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

ArgoUML Anwenderhandbuch : Eine Lernanleitung und eine Referenzbeschreibung

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

Zusammenfassung

Diese Version des Handbuches beschreibt die Version 0.35.1 von ArgoUML.

Inhaltsverzeichnis

Vorwort

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

Aktuelle CASE-Tools enthalten Automations- und grafische Anwender-Schnittstellen, die die manuelle Arbeit der Designeingabe reduzieren und den Entwurf in Programmcode transformieren. Sie unterstützen die Designer bei ihren Entscheidungen hauptsächlich durch die Visualisierung des Entwurfes und einfachen syntaktischen Überprüfungen. Darüber hinaus weisen viele CASE-Tools auch substantielle Vorteile im Bereich der Versionskontrolle und nebenläufiger Designmechanismen auf. Ein, bisher nicht besonders gut unterstützter Bereich ist die Analyse von Designentscheidungen.

Aktuelle CASE-Tools haben eine Anwenderschnittstelle (GUI), die es den Designern ermöglicht, auf alle, durch das Tool angebotenen Funktionen zuzugreifen. Und sie unterstützen den Entwurfsprozess, indem sie es dem Designer erlauben, Diagramme im Stil populärer Design-Methoden einzugeben. Üblicherweise enthalten sie aber keine Prozessunterstützung, die den Designer durch die Designschritte führt. Designer beginnen normalerweise mit einer leeren Seite und müssen jeden Aspekt des Entwurfes aus dem Kopf ableiten.

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.

Darüber hinaus ist es unser Ziel, eine verständliche Unterstützung für OCL (die Object Constraint Language (Objekt-Bedingungs-Sprache)) und XMI (dem XML Model Interchange Format(XML-Modell-Austauschformat)) bereitzustellen.

ArgoUML wurde ursprünglich durch eine kleine Gruppe von Leuten als Forschungsprojekt entwickelt. ArgoUML hat viele Eigenschaften, die es besonders machen. Aber es implementiert nicht alle Funktionen, kommerzieller CASE-Tools.

Die aktuelle Version (0.35.1) von ArgoUML implementiert alle Diagrammtypen des UML 1.4 Standard [http://www.omg.org/cgi-bin/doc?formal/01-09-67] (ArgoUML-Versionen vor 0.20 implementierten den UML 1.3 Standard [http://www.omg.org/cgi-bin/doc?formal/00-03-01]). Sie ist in Java geschrieben und läuft auf jedem Rechner, der die Java-Plattform, Version 5 oder eine neuere Version aufweist. Es verwendet zum Speichern offene Dateiformate, wie XMI [http://www.omg.org/cgi-bin/doc?formal/02-01-01] (XML Metadata Interchange Format) (für Modell-Informationen) und PGML [http://www.w3.org/TR/1998/NOTE-PGML] (Precision Graphics Markup Language) (für grafische Informationen). Wenn ArgoUML UML 2.0 implementiert, wird PGML durch die UML Diagram Interchange Specifikation (Diagramm Austauschspezifikation) ersetzt.

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 Handbuches unterstützten, um sie alle namentlich benennen zu können.

ArgoUML ist frei verfügbar und kann im kommerziellen Umfeld genutzt werden. Wenn Sie ArgoUML herunterladen, entnehmen Sie bitte die Nutzungsbedingungen den beigefügten Lizenzbedingungen. Wir bieten Ihnen den Quellcode von ArgoUML an, damit Sie sich diesen ansehen, an Ihre Bedürfnisse anpassen und verbessern können. Wir hoffen, dass sich ArgoUML nach und nach zu einem leistungsfähigen und nützlichen Tool für jedermann entwickelt.

Dieses Anwenderhandbuch ist für den Designer gedacht, der seine Entwürfe mit Hilfe von ArgoUML erstellen möchte. Das Handbuch setzt voraus, dass Sie mit UML vertraut sind. Eventuell unterstützt es aber auch diejenigen, für die UML neu ist.

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

Das ArgoUML-Projekt begrüßt all diejenigen, die tiefer 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.

Die Unterstützung für Objekte begann mit der Programmiersprache Simula 67- aber es war das Erscheinen der Hybridsprachen wie C++, Ada und Objekt Pascal in den 80'ern, die das Durchstarten der OOA&D ermöglichten. Diese Sprachen enthielten eine Unterstützung für Beides, der OO- und der prozeduralen Programmierung. Die objektorientierte *Programmierung* wurde zum Mainstream.

Ein OO-System wird als *Simulation* der realen Welt mit Hilfe von Softwarebausteinen entworfen und implementiert. Diese Prämisse ist so mächtig wie auch einfach. Durch die Anwendung des OO-*Design*-Ansatzes kann ein System entworfen und getestet (oder korrekter: simuliert) werden, ohne dass es vorher gebaut werden muss.

Es war 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

Während der 80'er Jahre wurden einige OOA&D-Vorgehensmethoden und -Notationen von unterschiedlichen Forscherteams entwickelt. Es wurde klar, dass es viele gemeinsame Themenbereiche gab und so wurde während der 90'er Jahre eine vereinheitlichte OOA&D-Notation unter der Schirmherrschaft der Object Management Group [http://www.omg.org] entwickelt. Dieser Standard wurde als Unified Modeling Language (UML - Einheitliche Modellierungssprache) bekannt, und ist heute die Standardsprache für die Kommunikation von OO-Konzepten.

ArgoUML war als Tool und Umgebung für Analyse und Design objektorientierter Softwaresysteme gedacht. In diesem Sinne ist es vergleichbar mit vielen kommerziellen CASE-Tools, die für die Modellierung von Softwaresystemen verkauft werden. ArgoUML enthält eine Anzahl sehr wichtiger Merkmale, wodurch es sich von diesen Tools unterscheidet.

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 Sofwaredesigner und -Architekten erhöhten.

3. ArgoUML nutzt sehr stark offene Standards - UML, XMI, SVG, OCL und andere.

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

5. ArgoUML ist ein Open-Source-Projekt. Die Verfügbarkeit des Quellcodes stellt sicher, dass eine neue Generation von Softwaredesignern und -forschern ein bewährtes Framework hat, auf deren Basis sie die Entwicklung und Evolution von CASE-Tooltechnologien vorantreiben können.

UML ist die bevorzugte OO-Modellierungssprache und Java eine der produktivsten OO-Entwicklungsplattformen. Jason Robbins und der Rest seines Forscherteams an der Universität Kalifornien vereinten diese Vorteile durch die Entwicklung von ArgoUML. Das Ergebnis ist ein solides Entwicklungstool und eine Umgebung für das OO-Systemdesign. Darüber hinaus bildet es eine Testumgebung für die Evolution der objektorientierten CASE-Tool-Entwicklung und -Forschung.

Das erste Release von ArgoUML war 1998 verfügbar. Mehr als 100.000 Downloads bis Mitte 2001 zeigte, dass dieses Projekt, sowohl in den Bildungs- als auch in den kommerziellen Bereichen populär wurde.

1.1.3. Mehr über das ArgoUML-Projekt

1.1.3.1. Wie wurde ArgoUML entwickelt

Jason Elliot Robbins gründete das Argo-Projekt und erhielt früh die Projektleitung. Solange Jason im Projekt aktiv war, hatte er alle Hände voll mit der Projektleitung zu tun. Das Projekt entwickelte sich kontinuierlich sehr stark weiter. Es gab mehr als 300 Mitglieder in der Entwickler-Mailliste (siehe http://argouml.tigris.org/servlets/ProjectMailingListList [http://argouml.tigris.org/servlets/ProjectMailingListList]). Mehrere Dutzend davon, bildeten die Haupt-Entwicklergruppe.

Die Entwickler-Mails sind der Ort, in dem die Diskussionen über die aktuellen Aktivitäten stattfinden und Entwickler die Richtung diskutieren, die das Projektes nehmen 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 diesem Zweck geschrieben.

1.1.3.2. Mehr über die Infrastruktur

Neben der Entwickler-Mailliste gibt es auch eine Mailliste für Anwender (siehe The ArgoUML Mailing List [http://argouml.tigris.org/servlets/ProjectMailingListList]), in der wir Probleme aus der Anwendersicht diskutieren können. Auch Entwickler lesen diese Mails, so dass grundsätzlich hoch qualifizierte Hilfe verfügbar ist.

Bevor Sie eine Frage per Mail stellen, sollten Sie einen Blick auf die von Ewan R. Grantham verwaltete Anwender-FAQ [http://argouml.tigris.org/faqs/users.html] werfen.

Weitere Informationen über ArgoUML und andere UML-bezogene Themen sind unter der von Linus Tolke verwalteten ArgoUML Webseite [http://argouml.tigris.org] verfügbar.

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 modularisiertem 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.

Diese Themen werden in „Das Programmierhandbuch (Cookbook)“ [http://argouml-stats.tigris.org/documentation/defaulthtml/cookbook/], „Die FAQ“ [http://argouml.tigris.org/faqs/users.html] und „Die Kurzanleitung“ [http://argouml-stats.tigris.org/documentation/quick-guide-0.22/] beschrieben.

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.

Für jedes einzelne UML-Konzept wird dessen Anwendung erläutert. Anschliessend wird dessen Einsatz innerhalb von ArgoUML beschrieben. Zum Schluss wird eine Fallstudie benutzt, um Beispiele für diese Konzepte zu geben.

1.3.2. Die Struktur des Referenzhandbuches

Kapitel 8, *Einleitung* ist ein Überblick über die Anwenderschnittstelle und enthält eine Zusammenfassung der verschiedenen UML-Diagrammtypen von ArgoUML. Kapitel 10, *Die Menüzeile* und Kapitel 11, *Der Explorer* beschreiben die Menüzeile und jedes Fenster der Anwenderschnittstelle.

Kapitel 15, *Die Hinweise* beschreibt Details aller im System vorhandenen kognitiven Hinweise. Unter Umständen wird ArgoUML direkt auf dieses Handbuch verweisen, wenn es die Hilfestellung zu einem Hinweis gibt.

Kapitel 16, *Modellreferenz auf höchster Ebene* ist ein Überblick über die Modellelemente (z.B. die UML-Elemente, die in den Diagrammen verwendet werden können) von ArgoUML. Die folgenden Kapitel (Kapitel 17, *Referenz der Modellelemente für Anwendungsfalldiagramme* bis Kapitel 24, *Profile*) beschreiben die Modellelemente, die in jedem ArgoUML-Diagramm erzeugt werden können, deren Eigenschaften, sowie einige Standard-Modellelemente die im System enthalten sind.

Es ist ein vollständiges Glossar vorhanden. Anhang A, *Ergänzendes Material für die Fallstudie* enthält Material, das die im Dokument durchgehend benutzte Fallstudie unterstützt. Anhang B, *UML-Ressourcen* und Anhang C, *UML-konforme CASE Tools* verweisen auf Hintergrundinformationen über UML und UML-CASE-Tools. Anhang F, *Open Publication Lizenz* ist eine Kopie der "GNU Freien Dokumentations Lizenz".

1.3.3. Anwender-Feedback

Bitte teilen Sie uns mit, was Sie über dieses Anwenderhandbuch denken. Ihre Kommentare helfen uns, Verbesserungen vorzunehmen. Mailen Sie Ihre Gedanken an die "ArgoUML Users Mailing List" [mailto:users@argouml.tigris.org]. Sollten Sie ein fehlendes Kapitel hinzufügen wollen, kontaktieren Sie bitte ArgoUML Developer Mailing List [mailto:dev@argouml.tigris.org]. Darüber können Sie prüfen, ob bereits jemand anderes an diesem Teil arbeitet. Sie können sich zu jeder dieser Maillisten über die ArgoUML Webseite [http://argouml.tigris.org] anmelden.

1.4. Annahmen

Diese Version des Handbuches unterstellt, dass der Leser mit UML sehr vertraut ist. Dies zeigt sich durch die sparsame Beschreibung der UML-Konzepte in diesem Übungshandbuch.

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.

Zuerst werden Sie mit dem Erscheinungsbild des Produktes vertraut gemacht und dann werden wir mit einem Testfall durch den Analyse- und Entwicklungsprozess gehen. Es wird aber nicht jeder Winkel und jede Ritze des Produktes demonstriert. Dieser Detaillierungsgrad wird im Referenzmaterial bereitgehalten, das Sie in den nachfolgenden Teilen dieses Dokumentes finden.

Der Zustand des Modelles am Ende der Hauptabschnitte wird in .zargo-Dateien verfügbar sein. Diese sind vorhanden, damit Sie verschiedene, nicht in dieser Übungsanleitung speziell behandelten Aspekte durchspielen und dann zum richtigen Zustand des Modelles zurückkehren können. Diese .zargo-Dateien werden am Ende der Abschnitte, deren Arbeit sie repräsentieren, ausgewiesen.

Ein ATM (automated teller machine) Geldautomaten-Projekt wurde als Fallstudie ausgewählt, um die verschiedenen Aspekte der von ArgoUML angebotenen Modellierungen zu demonstrieren. In den nachfolgenden Abschnitten werden wir einen „Geldautomaten“ vollständig in UML beschreiben. Die Übungsanleitung wird Sie allerdings nur durch bestimmte Teile davon führen.

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.

Die Fallstudie ist ein Geldautomat. Ihre Firma heißt „FlyByNight Industries“. Sie werden zwei Rollen spielen. Die des Projektmanagers und die des Analytikers.

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

Objektorientierung als Konzept kam in den 60'er Jahren auf und als Designkonzept ab 1972. Erst in den 80'ern entwickelte sie sich im Bereich Analyse und Design als ernstzunehmende Alternative zum *funktionalen Ansatz* . Wir können hierfür eine Anzahl von Treibern identifizieren.

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.

2. Die Entwicklung leistungsfähiger Workstations und das Erscheinen von fensterbasierten Anwenderumgebungen. Grafische Anwenderschnittstellen (GUI) haben eine eingebaute Objektstruktur.

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

Mehrere Forscher schlugen OOA&D-Prozesse und Notationen vor. Denjenigen, denen tatsächlich etwas Erfolg beschieden war, waren Coad-Yourdon, Booch, Rumbaugh OMT, OOSE/Jacobson, Shlaer-Mellor, ROOM (für Echtzeit-Design) und die hybride "Strukturierte Programmierung" von Jackson.

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.

UML wurde als gemeinsame Notation erdacht, die alle Interessen berücksichtigen sollte. Der Originalstandard wurde durch Rational Rose vorangetrieben (www.rational.com [http://www.rational.com], in der drei der wichtigsten Forscher arbeiteten (Booch, Jacobson und Rumbaugh waren involviert)). Sie produzierten im Laufe des Jahres 1996 Dokumente, die die UML, Version 0.9 und Version 0.91 beschrieben. Der Aufwand wurde industrieweit durch die Object Management Group (OMG) fortgesetzt, die bereits durch den CORBA-Standard sehr bekannt war. Ein erster Vorschlag, Version 1.0 wurde im Frühjahr 1997 veröffentlicht. Eine verbesserte Version 1.1 erschien im Herbst.

ArgoUML basiert auf UML, Version 1.4, die durch die OMG im September 2001 herausgegeben wurde. Die aktuelle offizielle Version, die durch ArgoUML unterstützt wird, ist UML, Version 1.4.2 vom Juli 2004, herausgegeben als ISO/IEC 19501. Die aktuelle UML Version ist UML, Version 2.1.2 vom November 2007.

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“.

