Bearbeiten).

15.4.14. Wählen Sie einen gültigen Namen aus


15.4.15. Ändern Sie den Namen des Modellelementes in ein nicht-reserviertes Wort

Hinweis, dass der Name dieses Modellelementes dem Namen eines in UML resevierten Wortes entspricht (or within one character of one). Dies ist nicht erlaubt.

15.4.16. Wählen Sie einen besseren Namen für die Operation aus

Hinweis, dass der Name einer Operation nicht der Namenskonvention entspricht, dass Namen von Operationen mit einem Kleinbuchstaben beginnen.

Achtung

Der Java und C++-Konvention folgend, geben die meisten Designer Ihren Konstruktoren den gleichen Namen wie der Klasse, die mit einem Grossbuchstaben beginnt. In ArgoUML wird dies diese Kritik auslösen, es sei denn, der Konstruktor erhält den Stereotyp «create».

Es ist wichtig sich daran zu erinnern, dass Kritiken Anleitungen anbieten, die nicht immer korrekt sein müssen. ArgoUML läßt es zu, dass Sie die entsprechenden Zu-Bearbeiten-Elemente über den Zu-Bearbeiten-Bereich verlassen (siehe Kapitel 14, Der Bereich Zu-Bearbeiten).

15.4.17. Wählen Sie einen besseren Attributnamen aus

Hinweis, dass ein Attribut nicht der Namenskonvention entspricht, dass Namen von Attributen mit einem Kleinbuchstaben beginnen.

15.4.18. Klassenname groß schreiben

Hinweis, dass eine Klasse nicht der Namenskonvention entspricht, dass Klassen mit einem Großbuchstaben beginnen.

196
Anmerkung

Obwohl sie diese Kritik nicht auslöst, sollte die gleiche Konvention auf Schnittstellen angewendet werden.

15.4.19. Paketname überarbeiten

Hinweis, dass ein Paket nicht der Namenskonvention entspricht, dass Kleinbuchstaben mit Punkten verwendet werden, um Subpakete zu kennzeichnen.

15.5. Speicher

Kritiken, die sich auf die Attribute von Klassen beziehen.

Die aktuelle Version von ArgoUML hat in dieser Kategorie die folgenden Kritiken.

15.5.1. Überarbeiten Sie die Attributnamen, um einen Konflikt zu vermeiden

Diese Kritik wurde in einer früheren Designmangel-Kategorie diskutiert (siehe Abschnitt 15.4.2, ,, Überarbeite die Attributnamen, um einen Konflikt zu vermeiden ").

15.5.2. Fügen Sie Instanzvariablen zu einer Klasse hinzu

Hinweis, dass für die angegebene Klasse keine Instanzvariable spezifiziert wurde. Solche Klassen können erzeugt werden, um statische Attribute und Methoden zu spezifizieren, aber sie sollten per Konvention das Stereotyp «utility» erhalten.

15.5.3. Fügen Sie der Klasse einen Konstruktor hinzu


Das Definieren guter Konstruktoren ist der Schlüssel für das Einrichten unveränderlicher Klassen und unveränderliche Klassen sind eine leistungsfähige Hilfe beim Schreiben soliden Codes.

Um dies zu beheben, fügen Sie manuell einen Konstruktor hinzu, indem Sie im Explorer auf die Klasse klicken und eine Operation mit Hilfe des kontextsensitiven Popup-Menüs des Eigenschaftsregisters hinzufügen, oder die Klasse im Diagramm markieren und das Werkzeug Operation hinzufügen verwenden.

Im UML 1.4 Standard ist ein Konstruktor eine Operation mit dem Stereotypen «create». Obwohl kein strikter Standard, wird ArgoUML auch «Create» als Stereotyp für Konstruktoren akzeptieren.

Gemäß Java und C++-Konvention hat ein Konstruktor den gleichen Namen wie die Klasse, ist nicht statisch und gibt keinen Wert zurück. ArgoUML wird auch jede Operation akzeptieren, die diesen Konventionen eines Konstruktors folgt, auch wenn sie nicht den Stereotypen «create» aufweist.

Achtung

Operatoren werden in ArgoUML mit einem Standard-Rückgabeparameter (return
genannt) erzeugt. Sie müssen diesen Parameter entfernen, um der Java/C++-Konvention zu entsprechen.

15.5.4. Reduzieren Sie die Zahl der Attribute in der Klasse

Hinweis, dass die Klasse für ein gutes Design zu viele Attribute aufweist und das Risiko eines Designengpasß in sich birgt.

Der Assistent dieser Kritik erlaubt das Einstellen eines Schwellwertes, z.B. die maximal erlaubte Anzahl von Attributen bevor diese Kritik ausgelöst wird.

Achtung

Diese Anzahl wird nicht dauerhaft gespeichert und es gibt keinen Weg, diese zu reduzieren, nachdem sie hochgesetzt wurde. Es sei denn, man erzeugt mehr Attribute, bis die Kritik erneut ausgelöst wird. Der Neustart von ArgoUML setzt dieses Anzahl wieder auf seinen Standardwert: 7.

15.6. Geplante Erweiterungen

Kritiken, die sich auf Schnittstellen und Subklassen beziehen.

Anmerkung

Es ist nicht klar, warum diese Kategorie den Namen „Geplante Erweiterungen“ hat.

Die aktuelle Version von ArgoUML hat drei Kritiken in dieser Kategorie.

15.6.1. Operationen in Schnittstellen müssen public sein

Hinweis, dass es keinen Punkt gibt, non-public Operationen in Schnittstellen haben zu müssen, da sie in einer realisierten Klasse sichtbar sein müssen.

15.6.2. Schnittstellen dürfen nur Operationen haben

Hinweis, dass in einer Schnittstelle Attribute definiert wurden. Der UML-Standard definiert Schnittstellen nur mit Operationen.

Achtung

ArgoUML erlaubt es Ihnen nicht, Schnittstellen Attribute hinzuzufügen, sodaß dies in einem ArgoUML-Modell niemals auftreten sollte. Sie kann ausgelöst werden, wenn ein Projekt mit XMI geladen wurde, die durch ein anderes Werkzeug erzeugt wurde.

15.6.3. Entferne die Referenz auf die spezifische Subklasse
Hinweis, dass eine Klasse in einem guten Design seine Subklassen nicht direkt über Attribute, Operationen oder Assoziationen referenzieren soll.

15.7. Zustandsautomaten

Kritiken, die sich auf Zustandsautomaten beziehen.

ArgoUML hat die folgenden Kritiken in dieser Kategorie.

15.7.1. Reduzieren Sie die Anzahl der Transitionen im <Zustand>

Hinweis, dass der angegebene Zustand so viele Transitionen aufweist, dass er zu einem Leistungsengpass werden dürfte.

Der Assistenz dieser Kritik erlaubt das Einstellen eines Schwellwertes, z.B. die maximale Anzahl von Transitionen bevor diese Kritik ausgelöst wird.

Achtung


15.7.2. Reduzieren Sie die Anzahl der Zustände im Automaten <Automat>

Hinweis, dass der angegebene Zustandsautomat so viele Zustände aufweist, dass er bereits Konfusion auslöst und vereinfacht (vielleicht durch unterteilen in mehrere Automaten, oder durch Nutzung einer Hierarchie) werden sollte.

Der Assistenz dieser Kritik erlaubt die Einstellung eines Schwellwertes, z.B. die maximale Anzahl von erlaubten Zuständen bevor diese Kritik ausgelöst wird.

Achtung


15.7.3. Fügen Sie dem <Zustand> Transitionen hinzu

Hinweis, dass der angegebene Zustand ankommende und abgehende Transitionen benötigt.

15.7.4. Fügen Sie dem Modellelement <Modellelement> ankommende Transitionen hinzu

Hinweis, dass der angegebene Zustand ankommende Transitionen benötigt.
15.7.5. Fügen Sie dem Modellelement <Modellelement> abgehende Transitionen hinzu

Hinweis, daß der angegebene Zustand abgehende Transitionen benötigt.

15.7.6. Entfernen Sie den zusätzlichen Initialzustand

Hinweis, daß es mehr als einen Initialzustand in dem Zustandsautomaten oder dem zusammengesetzten Zustand gibt, was in UML nicht erlaubt ist.

15.7.7. Fügen Sie einen initialen Zustand ein

Hinweis, daß es keinen initialen Zustand in dem Zustandsautomaten oder dem zusammengesetzten Zustand gibt.

15.7.8. Fügen Sie der Transition ein Signal oder einen Wächter hinzu

Hinweis, dass eine Transition entweder ein Signal oder einen Wächter vermisst. Einer davon ist mindestens notwendig.

15.7.9. Ändere Vereinigungs-Transitionen

Hinweis, dass der Pseudozustand "Vereinigen" eine ungültige Anzahl von Transitionen aufweist. Normalerweise sollten es eine abgehende und zwei oder mehrere ankommende sein.

15.7.10. Ändere Gabelungs-Transitionen

Hinweis, dass der Pseudozustand "Gabelung" eine ungültige Anzahl von Transitionen aufweist. Normalerweise sollten es eine ankommende und zwei oder mehrere abgehende sein.

15.7.11. Fügen Sie Entscheidungs-/Kreuzungstransitionen hinzu

Hinweis, dass der Pseudozustands-Zweig (Entscheidung oder Kreuzung) eine ungültige Zahl von Transitionen aufweist. Normalerweise sollten es mindestens eine ankommende Transition und mindestens eine abgehende Transition sein.

15.7.12. Fügen Sie der Transition einen Wächter hinzu

Hinweis, dass die Transition einen Wächter benötigt.

**Achtung**

Es ist nicht klar, ob dies eine gültige Kritik ist. Es ist sicherlich akzeptabel, eine Transition ohne Wächter zu haben - die Transition wird immer genommen, wenn das Signal ausgelöst wird.

15.7.13. Das Diagramm aufräumen
Diese Kritik wurde unter einer früheren Design-Kategorie diskutiert (siehe Abschnitt 15.3.3, „Diagramm aufräumen“).

15.7.14. Eine Kante sichtbarer machen

Hinweis, dass ein Kanten-Modellelement, wie zum Beispiel eine Assoziation oder Abstraktion so kurz ist, dass sie vermisst werden könnte. Ziehen Sie die verbundenen Modellelemente auseinander, um die Kanten sichtbarer zu machen.

15.7.15. Zusammengesetztes Assoziationsende mit der Kardinalität >1

Eine Instanz kann durch eine Komposition nicht zu mehr als einer Kompositions-Instanz gehören. Sie müssen die Kardinalität am zusammengesetzten Ende der Assoziation entweder auf 0..1 oder 1..1 setzen, damit das Modell Sinn macht.

Erinnern Sie sich, dass die Komposition die strengere Aggregationsart ist und die Aggregation schwächer ist. Das Problem kann mit einem Modell verglichen werden, in dem ein Finger der integrale Teil von mehreren Händen gleichzeitig sein kann.

Dies ist die zweite wohlgeformte Regel bei Assoziationsenden in UML 1.4.

15.8. Designmuster

Kritiken über die Nutzung von Designmustern in ArgoUML.


Die Klasse hat keine nicht-statischen Attribute noch Assoziationen, die von der Instanz dieser Klasse abgehen. Das bedeutet, dass jede Instanz dieser Klasse identisch mit jeder anderen Instanz sein wird, da es nichts über die Instanz gibt, was sie voneinander unterscheidbar macht.


Wenn es nicht Ihre Absicht ist nur eine Instanz zu haben, sollten Sie Instanzvariablen definieren (z.B. nicht-statische Attribute) und/oder abgehende Assoziationen, welche die Unterschiede zwischen den Instanzen repräsentieren.

Wenn Sie eine Klasse als Singleton spezifiziert haben, müssen Sie die Klasse so definieren, dass sie nur eine Instanz haben kann. Dies wird die Informationsdarstellung Ihres Designs vervollständigen. Um dies zu erreichen, müssen Sie folgendes tun.

2. Sie dürfen nur einen privaten Konstruktor haben, so dass keine neuen Instanzen durch anderen Code erzeugt werden kann. Das Erzeugen der einzigen Instanz kann durch eine passende Hilfsoperation erfolgen, die diesen privaten Konstruktor genau einmal aufruft.

3. Sie müssen mindestens einen Konstruktor haben, um den Standardkonstruktor zu überschreiben, so dass der Standardkonstruktor nicht dazu verwendet wird mehrere Instanzen zu erzeugen.

Die Definition eines Konstruktors gemäß UML-Standard 1.4 und dessen Erweiterungen, so dass die Definition durch ArgoUML akzeptiert wird, siehe Abschnitt 15.5.3, „Fügen Sie der Klasse einen Konstruktor hinzu“.

15.8.2. Singleton Stereotyp-Verletzung in <Klasse>


Immer, wenn Sie eine Klasse mit einem Stereotypen kennzeichnen, sollte die Klasse allen Bedingungen dieses Stereotyps entsprechen. Die ist ein wichtiger Teil bei der Erstellung eines konsistenten und verständlichen Designs. Die Nutzung des Singletonmusters kann Zeit und Speicherplatz einsparen.

Wenn Sie diese Klasse nicht länger als Singleton benötigen, entfernen Sie den Stereotypen «singleton» indem Sie auf die Klasse klicken und die leere Auswahl im Stereotyp-Kombinationsfeld des Registers Eigenschaften auswählen.

Beider Anwendung eines Singletonmusters sollten Sie den Anweisungen in Abschnitt 15.8.1, „Ziehen Sie Nutzung des Singleton-Musters für eine <class> in Betracht.“ folgen.

15.8.3. Knoten haben normalerweise keine Hülle

Ein Hinweis, dass Knoten nicht innerhalb anderer Modellelemente im Verteilungsdiagramm eingezeichnet werden sollten, da sie ein autonomies physikalisches Objekt repräsentieren.

15.8.4. Knoteninstanzen haben normalerweise keine Hülle

Ein Hinweis, dass Knoteninstanzen nicht innerhalb anderer Modellelemente im Verteilungsdiagramm eingezeichnet werden sollten, da sie ein autonomies physikalisches Objekt repräsentieren.

15.8.5. Komponenten befinden sich normalerweise innerhalb von Knoten

Ein Hinweis, dass Komponenten logische Entitäten innerhalb physikalischer Knoten repräsentieren und innerhalb eines Knoten eingezeichnet werden sollten, wobei Knoten in einem Verteilungsdiagramm dargestellt werden.

15.8.6. Komponenteninstanzen befinden sich normalerweise innerhalb von Knoten

Ein Hinweis, dass Komponenteninstanzen logische Entitäten innerhalb physikalischer Knoten...
repräsentieren und innerhalb einer Knoteninstanz eingezeichnet werden sollten, wobei Knoteninstanzen in einem Verteilungsdiagramm dargestellt werden.

15.8.7. Klassen befinden sich normalerweise innerhalb von Komponenten

Ein Hinweis, dass Klassen als Modellelemente Komponenten bilden und innerhalb von Komponenten in Verteilungsdiagrammen eingezeichnet werden sollten.

15.8.8. Schnittstellen befinden sich normalerweise innerhalb von Komponenten

Ein Hinweis, dass Schnittstellen als Modellelemente Komponenten bilden und innerhalb von Komponenten in Verteilungsdiagrammen eingezeichnet werden sollten.

15.8.9. Objekte befinden sich normalerweise innerhalb von Komponenten

Ein Hinweis, dass Objekte als Instanzen von Modellelementen Komponenten bilden, die innerhalb von Komponenten oder Komponenteninstanzen in Verteilungsdiagrammen eingezeichnet werden sollten.

15.8.10. Verknüpfungsenden haben nicht die gleiche Ebene


15.8.11. Klassifizierung einstellen (Verteilungsdiagramm)

Hinweis, dass es in einem Verteilungsdiagramm eine Instanz (Objekt) ohne eine verknüpfte Klassifizierung gibt (Klasse, Datentyp).

15.8.12. Return-Aktionen werden vermisst

Hinweis, dass ein Sequenzdiagramm eine Sende- oder Aufrufaktion ohne entsprechende Return-Aktion enthält.

15.8.13. Vermisse Aufruf(Sende)-Aktion

Hinweis, dass ein Sequenzdiagramm eine Return-Aktion enthält, aber keine vorhergehende Aufruf- oder Sende-Aktion.

15.8.14. Kein Auslöseimpuls bei diesen Verknüpfungen

Hinweis, dass ein Sequenzdiagramm eine Objekte verbindende Verknüpfung ohne verknüpften Auslöseimpuls aufweist (ohne den die Verknüpfung bedeutungslos ist).

Warnung

15.8.15. Klassifizierung einstellen (Sequenzdiagramm)

Hinweis, dass es in einem Sequenzdiagramm ein Objekt ohne eine damit verknüpfte Klassifizierung (Klasse, Datentyp) gibt.

15.8.16. Falsche Position dieses Auslöseimpulses

Hinweis, dass in einem Sequenzdiagramm die Initiierung eines Sende/Aufruf-Return-Nachrichtaustausches nicht richtig von links nach rechts initiiert wurde.

15.9. Beziehungen

Kritiken, die sich in ArgoUML auf Assoziationen beziehen.

Die aktuelle Version von ArgoUML hat die folgenden Kritiken in dieser Kategorie.

15.9.1. Zirkuläre Assoziation

Hinweis, dass eine Assoziationsklasse eine Rolle aufweist, die auf sich selbst verweist. Dies ist nicht erlaubt.

Warnung

Diese Kritik ist in der V0.14 von ArgoUML bedeutungslos, da sie Assoziationsklassen nicht unterstützt.

15.9.2. <Assoziation> navigierbar machen


15.9.3. Entferne die Navigation von der Schnittstelle via <Assoziation>

Assoziationen die eine Schnittstelle beinhalten können nicht von der Schnittstelle aus navigierbar sein. Dies ist so, weil Schnittstellen nur Operationsdeklarationen enthalten und keine Zeiger auf andere Objekte halten können.


Um dies zu beheben markieren Sie die Assoziation und nutzen das Register Eigenschaften, um nach und nach jedes Assoziationsende auszuwählen, das nicht mit der Schnittstelle verbunden ist.

204
Entfernen Sie die Markierung Navigierbar für jedes dieser Enden.

Die Assoziation sollte dann mit einem Pfeil in Richtung der Schnittstelle erscheinen.


15.9.4. Fügen Sie dem <Modellelement> eine Assoziation hinzu

Hinweis, dass das spezifizierte Modellelement (Akteur, Anwendungsfall oder Klasse) keine verbindenden Assoziationen zu anderen Modellelementen aufweist. Dies ist aber erforderlich, wenn das Modellelement in einem Design nützlich sein soll.

15.9.5. Entfernen Sie die Referenz auf die spezifische Subklasse

Diese Kritik wird unter einer früheren Designkategorie diskutiert (siehe Abschnitt 15.6.3, „Entferne die Referenz auf die spezifische Subklasse“).

15.9.6. Reduzieren Sie die Assoziationen des <Modellelementes>

Hinweis, dass das betreffende Modellelement (Akteur, Anwendungsfall, Klasse oder Schnittstelle) so viele Assoziationen aufweist, dass diese zu einem Wartungsengpass führen können.

Der Assistent dieser Kritik erlaubt das Einstellen eines Schwellwertes, z.B. die maximale Anzahl von erlaubten Assoziationen bevor diese Kritik ausgelöst wird.

Achtung


15.9.7. Kante sichtbarer machen

Diese Kritik wird in einer früheren Designkategorie diskutiert (siehe Abschnitt 15.7.14, „Eine Kante sichtbarer machen“).

15.10. Instanzen bilden

Kritiken, die sich auf das Bilden von Instanzen bei Klassifizierern in ArgoUML beziehen.


15.11. Modularität

Kritiken, die sich auf die modulare Entwicklung in ArgoUML beziehen.
Die aktuelle Version von ArgoUML hat die folgenden Kritiken in dieser Kategorie.

15.11.1. Der Klassifizierer befindet sich nicht im Namensraum seiner Assoziation.

Eine der wohlgeformten Regeln in UML 1.4 für Assoziationen lautet, dass alle Klassifizierer, die den Enden einer Assoziation zugewiesen werden, zum gleichen Namensraum gehören müssen wie die Assoziation.

Wenn dies nicht der Fall wäre, würde es keine Bezeichnung geben, über das jedes Ende alle anderen referenzieren kann.

Diese Kritik wird ausgelöst, wenn eine Assoziation nicht mit diesem Kriterium übereinstimmt. Die Lösung ist, die Assoziation zu löschen und im Diagramm neu zu erzeugen, sodass der Namensraum alle zugewiesenen Klassifizierer enthält.

Achtung


15.11.2. Fügen Sie Elemente zum Paket <Paket> hinzu.

Hinweis, dass das angegebene Paket keinen Inhalt hat. Gutes Design weist Pakete auf, die erzeugt wurden, Dinge hinein zu tun.

Anmerkung

Diese Kritik wird immer ausgelöst, wenn Sie erstmalig ein Paket erzeugen, da Sie kein Paket erzeugen können, das nicht leer ist.

15.12. Erwartete Verwendung

Kritiken, die sich auf eine generell akzeptierte, gute Praxis beziehen.

Die aktuelle Version von ArgoUML hat eine Kritik in dieser Kategorie.

15.12.1. Diagramm aufräumen

Diese Kritik wird in einer früheren Designkategorie diskutiert (siehe Abschnitt 15.3.3, „Diagramm aufräumen“).

15.13. Methoden

Kritiken, die sich auf Operationen in ArgoUML beziehen.

Die aktuelle Version von ArgoUML hat die folgenden Kritiken in dieser Kategorie.
15.13.1. Ändere Namen oder Signaturen im <Modellelement>

Diese Kritik wird in einer früheren Designkategorie diskutiert (siehe Abschnitt 15.4.3, „Ändere Namen oder Signaturen in einem Modellelement“).

15.13.2. Die Klasse muß abstrakt sein

Hinweis, dass eine Klasse, die abstrakte Operationen erbt oder definiert als abstrakte Klasse bezeichnet sein muß.

15.13.3. Fügen Sie der <Klasse> Operationen hinzu

Hinweis, dass in der angegebenen Klasse keine Operationen definiert wurden. Dies ist für die Klasse aber erforderlich, damit sie im Design einen Nutzen hat.

15.13.4. Reduzieren Sie die Anzahl der Operationen im <Modellelement>

Hinweis; dass das Modellelement (Klasse oder Schnittstelle) zu viele Operationen aufweist, um gutem Design zu entsprechen. Darüber hinaus beinhaltet es das Risiko, zum Design-Wartungsgap zu werden.

Der Assistent dieser Kritik erlaubt das Setzen eines Schwellwertes, z.B. die maximale Anzahl von erlaubten Operationen bevor diese Kritik ausgelöst wird.

Achtung


Kritiken, die sich auf die Code-Generierung in ArgoUML beziehen.

Die aktuelle Version von ArgoUML hat eine Kritik in dieser Kategorie.

15.14.1. Ändern Sie die Mehrfachvererbung in Schnittstellen

Hinweis, dass eine Klasse mehrere Vererbungen aufweist, was in UML erlaubt ist, aber nicht in Java-Code umgesetzt werden kann, da Java keine Mehrfachvererbung unterstützt.

15.15. Stereotypen

Kritiken, die sich auf Stereotypen in ArgoUML beziehen.

15.16. Vererbung

Kritiken, die sich mit der Generalisierung und Spezialisierung in ArgoUML befassen.

Die aktuelle Version von ArgoUML hat die folgenden Kritiken in dieser Kategorie.

15.16.1. Überprüfen Sie die Attributnamen, um einen Konflikt zu vermeiden.

Diese Kritik wird unter einer früheren Designkategorie diskutiert (siehe Abschnitt 15.4.2, „ Überarbeite die Attributnamen, um einen Konflikt zu vermeiden “).

15.16.2. Entfernen Sie die zirkuläre Vererbung der Klasse <Klasse>

Hinweis, dass eine Klasse über eine Kette von Vererbungen von sich selbst erbte, was nicht erlaubt ist.

Achtung

Diese Kritik ist in der aktuellen Release von ArgoUML standardmäßig als inaktiv markiert (die einzige so markierte Kritik). Sie wird nicht ausgelöst, bis sie aktiviert wird.

15.16.3. Die Klasse muß abstrakt sein

Diese Kritik wird in einer früheren Designkategorie diskutiert (siehe Abschnitt 15.13.2, „ Die Klasse muß abstrakt sein “).

15.16.4. Entfernen Sie das Schlüsselwort final oder entfernen Sie Subklassen

Hinweis, das eine als final deklarierte Klasse Spezialisierungen aufweist, was in UML nicht erlaubt ist.

15.16.5. Illegale Generalisierung

Hinweis, dass es eine Generalisierung zwischen Modellelementen unterschiedlicher UML-Metaklassen gibt, was nicht erlaubt ist.

Achtung

Es ist nicht klar, wie so eine Generalisierung in ArgoUML erzeugt werden könnte. Wahrscheinlich zeigt sie auf, dass das Diagramm durch Laden eines defekten Projektes, mit einer XMI-Datei erzeugt wurde, die so eine Generalisierung beschreibt. Wahrscheinlich wurde diese durch ein anderes Tool als ArgoUML erzeugt.

15.16.6. Entferne unnötige Realisierungen aus der Klasse <Klasse>

Hinweis, das die angegebene Klasse eine realisierte, direkte und indirekte Beziehung auf die gleiche Schnittstelle hat (durch Realisierung von zwei Schnittstellen, eine davon ist die Generalisierung der
anderen, zum Beispiel). Ein gutes Design vermeidet solche Duplizierungen.

15.16.7. Definiere eine konkrete (Sub-)Klasse

Hinweis, dass eine Klasse als abstrakt deklariert ist, und keine konkrete Subklassen aufweist, so dass sie niemals realisiert (erzeugt) werden kann.

15.16.8. Definieren Sie eine Klasse, um die Schnittstelle <Schnittstelle> zu implementieren

Hinweis, dass die referenzierte Schnittstelle keine Auswirkung auf das laufende System hat, da sie nicht durch eine Klasse implementiert wurde.

15.16.9. Ändere Mehrfachvererbung in Schnittstellen

Diese Kritik wird unter einer früheren Designkategorie diskutiert (siehe Abschnitt 15.14.1, „Ändern Sie die Mehrfachvererbung in Schnittstellen“).

15.16.10. Machen Sie die Kante sichtbarer

Diese Kritik wird unter einer früheren Designkategorie diskutiert (siehe Abschnitt 15.7.14, „Eine Kante sichtbarer machen“).

15.17. Containment

Kritiken, die sich auf das Containment in ArgoUML beziehen. D.h., wo ein Modellelement den Komponententeil eines anderen bildet.

Die aktuelle Version von ArgoUML hat die folgenden Kritiken in dieser Kategorie.

15.17.1. Entferne zirkuläre Komposition

Hinweis, dass es eine Reihe von Kompositionen (Assoziationen mit einem schwarzen Diamanten) gibt, die einen Zirkelbezug bilden, was nicht erlaubt ist.

15.17.2. Duplizieren Sie den Parameternamen

Hinweis, dass eine Parameterliste einer Operation oder eines Ereignisses zwei oder mehr Parameter mit dem gleichen Namen aufweist, was nicht erlaubt ist.

15.17.3. Zwei Aggregatenden (Rollen) in binärer Assoziation

Nur ein Ende (Rolle) einer binären Assoziation kann Aggregat oder Komposition sein. Dies ist eine wohlgeformte Regel des UML 1.4- Standards.

Aggregation und Komposition werden verwendet, um Ganzes-Teil- Beziehungen darzustellen und der „Teil“ kann per Definition kein Aggregat(Zusammenfassung) sein.

Um dies zu lösen, identifizieren Sie das „Teil“-Ende der Assoziation und verwenden Sie im Kritikassistenten die Schaltfläche Weiter >, setzen seine Aggregation manuell mit Hilfe des Taste


### 15.17.4. Aggregatende (Rolle) in 3-Wege (oder mehr) Assoziation

Drei-Wege- (oder mehr) Assoziationen können keine Aggregatenden (Rollen) haben. Dies ist eine wohlgeformte Regel des UML 1.4 Standards.

Aggregation und Komposition werden verwendet, um Ganzes-Teile-Beziehungen darzustellen. Diese können per Definition nur auf binäre Assoziationen zwischen Modellelementen angewendet werden.

Um dies zu lösen, markieren Sie die Assoziation manuell und setzen Sie jedes Ende (Rolle) dieser Aggregation mit Hilfe des Taste 2- Popup-Menüs oder dem Eigenschaftsregister auf none.

### 15.17.5. Verbergen Sie den Datentyp

Diese Kritik wird unter einer früheren Designkategorie diskutiert (siehe Abschnitt 15.3.1. „Datentyp verbergen“).
Teil 3. Modellreferenz
Kapitel 16. Modellreferenz auf höchster Ebene

16.1. Einleitung

Dieses Kapitel beschreibt jedes Modellelement, dass innerhalb von ArgoUML erzeugt werden kann. Das Kapitel deckt die „generellen“ Modellelemente auf höchster Ebene ab. Die folgenden Kapitel (siehe Kapitel 17, Referenz der Modellelemente für Anwendungsfalldiagramme bis Kapitel 23, Modellelement-Referenz Verteilungsdiagramm) decken jedes der ArgoUML-Diagramme ab.

Es gibt eine enge Beziehung zwischen diesem Material und dem Register Eigenschaften des Detailfensters (siehe Abschnitt 13.3, „Das Register Eigenschaften“). Dieser Abschnitt deckt die Eigenschaften im Allgemeinen ab, in diesem Kapitel werden sie mit den spezifischen Modellelementen verknüpft.

16.2. Das Modell

Das Modell ist in ArgoUML ein Modellelement auf höchster Ebene. Im UML-Metamodell ist es eine Subklasse von package. In vielerlei Hinsicht weist es innerhalb von ArgoUML Ähnlichkeiten mit einem package auf (siehe Abschnitt 18.2, „Paket“).

Anmerkung

ArgoUML ist auf ein Modell beschränkt.

Standard-Datentypen, Klassen und Pakete werden (der Standard, siehe Kapitel 24, Profile) als Subpakete des Modells geladen. Diese Subpakete sind im Modell zu Beginn nicht präsent, werden aber zum Modell hinzugefügt, wenn sie verwendet werden.

16.2.1. Register Modelldetails

Folgende Detail-Register sind für das Modell aktiv.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Stereotypen
Standard-Register. Dies enthält eine Liste von Stereotypen, die diesem Modell zugeordnet sind und eine Liste der verfügbaren Stereotypen, die auf dieses Modell angewendet werden können.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell hat das Modell die folgenden Standard-Eigenschaftswerte definiert.
• derived (von der Superklasse, Modellelement).


Abgeleitete Modelle haben ihren Wert in der Analyse, um nützliche Namen oder Konzepte einzuführen und im Design, um eine Nachverarbeitung zu verhindern.

16.2.2. Symbolleiste Modelleigenschaften

Nach oben
Navigiert durch die zusammengesetzte Struktur des Modelles nach oben.

Das das Modell das oberste Paket ist, kann nichts passieren. Aus diesem Grund ist die Schaltfläche immer deaktiviert.

Neues Paket
Dies erzeugt eine neues Paket (siehe Abschnitt 18.2, „Paket“) innerhalb des Modelles (das in keinem Diagramm erscheint), und springt sofort in das Register Eigenschaften des Paketes.

Tipp
Da es Sinn machen kann, Pakete für diese Modell auf diese Weise zu erzeugen, ist es gewöhnlich deutlich klarer, diese innerhalb der Diagramme zu erzeugen, wenn Sie welche benötigen.

Neuer Datentyp
Dies erzeugt einen neuen Datentyp (siehe Abschnitt 16.3, „Datatyp“) innerhalb des Modelles (der in keinem Diagramm erscheint) und springt sofort in das Register Eigenschaften dieses Datentyps.

Neue Aufzählung
Dies erzeugt eine neue Aufzählung (siehe Abschnitt 16.4, „Enumeration (Aufzählung)“) innerhalb des Modelles (die in keinem Diagramm erscheint) und springt sofort in das Register Eigenschaften dieser Aufzählung.

Neuer Stereotyp
Dies erzeugt einen neuen Stereotypen (siehe Abschnitt 16.6, „Stereotyp“) innerhalb des Modelles und springt sofort in das Register Eigenschaften dieses Stereotypen.

Löschen
Dieses Symbol ist immer deaktiviert, da es keinen Sinn macht, das Modell zu löschen!

16.2.3. Eigenschaftsfelder des Modelles
Name

Textfeld. Name des Modelles. Der Name des Modells, wie alle anderen Pakete, wird per Konvention in Kleinbuchstaben geschrieben.

**Anmerkung**

Der Standardname, der einem neuen Modell in ArgoUML zugewiesen wird ist unbenanntesModell. Er ist daher fehlerfrei und garantiert, dass ArgoUML immer mit mindestens einem Problem startet, was durch die Design- Kritiken ausgestellt wird.

Stereotyp

DropDown-Auswahl. Das Modell ist standardmäßig mit den UML-Standard-Stereotypen für Modelle (systemModel und metamodel) und dem Paket (facade, framework, stub) ausgestattet.

Das stereotypisieren von Modellen ist nützlich, obwohl es in ArgoUML nur von eingeschränktem Wert ist, da Sie nur ein einziges Modell haben.

Stereotypen steuern

Symbol. Wenn ein Stereotyp markiert wurde, wird dieser über das Stereotyp-Eigenschaftsfenster gesteuert (siehe Abschnitt 16.6, „Stereotyp“).

Namensraum


Sichtbarkeit

Auswahlbereich mit den Einträgen public, private, protected und package.

Zeichnet die Sichtbarkeit des Modelles auf. Da ArgoUML nur ein Modell erlaubt, hat diese Eigenschaft keine besondere Bedeutung.

Modifizierer

Markier-Bereich mit den Einträgen Abstract, Leaf und Root.


- Leaf gibt an, dass diese Modell keine weiteren Subpakete haben darf, während root angibt, das es das Modell auf oberster Ebene ist.

  Innerhalb von ArgoUML kann root nur auf das Modell sinnvoll angewendet werden, da alle Pakete sich innerhalb des Modells befinden. In Abwesenheit des Stereotypen topLevel, kann dies dazu verwendet werden, herauszustellen dass das Modell sich auf der obersten Ebene befindet.
Generalisierungen
Textbereich. Listet jedes Modell auf, das dieses Modell generalisiert.

**Anmerkung**
Da es in ArgoUML nur ein Modell gibt, gibt es keine Spezialisierung oder Generalisierung, die erzeugt werden könnte.

Spezialisierung
Textbereich. Listet jedes spezialisiertes Modell auf (z.B. für welches Modell diese Modell eine Generalisierung ist).

**Anmerkung**
Da es in ArgoUML nur ein Modell gibt, gibt es keine Spezialisierung oder Generalisierung, die erzeugt werden könnte.

Eigene Elemente

Ein Taste 1-Klick auf jedes dieser Modellelemente veranlasst, dass zu diesem Modellelement gesprungen wird.

### 16.3. Datatyp


Innerhalb von UML 1.4 ist ein Datatyp eine Subklasse der Metaklasse Klassifizierer. Dieser umfasst die vordefinierten primitiven Typen (byte, char, double, float, int, long und short), die vordefinierte Aufzählung, boolean und die anwenderdefinierten enumeration Typen.

**Anmerkung**
Auch void ist als Datentyp innerhalb von ArgoUML definiert.


16.3.1. Register Datentypdetails

Folgende Detail-Register sind für Datentypen aktiv:

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften


Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Quellcode

Eigenschaftswerte

Standard-Register. Das UML-Metamodell hat die folgenden Eigenschaftswerte für Datentyp definiert.

• persistence (von der Superklasse , Classifier). Werte transitory, geben an, dass der Zustand gelöscht wird, wenn eine Instanz gelöscht oder persistent wird, der markierte Zustand wird erhalten, wenn eine Instanz gelöscht wird.

  **Tipp**

  Da benutzerdefinierte Datentypen Aufzählungen sind, haben sie keinen zu erhaltenden Zustand und der Wert dieses Eigenschaftswertes ist irrelevant.

• semantics (von der Superklasse , Classifier). Der Wert ist eine Spezifikation der Semantik des Datentyps.

• derived (von der Superklasse , Modell-Element). Werte mit true bedeuten, dass die Klasse redundant ist - sie kann formal von anderen Elementen abgeleitet werden. false bedeutet, dass sie nicht abgeleitet werden kann.

  **Tipp**

  Obwohl formal verfügbar, hat ein abgeleiteter Datentyp keinen bestimmten Wert und daher sollten Datentypen immer mit derived=false bezeichnet werden.

16.3.2. Symbolleiste Datentypeigenschaften

Nach oben
Geht in der Paketstruktur nach oben.
Neuer Datentyp
Erzeugt innerhalb des gleichen Paketes einen neuen Datentyp (siehe Abschnitt 18.6, „Klasse“) als aktuellen Datentyp.

Tipp
Obwohl es Sinn machen kann, Datentypen auf diesem Weg zu erzeugen, kann es klarer sein, diese innerhalb eines Paketes oder Modèles, wo sie sie benötigen, zu erzeugen.

Neue Aufzählung
Erzeugt im gleichen Paket eine neue Aufzählung (siehe Abschnitt 16.4, „Enumeration (Aufzählung)“) als Datentyp und springt sofort in das Register Eigenschaften dieser Aufzählung.

Neue Operation
Erzeugt einen neuen Operation innerhalb des Datentyps und springt sofort in das Register Eigenschaften dieser Operation.

Neuer Stereotyp
Erzeugt innerhalb des gleichen Paketes einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) und springt sofort in das Register Eigenschaften dieses Stereotyps.

Löschen
Löscht den Datentyp aus dem Modell.

16.3.3. Eigenschaftsfelder für den Datentyp

Name

Anmerkung
Der Standardname neu erzeugter Datentypen ist ein leerer String „“.

Namensraum

Modifizierer
Markierfeld, mit den Einträgen Abstract, Leaf und Root.
- Abstract deklariert, dass dieser Datentyp nicht instantiiert werden kann und daher immer
spezialisieren werden muss.

**Anmerkung**

ArGoUML enthält keine Mechanismen, um Datentypen zu spezialisieren, so dass dieses Markierfeld wenig genutzt wird.

- **Leaf** gibt an, dass der Datentyp eine weiteren Subtypen haben kann, während **Root** angibt, dass es sich um einen Datentyp auf oberster Ebene handelt.

**Tipp**

Sie können die Spezialisierung eines Datentyps in einem Klassendiagramm definieren, indem Sie Generalisierungen zwischen ihnen einzeichnen.

**Sichtbarkeit**

Auswahlfeld, mit den Einträgen **public**, **private**, **protected**, und **package**.