2. Analyse. Wir nehmen die Anforderungen und beginnen diese in die Sprache der vermeintlichen Lösung umzuformen, in den *Lösungsbereich*. Zu diesem Zeitpunkt halten wir die Dinge auf einer hohen Abstraktionsebene, obwohl wir bereits in den Begrifflichkeiten der Lösung denken - abseits von den konkreten Details einer spezifischen Lösung. Dieser Vorgang ist als *Abstraktion* bekannt.

3. Design. Wir nehmen die Spezifikation aus der Analysephase und konstruieren die Lösung ganz detailliert. Wir bewegen uns von der *Abstraktion* des Problemes, hin zu seiner *Realisierung* in konkreter Form.

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

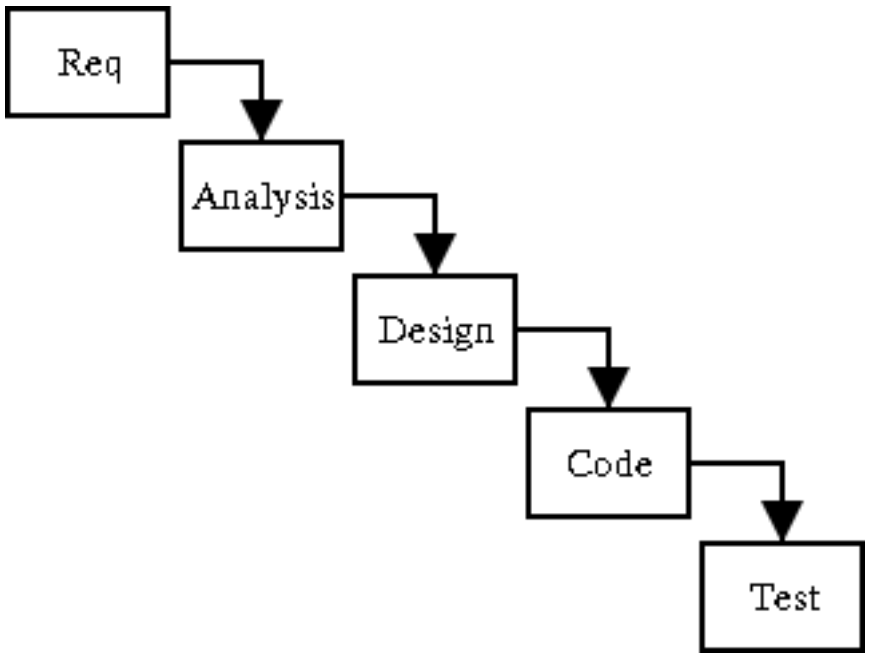
In diesem Abschnitt sehen wir uns die beiden Hauptprozesstypen an, die im Software-Engineering verwendet werden. Es gibt noch weitere, diese werden aber weniger häufig eingesetzt.

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



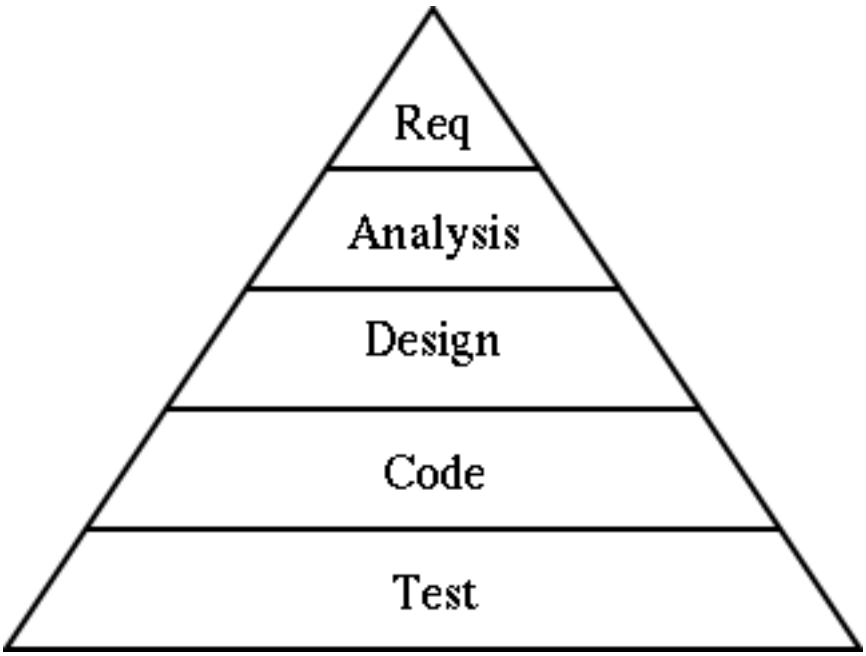
Dies ist ein sehr zufriedenstellender Prozess, in dem Anforderungen gut entworfen und keine Änderungen erwartet werden - zum Beispiel die Automatisierung eines bewährten, manuellen Systems.

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

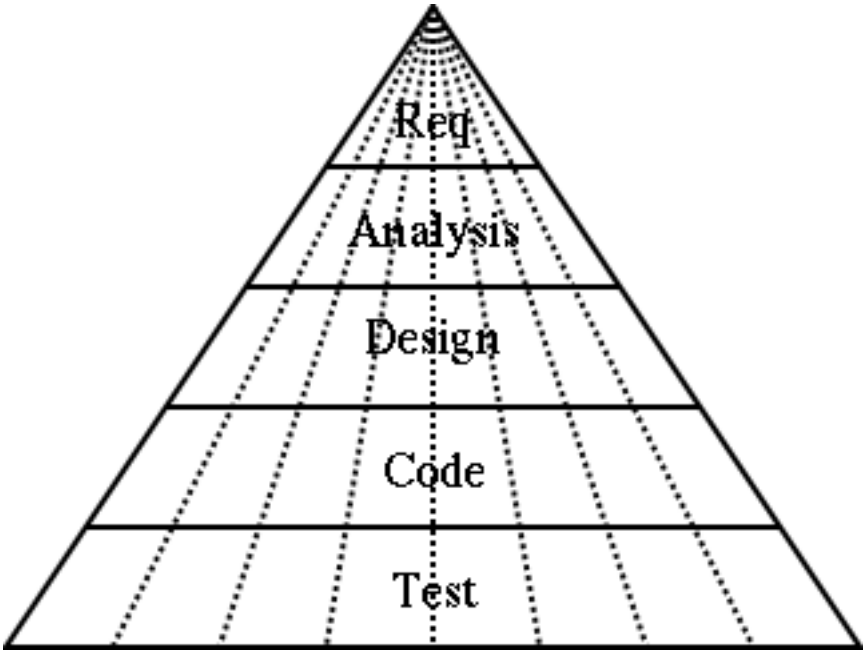
In den letzten Jahren wurde ein neuer Ansatz verwendet, dessen Ziel es war, einen kleinen Teil des Codes fertigzustellen und schnellstmöglichst auszuführen, um Probleme im Entwicklungszyklus sehr schnell aufzudecken.

Diese Prozesse verwenden eine Reihe von „Mini-Wasserfällen “, definieren einige Anforderungen (die wichtigsten zuerst, bringen diese durch den Analyse- und Design-, sowie den Buildprozess, um eine frühe Version des Produktes mit eingeschränkter Funktionalität zu erhalten. Die Rückmeldungen können dann dazu verwendet werden, die Anforderungen zu überarbeiten, punktuelle Probleme zu identifizieren usw. bevor noch mehr Arbeit investiert wird.

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.

Dieser Prozess wird wiederholt, bis alle Anforderungen implementiert wurden und das Produkt fertig ist. Es ist diese *Iteration*, die diesem Prozess seinen Namen gab. Abbildung 3.3, „ Zu erwartender Aufwand je Schritt des iterativen Prozesses “ zeigt, wie dieser Prozess mit der Pyramidenstruktur des Wasserfallprozesses verglichen werden kann.

Abbildung 3.3. Zu erwartender Aufwand je Schritt des iterativen Prozesses



Das Wachsen der Popularität iterativer Prozesse ist mit dem Wachsen der OOA&D eng verknüpft. Es ist die saubere Kapselung von Objekten, die es erlaubt, einen Teil des Systems mit klar definierten Leerroutinen für den verbleibenden Code zu bauen.

3.2.1.2.1. Der Rational Unified Process

Der wahrscheinlich bekannteste, iterative Prozess ist der Rational Unified Process (RUP) von Rational Software (www.rational.com [http://www.rational.com]).

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 spätere Iterationen sehr viel Aufwand in der Design- und Buildphase benötigen.

RUP erkannte, dass Iterationen entsprechend Ihrer Lage im Gesamtprojekt in *Phasen* aufgeteilt werden können. Jede Phase kann eine oder mehrere Iterationen umfassen.

- 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

Es sollte klar sein, dass Testen ein integraler Bestandteil aller Phasen ist. Gerade in den frühen Phasen sollten die Anforderungen und das Design getestet werden. Und dies wird durch ein gutes CASE-Tool unterstützt.

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

3.2.1.2.2. Iterationsgröße

Eine gute Regel ist es, dass eine Iteration in typischen kommerziellen Projekten zwischen sechs und zehn Wochen umfassen sollte. Bei längeren Zeiträumen haben Sie sich wahrscheinlich mehr Anforderungen vorgenommen, als Sie in einem Schritt verarbeiten können. Sie verlieren auch den Fokus auf den nächsten vollständigen Iterationsschritt. Bei kürzeren Zeiten haben Sie wahrscheinlich nicht genügend Anforderungen um einen spürbaren Fortschritt zu erzielen. In diesem Fall wird der mit einer Iteration verbundene zusätzliche Overhead zum Problem.

Die Gesamtanzahl von Iterationen hängt von der Grösse des Projektes ab. Nehmen Sie die voraussichtliche Gesamtdauer (geschätzte Ausarbeitungzeit für das gesamte Subjekt) und unterteilen diese in 8 Wochen grosse Abschnitte. Die Erfahrung sagt, dass Iterationen im Verhältnis 1:2:3:3 bezüglich der RUP-Phasen „Anfangsphase“, „ Ausarbeitungsphase“, „Konstruktionsphase“ und „Deploymentphase“ aufgeteilt werden sollten. Ein Projekt, dass grosse Unsicherheiten in seiner Spezifikation aufweist (einige Forschungsprojekte zum Beispiel) wird viel grössere frühe Phasen aufweisen.

Wenn Sie ein Produkt auf Basis eines Vertrages für einen Kunden erstellen, ist der Endpunkt wohl definiert. Wenn Sie jedoch ein neues Produkt für den Markt entwickeln, kann die Strategie verwendet werden, das Datum der Produktveröffentlichung zu nehmen und den Zeitpunkt für das Ende des Engieering zu bestimmen (einige Zeit vorher). Dieser Zeitraum wird dann in Iterationen unterteilt. In dieser Zeit wird so viel von dem Produkt entwickelt wie möglich. Der iterative Prozess ist sehr effektiv, wenn der Zeitpunkt der Markteinführung wichtiger ist, als die exakte Funktionalität.

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.

OOA&D mit seinen klaren Objektgrenzen unterstützt diesen Ansatz natürlich. Eine OOA&D mit rekursiver Entwicklung wird manchmal als OOA&D/RD abgekürzt.

Die rekursive Entwicklung kann parallel zu Wasserfall- oder iterativen Prozessen angewendet werden. Sie ist keine Alternative zu diesen Prozessen.

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

Um unsere Anforderungen zu erfassen, verwenden wir das UML-Konzept der *Anwendungsfälle*. Beginnend mit einem *Visions-Dokument* werden wir sehen, wie Anwendungsfälle entwickelt werden können, um alle Aspekte des Systemverhaltens im Problembereich zu beschreiben.

3.2.2.2. Analyse

Während der Analyse werden wir in das UML-Konzept der *Klassen* einführen. Dieses Konzept erlaubt es uns, eine Top-Level-Sicht von Objekten zu erzeugen, die die Lösung darstellen - manchmal auch als *Konzept-Diagramm* bekannt.

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.

Obwohl es kein zwingend erforderlicher Teil unseres Prozesses ist, werden wir uns ansehen, wie das UML- *Kollaborationsdiagramm* als Alternative oder Ergänzung zum *Sequenzdiagramm* verwendet werden kann. Desweiteren werden wir uns das UML- *Aktivitätsdiagramm* als Alternative oder Ergänzung zum Zustandsdiagramm ansehen.

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

3.2.2.4. Build

UML ist nicht wirklich mit dem Schreiben von Code verknüpft. Jedoch wollen wir an dieser Stelle zeigen, wie ArgoUML für die Codegenerierung eingesetzt werden kann.

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, evaluieren, 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*

  Eine Design-Visualisierungstheorie innerhalb der kognitiven Psychologie. Die Theorie beschreibt, dass Designer eine Brücke zwischen Ihrem mentalen Modell des Problemes oder der Situation und dem formalen Modell einer Lösung oder eines Systems überwinden müssen.

  Diese Theorie besagt, dass Programmierer Vorteile haben von:

  1. Mehreren Darstellungsweisen, wie die syntaktische Programmzerlegung, Zustandsübergänge, Steuer- und Datenfluss. Diese erlauben es dem Programmierer, die Elemente und Beziehungen innerhalb des Problemes und der Lösung besser zu identifizieren und daher die Abbildung zwischen deren Situationsmodellen und den funktionierenden Systemmodellen leichter herstellen zu können.

  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.

1. Das Design der Anwenderschnittstelle, die es dem Anwender erlaubt, das Design aus unterschiedlichen Sichten zu betrachten und es dem Anwender ermöglicht, Ziele über eine Anzahl alternativer Wege zu erreichen.

2. Die Verwendung von parallel ablaufenden Prozessen im Designtool, die das aktuelle Design gegen „best practice"-Entwurfsmodelle überprüfen. Diese Prozesse sind als *Design-Hinweise* bekannt.

3. Die Nutzung von *„zu bearbeiten Listen"* , die dem Anwender Vorschläge aus den Design-Hinweisen unterbreiten und es ihm auch erlauben, Bereiche für künftige Aktionen aufzuzeichnen.

4. Die Verwendung von Checklisten, um den Anwender durch einen komplexen Prozess zu führen.

3.3.2. Offene Standards

Die UML ist selbst ein offener Standard. ArgoUML hat überall versucht, offene Standards für alle seine Schnittstellen zu verwenden.

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 Diagramm- Layout verloren ging. ArgoUML umgeht dies, indem es die grafische Information separat vom Modell speichert. (siehes Abschnitt 3.4.3.1, „ Laden und Speichern “).

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

- Die *Encapsulated PostScript (EPS)* [http://en.wikipedia.org/wiki/Encapsulated_PostScript]-Datei weist zusätzliche Beschränkungen auf. Diese Beschränkungen sind dazu gedacht, es der Software einfacher zu machen, eine EPS-Datei in ein anderes PostScript-Dokument einzubetten.

- Das *Graphics Interchange Format (GIF)* [http://en.wikipedia.org/wiki/GIF] verwendet eine verlustlose Kompression und erhält scharfe Kanten, wodurch es für ArgoUML gut geeignet ist. Das GIF-Format ist ein patentgeschütztes Format. Aber aktuell sind alle Patente abgelaufen.

- *Precision Graphics Markup Language (PGML)* [http://en.wikipedia.org/wiki/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.

- *Portable Network Graphics (PNG)* [http://en.wikipedia.org/wiki/PNG] ist ein ISO/IEC-Standard (15948:2004) und auch eine W3C- Empfehlung. PNG ist ein Bitmap-Bildformat, das eine verlustlose Datenkompression verwendet. PNG wurde eingesetzt, um das GIF-Format durch ein Bildformat zu ersetzen, das die Nutzung einer Patentlizenz entbehrlich macht. PNG wird offiziell "ping" ausgesprochen, wird aber auch sehr oft buchstabiert - wahrscheinlich, um eine Verwechslung mit dem Netzwerktool ping zu vermeiden. PNG wird durch die libpng-Referenzbibliothek unterstützt, eine plattformunabhängige Bibliothek, die C-Funktionen für die Anwendung von PNG-Bildern enthält.