Gibt die Sichtbarkeit des Datentyps an.

**Abhängig von**

Textbereich. Listet jedes Element auf, das von diesem Datentyp abhängig ist.

**Achtung**

Es ist nicht klar, dass Abhängigkeiten zwischen Datentypen mehr Sinn machen.

**Notwendig für**

Textbereich. Listet jedes Element auf von dem dieser Datentyp abhängt.

**Achtung**

Es ist nicht klar, dass Abhängigkeiten zwischen Datentypen mehr Sinn machen.

**Generalisierungen**

Textbereich. Listet jeden Datentyp auf, der diesen Datentyp *vererbt (generalisiert)*.

**Spezialisierungen**

Textbereich. Listet jeden spezialisierten Datentyp auf (z.B. für den dieser Datentyp eine Generalisierung ist).

**Operationen**


- **Nach oben**. Nur verfügbar, wenn es zwei oder mehr Operationen gibt und sich die markierte Operation nicht ganz oben befindet. Sie wird um einen Schritt nach oben bewegt.

- **Nach unten**. Nur verfügbar, wenn es zwei oder mehr Operationen gibt und sich die markierte Operation nicht ganz unten befindet. Sie wird um einen Schritt nach unten bewegt.
Details über Operationen, siehe Abschnitt 18.8, „Operation“.

Achtung

ArgoUML behandelt alle Operationen gleich. Jede hier erzeugte Operation wird den gleichen Mechanismus wie Operationen in Klassen verwenden. Erinnern Sie sich, dass Operationen in Datentypen keine Seiteneffekte haben dürfen (sie sind read-only). Das heißt, der Modifizierer query muss bei allen Operationen ausgewählt sein.

16.4. Enumeration (Aufzählung)


Innerhalb von UML 1.4 ist Enumeration eine Subklasse der Metaklasse Datentyp.


Neue Enumerationen können in ArgoUML mit Hilfe der Schaltfläche Neue Enumeration im Register Eigenschaften des Modelles oder des Paketes (in diesem Fall ist die neue Enumeration auf den Scope des Paketes beschränkt), wie auch im Register Eigenschaften für Datentypen und Enumeration erzeugt werden. Enumerationen können auch mit dem Symbol in der Diagramm-Symbolleiste eines Klassendiagrammes erzeugt werden.


16.4.1. Die Detail-Register Enumeration

Folgende Detail-Register sind für Enumerationen aktiv.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften


Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung
Standard-Register.
Quellcode
Standards-Register.

Stereotyp
Standards-Register. Das UML-Metamodell hat standardmäßig die folgenden Stereotypen für einen Klassifizierer definiert, der auch auf eine Enumeration angewendet wird:

- `metaclass` (von der Superklasse, `Classifier`).
- `powertype` (von der Superklasse, `Classifier`).
- `process` (von der Superklasse, `Classifier`).
- `thread` (von der Superklasse, `Classifier`).
- `utility` (von der Superklasse, `Classifier`).

Eigenschaftswerte
Standards-Register. Im UML-Metamodell sind für die `Enumeration` keine Standards-Eigenschaftswerte definiert.

16.4.2. Eigenschaftssymbolleiste Enumeration

- **Nach oben**
  Geht in der zusammengesetzten Struktur nach oben.

- **Neuer Datentyp**
  Erzeugt im gleichen Paket einen neuen Datentyp (siehe Abschnitt 18.6, „Klasse“) als aktuelle Enumeration.

- **Neue Enumeration**
  Erzeugt innerhalb des gleichen Namensraumes eine neue Enumeration als aktuelle Enumeration und springt sofort in das Register Eigenschaften der neuen Enumeration.

- **Neues Enumerations-Literal**
  Erzeugt in der Enumeration ein neues Enumerations-Literal und springt sofort in das Register Eigenschaften dieses Literals.

- **Neue Operation**
  Erzeugt in der Enumeration eine neue Operation und springt sofort in das Register Eigenschaften dieser Operation.

- **Neuer Stereotyp**
  Erzeugt innerhalb des gleichen Paketes einen neuen Stereotypen (siehe Abschnitt 16.6, „Stereotyp“) und springt sofort in das Register Eigenschaften dieses Stereotypen.

- **Aus Modell entfernen**
  Entfernt den Datentyp aus dem Modell.
16.4.3. Eigenschaftsfelder für Enumerationen

Name


Anmerkung

Der Standardname für einen neu erzeugten Datentyp ist ein leerer String „“.

Enumerationen mit einem leeren String als Namen, wird mit dem Namen (Unbenannte Enumeration) im Explorer erscheinen.

Namensraum


Modifizierer

Markierfeld mit den Einträgen Abstract, Leaf und Root.

• Abstract wird verwendet, um zu deklarieren, dass diese Enumeration nicht instantiiert werden kann, aber immer spezialisiert werden muss.

• Leaf zeigt an, dass diese Enumeration keine weiteren Subtypen haben kann, während Root angibt, dass es sich um eine Enumeration auf oberster Ebene handelt.

Sichtbarkeit

Auswahlfeld mit den Einträgen public, private, protected und package.

Gibt die Sichtbarkeit der Enumeration wieder.

Abhängig von


• Hinzufügen.... Öffnet ein Dialogfenster, das es erlaubt, Abhängigkeiten von anderen Modellelementen zu erzeugen.

Notwendig für


• Hinzufügen.... Öffnet ein Dialogfenster, das es erlaubt, Abhängigkeiten auf andere Modellelemente zu erzeugen.

Generalisierungen

Textbereich. Listet jede Enumeration auf, die diese Enumeration generalisiert.

Spezialisierungen

Textbereich. Listet jede spezialisierte Enumeration auf (z.B. für wen diese Enumeration eine Generalisierung ist).
Operationen


- **Nach oben.** Nur verfügbar, wenn zwei oder mehr Operationen gelistet sind und die markierte Operation sich nicht oben befindet. Sie wird um einen Schritt nach oben bewegt.

- **Nach unten.** Nur verfügbar, wenn zwei oder mehr Operationen gelistet sind und die markierte Operation sich nicht unten befindet. Sie wird um einen Schritt nach unten bewegt.

Details über Operationen, siehe Abschnitt 18.8, „Operation“.

† Achtung


Literale


- **Nach oben.** Nur verfügbar, wenn es zwei oder mehr Literale gibt und sich das markierte Literal nicht ganz oben befindet. Es wird um einen Schritt nach oben bewegt.

- **Nach unten.** Nur verfügbar, wenn zwei oder mehr Literale aufgelistet sind und das markierte Literal sich nich ganz unten befindet. Es wird um einen Schritt nach unten bewegt.

16.5. Enumeration Literal

Ein Enumerations-Literal ist eines der vordefinierten Werte einer Enumeration.

16.6. Stereotyp


Neue Stereotypen werden von fast jedem Modellelement über das Register Eigenschaften hinzugefügt. Eigenschaften existierender Stereotypen können durch Auswahl des Registers Eigenschaften eines jeden Modellelementes mit diesem Stereotypen mit Hilfe der Navigationsschaltfläche erreicht werden ( ) im Register Eigenschaften.

16.6.1. Detail-Register Stereotyp

Die folgenden Detail-Register sind für Stereotypen aktiv.
Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Stereotyp
Standard-Register.

Warnung
Hier können Sie Stereotypen für Stereotypen einstellen, was nicht sehr sinnvoll ist.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell haben Stereotypen die folgenden Standard-Eigenschaftswerte definiert.

- derived (von der Superklasse, ModelElement). Der Wert true bedeutet, dass die Klasse redundant ist - sie kann formal von anderen Elementen abgeleitet sein. false bedeutet, dass sie es nicht kann.

Anmerkung
Dies zeigt jedes Element mit diesem Stereotyp an, dessen Eigenschaftswert entsprechend auf derived gesetzt wurde.

...  
Achtung
Eigenschaftswerte für Stereotypen unterscheiden sich deutlich von denen für Elemente der UML-Core-Architektur, da Sie auf alle Modellelemente angewendet werden, dem dieser Stereotyp zugewiesen wurde und nicht nur auf den Stereotypen selbst.

16.6.2. Eigenschaftssymbolleiste Stereotyp

Nach oben
Geht in der Paketstruktur des Modelles nach oben.

Stereotyp hinzufügen
Erzeugt im Modell (das in keinem Diagramm erscheint) einen neuen Stereotypen (siehe Abschnitt 16.6, „Stereotyp“) und springt sofort in das Register Eigenschaften dieses Stereotypen.

223
16.6.3. Eigenschaftsfelder für Stereotypen

**Name**


**Anmerkung**

ArgoUML erzwingt keine Namenskonvention für Stereotypen.

**Namensraum**


**Modifizierer**

Markierfeld mit den Einträgen `Abstract`, `Leaf` und `Root`.

- **Abstract** wird verwendet, um zu deklarieren, dass Modellelemente, die diesen Stereotypen verwenden nicht instantiiert werden können, aber immer spezialisiert werden müssen.

- **Leaf** gibt an, dass Modellelemente, die diesen Stereotypen verwenden, keine weiteren Subtypen haben können, während `Root` angibt, dass es sich um ein Modellelement auf oberster Ebene handelt.

**Achtung**

Erinnern Sie sich, dass diese Modifizierer auf Modellelemente angewendet werden die Stereotypen verwenden, nicht auf die Stereotypen selbst.

**Warnung**

Weder erzwingt ArgoUML, noch prüft es, dass Modellelemente, die Stereotypen verwenden, die Stereotyp-Modifizierer übernehmen.

**Sichtbarkeit**

Auswahlfeld mit den Einträgen `public`, `private`, `protected` und `package`.

Gibt die Sichtbarkeit für Stereotypen wieder.

**Generalisierungen**

Textbereich. Listet jeden Stereotypen auf, der diesen Stereotypen `generalisiert`. 

---

Löschen

Entfernt diesen Stereotypen aus dem Modell.
Spezialisierungen
Textbereich. Listet jeden spezialisierten Stereotypen auf (z.B. für den dieser Stereotyp eine Generalisierung ist).

Eigenschaftsdefinitionen
Textbereich. Listet jede Eigenschaftsdefinition auf, die für diesen Stereotypen definiert ist.

Basisklasse

16.7. Eigenschaftsdefinition
(Noch zu beschreiben)

16.8. Diagramm
Der UML-Standard spezifiziert acht prinziplielle Diagramme, alle werden von ArgoUML unterstützt.


• **Klassendiagramm.** Dieses Diagramm erfasst die statische Struktur eines zu entwerfenden Systems, zeigt die Klassen, Schnittstellen und Datentypen und wie sie zueinander in Beziehung stehen. Varianten dieses Diagrammes werden dazu verwendet, die Paketstrukturen innerhalb eines Systems (das *Paketdiagramm*) und die Beziehungen zwischen bestimmten Instanzen (das *Objektdiagramm*) darzustellen.


• **Verhaltensdiagramme.** Es gibt vier solcher Diagramme (oder, genauer gesagt, fünf, da das Anwendungsfalldiagramm eine Art Verhaltensdiagramm ist), die das dynamische Verhalten des Systems auf allen Ebenen darstellen.


  • **Interaktionsdiagramme.** Es gibt zwei Diagramme in dieser Kategorie, die dazu verwendet werden, die dynamische Interaktion zwischen Objekten im System (Klasseninstanzen).


• **Implementierungsdiagramme.** UML definiert zwei Implementierungsdiagramme, um die Beziehung zwischen den Softwarekomponenten, die das System ausmachen (das *Komponentendiagramm*) und der Beziehung zwischen der Software und der Hardware auf dem das System zu Laufzeit läuft (das *Verteilungsdiagramm*) darzustellen.


---

**Anmerkung**

ArgoUML verwendet sein Verteilungsdiagramm, um die UML 1.4- Komponenten, Verteilung- und Objektdiagramme zu erzeugen.

**Achtung**

Zustands- und Aktivitätsdiagramme sind mit einer bestimmten Klasse oder Operation verknüpft (oder letzten Endes auch mit einem Paket) und können nur erzeugt werden, wenn dieses Modellelement markiert wurde.

**Warnung**

In der Version 0.26 von ArgoUML, werden die UML 1.4-Objektdiagramme als Variante des Klassendiagrammes nicht direkt unterstützt. Jedoch ist es möglich, einfache Objektdiagramme mit dem ArgoUML-Verteilungsdiagramm zu erstellen.

---

**16.8.1. Detail-Register Diagramm**

Die folgenden Detail-Register sind für Diagramm aktiv.
**Zu-Bearbeiten-Element**

Standard-Register.

**Eigenschaften**


### 16.8.2. Eigenschaftssymbolleiste Diagramm

**Nach oben**

Geht in der Paketstruktur des Modelles nach oben.

**Löschen**

Entfernt das Diagramm aus dem Modell. Als Konsequenz werden im Falle eines Zustandsdiagrammes oder eines Aktivitätsdiagrammes auch alle darin enthaltenen Elemente gelöscht.

### 16.8.3. Eigenschaftsfelder für Diagramme

**Name**

Der Name des Diagrammes. Es gibt keine Konventionen für die Benennung von Diagrammen. Standardmäßig verwendet ArgoUML (Leerzeichen-separierte) Diagrammnamen und eine sequentielle Nummer, also Anwendungsfalldiagramm 1.

**Tipp**

Dieser Name wird dazu verwendet, einen Dateinamen zu generieren, wenn Sie das Menü „Speichere Grafiken...“ aktivieren.

**Ausgangsmodell**

Kapitel 17. Referenz der Modellelemente für Anwendungsfalldiagramme

17.1. Einleitung

Dieses Kapitel beschreibt jedes Modellelement, das innerhalb eines Anwendungsfalldiagrammes erzeugt werden kann. Beachten Sie, dass einige Sub-Modellelemente der Modellelemente nicht immer imDiagramm erscheinen können.

Es gibt eine sehr enge Beziehung zwischen diesem Material und dem Register Eigenschaften des Detailfensters (siehe Abschnitt 13.3, „Das Register Eigenschaften“). Dieser Abschnitt behandelt die Eigenschaften im Allgemeinen, in diesem Kapitel sind sie mit den spezifischen Modellelementen verknüpft.

Abbildung 17.1, „Typische Modellelemente in einem Anwendungsfalldiagramm.“ zeigt ein Anwendungsfalldiagramm mit allen typischen Modellelementen.

Abbildung 17.1. Typische Modellelemente in einem Anwendungsfalldiagramm.

17.1.1. ArgoUML-Einschränkungen, welche die Anwendungsfalldiagramme betreffen

Anwendungsfalldiagramme werden jetzt von ArgoUML gut unterstützt. Es gibt aber einige kleinere Einschränkungen, speziell bei Erweiterungspunkten.

Anmerkung

Frühere Versionen von ArgoUML (0.9 und früher) implementierten extend- und include-Beziehungen mit Hilfe einer stereotypisierten Abhängigkeitsbeziehung. Obwohl solche
Diagramme korrekt im Diagramm dargestellt werden, werden sie nicht richtig mit den Anwendungsfällen verbunden, und Sie sollten diese durch die richtigen extend- und include-Beziehungen im aktuellen System ersetzen.

17.2. Akteur

Ein Akteur repräsentiert eine externe Entität (Mensch oder Maschine) die mit dem System interagiert, die Eingabeinformationen gibt, Ausgabeinformationen empfängt oder beides.

Innerhalb des UML-Metamodelles ist der Akteur eine Subklasse von classifier.

Der Akteur wird durch ein „Strichmännchen“ im Diagramm repräsentiert (siehe Abbildung 17.1, „Typische Modellelemente in einem Anwendungsfalldiagramm.“).

17.2.1. Detail-Register Akteur

Die folgenden Detail-Register sind für Akteure aktiv.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung

Quellcode

Stereotyp
Standard-Register.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für den Akteur die folgenden Eigenschaftswerte definiert.

- persistence (von der Superklasse Classifier). Der Wert transitory gibt an, dass der angezeigte Zustand gelöscht wird, wenn eine Instanz gelöscht oder persistent wird, ein markierter Zustand bleibt erhalten, wenn eine Instanz gelöscht wird.

Tipp

Akteure sitzen außerhalb des Systems, so dass deren internes Verhalten von geringem Interesse. Dieser Eigenschaftswert sollte am Besten ignoriert werden.

- semantics (von der Superklasse Classifier). Der Wert ist eine Spezifikation der
Semantik des Akteurs.

- **derived** (von der Superklasse ModelElement). Der Wert true bedeutet, dass der Akteur redundant ist - er kann formal von anderen Elementen abgeleitet werden oder false bedeutet, er kann es nicht.

**Anmerkung**

Abgeleitete Akteure haben nur einen eingeschränkten Wert, da sie außerhalb des Systems designed wurden. Sie können Ihren Wert bei der Analyse haben, um nützliche Namen oder Konzepte einzuführen.

**Checkliste**

Standard-Register für einen Classifier.

17.2.2. Eigenschaftssymbolleiste Akteur

**Nach oben**

Geht in der Paketstruktur des Modelles einen Schritt nach oben.

**Akteur hinzufügen**

Erzeugt einen neuen Akteur im Modell (aber nicht im Diagramm) und springt unmittelbar in das Register Eigenschaften dieses Akteurs.

**Tipp**

Diese Methode, einen neuen Akteur zu erzeugen, kann irritierend sein. Es ist besser, einen Akteur im Diagramm zu erzeugen.

**Neuer Signaleingang**

Erzeugt innerhalb des Modelles einen neuen Signaleingang (aber nicht innerhalb des Diagrammes) und springt unmittelbar in das Register Eigenschaften dieses Signaleinganges.

**Tipp**

Ein Signaleingang ist eine Deklaration, dass der Akteur ein Signal verarbeitet. Die aktuelle Verarbeitung wird aber mit einem Zustandsautomaten spezifiziert.

**Löschen**

Entfernt den markierten Akteur aus dem Modell.

**Warnung**

Dies ist eine Löschung aus dem Modell, nicht aus dem Diagramm. Um einen Akteur
aus einem Diagramm zu entfernen, ihn aber im Modell zu erhalten, verwenden Sie das Hauptmenü Aus Diagramm entfernen (oder drücken Sie die Taste Entf).

17.2.3. Eigenschaftsfelder für einen Akteur

Name


Anmerkung

ArgoUML erzwingt keine Namenskonventionen für Akteure.

Namensraum


Modifizierer

Markierfeld mit den Einträgen Abstract, Leaf und Root.

• Abstract wird verwendet, um zu deklarieren, dass der Akteur nicht instanziiert werden kann, aber immer spezialisiert werden muss.

Achtung

Da Akteure spezialisiert und generalisiert werden können, ist nicht klar, ob ein abstrakter Akteur eine Bedeutung hat. Vielleicht könnte er dazu verwendet werden, dass der Akteur mit einem Anwendungsfall nicht selbst interagiert, aber es seine Kinder tun.

• leaf gibt an, dass dieser Akteur keine weiteren Kinder haben kann, während Root angibt, dass es sich um einen Akteur auf oberster Ebene ohne Eltern handelt.

Generalisierungen

Textbereich. Listet jeden Akteur auf, der diesen Akteur generalisiert.

Ein Taste 1-Doppelklick springt zur Generalisierung und öffnet dessen Register Eigenschaften.

Spezialisierungen

Textfeld. Listet jeden spezialisierten Akteur auf (z.B. für den dieser Akteur eine Generalisierung ist). Die spezialisierten Akteure können mit der gleichen Anwendungsfallinstanz kommunizieren, wie dieser Akteur.

Ein Taste 1-Doppelklick springt zur Generalisierung und öffnet dessen Register Eigenschaften.
Assoziationsende

Textbereich. Listet jedes Assoziationsende der mit diesem Akteur verbundenen Assoziationen auf.

Ein Tast-1-Doppelklick springt zu dem markierten Eintrag.

17.3. Anwendungsfall

Ein Anwendungsfall repräsentiert eine vollständig abgeschlossene Aktivitäts-„Folge“ des Systems in Beziehung zu seinem externen Anwender (Akteure), Mensch oder Maschine. Er repräsentiert den primären Weg der Anforderungen, die für das zu konstruierende System erfasst wurden.

Innerhalb des UML-Metamodell ist der Anwendungsfall eine Subklasse von classifier.

Das Symbol für einen Anwendungsfall ist ein Oval (siehe Abbildung 17.1, „Typische Modellelemente in einem Anwendungsfalldiagramm.“). Es kann in zwei Bereiche aufgeteilt sein, wobei der untere Bereich Erweiterungspunkte darstellt.

Achtung

Standardmäßig zeigt ArgoUML den Erweiterungspunkt-Bereich nicht an. Er kann mit Hilfe des kontextsensitiven Menüs Erweiterungspunkte anzeigen (Taste 2-Klick verwenden), oder über das Register Darstellung zur Anzeige gebracht werden.

17.3.1. Detail-Register Anwendungsfall

Die folgenden Detail-Register sind für die Anwendungsfälle aktiv.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften
Siehe Abschnitt 17.3.2, „Eigenschaftssymbolleiste Anwendungsfall“ und Abschnitt 17.3.3, „Eigenschaftsfelder für einen Anwendungsfall“ unten.

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung
Standard-Register. Die Füllfarbe wird für das Anwendungsfall-Oval verwendet.

Das Markierfeld Anzeige: Erweiterungspunkte steuert, ob der Erweiterungspunkt-Bereich dargestellt wird.

Quellcode

Stereotyp
Standard-Register.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für den Anwendungsfall folgende
Eigenschaftswerte definiert.

- **persistence** (von der Superklasse Classifier). Der Wert *transitory*, gibt an, dass der angezeigte Zustand gelöscht wird, wenn die Instanz gelöscht oder *persistent* gespeichert wird, der markierte Zustand bleibt erhalten, wenn die Instanz gelöscht wird.

**Tipp**

Im Allgemeinen ist die Instanziierung von Anwendungsfällen nicht der Hauptspektrum einer Designmethode (sie betreffen häufig die Anforderungserfassung). In den meisten OOA&D-Methoden kann diese Eigenschaft sicherlich ignoriert werden.

- **semantics** (von der Superklasse Classifier). Der Wert ist eine Spezifikation der Semantik des Anwendungsfalles.

- **derived** (von der Superklasse ModelElement). Der Wert *true* bedeutet, dass der Anwendungsfall redundant ist - er kann formal von anderen Elementen abgeleitet werden, oder *false* bedeutet, dass er es nicht kann.

**Anmerkung**

Abgeleitete Anwendungsfälle haben ihren Wert in der Analyse, um nützliche Namen oder Konzepte einzuführen.

**Checkliste**

Standard-Register für einen Classifier.

### 17.3.2. Eigenschaftssymbolleiste Anwendungsfall

**Nach oben**

Geht in der Paketstruktur des Modelles einen Schritt nach oben.

**Neuer Anwendungsfall**

Erzeugt innerhalb des Modells einen neuen Anwendungsfall (aber nicht innerhalb des Diagrammes) und zeigt unmittelbar das Register Eigenschaften dieses Anwendungsfalles.

**Tipp**


**Neuer Erweiterungspunkt**

Erzeugt einen neuen Erweiterungspunkt innerhalb des Namensraumes des aktuellen Anwendungsfalles, mit dem aktuellen Anwendungsfall als seinen mit ihm verbundenen Anwendungsfall. Er springt sofort in das Register Eigenschaften dieses Erweiterungspunktes.
17.3.3. Eigenschaftsfelder für einen Anwendungsfall

Name

Anmerkung
ArgoUML erzwingt keine Namenskonventionen für Anwendungsfälle.

Namensraum

Modifizierer
Markierfeld mit den Einträgen Abstract Leaf und Root.

- Abstract wird verwendet, um zu deklarieren, dass der Anwendungsfall nicht instanziert
werden kann, aber immer spezialisiert werden muss.

- leaf gibt an, dass dieser Anwendungsfall keine weiteren Kinder haben kann, während Root angibt, dass es sich um einen Anwendungsfall auf oberster Ebene ohne Eltern handelt.

**Abhängig von**
Textbereich. Listet die „abhängigen“ Enden einer Beziehung auf, z.B. das Ende, das ein anderes Ende benutzt.

Ein Taste 1-Doppelklick springt zur Abhängigkeit und öffnet dessen Register Eigenschaften.


**Notwendig für**
Textbereich. Listet die „notwendigen“ Enden der Beziehung auf, z.B. die Enden, die für das andere Ende notwendig sind.

Ein Taste 1-Doppelklick springt zu der Abhängigkeit und öffnet dessen Register Eigenschaften.

Ein Taste 2-Klick zeigt ein Popup-Menü mit einem Eintrag „hinzufügen“ an, das ein Dialogfenster öffnet, indem sie die notwendigen Modellelemente hinzufügen oder entfernen können.

**Generalisierungen**

**Spezialisierungen**
Textfeld. Listet jeden spezialisierten Anwendungsfall auf (z.B. für den dieser Anwendungsfall eine Generalisierung ist).

Ein Taste 1-Doppelklick springt zur Spezialisierung und öffnet dessen Register Eigenschaften.

**Erweitert**
Textfeld. Listet alle Klassen auf, die durch diesen Anwendungsfall erweitert werden.

Wo eine Erweitert-Beziehung erzeugt wurde, wird ein Taste 1-Doppelklick zu dieser Beziehung springen.

**Includes**
Textfeld. Listet jeden Anwendungsfall auf, der diesen Anwendungsfall einschliesst.

Wo eine include-Beziehung erzeugt wurde, wird ein Taste 1-Doppelklick zu dieser Beziehung springen.

**Attribute**

- **Nach oben.** Nur verfügbar, wenn zwei oder mehr Attribute gelistet werden und sich das markierte Attribut nicht ganz oben befindet. Es bewegt das Attribut um einen Schritt nach oben.
Nach unten. Nur verfügbar, wenn zwei oder mehr Attribute gelistet werden und sich das markierte Attribut nicht ganz unten befindet. Es bewegt das Attribut um einen Schritt nach unten.

Assoziationsenden
Textfeld. Listet alle Assoziationsenden (siehe Abschnitt 18.12, „Assoziation“) der mit diesem Anwendungsfall verbundenen Assoziationen auf.

Ein Taste 1-Doppelklick springt zu dem ausgewählten Eintrag.

Operationen

• Nach oben. Nur verfügbar, wenn zwei oder mehr Operationen aufgelistet sind und die markierte Operation sich nicht ganz oben befindet. Es bewegt die Operation um einen Schritt nach oben.

• Nach unten. Nur verfügbar, wenn zwei oder mehr Operationen aufgelistet sind und die markierte Operation sich nicht ganz unten befindet. Es bewegt die Operation um einen Schritt nach unten.

Erweiterungspunkte
Textfeld. Wenn dieser Anwendungsfall erweitert ist oder erweitert werden kann, listet dieses Feld alle Erweiterungspunkte dieses Anwendungsfalles auf.

**Anmerkung**

Erweiterungspunkte werden nicht anhand ihrer Namen sondern anhand des Erstellungsortes gelistet.

An der Stelle, wo ein Erweiterungspunkt erstellt wurde (siehe unten) wird ein Taste 1-Doppelklick zu dieser Beziehung springen. Taste 2 öffnet ein Popup-Menü mit den folgenden Einträgen.

• Neu. Fügt einen neuen Erweiterungspunkt hinzu und springt dort hin. Er macht diesen Anwendungsfall zum Eigentümer-Anwendungsfall dieses Erweiterungspunktes.


• Nach unten. Nur verfügbar, wenn zwei oder mehr Erweiterungspunkte aufgelistet sind und der sich der ausgewählte Erweiterungspunkt nicht ganz unten befindet. Er bewegt den Erweiterungspunkt um eine Position nach unten.

**17.4. Erweiterungspunkt**

Ein Erweiterungspunkt beschreibt einen Punkt in einem Anwendungsfall, bei dem ein erweiternder Anwendungsfall zusätzliches Verhalten enthalten kann.

Beispiele in einem Verkaufssystem für Vertriebler könnte der Anwendungsfall für das Bezahlten eines Tickets sein, das einen Erweiterungspunkt für die Art der Bezahlung aufweist. Erweiternde
Anwendungsfälle können diese Punkt durch die Bezahlung per Barzahlung, Kreditkarte usw. erweitern.


\[ \text{Name : Ort} \]

17.4.1. Detail-Register Erweiterungspunkt

Die aktiven Detail-Register für Erweiterungspunkte sind folgende.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften
Siehe Abschnitt 17.4.2, „Eigenschaftssymbolleiste Erweiterungspunkt“ und Abschnitt 17.4.3, „Eigenschaftsfelder des Erweiterungspunktes“ unten.

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Stereotyp
Standard-Register.

Erweiterungspunkte haben standardmäßig keinen definierten Stereotypen.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für die Erweiterungspunkte die folgenden Standard- Eigenschaftswerte definiert.

- derived (von der Superklasse Modellelement). Der Wert true bedeutet, dass der Erweiterungspunkt redundant ist - er kann formal von anderen Elementen abgeleitet werden, oder false bedeutet, er kann es nicht.

\[ \text{Anmerkung} \]

Es ist nicht klar, wie abgeleitete Erweiterungspunkte einen Wert in der Analyse aufweisen können.

17.4.2. Eigenschaftssymbolleiste Erweiterungspunkt

\[ \text{Nach oben} \]

Springt nach oben zu dem Anwendungsfall, der Eigentümer dieses Erweiterungspunktes ist.

\[ \text{Neuer Erweiterungspunkt} \]

Erzeugt unterhalb des markierten Erweiterungspunktes einen neuen Erweiterungspunkt und springt sofort in das Register Eigenschaften des neue erzeugten Erweiterungspunktes.
Neuer Stereotyp

Erzeugt für den markierten Erweiterungspunkt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) und springt sofort in das Register Eigenschaften dieses Stereotypen.

Löschen

Entfernt den markierten Erweiterungspunkt aus dem Modell.

17.4.3. Eigenschaftsfelder des Erweiterungspunktes

Name

Textfeld. Der Name des Erweiterungspunktes.

Tipp

Sehr häufig werden die Erweiterungspunkte während der Anwendungsfallanalyse unbenannt gelassen, weil sie immer anhand ihrer Erstellungsorte (innerhalb des Anwendungsfalles und der erweiterten Beziehung) aufgelistet werden.

Anmerkung

ArgoUML erzwingt keine Namenskonvention für Erweiterungspunkte.

Ort


Tipp

Erweiterungspunkte werden immer anhand ihres Erstellungsortes (innerhalb des Anwendungsfalles und der erweiterten Beziehungen) aufgelistet. Typischerweise wird die Nummer/der Name des Absatzes in der Spezifikation sein.

Basis-Anwendungsfall


Extend

Textfeld. Listet alle Anwendungsfälle auf, die den Basis-Anwendungsfall über diesen Erweiterungspunkt erweitern.

Dort, wo ein Erweiterungspunkt existiert, wird ein Taste 1-Doppelklick zu der Beziehung springen.
17.5. Assoziation

Eine Assoziation in einem Anwendungsfalldiagramm repräsentiert eine Beziehung zwischen einem Akteur und einem Anwendungsfall und zeigt, wie der Akteur in diesen Anwendungsfall eingebunden ist. Der Aufruf des Anwendungsfalles wird einige (signifikante) Änderungen zur Folge haben, die der Akteur wahrnimmt.

Assoziationen sind unter Klassendiagramme (siehe Abschnitt 18.12, „Assoziation“) vollständig beschrieben.

17.6. Assoziationsenden

Assoziationsenden sind unter Klassendiagramme (siehe Abschnitt 18.13, „Assoziationsende“) beschrieben.

17.7. Abhängigkeit

Abhängigkeiten sind unter Klassendiagramme (siehe Abschnitt 18.14, „Abhängigkeit“) beschrieben.

Achtung

Abhängigkeiten haben in Anwendungsfalldiagrammen wenig Sinn. Sie sind Notwendig, weil frühere Versionen von ArgoUML sie nutzten (unkorrektierweise), um include- und extend-Beziehungen zu implementieren.

17.8. Generalisierung


Beispiele in einem Verkaufssystem für reisende Verkäufer könnte der Anwendungsfall des Buchens als eine Generalisierung des Anwendungsfalles Flugbuchung und ein Akteur Verkäufer eine Generalisierung des Akteurs Kontrolleur (da Kontrolleure auch als Verkäufer fungieren können, aber nicht umgekehrt)

Die Generalisierung ist die Analogie zur Klassenvererbung in der OO-Programmierung.

Anmerkung


Innerhalb des UML-Metamodells ist die Generalisierung eine Subklasse der Beziehung.

Eine Generalisierung wird mit einem Pfeil, mit weiss ausgefüllter Spitze vom spezialisierten Anwendungsfall oder Akteur zum generalisierenden Anwendungsfall oder Akteur dargestellt (siehe Abbildung 17.1, „Typische Modellelemente in einem Anwendungsfalldiagramm.“).
17.8.1. Detail-Register Generalisierung

Die für Assoziationen aktiven Detail-Register sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung
Standard-Register.

Anmerkung

Die Werte im Feld "Begrenzung" der Generalisierung kann nicht editiert werden, weil sie durch die Eigenschaften der Endpunkte der Linie bestimmt werden.

Stereotyp
Standard-Register.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für die Generalisierung die folgenden Standard-Eigenschaftswerte definiert.

• derived (von der Superklasse ModellElement). Der Wert true bedeutet, dass die Generalisierung redundant ist - sie kann formal aus anderen Elementen abgeleitet werden, oder false wenn sie es nicht kann.

Anmerkung

Abgeleitete Generalisierungen haben Ihren Wert in der Analyse, um nützliche Namen oder Konzepte einzuführen und im Design, um eine wiederholte Berechnung zu verhindern.

17.8.2. Eigenschaftssymbolleiste Generalisierung

Nach oben

Geht in der Paketstruktur des Modelles nach oben. Bei der Generalisierung ist dies das Paket, das die Generalisierung enthält.

Neuer Stereotyp

Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Generalisierung und springt sofort in das Register Eigenschaften dieser Generalisierung.
Entfernt die markierte Generalisierung aus dem Modell.

**Warnung**

Die ist ein Entfernen aus dem Modell und *nicht* nur aus dem Diagramm. Um eine Generalisierung aus einem Diagramm zu entfernen, aber diese im Modell zu erhalten, verwenden Sie das Hauptmenü **Aus Diagramm entfernen** (oder drücken Sie die Taste Entf).

### 17.8.3. Eigenschaftsfelder der Generalisierung

**Name**

**Tipp**

In der Anwendungsfallanalyse wird die Generalisierung sehr häufig nicht benannt.

**Anmerkung**

ArgoUML erzwingt keine Namenskonvention für Assoziationen.

**Anmerkung**

Es gibt keine Darstellung des Namens einer Generalisierung im Diagramm.

**Textfeld**. Der Name des Diskriminators für die Spezialisierung. UML 1.4 erlaubt das Gruppieren von Spezialisierungen in eine Anzahl von Gruppen auf Basis dieses Wertes.

**Tipp**


Dies ist ein kleiner Punkt, wie dies innerhalb eines Anwendungsfalldiagrammes genutzt werden kann.

**Namensraum**

Textfeld mit Navigationsschaltfläche. Gibt den Namensraum der Generalisierung an. Dies ist die
Pakethierarchie.

Superklasse


Subklasse


Powertype

Drop-Down-Auswahl mit Zugriff auf alle Standard-UML-Typen, die durch ArgoUML unterstützt werden und allen neuen Klassen, die innerhalb des aktuellen Modelles erzeugt wurden.

Dies ist der Typ der Subklasse der Generalisierung.

Tipp


Abbildung 17.2. Beispiel eines Diskrimintators mit Generalisierung
17.9. Extend

Extend ist eine Beziehung zwischen zwei Anwendungsfällen. Wenn A B erweitert, bedeutet das, dass A zusätzliches Verhalten beschreibt, das bedingungsabhängig (unter bestimmten Umständen) irgendwann während des normalen Ablaufes von B ausgeführt wird.

In gewisser Hinsicht ist Extend der Generalisierung ähnlich. Der Hauptunterschied ist jedoch, dass die erweiterten Anwendungsfälle Erweiterungspunkte definieren (siehe Abschnitt 17.4, „Erweiterungspunkt“), welches die einzigen Stellen sind an denen sein Verhalten erweitert werden darf. Der erweiterte Anwendungsfall muss definieren, an welchem dieser Erweiterungspunkte er das Verhalten hinzufügt.

Dadurch ist die Verwendung von Extend strenger kontrolliert als die allgemeine Erweiterung. Sie wird daher bevorzugt, wo immer dies möglich ist.

Beispiele in einem Verkaufssystem für reisende Vertriebsmitarbeiter könnten die Anwendungsfälle für das Bezahlen eines Tickets sein, das einen Erweiterungspunkt in der in der Spezifikation des Bezahlvorganges hat. Erweiternde Anwendungsfälle können diese an diesem Punkt erweitern (extend), um mit Bargeld, Kreditkarte usw. zu bezahlen.

Innerhalb des UML-Metamodellses ist Extend eine Subklasse von Relationship (Beziehung).

Eine Extend-Beziehung wird als gepunktete Verbindung mit einer nicht ausgefüllten Pfeilspitze und der Bezeichnung «extend» dargestellt. Wenn eine Bedingung definiert wurde, wird diese unter der Bezeichnung «extend» angezeigt (siehe Abbildung 17.1, „Typische Modellelemente in einem Anwendungsfallesdiagramm.“).

17.9.1. Detail-Register Extend
Die Detail-Register, die bei einer Extend-Beziehung aktiv sind, sind folgende.

**Anmerkung**

Es gibt kein Register Quellcode, da es keinen Quellcode gibt, der für eine Extend-Beziehung generiert werden könnte.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung
Standard-Register

**Anmerkung**

Die Werte im Feld "Begrenzung" von Extend ist nicht editierbar, weil sie durch die Eigenschaften der Endpunkte dieser Linie bestimmt werden.

Stereotyp
Standard-Register.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für Extend die folgenden Standard-Eigenschaftswerte definiert.

- **derived** (von der Superklasse, ModelElement). Der Wert true bedeutet, dass die Extend-Beziehung redundant ist - sie kann formal von anderen Elementen abgeleitet werden, oder false, wenn sie nicht abgeleitet werden kann.

**Anmerkung**

Abgeleitete Extend-Beziehungen können ihren Wert in der Analyse haben, um nützliche Namen oder Konzepte einzuführen.

### 17.9.2. Extend Eigenschaftssymbolleiste

Nach oben
Navigiert in der Paketstruktur des Modelles nach oben. Bei Extend ist dies das Paket, in dem sich die Erweiterung befindet.

Neuer Erweiterungspunkt
Erzeugt einen neuen Anwendungsfall-Erweiterungspunkt im Namensraum der aktuellen Extend-Beziehung, mit der aktuellen Extend-Beziehung als dessen erste erweiternde Beziehung.

**Tipp**

Es ist zwar völlig in Ordnung, einen Erweiterungspunkt von einer Extend-Beziehung aus zu erzeugen. Der erzeugte Erweiterungspunkt wird aber keinen mit ihm verknüpften Anwendungsfall aufweisen (er kann später eingerichtet werden).

Üblicher wäre es, einen Erweiterungspunkt innerhalb eines Anwendungsfalles zu erzeugen und diesen anschliessend mit einer Extend-Beziehung zu verknüpfen (siehe Abschnitt 17.9.3, „Eigenschaftsfelder für Extend“ unten.

**Neuer Stereotyp**

Erzeugt einen neuen Stereotypen (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Extend-Beziehung und springt unmittelbar in das Register Eigenschaften dieses Stereotypen.

**Löschen**

Enfernt die markierte Extend-Beziehung aus dem Modell.

**Warnung**

Diese Funktion löscht aus dem Modell, *nicht* nur aus dem Diagramm. Um eine Extend-Beziehung aus einem Diagramm zu entfernen, diese aber im Modell zu erhalten, nehmen Sie *Aus Diagramm entfernen* aus dem Hauptmenü (oder drücken Sie die Taste Entf).

### 17.9.3. Eigenschaftsfelder für Extend

**Name**

Textfeld. Der Name der Extend-Beziehung.

**Tipp**

Sehr häufig läßt man die Extend-Beziehungen in der Anwendungsfall-Analyse unbenannt.

**Anmerkung**

ArgoUML erzwingt keine Namenskonvention für Extend-Beziehungen.

**Namensraum**

Ein Taste 1-Doppelklick auf den Eintrag navigiert in das Paket, das diesen Namensraum definiert (oder in das Modell mit dem obersten Namensraum).

**Basis-Anwendungsfall**


**Erweiterung**


**Erweiterungspunkte**

Textfeld. Listet die Erweiterungspunkte des Basis-Anwendungsfalles auf, bei denen die Erweiterung ausgeführt wird, wenn die Bedingung eintritt.

**Anmerkung**


Daher ist die Reihenfolge der Erweiterungspunkte irrelevant, ausser der Position des ersten Erweiterungspunktes; weil dieser bestimmt, wo die Bedingung geprüft wird.

Dort, wo ein Erweiterungspunkt erstellt wurde, wird ein Taste 1-Doppelklick zu dieser Beziehung navigieren. Taste 2 öffnet ein Popup-Menü mit den folgenden Einträgen.

- **Hinzufügen.** Das Fenster „Erweiterungspunkte hinzufügen/entfernen“ öffnet sich. In diesem Fenster ist es möglich, eine Liste der Erweiterungspunkte zu erstellen.
- **Neu.** Fügt einen neuen Erweiterungspunkt in die Liste ein und navigiert dort hin. Die aktuelle Extend-Beziehung wird als erste in die Liste der erweiternden Beziehungen des neuen Erweiterungspunktes aufgenommen.
- **Nach oben.** Nur verfügbar, wenn zwei oder mehr Erweiterungspunkte aufgeführt sind und der markierte Erweiterungspunkt sich nicht oben befindet. Sie bewegt den Erweiterungspunkt um eine Position nach oben.
- **Nach unten.** Nur verfügbar, wenn zwei oder mehr Erweiterungspunkte aufgeführt sind und der markierte Erweiterungspunkt sich nicht unten befindet. Sie bewegt den Erweiterungspunkt um eine Position nach unten.

**Bedingung**


Der hier eingegebene Text wird im Diagramm angezeigt.
17.10. Include

Include ist eine Beziehung zwischen zwei Anwendungsfällen. Wenn A B includiert, bedeutet das, dass B das Verhalten beschreibt, das in die Beschreibung des Verhaltens von A an einigen Punkten (intern in A definiert) einfügt.

Beispiele in einem Verkaufssystem für reisende Verkäufer könnte der Anwendungsfall „Reise buchen“ sein, der die Anwendungsfälle für das Buchen von Flügen und die Handhabung der Bezahlung beinhalten.

Innerhalb des UML-Metamodells ist Include eine Subklasse von Beziehung.

Eine Include-Beziehung wird als gepunktete Verbindung mit einem offen Pfeil und der Bezeichnung «include» dargestellt (siehe Abbildung 17.1, „Typische Modellelemente in einem Anwendungsfalldiagramm.“).

17.10.1. Detail-Register Include

Die aktiven Detail-Register für Include-Beziehungen sind folgende.

Anmerkung

Es gibt kein Register Quellcode, weil es keinen Quellcode gibt, der für eine Include-Beziehung generiert werden könnte.

Zu-Bearbeiten-Element

Standard-Register.

Eigenschaften

Siehe Abschnitt 17.10.2, „Include Eigenschaftssymbolleiste“ und Abschnitt 17.10.3, „Eigenschaftsfelder für Include“ unten.

Dokumentation

Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung

Standard-Register

Anmerkung

Die Werte im Feld "Begrenzung" der Include-Beziehung sind nicht editierbar, weil sie durch die Eigenschaften der Endpunkte der Linie bestimmt werden.

Eigenschaftswerte

Standard-Register. Im UML-Metamodell hat Include die folgenden Standard-Eigenschaftswerte definiert.

• derived (von der Superklasse ModelElement. Der Wert true bedeutet, dass die Include-Beziehung redundant ist - sie kann formal von anderen Elementen abgeleitet werden, oder false bedeutet, sie kann nicht abgeleitet werden.
Anmerkung
Abgeleitete Include-Beziehungen können ihren Wert in der Analyse haben, um nützliche Namen oder Konzepte einzubeziehen.

17.10.2. Include Eigenschaftssymbolleiste

Nach oben
Navigiert in der Paketstruktur des Modells nach oben. Bei Include ist dies das Paket, das den Include beinhaltet.

Neuer Stereotyp
Erzeugt einen neuen Stereotypen (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Include-Beziehung und springt sofort in das Eigenschaftsregister dieses Stereotypes.

Lösche
Entfernt die markierte Include-Beziehung aus dem Modell.

Warnung
Diese Funktion löscht aus dem Modell und \emph{nicht} nur aus dem Diagramm. Um eine Include-Beziehung aus dem Diagramm zu löschen, sie aber im Modell zu erhalten, verwenden Sie den Eintrag \emph{Aus Diagramm entfernen} des Hauptmenüs (oder drücken Sie die Taste Entf).

17.10.3. Eigenschaftsfelder für Include

Name
Textfeld. Der Name der Include-Beziehung.

Tipp
Sehr häufig bleibt die Include-Beziehung in der Analyse unbenannt.

Anmerkung
ArgoUML erzwingt keine Namenskonvention für Include-Beziehungen.

Namensraum
Textfeld. Gibt den Namensraum für die Include-Beziehung wieder. Dies entspricht der Pakethierarchie.
Ein Taste 1-Klick auf diesen Eintrag springt zu dem Paket, das diesen Namensraum definiert (oder das Modell als oberster Namensraum).

**Basis-Anwendungsfall**


**Eingefügter Anwendungsfall**

Kombinationsfeld. Gibt den Anwendungsfall wieder, der durch diese Include-Beziehung eingefügt wird. Ein Taste 1-Klick auf diesen Eintrag öffnet ein DropDown-Menü aller verfügbaren Anwendungsfälle (und ein leerer Eintrag), die durch einen Taste 1-Klick ausgewählt werden können.
Kapitel 18. Modellelement-Referenz
Klassendiagramm

18.1. Einleitung

Dieses Kapitel beschreibt jedes Modellelement, das innerhalb eines Klassendiagrammes erzeugt werden kann. Beachten Sie, dass einige Submodell-Elemente von Modellelementen nicht selbst im Diagramm erscheinen können.

Klassendiagramme werden nur für eines der statischen UML-Strukturdiagramme verwendet, das Klassendiagramm selbst. Objektdiagramme werden durch ArgoUML-Verteilungsdiagramme repräsentiert.

Zusätzlich verwendet ArgoUML das Klassendiagramm, um die Modellstruktur durch die Nutzung von Paketen darzustellen.

Es gibt eine enge Beziehung zwischen diesem Material und dem Register Eigenschaften des Detailfensters (siehe Abschnitt 13.3, „Das Register Eigenschaften“). Dessen Abschnitt behandelt die Eigenschaften allgemein, in diesem Kapitel sind sie mit spezifischen Modellelementen verknüpft.

Abbildung 18.1, „Denkbare Modellelemente in einem Klassendiagramm.“ zeigt ein Klassendiagramm mit allen denkbaren Modellelementen.


Abbildung 18.2, „Denkbare Modellelemente in einem Paketdiagramm.“ zeigt ein Paketdiagramm mit
allen denkbaren Modellelementen.

Abbildung 18.2. Denkbare Modellelemente in einem Paketdiagramm.

Abbildung 18.3. „Denkbare Modellelemente in einem Datentypdiagramm.“ zeigt ein Datentyp-Diagramm mit einem Datentyp und einer Aufzählung.

Abbildung 18.3. Denkbare Modellelemente in einem Datentypdiagramm.

Abbildung 18.4. „Denkbare Modellelemente in einem Stereotyp-Definitionsdiagramm.“ zeigt ein Stereotyp-Definitionsdiagramm mit allen denkbaren Modellelementen.

Abbildung 18.4. Denkbare Modellelemente in einem Stereotyp-Definitionsdiagramm.
18.1.1. Einschränkungen bei Klassendiagrammen in ArgoUML

In V0.26 von ArgoUML existieren verschiedene Einschränkungen bei Stereotyp-Definitionsdiagrammen. Z.B. erlaubt die Implementierung keine Stereotyp-Bereiche, die in Stereotyp-Definitionsdiagrammen dargestellt werden können.


18.2. Paket

Das Paket ist das hauptsächlich organisierende Modellelement in ArgoUML. Im UML-Metamodell ist es eine Subklasse von Namensraum und GeneralisierbaresElement.

Anmerkung

ArgoUML implementiert auch das UML-Modellelement Modell als Subklasse von Paket, aber nicht das Modellelement Subsystem.

ArgoUML implementiert auch einige weniger häufige Aspekte der UML-Modellverwaltung. Im Einzelnen ist die Beziehung UML 1.4 als Generalisierung definiert und die Subklassen-Abhängigkeit Berechtigung wird zwischen Paketen verwendet.

18.2.1. Detail-Register Paket

Die für Pakete aktiven Detail-Register sind folgende.

Zu-Bearbeiten-Element

Eigenschaften
Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung

Stereotyp
Standard-Register.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell hat das Paket die folgenden definierten Eigenschaftswerte.


**Anmerkung**
Abgeleitete Pakete haben Ihren Wert in der Analyse, um nützliche Namen oder Konzepte einzufügen und im Design, um eine wiederholte Verarbeitung zu verhindern.

### 18.2.2. Paket Eigenschaftssymbolleiste

- **Nach oben**
  Navigiert in der Paketstruktur nach oben.

- **Neues Paket**
  Erzeugt ein neues Paket innerhalb des Paketes (das in keinem Diagramm erscheint) und springt sofort in das Register Eigenschaften dieses Paketes.

- **Neuer Datentyp**

- **Neue Enumeration (Aufzählung)**
  Erzeugt eine neue Enumeration (Aufzählung; siehe Abschnitt 16.4, „Enumeration (Aufzählung)“) im markierten Paket und springt sofort in das Register Eigenschaften dieser Enumeration (Aufzählung).

- **Neuer Stereotyp**
  Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) im markierten Paket und springt sofort in das Register Eigenschaften dieses Stereotypen.

- **Neue Eigenschaftsdefinition**
Erzeugt eine neue Eigenschaftsdefinition (siehe Abschnitt 16.7, „Eigenschaftsdefinition“) innerhalb des Paketes (das in keinem Diagramm erscheint) und springt sofort in das Register Eigenschaften dieser Eigenschaftsdefinition.

**Paket löschen**

Entfernt das Paket aus dem Modell.

**Warnung**

Dies ist ein Entfernen aus dem Modell und *nicht* nur aus dem Diagramm. Um ein Paket aus dem Diagramm zu löschen, es aber innerhalb des Modells zu erhalten, verwenden Sie Aus Diagramm entfernen im Hauptmenü (oder drücken Sie die Taste Entf).

### 18.2.3. Eigenschaftsfelder für ein Paket

#### Name

Textfeld. Der Name des Paketes. Der Name eines Paketes ist, wie alle Pakete, wird per Konvention in Kleinbuchstaben geschrieben und enthält keinerlei Satzzeichen.

#### Anmerkung

Standardmäßig hat ein neues Paket keinen definierten Namen. Das Paket wird mit dem Namen (unbenanntes Paket) im Explorer erscheinen.

#### Namensraum


#### Sichtbarkeit

Auswahlfeld mit vier Einträgen public, private, protected und package. Gibt an, ob das Paket außerhalb des Paketes sichtbar ist.

#### Modifizierer

Markierfelder mit den Einträgen abstract, leaf und root.

- **Abstract** wird verwendet, um zu deklarieren, daß dieses Paket nicht instanziert werden kann, aber immer spezialisiert werden muss.

#### Tipp

Die Bedeutung von abstract, angewendet auf ein Paket ist nicht ganz klar. Es könnte bedeuten, dass das Paket Schnittstellen oder abstrakte Klassen ohne Realisierung enthält. Dies würde wahrscheinlich besser über Stereotypen auf das Paket gehandhabt (z.B. «facade»).

- **Leaf (Blatt)** gibt an, dass dieses Paket keine weiteren Subpakete haben darf.
- **Root (Wurzel)** gibt an, dass dies das Paket auf oberster Ebene ist.


Tipp

Innerhalb von ArgoUML kann Root (Wurzel) nur auf das Modell angewendet werden, da alle Pakete sich innerhalb des Modell es befinden. Dies kann dazu verwendet werden, um zu betonen, dass sich das Modell auf oberste Ebene befindet.

**Generalisierung**

Textbereich. Listet alle Pakete auf, die dieses Paket *generalisieren*.

Ein Taste 1-Doppelklick navigiert zur Generalisierung und öffnet dessen Register Eigenschaften.

**Spezialisierungen**

Textfeld. Listet alle spezialisierten Pakete auf (z.B. für die dieses Paket eine Generalisierung ist).

Ein Taste 1-Doppelklick navigiert zu der Spezialisierung und öffnet dessen Register Eigenschaften.

**Eigene Elemente**


Ein Taste 1-Doppelklick auf eines der aufgelisteten Einträge navigiert zu diesem Modellelement.

**Importierte Elemente**

Textbereich. Eine Liste aller importierten Elemente, z.B. Elemente, die zu einem anderen Paket gehören, aber ausdrücklich in diesem Paket sichtbar gemacht wurden.


- Entfernen. Entfernt das importierte Element.

**18.3. Datatyp**

Datentypen sind nicht spezifisch für Paket- oder Klassendiagramme und werden in den Kapiteln der Modellelemente auf oberster Ebene diskutiert (siehe Abschnitt 16.3, „Datatyp“).

**18.4. Enumeration (Aufzählung)**

Enumerationen (Aufzählungen) sind nicht spezifisch für Paket- oder Klassendiagramme und werden in den Kapiteln der Modellelemente auf oberster Ebene diskutiert (siehe Abschnitt 16.4, „Enumeration (Aufzählung)“).

**18.5. Stereotyp**
Stereotypen sind nicht spezifisch für Paket- oder Klassendiagramme und werden in den Kapiteln der Modellelemente auf oberster Ebene diskutiert (siehe Abschnitt 16.6, „Stereotyp“).

18.6. Klasse

Die Klasse ist das dominante Modellelement in einem Klassendiagramm. Im UML-Metamodell ist sie eine Subklasse von Classifier und GeneralizableElement.

Eine Klasse wird in einem Klassendiagramm als Rechteck mit drei Bereichen dargestellt. Der oberste Bereich zeigt den Klassennamen (und Stereotypen) an, der zweite Bereich alle Attribute und der Dritte alle Operationen. Diese letzten zwei Bereiche können optional versteckt werden.

18.6.1. Detail-Register Klasse

Die aktiven Detail-Register für Klassen sind folgende.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung

Quellcode
Standard-Register. Enthält ein Template für die Klassendeklaration und Deklarationen der verküpften Klassen.

Randbedingungen

Stereotypen
Standard-Register.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell hat die Klasse die folgenden Standard-Eigenschaftswerte definiert.

- persistence (von der Superklasse Classifier). Die Werte transitory, gibt an, dass der Zustand gelöscht wurde, nachdem die Instanz gelöscht oder persistent wurde, mit der Kennzeichnung, dass der Zustand erhalten bleibt, wenn die Instanz gelöscht wurde.

- semantics (von der Superklasse Classifier). Der Wert ist eine Spezifikation der Semantik der Klasse.

- derived (von der Superklasse ModelElement). Der Wert true bedeutet, dass die Klasse
redundant ist - sie kann formal von anderen Elemente abgeleitet werden. Oder false, wenn sie nicht abgeleitet werden kann.

**Anmerkung**
Abgeleitete Klassen haben ihren Wert bei der Analyse, um nützliche Namen oder Konzepte einzubinden und in der Analyse, um die wiederholte Verarbeitung zu verhindern.

**Anmerkung**
Die UML Element Metaklasse, aus der alle anderen Modellelemente abgeleitet werden, enthält das Eigenschaftselement documentation, das durch das Register Dokumentation in ArgoUML bearbeitet wird.

**Checkliste**
Standard-Register für einen Klassifizierer.

### 18.6.2. Eigenschaftssymbolleiste Klasse

- **Nach oben**
  Navigiert in der Paketstruktur nach oben.

- **Neues Attribut**
  Erzeugt ein neues Attribut (siehe Abschnitt 18.7, „Attribute“) in der Klasse und navigiert sofort in das Register Eigenschaften dieses Attributes.

- **Neue Operation**
  Erzeugt eine neue Operation (siehe Abschnitt 18.8, „Operation“) in der Klasse und navigiert sofort in das Register Eigenschaften dieser Operation.

- **Neues Empfangssignal**
  Erzeugt ein neues Empfangssignal und navigiert sofort in das Register Eigenschaften dieses Empfangssignales.

- **Neue innere Klasse**

**Anmerkung**
Die innere Klasse ist kein separates Konzept in der UML. Sie ist eine praktische Abkürzung für eine Klasse, die auf den Namensraum seiner Eigentümerklasse beschränkt ist.
Neue Klasse
Erzeugt im gleichen Namensraum der aktuellen Klasse eine neue Klasse (die in keinem Diagramm erscheint).

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Klasse und navigiert sofort in das Register Eigenschaften dieses Stereotypen.

Löschen
Löscht die Klasse aus dem Modell.

Warnung
Dies ist ein Entfernen aus dem Modell und nicht nur aus dem Diagramm. Um eine Klasse aus einem Diagramm zu entfernen, diese aber im Modell zu erhalten, verwenden Sie Aus Diagramm entfernen im Hauptmenü (oder drücken die Tasten Entf).

18.6.3. Eigenschaftsfelder für eine Klasse

Name
Textfeld. Der Name der Klasse. Der Name der Klasse beginnt mit einem Großbuchstaben und mit Wörtern, die durch „wechselnde Groß-/Kleinschreibung“ getrennt werden.

Anmerkung
Die ArgoUML-Kritik wird sich über Klassennamen beschweren, die nicht mit einem Großbuchstaben beginnen.

Namensraum

Ein Taste 1-Klick auf den Eintrag verschiebt die Klasse in den markierten Namensraum.

Modifizierer
• Abstract wird verwendet, um zu deklarieren, dass diese Klasse nicht instanziert werden kann, aber immer Spezialisiert werden muss. Der Name einer abstrakten Klasse wird im Diagramm kursiv dargestellt.

Achtung
Wenn eine Klasse eine abstrakte Operation aufweist, dann sollte sie als abstrakt deklariert werden. ArgoUML erzwingt dies allerdings nicht.
• Leaf(Blatt) gibt an, dass diese Klasse nicht weiter spezialisiert werden kann, während Root (Wurzel) angibt, dass sie keine weitere Subklasse haben kann. Es ist bei einer Klasse möglich, dass Sie abstrakt und Leaf (Blatt) ist, weil seine statischen Operationen referenziert werden können.

• Active gibt an, dass diese Klasse ein dynamisches Verhalten aufweist (und ist daher mit einem Zustands- oder Aktivitätsdiagramm verknüpft).

**Sichtbarkeit**
Auswahlfelder mit vier Einträgen public, private, protected und package. Gibt an, ob die Klassen außerhalb des Namensraumes sichtbar ist.

**Abhängig von**

Ein Taste 1-Doppelklick navigiert zu der Abhängigkeit und öffnet dessen Register Eigenschaften.


**Notwendig für**
Textbereich. Listet alle Enden der Beziehung auf, die von dieser Klasse abhängen. Z.B. das Ende, welches notwendig für das andere Ende ist.

Ein Taste 1-Doppelklick navigiert zu der abhängigen Klasse und öffnet dessen Register Eigenschaften.


**Generalisierungen**
Textbereich. Listet jede Klasse auf, die diese Klasse generalisiert.

Ein Taste 1-Doppelklick navigiert zur Generalisierung und öffnet dessen Register Eigenschaften.

**Spezialisierungen**
Textfeld. Listet jede spezialisierte Klasse auf (z.B. für die diese Klasse eine Generalisierung ist).

Ein Taste 1-Doppelklick navigiert zu der Spezialisierung und öffnet dessen Register Eigenschaften.

**Attribute**


**Endpunkte (Assoziation)**
Textfeld. Listet jeden Endpunkt der Assoziationen auf (siehe Abschnitt 18.12, „Assoziation“), die mit dieser Klasse verbunden sind.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

**Operationen**

- **Nach oben.** Nur verfügbar, wenn mehr als zwei Operationen gelistet sind und sich die markierte Operation nicht ganz oben befindet. Bewegt die Operation um eine Position nach oben.
- **Nach unten.** Nur verfügbar, wenn mehr als zwei Operationen gelistet sind und sich die markierte Operation nicht ganz unten befindet. Bewegt die Operation um eine Position nach unten.

**Eigene Elemente**


Ein Taste 1-Doppelklick auf eines der Modellelemente navigiert zu diesem Modellelement.

**Tipp**


### 18.7. Attribute

Ein Attribut ist ein benanntes Element innerhalb einer Klasse (oder ein anderer Klassifizierer), das den Wertebereich beschreibt, den eine Klasseninstanz aufweisen kann. Im UML-Metamodell ist dies eine Subklasse von StructuralFeature, die widerum selbst ein Subklasse von Feature ist.

Ein Attribut wird in einem Diagramm innerhalb des Attributbereiches der Klasse als separate Zeile dargestellt. Seine Syntax ist wie folgt:

\[ \text{Sichtbarkeit AttributeName : Typ [= Anfangswert]} \]

Die Sichtbarkeit ist +, #, - oder ~ entsprechend den Werten public, protected, private, oder package.

AttributeName ist der aktuelle Name des deklarierten Attributes.

Typ ist der für das Attribut deklarierte Typ (UML-Datentyp, Klasse oder Schnittstelle).


Desweiteren wird der ganze Eintrag eines als static deklarierten Attributes im Diagramm unterstrichen.

#### 18.7.1. Detail-Register Attribut

Die für Attribute aktiven Detail-Register sind folgende.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Randbedingungen

Stereotyp
Standard-Register.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind die folgenden Eigenschaftswerte für Attribute definiert.

- transient.

- volatile. Dies ist eine ArgoUML-Erweiterung des UML 1.4-Standards, um anzuzeigen, dass dieses Attribut in flüchtiger Form realisiert wurde (z. B. könnte es ein dem Speicher zugeordnetes Steuerregister sein).

Anmerkung
Die UML-Metaklasse Element, von der alle anderen Modellelemente abgeleitet sind, enthält den Eigenschaftswert documentation, der in ArgoUML über das Register Dokumentation verwaltet wird.

Checkliste
Standard-Register für ein Attribut.

18.7.2. Eigenschaftssymbolleiste Attribut

Nach oben
Navigiert in der Paketstruktur nach oben.

Vorheriges
Navigiert zum vorherigen Attribut der Klasse. Diese Schaltfläche ist deaktiviert, wenn das aktuelle Attribut das Erste ist.

Nächstes
Navigiert zum nächsten Attribut der Klasse. Diese Schaltfläche ist deaktiviert, wenn das aktuelle Attribut das Erste ist.

Neues Attribut
Erzeugt ein neues Attribut innerhalb der Klasse und springt sofort in das Register Eigenschaften dieses Attributes.
Tipp

Dies ist ein sehr bequemer Weg, mehrere Attribute, eines nach dem anderen, zu der Klasse hinzuzufügen.

**Neuer Datentyp**

Erzeugt einen neuen Datentyp (siehe Abschnitt 16.3, „Datatyp“) für das markierte Attribut und springt sofort in das Register Eigenschaften dieses Datentyps.

**Neue Enumeration (Aufzählung)**

Erzeugt eine neue Enumeration (Aufzählung; siehe Abschnitt 16.4, „Enumeration (Aufzählung)“) für das Paket, welches die Klasse enthält und springt sofort in das Register Eigenschaften dieser Enumeration (Aufzählung).

**Neuer Stereotyp**

Erzeugt einen neuen Stereotypen (siehe Abschnitt 16.6, „Stereotyp“) für das markierte Attribut und springt sofort in das Register Eigenschaften dieses Stereotypen.

**Löschen**

Löscht das Attribut aus dem Modell.

**Warnung**


### 18.7.3. Eigenschaftsfelder für Attribute

**Name**

Textfeld. Der Name des Attributes. Der Name eines Attributes beginnt mit einem Kleinbuchstaben und mit Worten die durch Groß-/Kleinschreibung getrennt sind.

**Anmerkung**

Die ArgoUML-Kritiken werden sich über Attributnamen beschweren, die nicht mit einem Kleinbuchstaben beginnen.

**Eigentümer**

Textfeld. Gibt die Klasse wieder, die dieses Attribut enthält.

Ein Taste 1-Doppelklick auf diesen Eintrag navigiert zu der Klasse.
Kardinalität

Editierbares Auswahlfeld mit Markierfeld. Der Standardwert (1) heißt, dass es eine Attributinstanz für jede Klasseninstanz gibt. Z.B. es handelt sich um ein skalares Attribut. Das Auswahlfeld enthält eine Anzahl häufig verwendeter Spezifikationen für nicht-skalare Attribute.

Wenn das Markierfeld unmarkiert ist, dann bleibt die Kardinalität im Modell undefiniert (und das Auswahlfeld ist deaktiviert).

Anmerkung

ArgoUML stellt eine Anzahl vordefinierter Kardinalitätsbereiche für die vereinfachte Auswahl zur Verfügung. Der Anwender kann aber jeden benutzerdefinierten Bereich eingeben, der der UML-Syntax folgt, wie z.B. „1..3,7,10“.

Der Wert 1..1 entspricht dem Standard (exact einer skalaren Instanz). Die Auswahl 0..1 kennzeichnet ein optional skalares Attribut.

Sichtbarkeit

Auswahlfeld mit den vier Einträgen public, private, protected und package.

• public. Das Attribut ist für jedes Modellelement verfügbar, welches die Klasse sehen kann.

• private. Das Attribut ist nur innerhalb der Klasse verfügbar (und jeder inneren Klasse).

• protected. Das Attribut ist nur in der Klasse oder in Modellelementen verfügbar, die Subklassen der Klasse sind.

• package. Das Attribut ist nur für Modellelemente verfügbar, die sich im gleichen Paket befinden.

Änderbarkeit

Auswahlfeld mit den Einträgen Nur hinzufügen, veränderbar und unveränderlich.


• veränderlich. Es gibt keine Einschränkungen hinsichtlich der Veränderbarkeit.


Modifizierer

Markierfeld für static. Ist dies nicht markiert (der Standard), dann hat das Attribut den Fokus „Instanz“. Ist es markiert, dann ist das Attribut static; es hat den Fokus „Klasse“. Statische Attribute werden im Diagramm unterstrichen dargestellt.

Typ

Das Drücken der Navigationsschaltfläche navigiert zum Eigenschaftsfenster des aktuell markierten Typs. (Siehe Abschnitt 18.6, „ Klasse “, Abschnitt 18.3, „ Datatyp “ und Abschnitt 18.16, „ Schnittstelle “).

**Anmerkung**

Ein Typ muss dekariert werden (er kann void sein). Standardmäßig ist in ArgoUML int als Typ voreingestellt.

**Anfangswert**

Textfeld mit 2 Bereichen. Dies erlaubt es Ihnen, wenn gewünscht, einen Anfangswert für das Attribut festzulegen (dies ist optional). Das DropDown-Menü erlaubt den Zugriff auf die vordefinierten Werte 0, 1, 2 und null.

Der linke Teil des Feldes enthält den Rumpf des Ausdrucks, der den Anfangswert bildet. Der rechte Teil definiert die Sprache, in der der Ausdruck geschrieben wurde.

Wenn Sie den Mauszeiger über diese Felder bewegen, wird ein Hinweis Rumpf oder Sprache erscheinen, um Ihnen zu helfen, welches Feld für was gedacht ist.

**Achtung**

Jede Konstruktor-Operation darf diesen Anfangswert ignorieren.

### 18.8. Operation

Eine Operation ist eine Dienstleistung, die von einem Objekt angefordert werden kann, um ein Verhalten auszulösen. Im UML- Metamodell ist sie eine Subklasse von BehavioralFeature, die widerum selbst eine Subklasse von Feature ist.

Im Diagramm wird eine Operation innerhalb des Operationen-Bereiches der Klasse durch eine Zeile dargestellt. Die Syntax ist wie folgt:

\[
\text{Sichtbarkeit Name (Parameterliste) : Rückgabetyp \{Eigenschafts-String\}}
\]

Sie können diese Zeile im Diagramm direkt editieren, indem Sie einen Doppelklick darauf ausführen. Alle Elemente sind optional und, wenn sie unspezifiziert bleiben, bleiben die alten Werte erhalten.


Die folgenden Eigenschaften werden in ihrer speziellen Bedeutung erkannt: abstract, concurrency, concurrent, guarded, leaf, query, root und sequential.

Die Sichtbarkeit ist +, #, - oder ~ und entspricht der public, protected, private Sichtbarkeit, oder der Sichtbarkeit package.

static und final erscheinen zusätzlich, wenn die Operation diese Modifizierer aufweist. Bei jeder static deklarierten Operation wird deren ganzer Eintrag im Diagramm unterstrichen.

Es können Null oder mehr Einträge in der Parameterliste durch Kommata getrennt vorhanden sein. Jeder Eintrag besteht aus einem Paar in der Form:
Name: Typ

Der Rückgabetyp ist der Typ des zurückgegebenen Ergebnisses (UML-Datentyp, Klasse oder Schnittstelle).

Schließlich wird der gesamte Eintrag kursiv dargestellt, wenn die Operation als abstract deklariert wurde.

### 18.8.1. Detail-Register Operation

Die aktiven Detail-Register für Operationen sind die folgenden.

**Zu-Bearbeiten-Element**
Standard-Register.

**Eigenschaften**

**Dokumentation**
Standard-Register. Siehe Abschnitt 13.4, „ Das Register Dokumentation “.

**Darstellung**
Standard-Register. Das Feld Begrenzung erlaubt das Editieren, aber die Änderungen haben keine Auswirkungen.

**Quellcode**
Standard-Register. Dieses enthält die Deklaration für die Operation.

**Randbedingungen**

**Eigenschaftswerte**
Standard-Register. Im UML-Metamodell sind für Operation die folgenden Standard-Eigenschaftswerte definiert.

- semantics. Der Wert ist eine Spezifikation der Semantik der Operation.

### Anmerkung

Abgeleitete Operationen können Ihren Wert in der Analyse besitzen, um nützliche Namen oder Konzepte einzuführen. Im Design, um die wiederholte Verarbeitung zu verhindern.

### Anmerkung

Die UML-Metaklasse Element, von der alle anderen Modellelemente abgeleitet
werden, beinhaltet den Eigenschaftswert documentation, der in ArgoUML im Register Dokumentation verwaltet wird.

Checkliste
Standard-Register für eine Operation.

18.8.2. Eigenschaftssymbolleiste Operation

Nach oben
Navigiert in der Paketstruktur nach oben.

Neue Operation
Erzeugt eine neue Operation in der Klasse und navigiert sofort in das Register Eigenschaften dieser Operation.

Tipp
Dies ist ein sehr bequemer Weg, eine Anzahl von Operationen, eine nach der anderen, eine Klasse hinzuzufügen.

Neuer Parameter
Erzeugt einen neuen Parameter für die Operation und navigiert sofort in das Register Eigenschaften dieses Parameters.

Neues Signal
Erzeugt ein neues Signal für die Operation und navigiert sofort in das Register Eigenschaften dieses Signals.

Neuer Datentyp

Neuer Stereotyp
Erzeugt einen neuen Stereotypen (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Operation und springt sofort in das Register Eigenschaften dieses Stereotypen.

Löschen
Löscht die Operation aus dem Modell.

Warnung
18.8.3. Eigenschaftsfelder für Operation

**Name**
Textfeld. Der Name der Operation. Der Name einer Operation beginnt mit einem Kleinbuchstaben und besteht aus Wörtern, die durch Groß-/Kleinschreibung getrennt sind.

**Anmerkung**
Die ArgoUML-Kritiken werden sich über Operationsnamen beschweren, die nicht mit einem Kleinbuchstaben beginnen.

**Tipp**
Wenn Sie der Java-Konvention folgen wollen, dass Konstruktoren den gleichen Namen wie die Klasse haben, werden Sie diese Regel brechen. Bringen Sie diese Kritik zum Schweigen, indem Sie den Stereotypen create für die Konstruktor-Operation vergeben.

**Stereotyp**

**Tipp**
Sie sollten create als Stereotyp für Konstruktoren und destroy für Destruktoren (die über die „finalize“-Methoden in Java aufgerufen werden).

**Stereotype navigieren**
Symbol. Wenn ein Stereotyp markiert wurde, navigiert das Klicken auf die Taste 1 in das Eigenschaftsfenster des Stereotypen (siehe Abschnitt 18.5, „Stereotyp“).

**Eigentümer**
Textfeld. Gibt die Klasse wieder, die diese Operation enthält.
Ein Taste 1-Doppelklick auf diesen Eintrag navigiert zu der Klasse.

**Sichtbarkeit**
Auswahlfelder mit den Einträgen public, private, protected und package.
- **public.** Die Operation ist für jedes Modellelement verfügbar, dass die zugehörige Klasse sehen kann.
- **private.** Die Operation ist nur innerhalb der zugehörigen Klasse (und jeder inneren Klasse) verfügbar.
- **protected.** Die Operation ist nur in der zugehörigen Klasse oder den Modellelementen, die Subklassen der zugehörigen Klasse sind, verfügbar.
• **package.** Die Operation ist nur für die Modellelemente verfügbar, die sich im gleichen Paket befinden.

**Modifizierer**

Markierfelder mit den Einträgen abstract, leaf, root, query, und static.

• **abstract.** Diese Operation wird in dieser Klasse nicht implementiert. Die Implementierung muss in einer Subklasse enthalten sein.

  **Wichtig**

  Jede Klasse mit einer abstrakten Operation muss selbst als abstract deklariert werden.

• **leaf (Blatt).** Die Implementierung dieser Operation darf nicht durch eine Subklasse überschrieben werden.

• **root (Wurzel).** Die Deklaration dieser Operation darf nicht durch eine Deklaration einer Operation einer Subklasse überschrieben werden.

• **query (Abfrage).** Dies gibt an, dass die Operation keine Seiteneffekte haben darf (z.B. sie darf den Zustand des Systems nicht verändern). Sie kann nur einen Wert zurückgeben.

  **Achtung**

  Operationen benutzerdefinierter Datentypen müssen diesen Modifizierer immer prüfen.

• **static.** Es gibt nur eine Instanz von dieser, mit der Klasse verknüpften Operation (im Gegensatz zu einer Operation für jede Instanz einer Klasse). Dies ist das Attribut OwnerScope der Metaklasse Feature in UML. Jede static deklarierte Operation wird im Klassendiagramm unterstrichen dargestellt.

**Gleichzeitigkeit**

Auswahlfelder mit den Einträgen guarded, sequential und concurrent.

• **guarded.** Mehrere Aufrufe konkurrierender Threads können für eine Instanz gleichzeitig auftreten (bei jeder geschützten Operation), aber nur eine darf beginnen. Die anderen werden so lange blockiert, bis die erste Operation abgeschlossen ist.

  **Achtung**

  Der Systemdesigner muss sicherstellen, dass kein Deadlock auftreten kann. Es liegt in der Verantwortung der Operation, das blockierende Verhalten zu implementieren (im Gegensatz zum System).

• **sequential.** Nur ein Aufruf auf die Instanz (der Klasse mit der Operation) darf zu einer bestimmten Zeit anstehen. Es gibt keinen Schutz und keine Garantie des Verhaltens, wenn das System diese Regel verletzt.

• **concurrent.** Mehrere Aufrufe auf eine Instanz können gleichzeitig ausgeführt werden. Die Operation ist in der Lage, korrektes Verhalten sicherzustellen. Dies muss verwaltet werden, wenn es andere sequenzielle oder synchronisierte (geschützte) Operationen gibt, die zur
gleichen Zeit ausgeführt werden.

Parameter

Textbereich, mit Einträgen für alle Parameter der Operation (siehe Abschnitt 18.9, „Parameter“). Eine neue Operation wird immer mit einem neuen Parameter return erzeugt, um den Rückgabetyp der Operation zu definieren.


Ausgelöste Signale

Textbereich mit Einträgen für alle Signale (siehe Abschnitt 18.10, „Signal“), die durch diese Operation ausgelöst werden.

Achtung

Das aktuelle ArgoUML (V0.18) enthält nur einen eingeschränkten Support für Signale. Sie sind insbesondere nicht mit Signalereignissen verknüpft, die Zustandsautomaten aktivieren könnten.

Ein Taste 1-Doppelklick auf jedes der Signale navigiert zu diesem Signal.

18.9. Parameter

Ein Parameter ist eine Variable, die übergeben werden kann. Im UML-Metamodell ist er eine Subklasse von ModelElement.

Ein Parameter wird innerhalb der Operationsdeklaration im Operationenbereich der Klasse wie folgt dargestellt.

Name : Typ

Name ist der Name des Parameters.

Typ ist der Typ (UML-Datentyp, Klasse oder Schnittstelle) des Parameters.

Die Ausnahme ist ein Parameter, der einen Rückgabewert darstellt, dessen Typ nur am Ende der Operationsdeklaration angezeigt wird.