- *PostScript (PS)* [http://en.wikipedia.org/wiki/PostScript/] ist eine Seitenbeschreibungs- und Programmiersprache, die primär in elektronischen und Desktop-Publishing-Bereichen eingesetzt wird.

- *Scalable Vector Graphics (SVG)* [http://en.wikipedia.org/wiki/Scalable_Vector_Graphics] ist eine XML-Markup-Sprache für die Beschreibung zweidimensionaler Vektorgrafiken, die sowohl statisch als auch animiert und entweder deklarativ oder beschreibend. Sie ist ein offener Standard, der durch das World Wide Web Konsortium erstellt wurde. Der Verwendung von SVG im Web befindet sich noch im Anfangsstadium. Es gibt eine grosse Trägheit aufgrund der langjährigen Nutzung reiner Raster- und anderer Formate wie Macromedia Flash oder Java-Applets. Aber auch die Browserunterstützung ist uneinheitlich, mit eingebauter Unterstützung in Opera und Firefox, während Safari und der Internet Exlorer ein Plugin benötigen. Siehe PGML oben.

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

Die *Object Constraint Language (OCL)* [[http://en.wikipedia.org/wiki/Object_Constraint_Language] 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

Java wurde als interpretierende Sprache erdacht. Es ist kein Compiler, der Code für jede beliebige Zielmaschine erzeugt. Sie übersetzt den Code für sein eigenes Ziel, der *Java Virtual Machine (JVM)*.

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.

Der Vorteil ist, dass ein kleines Projekt wie ArgoUML immer für zusätzliche Unterstützung offen ist. Insbesondere für diejenigen, die Ihre Ideen einbringen, wie das Programm verbessert werden kann. Jederzeit können 10, 15, 20 oder mehr Personen signifikante Beiträge zu ArgoUML leisten. Dies kommerziell zu machen, würde mehr als 1 Mio. $ pro Jahr kosten.

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!

Open Source schliesst das Geld verdienen nicht aus. Gentleware www.gentleware.com [http://www.gentleware.com] verkauft eine kommerzielle Version von ArgoUML: Poseidon. Deren Angebot ist nicht der Code. Es ist der kommerzielle Support, der das Risiko bei der Nutzung von ArgoUML in einer kommerziellen Entwicklung reduziert und es dem Kunden erlaubt, die Vorteile von ArgoUML's führender Technologie zu nutzen.

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

Da ArgoUML in 100% reinem Java geschrieben ist, sollte es auf jedem Rechner mit installiertem Java laufen. Es wird Java, Version 5 oder höher benötigt. Möglicherweise ist es bereits installiert, aber falls nicht, können Sie es von www.java.com [http://www.java.com] frei herunterladen. Beachten Sie, dass Sie nur die Java Runtime Umgebung (JRE) benötigen. Es gibt keine Notwendigkeit das gesamte Java Development Kit (JDK) herunterzuladen.

ArgoUML benötigt eine vernünftige Rechenleistung. Jeder PC, der in der Lage ist ein Betriebssystem mit einer grafischen Benutzeroberfläche auszuführen ist dazu in der Lage. Laden Sie den Code über den Download-Bereich der Projekt-Webseite argouml.tigris.org [http://argouml.tigris.org] herunter. Wie im nächsten Abschnitt beschrieben, wählen Sie die Version aus, die Ihren Ansprüchen am Besten genügt.

3.4.1.2. Download-Möglichkeiten

Sie haben ein paar unterschiedliche Optionen, wie Sie ArgoUML bekommen können.

1. Sie starten ArgoUML direkt von der Web Seite mit Hilfe von Java Web Start. Dies ist die einfachste Möglichkeit für den gelegentlichen Einsatz.

2. Laden Sie das Windows-Installationsprogramm herunter. Dies ist die richtige Option, wenn Sie mit Windows arbeiten und beabsichtigen ArgoUML normal zu nutzen.

3. Sie laden den binären, ausführbaren Code herunter. Dies ist die richtige Option, wenn Sie beabsichtigen, ArgoUML normal zu nutzen. Es sei denn, Sie arbeiten unter Windows.

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 “).

Alle vier Optionen sind über die Projekt-Webseite argouml.tigris.org [http://argouml.tigris.org] frei verfügbar.

3.4.1.3. ArgoUML mit Java Web Start aufrufen

Hierfür sind zwei Schritte auszuführen.

1. Sie installieren Java Web Start auf Ihrem Rechner. Java Web Start ist unter java.sun.com/products/javawebstart [http://java.sun.com/products/javawebstart] , oder über den `Java Web Start`-Link auf der ArgoUML Home page [http://argouml.tigris.org] verfügbar.

2. Sie klicken auf den Link `Launch latest stable release` auf der ArgoUML Home page [http://argouml.tigris.org].

Java Web Start wird ArgoUML herunterladen, zwischenspeichern und das erste Mal starten. Bei darauf folgenden Starts prüft es, ob ArgoUML seitdem aktualisiert wurde, lädt nur die aktualisierten Teile herunter und startet es. Die ArgoUML Home page [http://argouml.tigris.org] enthält weitere Details, wie man ArgoUML mit der Java Web Start-Konsole startet.

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.

ArgoUML steht als `.zip` oder `tar.gz` Variante bereit. Wählen Sie die erste Variante, wenn Sie Microsoft Windows Anwender sind und die letzte, wenn Sie mit einer UNIX-Variante arbeiten. Es gibt auch eine Mac OS X Version mit `.app.tgz` -Erweiterung. Das Entpacken erfolgt wie nachfolgend beschrieben.

- Unter Windows. Entpacken Sie die `.zip`-Datei mit WinZip oder mit Windows, wenn Ihre Version dies unterstützt. Kopieren Sie die Dateien aus dem komprimierten Verzeichnis in ein Verzeichnis Ihrer Wahl.
- Unter Unix. Nutzen Sie GNU tar, um die Dateien in ein Verzeichnis Ihrer Wahl zu entpacken.

  `tar zxvf <file>.tar.gz`.

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

  ```
  gunzip < file.tar.gz | tar xvf -.
  ```

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

Wenn das Herunterladen vollständig abbricht und Sie keine lokale Unterstützung haben, versuchen Sie es auf der Webseite FAQ [http://argouml.tigris.org/faqs/users.html]. Wenn nicht schon Ihr Problem löst, versuchen Sie es über die ArgoUML Mailingliste für Anwender.