18.9.1. Detail-Register Parameter

Die aktiven Detail-Register für Parameter sind die folgenden.

Zu-Bearbeiten-Element
   Standard-Register.
Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Quellcode
Standard-Register. Enthält die Deklaration für einen Parameter.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell hat ein Parameter die folgenden definierten Standard-Eigenschaftswerte.

- **derived** (von der Superklasse ModelElement). Der Wert *true* bedeutet, dass der Parameter redundant ist - er kann formal von anderen Elementen abgeleitet werden oder *false*, wenn er nicht abgeleitet werden kann.

  **Achtung**

  Ein abgeleiteter Parameter ist ein bedeutungsloses Konzept.

  **Anmerkung**


18.9.2. Eigenschaftssymbolleiste Parameter

Nach oben
Navigiert in der Paketstruktur nach oben.

Neuer Parameter
Erzeugt einen neuen Parameter für die gleiche Operation des aktuellen Parameters und springt sofort in das Register Eigenschaften dieses Parameters.

**Tipp**
Dies ist ein bequemer Weg, um eine Reihe von Parametern der gleichen Operation hinzuzufügen.

Neuer Datentyp

270
Neuer Stereotyp

Erzeugt einen neuen Stereotypen (siehe Abschnitt 16.6, „Stereotyp“) für dem markierten Parameter und springt sofort in das Register Eigenschaften dieses Stereotyps.

Löschen

Löscht den Parameter aus dem Modell.

Warnung


18.9.3. Eigenschaftsfelder für Parameter

Name

Textfeld. Der Name des Parameters. Der Name des Parameters beginnt per Konvention mit einem Kleinbuchstaben und ist aus Wörter zusammengesetzt, die durch Groß-/Kleinschreibung getrennt werden.

Anmerkung

Die ArgoUML-Kritiken werden sich nicht über Parameternamen beschweren, die nicht mit einem Kleinbuchstaben beginnen.

Stereotyp

Kombinationsfeld. Es gibt keine UML-Standard-Stereotypen für Parameter.

Stereotyp navigieren

Symbol. Wenn ein Stereotyp markiert wurde, wird ArgoUML zum Eigenschaftsfenster dieses Stereotypen springen. (siehe Abschnitt 16.6, „Stereotyp“).

Eigentümer

Textbereich. Gibt die Operation wieder, die diesen Parameter beinhaltet.

Ein Taste 1-Doppelklick auf diesen Eintrag navigiert zu der entsprechenden Operation.

Typ


Anmerkung

Ein Typ muss deklariert sein (er kann void sein, dies macht aber nur für Return-Parameter Sinn). Standardmäßig ist int beim Erzeugen eines Parameters
voreingestellt und daher der Typ des am häufigsten erzeugten Parameters.

**Standardwert**
Auswahlfeld. Dieser erlaubt es Ihnen, wenn gewünscht, einen Anfangswert für den Parameter einzustellen (dies ist optional). Das Auswahlmenü enthält Zugriff auf die folgenden Werte 0, 1, 2 und null.

![Achtung]

Dies macht nur Sinn für out- oder return-Parameter.

**Art**
Auswahlfeld mit den Einträgen out, in/out, return und in.

- **out**. Der Parameter wird nur dazu verwendet, um Werte von der Operation zurückzugeben.
- **in/out**. Der Parameter wird für beides verwendet, um Werte an die Operation zu übergeben und Ergebnisse von der Operation zurückzugeben.

![Anmerkung]

Dies ist der Standard für jeden neuen Parameter.

- **return**. Der Parameter ist ein Rückgabe-Ergebnis des Aufrufes.

![Anmerkung]

Es gibt nichts, was Sie davon abhält mehr als einen Return-Parameter zu deklarieren (einige Programmiersprachen unterstützen so ein Konzept).

![Tipp]

Der Name des Return-Parameters erscheint nicht im Diagramm, aber es ist bequemer ihm einen entsprechenden Namen zu geben (wie z.B. Standard-return, um ihn in der Liste der Parameter im Register Eigenschaften der Operation identifizieren zu können).

- **in**. Der Parameter wird nur zur Übergabe eines Wertes an die Operation verwendet.

### 18.10. Signal
Ein Signal ist eine Spezifikation eines asynchronen Impulses, der zwischen Instanzen kommuniziert. Im UML-Metamodell ist dies eine Subklasse von Classifier.

In ArgoUML werden Signale nicht vollständig verarbeitet. Ihr Wert besteht darin, wenn sie als Signalereignisse empfangen werden, treiben Sie das asynchrone Verhalten der Zustandsautomaten und
wenn Sie mit *Senden-Aktionen* in Zustandsautomaten und Nachrichten in Kollaborationsdiagrammen verknüpft werden.

**Tipp**


### 18.10.1. Detail-Register Signal

Die aktiven Detail-Register für Signale sind die folgenden.

**Zu-Bearbeiten-Element**

Standard-Register.

**Eigenschaften**


**Dokumentation**

Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

**Quellcode**

Standard-Register. Es wird nichts für ein Signal generiert.

**Eigenschaftswerte**

Standard-Register. Im UML-Metamodell sind für ein Signal die folgenden Standard-Eigenschaftswerte definiert.

- **persistence** (von der Superklasse Classifier). Der Wert *transitory* gibt an, dass sein Zustand zerstört wird, wenn eine Instanz zerstört wird oder *persistent*, die Kennzeichnung des Zustandes bleibt erhalten, wenn eine Instanz zerstört wird.

- **semantics** (von der Superklasse Classifier). Der Wert ist eine Spezifikation der Semantik des Signals.


**Anmerkung**

Abgeleitete Signale haben Ihren Wert in der Analyse, um nützliche Namen oder Konzepte einzubinden und im Design, um eine Wiederverarbeitung zu vermeiden.

**Anmerkung**

18.10.2. Eigenschaftssymbolleiste Signal

Nach oben
Navigiert in der Paketstruktur nach oben.

Neues Signal
Erzeugt ein neues Signal und springt sofort in das Register Eigenschaften dieses Signals.

Achtung
Das Signal ist nicht mit der gleichen Operation wie das Originalsignal verknüpft, so dass dies im Anschluss daran noch getan werden muss.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für das markierte Signal und springt sofort in das Register Eigenschaften dieses Stereotypen.

Löschen
Löschts das Signal aus dem Modell.

Warnung
Dies ist ein Löschen aus dem Modell.

18.10.3. Eigenschaftsfelder für ein Signal

Name
Textbereich. Der Name des Signals. Auf Grund ihrer Ähnlichkeit zu Klassen beginnt der Name eines Signals per Konvention mit einem Großbuchstaben und setzt sich aus Wörtern zusammen die durch Groß-/Kleinschreibung getrennt sind.

Anmerkung
Die ArgoUML-Kritiken beschweren sich nicht über Signalnamen, die nicht mit einem Großbuchstaben beginnen.

Stereotyp
Kombinationsfeld. Ein Signal enthält standardmäßig die UML-Standard-Stereotypen seiner Eltern Classifier im UML-Metamodell (metaclass, powerType, process, thread und utility).

Stereotyp navigieren
Symbol. Wenn ein Stereotyp markiert wurde, navigiert es in das Eigenschaftsfenster des Stereotypen (siehe Abschnitt 16.6, „Stereotyp“).

Namensraum

Kontexte


Abbildung 18.5. Das Dialogfenster „Kontext hinzufügen/entfernen“
18.11. Empfangssignal (noch zu beschreiben)

Ein Empfangssignal ist ...

18.12. Assoziation


Innerhalb des UML-Metamodells ist die Association eine Subklasse von Relationship und GeneralizableElement.

Die Assoziation wird als durchgehende Linie dargestellt, die einen Akteur und einen Anwendungsfall oder eine Klasse oder eine Schnittstelle verbindet (siehe Abbildung 18.1, „Denkbare Modellelemente in einem Klassendiagramm. “). Der Name der Assoziation und die Stereotypen erscheinen oberhalb der Linie.

ArgoUML ist bei binären Assoziationen nicht eingeschränkt. Siehe mehr darüber unter Abschnitt 18.12.1, „Drei-Wege und größere Assoziationen und Assoziationsklassen “.

Assoziationen sind erlaubt zwischen Schnittstellen und Klassen, aber UML 1.3 spezifiziert, dass sie nur in Richtung der Schnittstelle navigierbar sein müssen - mit anderen Worten, die Schnittstelle kann die Klasse nicht sehen. ArgoUML wird solche Assoziationen mit der entsprechenden Navigation zeichnen.

Assoziationen sind oft unbenannt, wenn deren Bedeutung aus dem Kontext klar ersichtlich ist.

Anmerkung

ArgoUML enthält keinen spezifischen Weg, die Richtung der Assoziation darzustellen, wie im UML 1.4-Standard beschrieben. Die Benennung sollte versuchen, dies klarzustellen.

Die Assoziation enthält mindestens zwei Enden, zu denen über das Register Eigenschaften der Assoziation navigiert werden kann. Weitere Informationen siehe Abschnitt 18.13, „Assoziationsende “.

18.12.1. Drei-Wege und größere Assoziationen und Assoziationsklassen


Assoziationsklassen werden genauso gezeichnet wie eine normale Assoziation. z.B. zwischen zwei Klassen, aber mit einem anderen Werkzeug aus der Diagrammsymbolleiste.

18.12.2. Detail-Register Assoziation

Die aktiven Detail-Register für Assoziationen sind die folgenden.
Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung
Standard-Register.

Anmerkung
Die Werte für die Begrenzung der Assoziation haben keine Bedeutung, weil sie durch die Lage der verbundenen Elemente bestimmt werden. Eine Änderung hat keine Auswirkung auf das Diagramm.

Quellcode

Eigenschaftswerte
Standard-Register. Im UML-Metamodell einer Assoziation sind die folgenden Standard-Eigenschaftswerte definiert.

• persistence. Der Wert transitory gibt an, dass der Zustand zerstört wird, wenn die Instanz zerstört wird, oder persistent kennzeichnet, dass der Zustand erhalten bleibt, wenn die Instanz zerstört wird.

• derived (von der Superklasse ModelElement). Der Wert true bedeutet, dass die Assoziation redundant ist - sie kann formal aus anderen Elementen abgeleitet werden, oder false bedeutet, sie kann es nicht.

Anmerkung
Abgeleitete Assoziationen haben ihren Wert in der Analyse, um nützliche Namen oder Konzepte einzubinden und im Design, um eine Wiederverarbeitung zu verhindern.

Anmerkung

18.12.3. Eigenschaftssymbolleiste Assoziation
Nach oben

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Assoziation und springt sofort in das Register Eigenschaften dieses Stereotypen.

Löschen
Löschte die markierte Assoziation aus dem Modell.

Warnung
Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm. Um eine Assoziation aus dem Diagramm zu löschen, sie aber im Modell zu erhalten, verwenden Sie Aus Diagramm entfernen des Hauptmenüs (oder drücken Sie die Taste Entf).

18.12.4. Eigenschaftsfelder für eine Assoziation

Name

Anmerkung
ArgoUML erzwingt keine Namenskonvention für Assoziationen.

Tipp
Obwohl die Designkritiken dies fordern, ist es auf der anderen Seite normal, die Assoziationen in einem Klassendiagramm nicht zu benennen, da die Beziehung der Klasse und Schnittstelle sehr häufig offensichtlich ist.

Stereotyp
Auswahlfeld. Die Assoziation bietet standardmäßig die UML-Standard-Stereotypen für eine Assoziation an (implicit).

Stereotypisieren kann nützlich sein, wenn Sie Assoziationen im Problembereich (Anforderungsanalyse) und dem Lösungsbereich (Analyse) erzeugen, wie auch bei auf Mustern basierenden Prozessen.

Der Stereotyp wird zwischen « and » unterhalb des Assoziationsnamens im Diagramm angezeigt.
Stereotyp navigieren
Symbol. Wenn ein Stereotyp markiert wurde, wird es in das Eigenschaftsfenster des Stereotypen navigieren. (siehe Abschnitt 16.6, „Stereotyp“).

Namensraum
Kombinationsfeld. Gibt den Namensraum für die Assoziation wieder und erlaubt dessen Änderung. Dies ist die Pakethierarchie.

Verbindungen

Die Namen der Assoziationsenden werden aufgelistet, es sei denn das Assoziationsende hat keinen Namen (dies ist der Fall, wenn sie das erste Mal erzeugt wird). In diesen Fall wird (Unbekanntes Assoziationsende) angezeigt.

Anmerkung
Die einzige Darstellung von Assoziationsenden im Diagramm ist, dass deren Name am relevanten Ende der entsprechenden Assoziation erscheint.

Ein Taste 1-Doppelklick auf ein Assoziationsende navigiert zu diesem Ende.

Assoziationsrollen
Textbereich. (Noch zu beschreiben)

Verknüpfungen
Textbereich. (Noch zu beschreiben)

18.13. Assoziationsende
Zwei oder mehr Assoziationsenden sind mit jeder Assoziation verknüpft (siehe Abschnitt 17.5, „Assoziation“).

Innerhalb des UML-Metamodelles ist AssociationEnd eine Subklasse von ModelElement.

Das Assoziationsende hat keinen direkten Zugriff auf die Diagramme für binäre Assoziationen. Die Enden einer N-fach Assoziation können durch Anklicken der Zeile im Diagramm ausgewählt werden. Der Stereotyp, der Name und die Kardinalität werden am relevanten Ende der Eltern-Assoziation angezeigt (siehe Abbildung 17.1, „Typische Modellelemente in einem Anwendungsfalldiagramm.“). Wo eine trennbare oder eine untrennbare Aggregation für eine Assoziationsende ausgewählt wurde, wird das gegenüber liegende Ende als ausgefüllter Diamant (untrennbare Aggregation) oder als nicht ausgefüllter Diamant (trennbare Aggregation) angezeigt.

Tipp
Obwohl sie die Attribute der Assoziationsenden beim Erzeugen eines Anwendungsfallmodells ändern können, ist dies oft nicht notwendig. Viele der Eigenschaften eines Assoziationsendes beziehen sich auf seine Nutzung in Klassendiagrammen und sind von begrenzter Relevanz hinsichtlich der Anwendungsfälle. Die meisten nützlichen Attribute, die geändert werden können sind der Name (als
Rollenname verwendet) und die Kardinalität.

**Anmerkung**

ArgoUML unterstützt aktuell nicht das Anzeigen von Qualifizierern im Diagramm, wie im UML 1.3-Standard beschrieben.

### 18.13.1. Detail-Register Assoziationsenden

Die aktiven Detail-Register für Assoziationen sind die folgenden.

**Zu-Bearbeiten-Element**

Standard-Register.

**Eigenschaften**


**Dokumentation**

Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

**Darstellung**

Standard-Register.

**Quellcode**


**Eigenschaftswerte**

Standard-Register. Im UML-Metamodell sind für AssociationEnd die folgenden Standard-Eigenschaftswerte definiert.

- **derived** (von der Superklasse ModelElement). Der Wert `true` bedeutet, dass die Assoziation redundant ist - sie kann formal von anderen Elementen abgeleitet werden, oder `false`, wenn sie es nicht kann.

**Tipp**

Abgeleitete Assoziationsenden haben Ihren Wert in der Analyse, um nützliche Namen oder Konzepte einzubinden und im Design, um eine Wiederverarbeitung zu verhindern. Der Eigenschaftswert macht bei einem Assoziationsende jedoch nur Sinn, wenn er auch auf die Eltern-Assoziation angewendet wird.

**Anmerkung**

Die UML-Metaklasse Element, von der alle anderen Modellelemente abgeleitet werden, beinhaltet den Eigenschaftswert `documentation`, der in ArgoUML im Register `Dokumentation` bearbeitet wird.
18.13.2. Eigenschaftssymbolleiste Assoziationsende

Nach oben
Navigiert zu der Assoziation, zu der dieses Ende gehört.

Gehe zum anderen Ende
Navigiert zum anderen Ende der Assoziation.

Neuer Qualifizierer
Erzeugt einen neuen Qualifizierer für das markierte Assoziationsende und springt sofort in das Register Eigenschaften dieses Qualifizierers.

Warnung
Qualifizierer werden nur teilweise von ArgoUML V0.18 unterstützt. Aus diesem Grund erzeugt die Aktivierung dieser Schaltfläche einen Qualifizierer im Modell, der im Diagramm nicht angezeigt wird. Auch das Eigenschaftsfenster eines Qualifizierers entspricht dem eines regulären Attributes.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für das markierte Assoziationsende und springt sofort in das Register Eigenschaften dieses Stereotyps.

Löschen
Löscht das markierte Assoziationsende aus dem Modell.

Anmerkung

18.13.3. Eigenschaftsfelder für ein Assoziationsende

Name

Anmerkung
ArgoUML erzwingt keine Namenskonvention für Assoziationsenden.
Stereotyp

Kombinationsfeld. Das Assoziationsende bietet standardmäßig die UML-Standard-Stereotypen für Assoziationsenden (association, global, local, parameter, self) an.

Stereotyp navigieren

Symbol. Wenn ein Stereotyp markiert wurde, navigiert es zum Eigenschaftsfenster des Stereotypen. (siehe Abschnitt 16.6, „Stereotyp“).

Assoziation


Typ

Das Kombinationsfeld bietet Zugriff auf alle Standard-UML-Typen von ArgoUML und alle neuen Klasse an, die innerhalb des aktuelle Modelles erzeugt wurden.

Dies ist der zu diesem Ende der Assoziation hinzugefügte Typ der Entität.

Tipp


Kardinalität


Modifizierer

Es gibt 3 Modifizierer: navigable, ordered und static. Alle 3 sind Markierfelder.

• navigable. Gibt an, dass dieses Ende vom anderen Ende aus erreicht werden kann.

Anmerkung

Der UML 1.4-Standard bietet eine Anzahl Optionen an, wie die Navigation an einem Assoziationsende angezeigt wird. ArgoUML verwendet die Option 3, was bedeutet, dass Pfeile am Ende einer Assoziation angezeigt werden, bei der die Navigation nur an einem Ende aktiviert ist, um die Richtung anzuzeigen, in der die Navigation möglich ist. Das bedeutet, dass der Standard, mit beiden Enden navigierbar, keine Pfeile aufweist.

• ordered. Wenn an einem Ende plaziert, spezifiziert es, ob der Satz von Verknüpfungen von der anderen Instanz zu dieser Instanz eine Reihenfolge aufweist. Die Reihenfolge muss bestimmt und durch Operationen, die Verknüpfungen hinzufügen verwaltet werden. Es stellt zusätzliche Information bereit, die in den Objekten oder Verknüpfungen selbst nicht innenwohnen.

- **Static** (Noch zu beschreiben)

**Spezifikation**

Liste. Benennt Null oder mehr Klassifizierer, welche die Operationen spezifizieren, die auf eine Instanz angewendet werden können, auf die über die Assoziationsenden der Assoziation zugegriffen werden kann. Diese bestimmen die minimale Schnittstelle, die durch den aktuellen Klassifizierer am Ende realisiert werden muss um die Absicht der Assoziation zu unterstützen. Es kann eine Schnittstelle oder ein anderer Klassifizierer sein. Der Typ des Klassifizierers wird als Symbol angezeigt.


- **Hinzufügen**. Fügt einen neuen Spezifikationsklassifizierer hinzu. Dies öffnet das Dialogfenster *Spezifikationen hinzufügen/ entfernen* (siehe Bild unten), das es erlaubt, zwischen allen möglichen Klassifizierern auszuwählen und diese zu der ausgewählten Liste hinzuzufügen oder zu entfernen.

**Abbildung 18.6. Das Dialogfenster „Spezifikationen hinzufügen/ entfernen“**
Qualifizierer

Aggregation
Auswahlfeld mit den drei Einträgen composite, none und aggregate. Gibt an, ob die Beziehung mit dem fernen Ende eine trennbare Ganzes-Teile-Beziehung darstellt (aggregation) oder eine untrennbare Ganzes-Teile-Beziehung (composite).

Eine trennbare Aggregation wird durch einen unausgefüllten Diamanten am „Ganzes“-Ende der Assoziation dargestellt. Eine untrennbare Aggregation wird durch einen ausgefüllten Diamanten dargestellt.
Anmerkung

Sie dürfen nicht an beiden Enden einer Assoziation eine Aggregation haben. ArgoUML erzwingt diese Bedingung nicht.

Das „Ganzes“-Ende einer untrennbaren Aggregation sollte die Kardinalität eins haben. ArgoUML erzwingt diese Bedingung nicht.

Änderbarkeit

Auswahlfeld mit den drei Einträgen add only, changeable und frozen. Gibt an, ob Instanzen dieses Endes des Assoziationsendes sein darf: i) erzeugt, aber nicht gelöscht nachdem die Zielinstanz erzeugt wurde; ii) erzeugt und gelöscht; oder iii) nicht erzeugt oder gelöscht durch die Quelle nadem die Zielinstanz erzeugt wurde.

Sichtbarkeit

Auswahlfeld mit den vier Einträgen public, private, protected und package. gibt an, ob die Navigation zu diesem Ende sein darf: i) jeder Klassifizierer; ii) nur durch die Quellklassifizierer; oder iii) nur durch die Quellklassifizierer und deren Kinder.


Eine Abhängigkeit ist eine Beziehung zwischen zwei Modellelementen, die zeigt, dass das eine vom anderen abhängt.

Innerhalb des UML-Metamodelles ist Dependency eine Subklasse von Relationship.


18.14.1. Detail-Register Abhängigkeit

Die aktiven Detail-Register für Abhängigkeiten sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung
Standard-Register

Anmerkung

Die Werte im Feld "Begrenzung" der Abhängigkeit sind nicht editierbar, weil sie durch die Eigenschaften der Linienendpunkte bestimmt werden.
Eigenschaftswerte

Standard-Register. Im UML-Metamodell hat die Dependency keine Eigenschaftswerte über sich selbst. Aber über die Superklassen sind die folgenden Standard-Eigenschaftswerte definiert.

- **derived** (von der Superklasse ModelElement). Der Wert **true** bedeutet, dass die Abhängigkeitsbeziehung redundant ist - sie kann formal von anderen Elementen abgeleitet werden, oder **false**, sie kann es nicht.

**Anmerkung**

Abgeleitete Abhängigkeiten können ihren Wert in der Analyse haben, um nützliche Namen oder Konzepte einzuführen.

### 18.14.2. Eigenschaftssymbolleiste Abhängigkeit

**Nach oben**

Navigiert durch die Paketstruktur des Modells nach oben. Bei einer Abhängigkeit wird dies das Paket sein, das die Abhängigkeit enthält.

**Neuer Stereotyp**

Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Abhängigkeit und springt sofort in das Register Eigenschaften dieses Stereotyps.

**Löschen**

Löscht die markierte Abhängigkeit aus dem Modell.

**Warnung**

Dies ist ein Löschen aus dem Modell, **nicht** nur aus dem Diagramm. Um eine Abhängigkeit im Diagramm zu löschen, aber im Modell zu erhalten, verwenden Sie **Aus Diagramm entfernen** im Hauptmenü (oder drücken Sie die Taste Entf).

### 18.14.3. Eigenschaftsfelder für eine Abhängigkeit

**Name**

Textfeld. Der Name der Abhängigkeit.

**Tipp**

Es kommt sehr häufig vor, die Abhängigkeiten unbenannt zu lassen.

**Anmerkung**
ArgoUML erzwingt keine Namenskonvention für Abhängigkeiten.

**Anmerkung**

Es gibt keine Darstellung des Namens einer Abhängigkeit im Diagramm.

**Stereotyp**


**Stereotyp navigieren**

Symbol. Wenn ein Stereotyp markiert wurde, wird es in das Eigenschaftsfenster dieses Stereotypen navigieren (siehe Abschnitt 16.6, „Stereotyp“).

**Namensraum**

Textfeld. Nimmt den Namensraum für die Abhängigkeit auf. Dies ist die Pakethierarchie.

**Lieferanten**

Textbereich. Liste das Ende der Beziehung auf, die das liefert, was für das andere Ende erforderlich ist.

Ein Taste 1-Doppelklick auf einen Lieferanten wird zu diesem Element navigieren.

**Clients**

Textbereich. Listet die „abhängigen“ Enden der Beziehung auf, z.B. das Ende, welches das andere verwendet.

Ein Taste 1-Doppelklick auf einen Client wird zu diesem Element navigieren.

### 18.15. Generalisierung

Die Generalisierung ist unter Anwendungsfalldiagramme beschrieben (siehe Abschnitt 17.8, „Generalisierung“).

**Anmerkung**

Innerhalb des Kontextes von Klassen sind die Generalisierung und die Spezialisierung UML-Begriffe, welche die Klassenvererbung beschreiben.

### 18.16. Schnittstelle

Eine Schnittstelle ist ein Satz von Operationen, die das Verhalten eines Elementes beschreiben. Sie kann als abstrakte Klasse ohne Attribute und ohne nicht-abstrakte Operationen angesehen werden. Im UML-
Metamodell ist sie eine Subklasse von Classifier und dadurch von GeneralizableElement.

Eine Schnittstelle wird in einem Klassendiagramm als Rechteck mit zwei horizontalen Bereichen dargestellt. Der oberste Bereich zeigt den Schnittstellenamen an (und darüber «interface ») und im zweiten Bereich jede Operation. Wie bei der Klasse, kann der Operationsbereich versteckt werden.

18.16.1. Detail-Register Schnittstelle

Die aktiven Detail-Register für eine Schnittstelle sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „ Das Register Dokumentation “.

Darstellung

Quellcode
Standard-Register. Enthält ein Template für die Schnittstellendeklaration und der Deklarationen von verbundenen Schnittstellen.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für die Schnittstelle die folgenden Standard-Eigenschaftswerte definiert.

• persistence (von der Superklasse Classifier). Der Wert transitory gibt an, dass der Zustand zerstört wird, wenn die Instanz zerstört wird, oder persistent, wenn der gekennzeichnete Zustand erhalten bleibt, wenn die Instanz zerstört wird.

Warnung
Da Schnittstellen per Definition abstrakt sind, können sie keine Instanz haben. Daher müssen sich diese Eigenschaftswerte sich auf die Eigenschaften der realisierenden Klasse beziehen.

• semantics (von der Superklasse Classifier). Der Wert ist eine Spezifikation der Semantik der Schnittstelle.

• derived (von der Superklasse ModelElement). Der Wert true bedeutet, dass die Schnittstelle redundant ist - sie kann formal von anderen Elementen abgeleitet werden, oder false, wenn sie es nicht kann.

Anmerkung
Abgeleitete Schnittstellen können ihren Wert in der Analyse haben, um nützliche Namen oder Konzepte einzuführen und im Design, um eine Wiederverarbeitung zu verhindern.
Anmerkung

Die UML-Metaklasse Element, von der alle anderen Modellelemente abgeleitet werden, beinhaltet den Eigenschaftswert documentation, der in ArgoUML im Register Dokumentation bearbeitet wird.

Checkliste
Standard-Register für eine Schnittstelle.

18.16.2. Eigenschaftssymbolleiste Schnittstelle

Nach oben
Navigiert in der Paketstruktur nach oben.

Neue Operation
Erzeugt innerhalb der Schnittstelle eine neue Operation (siehe Abschnitt 18.8, „Operation“) und springt sofort in das Register Eigenschaften dieser Operation.

Neues Empfangssignal
Erzeugt ein neues Empfangssignal und springt sofort in das Register Eigenschaften dieses Empfangssignals.

Neue Schnittstelle
Erzeugt im gleichen Namensraum wie die markierte Schnittstelle eine neue Schnittstelle und springt sofort in das Register Eigenschaften dieser neuen Schnittstelle.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Schnittstelle und springt sofort in das Register Eigenschaften dieses Stereotypes.

Löschen
Löscht die Schnittstelle aus dem Modell.

Warnung
Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm. Um eine Schnittstelle aus dem Diagramm zu löschen, aber es im Modell zu behalten, verwenden Sie Aus Diagramm entfernen im Hauptmenü (oder drücken Sie die Taste Entf).

18.16.3. Eigenschaftsfelder einer Schnittstelle

Name
Textfeld. Der Name der Schnittstelle. Der Name der Schnittstelle beginnt mit einem Großbuchstaben und besteht aus Wörtern, die durch Groß-/Kleinschreibung voneinander getrennt sind.

Anmerkung
Wie bei Klassen werden sich die ArgoUML-Kritiken nicht beschweren, wenn die Schnittstellennamen nicht mit einem Großbuchstaben beginnen.

Stereotyp
Kombinationsfeld. Die Schnittstelle bietet standardmäßig die UML-Standard-Stereotypen für die Eltern-Metaklasse Classifier (metaclass, powertype, process, thread und utility) an.

Stereotyp navigieren
Symbol. Wenn ein Stereotyp markiert wird, navigiert es in das Eigenschaftsfenster Stereotyp (siehe Abschnitt 16.6., „Stereotyp“).

Namensraum
Kombinationsfeld. Nimmt den Namensraum für die Schnittstelle auf und erlaubt deren Änderung. Dies ist die Pakethierarchie.

Modifizierer
Markierfelder mit den Einträgen Abstract, Leaf und Root.

• Abstract wird verwendet, um zu deklarieren, dass diese Schnittstelle nicht instanziiert werden kann, aber immer spezialisiert werden muss. Der Name einer abstrakten Schnittstelle wird in kursiver Schrift im Diagramm dargestellt.

Achtung
Dies ist bedeutungslos, weil die Schnittstelle per Definition eine abstrakte Entität ist. Der UML 1.3-Standard bietet hier keine Erläuterung an.

• Leaf gibt an, dass diese Schnittstelle nicht weiter spezialisiert werden kann, während Root angibt, dass sie keine Generalisierungen haben kann.

Sichtbarkeit

Generalisierungen
Textbereich. Listet jede Schnittstelle auf, die diese Schnittstelle generalisiert.

Ein Taste 1-Doppelklick navigiert zu der Generalisierung und öffnet deren Register Eigenschaften.

Spezialisierungen
Textfeld. Listet alle spezialisierten Schnittstellen auf (z.B. für die diese Schnittstelle eine Generalisierung ist).

Ein Taste 1-Doppelklick navigiert zu der Spezialisierung und öffnet deren Register Eigenschaften.

Assoziationsenden
Textfeld. Listet alle Assoziationsenden auf (siehe Abschnitt 18.13, „Assoziationsende“), die mit
Anmerkung


Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

Operationen


Achtung

Alle Operationen in einer Schnittstelle müssen public sein. Die ArgoUML-Kritiken werden sich darüber beschweren, wenn dies nicht der Fall ist.

18.17. Abstraktion

Eine Abstraktion ist eine Abhängigkeitsbeziehung, die zwei Modellelemente innerhalb des Modelles auf unterschiedlichen Abstraktionsebenen miteinander verbindet. Innerhalb von ArgoUML wird dies prinzipiell über seine spezifischen Stereotyp realize, um Realisierungsabhängigkeiten zu definieren, die Modellelemente die Verhalten beschreiben mit den Modellelementen die Verhalten implementieren zu verknüpfen.

Im UML-Metamodell ist die Abstraction eine Subklasse von Dependency und darüber von Relationship.

Eine Abstraktion mit dem Stereotyp realize wird in einem Klassendiagramm als gepunktete Linie mit einem weissen Kopf am spezifizierenden Ende dargestellt.

Achtung

Alle anderen Stereotypen einer Abstraktion sollten mit einem offenen Pfeil dargestellt werden, aber dies wird durch ArgoUML nicht unterstützt.

18.17.1. Detail-Register Abstraktion

Die aktiven Detail-Register für eine Abstraktion sind die folgenden.

Zu-Bearbeiten-Element

Standard-Register.

Eigenschaften

Dokumentation
Standard-Register. Siehe Abschnitt 13.4, „Das Register Dokumentation“.

Darstellung
Standard-Register.

Anmerkung
Der Werte im Feld "Begrenzung der Abstraktion sind nicht editierbar, weil sie durch
die Eigenschaften der Linienendpunkte bestimmt werden.

Quellcode
Standard-Register. Enthält den deaktivierten Text N/A.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für Abstraktion die folgenden Standard-
Eigenschaftswerte definiert.

• derived (von der Superklasse ModelElement). Der Wert true bedeutet, dass die
Abstraktion redundant ist - sie kann formal von anderen Elementen abgeleitet werden, oder
false, wenn sie es nicht kann.

Anmerkung
Abgeleitete Abstraktionen haben ihren Wert in der Analyse, um nützliche Namen
oder Konzepte einzuführen und im Design um eine Wiederverarbeitung zu
verhindern.

Anmerkung
Die UML-Metaklasse Element, von der alle anderen Modellelemente abgeleitet
werden, beinhaltet den Eigenschaftswert documentation, der in ArgoUML im
Register Dokumentation bearbeitet wird.

18.17.2. Eigenschaftssymbolleiste Abstraktion

Nach oben
Navigiert in der Paketstruktur nach oben.

Löschen
Löscht die Abstraktion aus dem Modell.

Warnung
Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm. Um eine
Abstraktion aus dem Diagramm zu entfernen, es aber im Modell zu erhalten,
verwenden Sie Aus Diagramm entfernen im Hauptmenü (oder drücken Sie die
Taste Entf).
18.17.3. Eigenschaftsfelder für eine Abstraktion

Name
Textfeld. Der Name der Abstraktion. Es gibt keine Randbedingungen für die Namen einer Abstraktion, der im Diagramm nicht angezeigt wird.

Stereotyp
Kombinationsfeld. Die Abstraktion bietet standardmäßig die UML-Standard-Stereotypen derive, realize, refine und trace an.

Achtung

Stereotyp navigieren
Symbol. Wenn ein Stereotyp markiert wurde, navigiert es in das Eigenschaftsfenster des Stereotyps (siehe Abschnitt 16.6, „Stereotyp“).

Namensraum

Lieferanten
Textbereich. Listet die Modellelemente auf, die das Lieferantenende dieser Abstraktion darstellen (bei einer Realisierung ist dies das Ende, das die Implementierung enthält).

Anmerkung
Obwohl dies ein Textbereich ist, gibt es keinen Mechanismus, mehr als einen Lieferanten hinzuzufügen.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

Clients
Textbereich. Listet die Modellelemente auf, die das Client-Ende dieser Abstraktion darstellen (bei einer Realisierung ist dies das Ende, das die Spezifikation enthält).

Anmerkung
Obwohl dies ein Textbereich ist, gibt es keinen Mechanismus mehr als einen Client hinzuzufügen.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.
Kapitel 19. Modellelement-Referenz
Sequenzdiagramm

19.1. Einleitung


Es gibt eine enge Beziehung zwischen diesem Material und dem Register Eigenschaften des Detailfensters (siehe Abschnitt 13.3, „Das Register Eigenschaften“). Dieser Abschnitt behandelt die Eigenschaften allgemein, in diesem Kapitel sind sie mit spezifischen Modellelementen verknüpft.

Achtung

Sequenzdiagramme sind aktuell nicht vollständig in ArgoUML implementiert. Viele Aspekte sind nicht vollständig, oder verhalten sich nicht wie erwartet.

Abbildung 19.1, „Denkbare Modellelemente in einem Sequenzdiagramm.“ zeigt ein Sequenzdiagramm mit allen denkbaren Modellelementen.

19.1.1. Einschränkungen, die Sequenzdiagramme in ArgoUML betreffen

Das Sequenzdiagramm ist in ArgoUML noch recht unterentwickelt.

Die größten Unterschiede befinden sich in den Aktionen hinter den Impulsen. Diese sind vollständig textuell implementiert und gibt es keinen Weg, diese mit deren verknüpften Operationen oder Signalen zurückzuverfolgen.

19.2. Objekt


**Achtung**

Die aktuelle Release von ArgoUML zeigt Interaktionen zwischen Objekten, obwohl der UML-Standard für Sequenzdiagramme für Interaktionen zwischen Instanzen eines Klassifiziers gedacht ist.


19.2.1. Detail-Register Objekt

Die aktiven Detail-Register für Objekte sind die folgenden.

**Zu-Bearbeiten-Element**  
Standard-Register.

**Eigenschaften**


**Dokumentation**  
Standard-Register.

**Darstellung**

Standard-Register. Die Werte für die Begrenzung des Objektes definieren theoretisch den begrenzenden Rahmen des Objektes und seine Zeitlinie. Wenn Sie diese jedoch ändern, hat dies keinen Effekt und die Originalwerte werden wieder eingestellt, wenn Sie das Register erneut aufsuchen.

**Quellcode**  
Standard-Register, aber ohne Inhalte.
Achtung

Ein Objekt sollte keinen Code generieren, so dass die Tatsache, dass dieses Register aktiv ist, wahrscheinlich einen Fehler darstellt.

Eigenschaftswerte

Standard-Register. Im UML-Metamodell sind für Object die folgenden Eigenschaftswerte definiert.

- persistence (von der Superklasse Instance). Zeigt die Dauerhaftigkeit der mit dem Objekt verknüpften Zustandsinformationen an. Werte sind transitory (der Zustand wird zerstört, wenn das Objekt zerstört wird) und persistent (der Zustand bleibt erhalten, wenn das Objekt zerstört wird).


Anmerkung

Abgeleitete Objekte haben ihren Wert in der Analyse und im Design, um nützliche Namen oder Konzepte einzuführen und im Design, um eine Wiederverarbeitung zu verhindern.

Anmerkung

Die UML-Metaklasse Element, von der alle anderen Modellelemente abgeleitet werden, beinhaltet den Eigenschaftswert documentation, der in ArgoUML im Register Dokumentation bearbeitet wird.

Checkliste

Standard-Register für einen Klassifizierer.

19.2.2. Eigenschaftssymbolleiste Objekt

Nach oben

Navigiert in der Paketstruktur nach oben.

Neuer Stereotyp

Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für das markierte Objekt und springt sofort in das Register Eigenschaften dieses Stereotypes.

 Löschen

Löscht das Objekt aus dem Modell.
Warnung

Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm. Um ein Objekt aus dem Diagramm zu löschen, es aber im Modell zu erhalten, verwenden Sie Aus Diagramm entfernen im Hauptmenü (oder drücken Sie die Taste Entf).

19.2.3. Eigenschaftsfelder für ein Objekt

Name

Textfeld. Der Name des Objekts. Per Konvention beginnt der Objektname mit einem Kleinbuchstaben und verwendet die Groß-/Kleinschreibung, um Wörter innerhalb des Namens voneinander zu trennen.

Anmerkung

ArgoUML erzwingt keine Namenskonvention.

Stereotyp

Kombinationsfeld. Das Objekt hat im UML-Standard standardmäßig keine Stereotypen.

Stereotyp navigieren

Symbol. Wenn ein Stereotyp markiert wurde, wird es in das Eigenschaftsfenster dieses Stereotyps navigieren (siehe Abschnitt 18.5, „Stereotyp“).