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
  ```

  ein. Diese Methode hat den Vorteil, dass die Fortschritts- und Debug-Informationen im DOS-Fenster angezeigt werden. Als Alternative erstellen Sie eine Batch- Datei (.bat), die das oben angeführte Kommando enthält und verlinken diese mit dem Desktop. Die Batchdatei sollte mit einer „pause“-Anweisung enden, für den Fall, dass Debug-Informationen während der Ausführung erzeugt werden. Auf einigen Systemen funktioniert auch ein einfacher (Doppel)- Klick auf die `argouml.jar`-Datei. Auf anderen wird damit ein Zip-Tool aufgerufen. Sehen Sie in der Beschreibung zum Betriebssystem oder in der Hilfe nach, wie dies zu konfigurieren ist.

- 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).

- Falsche Sprache. Wenn das Produkt in einer Sprache startet, die sie nicht lesen können oder wollen, gehen Sie in das zweite Menü von links. Wählen Sie den untersten Menüeintrag aus. Abbildung 3.5, „ Die Sprache unter Erscheinungsbild einstellen “ zeigt dies auf russisch. Dann klicken Sie auf den Reiter „Erscheinungsbild “. Öffnen Sie die Drop-Down-Liste Abbildung 3.5, „ Die Sprache unter Erscheinungsbild einstellen “ und wählen Sie eine Sprache aus. Beachten Sie, dass die Namen der Sprache in der jeweiligen Sprache angezeigt werden. Die dargestellte, ausgewählte Sprache ist Deutsch. Damit die Änderungen wirksam werden, müssen Sie ArgoUML beenden und neu starten. Verwenden Sie die Taste X oben rechts.

Abbildung 3.4. Den Einstellungs-Assistenten finden

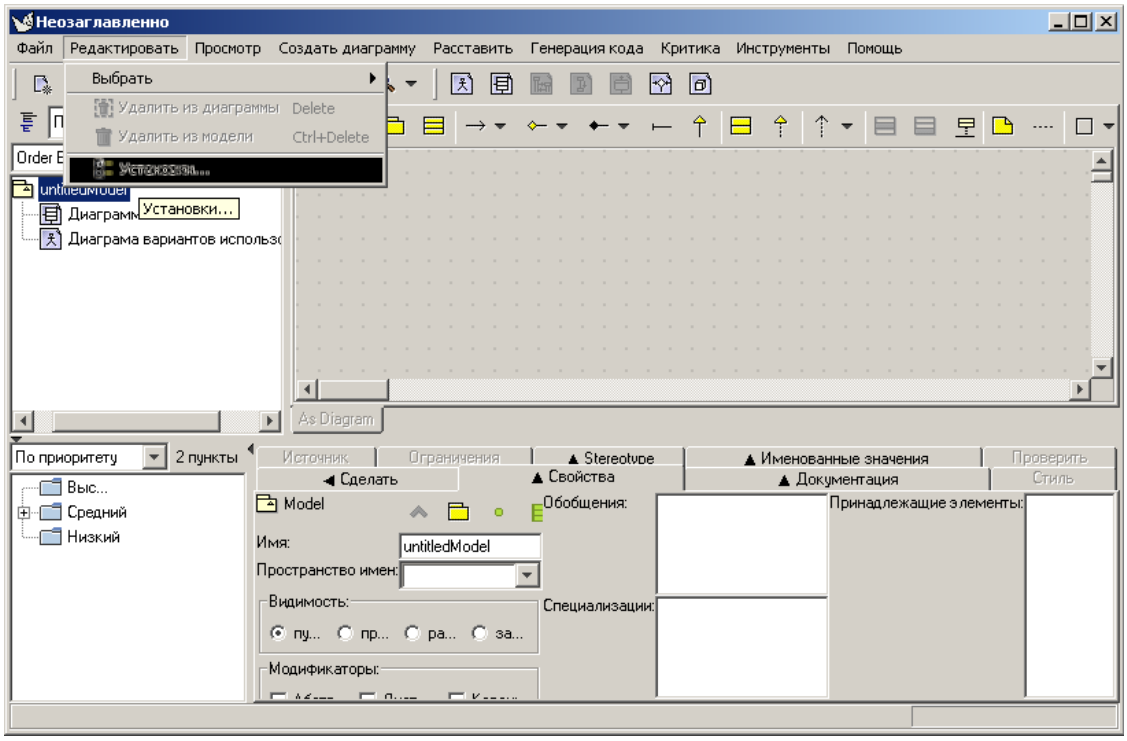
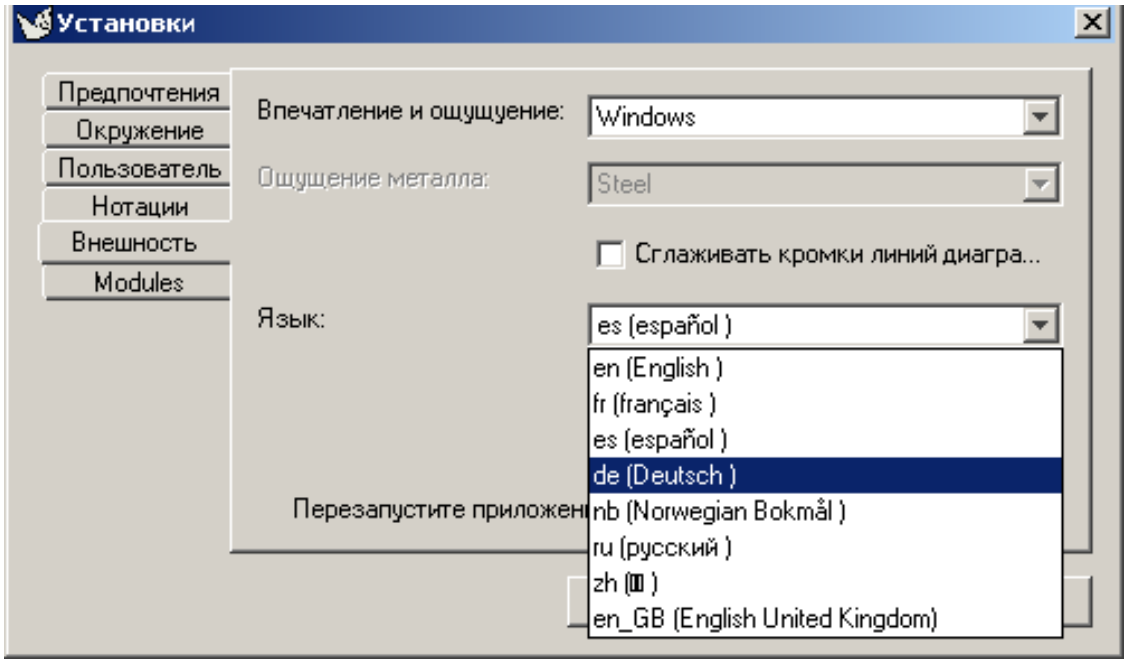


Abbildung 3.5. Die Sprache unter Erscheinungsbild einstellen



3.4.2. Die ArgoUML-Anwenderschnittstelle

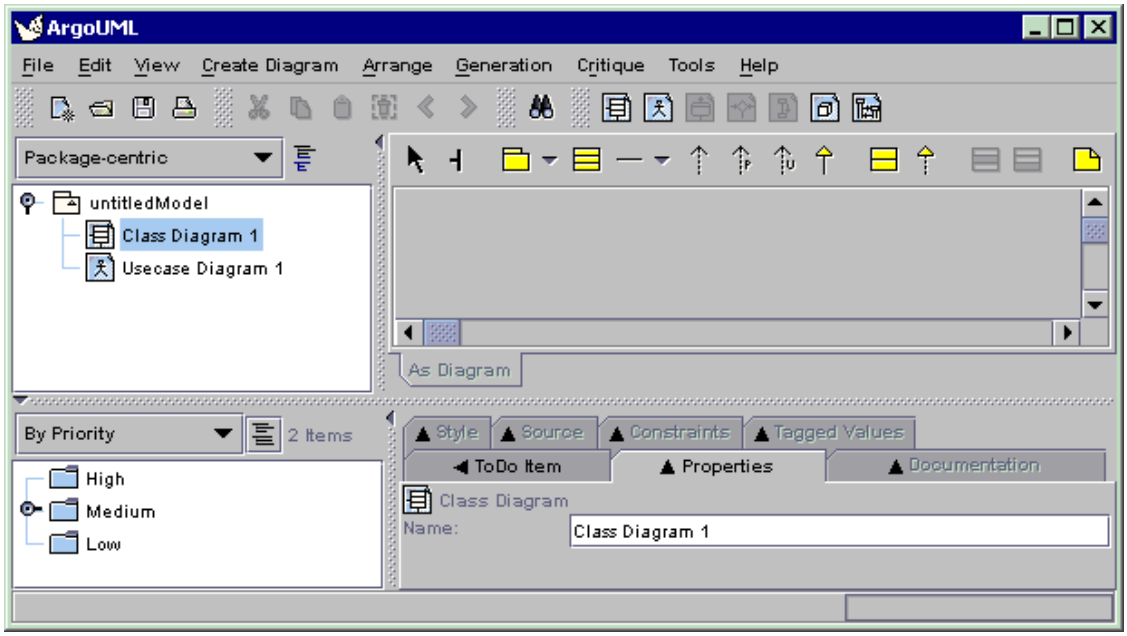
Bevor Sie mit der Fallstudie beginnen, müssen Sie sich noch mit der Anwenderschnittstelle vertraut machen. Lesen Sie hierfür die Einleitung in der Referenz zur Anwenderschnittstelle. Siehe Kapitel 8,

Einleitung . Sie sollten auch Abschnitt 8.2, „ Generelles Verhalten der Maus in ArgoUML “ lesen.

Wenn Sie diese Übungsanleitung durchgehen, wird Ihnen erläutert, was Sie tun müssen. Wie Sie es tun müssen, steht oftmals in der Referenz zur Anwenderschnittstelle. Zu diesem Zeitpunkt ist es nicht notwendig, dass Sie die gesamte Referenz durchlesen. Aber Sie sollten tief genug eingestiegen sein, damit Sie Dinge in der Referenz finden können. An den entsprechenden Stellen in der Übungsanleitung wird der Versuch unternommen, Sie direkt zu den entsprechenden Referenz-Abschnitten zu führen.

Abbildung 3.6, „ Das erste ArgoUML-Fenster “, zeigt das ArgoUML-Fenster, wie es beim ersen 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.

Menü: *Datei*. Die Elemente des Standard-Datei-Menü's enthalten keine Überraschungen. Wenn sie benötigt werden, werden wir sie einfach verwenden, ohne vorher zu zeigen wie sie arbeiten. Eine Vielzahl anderer, ArgoUML-spezifische Aktionen sind hier verfügbar. Jetzt werden wir sie allerdings überspringen.

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 in der kommenden Dateiauswahl die "DeleteThis.xmi"-Datei, die Sie gerade weggeschrieben haben. Überprüfen Sie den Namen des importierten Modelles 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.

4. *Datei=>(Alle) Grafik(en) exportieren...* . Markieren Sie im Explorer-Fenster eines der Diagramme. Entweder das "Klassendiagramm 1" oder "Anwendungsfalldiagramm 1" (dabei wurde unterstellt, dass Sie diese nicht umbenannt oder gelöscht haben). Führen Sie *Datei=>Grafik exportieren...* aus. Wenn sich die Dateiauswahl öffnet, steht der letzte gespeicherte Name im Feld Dateiname (auch von einen Projekt, das nicht mehr geöffnet ist). Die Dateiauswahl ermöglicht es Ihnen, eine Reihe von Formaten auszuwählen. Öffnen Sie das Kombinationsfeld Dateityp und treffen Sie Ihre Wahl. Brechen Sie ab, wenn nichts sinnvolles zum Speichern vorhanden ist. Führen Sie *Datei=>Alle Grafiken exportieren...* aus. Beachten Sie, dass Sie zum jetzigen Zeitpunkt keinen Dateinamen und kein Dateiformat auswählen können. ArgoUML wird Ihnen nur die Eingabe eines Verzeichnisses erlauben. ArgoUML wird dann eine Datei für jedes Ihrer Diagramme erzeugen. Der Diagrammname wird der Dateiname und die Dateierweiterung wird durch das Standard-Grafik-Format bestimmt. Obwohl Sie keine Dateinamen im Browserfenster auswählen können, können Sie einen Dateinamen in das Editierfeld eingeben. Aber wenn Sie das tun, wird nichts passieren. Sie werden mehr über das Standard-Grafik-Format erfahren, wenn wir zum *Bearbeiten*-Menü gehen.

5. *Datei=>Notation.* Wir gehen jetzt ein Stück weiter und erstellen ein kleines Klassendiagramm, so dass wir sehen können, was es mit der Notation auf sich hat. Im Explorer markieren oder erzeugen Sie ein Klassendiagramm. Siehe Abschnitt 10.6, „ Das Menü "Neues Diagramm" “ und Abschnitt 12.4.3, „ Zeichen-Symbole “. Erstellen Sie eine Klasse im Diagramm. Gehen Sie in das Detailfenster und erstellen Sie ein Attribut in der Klasse. Siehe Abschnitt 18.6.2, „ Eigenschaftssymbolleiste Klasse “. Im Register Eigenschaften des Detailfensters ändern Sie die Kardinalität auf "1..*". Jetzt gehen Sie in das *Datei*-Menü und wählen *Notation* aus. Wechseln Sie zwischen UML und Java und beobachten Sie die Änderungen im Editierfenster.

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 Diagramm erstellen, wird dieser neue Schatten erscheinen.

Menü: *Bearbeiten*. Das Menü Bearbeiten sieht nicht so aus, wie Sie es von anderen Produkten gewohnt sind. Es gibt keine Aktionen "Ausschneiden", "Kopieren" oder "Einfügen". Alle Aktionen sind ArgoUML-spezifisch, so dass wir Sie alle im Detail behandeln.

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.

- Jede Klasse im Diagramm besteht aus drei vertikalen Abschnitten. Führen Sie einen Doppelklick auf den obersten Abschnitt aus und geben Sie einen Namen für die Klasse ein. Anschliessend betätigen Sie die Enter-Taste. Benennen Sie die Klassen jetzt mit "A", "B" und "C". Markieren Sie die Klasse A, dann die Klasse B und dann die Klasse C, entweder im Editierfenster oder im Explorer.

- Führen Sie den Befehl *Bearbeiten=>Markieren=>Zurück* aus. Jetzt sollte die Klasse B markiert sein. Führen Sie erneut *Bearbeiten=>Markieren=>Zurück* aus. Die Klasse A sollte jetzt markiert sein. Abschliessend führen Sie den Befehl *Bearbeiten=>Markieren=>Vorwärts* aus. Die Klasse B sollte erneut markiert sein.

- Führen Sie jetzt den Befehl *Bearbeiten=>Markieren=>Umkehren* aus. Jetzt sollten die Klassen A und C markiert sein. Führen Sie erneut den Befehl *Bearbeiten=>Markieren=>Umkehren* aus. Die Klasse B sollte wieder markiert sein.

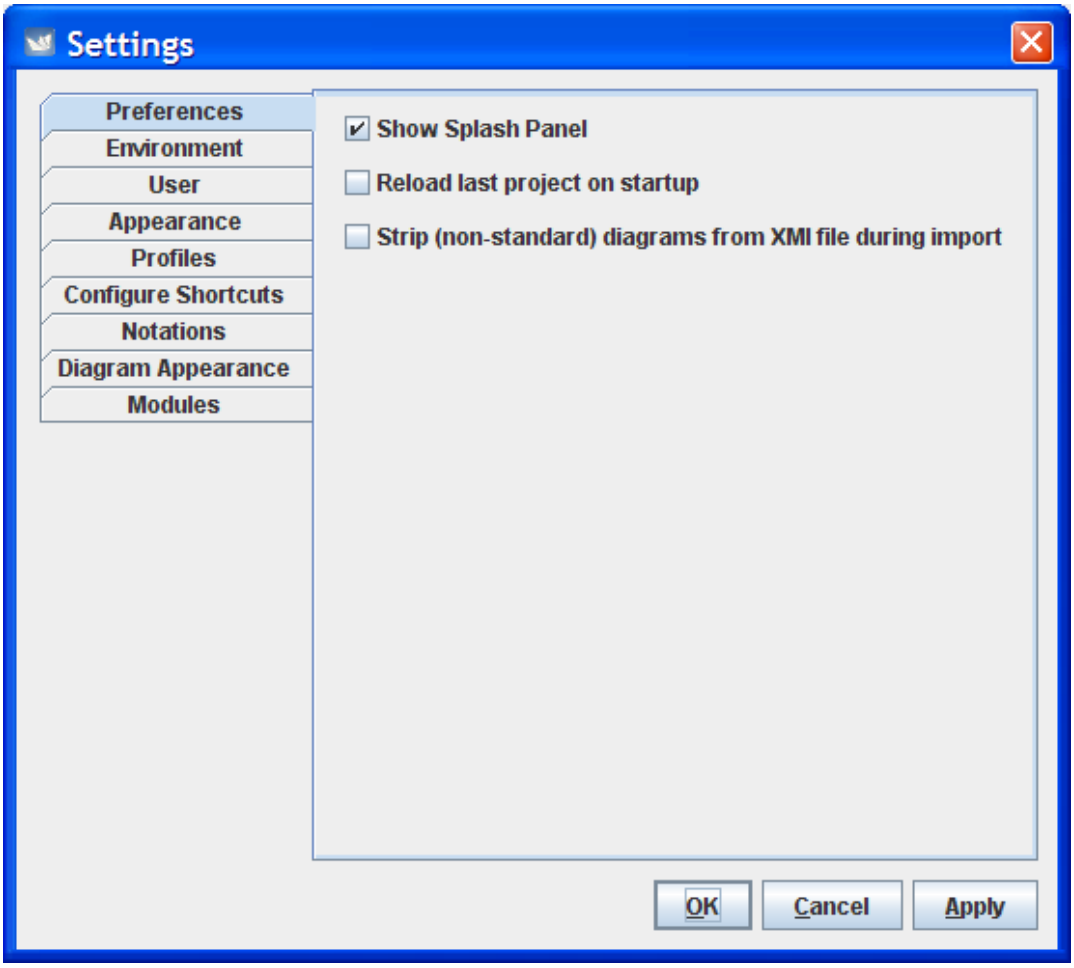
- Führen Sie den Befehl *Bearbeiten=>Markieren=>Aus Diagramm entfernen* aus. Beachten Sie, dass die Klasse B aus dem Diagramm verschwunden ist, aber im Explorer immer noch existiert.

- Markieren Sie die Klasse B im Explorer, klicken Sie mit der rechten Maustaste auf die Klasse und wählen Sie "Zum Diagramm hinzufügen" aus. Bewegen Sie den Cursor in das Editierfenster und klicken mit der linken Maustaste auf den Teil des Diagrammes, wo Sie glauben, dass die Klasse am Besten hinpasst. Sie sollten die Klasse an die Stelle zurückbringen, von der Sie sie vorher aus dem Diagramm entfernt haben. Führen Sie den Befehl *Bearbeiten=>Markieren=>Aus Modell entfernen* aus. Jetzt sollte die Klasse B sowohl aus dem Diagramm als auch aus dem Explorer verschwinden.

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



Bearbeiten=>Einstellungen...=>Voreinstellungen . Geben Sie Hilfe=>Über ArgoUML ein. Sehen Sie sich das Bild im Register Logo an. Dies wird als Startfenster bezeichnet. Gehen Sie in Bearbeiten=>Einstellungen...=> Voreinstellungen. Schalten Sie die Optionsschaltfelder "Startfenster anzeigen" und "Gemeinsame Klassen im Voraus laden" aus. Beenden Sie ArgoUML und starten Sie es erneut. Beachten Sie, dass das Startfenster währends des Hochfahrens nicht mehr erscheint.

4. *Bearbeiten=>Einstellungen...=>Umgebung* . Führen Sie den Befehl *Datei=>Grafik exportieren...* aus und beobachten Sie die Dateierweiterung im Dialogfeld "Dateityp" des Dateiauswahl-Dialoges. Gehen Sie in *Bearbeiten=>Einstellungen...=> Umgebung* und wählen einen anderen Wert für das Standard-Grafik-Format aus. Klicken Sie auf "Übernehmen" und dann auf "OK". Gehen Sie zurück in den *Datei=>Grafik exportieren...* -Dialog und beachten Sie, dass das neue Format jetzt der Standard ist.

5. *Bearbeiten=>Einstellungen...=>Benutzer* . Geben Sie Ihren Namen und Ihre E-Mailadresse ein.

6. *Bearbeiten=>Einstellungen...=>Erscheinungsbild* . Ändern Sie "Aussehen" in "Metall". Beachten Sie, dass das Dialogfeld "Metall-Thema" aktiviert wird. Ändern Sie das Thema in "Sehr grosse Schrift". Klicken Sie auf "Übernehmen" und dann auf "OK". Beachten Sie, dass nichts passiert ist. Beenden Sie ArgoUML und starten Sie es erneut. Die Darstellung sollte deutlich anders aussehen. Machen Sie Ihre Änderungen rückgängig. Nehmen Sie Werte, die Sie bevorzugen.

7. *Bearbeiten=>Einstellungen=>Profile*. Sehen Sie im Explorer unter "Profil-Konfiguration" nach.

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.

8. *Bearbeiten=>Einstellungen=>Tastenkürzel konfigurieren*. (Noch zu beschreiben).

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.

   In beiden tun Sie bitte folgendes. Schalten Sie alle Optionsschaltfelder ein. Führen Sie den Befehl *Datei=>Neu* aus und erstellen Sie eine Klasse in einem Klassendiagramm. Erstellen Sie ein Attribut mit Hilfe eines Doppelklicks auf den Abschnitt Attribute. Erstellen Sie eine Operation mit Hilfe eines Doppelklicks auf den Abschnitt Operationen. Beobachten Sie die Unterschiede in den Darstellungen.

10. *Bearbeiten=>Einstellungen=> Diagramm-Darstellung*. Diese Einstellung ermöglicht es, die in den Diagrammen verwendete Schriftart und Schriftgröße zu ändern. Deshalb hat diese Einstellung keinen Einfluss auf die in der Benutzeroberfläche von ArgoUML verwendete Schriftart. Die Standard-Schriftart für ArgoUML ist Dialog.

11. *Bearbeiten=>Einstellungen...=>Module*. Dies zeigt die aktuell geladenen Module. Wir wollen in dieser Version des Tutorials nicht weiter darauf eingehen.

Menü: *Ansicht*. Dieses Menü erlaubt Ihnen zwischen Diagrammen hin- und herzuschalten, Modellelemente in einem Modell aufzufinden, in ein Diagramm hineinzuzoomen, das Gitter einzustellen, die Darstellung des Seitenumbruches zu ändern und die XML-Darstellung des Projektes anzuzeigen. Um zu einem definierten Punkt zurückzukehren, führen Sie *Datei=>Neu...* aus. Erstellen Sie je Diagramm ein Beispiel und führen Sie noch keine Tätigkeit im Explorer aus. Um die Baumknoten zu expandieren, klicken Sie auf die (+)-Zeichen im Explorer. Markieren Sie das entsprechende Klassendiagramm und vergeben Sie einen Namen.

1. *Ansicht=>Gehezu 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 gehen Sie wieder in *Ansicht=>Gehezu 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=>Gehezu 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=>Gehezu 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 "Gehezu Diagramm"-Fenster. Beobachten Sie die Änderungen im Explorer.

2. *Ansicht=>Suchen....* Zu diesem Zeitpunkt sollten sich drei normale Klassen und eine Beziehungsklasse im Explorer befinden. Benennen Sie diese "AA", "AB", "B" und "C". Führen Sie eine *Ansicht=>Suchen...*- Operation aus. Klicken Sie auf die Schaltfläche "Suchen". Beachten Sie, dass ein Register "* in *" erzeugt wurde. Dieses Register sollte sehr viele neue Dinge zeigen. Ändern Sie im "Im Diagramm"-Editor "*" in "B*" und klicken Sie auf die Schaltfläche "Suchen". Beobachten Sie den Inhalt des neuen Registers "* in B*". Sie sollten drei Klassen, die Verbindung (eine Assoziation) und die Assoziationsklasse sehen. Markieren Sie in der Kombinationsliste "Element-Typ" das Element "Schnittstelle" und klicken Sie auf die Schaltfläche "Suchen". Im neuen Register "* in B* Inte..." sollten keine Einträge vorhanden sein, da wir keine Schnittstellen definiert haben. Markieren Sie in der Kombinationsliste "Element-Typ" das Element "Klasse" und klicken Sie auf die Schaltfläche "Suchen". Im neuen Register "* in B* Class" sollte ein Eintrag weniger vorhanden sein als im Register "* in B*". Wechseln Sie zwischen diesen beiden Registern hin und her und beobachten Sie die Unterschiede. Markieren Sie in einigen dieser Register ein Element und klicken Sie auf die Schaltfläche "Gehe zur Auswahl". Beobachten Sie die Änderungen im Diagramm und im Explorer.

3. *Ansicht=>Zoom.* Als Ausnahme zur generellen Regel arbeitet das Symbolleisten- Equivalent von *Ansicht=>Zoom* nicht auf die gleiche Art und Weise wie das entsprechende Menüelement. Durch das Hervorheben von *Ansicht=>Zoom* erscheint ein Untermenü, welches die Elemente "Vergrössern", "Rückgängig" und "Verkleinern" enthält. Klicken Sie mehrmals auf diese Elemente und beobachten Sie die Effekte im Diagramm. Anschliessend klicken Sie auf die Zoom-Schaltfläche in der Symbolleiste. Es erscheint ein Schieberegler, dessen Zeiger Sie bewegen können. Verschieben Sie den Zeiger und bewegen Sie ihn nach oben und nach unten und beobachten Sie dabei den Effekt auf das Diagramm.

4. *Ansicht=>Raster einstellen.*

5. *Ansicht=>Einrasten einstellen.*

6. *Ansicht=>Seitenumbrüche.*

7. *Ansicht=>XML Dump.*

Menü: *Neues Diagramm*. Das Menü erlaubt es Ihnen, jedes der sieben von ArgoUML unterstützten UML-Diagrammtypen (Klasse, Anwendungsfall, Zustands-, Aktivitäts-, Kollaborations-, Verteilungs- und Sequenzdiagramm) zu erstellen.

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

Menü *Werkzeuge*. Diese Menü ist dauerhaft deaktiviert, es sei denn es sind Werkzeuge in Ihrer Version von ArgoUML verfügbar.

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

Symbolleiste: Datei. Diese Symbolleiste enthält einige Symbole des Datei-Menü's.

Symbolleiste: Bearbeiten. Diese Symbolleiste enthält einige Symbole des Bearbeiten- Menü's.

Symbolleiste: Ansicht Diese Symbolleiste enthält einige Symbole des Bearbeiten- Menü's.

Symbolleiste: Neues Diagramm. Diese Symbolleiste enthält einige Symbole des Menü's Neues Diagramm.

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.

Markieren Sie entweder das Klassendiagramm 1 oder das Anwendungsfalldiagramm 1 und beobachten Sie, wie sich der Inhalt des Detailfensters sich je nach markiertem Element im Explorer ändert. Das Detail-Fenster ist im Kapitel 12 beschrieben. Zum jetzigen Zeitpunkt ist es nicht erforderlich das Kapitel 12 zu lesen. Aber, es kann auch nichts schaden.

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 Explorer das "Klassendiagramm 1". Der Name ist unwichtig. Wenn Sie ihn geändert haben, markieren Sie nun den neuen Namen. Wenn Sie es gelöscht haben, führen Sie zuerst eine Neues Diagramm=>Klassendiagramm-Aktion durch. Klicken Sie auf die Schaltfläche "Neues Paket" in der Werkzeugleiste des Editierfensters. Klicken Sie im Editierfenster irgendwohin. Beachten Sie, dass im Explorer ein Paket mit dem Namen (unbenannt Paket) erscheint.

Führen Sie einen Doppelklick in der Werkzeugleiste des Editierfensters auf der Schaltfläche "Neue Klasse" aus. Klicken Sie zuerst in das Paket und einmal ausserhalb des Paketes. Beachten Sie, dass im Explorer im Baum zwei Klassen beide mit dem Namen (unbenannt Klasse) erscheinen. Eine direkt mit dem Modellknoten und die andere mit dem Knoten (unbenannt Paket) verknüpft.

Klicken Sie auf die Schaltfläche Auswählen in der Werkzeugleiste des Editierfensters und Sie können im Editierfenster arbeiten ohne neue Klassen hinzuzufügen. Markieren Sie im Explorer die Klasse, die nicht dem Paket zugeordnet ist. Dies markiert die entsprechende Klasse im Diagramm. Greifen Sie sich diese Klasse und verschieben Sie sie in das Paket. Beachten Sie, dass diese Klasse jetzt im Explorer ebenfalls dem Paketknoten untergeordnet wurde.

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.
- Löst 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.

Genauso wie für UML-Modellelemente liefert die Werkzeugleiste des Editierfensters Zeichenobjekte (Rechtecke, Kreise, Linien, Polygone, Kurven, Text), um ergänzende Informationen in Diagramme

einbringen zu können.

3.4.2.6.1. Diagrammelemente bewegen

Es gibt verschieden 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 Hauptfenster ist in vier Fenster unterteilt. Darüber befinden sich das Menü und die Symbolleiste. Wenn wir mit dem Fenster oben links beginnen und im Uhrzeigersinn weitergehen, können Sie den Explorer, der eine Baumansicht Ihres UML-Modelles anzeigt, den Editor mit seiner Symbolleiste, zwei Scrollleisten und einer grauen Zeichenfläche, das Detailfenster mit markiertem "zu bearbeiten"-Register und das "zu bearbeiten" -Fenster mit einer Baumansicht der zu bearbeitenden Elemente, auf verschiedene Art und Weise sortiert, je nach Auswahl in der Drop-Down-Liste im oberen Bereich des Fenster.

Jedesmal wenn ArgoUML ohne Projektdatei als Argument gestartet wird, wird ein neues leeres Projekt erzeugt. Dieses Projekt enthält ein Modell mit dem Namen `unbenanntesModell` . Diese Modell enthält ein leeres Klassendiagramm, `Klassendiagramm 1` genannt und ein leeres Anwendungsfall-Diagramm, `Anwendungsfalldiagramm 1` genannt.

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 `einkaufsmodell` geben und Sie wollen es in einer Datei mit dem Namen `ErstesProjekt` speichern.

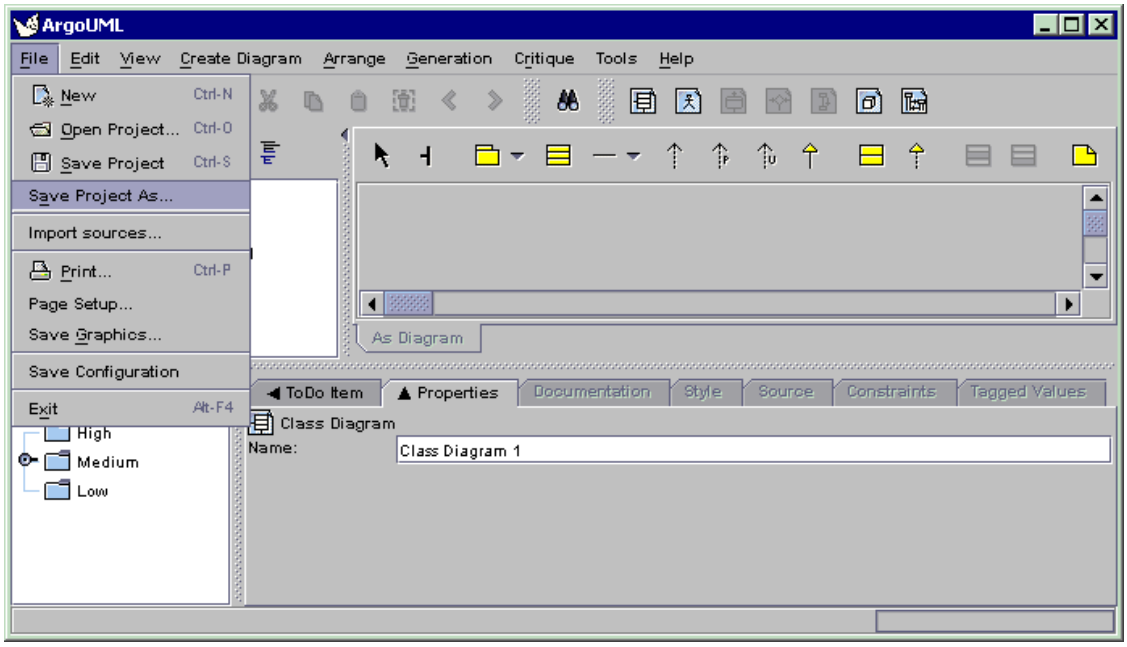
3.4.2.7.2. Ein Projekt speichern - Das Menü: Datei

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.

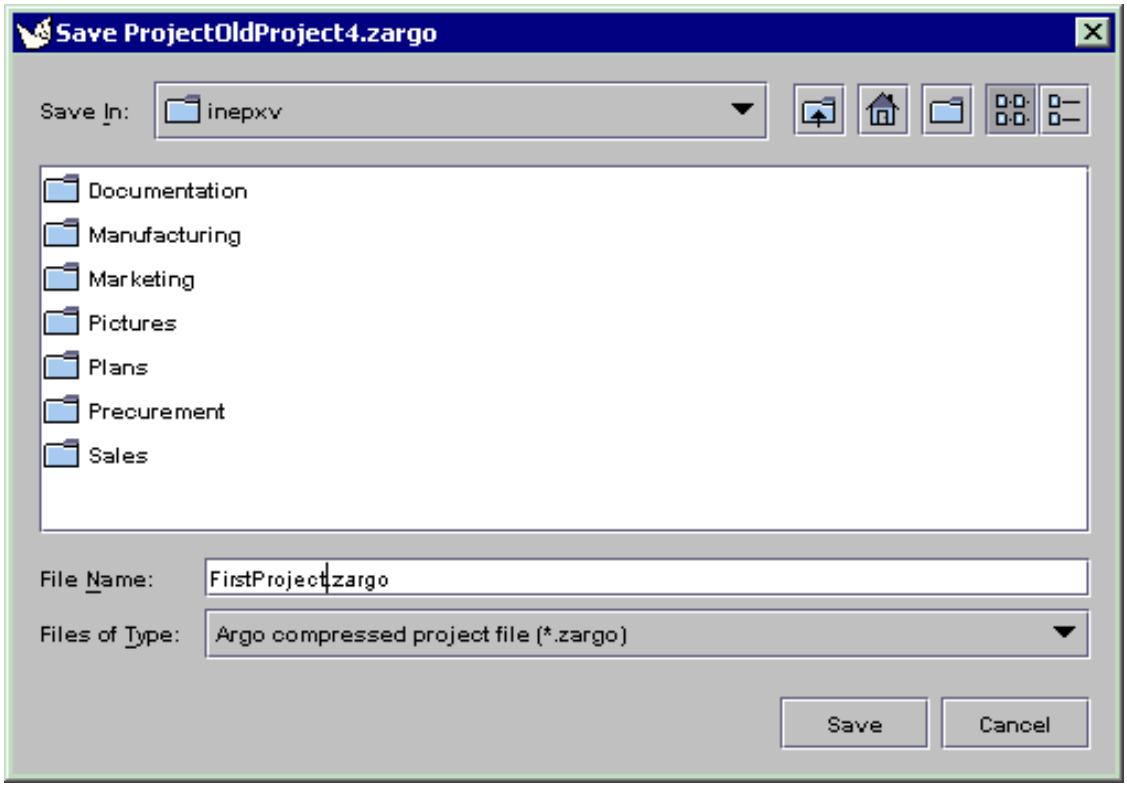
Einige diese Menü-Kommandos können auch mit Hilfe einer Tastenkombination, wie im Drop-Down-Menü dargestellt, ausgelöst werden. Das Festhalten der Taste „Strg“ und das Drücken der Taste „N“ wird zum Beispiel ein neues Projekt erzeugen.

In der aktuellen Version kann ArgoUML nur ein aktives Projekt gleichzeitig bearbeiten. Zusätzlich kann ein Projekt nur ein UML-Modell enthalten. Da ein UML-Modell eine unbegrenzte Anzahl von Elementen und Diagrammen beinhalten kann, sollte es keine schwerwiegenden Begrenzugen aufweisen, gerade für die Modellierung sehr grosser und komplexer Systeme.

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.

Die Hauptfunktion ist die scrollbare Verzeichnisliste in der Mitte des Dialoges. Mit Hilfe der Scrollleiste auf der rechten Seite können Sie sich in der Liste der im aktuell markierten Verzeichnis befindlichen Verzeichnisse nach oben und nach unten bewegen. Ob sie scrollbar oder nicht scrollbar ist, hängt von Menge der angezeigten Dateien und Verzeichnisse und deren Darstellung ab. Wenn alles in das Fenster passt, ist es nicht scrollbar, wie im Bild dargestellt.

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