Namensraum


Impulse gesendet

Textbereich. Listet alle Impulse auf, die zu diesem Objekt gesendet wurden.

Impulse empfangen

Textbereich. Listet die von diesem Objekt empfangenen Impulse auf.

Klassifizierer

Kombinationsfeld. Der Name des Klassifizierers von dem dies ein Objekt ist.

Achtung

In der aktuellen Release von ArgouML enthält das Kombinationsfeld alle Klassifizierer (z.B. Schnittstellen, Akteure, Anwendungsfälle und Datentypen), was das ist, was im Diagramm gewollt ist, obwohl es richtigerweise als eine Instanz anstelle von Objekt bezeichnet werden sollte. In der Praxis machen nur Instanzen von Klassen und Akteuren einen Sinn.

Anmerkung

In der aktuellen Release von ArgoUML wird die gleiche grafische Darstellung verwendet, auch wenn das Objekt aktuell die Instanz eines Akteurs darstellt (auch
19.3. Impuls


Die Verbindung (siehe Abschnitt 19.9, „Verknüpfung“) mit einem Impuls wird in einem Sequenzdiagramm von ArgoUML als Pfeil zwischen den Zeitlinien der Objektinstanzen (oder dem Objektkopf im Fall des Erzeugens eines Impulses; unten beschrieben) dargestellt, mit dem Namen der Aktion bezeichnet (sofern gewünscht) und der, durch ein Doppelpunkt (:) getrennte Aktion. Der Linietyp und die Pfeile hängen vom Typ der Aktion ab, die den Impuls generieren:


- **Impuls Rückgabe**. Von einer Objektinstanz generiert, die vorher einen Impulsaufruf empfangen hat und das Ergebnis an die aufrufende Objektinstanz zurückliefert. Dargestellt als gepunktete Linie mit einem offenen Pfeil.

**Anmerkung**

ArgoUML erlaubt es Ihnen nicht, Impulse direkt zu erzeugen. Es bietet aber Werkzeuge an, um alle oben genannten fünf Impulstypen als Impuls zu erzeugen.

**Achtung**

19.3.1. Detail-Register Impuls

Die aktiven Detail-Register für Impulse sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Darstellung
Standard-Register. Die Werte für die Begrenzung des Impulses ist theoretisch durch die Grenzen des Impulsrahmens und seiner Zeitlinie definiert. Wenn Sie diese jedoch ändern, wird dies keinen Effekt zeigen und der Ursprungswert wird wieder eingestellt, wenn Sie das Register wieder aufsuchen.

Das Ändern der Einträge Füllfarbe und Schatten hat keinen Effekt. Sie können den Eintrag Linienfarbe verändern und es wird eine Linie um das Signal gezeichnet, was keine UML-Darstellung ist und eher bizarr ist.

**Tipp**
Um die Farbe der Linie zu ändern, sollten Sie die verknüpfte Verbindung markieren (etwas abseits des Impulses klicken) und dessen Register Darstellung verwenden (siehe Abschnitt 19.9, „ Verknüpfung “).

**Achtung**
In der aktuellen Release von ArgoUML ist das ändern des Feldes Begrenzung möglich, wird aber die Position des Impulses nur temporär ändern. Das Markieren irgend eines Modellelementes auf dem Bildschirm veranlasst den Impuls wird zu seiner Ursprungsposition zurückzukehren und die Ursprungswerte werden zurückgeholt.

Quellcode
Standard-Register, aber ohne Inhalte.

**Achtung**
Ein Impuls sollte keinen Code generieren, so dass die Tatsache, dass dieses Register aktiv ist, wahrscheinlich einen Fehler darstellt.

Randbedingungen
Standard-Register. ArgoUML unterstützt nur Randbedingungen bei Klassen und Features (Attribute, Operationen, Signalen und Methoden), so dass dieses Register deaktiviert ist.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für Stimulus die folgenden Standard-Eigenschaftswerte definiert.

- **derived** (von der Superklasse ModelElement). Der Wert true bedeutet, dass der Impuls redundant ist - er kann formal von anderen Elementen abgeleitet werden, oder false, wenn er es nicht kann.

**Anmerkung**


**Anmerkung**

Die UML-Metaklasse Element, von der alle anderen Modellelemente abgeleitet werden, beinhaltet den Eigenschaftswert documentation, der in ArgoUML im Register Dokumentation bearbeitet wird.

### 19.3.2. Eigenschaftssymbolleiste Impuls

- **Nach oben**
  Navigiert in der Paketstruktur nach oben.

- **Löschen**
  Löscht den Impuls aus dem Modell.

**Warnung**

Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm. Um einen Impuls aus dem Diagramm zu löschen, ihn aber im Modell zu erhalten, verwenden Sie Aus Diagramm entfernen im Hauptmenü (oder drücken Sie die Taste Entf).

### 19.3.3. Eigenschaftsfelder für einen Impuls

- **Name**
  Textfeld. Es gibt keine Namenskonvention für die Benennung von Impulsen und es ist normal, diesen unbenannt zu lassen. Die Aktion ist eine ausreichende Identifikation.

**Tipp**

Es ist manchmal nützlich, Impulsen einfache Namen zu geben, so dass sie in den hinzugefügten Notizen, welche die zeitlichen Randbedingungen beschreiben, referenziert werden können.
Aktion
Textfeld. Wird verwendet, um die Aktion zu identifizieren, die den Impuls generiert hat.

Achtung
Die aktuelle Release von ArgoUML implementiert Aktionen nur als textuelle Beschreibungen.

Stereotyp
Kombinationsfeld. Ein Impuls hat im UML-Standard standardmäßig keine Stereotypen, aber ArgoUML bietet die Stereotypen machine, organization und person an.

Achtung
ArgoUML bietet auch den Stereotyp realize für einen Impuls an. Dies scheint ein Fehler zu sein, weil dieser Stereotyp richtigerweise zur Metaklasse Abstraction gehört.

Stereotyp navigieren
Symbol. Wenn ein Stereotyp markiert wurde, wird es in das Eigenschaftsfenster des Stereotyps navigieren (siehe Abschnitt 18.5, „Stereotyp“).

Sender
Textfeld. Identifiziert die Instanz, die diesen Impuls sendete.
Ein Taste 1-Klick navigiert zur Senderinstanz, Taste 2 öffnet ein Popup-Menü mit einem Eintrag.
• Öffnen. Navigiert zur markierten Senderinstanz.

Empfänger
Textfeld. Identifiziert die Instanz, die den Impuls empfängt.
• Öffnen. Navigiert zu der markierten Empfängerinstanz.

Warnung
In der aktuellen Release von ArgoUML ist dieses Feld ohne Funktion. Es zeigt immer den Eintrag none und das Popup-Menü ist deaktiviert.

Namensraum

Ein Taste 1-Klick auf diesen Eintrag navigiert zu dem Paket, das diesen Namensraum definiert (oder zum Modell als Namensraum auf oberster Ebene).

19.4. Impuls Aufrufen

Diese Werkzeug erzeugt einen, mit einer Aufrufaktion verknüpften Impuls im Diagramm und erzeugt zur gleichen Zeit die verknüpfte Verbindung zwischen der Sender- und Empfängerinstanz.


Anmerkung
Da die aktuelle Release von ArgoUML Aktionen nicht vollständig implementiert hat, gibt es keine Durchgängigkeit zwischen der Beziehung und der Aufrufaktion.

19.5. Impuls Erzeugen

Dieses Werkzeug erzeugt einen, mit einer Erzeugen-Aktion verknüpften Impuls im Diagramm und erzeugt zur gleichen Zeit die verknüpfte Verbindung zwischen der Sender- und Empfänger-Instanz.


Anmerkung

19.6. Impuls Zerstören

Dieses Werkzeug erzeugt einen, mit einer Zerstören-Aktion verknüpften Impuls im Diagramm und erzeugt zur gleichen Zeit die verknüpfte Verbindung zwischen der Sender- und Empfänger-Instanz.


Anmerkung
19.7. Impuls Senden

Dieses Werkzeug erzeugt einen, mit einer Senden-Aktion verknüpften Impuls im Diagramm und erzeugt zur gleichen Zeit die verknüpfte Verbindung zwischen der Sender- und Empfänger-Instanz.


Anmerkung


19.8. Impuls Rückgabe

Dieses Werkzeug erzeugt einen, mit einer Rückgabe-Aktion verknüpften Impuls im Diagramm und erzeugt zur gleichen Zeit die verknüpfte Verbindung zwischen der Sender- und Empfänger-Instanz.


Anmerkung


19.9. Verknüpfung

Eine Verknüpfung ist eine Instanz einer Assoziation. Im UML-Metamodell ist Link eine Subklasse von Instance. In Sequenzdiagrammen werden Verknüpfungen indirekt erzeugt, wenn ein verknüpfter Impuls erzeugt wird.

Eine Verknüpfung wird in einem Sequenzdiagramm in ArgoUML als eine Linie dargestellt, welche die betreffenden Instanzen miteinander verbindet. In einem Sequenzdiagramm ist die Darstellung jedoch modifiziert, um den Aktionstyp, der mit dem, auf die Verknüpfung wirkenden Impuls verknüpft ist, wiederzugeben.

19.9.1. Detail-Register Verknüpfung

Die aktiven Detail-Register für Verknüpfungen sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Darstellung
Standard-Register.

Anmerkung
Die Werte im Feld "Begrenzung" der Verknüpfung sind nicht editierbar, weil sie durch die Eigenschaften der Endpunkte der Linie bestimmt werden.

Quellcode
Standard-Register, aber ohne Inhalte.

Achtung
Eine Verknüpfung sollte keinen Code generieren, so dass die Tatsache, dass dieses Register aktiv ist, ein Fehler zu sein scheint.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für Link die folgenden Standard-Eigenschaftswerte definiert.

• persistence (von der Superklasse Instance). Zeigt die Dauerhaftigkeit der mit der Verknüpfung verbundenen Zustandsinformation an. Werte sind transitory (der Zustand wird zerstört, wenn die Verknüpfung zerstört wird) und persistent (der Zustand bleibt erhalten, wenn die Verknüpfung zerstört wird).

• derived (von der Superklasse ModelElement). Der Wert true bedeutet, dass die Verknüpfung redundant ist - sie kann formal von anderen Elementen abgeleitet werden, oder false, wenn sie es nicht kann.

Anmerkung

Anmerkung

Checkliste
Standard-Register für einen Klassifizierer.

19.9.2. Eigenschaftssymbolleiste Verknüpfung
Nach oben
Navigiert in der Paketstruktur nach oben.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Verknüpfung und springt sofort in das Register Eigenschaften dieses Stereotypes.

Löschen
 Löscht die Verknüpfung aus dem Modell.

Warnung
Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm. Um eine Verknüpfung aus dem Diagramm zu löschen, aber im Modell zu erhalten, verwenden Sie Aus Diagramm entfernen im Hauptmenü (oder drücken Sie die Taste Entf).

19.9.3. Eigenschaftsfelder für die Verknüpfung

Name
Textfeld. Der Name der Verknüpfung. Per Konvention beginnen Verknüpfungsnamen mit einem Kleinbuchstaben und verwenden die Groß-/Kleinschreibung, um Wörter im Namen voneinander zu trennen.

Anmerkung
ArgoUML erzwingt keine Namenskonvention.

Stereotyp
Kombinationsfeld. Eine Verknüpfung hat im UML-Standard standardmäßig keine Stereotypen.

Stereotyp navigieren
Sybmol. Wenn ein Stereotyp markiert wurde, navigiert es in das Eigenschaftsfenster des Stereotypen (siehe Abschnitt 18.5, „Stereotyp“).

Namensraum

Verbindungen
Listenfeld. Listet die Verbindungen einer Verknüpfung auf, z.B. die Verknüpfungsenden.

Ein Taste 1-Doppelklick auf den Eintrag navigiert zu dem Verknüpfungsende.
Kapitel 20. Modellelement-Referenz
Zustandsdiagramm

20.1. Einleitung

Dieses Kapitel beschreibt jedes Modellelement, das innerhalb eines Zustandsdiagrammes erzeugt werden kann. Beachten Sie, dass einige Submodellelemente von Modellelementen des Diagrammes aktuell nicht selbst im Diagramm erscheinen können.

Es gibt eine enge Beziehung zwischen diesem Material und dem Register Eigenschaften im Eigenschaftsfenster (siehe Abschnitt 13.3, „Das Register Eigenschaften“). Dieser Abschnitt behandelt die Eigenschaften im Allgemeinen, in diesem Kapitel werden sie mit spezifischen Modellelementen verknüpft.


20.1.1. Einschränkungen, die Zustandsdiagramme in ArgoUML betreffend


Die Codegenerierung von Zustandsdiagrammen ist zur Zeit nicht implementiert.

20.2. Zustand

Ein Zustand modelliert eine Situation, in denen einige (in der Regel implizite) invariante Bedingung für die Elternklasse gelten. Diese Invarianz, kann eine statische Situation sein, wie ein Objekt, dass auf einige extern auftretende Ereignisse wartet, oder einige „laufende“ dynamische Aktivitäten.


20.2.1. Detail-Register Zustand

Die aktiven Detail-Register für Zustände sind die folgenden.

Zu-Bearbeiten-Element
   Standard-Register.

Eigenschaften

Dokumentation
   Standard-Register.
Darstellung

Stereotyp
Standard-Register.

Eigenschaftswerte
Standard-Register.

20.2.2. Eigenschaftssymbolleiste Zustand

Nach oben
Navigiert in der Paketstruktur nach oben.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für den markierten Zustand und springt sofort in das Register Eigenschaften dieses Stereotyps.

Löschen
Löscht den Zustand aus dem Modell.

Anmerkung

20.2.3. Eigenschaftsfelder für einen Zustand

Name
Textfeld. Der Name des Zustandes. Per Konvention beginnen die Zustandsnamen mit einem Kleinbuchstaben und verwenden die Groß-/Kleinschreibung, um Wörter innerhalb des Namens voneinander zu trennen.

Anmerkung
ArgoUML erzwingt diese Namenskonvention nicht.

Container

Eintritt-Aktion
Textfeld. Zeigt den Namen der Aktion (falls vorhanden), die beim Eintritt in diesen Zustand ausgeführt wird.

Anmerkung
Dieses Feld zeigt den Namen der Aktion, während im Diagramm der Ausdruck der Aktion dargestellt wird.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag, Taste 2 öffnet ein Popup-Menü mit zwei Einträgen:
• Aus Modell entfernen. Löscht die Eintritt-Aktion.

Austritt-Aktion
Textfeld. Zeigt die Aktion (falls vorhanden) die beim Verlassen des Zustandes ausgeführt wird.

Ein Taste 1-Klick navigiert zu der markierten Aktion, Taste 2 öffnet ein Popup-Menü mit zwei Einträgen.
• Aus Modell entfernen. Löscht die Austritt-Aktion.

Ausführen-Aktion
Textfeld. Zeigt die Aktion (falls vorhanden), die ausgeführt wird, solange Sie sich in dem Zustand befinden.

Ein Taste 1-Klick navigiert zu der markierten Aktion, Taste 2 öffnet ein Popup-Menü mit zwei Einträgen.
• Aus Modell entfernen. Löscht die Ausführen-Aktion.

Verzögerbare Ereignisse
Textfeld. Zeigt eine Liste von Ereignissen, die Kandidaten dafür sind, dass sie durch den Zustandsautomaten zurückgehalten werden, wenn sie keine Transitionen aus dem Zustand auslösen (nicht verbraucht).

Ein Taste 1-Klick navigiert zu dem markierten Ereignis, Taste 2 auf ein Ereignis öffnet ein Popup-Menü mit den folgenden Einträgen:
• Auswählen. Erlaubt das hinzufügen bereits existierender Ereignisse zur Liste der aufgeschobenen Ereignisse.
• Aus Modell entfernen. Löscht das Ereignis.

Ankommend
Textbereich. Listet alle Transitionen auf, die in diesen Zustand eintreten.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eingang.

Abgehend
Textbereich. Listet alle Transitionen auf, die diesen Zustand verlassen.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eingang.

Interne Transitionen

Anmerkung
Dieses Feld zeigt den Namen der Transition, während im Diagramm der Name des Auslösers, mit einem / getrennt vom Auswirkungsskript angezeigt wird.


• Neu. Fügt eine neue interne Transition hinzu.

20.3. Aktion


Es gibt eine Anzahl unterschiedlicher Aktionstypen, die im UML-Metamodell Kinder von Action sind.

• CreateAction. Verknüpft mit einem Klassifizierer erzeugt diese Aktion eine Instanz des Klassifizierers.
• CallAction. Verknüpft mit einer Operation ruft diese Aktion die angegebene Operation auf.
• ReturnAction. Eine Aktion, die dazu verwendet wird, das Ergebnis eines früheren Aufrufes zurückzugeben.
• SendAction. Verknüpft mit einem Signal löst diese Aktion das Signal aus.
• TerminateAction. Veranlasst das aufgerufene Objekt sich selbst zu beenden.
• UninterpretedAction. Eine Aktion, die dazu verwendet wird, sprachspezifische Aktionen zu spezifizieren, die nicht anderen Aktionstypen zugeordnet werden können.
• DestroyAction. Zerstört das angegebene Zielobjekt.
Eine Aktion wird im Diagramm durch den Text eines Ausdrucks dargestellt.

**Achtung**

Die Release V0.20 von ArgoUML implementiert nur teilweise Aktionen. Als praktikable Konvention wird angenommen, dass Aufruf-Aktionen als Name der Operation dargestellt werden, die die Aktion mit in Klammer gesetzten Argumenten generiert und Sendeaaktionen werden als Name des Signals dargestellt, die die Aktion mit in Klammer gesetzten Argumenten generiert. Rückgabe-Aktionen sollten als Ausdruck des Wertes den sie zurückgeben dargestellt werden, oder im anderen Fall leer sein. Erzeugen- und Zerstören-Aktionen sollten als `create(<target>)` und `destroy(<target>)`. Beenden-Aktionen sollten als `terminate` dargestellt werden.

### 20.3.1. Detail-Register Aktion

Die aktiven Detail-Register für Aktionen sind die folgenden.

- **Zu-Bearbeiten-Element**
  Standard-Register.

- **Eigenschaften**

- **Dokumentation**
  Standard-Register.

- **Stereotyp**
  Standard-Register. Im UML-Metamodell sind für `Action` keine Standard-Stereotypen definiert.

- **Eigenschaftswerte**
  Standard-Register. Im UML-Metamodell sind für `Action` keine Standard-Eigenschaftswerte definiert.

### 20.3.2. Eigenschaftssymbolleiste Aktion

- **Nach oben**
  Navigiert in der hierarchischen Struktur nach oben.

- **Neuer Stereotyp**
  Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Aktion und springt sofort in das Register Eigenschaften dieses Stereotypes.

- **Löschen**
  Löscht die Aktion aus dem Modell.

### 20.3.3. Eigenschaftsfelder für eine Aktion
Name
Textfeld. Der Name der Aktion. Per Konvention beginnt der Aktionsname mit einer Kleinbuchstaben und nutzt die Groß-/Kleinschreibung um Wörter innerhalb des Namens voneinander zu trennen.

Anmerkung
ArgoUML erzwingt diese Namenskonvention nicht.

Asynchron
Markierfeld. Gibt an, ob ein verteilter Impuls asynchron ist oder nicht.

Skript
Doppeltes Textfeld mit einem Ausdruck, der die Aktion definiert. Dieses Feld besteht aus zwei Teilen. Der erste enthält den Rumpf (Skript) des Ausdrucks und der zweite enthält die speziell verwendete Programmmiersprache, mit der dieser Ausdruck geschrieben wurde.

Rekursion

Argumente
Textfeld. Dies ist eine geordnete Liste mit Argumenten der Aktion.

Ein Taste 1-Doppellklick auf einen der Argumente navigiert zu diesem Argument, ein Taste 2-Klick öffnet ein Popup-Menü mit zwei Einträgen.

• Neu. Erzeugt ein neues Argument und navigiert dort hin.
• Entfernen. Löscht das Argument aus dem Modell.

Instanzierung (nur für CreateAction)
Textfeld. Dies zeigt den Klassifizierer, der durch die Erzeugen-Aktion instanziiert wird.

Ein Taste 1-Doppellklick auf den Klassifizierer navigiert zu diesem Argument, ein Taste 2-Klick öffnet ein Popup-Menü mit einem Eintrag.

• Hinzufügen... Öffnet ein Dialogfenster, dass die Auswahl des einen Klassifizierers erlaubt, der erzeugt wird.

20.4. Zusammengesetzter Zustand

Ein zusammengesetzter Zustand ist ein Zustand, der andere Zustände enthält (bekannt als Sub-Zustände) und erlaubt das Konstruieren hierarchischer Zustandsautomaten.


Sub-Zustände werden innerhalb eines zusammengesetzten Automaten plaziert, indem Sie in dem zusammengesetzten Zustand vollständig plaziert werden. Das kann zum Zeitpunkt des Erzeugens getan werden.
werden, z.B., wenn der Zustand das erste Mal im Editierfenster erzeugt wird. Alternativ kann ein existierender Zustand auf den zusammengesetzten Zustand gezogen werden.

die Beschreibung eines zusammengesetzten Zustandes ist immer identisch mit dem des Zustandes (siehe Abschnitt 20.2, „Zustand“ und wurde daher hier nicht dupliziert. Der einzige Unterschied ist ein zusätzliches Werkzeug, ein fehlendes Feld und ein zusätzliches Feld, die im folgenden beschrieben werden.

**Neue nebenläufige Region**

Fügt eine neue nebenläufige Region dem markierten zusammengesetzten Zustand hinzu.

**Verzögerte Ereignisse**

Dieses Feld fehlt in V0.20 von ArgoUML.

**Unterpunkte**

Textbereich. Listet alle, in diesem zusammengesetzten Zustand enthaltenen Sub-Zustände auf.


**Warnung**


- **Aus Modell entfernen.** Löscht den markierten Zustand aus dem Modell.

### 20.5. Nebenläufige Region

Eine nebenläufige Region ist eine „rechtwinklige verbindende“ Komponente eines zusammengesetzten Zustandes, die es erlaubt, Nebenläufigkeit zu konstruieren.

Eine nebenläufige Region wird im Diagramm durch das Teilen eines zusammengesetzten Zustandes dargestellt, bei dem andere Regionen durch eine gestrichelte Linie voneinander getrennt sind.

ArgoUML unterstützt aktuell nur eine horizontale Teilung eines nebenläufigen zusammengesetzten Zustandes in Regionen.

Die Beschreibung der Detailfenster einer nebenläufigen Region ist identisch mit der eines zusammengesetzten Zustandes (siehe Abschnitt 20.4, „Zusammengesetzter Zustand“ und wird daher jetzt nicht wiederholt.

### 20.6. Subautomaten Zustand

Der Subautomaten-Zustand wird dargestellt wie ein normaler Zustand mit einer zusätzlichen include-Deklaration (und durch eine Linie getrennt) oberhalb seines internen Transitionsbereiches. Der Ausdruck, der dem reservierten Wort include folgt, ist der Name des aufgerufenen Subautomaten.

ArgoUML unterstützt aktuell nur die horizontale Teilung eines nebenläufigen zusammengesetzten Zustandes in Regionen.

Die Beschreibung der Detailfenster einer nebenläufigen Region ist immer identisch mit einem zusammengesetzten Zustand (siehe Abschnitt 20.4, „Zusammengesetzter Zustand“ und wird daher hier nicht wiederholt. Der einzige Unterschied ist ein zusätzliches Feld:

Subautomat
   Kombinationsfeld. Erlaubt das Auswählen des Subautomaten, der in diesen zusammengesetzten Zustand eingefügt wird.

20.7. Flacher Zustand

Ein flacher Zustand erscheint nur in einem Subautomatenzustand.


Jeder flache Zustand hat im Diagramm eine Bezeichnung, die mit dem Pfadnamen korrespondiert, der durch das Attribut „Referenzzustand“ des flachen Zustandes repräsentiert wird.

Die Beschreibung der Detailfenster eines flachen Zustandes ist immer mit der eines Pseudozustandes identisch (siehe Abschnitt 20.11, „Pseudozustand“ und wird daher hier nicht wiederholt. Der einzige Unterschied ist ein zusätzliches Feld:

Referenzzustand
   Kombinationsfeld. Erlaubt die Eingabe eines Pfadnamens auf den Referenzzustand.

20.8. Transition


Eine Transition wird in einem Zustandsdiagramm in ArgoUML als Linie mit einem Pfeil dargestellt, der

20.8.1. Detail-Register Transition

Die aktiven Detail-Register für Transitionen sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Darstellung
Standard-Register.

Anmerkung
Die Werte im Feld "Begrenzung" der Transition sind nicht editierbar, weil sie durch die Eigenschaften der Endpunkte der Linie bestimmt werden.

Stereotyp
Standard-Register. Im UML-Metamodell sind für Transition standardmäßig keine Stereotypen definiert.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für Transition keine Standard-Eigenschaftswerte definiert.

Checkliste
Standard-Register für eine Transition.

20.8.2. Eigenschaftssymbolleiste Transition

Nach oben
Navigiert in der Hierarchie nach oben zum Eltern-Zustandsautomaten

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „ Stereotyp “) für die markierte Transition und springt sofort in das Register Eigenschaften dieses Stereotypes.

Löschen
Löscht die Transition aus dem Modell.

**Warnung**

Dies ist ein Löschen aus dem Modell, *nicht* nur aus dem Diagramm. Um eine Transition aus einem Diagramm zu entfernen, sie aber im Modell zu erhalten, verwenden Sie *Aus Diagramm entfernen* des Hauptmenüs (oder drücken Sie die Taste Entf).

### 20.8.3. Eigenschaftsfelder für eine Transition

**Name**
Textfeld. Der Name der Transition. Per Konvention beginnen Transitionsnamen mit einem Kleinbuchstaben und verwenden die Groß-/Kleinschreibung, um Wörter innerhalb des Namens zu unterscheiden.

**Anmerkung**

ArgoUML erzwingt diese Namenskonvention nicht.

**Zustandsautomat**

Ein Taste 1-Doppeldick navigiert zu dem gezeigten Zustandsautomaten.

**Zustand**
Textfeld. Zeigt den Namen des Eltern-Zustandes im Fall einer internen Transition.

Ein Taste 1-Doppeldick navigiert zu dem gezeigten Zustand.

**Quellcode**
Textfeld. Zeigt den Ursprungszustand der Transition an.

Ein Taste 1-Doppeldick navigiert zu dem markierten Eintrag.

**Ziel**
Textfeld. Zeigt den Zielzustand der Transition.

Ein Taste 1-Doppeldick navigiert zu dem markierten Eintrag.

**Auslöseimpuls**
Textfeld. Zeigt das Auslöseimpuls-Ereignis (falls vorhanden), welches diese Transition aufruft.

**Anmerkung**

In der UML ist es nicht erforderlich, dass ein Auslöseimpuls existiert, z.B., wenn ein Wächter definiert ist.


### Wächter

**Textfeld.** Zeigt den Namen eines Wächters (falls vorhanden). Der Ausdruck eines Wächters muss true sein, bevor diese Transition ausgeführt werden kann.


• **Neu.** Fügt einen neuen Wächter hinzu.

### Effekt

**Textfeld.** Zeigt die Aktion (falls vorhanden), die aufgerufen wird, wenn dies Transition ausgeführt wird.


• **Aus Modell entfernen.** Löscht die markierte Aktion aus dem Modell.

## 20.9. Ereignis

Ein Ereignis ist ein beobachtbares Vorkommen. Im UML-Metamodell ist es ein Kind von ModelElement.

Es gibt eine Anzahl unterschiedlicher Typen von Ereignissen, die Kinder eines Ereignisses innerhalb des UML-Metamodells sind.

• **Aufruf-Ereignis.** Verknüpft mit einer Operation einer Klasse, verursacht dieses Ereignis einen Aufruf der angegebenen Operation. Der erwartete Effekt ist, dass die Operationsschritte ausgeführt werden.

• **Signal-Ereignis.** Verknüpft mit einem Signal, verursacht diese Ereignis das Auslösen eines Signales.

• **Zeit-Ereignis.** Ein Ereignis, verursacht durch den Ablauf einer zeitlichen Frist.

• **Änderungs-Ereignis.** Ein Ereignis, verursacht durch einen bestimmten Ausdruck (von Attributen und Assoziationen) der true wird.

Ein Ereignis wird durch seinen Namen dargestellt.
20.9.1. Detail-Register Ereignis

Die aktiven Detail-Register für Ereignisse sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Stereotyp
Standard-Register. Im UML-Metamodell sind für Event die folgenden Standard-Stereotypen definiert.

• Erzeugen (nur für ein Aufruf- Ereignis). Erzeugen ist ein stereotypisiertes Aufruf-Ereignis, das kennzeichnet, dass die empfangende Instanz dieses Ereignis bereits erzeugt hat. In Zustandsautomaten löst es die initiale Transition auf oberster Ebene des Zustandsautomaten aus (und ist die einzige Art Auslöseimpuls, die auf eine initiale Transition angewendet werden darf.

• Zerstören (nur für ein Aufruf- Ereignis). Zerstören ist ein stereotypisiertes Aufruf-Ereignis, das kennzeichnet, dass die das Ereignis empfangende Instanz gerade zerstört wird.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für Event keine Standard-Eigenschaftswerte definiert.

20.9.2. Eigenschaftssymbolleiste Ereignis

Nach oben
Navigiert durch die zusammengesetzte Struktur nach oben.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „ Stereotyp “) für das markierte Ereignis und springt sofort in das Register Eigenschaften dieses Stereotyps.

Neuer Parameter
Erzeugt einen neuen Parameter für die Ereignisoperation als aktuellen Parameter und springt sofort in das Register Eigenschaften dieses Parameters (siehe Abschnitt 18.9, „ Parameter “).

Löschen
Löscht das Ereignis aus dem Modell.

20.9.3. Eigenschaftsfelder für ein Ereignis

Name
Textfeld. Der Name des Ereignisses. Per Konvention beginnt der Ereignisname mit einem Kleinbuchstaben und verwendet Groß-/Kleinschreibung, um Wörter innerhalb des Namens auf die gleiche Art und Weise wie Operationen zu unterscheiden.

**Anmerkung**

ArgoUML erzwingt diese Namenskonvention nicht.

**Tipp**

Für Aufruf-Ereignisse macht es Sinn, den Namen der verknüpften Operation zu verwenden. Für Signalereignisse macht es Sinn, den Namen des Signals mit einem vorangestellten Präfix \([\text{sig}]\) zu verwenden. Für Zeitereignisse verwenden Sie einen Zeitausdruck mit dem Präfix \([\text{time}]\) und für Änderungereignisse den Änderungsausdruck mit dem Präfix \([\text{change}]\).

**Namensraum**


**Parameter**

Textbereich mit den Einträgen für alle aktuellen Parameterwerte des Ereignisses (siehe Abschnitt 18.9, „Parameter“).

Ein Taste 1-Doppelklick auf einen der Parameter navigiert zu diesem Parameter, ein Taste 2-Klick öffnet ein Popup-Menü mit einem Eintrag.

- **Neuer Parameter**. Erzeugt einen neuen Parameter und navigiert zu ihm.

**Transition**

Zeigt die von dem Ereignis verursachte Transition.

Ein Taste 1-Doppelklick auf die Transition navigiert zu dieser Transition.

**Operationen**


**Signal**


- **Hinzufügen...**. Öffnet ein Dialogfenster, das es erlaubt, ein bereits existierendes Signal auszuwählen.

- **Neues Signal**. Erzeugt ein neues Signal und navigiert dort hin.

**Wenn**


Das erste der zwei Felder ist für den Ausdruck und der zweite für die Sprache, in der dieser geschrieben wurde.
Warnung
In ArgoUML V0.20 fehlt im Register Eigenschaften des Änderungereignisses ein Feld, um dem Änderungsausdruck einzugeben.

20.10. Wächter

Ein Wächter wird im Diagramm durch den Text seines Ausdrucks in eckigen Klammern ([[]]) dargestellt.

20.10.1. Detail-Register Wächter
Die aktiven Detail-Register für Wächter sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Stereotyp
Standard-Register, welches die Stereotypen für den Wächter enthält. Im UML-Metamodell sind für Guard keine Standard-Stereotypen definiert.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für Guard keine Standard-Eigenschaftswerte definiert.

20.10.2. Eigenschaftssymbolleiste Wächter

Nach oben
Navigiert in der Paketstruktur nach oben.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für den markierten Wächter und springt sofort in das Register Eigenschaften dieses Stereotyps.

Aus Modell entfernen
Löscht den Wächter aus dem Modell.
Warnung

Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm.

20.10.3. Eigenschaftsfelder für einen Wächter

Name
Textfeld. Der Name des Wächters. Per Konvention beginnt der Wächtername mit einem Kleinbuchstaben und verwendet Groß-/Kleinschreibung, um die Wörter innerhalb des Namens zu unterscheiden.

Anmerkung

ArgoUML erzwingt diese Namenskonvention nicht.

Transition
Textfeld, zeigt die Transition, die diesen Wächter beinhaltet.
Ein Tast-1-Doppelklick auf die Transition navigiert zu dieser Transition.

Ausdruck
Textfeld. Der Ausdruck, der den Wächter definiert.

Sprache
Textfeld. Gibt an, dass der Ausdruck in einer bestimmten interpretierenden Sprache geschrieben wurde, um den Text zu evaluieren.

20.11. Pseudozustand


20.11.1. Detail-Register Pseudozustand

Die aktiven Detail-Register für Pseudozustände sind die folgenden.

321
Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Darstellung
Standard-Register.

Stereotyp
Standard-Register, welches die Stereotypen des Pseudozustandes enthält. Im UML-Metamodell sind für Pseudozustand keine Standard-Stereotypen definiert.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für Pseudozustand keine Standard-Eigenschaftswerte definiert.

### 20.11.2. Eigenschaftssymbolleiste Pseudozustand

![Nach oben]
Navigiert in der Paketstruktur nach oben.

![Neuer Stereotyp]
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für den markierten Pseudozustand und springt sofort in das Register Eigenschaften dieses Stereotyps.

![Aus Modell entfernen]
Löscht den Pseudozustand aus dem Modell.

**Warnung**
Dies ist ein Löschen aus dem Modell, *nicht* nur aus dem Diagramm.

### 20.11.3. Eigenschaftsfelder für einen Pseudozustand

**Name**
Textfeld. Der Name des Pseudozustandes. Per Konvention beginnen Pseudozustandsnamen mit einem Kleinbuchstaben und verwendet die Groß-/Kleinschreibung um die Wörter innerhalb des Namens zu unterscheiden.

**Anmerkung**
ArgoUML erzwingt diese Namenskonvention nicht.
Tipp

Namen von Pseudozuständen werden im Diagramm nicht angezeigt und es ist normalerweise nicht notwendig ihnen einen Namen zu geben.

**Container**


Ein Taste 1-Doppelklick auf den Eintrag wird zu dem zusammengesetzten Zustand navigieren, der diesen Zustand enthält (oder den Zustand auf oberster Ebene, den die Wurzel der Zustandshierarchie.

**Ankommend**

Textbereich. Listet alle ankommenden Transitionen des Pseudozustandes auf.

Ein Taste 1-Doppelklick navigiert zu der markierten Transition.

**Abgehend**

Textbereich. Listet alle abgehenden Transitionen des Pseudozustandes auf.

Ein Taste 1-Doppelklick navigiert zu der markierten Transition.

20.12. Start-Zustand


Als Konsequenz daraus ist es nicht möglich, ankommende Transitionen zu haben. ArgoUML wird sie solche Transitionen nicht erzeugen lassen und wenn Sie ein Modell importieren, das solche Transitionen aufweist, wird sich eine Kritik beschweren.

Bei den meisten kann ein initialer Pseudozustand in einem zusammengesetzten Zustand vorhanden sein, der eine abgehende Transition aufweisen muss (bei den meisten).

Ein Startzustand wird im Diagramm als ausgefüllte Scheibe dargestellt.

20.13. Endezustand

Wenn eine Transition ihren Endzustand erreicht, bedeutet es den Abschluss der mit diesem zusammengesetzten Zustand verknüpften Aktivitäten, oder ganz oben, das Ende des Zustandsautomaten. Im UML-Metamodell ist FinalState ein Kind von State.

**Anmerkung**

Ein Endzustand ist ein richtiger Zustand (mit all seinen Attributen), kein Pseudozustand.

Das Beenden ganz oben bedeutet das beenden (z.B. zerstören) der Objektinstanz.

Die Darstellung eines Endzustandes im Diagramm ist ein Kreis mit einer schmalen Scheibe im Zentrum.
20.13.1. Detail-Register Endezustand

Die aktiven Detail-Register für einen Endezustand sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Darstellung
Standard-Register.

Stereotyp
Standard-Register, welches die Stereotypen des Endzustandes enthält. Im UML-Metamodell sind für einen Final State keine Standard-Stereotypen definiert.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für einen Final State keine Standard-Eigenschaftswerte definiert.

20.13.2. Eigenschaftssymbolleiste Endzustand

Nach oben
Navigiert in der Paketstruktur nach oben.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für den markierten Zustand und springt sofort in das Register Eigenschaften dieses Stereotyps.

Aus Modell entfernen
Löscht den Endzustand aus dem Modell.

Warnung
Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm.

20.13.3. Eigenschaftsfelder für einen Endzustand

Name
Textfeld. Der Name des Endzustandes. Per Konvention beginnt der Endzustandname mit einem Kleinbuchstaben und verwendet die Groß-/Kleinschreibung um Wörter innerhalb des Namens zu unterscheiden.
Anmerkung
ArgoUML erzwinge diese Namenskonvention nicht.

Tipp
Endzustandnamen werden im Diagramm dargestellt, aber es ist normalerweise nicht notwendig, ihnen einen Namen zu geben.

Container

Ein Taste 1-Doppelklick auf den Eintrag wird zu dem zusammengesetzten Zustand navigieren, der diesen Zustand enthält (oder den Zustand auf oberster Ebene, die die Wurzel der Zustandshierarchie ist).

Eintritts-Aktion
Textfeld. Zeigt den Namen der Aktion (falls vorhanden), die beim Eintritt in diesen Endzustand ausgeführt wird.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag, Taste 2 öffnet ein Popup-Menü mit zwei Einträgen:


- **Aus Modell entfernen**: Löscht die Eintritts-Aktion.

Ankommend
Textbereich. Listet alle ankommenden Transitionen für den Endzustand auf.

Ein Taste 1-Doppelklick navigiert zu der markierten Transition.

Intere Transitionen

Ein Taste 1-Doppelklick navigiert zu der markierten Transition.