Beachten Sie, dass nur Verzeichnisnamen und keine Dateinamen im Scrollbereich angezeigt werden. Tatsächlich ist der Dialog aktuell so eingestellt, dass er nur ArgoUML-Projektdateien mit der Erweiterung `.zargo` anzeigt. Dies können Sie im unteren Drop-Down-Steuerelement mit dem Namen `Dateityp:` sehen.

Beachten Sie auch, dass der aktuell markierte Verzeichnisname im oberen Drop-Down-Steuerlement 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- Auswahlbereich, 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.

- 

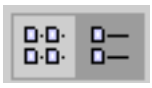
  Das Symbol: Verzeichnis-nach-oben. Klicken auf dieses Symbol wird Sie in das übergeordnete Verzeichnis des aktuellen Verzeichnisses bringen.

- 

  Das Symbol: Home-Verzeichnis. Klicken auf dieses Symbol wird Sie in Ihr Home-Verzeichnis bringen.

- 

  Das Symbol: Neues Verzeichnis. Klicken auf dieses Symbol wird ein neues Verzeichnis, genannt „Neues Verzeichnis“ unter dem aktuellen Verzeichnis erzeugen. Nachdem das Verzeichnis erzeugt wurde, markieren Sie es und klicken Sie auf den Namen. Dies erlaubt es uns, den Namen unserer Wahl zu vergeben.

- 

  Das Symbol: Darstellung der Verzeichnisse.

OK, jetzt navigieren wir in das Verzeichnis, in das wir unsere ArgoUML-Projektdatei speichern wollen. Füllen Sie `Dateiname:` mit dem entsprechenden Namen aus, wie z.B. `ErstesProjekt` und klicken Sie auf die Schaltfläche `Speichern`.

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“.

All diese Dateien werden in eine Datei mit der Erweiterung `.zargo` gepackt. Sie können die `.xmi` -Datei aus der `.zargo`-Datei mit Hilfe einer generischen ZIP- Anwendung extrahieren. Versuchen Sie es und blicken Sie in den Zauber von Argo.

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)

GEF ist ein Softwarepaket, welches die Grundlage für die im Editierfenster erscheinenden Diagramme bildet. GEF war ein integraler Bestandteil von ArgoUML, der jetzt separiert wurde. Wie ArgoUML ist es ein Open-Source-Projekt und via Tigris [http://www.tigris.org] verfügbar.

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/ [http://www.w3.org/TR/SVG/]).

In modernen Browsern ist die Unterstützung des Formates eingebaut, für ältere Browser können Sie aber auch ein Plugin von adobe.com [http://www.adobe.com] erhalten.

3.4.3.2.6. Diagramme als SVG speichern

1. Wählen Sie `.svg` als Dateityp aus.
2. Geben Sie den gewünschten Namen der Datei mit der `.svg`-Erweiterung am Ende ein. Beispiel: `meinumldiagramm.svg`

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.

Die meisten modernen Browser unterstützen SVG. Wenn Ihrer das nicht tut, versuchen Sie es unter Firefox [http://www.mozilla.com/firefox/] oder holen Sie sich ein Plugin für Ihren aktuellen Browser von adobe.com [http://www.adobe.com]

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 transferriert werden.

Es gibt auch ein Werkzeug, das XMI nach HTML konvertiert. Für mehr Informationen , siehe http://www.objectsbydesign.com/projects/xmi_to_html_2.html [http://www.objectsbydesign.com/projects/xmi_to_html_2.html].

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

ArgoUML-Versionen vor 0.19.7 unterstützen UML 1.3/XMI 1.0. Danach ist das Speicherformat UML 1.4/XMI 1.2, das nicht rückwärtskompatibel ist. Neuere Versionen von ArgoUML werden Projekte älterer Versionen lesen, aber nicht umgekehrt. Wenn Sie glauben, zu einer älteren Version von ArgoUML zurückkehren zu müssen, sollten Sie vorsichtig sein und ein Backup Ihres alten Projektes abspeichern.

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 Methodenrümpfe implementieren, um verwendbare Ergebnisse zu erzielen.

3.4.3.4.2. Code für Methoden generieren

Im Moment können Sie keinen Code für Methoden (Operationen) in ArgoUML schreiben. Das Quellcode-Fenster ist editierbar, aber die Änderungen werden ignoriert. ArgoUML ist aktuell ein reines Design-Werkzeug, es ist keine IDE-Funktionalität vorhanden, aber der Wunsch ist da. Sie können Forte und ArgoUML zusammenarbeiten lassen; dies ist ein guter Workaround.

Sie können uns hier helfen, wenn Sie wollen!

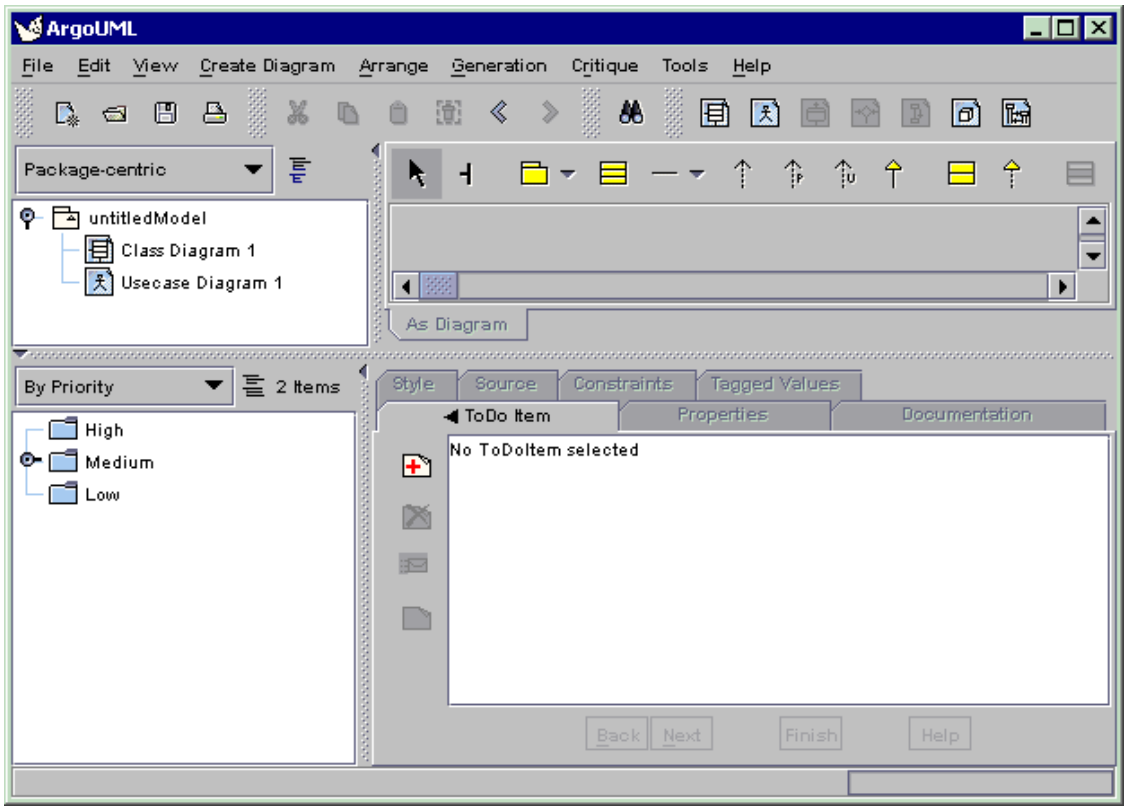
3.4.4. Arbeiten mit Design-Hinweisen

Design-Hinweise 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-Hinweisen

Wo stehen wir aktuell? Ein neues Projekt wurde erstellt und wurde in der Datei `ErstesProjekt.zargo` gespeichert. Abbildung 3.10, „ ArgoUML-Fenster mit gespeichertem Projekt `ErstesProjekt.zargo` “ zeigt, wie Ihr ArgoUML-Fenster zu diesem Zeitpunkt aussehen sollte.

Abbildung 3.10. ArgoUML-Fenster mit gespeichertem Projekt `ErstesProjekt.zargo`



Das Projekt enthält ein Paket auf oberster Ebene, genannt `unbenanntesModell`, 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.

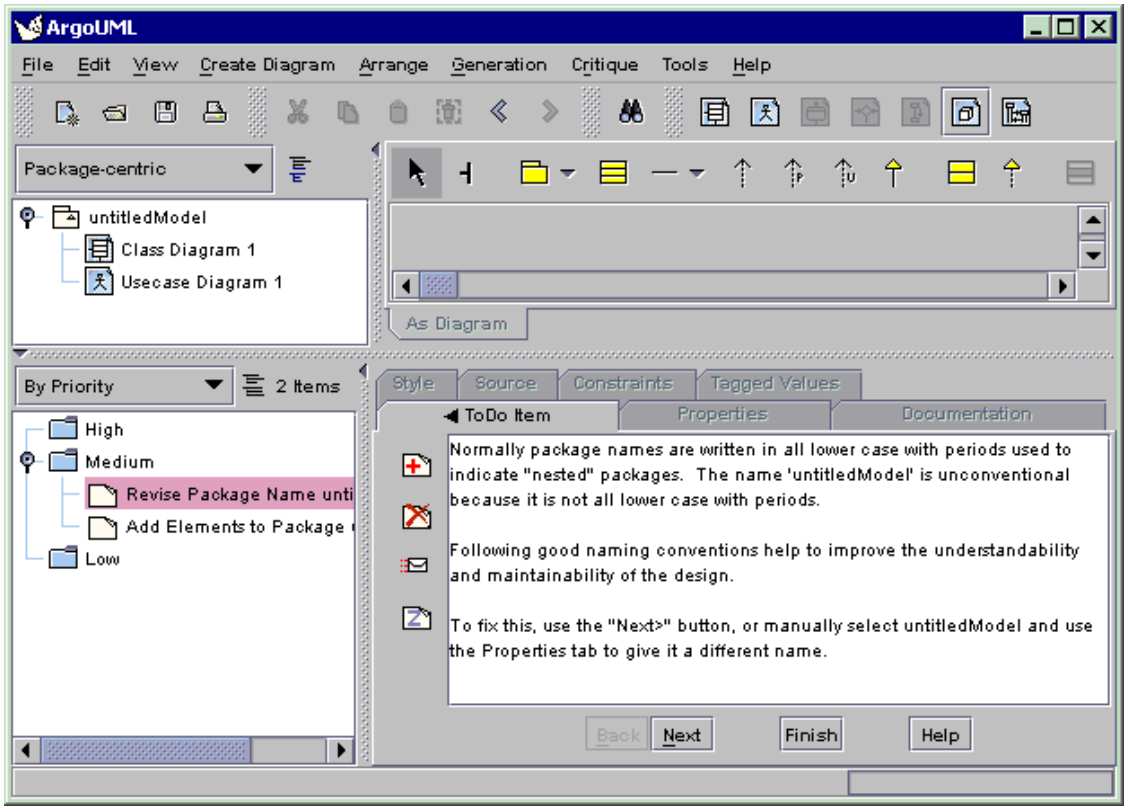
Aber, was ist dieses „zu bearbeiten“-Fenster überhaupt? Sie haben bis jetzt noch nichts aufgezeichnet, was zu tun wäre. Wo kommen diese "zu bearbeiten"-Elemente jetzt her?

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-Hinweis* 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.

Hinweise 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 Hinweise uns mitteilen. Klicken Sie auf das  Symbol in der Nähe des Verzeichnisses `Mittel` und klicken Sie auf das Element `Überprüfen Sie den Paketnamen unbenanntesModell`.

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

Abbildung 3.11. ArgoUML-Fenster zeigt das Hinweis-Element `Überprüfen Sie den Paketnamen unbenanntesModell`



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.

Was Ihnen ArgoUML wahrscheinlich mitteilen will, Paketnamen werden in Kleinbuchstaben geschrieben. Das oberste, von ArgoUML erzeugte Standard-Paket wird `unbenanntesModell` genannt und verletzt daher ein Designprinzip. (Aktuell kann dies als Bug von ArgoUML betrachtet werden, aber es kommt gerade gelegen, um die Arbeitsweise von Hinweisen zu demonstrieren).

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

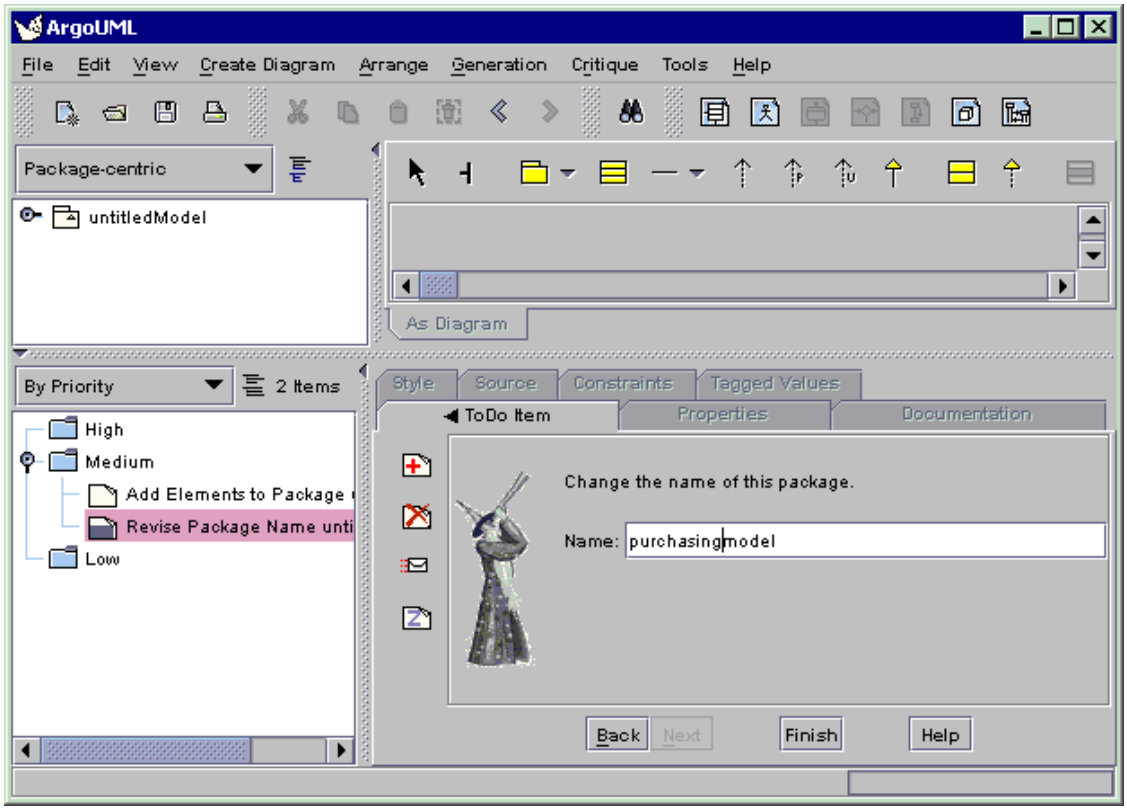
Wir werden nichts davon tun (wir kommen darauf zurück, wenn wir über die Design-Hinweise detaillierter sprechen), aber wir werden eine andere einfache Funktion von ArgoUML nutzen - die Autokorrektur-Funktion.

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-Hinweise bei der Arbeit: Der Paket-Umbenennungs-Assistent

Ersetzen Sie den Namen `unbenanntesModell` durch `einkaufsmodell` und klicken Sie auf die Schaltfläche `Fertig stellen`. Abbildung 3.12, „ Das ArgoUML-Fenster zeigt den Hinweis-Assistenten, zur Umbenennung des Paketes “ zeigt, wie das ArgoUML- Fenster nun aussehen sollte.

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



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

Wenn dies nicht sofort passiert, warten Sie einige Sekunden. ArgoUML macht ausgiebigen Gebrauch von mehreren, parallel ausgeführten Ausführungsthreads. Dies kann einige Sekunden Verzögerung verursachen bevor die Information auf dem Bildschirm auf den neusten Stand gebracht wird.

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

ArgoUML-Projekt anzusehen, werden Sie eine grosse Anzahl von "Code-" und Dokumentationszeilen, die Teil des Projektes sind, finden. Sie sind aber nicht Teil des Produktes. Beispiel: Dieses Dokument ist Teil des Produktes während das Kochbuch und die build.xml-Dateien nur Teil des Projektes sind. Mindestens die Dateistruktur des Projektes könnte in einem Paketdiagramm dargestellt werden.

...

Kapitel 4. Erfassen der Anforderungen

4.1. Einleitung

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

Der Schlüssel zu diesem Zeitpunkt ist, dass wir uns im Problembereich befinden. Zu diesem Zeitpunkt müssen wir alles aus der Perspektive des „Kunden“ und in der Sprache des „Kunden“ beschreiben.

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

Der einzige Teil der vom UML-Standard spezifizierten Anforderungsnotation ist das Anwendungsfalldiagramm. Der Rest ist prozessspezifisch. Der in diesem Kapitel beschriebene Prozess ist sehr stark an den Rational Unified Prozess angelehnt.

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.
3. Nachvollziehbarkeit aller an den zentralen Bankcomputer gesendeten Aktivitäten.

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.

Anwendungsfälle sollte signifikante Aktivitäten für das System sein. Der Nutzung des Geldautomaten durch den Kunden ist zum Beispiel ein Anwendungsfall. Die Eingabe einer PIN-Nummer ist es nicht.

Es gibt eine Grauzone zwischen diesen beiden Extremen. Wie wir sehen werden, ist es oft nützlich, sehr grosse Anwendungsfälle in kleinere Sub-Anwendungsfälle zu unterteilen. Wir können zum Beispiel Bargeld lagern, Bargeld auszahlen und Kontoabfragen als Sub- Anwendungsfälle definieren.

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 stellen dann die Beziehungen zwischen den Anwendungsfällen und Akteuren in einem oder mehreren Anwendungsfalldiagrammen dar. In grösseren Projekten wird mehr als ein Diagramm benötigt. Normalerweise werden Gruppen zusammengehöriger Anwendungsfälle in einem Diagramm dargestellt.

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 *Anwendungsfallszenario* 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.

4. Fassen Sie alle nicht-funktionalen Anforderungen in einer *Ergänzenden Anforderungs-Spezifikation* zusammen.

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.
- *Nutzer*. Die Nutzer (im weitesten Sinne) des Systems. Viele davon werden als Akteure, oder Steuer-Equipment das in Akteure umgewandelt wird, abgebildet.
- *Haupteigenschaften*. Auf einer sehr hohen Abstraktionsebene: Was sind die hauptsächlichen Aspekte des Problems/der gewünschten Lösung. Es ist hilfreich, hier bereits zu priorisieren.
- *Randbedingungen*. Eine Sicht auf die nicht-funktionalen Parameter des Systems auf einer sehr hohen Abstraktionsebene. Diese werden in der ergänzenden Anforderungsspezifikation detailliert ausgearbeitet.
- *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

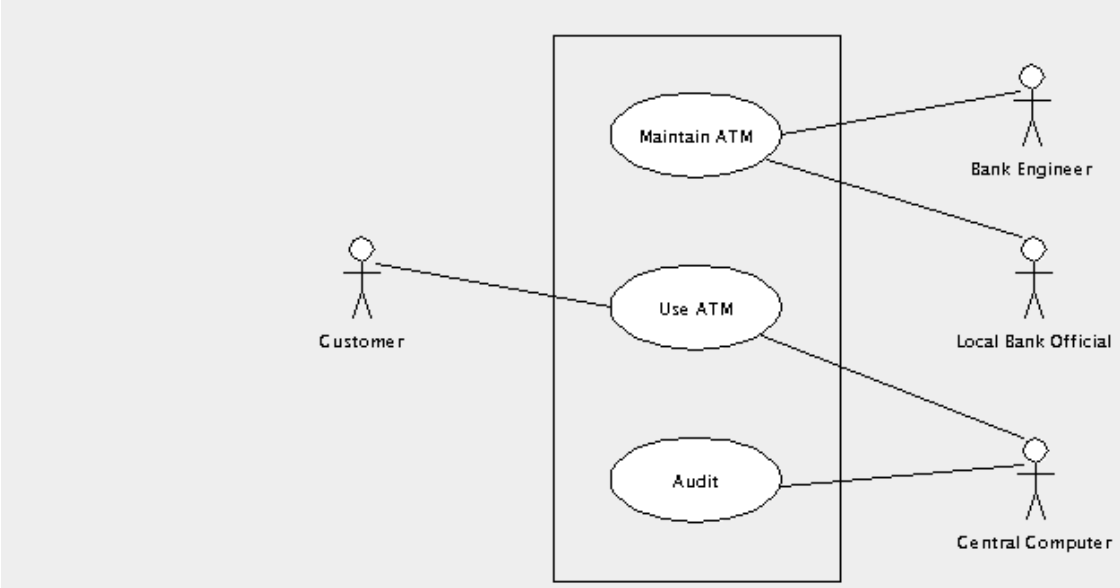
Das Visions-Dokument identifizierte die Anwendungsfälle und Akteure. Das Anwendungsfalldiagramm stellt dar, wie sie interagieren. In unserem ATM (Geldautomat) Beispiel haben wir „Kunde nutzt den Automaten“, „Wartung des Automaten“ und „Revision“ als die drei Hauptanwendungsfälle identifiziert. Als Akteure haben wir „Kunde“, „Wartungsingenieur“, „lokaler Zweigstellenbeamter “ und „Zentralcomputer“ identifiziert.

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

Aktive Akteure initiieren eine Interaktion mit dem System. Dies kann durch einen Pfeil in der Assoziation vom Akteur zum Anwendungsfall dargestellt werden. Im Geldautomaten-Beispiel ist der Kunde ein aktiver Akteur.

Die Interaktion mit *passiven* Akteuren wird durch das System initiiert. Dies kann durch einen Pfeil in der Assoziation vom Anwendungsfall zum Akteur dargestellt werden. Im Geldautomaten-Beispiel ist der Zentralcomputer ein passiver Akteur.

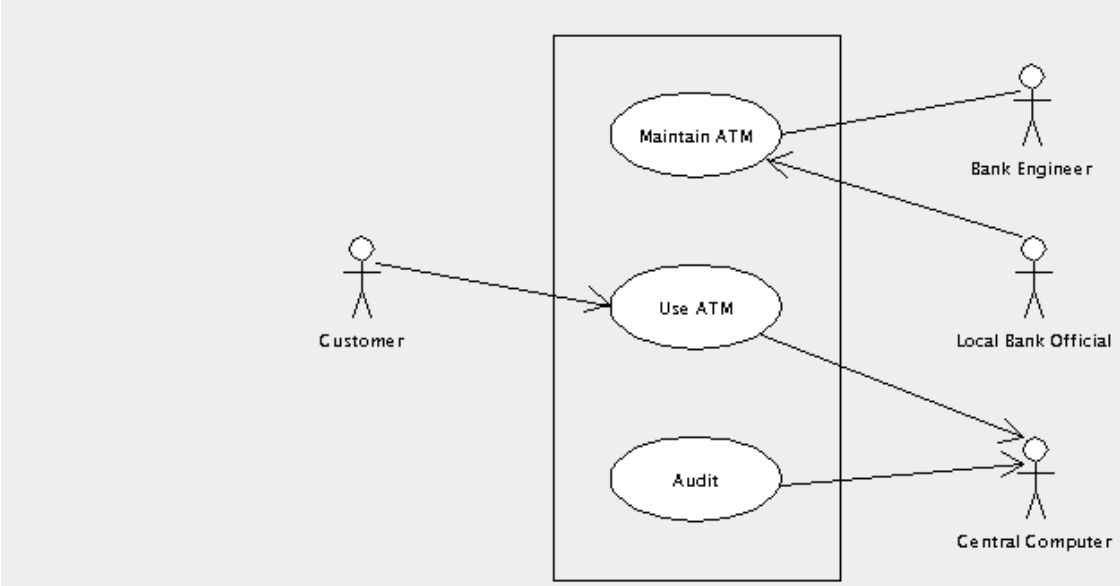
Dies ist ein gutes Beispiel, wo die Pfeile helfen, da es uns erlaubt, ein ereignisgetriebenes System (der Geldautomat initiiert die Interaktion mit dem Zentralcomputer) von einem zyklisch abfragenden System (der Zentralcomputer fragt den Geldautomaten von Zeit zu Zeit ab) zu unterscheiden.

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.

Die Verwendung von Pfeilen bei Assoziationen wird als *Navigation* der Assoziation bezeichnet. Wir sollten dies später in der UML im Einsatz sehen. Die Wahl der OMG eine bidirektionale Assoziation durch keine anstelle von zwei Pfeilspitzen darzustellen ist unvorteilhaft. Mit dieser Konvention können Sie nicht zwischen einer Assoziation unterscheiden, deren Navigation noch zu bestimmen ist und einer die bidirektional ist.

Abbildung 4.2, „ Anwendungsfalldiagramm für einen Geldautomaten mit Navigation. “ zeigt das Geldautomaten-Anwendungsfalldiagramm mit dargestellter Navigation.

Abbildung 4.2. Anwendungsfalldiagramm für einen Geldautomaten mit Navigation.



4.3.2.2. Kardinalität

Es kann sehr nützlich sein, die *Multiplizität* von Assoziationen zwischen Akteuren und Anwendungsfällen darzustellen. Damit meinen wir, wie viele Ausprägungen eines Akteurs mit wie vielen Ausprägungen eines Anwendungsfalles interagieren.

Standardmäßig nehmen wir an, dass eine Ausprägung eines Akteurs mit einer Ausprägung eines Anwendungsfalles interagiert. In den anderen Fällen können wird die Multiplizität an einem Ende der Assoziation kennzeichnen. Entweder mit einer Nummer, um darzustellen, wie viele Ausprägungen involviert sind, oder mit einem durch zwei Punkten separierten Bereich (`..`). Ein Stern (`*`) wird verwendet, um eine beliebige Zahl darzustellen.

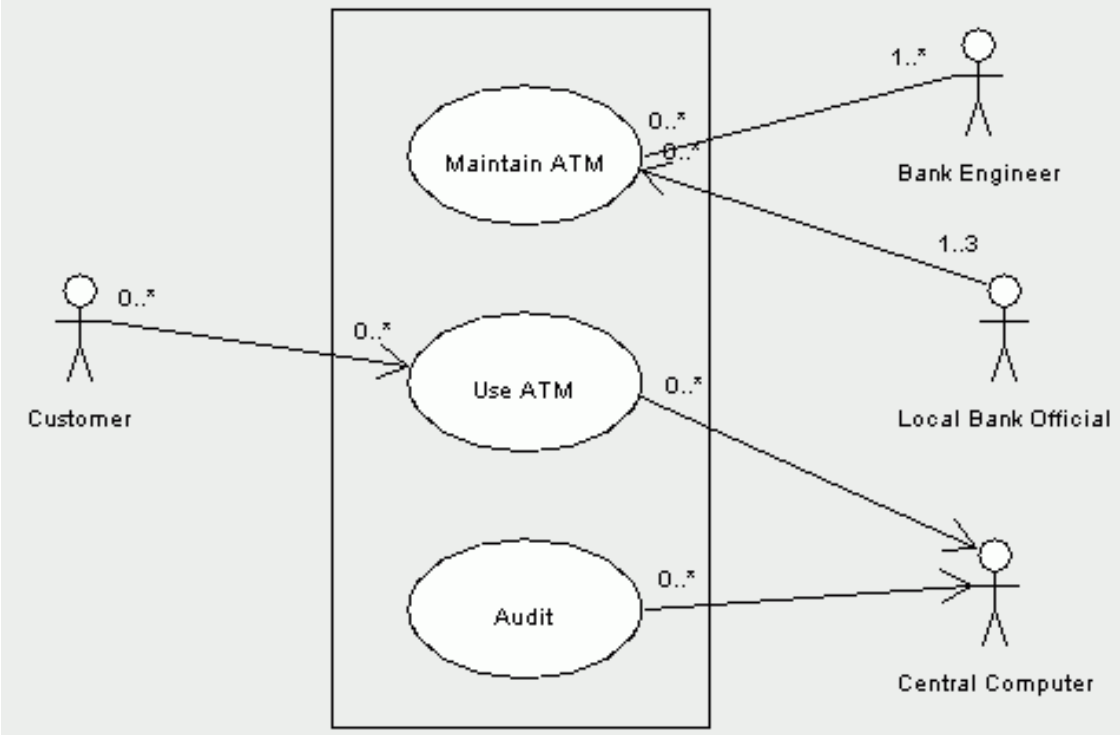
Im Geldautomaten-Beispiel gibt es nur einen Zentralcomputer, aber es können beliebig viele Nutzungen des Geldautomaten aufgezeichnet werden. Daher plazieren wir das Kennzeichen `0..*` am Ende des Anwendungsfalles. Es gibt keine Notwendigkeit für eine Kennzeichnung am anderen Ende, da der Standard 1 ist.

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 Anwendungsfall `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.

Abbildung 4.3. Anwendungsfalldiagramm für einen Geldautomaten und dargestellter Kardinalität.



Die Multiplizität kann ein Diagramm überladen und wird oft nicht angezeigt, ausser wo es für das Verständnis kritisch ist. Im Beispiel Geldautomat würde wir nur `1..3` zum lokalen Bankbeamten darstellen, da alle anderen aus dem Kontext heraus offensichtlich sind.

4.3.2.3. 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.

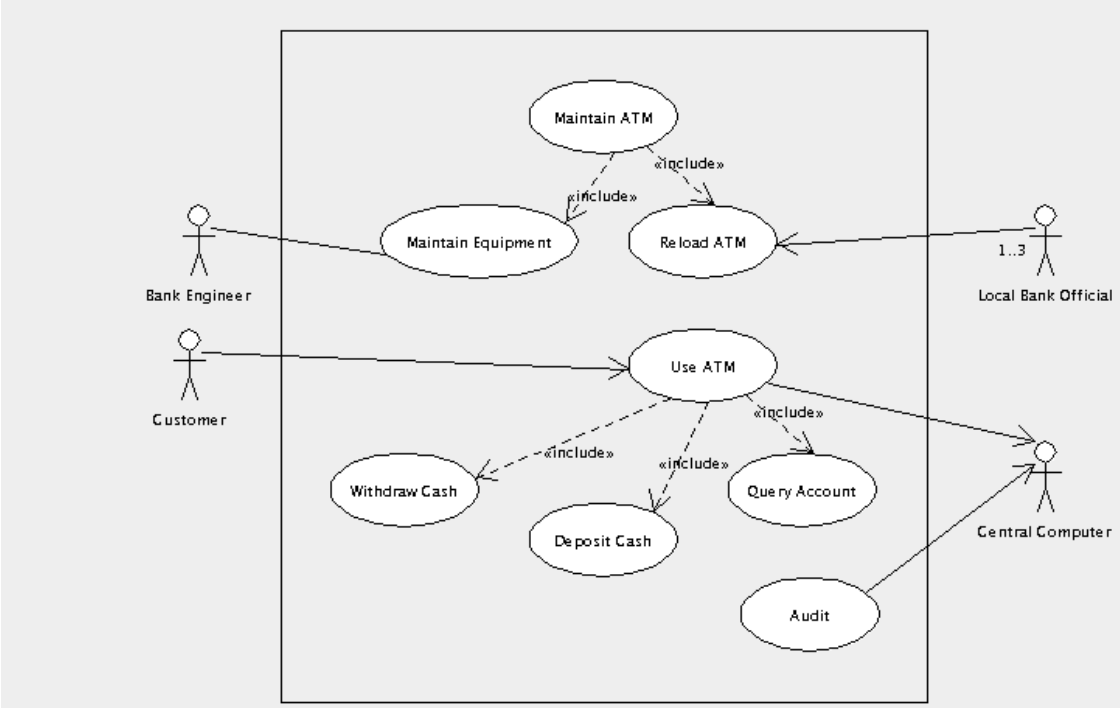
Wir könnten zum Beispiel das Verhalten des Anwendungsfalles „Geldautomat nutzen“ durch das Verhalten von drei einfacheren Anwendungsfällen, wie „Bargeld lagern“, „Bargeld ausgeben“ und „Konto abfragen“ ausdrücken. Der Hauptanwendungsfall könnte durch *einbinden (include)* des Verhaltens der benötigten Unteranwendungsfälle spezifiziert werden.

Der Anwendungsfall „Geldautomat warten“ könnte ebenfalls durch zwei Anwendungsfälle „Equipment warten “ und „Geldautomat neu starten“ definiert werden. In diesem Fall sind die zwei im Hauptanwendungsfall involvierten Akteure natürlich nur in einem oder dem anderen der beiden Unteranwendungsfälle involviert. Dies wird im nachfolgenden Diagramm dargestellt.

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

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.

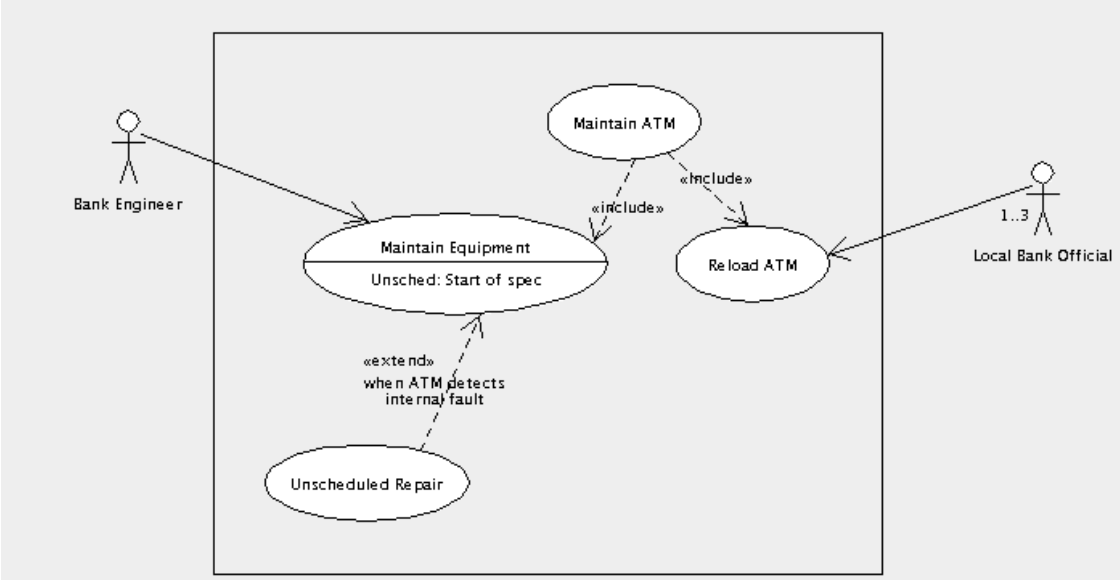
Im Geldautomatenbeispiel haben wir einen Anwendungsfall „ Equipment warten“, der die Routinewartung des Geldautomaten beinhaltet. Wir wollen aber auch den Spezialfall einer ungeplanten Reparatur abdecken, die durch den vom Geldautomaten erkannten internen Fehlers ausgelöst wurde.

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.

Abbildung 4.5, „ Anwendungsfalldiagramm für einen Geldautomaten. Zeigt eine extend-Beziehung. “ zeigt das Geldautomaten-Anwendungsfalldiagramm mit einer extend- Beziehung auf einen Anwendungsfall für ungeplante Reparaturen. Das Diagramm ist jetzt sehr komplex so dass wir es in zwei aufteilen sollten. Eines für die hauptsächlichen Dinge, das andere für den Kundennutzen und Revision.

Der Anwendungsfall „Equipment warten“ definiert den Namen „Unshed“ am Anfang seiner Beschreibung. Der erweiterte Anwendungsfall „ungeplante Reparatur“ wird herangezogen, wenn der Geldautomat einen internen Fehler entdeckt.

Abbildung 4.5. Anwendungsfalldiagramm für einen Geldautomaten. Zeigt eine extend-Beziehung.



Anwendungsfälle können auch auf andere Art und Weise miteinander verknüpft sein. Ein Anwendungsfall kann eine *Generalisierung* eines Sub-Anwendungsfalles (oder alternativ: Der Sub-Anwendungsfall ist eine *Spezialisierung* des Hauptanwendungsfalles). Das ist der extend-Beziehung sehr ähnlich, aber ohne die Bedingung des spezifischen Erweiterungspunktes, unter welcher der Haupt-Anwendungsfall erweitert wird und ohne Bedingung unter welcher der Sub-Anwendungsfall verwendet wird.

Der Generalisierung wird in einem Anwendungsfalldiagramm durch einen Pfeil mit durchgehender Linie und einer weissen Pfeilspitze vom Sub-Anwendungsfall zum Hauptanwendungsfall dargestellt. Das kann hilfreich sein, wenn ein Sub-Anwendungsfall das Verhalten des Hauptanwendungsfalles in einer Vielzahl von Positionen und in einem weiten Bereich von Sachverhalten spezialisiert. Das Fehlen jeglicher Einschränkungen macht es sehr schwierig, die Generalisierung genau zu beschreiben. Im Normalfall verwenden Sie besser eine extend-Beziehung.

4.3.3. Die Anwendungsfall-Spezifikation

Jeder Anwendungsfall muss dokumentiert sein, um das das spezifizierte Verhalten im Detail zu erläutern. ArgoUML unterstützt Sie in diesem Bereich durch das Generieren graphischer Dateien, die in diese Dokumentation eingebunden werden kann. Dieses Dokument ist unter verschieden Namen in verschiedenen Prozessen bekannt: *Anwendungsfall-Spezifikation*, *Anwendungsfall-Szenario* oder auch (verwirrend) nur *Anwendungsfall*.

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 fehlschlagen, 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 Anwendungsfallanforderungen 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.

Ein Ablauf wird durch eine Reihe von numerierten Schritten spezifiziert. Jeder Schritt muss einige Interaktionen mit einem Akteur oder mindestens eine Änderung generieren, die durch einen Akteur extern überwacht wird. Das Anforderungen erfassen sollte nicht das versteckte interne Verhalten eines Systems 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.

Zur Erinnerung: Dies ist ein Sub-Anwendungsfall der im Haupt- Anwendungsfall „Geldautomat benutzen“ enthalten ist, der voraussichtlich das Prüfen von Karten und PINs handhaben wird, bevor dieser eingebundene Anwendungsfall aufgerufen wird.

Anmerkung

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

der Kunden eine Quittung haben möchte. Der Fall, wo der Kunde keine Quittung haben will, wird ein alternativer Ablauf sein.

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.3. Die Kommunikation mit dem Zentralcomputer ist unterbrochen.

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.
1.
A.1. Der Standardablauf geht von Schritt 2 in Schritt 4, Schritt 5 wird nicht benötigt.
2.

Die Konvention ist, dass die verschiedenen alternativen Abläufe numeriert werden, wie A.1, A.2, A.3, usw. Die Schritte innerhalb eines alternativen Ablaufes werden darauf aufbauend numeriert. So dass die Schritte der ersten Alternative wie folgt lauten würden: A.1.1, A.1.2, A.1.3, usw.

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.

Anmerkung

Einige Analytiker plazieren nicht-funktionale Anforderungen in einem Abschnitt am Ende einer jeden Anwendungsfallspezifikation, der die nicht-funktionalen, anwendungsfallbezogenen Anforderungen enthält.