### 20.14. Kreuzung


Ein vordefinierter Wächter bedeutet **entweder**, er kann meistens für eine abgehende Transition definiert werden. Diese Transition ist freigegeben, wenn alle Wächter die anderen Transitionen mit false kennzeichnen.

Entsprechend dem UML-Standard ist sein Symbol ein kleiner schwarzer Kreis. Alternativ kann er durch
einen Diamanten dargestellt werden (im Fall der "Entscheidung" für Aktivitätsdiagramme). ArgoUML stellt eine Kreuzung in einem Diagramm nur als ausgefüllter (standardmäßig weiss) Diamanten dar und unterstützt nicht das schwarze Kreissymbol für eine Kreuzung.

20.15. Entscheidung


Ein vordefinierter Wächter bedeutet andernfalls, dass er meistens für eine abgehende Transition definiert werden kann. Diese Transition wird freigegeben, wenn alle Wächter die anderen Transitionen mit false kennzeichnen.

Anmerkung

Diese Sorte von Pseudozustand wurde formal in ArgoUML Abzweig genannt.

Eine Entscheidung wird im Diagramm als kleiner gefüllter (standardmäßig weiss) Kreis dargestellt (erinnert an ein kleines Zustandssymbol).

20.16. Gabelung

Eine Gabelung ist ein Pseudozustand (siehe Abschnitt 20.11, „Pseudozustand“), der eine Transition in zwei oder mehr nebeneinanderliegende Transitionen aufsplittet.

Achtung

Die abgehende Transition sollte keine Wächter haben. ArgoUML erzwingt dies aber nicht.

Eine Gabelung wird im Diagramm als ausgefüllte (standardmäßig schwarz) horizontaler Balken dargestellt.

Tipp

Dieser Balken kann vertikal ausgerichtet werden, indem Sie die Gabelung markieren und diese mit der Taste 1 in eine seiner Ecken ziehen.

20.17. Vereinigung

Eine Vereinigung ist ein Pseudozustand (siehe Abschnitt 20.11, „Pseudozustand“), der zwei oder mehr nebeneinanderliegende Transitionen zu einer Transition vereinigt.

Achtung

Die ankommenden Transitionen sollten keine Wächter aufweisen. ArgoUML erzwingt dies
jedoch nicht.

Eine Verzweigung wird im Diagramm als ausgefüllte (standardmäßig schwarz) horizontaler Balken dargestellt.

**Tipp**

Dieser Balken kann vertikal ausgerichtet werden, indem Sie die Vereinigung markieren und diese mit der Taste 1 in eine seiner Ecken ziehen.

### 20.18. Flache Historie


Wenn in eine mehrstufige Hierarchie von zusammengesetzten Zuständen plaziert, erinnert sich die flache Historie nur an Zustände, die den gleichen Container haben, wie der Pseudozustand der Historie. Sie speichert keine Unterzustände zurück, die tiefer liegen als der Pseudozustand der Historie selbst.

Eine flache Historie wird im Diagramm als ein Kreis, der den Buchstaben \(H\) enthält, dargestellt.

### 20.19. Tiefe Historie


Wenn in eine mehrstufige Hierarchie von zusammengesetzten Zuständen plaziert, erinnert sich die tiefe Historie rekursiv an alle Zustände, die in der Container-Pseudozustand-Historie enthalten sind. Sie speichert alle Unterzustände zurück, egal wie tief sie in der Historie waren.

Eine tiefe Historie wird im Diagramm als ein Kreis, der den Buchstaben \(H^*\) enthält, dargestellt.

### 20.20. Synchronisationszustand


Ein Synchronisationszustand wird als kleiner Kreis mit der oberen Grenze innerhalb des Kreises
20.20.1. Detail-Register Synchronisierungszustand

Die aktiven Detail-Register für einen Synchronisierungszustand sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Darstellung
Standard-Register.

Stereotyp
Standard-Register, welches die Stereotypen des Synchronisierungszustandes enthält. Im UML-Metamodell sind für Synch State keine Standard-Stereotypen definiert.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für Synch State keine Standard-Eigenschaftswerte definiert.

20.20.2. Eigenschaftssymbolleiste Synchronisierungszustand

Nach oben
Navigiert in der Paketstruktur nach oben.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für den markierten Synchronisierungszustand und springt sofort in das Register Eigenschaften dieses Stereotyps.

Aus Modell entfernen
Löscht den Synchronisierungszustand aus dem Modell.

Warnung
Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm.

20.20.3. Eigenschaftsfelder für einen
Synchronisationszustand

Name
Textfeld. Der Name des Synchronisationszustandes. Per Konvention beginnen Synchronisationszustandsnamen mit einem Kleinbuchstaben und verwenden die Groß-/Kleinschreibung, um Wörter innerhalb des Namens unterscheiden zu können.

Anmerkung
ArgoUML erzwingt diese Namenskonvention nicht.

Tipp
Synchronisationszustandsnamen werden im Diagramm nicht angezeigt und es ist normalerweise auch nicht notwendig ihnen Namen zu geben.

Container

Grenze
Editierbares Textfeld. Zeigt die Grenze des Synchronisationszustandes. Dies ist eine positive Integerzahl oder der Wert unbegrenzt (dargestellt durch ein "*"), die die maximale Anzahl des Synchronisationszustandes spezifiziert. Die Anzahl ist die Differenz zwischen der Anzahl der ankommenden und abgehenden Transitionen eines feuernden Synchronisationszustandes.

Ankommend

Abgehende Transitionen
Kapitel 21. Modellelement-Referenz
Kollaborationsdiagramm

21.1. Einleitung

Dieses Kapitel beschreibt jedes Modellelement, das innerhalb von Kollaborationsdiagrammen erzeugt werden kann. Beachten Sie, dass einige Sub-Modellelemente von Modellelementen im Diagramm nicht selbst im Diagramm erscheinen können.


Achtung


21.1.1. Einschränkungen die Kollaborationsdiagramme in ArgoUML betreffend

Das Kollaborationsdiagramm ist in ArgoUML aktuell eher unterentwickelt. Im Einzelnen gibt es keinen Weg, Instanzkollaborationen darzustellen (basierend auf Objekten und Verknüpfungen), stattdessen die Spezifikation von Kollaborationen.


21.2. Klassifizierer Rolle

Eine Klassifizierer Rolle ist eine Spezialisierung eines Klassifizierers, der dazu verwendet wird, sein Verhalten in einem bestimmten Kontext darzustellen. Im UML-Metamodell ist Classifier Role eine Subklasse von Classifier. Innerhalb eines Kollaborationsdiagramms können Klassifizierer Rollen auf einen von zwei Wegen verwendet werden:

- Um den Klassifizierer in einem bestimmten Verhaltenskontext darzustellen (der Spezifikations-Level; oder
• um eine bestimmte Instanz des Klassifizierers zu spezifizieren (der Instanz-Level).

In dieser letzten Form sind Klassifizierer Rollen identisch mit den in Sequenzdiagrammen verwendeten Instanzen (siehe Kapitel 19, Modellelement-Referenz Sequenzdiagramm) und ein Kollaborationsdiagramm zeigt die gleichen Informationen wie das Sequenzdiagramm, aber in einer anderen Darstellung.

**Achtung**

Ein Kollaborationsdiagramm sollte die auf dem Spezifizierungs-Level und dem Instanz-Level verwendeten Klassifizierer Rollen nicht vermischen.


**Achtung**

Eine Klassifizierer Rolle sollte wahrscheinlich auch den Objektnamen anzeigen (falls vorhanden), dem der Rollenname des Klassifiziers, getrennt durch einen Schrägstrich (/) vorangestellt wird. Dies erlaubt es, das Klassifizierer Rollen auf dem Spezifikations-Level im Diagramm von Instanzen auf dem Instanz-Level unterschieden werden können.

ArgoUML zeigt den Schrägstrich nicht an, aber es gibt keinen Weg die Instanz zu definieren.

### 21.2.1. Detail-Register Klassifizierer Rollen

Die aktiven Detail-Register für Klassifizierer Rollen sind die folgenden.

**Zu-Bearbeiten-Element**

Standard-Register.

**Eigenschaften**


**Dokumentation**

Standard-Register.

**Darstellung**

Standard-Register.

**Quellcode**

Standard-Register, aber ohne Inhalte.

**Achtung**

Eine Klassifizierer Rolle sollte keine Code generieren, so dass die Tatsache, dass dieses Register aktiv ist, wahrscheinlich ein Fehler ist.
Eigenschaftswerte

Standard-Register. Im UML-Metamodell sind für Classifier Role die folgenden Standard-Eigenschaftswerte definiert.

- **persistence** (von der Superklasse Classifier). Zeigt die Dauerhaftigkeit der Zustandsinformation, die mit der Klassifizierer Rolle verknüpft ist. Werte sind *transitory* (der Zustand wird zerstört, wenn die Klassifizierer Rolle zerstört wird) und *persistent* (der Zustand bleibt erhalten, wenn die Klassifizierer Rolle zerstört wird).

- **semantics** (von der Superklasse Classifier). Der Wert ist eine Spezifikation der Semantik der Klassifizierer Rolle.

- **derived** (von der Superklasse ModelElement). Der Wert *true* bedeutet, dass die Klassifizierer Rolle redundant ist - sie kann formal von anderen Elementen abgeleitet werden, oder *false*, wenn sie es nicht kann.

**Anmerkung**


**Anmerkung**


### 21.2.2. Eigenschaftssymbolleiste Klassifizierer Rolle

- **Nach oben**
  Navigiert durch die Paketstruktur nach oben.

- **Neues Empfangssignal**
  Erzeugt ein neues Empfangssignal und springt sofort in das Register Eigenschaften dieses Empfangssignales.

- **Neuer Stereotyp**
  Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Klassifizierer Rolle und springt sofort in das Register Eigenschaften dieses Stereotyps.

- **Löschen**
  Löscht die Klassifizierer Rolle aus dem Modell.

**Warnung**
Dies ist ein Löschen aus dem Modell, *nicht* nur aus dem Diagramm. Um eine Klassifizierer Rolle aus dem Diagramm zu löschen, aber im Modell zu erhalten, verwenden Sie *Aus Diagramm entfernen* im Hauptmenü (oder drücken Sie die Taste Entf).

### 21.2.3. Eigenschaftsfelder für eine Klassifizierer Rolle

**Name**
Textfeld. Der Name der Klassifizierer Rolle. Per Konvention beginnen Klassifizierer Rollennamen mit einem Kleinbuchstaben und verwenden die Groß-/Kleinschreibung, um Wörter innerhalb des Namens zu unterscheiden.

**Anmerkung**
ArgoUML erzwingt diese Namenskonvention nicht.

**Stereotyp**
Kombinationsfeld. Die Klassifizierer Rolle bietet standardmäßig die UML-Standard-Stereotypen für einen Klassifizierer an (*metaclass*, *powertype*, *process*, *thread* und *utility*).

**Stereotyp navigieren**
Symbol. Wenn ein Stereotyp markiert wurde, navigiert es zum Eigenschaftsfenster des Stereotypen (siehe Abschnitt 16.6, „Stereotyp“).

**Namensraum**
Textfeld. Gibt den Namensraum für die Klassifizierer Rolle wieder, welche immer der aufnehmenden Kollaboration entspricht.

Ein Taste 1-Doppelklick auf den Eintrag navigiert zu der Kollaboration.

**Kardinalität**
Editierbares Kombinationsfeld. Der Standardwert ist *, was bedeutet, dass es beliebig viele Instanzen dieser Klassifizierer Rolle gibt, die in der Kollaboration eine Rolle spielen. Das Kombinationsfeld bietet einige andere Kardinalitäten an. Z.B. 1..1 würde bedeuten, dass nur eine Instanz in der Kollaboration eine Rolle spielen würde.

ArgoUML schränkt Sie hinsichtlich der vordefinierten Kardinalitäten nicht ein. Sie können diese Feld frei editieren.

**Basis**


### Abbildung 21.2. Das Dialogfenster „Kontext hinzufügen“
• Entfernen. Erlaubt das Entfernen von Klassifizierern aus der Liste, ohne das Dialogfenster zu benutzen.

Generalisierungen
Textbereich. Listet jede Klassifizierer Rolle auf, die diese Klassifizierer Rolle generalisiert.

Ein Taste 1-Doppelklick navigiert zu der Generalisierung und öffnet dessen Register Eigenschaften.

Spezialisierungen
Textfeld. Listet jede spezialisierte Klassifizierer Rolle (z.B. für die diese Klassifizierer Rolle eine Generalisierung ist) auf.

Ein Taste 1-Doppelklick navigiert zu der Spezialisierung und öffnet dessen Register Eigenschaften.

Rolle (Assoziationsende)
Textbereich. Listet die Rollen (Assoziationsenden) auf, die mit dieser Klassifizierer Rolle verbunden sind.
Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

Verfügbare Inhalte
Textbereich. Listet das Subset von Modellelementen auf, die im Basis-Klassifizierer enthalten ist, der in der Kollaboration verwendet wird.

Ein Taste 1-Doppelklick navigiert zu dem Modellelement und öffnet dessen Register Eigenschaften.

Verfügbare Eigenschaften
Textfeld. Listet das Subset an Eigenschaften des Basis-Klassifizierers auf, der in der Kollaboration verwendet wird.

Ein Taste 1-Doppelklick navigiert zu der Eigenschaft und öffnet dessen Register Eigenschaften.

21.3. Assoziationsrolle


Eine Assoziationsrolle wird in einem Kollaborationsdiagramm von ArgoUML als Linie dargestellt, welche die betreffenden Instanzen miteinander verbindet. In einem Sequenzdiagramm ist die Darstellung jedoch modifiziert, um den Aktionstyp zu reflektieren, der im Zusammenhang mit dem Impuls auf die Verknüpfung ausgeführt wird (siehe Abschnitt 19.3, „Impuls“).

Die Assoziationsrolle wird mit dem Namen der Assoziationsrolle (falls vorhanden) gekennzeichnet.

Eine Assoziationsrolle zeigt ihren Namen und den Assoziationsnamen entsprechend der folgenden Syntax an:

/ Assoziationsrollename : Assoziationsname

auf die gleiche Art und Weise wie eine Klassifiziererrolle. Die generische Syntax ist:

I / R : C

was für eine Instanz mit dem Namen I, vom Klassifizierer C abstammend und die Rolle R spielend steht.

21.3.1. Detail-Register Assoziationsrolle

Die aktiven Detail-Register für Assoziationsrollen sind die folgenden.

Zu-Bearbeiten-Element

Eigenschaften

Dokumentation
Standard-Register.

Darstellung
Standard-Register.
Anmerkung

Die Werte im Feld "Begrenzung" der Assoziationsrolle sind nicht editierbar, weil sie durch die Eigenschaften der Endpunkte der Linie bestimmt werden.

Quellcode
Standard-Register, aber ohne Inhalte.

Achtung

Eine Assoziationsrolle sollte keinen Code generieren, so dass die Tatsache, dass dieses Register aktiv ist, wahrscheinlich einen Fehler darstellt.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für AssociationRole die folgenden Standard-Eigenschaftswerte definiert.

- persistence (von der Superklasse Association). Der Wert transitory gibt an, dass der Zustand zerstört wird, wenn die Instanz zerstört wird, oder persistent kennzeichnet, dass der Zustand erhalten bleibt, wenn die Instanz zerstört wird.

- derived (von der Superklasse ModelElement). Der Wert true bedeutet, dass die Assoziation redundant ist - sie kann formal von anderen Elementen abgeleitet werden, oder false, wenn sie es nicht kann.

Anmerkung

Abgeleitete Assoziationsrollen haben ihren Wert in der Analyse, um nützliche Namen oder Konzepte einzuführen und im Design, um eine Wiederverarbeitung zu verhindern.

Anmerkung

Die UML-Metaklasse Element, aus der alle anderen Modellemente abgeleitet werden, beinhaltet den Eigenschaftswert documentation, der in ArgoUML im Register Dokumentation bearbeitet wird.

Checkliste
Standard-Register für eine Assoziationsrolle.

21.3.2. Eigenschaftssymbolleiste Assoziationsrolle

Nach oben
Navigiert in der Paketstruktur nach oben.

Löschen
Löscht die Assoziationsrolle aus dem Modell.

**Warnung**

Dies ist ein Löschen aus dem Modell, *nicht* nur aus dem Diagramm. Um eine Assoziationsrolle aus dem Diagramm zu löschen, sie aber im Modell zu erhalten, verwenden Sie *Aus Diagramm entfernen* im Hauptmenü (oder drücken Sie die Taste Entf).

### 21.3.3. Eigenschaftsfelder für eine Assoziationsrolle

**Name**

Textfeld. Der Name der Assoziationsrolle, der im Diagramm angezeigt wird. Per Konvention beginnt der Name einer Assoziationsrolle mit einem Kleinbuchstaben und es werden Groß-/Kleinschreibung verwendet, um Wörter innerhalb des Namens zu unterscheiden.

**Anmerkung**

ArgoUML erzwingt diese Namenskonvention nicht.

**Stereotyp**


**Stereotyp navigieren**

Symbol. Wenn ein Stereotyp markiert wurde, navigiert es in das Eigenschaftsfenster des Stereotyps (siehe Abschnitt 18.5, „Stereotyp“).

**Namensraum**

Textfeld. Nimmt den Namensraum für die Assoziationsrolle auf. Dies ist die Pakethierarchie.

Ein Taste 1-Doppelklick auf den Eintrag navigiert zu dem gezeigten Element.

**Basis**

Kombinationsfeld. Nimmt die Assoziation auf, die Basis der Assoziationsrolle ist.

Das Kombinationsfeld zeigt alle Assoziationen, die zwischen den Klassifizieren existieren, die mit der verbundenen Klassifiziererrolle korrespondieren.

**Endpunkte (Assoziationsrolle)**

Textbereich. Listet die Enden dieser Assoziationsrolle auf. Eine Assoziationsrolle kann beliebig viele Enden haben, aber im Allgemeinen sind zwei die einzig sinnvolle Anzahl (das Verbinden von Objekten kann in Instanz-Level-Diagrammen zu einem dritten Ende führen, aber dies wird von ArgoUML nicht unterstützt). Mehr über Assoziationsrollen, siehe Abschnitt 21.4, „Rolle Assoziationsende“.

Die Namen werden aufgelistet, es sei denn die Rolle des Assoziationsendes hat keinen Namen, dann wird sie als *(Unbenannte AssociationEndRole)* dargestellt.

Ein Taste 1-Doppelklick auf die Rolle Assoziationsende navigiert zu diesem Ende.
Nachrichten
Textbereich. Listet die Nachrichten auf, die mit dieser Assoziationsrolle verknüpft sind.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

21.4. Rolle Assoziationsende

Eine Rolle Assoziationsende ist eine Spezialisierung eines Assoziationsendes, und wird dazu benutzt, das Verhalten eines Assoziationsendes in einem bestimmten Kontext zu beschreiben. Im UML-Metamodell ist AssociationEndRole eine Subklasse von AssociationEnd.


Die Rolle Assoziationsende hat keinen direkten Zugriff auf ein Diagramm, obwohl dessen Stereotyp, Name und Kardinalität am relevanten Ende der Eltern-Assoziationsrolle angezeigt wird (siehe Abbildung 21.1, „Denkbare Modellelemente in einem Kollaborationsdiagramm.“), und einige seiner Eigenschaften direkt mit einem Taste 2-Klick eingestellt werden können. Dort wo trennbare oder untrennbare Aggregation für eine Rolle Assoziationsende ausgewählt wurde, wird das gegenüberliegende Ende als ausgefüllter Diamant (untrennbare Aggregation) oder unausgefüllter Diamant (trennbare Aggregation) dargestellt.

Anmerkung
ArgoUML (V0.18) unterstützt aktuell nicht die Anzeige von Qualifizierern im Diagramm, wie im UML 1.4-Standard beschrieben.

Achtung

21.4.1. Detail-Register Rolle Assoziationsende

Die aktiven Detail-Register für eine Rolle Assoziationsende sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Quellcode
Standard-Register. Es wird kein Code für eine Rolle Assoziationsende generiert.
Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für AssociationEndRole die folgenden Standard-Eigenschaftswerte definiert.

- **derived** (von der Superklasse ModelElement). Der Wert **true** bedeutet, dass die Rolle Assoziationsende redundant ist - sie kann von anderen Elementen abgeleitet werden, oder **false**, wenn sie es nicht kann.

**Tipp**

**Anmerkung**

### 21.4.2. Eigenschaftssymbolleiste Rolle Assoziationsende

- **Nach oben**
  Navigiert zu der Assoziationsrolle, zu der dieses Rollenende gehört.
- **Gehe zum anderen Ende**
  Navigiert zum anderen Ende der Assoziationsrolle.
- **Neuer Qualifizierer**
  Erzeugt einen neuen Qualifizierer für die markierte Rolle Assoziationsende und springt sofort in das Register Eigenschaften dieses Qualifizierers.

**Warnung**
Qualifizierer werden in ArgoUML V0.18 nur teilweise unterstützt. Daher erzeugt die Aktivierung dieser Schaltfläche einen Qualifizierer im Modell, der nicht im Diagramm angezeigt wird. Auch das Eigenschaftsfenster dieses Qualifizierers entspricht dem eines regulären Attributes.

- **Neuer Stereotyp**
  Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Rolle.
Assoziationsende und springt sofort in das Register Eigenschaften dieses Stereotyps.

Löschen
Löscht das markierte Assoziationsende aus dem Modell.

Anmerkung

21.4.3. Eigenschaftsfelder für eine Rolle Assoziationsende

Name
Textfeld. Der Name einer Rolle Assoziationsende, die einen Rollename für dieses Ende der Assoziationsrolle aufnimmt. Dieser Rollename kann für die Navigation verwendet werden und im Implementationskontext erlaubt ein Name, wie das Ursprungsende einer Assoziationsrolle mit dem Zielende referenziert werden kann.

Anmerkung
ArgoUML erzwingt keine Namenskonventionen für Rollen Assoziationsenden.

Stereotyp
Kombinationsfeld. Die Rolle Assoziationsende bietet standardmäßig die UML-Standard-Stereotypen für AssociationEndRole (association, global, local, parameter, self) an.

Stereotyp navigieren
Symbol. Wenn ein Stereotyp markiert wurde, navigiert es in das Eigenschaftsfenster des Stereotyps (siehe Abschnitt 16.6, „Stereotyp“).

Basis

Assoziationsrolle

Typ
Das Kombinationsfeld bietet Zugriff auf alle Standard-UML-Typen, die durch ArgoUML unterstützt werden und alle neuen, innerhalb des aktuellen Modells erzeugten Klassen.

Dies ist der Entitätstyp, der diesem Ende der Assoziationsrolle zugewiesen wurde.

Kardinalität

Alle restlichen Eigenschaften
Siehe Abschnitt 18.13.3, „Eigenschaftsfelder für ein Assoziationsende“. Da diese mit den Feldern eines Assoziationsendes vollständig übereinstimmen, werden sie hier nicht wiederholt.

21.5. Nachricht


Warnung
Die aktuelle Release von ArgoUML kann die Positionierung der Nachricht nach dem erneuten Öffnen des Projektes nicht wiederherstellen, z.B. wie als würden die Positionen in der Projektdatei nicht gespeichert.

21.5.1. Detail-Register Nachricht

Die aktiven Detail-Register für Nachrichten sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Darstellung

Achtung
Das Ändern der Position einer Nachricht durch das Editieren der Werte des Feldes Begrenzung ist in der Release ArgoUML V0.18 möglich, wird aber die Position der Nachricht nur temporär verändern, wie oben beschrieben.
Quellcode
Standard-Register, das die Nummer der Nachricht und den, durch einen Doppelpunkt separierten Aktionsausdruck anzeigt (wenn UML 1.4 im Auswahlfeld markiert wurde).

Achtung
Eine Nachricht sollte wahrscheinlich keine Code über sich selbst generieren. Dies sollte der Aktion und wahrscheinlich dem damit verknüpften Impuls überlassen bleiben. In jedem Fall, werden Änderungen in diesem Register ignoriert.

Eigenschaftswerte
Standard-Register. Im UML-Metamodell sind für Message die folgenden Standard-Eigenschaftswerte definiert.

- derived (von der Superklasse ModelElement). Der Wert true bedeutet, dass die Nachricht redundant ist - sie kann formal von anderen Elementen abgeleitet werden, oder false, wenn sie es nicht kann.

Anmerkung
Abgeleitete Nachrichten haben ihren Wert in der Analyse und Design, um nützliche Namen oder Konzepte einzuführen und im Design, um eine Wiederverarbeitung zu verhindern.

Anmerkung
Die UML-Metaklasse Element, von der alle anderen Modellelemente abgeleitet werden, beinhaltet einen Eigenschaftswert documentation, der in ArgoUML im Register Dokumentation bearbeitet wird.

21.5.2. Eigenschaftssymbolleiste Nachricht

Nach oben
Navigiert in der Paketstruktur nach oben.

Neue Aktion
Erzeugt eine neue Aktion (siehe Abschnitt 20.3, „Aktion“) für das markierte Objekt und springt sofort in das Register Eigenschaften dieser Aktion.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Nachricht und springt sofort in das Register Eigenschaften dieses Stereotypes.

Löschen
Löscht die Nachricht aus dem Modell.
Warnung

Dies ist ein Löschen aus dem Modell, *nicht* nur aus dem Diagramm. Um eine Nachricht aus dem Diagramm zu löschen, sie aber im Modell zu erhalten, verwenden Sie *Aus Diagramm entfernen* im Hauptmenü (oder drücken Sie die Taste Entf).

21.5.3. Eigenschaftsfelder für eine Nachricht

**Name**
Textfeld. Der Name der Nachricht ist gewöhnlich eine Sequenznummer, entweder eine Integerzahl oder eine Dezimalzahl (erlaubt es, alternative Nachrichtenhierarchien klar zu beschreiben). ArgoUML wird standardmäßig eine Interzahl als Sequenznummer unterstützen.

**Stereotyp**

**Stereotyp navigieren**
Symbol. Wenn ein Stereotyp markiert wurde, navigiert es in das Eigenschaftsfenster des Stereotyps (siehe Abschnitt 16.6, „Stereotyp“).

**Interaktion**
Textfeld. Nimmt die Interaktion auf, von der die Nachricht ein Teil ist.

Ein Taste 1-Doppelklick auf den Eintrag navigiert zu der Interaktion.

**Sender**
Textfeld. Identifiziert die Klassifiziererrolle, die diese Nachricht sendet.

Ein Taste 1-Doppelklick navigiert zu dem Sender Klassifiziererrolle.

**Empfänger**
Textfeld. Identifiziert die Klassifiziererrolle, die diese Nachricht empfängt.

Ein Taste 1-Doppelklick navigiert zu dem Empfänger Klassifiziererrolle.

**Auslöser**
Kombinationsfeld. Indentifiziert die Nachricht, die das Verhalten aufruft, die das Senden dieser Nachricht verursacht.

Ein Taste 1-Klick erlaubt das Auswählen der Nachricht.

**Aktion**
Textfeld. Listet die Aktion (siehe Abschnitt 20.3, „Aktion“) auf, die diese Nachricht aufruft, um einen Impuls auszulösen.


- *Neu.* Für eine neue Aktion hinzu.

Dieses Element ist deaktiviert, wenn eine Aktion bereits existiert.
Vorgänger
Textbereich. Identifiziert die Nachricht, deren Ausführungsabschluss diese Nachricht ermöglicht.


- Hinzufügen. Öffnet ein Dialogfenster, das es erlaubt, Nachrichten auszuwählen. Sie Bild unten.

Dieser Eintrag ist deaktiviert, wenn keine Nachricht existiert.

Abbildung 21.3. Das Dialogfenster „Vorgänger hinzufügen“
Kapitel 22. Modellelement-Referenz
Aktivitätsdiagramm

22.1. Einleitung

Dieses Kapitel beschreibt jedes Modellelement, das innerhalb eines Aktivitätsdiagrammes erzeugt werden kann. Beachten Sie, dass einige Submodellelemente von Modellelementen aktuell nicht selbst im Diagramm erscheinen.

Es gibt eine enge Beziehung zwischen diesem Material und dem Register Eigenschaften des Detailfensters (siehe Abschnitt 13.3, „Das Register Eigenschaften“). Dieser Abschnitt behandelt die Eigenschaften im Allgemeinen, während sie in diesem Kapitel mit spezifischen Modellelementen verknüpft sind.

Abbildung 22.1, „Denkbare Modellelemente in einem Aktivitätsdiagramm“ zeigt ein Aktivitätsdiagramm mit allen denkbaren Modellelementen.

Abbildung 22.1. Denkbare Modellelemente in einem Aktivitätsdiagramm.
22.1.1. Einschränkungen, die Aktivitätsdiagramme in ArgoUML betreffend

Aktivitätsdiagramme sind in ArgoUML noch nicht vollständig entwickelt. Einige Aspekte sind noch nicht vollständig implementiert, oder verhalten sich nicht wie erwartet. Insbesondere fehlen Aufrufzustände, Wellenlinien (swim lanes), Steuersymbole (Signale), Subaktivitäten, Synchronisationszustände. Interaktionen mit anderen Klassifizierern werden über einen Objekt-Fluss-Zustand angeboten, der nur teilweise implementiert ist.

22.2. Aktionszustand

Ein Aktionszustand stellt die Ausführung einer atomaren Aktion dar, in der Regel der Aufruf einer

**Achtung**


**Anmerkung**

Wie bei einem gewöhnlichen Zustand, einer internen Transition, sind bei Aktionszuständen eine Austrittsaktion und eine Ausführenaktion nicht erlaubt.


**Achtung**

Der UML-Standard spezifiziert, dass der im Aktionszustand angezeigte Text in einem Aktivitätsdiagramm den mit der Eintrittsaktion verknüpften Ausdruck enthalten sollte - was als solches seit ArgoUML V0.18 implementiert ist. In den letzten Versionen von ArgoUML (0.16.1 und vorher), zeigte das verwendete Diagramm den Namen des Aktionszustandes. Das Laden eines, mit einer älteren Version erzeugten Projektes, konvertierte dieses in das richtige Format, um mit dem UML-Standard konform zu sein. Dieser Prozess ist für den Anwender transparent entworfen worden, und die einzige Auswirkung ist, dass das Aktivitätsdiagramm im Projekt nicht richtig angezeigt wird, wenn es wieder in einer älteren Version von ArgoUML geladen wird.

### 22.2.1. Detail-Register Aktionszustand

Die aktiven Detail-Register für Aktionszustände sind die folgenden.

**Zu-Bearbeiten-Element**

Standard-Register.

**Eigenschaften**


**Dokumentation**

Standard-Register.

**Darstellung**

Standard-Register. Die Werte für die Begrenzu des Aktionszustandes definieren den umgebenden Rahmen des Aktionszustandes.

**Stereotyp**

Standard-Register, das die Stereotypen eines Aktionszustandes anzeigt. Im UML-Metamodell sind
standardmäßig keine Stereotypen für Aktionszustände definiert.

**Eigenschaftswerte**

Standard-Register. Im UML-Metamodell sind für ActionState keine Standard-Eigenschaftswerte definiert.

### 22.2.2. Eigenschaftssymbolleiste Aktionszustand

**Nach oben**

Navigiert durch die Umgebungsstruktur. Aktionszustände sind im obersten Zustand enthalten (ansonsten unsichtbar).

**Neuer Stereotyp**

Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für den markierten Aktionszustand und springt sofort in das Register Eigenschaften dieses Stereotyps.

**Aus Modell entfernen**

Löscht den Aktionszustand aus dem Modell.

**Warnung**


Aus diesem Grund zeigt ArgoUML für Aktionszustände nicht das Popup-Menü **Aus Diagramm entfernen**.

### 22.2.3. Eigenschaftsfelder für einen Aktionszustand

**Name**

Textfeld. Der Name eines Aktionszustandes. Per Konvention beginnt der Name eines Aktionszustandes mit einem Kleinbuchstaben und verwendet Groß-/Kleinschreibung, um Wörter innerhalb des Namens zu unterscheiden.

**Anmerkung**

ArgoUML erzwingt diese Namenskonvention nicht.

**Container**


**Eintrittsaktion**


- **Aus Modell entfernen**. Löscht die Eintrittsaktion.

**Verzögerte Ereignisse**

Textfeld. Die verzögerten Ereignisse des Aktionszustandes.

**Ankommend**

Textbereich. Listet die Transitionen auf, die in diesen Aktionszustand eintreten.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

**Abgehend**

Textbereich. Listet die Transitionen auf die diesen Aktionszustand verlassen (austreten).

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

### 22.3. Aktion

Dieses Modellelement ist im Kontext des Zustandsdiagrammes beschrieben (siehe Abschnitt 20.3, „Aktion“).

### 22.4. Transition

Dieses Modellelement ist im Kontext des Zustandsdiagrammes beschrieben (siehe Abschnitt 20.8, „Transition“).

**Achtung**


Die aktuelle Release von ArgoUML überprüft derzeit nicht, ob diese Bedingung erfüllt ist.

**Anmerkung**

Transitionen zu und von Objektflusszuständen werden gepunktet, um den Objektfluss vom Steuerungsfluss zu unterscheiden.

### 22.5. Wächter

Dieses Modellelement ist im Kontext des Zustandsdiagrammes beschrieben (siehe Abschnitt 20.10, „Wächter“).
22.6. Startzustand
Dieses Modellelement ist im Kontext des Zustandsdiagrammes beschrieben (siehe Abschnitt 20.12, "Start-Zustand ").

22.7. Endezustand
Dieses Modellelement ist im Kontext des Zustandsdiagrammes beschrieben (siehe Abschnitt 20.13, "Endezustand ").

22.8. Kreuzung (Entscheidung)
Dieses Modellelement ist im Kontext des Zustandsdiagrammes beschrieben (siehe Abschnitt 20.14, "Kreuzung ").

22.9. Gabelung
Dieses Modellelement ist im Kontext des Zustandsdiagrammes beschrieben (siehe Abschnitt 20.16, "Gabelung ").

22.10. Vereinigung
Dieses Modellelement ist im Kontext des Zustandsdiagrammes beschrieben (siehe Abschnitt 20.17, "Vereinigung ").

22.11. Objektflosszustand
(Noch zu beschreiben)
Kapitel 23. Modelelement-Referenz
Verteilungsdiagramm

23.1. Einleitung

Dieses Kapitel beschreibt jedes Modelelement, das innerhalb eines Verteilungsdiagrammes erzeugt werden kann. Beachten Sie, dass einige Submodelelemente von Modelelementen des Diagrammes aktuell im Diagramm nicht selbst erscheinen.

Es gibt eine enge Beziehung zwischen diesem Material und dem Register Eigenschaften des Detailfensters (siehe Abschnitt 13.3, „Das Register Eigenschaften“). Dieser Abschnitt behandelt die Eigenschaften im Allgemeinen, während sie in diesem Kapitel mit spezifischen Modelelementen verknüpft sind.


Achtung


Abbildung 23.1, „Denkbare Modelelemente in einem Komponentendiagramm.“ zeigt ein Komponentendiagramm mit allen denkbaren Modelelementen.

Abbildung 23.1. Denkbare Modelelemente in einem Komponentendiagramm.
Abbildung 23.2, „Denkbare Modellelemente in einem Verteilungsdiagramm.“ zeigt ein Verteilungsdiagramm mit allen denkbaren Modellelementen.

Abbildung 23.2. Denkbare Modellelemente in einem Verteilungsdiagramm.

23.1.1. Einschränkungen, Verteilungsdiagramme in ArgoUML betreffend

Das Verteilungsdiagramm wird in der Regel gut gezeichnet, aber es gibt nur eine Teilmenge von Beziehungen, die als verfügbar dargestellt werden sollten, was die Fähigkeit, dynamisches Verhalten des entwickelten Codes darzustellen einschränkt.

Es ist nicht möglich, neue Schnittstellen in diesem Diagramm zu erzeugen; sie können nur hinzugefügt werden, wenn sie zuerst im Modell erzeugt wurden (durch einzeichnen in ein Klassendiagramm).

Es ist ein Nachteil, dass die alternative Darstellung einer Schnittstelle (als kleiner Kreis) nicht unterstützt wird.

23.2. Knoten

Ein Knoten ist zur Laufzeit ein physikalisches Objekt auf dem Komponenten ausgeführt werden können.

Ein Knoten wird in einem Klassendiagramm als dreidimensionaler Kasten dargestellt, der mit seinem Namen beschriftet ist.

23.2.1. Detail-Register Knoten

Die aktiven Detail-Register für Knoten sind die folgenden.

Zu-Bearbeiten-Element
**Eigenschaften**


**Dokumentation**

Standard-Register.

**Darstellung**


---

**Warnung**

Vorsicht, in der Release V0.18 von ArgoUML bezieht sich der umgebende Kasten nur auf den vorderen Teil des Kubus. Das bedeutet, dass die dreidimensionalen Teile ignoriert werden, z.B. wenn die Grenzen eines Diagrammes zum Speichern der Grafik bestimmt werden.

---

**Quelldatei**

Standard-Register, aber ohne Inhalte.

---

**Achtung**

Ein Knoten sollte keinen Code generieren, so dass die Tatsache, dass dieses Register aktiv ist, wahrscheinlich einen Fehler darstellt.

---

**Eigenschaftswerte**

Standard-Register. Im UML-Metamodell sind für Knoten die folgenden Standard-Eigenschaftswerte definiert.

- **persistence** (von der Superklasse Classifier). Der Wert **transitory** gibt an, dass der Zustand zerstört wird, wenn die Instanz zerstört wird, oder **persistent** kennzeichnet, dass der Zustand erhalten bleibt, wenn die Instanz zerstört wird.

- **semantics** (von der Superklasse Classifier). Der Wert ist eine Spezifikation der Semantik des Knotens.

- **derived** (von der Superklasse ModelElement). Der Wert **true** bedeutet, dass der Knoten redundant ist - er kann formal von anderen Elementen abgeleitet werden, oder **false**, wenn er es nicht kann.

---

**Anmerkung**

Abgeleitete Knoten haben ihren Wert in Analyse, um nützliche Namen oder Konzepte einzuführen. Und im Design, um eine Wiederverarbeitung zu verhindern.

---

**Anmerkung**

Die UML-Metaklasse **Element**, von der alle anderen Modellelemente abgeleitet
werden, beinhaltet den Eigenschaftswert documentation, der in ArgoUML im Register Dokumentation bearbeitet wird.

23.2.2. Eigenschaftssymbolleiste Knoten

Nach oben
Navigiert in der Paketstruktur nach oben.

Neues Empfangssignal
Erzeugt ein neues Empfangssignal und springt sofort in das Register Eigenschaften dieses Empfangssignales.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für den markierten Knoten und springt sofort in das Register Eigenschaften dieses Stereotypes.

Löschen
Löscht den Knoten aus dem Modell.

Warnung
Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm. Um eine Klassifizierer Rolle aus dem Diagramm zu löschen, aber im Modell zu erhalten, verwenden Sie Aus Diagramm entfernen im Hauptmenü (oder drücken Sie die Taste Entf).

23.2.3. Eigenschaftsfelder für einen Knoten

Name
Textfeld. Der Name des Knotens. Per Konvention beginnen Knoten mit einem Großbuchstaben und verwenden die Groß-/Kleinschreibung, um Wörter innerhalb des Namens zu unterscheiden.

Anmerkung
ArgoUML erzwingt diese Namenskonvention nicht.

Stereotyp

Stereotyp navigieren
Symbol. Wenn ein Stereotyp markiert wurde, navigiert es zum Eigenschaftsfenster des Stereotypen (siehe Abschnitt 16.6, „Stereotyp“).

Namensraum

Modifizierer
Markierfelder mit den Einträgen Abstract, Leaf und Root.
- Abstract wird verwendet, um zu deklarieren, dass dieser Knoten nicht instanziiert werden kann, aber immer spezialisiert werden muss. Der Name eines abstrakten Knotens wird in kursiver Schrift im Diagramm dargestellt.
- Leaf gibt an, dass dieser Knoten nicht weiter spezialisiert werden kann.
- root gibt an, dass der Knoten keine Generalisierung haben kann.

Generalisierungen
Textbereich. Listet jeden Knoten auf, die diesen Knoten generalisiert.
Ein Taste 1-Doppelklick navigiert zu der Generalisierung und öffnet deren Register Eigenschaften.

Spezialisierungen
Textfeld. Listet alle spezialisierten Knoten auf (z.B. für die dieser Knoten eine Generalisierung ist).
Ein Taste 1-Doppelklick navigiert zu der Spezialisierung und öffnet deren Register Eigenschaften.

Speicherresident
Textfeld. Listet alle Speicherresidenten Elemente auf (siehe Abschnitt 23.4, „Komponente“), die entworfen wurden, um diesen Knotentyp auszuführen.
Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

### 23.3. Knoteninstanz

Eine Knoteninstanz ist der Fall eines Knotens, wo sich Komponenteninstanzen (siehe Abschnitt 23.5, „Komponenteninstanz“) befinden können. Im UML-Metamodell ist NodeInstance eine Subklasse von Instance und ist speziell eine Instanz, die von einem Knoten abgeleitet wurde.


**Tipp**

Die Präsenz des Doppelpunktes (:) und der unterstrichene Name und der Typ unterscheidet eine Knoteninstanz von einem Knoten.

### 23.3.1. Detail-Register Knoteninstanz

Die aktiven Detail-Register für eine Knoteninstanz sind die folgenden.
Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften
Siehe Abschnitt 23.3.2, „Eigenschaftssymbolleiste Konteninstanz“ und Abschnitt 23.3.3, „Eigenschaftsfelder für eine Knoteninstanz“ unten.

Dokumentation
Standard-Register.

Darstellung

Warnung

Quelldatei
Standard-Register, enthält nur den Namen der Knoteninstanz.

Achtung
Eine Knoteninstanz sollte keinen Code generieren, so dass die Tatsache, dass dieses Register aktiv ist, wahrscheinlich einen Fehler darstellt.

Eigenschaftswerte
Standard-Register.

Anmerkung

Checkliste
Standard-Register für eine Instanz.

23.3.2. Eigenschaftssymbolleiste Konteninstanz

Nach oben
Navigiert in der Paketstruktur nach oben.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Knoteninstanz und springt sofort in das Register Eigenschaften dieses Stereotypes.

Löschen
Löscht die Knoteninstanz aus dem Modell.

**Warnung**
Dies ist ein Löschen aus dem Modell, *nicht* nur aus dem Diagramm. Um eine Knoteninstanz aus dem Diagramm zu löschen, aber im Modell zu erhalten, verwenden Sie Aus Diagramm entfernen im Hauptmenü (oder drücken Sie die Taste Entf).

### 23.3.3. Eigenschaftsfelder für eine Knoteninstanz

**Name**
Textfeld. Der Name der Knoteninstanz. Per Konvention beginnt eine Knoteninstanz mit einem Kleinbuchstaben und verwendet die Groß-/Kleinschreibung, um Wörter innerhalb des Namens zu unterscheiden.

**Anmerkung**
ArgoUML erzwingt diese Namenskonvention nicht.

**Stereotyp**

**Stereotyp navigieren**
Symbol. Wenn ein Stereotyp markiert wurde, navigiert es zum Eigenschaftsfenster des Stereotypen (siehe Abschnitt 16.6, „Stereotyp“).

**Namensraum**
Kombinationsfeld. Nimmt den Namensraum für die Knoteninstanz auf. Dies ist die Pakethierarchie.

**Auslöseimpuls gesendet**
(Noch zu beschreiben).

**Auslöseimpuls empfangen**
(Noch zu beschreiben).

**Speicherresident**
Textfeld. Listet alle Speicherresidenten Elemente auf (siehe Abschnitt 23.4, „Komponente“), die entworfen wurden, um diesen Knotentyp auszuführen.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

**Klassifizierer**
Textfeld. Hier kann ein Knoteninstanztyp ausgewählt werden.
Achtung

ArgoUML V0.18 listet viel mehr Elemente in der Auswahlliste auf, als ausschliesslich Knoten. Stellen Sie sicher, dass Sie nur Knoten auswählen.

23.4. Komponente


Eine Komponente wird in einem Klassendiagramm als Kasten mit zwei kleinen Rechtecken, links hervorstehend und mit seinem Namen bezeichnet dargestellt.

23.4.1. Detail-Register Komponente

Die aktiven Detail-Register für Komponenten sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften

Dokumentation
Standard-Register.

Darstellung

Quelldatei
Standard-Register, aber ohne Inhalte.

Achtung

Eine Komponente sollte keinen Code generieren, so dass die Tatsache, dass dieses Register aktiv ist, wahrscheinlich einen Fehler darstellt.

Eigenschaftswerte
Standard-Register.

Anmerkung

Die UML-Metaklasse Element, von der alle anderen Modellelemente abgeleitet werden, beinhaltet den Eigenschaftswert documentation, der im Register
Dokumentation von ArgoUML bearbeitet wird.

23.4.2. Eigenschaftssymbolleiste Komponente

Nach oben
Navigiert in der Paketstruktur nach oben.

Neues Empfangssignal
Erzeugt ein neues Empfangssignal und springt sofort in das Register Eigenschaften dieses Einempfangssignales.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Komponente und springt sofort in das Register Eigenschaften dieses Stereotyps.

Löschen
Löscht die Komponente aus dem Modell.

Warnung
Dies ist ein Löschen aus dem Modell, nicht nur aus dem Diagramm. Um eine Komponente aus dem Diagramm zu löschen, aber im Modell zu erhalten, verwenden Sie Aus Diagramm entfernen im Hauptmenü (oder drücken Sie die Taste Entf).

23.4.3. Eigenschaftsfelder für eine Komponente

Name
Textfeld. Der Name der Komponente. Per Konvention beginnen Komponenten mit einem Großbuchstaben und verwenden die Groß-/Kleinschreibung, um Wörter innerhalb des Namens zu unterscheiden.

Anmerkung
ArgoUML erzwinge diese Namenskonvention nicht.

Stereotyp

Stereotyp navigieren
Symbol. Wenn ein Stereotyp markiert wurde, navigiert es zum Eigenschaftsfenster des
Stereotypen (siehe Abschnitt 16.6, „Stereotyp“).

Namensraum

Modifizierer
Markierfelder mit den Einträgen Abstract, Leaf und Root.
- Abstract wird verwendet, um zu deklarieren, dass diese Komponente nicht instanziert werden kann, aber immer spezialisiert werden muss.
- Leaf gibt an, dass diese Komponente nicht weiter spezialisiert werden kann.
- root gibt an, dass die Komponente keine Generalisierung haben kann.

Generalisierungen
Textbereich. Listet jede Komponente auf, die diese Komponente generalisiert.

Spezialisierungen
Textfeld. Listet alle spezialisierten Komponenten auf (z.B. für die diese Komponente eine Generalisierung ist).

Abhängig von

Notwendig für

Speicherresident
Textfeld. Listet alle Speicherresidenten Elemente auf (siehe Abschnitt 23.4, „Komponente“), die entworfen wurden, um diesen Knotentyp auszuführen.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

23.5. Komponenteninstanz

Eine Komponenteninstanz ist eine Instanz einer Komponente (siehe Abschnitt 23.4, „Komponente“), die sich in einer Knoteninstanz befinden kann (siehe Abschnitt 23.3, „Knoteninstanz“). Im UML-Metamodell ist ComponentInstance eine Subklasse von Instance und eine spezielle Instanz, die von einer Komponente abgeleitet wurde.

Eine Komponente wird in einem Klassendiagramm als Kasten mit zwei schmalen Rechtecken, links hervorgehoben und mit seinem Namen bezeichnet, dargestellt.


**Tipp**

Die Präsenz des Doppelpunktes (:) und der unterstrichene Name und der Typ unterscheidet eine Komponenteninstanz von einer Komponente.
23.5.1. Detail-Register Komponenteninstanz

Die aktiven Detail-Register für eine Komponenteninstanz sind die folgenden.

Zu-Bearbeiten-Element
Standard-Register.

Eigenschaften
Siehe Abschnitt 23.5.2, „Eigenschaftssymbolleiste Komponenteninstanz“ und Abschnitt 23.5.3, „Eigenschaftsfelder für eine Komponenteninstanz“ unten.

Dokumentation
Standard-Register.

Darstellung

Quelldatei
Standard-Register, enthält nur den Namen der Komponenteninstanz.

Achtung

Eine Komponenteninstanz sollte keinen Code generieren, so dass die Tatsache, dass dieses Register aktiv ist, wahrscheinlich einen Fehler darstellt.

Eigenschaftswerte
Standard-Register.

Anmerkung


Checkliste
Standard-Register für eine Instanz.

23.5.2. Eigenschaftssymbolleiste Komponenteninstanz

Nach oben
Navigiert in der Paketstruktur nach oben.

Neuer Stereotyp
Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „Stereotyp“) für die markierte Komponenteninstanz und springt sofort in das Register Eigenschaften dieses Stereotypes.

Löschen
Löscht die Komponenteninstanz aus dem Modell.

**Warnung**

Dies ist ein Löschen aus dem Modell, *nicht* nur aus dem Diagramm. Um eine Komponenteninstanz aus dem Diagramm zu löschen, aber im Modell zu erhalten, verwenden Sie *Aus Diagramm entfernen* im Hauptmenü (oder drücken Sie die Taste Entf).

### 23.5.3. Eigenschaftsfelder für eine Komponenteninstanz

**Name**
Textfeld. Der Name der Komponenteninstanz. Per Konvention beginnt eine Komponenteninstanz mit einem Kleinbuchstaben und verwendet die Groß-/Kleinschreibung, um Wörter innerhalb des Namens zu unterscheiden.

**Anmerkung**
ArgoUML erzwingt diese Namenskonvention nicht.

**Stereotyp**

**Stereotyp navigieren**
Symbol. Wenn ein Stereotyp markiert wurde, navigiert es zum Eigenschaftsfenster des Stereotypen (siehe Abschnitt 16.6, „Stereotyp“).

**Namensraum**
Kombinationsfeld. Nimmt den Namensraum für die Komponenteninstanz auf. Dies ist die Pakethierarchie.

**Auslöseimpuls gesendet**
(Noch zu beschreiben).

**Auslöseimpuls empfangen**
(Noch zu beschreiben).

**Speicherresident**
Textfeld. Listet alle Speicherresidenten Elemente auf (siehe Abschnitt 23.4, „Komponente“), die entworfen wurden, um diese Komponente auszuführen.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

**Klassifizierer**
Textfeld. Hier kann ein Komponenteninstanztyp ausgewählt werden.

**Achtung**
ArgoUML V0.18 listet viel mehr Elemente in der Auswahlliste auf, als ausschließlich
Komponenten. Stellen Sie sicher, dass Sie nur Komponenten auswählen.

23.6. Abhängigkeit

Ein wichtiger Teil eines Komponenten- oder eines Verteilungsdiagrammes ist es, Abhängigkeiten darzustellen. Details entnehmen Sie Abschnitt 18.14, „Abhängigkeit“.

Achtung

UML stützt sich auf das Stereotypisieren von Komponenten-Abhängigkeiten und Verteilungsdiagrammen, um die Beziehungstypen zu charakterisieren. In der aktuellen Release von ArgoUML gibt es Einschränkungen bei der Implementierung von Abhängigkeiten, die diese Funktionalität einschränken.

23.7. Klasse

Ein Komponentendiagramm kann die interne Struktur von Komponenten darstellen, einschließlich der Klassen innerhalb der Komponente. Details entnehmen Sie Abschnitt 18.6, „Klasse“.

Achtung

Klassen können nur zu einem Komponentendiagramm hinzugefügt werden, wenn sie bereits im Modell existieren (durch markieren im Explorer und ausführen von "Zum Diagramm hinzufügen"). Es gibt keinen Weg, eine neue Klasse einem Komponentendiagramm zu erzeugen.

23.8. Schnittstelle

Ein Komponenten- oder Verteilungsdiagramm kann Komponenten oder Komponenteninstanzen darstellen, die Schnittstellen implementieren. Weiter Details entnehmen Sie Abschnitt 18.16, „Schnittstelle“.

Achtung

Die V0.18 Release von ArgoUML verwendet die gleiche Darstellung von Schnittstellen wie im Klassendiagramm. Der UML-Standard empfiehlt, dass eine Schnittstelle in einem Komponenten- oder Verteilungsdiagramm als kleiner offener Kreis dargestellt werden sollte, verbunden mit der Komponente, die diese Schnittstelle realisiert.

Warnung

Es gibt keinen Weg, die Verbindung einer Schnittstelle mit einer Komponente oder Komponenteninstanz in der Release V0.18 von ArgoUML darzustellen.
23.9. Assoziation

Komponenten können miteinander verknüpft werden. Weiter Details über Assoziationen, siehe Abschnitt 18.12, „Assoziation“.

Wo Klassen oder Schnittstellen innerhalb von Komponenten in Komponentendiagrammen dargestellt werden, können Sie als verbundene Assoziationen dargestellt werden.

23.10. Objekt

Wie Komponenten die Klassifizierer darstellen können, die deren interne Struktur ausmachen, können Komponenteninstanzen in Verteilungsdiagrammen die Klassifiziererinstanzen darstellen, die deren interne Struktur ausmachen. In der Praxis ist die einzige Instanz die verwendet wird ein Objekt (eine Instanz einer Klasse). Details siehe Abschnitt 19.2, „Objekt“.

23.11. Verbindung

Wo Objekte (Knoteninstanzen oder Klasseninstanzen) innerhalb von Komponenteninstanzen in Verteilungsdiagrammen dargestellt werden, können ihre Beziehungen zueinander als Verbindungen (Instanzen einer Assoziation) dargestellt werden. Siehe Abschnitt 19.9, „Verknüpfung“ für weitere Details.
Kapitel 24. Profile

Achtung

Dieser Abschnitt wurde "Eingebaute Datentypen, Klassen, Schnittstellen und Stereotypen" genannt und wurde seit der Einführung von Profilen nicht vollständig auf den neuesten Stand gebracht.

24.1. Einleitung

Dieses Kapitel beschreibt ArgoUML's eingebaute Profile, die Datentypen, Aufzählungen, Klassen, Schnittstellen, Stereotypen, Tag-Definitionen und Kritiken enthalten.

Datentypen, Aufzählungen, Klassen und Schnittstellen sind allgemein verfügbar und können überall genutzt werden, wo eine Klasse im Register Eigenschaften ausgewählt wurde.

Stereotypen und Tag-Definitionen werden häufig für applikationsspezifische Erweiterungen des Modell verwendet, z.B., um Elemente zu unterscheiden, die unterschiedlich implementiert werden müssen.

Kritiken wurden in Profile eingebunden, um geschützte Randbedingungen in Profilen der Zielapplikation zu unterstützen.

Die Modellelemente in einem UML-Profil sind als Hierarchie unterhalb des Model organisiert, sie sind aber nicht Teil Ihres Projektes. Aus diesem Grund werden sie separat geladen und müssen, einmal verwendet, vorhanden sein, damit das Projekt geöffnet werden kann.

Abbildung 24.1. Hierarchie der Standardprofile innerhalb von ArgoUML

Nach den eingebauten Profilen macht es ArgoUML möglich, eigene Profile zu erzeugen oder Profile Dritter zu verwenden.

24.2. Das Profil UML 1.4 Standard-Elemente
### 24.2.1. Datatypen

Dies sind die UML-Datentypen. Ihre Definition entnehmen Sie bitte dem UML-Standard.

- **Integer**
- **String**
- **UnlimitedInteger**

### 24.2.2. Aufzählungen

Dies sind die UML-Aufzählungen. Ihre Definition entnehmen Sie bitte dem UML-Standard.

- **Boolean**

### 24.2.3. Stereotypen

UML 1.4 definiert eine grosse Anzahl von Stereotypen, die alle von ArgoUML unterstützt werden. Die unten stehende Tabelle listet all diese Stereotypen auf.

Der UML 1.4-Standard spezifiziert auch viele Stereotypen in den Kapiteln „Example Profiles“: einen für „Software Development“ und einen für „Business Modeling“. Auf Grund des speziellen Charakters dieser Profile, wurde die Implementierung in ArgoUML auf einen späteren Zeitpunkt verschoben.

#### Tabelle 24.1. UML 1.4 und ArgoUML definierte Stereotypen

<table>
<thead>
<tr>
<th>Stereotyp</th>
<th>Basiselement</th>
</tr>
</thead>
<tbody>
<tr>
<td>access</td>
<td>Permission</td>
</tr>
<tr>
<td>appliedProfile</td>
<td>Package</td>
</tr>
<tr>
<td>association</td>
<td>AssociationEnd</td>
</tr>
<tr>
<td>auxiliary</td>
<td>Class</td>
</tr>
<tr>
<td>become</td>
<td>Flow</td>
</tr>
<tr>
<td>call</td>
<td>Usage</td>
</tr>
<tr>
<td>copy</td>
<td>Flow</td>
</tr>
<tr>
<td>create</td>
<td>BehavioralFeature</td>
</tr>
<tr>
<td>create</td>
<td>CallEvent</td>
</tr>
<tr>
<td>create</td>
<td>Usage</td>
</tr>
<tr>
<td>Stereotyp</td>
<td>Basiselement</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>derive</td>
<td>Abstraction</td>
</tr>
<tr>
<td>destroy</td>
<td>BehavioralFeature</td>
</tr>
<tr>
<td>destroy</td>
<td>CallEvent</td>
</tr>
<tr>
<td>document</td>
<td>Abstraction</td>
</tr>
<tr>
<td>executable</td>
<td>Abstraction</td>
</tr>
<tr>
<td>facade</td>
<td>Package</td>
</tr>
<tr>
<td>file</td>
<td>Abstraction</td>
</tr>
<tr>
<td>focus</td>
<td>Class</td>
</tr>
<tr>
<td>framework</td>
<td>Package</td>
</tr>
<tr>
<td>friend</td>
<td>Permission</td>
</tr>
<tr>
<td>global</td>
<td>AssociationEnd</td>
</tr>
<tr>
<td>implementation</td>
<td>Class</td>
</tr>
<tr>
<td>implementation</td>
<td>Generalization</td>
</tr>
<tr>
<td>implicit</td>
<td>Association</td>
</tr>
<tr>
<td>import</td>
<td>Permission</td>
</tr>
<tr>
<td>instantiate</td>
<td>Usage</td>
</tr>
<tr>
<td>invariant</td>
<td>Constraint</td>
</tr>
<tr>
<td>library</td>
<td>Abstraction</td>
</tr>
<tr>
<td>local</td>
<td>AssociationEnd</td>
</tr>
<tr>
<td>metaclass</td>
<td>Class</td>
</tr>
<tr>
<td>metamodel</td>
<td>Package</td>
</tr>
<tr>
<td>modelLibrary</td>
<td>Package</td>
</tr>
<tr>
<td>parameter</td>
<td>AssociationEnd</td>
</tr>
<tr>
<td>postcondition</td>
<td>Constraint</td>
</tr>
<tr>
<td><strong>Stereotyp</strong></td>
<td><strong>Basiselement</strong></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>powertype</td>
<td>Class</td>
</tr>
<tr>
<td>precondition</td>
<td>Constraint</td>
</tr>
<tr>
<td>process</td>
<td>Classifier</td>
</tr>
<tr>
<td>profile</td>
<td>Package</td>
</tr>
<tr>
<td>realize</td>
<td>Abstraction</td>
</tr>
<tr>
<td>refine</td>
<td>Abstraction</td>
</tr>
<tr>
<td>requirement</td>
<td>Comment</td>
</tr>
<tr>
<td>responsibility</td>
<td>Comment</td>
</tr>
<tr>
<td>self</td>
<td>AssociationEnd</td>
</tr>
<tr>
<td>send</td>
<td>Usage</td>
</tr>
<tr>
<td>signalflow</td>
<td>ObjectFlowState</td>
</tr>
<tr>
<td>source</td>
<td>Abstraction</td>
</tr>
<tr>
<td>stateInvariant</td>
<td>Constraint</td>
</tr>
<tr>
<td>stub</td>
<td>Package</td>
</tr>
<tr>
<td>systemModel</td>
<td>Package</td>
</tr>
<tr>
<td>table</td>
<td>Abstraction</td>
</tr>
<tr>
<td>thread</td>
<td>Classifier</td>
</tr>
<tr>
<td>topLevel</td>
<td>Package</td>
</tr>
<tr>
<td>trace</td>
<td>Abstraction</td>
</tr>
<tr>
<td>type</td>
<td>Class</td>
</tr>
</tbody>
</table>

### 24.2.4. Tag-Definitionen


- derived
24.3. Das Javaprofil

Das Javaprofil enthält Elemente, die kein Mandat des UML-Standards haben und nur in einer Entwicklungsumgebung verwendet werden sollten, die Java verwendet.


Abbildung 24.2. Hierarchie des Javaprofils in ArgoUML
24.3.1. Datentypen

Dies sind eingebaute atomare Typen. Sie können diese ändern, wenn Sie wollen. Jedoch ist dies keine gute Praxis.

All dies finden Sie im `java.lang`-Subpaket des Hauptmodelles.

Dies sind die Standard-Datentypen. Ihre Definition entnehmen Sie bitte dem UML-Standard.

- boolean
- byte
- char
- double
- float
- int
- long
- short
- void

Anmerkung

Bei void spricht man nicht von einem Typ, sondern von der Abwesenheit des Typs. ArgoUML kennt void und erlaubt es als Option, wo Datentypen ausgewählt werden müssen.

24.3.2. Klassen

Dies sind die gemeinsamen Klassen, die den Klassen entsprechen, die in der Standard-Java-Umgebung definiert sind. Es liegt an Ihnen, wenn Sie diese ändern wollen.

Diese finden sich in allen vier Subpaketen des `java`-Subpaketes wieder.

Die Definition dieser Klassen entnehmen Sie bitte den Java Sprach- und Bibliotheks-Definitionen.

24.3.2.1. Klassen aus `java.lang`

Dies sind die Klassen innerhalb des `java.lang`-Paketes.

- Boolean
- Byte
- Char
- Double
• Float
• Integer
• Long
• Object
• Short
• String

24.3.2.2. Klassen aus java.math
Dies sind die Klassen innerhalb des java.math-Paketes.

• BigDecimal
• BigInteger

24.3.2.3. Klassen aus java.net
Dies sind die Klassen innerhalb des java.net-Paketes.

• URL

24.3.2.4. Klassen aus java.util
Dies sind die Klassen innerhalb des java.util-Paketes.

• Date
• Time
• Vector

24.3.3. Interfaces
Dieses sind einige nützliche Schnittstellen, die Klassen innerhalb der Java-Umgebung entsprechen. Schnittstellen haben viele Eigenschaften von Klassen (wie alle Typen) und Sie können diese ändern, wenn Sie dies wünschen.

All dies finden Sie im java.util-Subpaket des Hauptmodelles.

Dies sind die im Paket java.util definierten Schnittstellen. Deren Definition entnehmen Sie bitte der Java Sprach- und Bibliotheks-Referenz.

• Collection
• Iterator
• List
• Set
• SortedSet
## Glossar

### A

<table>
<thead>
<tr>
<th>Begriff</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aktivitätsdiagramm</td>
<td>Ein UML-Diagramm, welches das dynamische Verhalten eines Systems oder Subsystems beschreibt. Weitere Informationen, siehe Abschnitt 6.10, „Aktivitätsdiagramme (Noch zu beschreiben)“.</td>
</tr>
<tr>
<td>Akteur</td>
<td>Die Darstellung eines Agenten (animiert oder nicht animiert) in einem Anwendungsfalldiagramm, der ausserhalb des zu entwerfenden Systems steht.</td>
</tr>
<tr>
<td>Analyse</td>
<td>Analyse ist der Prozess die „Kunden“-Anforderungen aufzunehmen und diese aus der Perspektive der mutmaßlichen Lösung in die Sprache der mutmaßlichen Lösung umzuwandeln.</td>
</tr>
<tr>
<td>Anforderungserfassung</td>
<td>Die Anforderungserfassung ist der Prozess des Identifizierens was der „Kunde“ von dem gewünschten System wünscht. Eine ausführlichere Beschreibung, siehe Kapitel 4, Erlassen der Anforderungen.</td>
</tr>
<tr>
<td>Anwendungsfall</td>
<td>Eine UML-Notation, welche die Anforderungen an ein System oder Subsystem beinhaltet. Weitere Informationen, siehe Abschnitt 4.3, „Ergebnis des Anforderungs-Erfassungs-Prozesses“.</td>
</tr>
<tr>
<td>Anwendungsfalldiagramm</td>
<td>Ein UML-Diagramm, welches die Beziehungen zwischen Akteuren und Anwendungsfällen darstellt. Weitere Informationen, siehe Abschnitt 4.3, „Ergebnis des Anforderungs-Erfassungs-Prozesses“.</td>
</tr>
<tr>
<td>Anwendungsfallspezifikation</td>
<td>Das Dokument beinhaltet die detaillierten Anforderungen eines Anwendungsfalles.</td>
</tr>
<tr>
<td>Assoziationsklasse</td>
<td>Eine Klasse, die eine Assoziation zwischen zwei anderen Klassen charakterisiert.</td>
</tr>
<tr>
<td>Assoziation</td>
<td>Eine Beziehung zwischen zwei Klassen in einem Klassendiagramm oder zwischen Anwendungsfällen oder Anwendungsfällen und Akteuren in einem Anwendungsfalldiagramm.</td>
</tr>
<tr>
<td>Attribut (einer Klasse oder eines Objektes)</td>
<td>Ein Attribut einer Klasse oder eines Objektes ist eine Spezifikation eines Datenelementes, gekapselt durch dieses Objekt.</td>
</tr>
</tbody>
</table>
**CASE**

Computer Aided Software Engineering (Computergestützte Softwareentwicklung).

**Ergänzende Anforderungsspezifikation**

Das Dokument beinhaltet nicht-funktionale Anforderungen, die nicht mit einem Anwendungsfall verknüpft werden können.

**Extend-Beziehung**

Eine Beziehung zwischen zwei Anwendungsfällen, wo der erweiterte Anwendungsfall eine spezielle Variante des zu erweiternden Anwendungsfalles beschreibt.

**Fenster**

Ein Subfenster im Hauptfenster der ArgoUML-Anwenderschnittstelle.

**Generalisierungs-Beziehung**

Eine Beziehung zwischen einem generalisierenden Anwendungsfall und einem oder mehreren generalisierten Anwendungsfällen, bei dem der generalisierte Anwendungsfall besondere Beispiele des generalisierten Anwendungsfalles sind.

**GUI**

Graphical User Interface (Grafische Anwenderschnittstelle).

**Hierarchisches Zustandsdiagramm**

Ein Zustandsdiagramm, das untergeordnete Zustandsdiagramme mit individuellen Zuständen enthält.

**Include-Beziehung**

Eine Beziehung zwischen zwei Anwendungsfällen, wo der eingefügte Anwendungsfall Teile der Funktionalität des einfügenden Anwendungsfalles beschreibt.
Iterativer Designprozess

Ein Designprozess, in dem alle Phasen (Anforderungen, Analyse, Design, Build, Test) partiell in einer Abfolge von Iterationen durchlaufen werden. Mehr Informationen, siehe Abschnitt 3.2.1, „Prozessarten“. 

Java


Die wichtigsten Komponenten von Java waren die Integration der JVM in Webbrowsern, die es erlaubt, Code (Applets) herunterzuladen und über das Web auszuführen.

ArgoUML ist in Java geschrieben.

Klasse

Die Kapselung von Daten verbunden mit einem Modellelement (seine Attribute) und die Aktionen, die mit dem Modell verbunden sind (seine Methoden).

Eine Klasse spezifiziert die Charakteristika eines Modell. Ein Objekt repräsentiert eine Instanz eines Modells.

Klassen und Objekte werden in UML in Aktivitätsdiagrammen, Klassendiagrammen, Kollaborationsdiagrammen und Sequenzdiagrammen dargestellt.

Klassendiagramm

Ein UML-Diagramm, das die strukturelle Beziehung zwischen Klassen darstellt. Weitere Informationen, siehe Abschnitt 5.2, „Klassendiagramme (Noch zu beschreiben)“. 

Kollaboration

Ein Prozess, in dem mehrere Objekte kooperieren, um ein höherwertiges Verhalten anzubieten, das umfangreicher ist als die Summe der Verhaltensweisen von Objekten.

Kollaborationsdiagramm

Ein UML-Diagramm, welches das dynamische Verhalten darstellt, wie Botschaften die zwischen Objekten ausgetauscht werden. Ähnlich einem Sequenzdiagramm. Dessen Darstellung ist ähnlich, abhängig vom betrachteten Problem.

Kollaborator

Ein Objekt das an einer Kollaboration teilnimmt.

Konzept Klassendiagramm
Ein, während der Analysephase konstruiertes Klassendiagramm zeigt die strukturellen Hauptkomponenten des Problems, das in der Anforderungsphase identifiziert wurde. Weitere Informationen, siehe Kapitel 5, Analyse.

Kritik


|   | Methode (einer Klasse oder eines Objektes) | Eine Methode einer Klasse oder eines Objektes ist eine Spezifikation des Verhaltens, gekapselt durch das Objekt. |

| O | Objekt | Eine Instanz einer *Klasse*. |
|   | Klassen und Objekte werden in der UML in *Aktivitätsdiagrammen, Klassendiagrammen*, *Kollaborationsdiagrammen* und *Sequenzdiagrammen* dargestellt. |

| OCL | Object Constraint Language. Eine Sprache zur Beschreibung von Bedingungen in der UML. |


|       | UML ist eine Notation, welche die OOA&D unterstützt. |

**Oppportunistisches Design**
Eine Theorie aus der kognitiven Psychologie, die aussagt, dass obwohl Designer ihre Arbeit auf eine geordnete, hierarchische Art planen und beschreiben, sie aufeinanderfolgende Tätigkeiten auf Basis der Kriterien der kognitiven Kosten auswählen. Einfach gesagt, Designer folgen nicht ihren eigenen Plänen und Reihenfolgen, sondern wählen ihre Schritte danach, welche davon mental weniger aufwändig sind im Gegensatz zu den Alternativen.

**Realisation Anwendungsfall**
Ein *Anwendungsfall* bei dem das *Anwendungsfalldiagramm* und die *Anwendungsfallspezifikation* in der Sprache des Lösungsbereiches und nicht in der Sprache des Problembereiches beschrieben wurde.

**Reflektion-während-Aktion**

**Szenario**
Eine spezifische Abfolge von Aktionen, die das Verhalten illustriert.

**Sequenzdiagramm**
Ein UML-Diagramm, welches das dynamische Verhalten darstellt, wie Botschaften, die zwischen Objekten ausgetauscht werden. Ähnlich dem *Kollaborationsdiagramm*. Welche Darstellung angemessen ist, hängt von dem betrachteten Problem ab. Weitere Informationen, siehe Abschnitt 5.4, „Sequenzdiagramme (Noch zu beschreiben)“.

**SGML**

**Simula 67**

**Stereotypen und Stereotypisieren**
In UML kann jedem Modellelement ein *Stereotyp* gegeben werden, um seine Beziehung mit einer bestimmten Rolle im Design darzustellen. Ein Stereotyp *spqr* wird allgemein in der Notation `<<spqr>>` dargestellt.

**SVG**


**Systemsequenzdiagramm**


**Systemzustandsdiagramm**

Ein Zustandsdiagramm zeigt in der Analysephase das dynamische Verhalten eines aktiven Systemobjektes der obersten Ebene. Weitere Informationen, siehe Kapitel 5, Analyse.

**T**

**To-Do Liste**

Eine Eigenschaft von ArgoUML, die es dem Anwender erlaubt, Aktivitäten aufzunehmen, die noch vervollständigt werden müssen.

**Transition**

Die Änderung zwischen Zuständen in einem Zustandsdiagramm.

**U**

**UML**


**V**

**Verantwortlichkeit**

Ein Verhalten, für das ein Objekt verantwortlich ist. Eine Verantwortlichkeit bezeichnet die Verpflichtung eines Objektes, ein bestimmtes Verhalten sicherzustellen.

**Verstehen und Problemlösung**

Eine Designvisualisierungstheorie innerhalb der kognitiven Psychologie. Die Theorie besagt, dass Designer den Schritt zwischen Ihrem mentalen Modell und dem Problem oder der
Situation und dem formalen Modell der Lösung oder des Systems überbrücken müssen.

Diese Theorie besagt, dass Programmierer einen Vorteil haben von:


2. Vertraute Aspekte eines Situationsmodelles, verbessert die Fähigkeiten des Designers, die Lösung zu formulieren.

**Visions-Dokument**
Das Toplevel-Dokument, das beschreibt, was das zu entwickelnde System leisten können soll.

**W**

**W3C**

**Wasserfall-Designprozess**
Ein Designprozess, bei dem jede Phase (Anforderungen, Analyse, Design, Build, Test) vollständig bearbeitet wird, bevor die nächste beginnt. Weitere Informationen, siehe Abschnitt 3.2.1, „Prozesstypen“.

**X**

**XMI**
XML-Model Interchange format. Ein Format für die Speicherung von UML-Modellen in einer Datei. Aktuell unvollständig, weil sie nicht alle grafischen Layoutinformationen enthält, so dass sie durch Dateien ergänzt werden muss, die diese Informationen enthalten.

**XML**
eXtensible Markup Language. Eine, durch die W3C vereinfachte Ableitung von SGML.

**Z**

**Zustand**
Innerhalb eines Zustandsdiagrammes eine der möglichen
Konfigurationen des Automaten.

Zustandsdiagramm

Ein UML-Diagramm, welches das dynamische Verhalten eines aktiven Objektes darstellt. Weitere Informationen, siehe Abschnitt 5.6, „Zustandsdiagramme (Noch zu beschreiben)“.
Anhang A. Ergänzendes Material für die Fallstudie

A.1. Einleitung

Die Fallstudie erfordert verschiedene Materialien (meist Dokumente), die neben dem Designdiagramm existieren.

A.2. Anforderungsdokumente (Noch zu beschreiben)

A.2.1. Visionsdokument (Noch zu beschreiben)

A.2.2. Anwendungsfall-Spezifikationen (Noch zu beschreiben)

A.2.2.1. UC Spezifikation 1 (Noch zu beschreiben)

A.2.3. Ergänzende Anforderungsspezifikation (Noch zu beschreiben)
Anhang B. UML-Ressourcen

B.1. Die UML-Spezifikationen (Noch zu beschreiben)
   Noch zu beschreiben...

B.2. UML-Papiere (Noch zu beschreiben)
   Noch zu beschreiben...

B.2.1. UML-Aktionsspezifikationen (Noch zu beschreiben)
   Noch zu beschreiben...

B.3. UML-Webseiten (Noch zu beschreiben)
   Noch zu beschreiben...
Anhang C. UML-konforme CASE Tools

C.1. Andere Open Source Projekte (Noch zu beschreiben)
   Noch zu beschreiben...

C.2. Kommerzielle Werkzeuge (Noch zu beschreiben)
   Noch zu beschreiben...
Anhang D. Das C++ Modul

Das ArgoUML C++ Modul (C++-Modul) enthält Funktionalitäten zur C++-Codegenerierung und die C++-Notation. Es arbeitet auf die gleiche Weise wie die anderen Sprachmodule.

D.1. Modellieren für C++

Die C++-Programmiersprache hat Konstrukte, die in UML nicht standardmäßig enthalten sind. Beispiele sind Zeiger, globale Funktionen und Variablen, Referenzen und das überladen von Operatoren. Um uns in die Lage zu versetzen, diese Konstrukte in unseren Modellen anwenden zu können und in der Lage zu sein, die Vorteile dieser Konstrukte bei der Codegeneration und C++-Notation in UML-Diagrammen einsetzen zu können, benutzen die C++-Module die Erweiterungseigenschaften von UML, wie Eigenschaftswerte, Stereotypen und Datentypen.


Das C++-Modul enthält ein UML-Profil für C++, das Stereotypen und Eigenschaftswerte definiert, die das Modellieren von C++-spezifischen Konstrukten wie Zeiger und Referenzen ermöglichen. Es enthält auch Datentypen, die die eingebauten C++-Typen wie unsigned long int modellieren.

Um diese Konstrukte in unserem Modell verfügbar zu haben, müssen wir das UML-Profil für C++ im Modell ausdrücklich über den Dialog Projekteigenschaften im Register Profile konfigurieren können (siehe Abschnitt 10.3.14, „Projekteinstellungen...“).

Eigenschaftswerte ist eines der wichtigsten Mittel, über die wir das Verhalten bei der Codegenerierung definieren können. Sie haben einen Namen - die Eigenschaft - und einen Wert, und sind einem Modellelement zugewiesen. Für jedes der möglichen Eigenschaftswerte enthält das C++-Profil eine Eigenschaftswert-Definition, die in einem Stereotypen enthalten ist und auf das Modellelement angewendet werden kann, zu denen wir den Wert eines Eigenschaftswertes definieren wollen, um ein spezielles Verhalten zu spezifizieren. Um, zum Beispiel, zu definieren, dass der Parameter x eine Referenz ist, nehmen Sie den Stereotypen cppParameter und fügen diesem den Eigenschaftswert reference mit dem Wert true hinzu.

Die Eigenschaftswerte, die im C++-Modul verwendet werden, sind in zwei Kategorien unterteilt:

- Werte in einem freien Format - jeder String ist gültig, ausser dem leeren String.
- Formatierte Werte - der Wert muss einigen Beschränkungen gehorchen, z.B. ein Wert aus true oder false sein (abgekürzt: true || false).

Bei Eigenschaftswerten vom Typ Boolean sind nur die Werte "true" or "false" anwendbar. Wenn ein Eigenschaftswert vom Typ Boolean nicht existiert oder für ein Modellelement ungültig ist, wird vom Codegenerator ein Standardwert angenommen. In der nachfolgenden Dokumentation ist der Standardwert jeweils markiert.

Eigenschaftswerte im freien Format sind nur signifikant, wenn sie vorhanden sind und wenn der Wert kein leerer String ist. Wenn der Wert einer Sortierung folgen muss, wird die ausdrücklich erwähnt. In diesem Fall gibt es eine Chance, dass der Wert ungültig ist. Wenn der Wert ungültig ist, wird keine
Annahme getroffen; der Generator wird das Problem verfolgen und den Eigenschaftswert ignorieren.

**D.1.1. Eigenschaftswerte für eine Klasse**

Um eine Eigenschaftswert-Definitionen für eine C++-Klasse zur Verfügung zu stellen, wenden wird den Stereotypen `cppClass` auf die Klasse an.

**Konstruktor**
- `true` - generiert einen Standardkonstruktor für die Klasse.
- `false` (Standard) - es wird kein Standardkonstruktor generiert, es sei denn, er wurde ausdrücklich mit dem Stereotypen «create» modelliert.

**header_incl**
Name der Datei, die in den Kopf eingefügt werden soll.

**Anmerkung**
Wenn wir mehrere Header auf diesem Wege einfügen wollen, verwenden Sie mehrere Eigenschaftswerte mit der Bezeichnung (Tag) `header_incl`.

Andere, für die C++-Modellierung verwendete Eigenschaftswerte können auch auf diesem Weg verwendet werden. Dieser Hinweis wird in diesen Fällen nicht wiederholt werden.

**source_incl**
Der Name der Datei, die in die Quelldatei (.cpp -Datei) eingefügt werden soll.

**typedef_public**
- `<source_type><type_name>` - erzeugt eine typedef-Zeile im Public-Bereich der Klasse mit `typedef <source_type><type_name>`.

**typedef_protected**
Wie bei `typedef_public`, aber im `protected`-Bereich.

**typedef_private**
Wie bei `typedef_public`, aber im `private`-Bereich.

**typedef_global_header**
Wie bei `typedef_public`, aber im globalen Bereich es Headers (Kopfes).

**typedef_global_source**
Wie bei `typedef_global_source`, aber in der Quelldatei.

**TemplatePath**

**email**
- `name@domain.country` - ersetzt das Tag `[EMAIL]` der Templatedatei.

**author**
- `name` - ersetzt das Tag `[AUTHOR]` der Templatedatei.
Anmerkung
Sie können einfach die Autor-Eigenschaft im Register Dokumentation verwenden.

D.1.2. Eigenschaftswerte für Attribute

UML-Attribute werden auf die Membervariablen der Klassen zugeordnet. Um die Tag-Definitionen für die C++-Membervariablen verfügbar zu machen, verwenden wird den Stereotypen cppAttribute.

pointer
true - der Typ der Membervariable wird ein Zeiger auf den Attributtyp sein.

Zum Beispiel: Wenn Sie das UML-Attribut: name:std::string mit dem Eigenschaftswert pointer auf true haben, ist die generierte Membervariable: std::string*name;

false (Standard) - es wird kein Zeiger angewendet.

reference
true - der Typ der Membervariable wird eine Referenz auf den Attributtyp sein.

false (Standard) - es wird keine Referenz angewendet.

usage

MultiplicityType
list || slist || vector || map || stack || stringmap - wird eine Kardinalität als korrespondierenden STL-Container definieren, wenn der Kardinalitätsbereich des Attributes variabel ist (bei festen Bereichen wird die Einstellung ignoriert).

set
private || protected || public - erzeugt eine einfache Funktion, um das Attribut durch eine Funktion zu setzen (Aufruf per Referenz wird für Klassentypen anstelle von Aufruf per Wert verwendet); plaziert die Funktion in den angegebenen Sichtbarkeitsbereich.

get
private || protected || public - wie bei set.

D.1.3. Parameter

Um eine Tag-Definitionen für ein C++-Argument verfügbar zu machen, wenden wir den Stereotyp cppParameter an ihn an.

D.1.3.1. Übergabesemantik Variable

Wenn ein Parameter für eine Operation als out oder inout gekennzeichnet ist, wird die Variable per Referenz (Standard) oder Zeiger (benötigt den Eigenschaftswert pointer - siehe oben) übergeben. Im anderen Fall per Wert.

Rückgabewerte sind in UML einfach Parameter, gekennzeichnet mit return. Also alles, was hier...
für sie gilt, sofern nicht ausdrücklich darauf hingewiesen.

Warnung

Beachten Sie, dass UML mehrere Rückgabewerte erlaubt. Dies ist möglich, und wird in C++ durch out-Parameter unterstützt, aber, es wird derzeit vom Generator nicht unterstützt.


D.1.3.2. Eigenschaftswerte für Parameter

pointer
true || false (Standard) - das gleiche wie bei Attribute.

reference
dto.

D.1.4. Generalisierung

Um die Tag-Definition für eine C++-Generalisierung zur Verfügung zu stellen, wenn wir den Stereotypen cppGeneralization an.

D.1.4.1. Eigenschaftswerte für Generalisierung

cpp_virtual_inheritance
true || false (Standard) - wird zur Spezifikation der virtuellen Vererbung verwendet.

cpp_inheritance_visibility
public (Standard) || private || protected - wird verwendet, um die Sichtbarkeit der Vererbung zu spezifizieren.

D.1.5. Realisierung

Um die Tag-Definition für die C++-Realisierung zur Verfügung zu stellen, wenden wir den Stereotypen cppRealization an.

D.1.5.1. Eigenschaftswerte für Realisierung

cpp_inheritance_visibility
public (Standard) || private || protected - wird verwendet, um die Sichtbarkeit der Vererbung zu spezifizieren.

D.1.6. Geschützte Abschnitte

Bei jeder Codegenerierung werden spezielle Kommentare vor und nach den Funktionsdefinitionen generiert:

389

Die passiert auch, wenn Sie die Methodennamen nach dem Generieren ändern.

```cpp
void newOperation(std::string test = "fddsaffa")
    // section 603522:ec4c7ff768:-7ffc begin
    {
    }
    // section 603522:ec4c7ff768:-7ffc end
```

Wenn Sie eine Operation im Modell löschen. Da nächste Mal, wenn die Klasse generiert wird, wird der gelöschte Code - z.B. die ganze Funktionsdefinition - als Kommentar am Ende der Datei hinzugefügt.
Anhang E. Beschränkungen und Mängel


E.1. Größe der Diagramm-Leinwand

Auf Grund der zugrunde liegenden Grafiksoftware, ist die Leinwandgröße für Diagramme auf 6000 Einheiten in der Höhe und in der Breite begrenzt.

E.2. Fehlende Funktionen
Anhang F. Open Publication Lizenz

This Appendix contains the applied version of the license of this User Manual i.e. the Open Publication License v1.0 with. The latest version is presently available at http://www.opencontent.org/openpub/ [http://www.opencontent.org/openpub/].

Dieser Anhang enthält die Lizenzversion, die bei diesem Anwenderhandbuch zur Anwendung kommt; z.B. Open Publication Lizenz v1.0

F.1. Einleitung

Arbeiten auf Basis der Open Publication Lizenz dürfen als Ganzes oder in Teilen auf jedem physikalischen oder elektronischen Medium vervielfältigt oder verteilt werden, vorausgesetzt, die Bedingungen dieser Lizenz werden eingehalten und das diese Lizenz oder ein Verweis auf diese Lizenz (mit jeder, von den Autor(en) und/oder Herausgebern gewählten Option) in die Reproduktion aufgenommen wird.

Die richtige Form für die Aufnahme von Informationen mittels Verweis lautet wie folgt:


Die kommerzielle Weiterverbreitung von Open Publication lizensiertem Material ist erlaubt.


F.2. Copyright

Das Copyright jeder Open Publication gehört dem Autor(en) oder den Zeichnungsberechtigten.

F.3. Bezugsbereich der Lizenz

Die folgenden Lizenzbestimmungen gelten für alle Open-Publication-Arbeiten, sofern nicht explizit anders lautend im Dokument erwähnt.


SALVATORISCHE KLAUSEL. Wenn irgendein Teil dieser Lizenz in der Rechtssprechung eines Staates nicht vollstreckbar ist, bleiben die restlichen Teile der Lizenz in Kraft.

KEINE GARANTIE. Open-Publication-Arbeiten werden ohne jegliche Garantie lizensiert und verbreitet. Sie sind weder ausdrücklich noch stillschweigend, einschließlich, aber nicht beschränkt auf die stillschweigende Garantie der Markttauglichkeit und der Eignung für einen bestimmten Zweck oder
F.4. Anforderungen an modifizierte Arbeiten

Alle modifizierten Versionen, die durch diese Lizenz abgedeckt werden, einschließlich Übersetzungen, Anthologien, Zusammenstellungen und Teildokumenten, müssen die folgenden Voraussetzungen erfüllen:

1. Die modifizierte Version muss als solche gekennzeichnet werden.
2. Die Person, die die Änderung vornahm muss identifiziert und die Änderungen müssen mit einem Datum versehen sein.
4. Der Ablageort des originalen unmodifizierten Dokumentes muss benannt werden.
5. Der ursprüngliche Name des Autors (oder der Autoren) darf ohne dessen Billigung in dem modifizierten Dokument nicht verwendet werden.

F.5. Empfehlungen für die gute Praxis

Zusätzlich zu den Anforderungen dieser Lizenz, wird den Distributoren dringend empfohlen, dass:

1. Wenn Sie Open-Publication-Arbeiten als Hardcopy oder auf CD-ROM, sollten Sie mindestens dreißig Tage bevor Ihr Manuskript oder das Medium endgültig fertiggestellt ist, eine E-Mail-Benachrichtigung an die Autoren senden, in der Sie Ihre Absicht der Weiterverteilung ankündigen, um den Autoren Zeit zu geben, die aktualisierten Dokumente zu sichten. Diese Ankündigung sollte die im Dokument vorgenommenen Änderungen beschreiben.
2. Alle substantiellen Modifikationen (einschließlich Löschungen) sind entweder deutlich im Dokument oder in einem Anhang zu dem Dokument zu beschreiben.
Anhang G. Die CRC-Karten Methode

Eine CRC-Karte ist vordergründig eine Indexkarte, die verwendet wird, um Klassen, deren Verantwortlichkeiten und die Interaktionen zwischen Ihnen darzustellen. Der Begriff CRC-Karte wird auch benutzt, um sich auf eine Methode für die objektorientierte Modellierung auf der Basis ihrer Verwendung zu beziehen.


Warum CRC-Karten verwenden?

• Sie sind portabel. Es sind keine Rechner erforderlich, so dass sie überall verwendet werden können. Auch ausserhalb des Büros.
• Sie ermöglichen es den Teilnehmern aus erster Hand zu erfahren, wie das System funktioniert. Kein Computer-Tool kann die Interaktion ersetzen, die durch das physische abholen der Karten und das Spielen der Objektrolle geschieht.
• Sie sind nützliche Werkzeuge, um Personen im objektorientierten Paradigma zu unterrichten.
• Sie kann als eine Methode sich selbst oder als Front-End zu einem formellen Verfahren, wie Booch, Wirfs-Brock, Jacobson, usw. verwendet werden. Obwohl CRC-Karten für den Unterricht erstellt wurden, haben sie sich für vieles mehr nützlich erwiesen.

G.1. Die Karte

Das genaue Format der Karte kann an die Präferenzen der Gruppe angepasst werden, aber die minimal erforderlichen Informationen ist der Name der Klasse, die Unterklassen und Oberklassen, seine Verantwortlichkeiten und die Kollaboratoren für jede dieser Verantwortlichkeiten. Die Rückseite der Karte kann für eine Beschreibung der Klasse verwendet werden. Während der Design-Phase können die Attribute der Klasse auf der Rückseite wiedergegeben werden. Eine Möglichkeit die Karte zu interpretieren ist, dass die Vorderseite die öffentlichen Informationen, und die Rückseite die gekapselten, Implementierungsdetails enthält. Wenn eine eine Klasse definiert wird, wird eine Karte für diese Klasse mit dem entsprechenden Namen erstellt. Wenn eine Klasse einer Person zugeordnet ist, legt die Person (oder die Gruppe) einen grundlegenden Satz von Verantwortlichkeiten für die Klasse fest. Dieser grundlegende Satz von Verantwortlichkeiten sollte alles umfassen, was (wenn überhaupt) sofort klar ist.
G.2. Die Gruppe

Unabhängig davon, ob sie implizit oder explizit definierte Anforderungen an das System kennen, müssen Sie mit den Menschen in der Gruppe vertraut sein.


Die Gruppe von fünf oder sechs Personen der Kerngruppe sollte aus Entwicklern, Fachexperten und einem objektorientierten Technologie- Moderator bestehen.

G.3. Die Sitzung

Bevor Sie mit der Sitzung beginnen, muss ein Teil des Problems für die Sitzung ausgewählt werden, auf dem der Focus liegen soll. Im Wesentlichen bedeutet dies das Kommissionieren einer Menge von zu verwendenden Klassen.

Wählen Sie die Szenarien aus, die Sie durchgehen wollen und die die ausgewählten Klassen verwenden. Beginnen Sie mit den Szenarien, die Teil des normalen Betriebs des Systems ist zuerst, und dann außergewöhnlichen Szenarien, wie zum Beispiel Fehlererkennung.


Das Szenarien durchspielen ist das Herz einer CRC-Kartensitzung. Um ein Szenario durchzuspielen, adressieren Sie jede darin enthaltene Aktion ein mal. Zuerst entscheiden Sie sich, welche Klasse für diese Funktion verantwortlich ist. Der Eigentümer der Klasse nimmt seine Karte und hält Sie nach oben. Wenn ein Karte oben ist, ist sie ein Objekt, das Dinge ausführen kann. Der Eigentümer gibt bekannt, was er zur Erfüllung der Verantwortlichkeit benötigt. Die Verantwortlichkeit wird, wenn möglich, in kleinere Aufgaben unterteilt. Diese kleineren Aufgaben können von dem Objekt erfüllt werden, wenn sie angemessen ist, oder sie kann durch die Interaktion mit anderen Objekten (Kollaboratoren) erfüllt werden. Wenn keine entsprechende Klasse existiert, müssen Sie eine erstellen und diese einer Person zuweisen.

G.4. Der Prozess

CRC-Karten werden in der Phase Analyse und Design verwendet. Der Prozess für diese Phasen unterscheidet sich primär dadurch, wie die Klassen und Szenarien ausgewählt werden.

In der Analysephase sind die Klassen und Szenarien aus dem Problembereich und werden allgemein von den Anforderungen abgeleitet. In der Designphase werden Klassen und Szenarien des Lösungsbereiches hinzugefügt. In der Analysephase beginnt die allererste Sitzung ohne Klassen und Szenarien, um sie in einer speziellen Sitzung auszuwählen und zu erstellen.
Stichwortverzeichnis

Die Erstellung des Index im Dokument ist ein wenig nach dem Zufallsprinzip entstanden und noch nicht vertrauenswürdig. Bitte helfen Sie neue Index-Einträge auszuwählen!

Symbole
Änderbarkeit
   eines Assoziationsendes, 285
   eines Attributes, 263
Änderbarkeit Assoziationsende, 285
Änderbarkeit Attribut, 263
Überblick
   über das Anwenderhandbuch, 4

A
Abhängigkeit, 285
Abhängigkeits Clients, 287
Abhängigkeitsname, 286
Aggregation
   eines Assoziationsendes, 284
Akteur, 40, 51, 229, 375
Akteur Assoziationsendes, 232
Akteur Detail-Register, 229
Akteur Generalisierungen, 231
Akteur hinzufügen, 230
Akteur Modifizierer, 231
Akteur Name, 231
Akteur Namensraum, 231
Akteur Spezialisierungen, 231
Aktion, 375
Aktion hinzufügen, 343
Aktiver Akteur, 43
Aktivitätsdiagramm, 375
Alle Elemente, 105
Alle Klassen generieren, 127
Alternative Abläufe
   des Anwendungsfall, 48, 50
Alternative Szenarien, 50
Analyse, 1, 8, 12
   objektorientiert, 378
Analysis, 375
Anfangswert
   des Attributs, 264
   des Parameters, 272
Anfangswert Attribut, 264
Anfangswert Parameter, 272
Anforderung
   Erfassung, 40
Anforderungserfassung, 375
Annahmen zu Beginn
   des Anwendungsfall, 48
Anwender-Feedback, 4
Anwendungsfall, 41, 41, 52, 232, 375
   Alternative Abläufe, 48, 50
   Hierarchie, 45
   Standardablauf, 48, 49
   Anwendungsfall Detail-Register, 232
   Anwendungsfall Erweitert-Beziehung, 235
   Anwendungsfall Generalisierung, 47, 235
   Anwendungsfall hinzufügen, 233
   Anwendungsfall Include-Beziehungen, 235
   Anwendungsfall Modifizierer, 234
   Anwendungsfall nachfolgende Annahmen, 48
   Anwendungsfall Name, 234
   Anwendungsfall Namensraum, 234
   Anwendungsfall Realisierung, 379
   Anwendungsfall Spezialisierung, 47, 235
   Anwendungsfall, hierarchisch, 55
   Anwendungsfall-Name, 47
   Anwendungsfall-Spezifikation, 41, 47
   Anwendungsfall-Szenario, 47
   Anwendungsfall-Vorbedingung, 48
   Anwendungsfall-Ziel, 47
   Anwendungsfalldiagramm, 42, 228, 375
   Anwendungsfallspezifikation, 375
Art
   des Parameters, 272
   Art Parameter, 272
   Assoziation, 276, 375
   in einem Anwendungsfalldiagramm, 53
   Assoziations-Stereotyp, 278
   Assoziationsende, 279
   Assoziationsende Aggregation, 284
   Assoziationsenden
   des Akteurs, 232
   einer Assoziation, 279
   Assoziationsklasse, 375
   Assoziationsname, 278
   Attribut
   einer Klasse, 375
   eines Objektes, 375
   Attribute, 260
   Attribute Name, 262
   Attributtyp, 263
   Aufzählung hinzufügen, 213, 217
   Aus Diagramm entfernen, 106
   Aus Modell entfernen, 106
   Ausgangsmodell, 227
   von Diagrammen, 227
   Ausgelöste Signale
   einer Operation, 269

B
Basis
   der Include-Beziehung, 249
   Basis der Include-Beziehung, 249
Basis-Anwendungsfall
   der Extend-Beziehung, 246
   des Erweiterungspunktes, 238
   Basis-Anwendungsfall der Extend-Beziehung, 246
   Basis-Anwendungsfall Erweiterungspunkt, 238
Basisklasse
  von Stereotypen, 225
Beenden, 105
Beziehung
  Extend, 46, 55, 376
  Generalisierung, 56, 376
  Include, 45, 55, 376
Build, 13, 17

CASE, 376
Clients
  einer Abhängigkeit, 287
Code generieren
  aus der statischen Struktur, 80
  aus Interaktionen, 81
  aus Kollaborationsdiagrammen, 81
  aus Sequenzdiagrammen, 81
  aus Zustandsdiagrammen, 81
Codegenerierung, 80

Das Visions-Dokument, 40
Datatyp Eigenschaftswerte, 216
Datatyp hinzufügen, 266
Datatyp-Eigenschaften, 216
Datatyp-Modifizierer, 217
Datatyp-Sichtbarkeit, 218
Datentyp, 215
Datentyp hinzufügen, 213, 253, 262, 270
Datentyp-Eigenschaftsfelder, 217
Datentyp-Name, 217
Design, xvii, 1, 8, 12
  objektorientiert, 378
  opportunistisch, 378
Designprozess
  Iterativ, 377
  Wasserfall, 381
Detail-Register
  einer Abhängigkeit, 285
  einer Extend-Beziehung, 243
  für Anwendungsfälle, 232
  für den Akteur, 229
  für Diagramme, 226
  für die Generalisierung, 240
  für ein Assoziationsende, 280
  für ein Attribut, 260
  für ein Paket, 252
  für ein Signal, 273
  für eine Assoziation, 276
  für eine Include-Beziehung, 247
  für eine Klasse, 256
  für eine Operation, 265
  für einen Erweiterungspunkt, 237
  für einen Parameter, 269
  für Enumeration, 219
  für Stereotypen, 222
Detail-Register Abhängigkeit, 285
Detail-Register Assoziation, 276
Detail-Register Assoziationsenden, 280
Detail-Register Attribut, 260
Detail-Register Erweiterungspunkt, 237
Detail-Register Extend-Beziehung, 243
Detail-Register Generalisierung, 240
Detail-Register Include-Beziehung, 247
Detail-Register Klasse, 256
Detail-Register Operation, 265
Detail-Register Paket, 252
Detail-Register Parameter, 269
Detail-Register Signal, 273
Diagramm, 225
  Aktivität, 375
  Anwendungsfall, 42, 375
  Klasse, 377
  Kollaboration, 377
  Sequenz, 379
  Systemsequenz, 380
  Systemzustand, 380
  Zustand, 382
Diagramm Detail-Register, 226
Diagramm Eigenschaftsfelder, 227
Diagramm-Name, 227
Diskriminator
  der Generalisierung, 241
Diskriminator Generalisierung, 241
Dokumentation in Anwendungsfalldiagrammen, 57
Drucken ..., 99

Eigene Elemente
  des Paketes, 255
  von Modell, 215
Eigenschaften
  von Datatypen, 216
  von Enumeration, 219
Eigenschaftsfelder
  für Attribute, 262
  für Datatyp, 217
  für Diagramme, 227
  für ein Assoziationsende, 281
  für ein Signal, 274
  für eine Assoziation, 278
  für eine Klasse, 258
  für eine Operation, 267
  für einen Parameter, 271
  für Enumeration, 221
  für Stereotypen, 224
Eigenschaftsfelder Assoziation, 278
Eigenschaftsfelder Assoziationsende, 281
Eigenschaftsfelder Attribut, 262
Eigenschaftsfelder Klasse, 258
Eigenschaftsfelder Operation, 267
Eigenschaftsfelder Parameter, 271
Eigenschaftsfelder Signal, 274
Eigenschaftssymbolleiste
für ein Assoziationsende, 281
für ein Attribut, 261
für ein Signal, 274
für eine Assoziation, 277
für eine Klasse, 257
für eine Operation, 266
für einen Parameter, 270
für Enumeration, 220
für Stereotypen, 223

Eigenschaftssymbolleiste Assoziation, 277
Eigenschaftssymbolleiste Assoziationsende, 281
Eigenschaftssymbolleiste Attribut, 261
Eigenschaftssymbolleiste Enumeration, 220
Eigenschaftssymbolleiste Klasse, 257
Eigenschaftssymbolleiste Operation, 266
Eigenschaftssymbolleiste Parameter, 270
Eigenschaftssymbolleiste Signal, 274
Eigenschaftssymbolleiste Stereotyp, 223

Eigenschaftswerte

- der Operation, 265
- einer Assoziation, 277
- eines Assoziationsendes, 280
- eines Parameters, 270
- eines Signals, 273
- von Attribut, 261
- von Datatypen, 216
- von Enumeration, 220
- von Klasse, 256

Eigenschaftswerte Assoziation, 277
Eigenschaftswerte Assoziationsende, 280
Eigenschaftswerte Attribut, 261
Eigenschaftswerte Enumeration, 220
Eigenschaftswerte Klasse, 256
Eigenschaftswerte Operation, 265
Eigenschaftswerte Parameter, 270
Eigenschaftswerte Signal, 273

Einbringen

- in ArgoUML, 2

Einen Anwendungsfall generalisieren, 47
Eingefügter Anwendungsfall

- der Include Relationship, 249
- Eingefügter Anwendungsfall der Include-Beziehung, 249

Entwickler-Programmierhandbuch, Das, 2
Entwicklerbereich, 2

Entwurf, xviii, 1

Enumeration, 219
Enumeration (Aufzählung) hinzufügen, 253, 262
Enumeration Detail-Register, 219
Enumeration Eigenschaften, 219
Enumeration Eigenschaftsfelder, 221
Enumeration Literal, 222
Enumeration Literale, 222
Enumeration Modifizierer, 221
Enumeration Name, 221
Enumeration Sichtbarkeit, 221

EPS, 15

Ergänzende Anforderungs-Spezifikation, 41
Ergänzende Anforderungsspezifikation, 41, 50, 376
Erstelle neu

- Anwendungsfall, 52
- Erweiterungspunkt, 52

Erweiternde Anwendungsfälle

- eines Erweiterungspunktes, 238
- Erweiterte Beziehung
des Anwendungsfalles, 235

Erweiterung

- der Extend-Beziehung, 246
- Erweiterung Extend-Beziehung, 246
- Erweiterungspunkt, 52, 236
der Extend-Beziehung, 246
eines Anwendungsfalles, 236
- Erweiterungspunkt erweitert Anwendungsfälle, 238
- Erweiterungspunkt Extend-Beziehung, 246
- Erweiterungspunkt hinzufügen, 233, 237
- Erweiterungspunkte Anwendungsfall, 236

Erzeuge neuen

- Stereotyp, 343

Erzeuge neue

- Aktion, 343
  Assoziation in einem Anwendungsfalldiagramm, 53
  Aufzählung, 217
  Enumeration (Aufzählung), 253, 262
  Extend-Beziehung, 244
  Extend-Beziehung in einem Anwendungsfalldiagramm, 55
  Generalisierungs-Beziehung in einem Anwendungsfalldiagramm, 56
  Include Beziehung in einem Anwendungsfalldiagramm, 55

Erzeuge neuen

- Akteur, 230
- Anwendungsfall, 233
- Datatyp, 253, 262
- Datentyp, 270
- Erweiterungspunkt, 233, 237
- Qualifizierer, 281, 340
- Signaleingang, 230
- Stereotyp, 217, 238, 240, 245, 248, 253, 258, 262,
  271, 274, 278, 281, 286, 289, 296, 305, 308, 311,
  315, 318, 320, 322, 324, 328, 333, 349, 355, 357,
  360, 362
- Stereotypen, 220, 266

Erzeugt neuen

- Datatyp, 266
- Stereotyp, 340

Explorer, 137
- Verhalten der Maus, 137

Extend Beziehung, 46

- Extend-Beziehung, 243, 376
  in einem Anwendungsfalldiagramm, 55
  Extend-Beziehung hinzufügen, 244

Externe Entität, 229

F

FAQ, 2
Feedback, 4
Fenster, 376

G
Generalisierung, 239
des Anwendungsfalles, 235
Generalisierungen
des Akteur, 231
von Paket, 255
Generalisierungs-Beziehung, 376
in einem Anwendungsfalldiagramm, 56
GIF, 15
Gleichzeitigkeit
einer Operation, 268
Gleichzeitigkeit Operation, 268
GUI, 376

H
Häufig gestellte Fragen, 2
Haupteigenschaften
im Visions-Dokument, 42
Hierarchie von Anwendungsfällen, 45
Hierarchische Anwendungsfälle, 55
Hierarchisches Zustandsdiagramm, 376

I
Importierte Elemente
eines Paketes, 255
Include Beziehung
in einem Anwendungsfalldiagramm, 55
Include-Beziehung
eines Anwendungsfalles, 235
Include-Beziehung, 45, 247, 376
Iteration, 10
Iterative Prozesse, 9
Iterativer Designprozess, 377

J
Jason Robbins, 2
Java, 377

K
Kardinalität
eines Assoziationsendes, 282
einstellen, 55
in einem Anwendungsfalldiagramm, 44
Kardinalität Assoziationsende, 282
Kardinalität Attribut, 263
Kardinalität einstellen
für eine Assoziation in einem
Anwendungsfalldiagramm, 55
Kardinalität
der Attribute, 263
Klasse, 256, 377
Klassendiagramm, 250, 377
Klassenmethode, 378
Klassenmodifizierer, 258
Klassenname, 258
Kollaboration, 377
Kollaborationsdiagramm, 377
Kollaborator, 377
Konzepte
eines Signals, 275
Konzept Klassendiagramm, 377
Kritik, 378

L
Lieferanten
einer Abhängigkeit, 287
Lieferanten Abhängigkeit, 287
Literale
von Enumeration, 222

M
Mail-Listen, 2, 2
Marktumfeld
im Visions-Dokument, 42
Mealy Machine, 378
Menü Hilfe, 27
Menü Kritik, 26
Menü Werkzeuge, 26
Menü: Ansicht, 25
Menü: Bearbeiten, 22
Menü: Datei, 21
Menüleiste, 21
Methode
einer Klasse, 378
eines Objektes, 378
Modell Stereotyp, 214
Modell, Das, 212
Modell-eigene Elemente, 215
Modell-Modifizierer, 214
Modell-Namensraum, 214
Modell-Sichtbarkeit, 214
Modifizierer
der Enumeration, 221
des Akteurs, 231
einer Klasse, 258
eines Anwendungsfalles, 234
eines Assoziationsendes, 282
eines Datentyps, 217
für eine Operation, 268
von Modell, 214
von Paket, 254
von Stereotypen, 224
Modifizierer Assoziationsende, 282
Moore Machine, 378

N
Nach-Bedingungen
des Anwendungsfalles, 48
Nachfolgende Annahmen
zum Anwendungsfall, 48
Name

der Enumeration, 221
der Extend-Beziehung, 245
der Generalisierung, 241
der Include-Beziehung, 248
der Klasse, 258
der Operation, 267
des Akteurs, 231
des Anwendungsfalles, 47, 234
des Datentyps, 217
des Diagrammes, 227
des Erweiterungspunktes, 238
des Paketes, 254
des Parameters, 271
des Stereotypen, 224
einer Abhängigkeit, 286
einer Assoziation, 278
 eines Assoziationsendes, 281
eines Attributes, 262
eines Signals, 274
von Modell, 214
Name Assoziationsende, 281
Name der Generalisierung, 241
Name der Operation, 267
Name des Modelles, 214
Name des Paketes, 254
Name Erweiterungspunkt, 238
Name Extend-Beziehung, 245
Name der Include-Beziehung, 248
Namensraum

der Extend-Beziehung, 245
der Generalisierung, 241
der Include-Beziehung, 248
des Akteurs, 231
des Paketes, 254
einer Abhängigkeit, 287
 eines Anwendungsfalles, 234
von Modell, 214
von Stereotypen, 224
Namensraum Abhängigkeit, 287
Namensraum der Include-Beziehung, 248
Namensraum Extend-Beziehung, 245
Namensraum Generalisierung, 241
Namensraum Paket, 254
Navigation

Baum, 137
einstellen, 54
Fenster, 137
Navigation einstellen
auf eine Assoziation in einem
Anwendungsfalldiagramm, 54
Navigations

Baum, 137
Fenster, 137
Neu, 93
Neu erstellen
Akteur, 51
Neu erzeugen

Name

der Enumeration, 221
der Extend-Beziehung, 245
der Generalisierung, 241
der Include-Beziehung, 248
der Klasse, 258
der Operation, 267
des Akteurs, 231
des Anwendungsfalles, 47, 234
des Datentyps, 217
des Diagrammes, 227
des Erweiterungspunktes, 238
des Paketes, 254
des Parameters, 271
des Stereotypen, 224
einer Abhängigkeit, 286
einer Assoziation, 278
 eines Assoziationsendes, 281
eines Attributes, 262
eines Signals, 274
von Modell, 214
Name Assoziationsende, 281
Name der Generalisierung, 241
Name der Operation, 267
Name des Modelles, 214
Name des Paketes, 254
Name Erweiterungspunkt, 238
Name Extend-Beziehung, 245
Name der Include-Beziehung, 248
Namensraum

der Extend-Beziehung, 245
der Generalisierung, 241
der Include-Beziehung, 248
des Akteurs, 231
des Paketes, 254
einer Abhängigkeit, 287
 eines Anwendungsfalles, 234
von Modell, 214
von Stereotypen, 224
Namensraum Abhängigkeit, 287
Namensraum der Include-Beziehung, 248
Namensraum Extend-Beziehung, 245
Namensraum Generalisierung, 241
Namensraum Paket, 254
Navigation

Baum, 137
einstellen, 54
Fenster, 137
Navigation einstellen
auf eine Assoziation in einem
Anwendungsfalldiagramm, 54
Navigations

Baum, 137
Fenster, 137
Neu, 93
Neu erstellen
Akteur, 51
Neu erzeugen

Aufzählung, 213
Datentyp, 213
Package, 213
Stereotyp, 213
Neue Aktion, 343
Neue Aufzählung, 213, 217
Neue Enumeration (Aufzählung), 253, 262
Neue Extend-Beziehung, 244
Neuer Akteur, 230
Neuer Anwendungsfall, 233
Neuer Datatyp, 262, 266
Neuer Datatyp, 213, 253, 270
Neuer Erweiterungspunkt, 233, 237
Neuer Qualifizierer, 281, 340
Neuer Signaleingang, 230
Neuer Stereotyp, 213, 217, 220, 238, 240, 245, 248, 253,
258, 262, 266, 271, 274, 278, 281, 286, 289, 296, 305,
308, 311, 315, 318, 320, 322, 324, 328, 333, 340, 343,
349, 355, 357, 360, 362
Neues Diagramm, 26
Neues Paket, 213
Nicht-funktionale Anforderungen, 41, 50
Nicht-funktionale Parameter
im Visions-Dokument, 42
Nicht-funktionale Randbedingungen, 50
Nutzer
im Visions-Dokument, 42

O

Object Constraint Language, 378
Object Management Group, 378
Objekt, 378
Objektdiagramme, 250
Objektmethode, 378
OCL, 378
OMG, 378
OOA&D, 378
Operation, 264
Operation Ausgelöste Signale, 269
Operation Modifizierer, 268
Operation Parameter, 269
Opportunistisches Design, xviii, 14, 378
Ort
des Erweiterungspunktes, 238
Ort Erweiterungspunkt, 238

P

Paket, 252
Paket hinzufügen, 213
Paket Importierte Elemente, 255
Paket-eigene-Elemente, 255
Paket-Generalisierungen, 255
Paket-Modifizierer, 254
Paket-Spezialisierungen, 255
Paketdiagramme, 250
Parameter, 269
einer Operation, 269
Parameter-Name, 271
Parameter-Stereotyp, 271
Parameter-Typ, 271
Passiver Akteur, 43
PGML, 15
PNG, 15
Powertype
der Generalisierung, 242
Powertype Generalisierung, 242
Problemlösung, xviii, 14, 380
Programmierhandbuch, 2
Projekt öffnen..., 93
Projekt speichern, 94
PS, 15
Qualifizierer hinzufügen, 281, 340
Randbedingungen
im Visions-Dokument, 42
Realisation Anwendungsfall, 379
Reflektion-während-Aktion, xviii, 13, 379
Register
für Datatypdetails, 216
für Modelldetails, 212
Register Datatypdetails, 216
Register Modelldetails, 212
Robbins, Jason, 2
Round-Trip Engineering, 82
Siehe einrichten .... 99
Sequenzdiagramm, 379
SGML, 379
Sichtbarkeit
der Enumeration, 221
der Operation, 267
des Datentyps, 218
eines Assoziationsendes, 285
eines Attributes, 263
don Modell, 214
den Stereotypen, 224
Sichtbarkeit Assoziationsendes, 285
Sichtbarkeit Attribut, 263
Sichtbarkeit Operation, 267
Signal, 272
Signal-Stereotyp, 274
Signaleingang hinzufügen, 230
Signalkontexte, 275
Signalname, 274
Simula 67, 379
Spezialisierungen
des Akteurs, 231
von Anwendungsfällen, 47, 235
von Paket, 255
Spezifikation
eines Anwendungsfallen, 47
Standard Graphical Markup Language, 379
Standardablauf
des Anwendungsfallen, 48, 49
Standardwert
des Parameters, 272
Standardwert Parameter, 272
Stereotyp, 222, 379
der Operation, 267
einer Abhängigkeit, 287
einer Assoziation, 278
eines Assoziationsendes, 282
eines Parameters, 271
eines Signals, 274
don Modell, 214
Stereotyp Abhängigkeit, 287
Stereotyp Assoziationsende, 282
Stereotyp Basisklasse, 225
Stereotyp Detail-Register, 222
Stereotyp Eigenschaftsfelder, 224
Stereotyp Modifizierer, 224
Stereotyp Namensraum, 224
Stereotyp Operation, 267
Stereotyp Sichtbarkeit, 224
Stereotyp-Name, 224
Stereotypen
in Anwendungsfalldiagrammen, 56
Stereotypisieren, 379
Stereotyp hinzufügen, 238
Subklasse
der Generalisierung, 242
Subklasse Generalisierung, 242
Superklasse
der Generalisierung, 242
Superklasse Generalisierung, 242
SVG, 15, 380
Symbolleiste
den Datentypeigenschaften, 216
Symbolleiste Datei, 27
Symbolleiste Datentypeigenschaften, 216
Symbolleiste: Ansicht, 27
Symbolleiste: Bearbeiten, 27
Symbolleiste: Neues Diagramm, 27
Symbolleisten, 21
Systemgrenzen in Anwendungsfalldiagrammen, 57
Systemsequenzdiagramm, 380
Systemzustandsdiagramm, 380
Szenario, 48, 379
Tastenkürzel
Alt-F4., 105
Stichwortverzeichnis

Entf, 106
F7, 127
Strg-A, 105
Strg-Entf, 106
Strg-N, 93
Strg-O, 93
Strg-P, 99
Strg-S, 94
To-Do Liste, 380
Transition, 380
Typ
des Attributes, 263
des Parameters, 271
eines Assoziationsendes, 282
Typ Assoziationsende, 282

U
Umkehr-Engineering, 82
UML, 380

V
Verantwortlichkeit, 380
Verbindungen
einer Assoziation, 279
Verhalten der Maus
im Explorer, 137
Verständlichkeit, 14
Verständnis, xviii
Verstehen, 380
Visions-Dokument, 41, 42, 381
Fallstudie, 58
Vorbedingung
des Anwendungsfalles, 48

W
W3C, 381
Wasserfall-Designprozess, 381

X
XMI, xviii, 14, 32, 34, 381
XML, xviii, xix, 381

Z
Ziel
des Anwendungsfalles, 47
Ziele
im Visions-Dokument, 42
Zustand, 381
Zustandsdiagram, 382
Zustandsdiagramm, Hierarchisch, 376