Dies kann einige Probleme verursachen. Der erste Punkt ist, dass nicht-funktionale Anforderungen (zum Beispiel über die Performance) in vielen Anwendungsfällen erscheinen muß und es ist eine schlechte Praxis, Informationen zu replizieren. Zweitens gibt es einige invariable systemweite nicht-funktionale Anforderungen, die ein systemweites Dokument erfordern. Daher meine Präferenz für eine einzige ergänzende Anforderungsdokumentation.

Es sollte ein Abschnitt für jeden Hauptbereich der nicht-funktionalen Anforderungen geben. Die Checkliste von Ian Sommerville in seinem Buch *Software Engineering* (Third Edition, Addison-Wesley, 1989) ist eine hilfreiche Anleitung.

- *Geschwindigkeit*. Prozessorleistung, Anwender-/Ereignis-Antwortzeiten, Bildauffrischungszeiten.
- *Grösse*. Hauptspeicher (und mögliche Zwischenspeicher), Plattenkapazität.
- *Leichte Anwendbarkeit*. Ausbildungszeit, Stil und Details des Hilfesystems.
- *Zuverlässigkeit*. Durchschnittliche Fehlerauftrittszeit, 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 Bearbeitungsfenster (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 Bearbeitenfensters () 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 1herunterdrücken, 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.

Wenn sie einen Akteur erstellt haben, werden Sie sehen, dass er im Exlorer (der obere linke Quadrant des ArgoUML-Fensters) erscheint. Dieser zeigt alle im UML-Design erstellten Modellelemente. Eine Kombinationsfeldliste am oberen Ende des Explorers beeinflusst die Reihenfolge der Modellelemente im Explorer. Die nützlichsten sind `Nach Paketen` (Standard) und `Nach Diagrammen`. Das letztere gruppiert die Modellelemente nach Diagrammen.

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`.

2. Durch klicken mit der Taste 2 auf dem Anwendungsfall im Bearbeitenfenster erscheint ein kontext-sensitives Popup-Menü und aus diesem wählen Sie `Darstellung/ Erweiterungspunkte anzeigen` aus.

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.

1. Markieren Sie mit der Taste 1 den Anwendungsfall im Bearbeitenfenster. Dann klicken Sie in der Werkzeugleiste auf das Symbol `Neue Erweiterung` ( und ein neuer Erweiterungspunkt mit Standardnamen und -ort wird im Anschluss an die exisiterenden Erweiterungspunkte hinzugefügt.

Anmerkung

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

2. Markieren Sie den Anwendungsfall im Bearbeitenfenster mit dem Taste 1-Klick und dann wählen Sie das Register Eigenschaften im Detailfenster aus. Ein Taste 2-Klick über dem Feld `Erweiterungspunkt:` wird ein kontext-sensitives Popup-Menü öffnen. Markieren Sie `Hinzufügen` aus, um einen neuen Erweiterungspunkt hinzuzufügen.

   Wenn bereits ein Erweiterungspunkt existiert, werden Sie in diesem Feld im Register Eigenschaften angezeigt. Der neue Erweiterungspunkt wird unmittelbar vor dem Eintrag über dem das Popup-Menü aufgerufen wurde eingefügt. Diese Reihenfolge kann später durch die `Nach oben` und `Nach unten`-Einträge im Popup-Menü verändert werden.

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

1. Wenn für den Anwendungsfall der Erweiterungspunkt-Bereich im Diagramm angezeigt wird, markieren Sie den Anwendungsfall mit dem Taste 1-Klick und dann markieren Sie den Erweiterungspunkt mit einem weiteren Taste 1-Klick. Das Register Eigenschaften kann dann im Detailfenster ausgewählt 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.

Wenn der Erweiterungspunktbereich angezeigt wird, ist der Doppelklick auf den Erweiterungspunkt eine Abkürzung und erlaubt es Ihnen den Text direkt einzugeben. Dies wird analysiert, um den Namen und den Ort für den Erweiterungspunkt zu setzen.

Erweiterungspunkte dürfen gelöscht oder deren Reihenfolge mit Hilfe des Taste 2-Popup-Menüs über dem Feld `Erweiterungspunkte` im Register Eigenschaften des Anwendungsfalles geändert 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

Linie segmentieren, indem Sie die Taste 1 auf der Linie gedrückt halten und vor dem loslassen bewegen. Der Linie wird ein Schnittpunkt hinzugefügt, den Sie mit der Taste 1 bewegen können. Durch Aufnehmen und durch verschieben zu einem Ende der Linie kann der Schnittpunkt entfernt werden.

Es können mehrere Assoziationen gleichzeitig hinzugefügt werden, indem Sie auf das Assoziationssymbol einen Taste 1-Doppelklick ausführen. Jede darauf folgende Taste 1-drücken/bewegen/loslassen- Sequenz wird einen Akteur mit einem Anwendungsfall verbinden. Das Anklicken des Auswahlsymbols () mit der Taste 1 stoppt das Hinzufügen von Assoziationen.

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

Das Ziehen eines Griffes von einem Anwendungsfall in den leeren Raum wird einen neuen Akteur am anderen Ende erzeugen. Umgekehrt wird das Ziehen eines Griffes von einem Akteur in den leeren Raum einen neuen Anwendungsfall erzeugen.

Es ist möglich, der Assoziation einen Namen zu geben, der die Beziehung des Akteurs zum Anwendungsfall beschreibt. Obgleich das gewöhnlich nicht notwendig ist. Dies wird über das Eigenschaftsregister der Assoziation getan. Solch ein Name erscheint entlang der Assoziation in der Nähe der Mitte.

4.4.3.1. Navigation einstellen

Es gibt zwei Wege, die Navigation einer Assoziation einzustellen.

1. Das Anklicken der Assoziation mit der Taste 2 bringt ein kontextsensitives Popup-Menü nach oben. Das Untermenü `Navigierbarkeit` enthält Optionen für die bidirektionale Navigation (der Standard, ohne Pfeile) und für die Navigierbarkeit Akteur->Anwendungsfall und Anwendungsfall->Akteur.

2. Verwenden Sie die Taste 1, um die Assoziation zu markieren und markieren Sie sein Eigenschaftsregister im Detailfenster. Dieses enthält ein Feld mit der Bezeichnung `Verbindungen:` mit Einträgen für jedes mit einem Akteur oder Anwendungsfall verbundenen Ende und seiner Kardinalität. Markieren Sie das Ende welches das Ende des Pfeiles darstellt mit einem Taste 1- Doppelklick. Dies bringt das Eigenschaftsregister für das Assoziationsende nach oben. Verwenden Sie den Taste 1- Klick, um die Markierung des Kästchens `navigierbar` zu entfernen.

Anmerkung

Dies erscheint anti-intuitiv, aber tatsächlich sind Assoziationen standardmäßig in beide Richtungen navigierbar (wenn keine Pfeile angezeigt werden). Dieser Prozess ist eher das *Ausschalten* der Navigation an einem Ende, als das Einschalten am anderen Ende.

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 Zeitmanagmentsystem 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 Kardinaltitätsoptionen aus, die ausgewählt werden können.

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

4.4.4. Hierarchische Anwendungsfälle

Der Originalentwurf der UML erlaubt es, dass Anwendungsfälle in Paketen gruppiert, sowie Beziehungen zwischen ihnen spezifiziert werden können. In ArgoUML werden nur die Beziehungsmechanismen unterstützt. Alle drei Beziehungen, die auf Anwendungsfälle angewendet werden können werden unterstützt. Dieses sind *include*, *extend* und *generalization*.

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 mit 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 *erweiternde* (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 öffenen Sie das Eigenschaftsregister im Detailfenster (Taste 1-Klick auf das Register). Der Text der Bedingung kann in das `Bedingungs`-Feld eingegeben 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

Die Prozedur zum Hinzufügen einer Generalisierung ist die gleiche, wie für das Hinzufügen einer Extend-Beziehung. Allerdings verwenden Sie das Generalisierungs-Symbol aus der Editier-Werkzeugleiste ().

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).

Es ist auch möglich, die Generalisierung mit Hilfe der kleinen Griffe, die oben und unten am Anwendungsfall erscheinen, wenn er markiert ist, hinzuzufügen. Das Ziehen des oberen Griffes auf einen anderen Anwendungsfall erzeugt eine Generalisierung. Der Original-Anwendungsfall ist das spezialisierte Ende und der Anwendungsfall auf den der Griff gezogen wurde wird das generalisierte Ende. Das ziehen in den leeren Raum wird einen neuen Anwendungsfall mit einem generalisierten Ende erzeugen.

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

Die Generalisierung ist auch zwischen Akteuren erlaubt, obwohl dieser Gebrauch ausserhalb des Bezugsbereiches dieses Tutorials liegt. Im Gegensatz zu den Anwendungsfällen gibt es keine Generalisierungs-Griffe bei Akteuren, so dass Generalisierungen mit Hilfe der Symbole in der Werkzeugleiste erzeugt werden müssen.

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.

Stereotypen werden zur Kennzeichnung von UML-Modellelementen wie Anwendungsfälle benutzt, um darzustellen, dass diese zu einer bestimmten Kategorie gehören. Diese Kennzeichen werden in guillemots (« ») über dem Namen des Modellelementes im Diagramm dargestellt. Der UML-Standard definiert eine Anzahl von Standard-Stereotypen und der Anwender darf weitere Stereotypen selbst definieren.

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

In ArgoUML fehlen einige der Standard-UML-Stereotypen. Zusätzlich stellen nicht alle Modellelemente die Stereotypen im Diagramm dar. Aktuell sind sie in Anwendungsfällen und Akteuren enthalten.

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 ().

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*).

Das Rechteck kann verwendet werden, um die Grenzen des Systems darzustellen. Nutzen Sie das Taste 2-kontextsensitive Popup- Menü `Reihenfolge`, um diese hinter alle anderen Elemente zu plazieren.

Dann ändern Sie die Füllfarbe von Standard-Weiss in "Keine Füllfarbe" im Darstellungsfenster.

Anmerkung

Das Editierfenster in ArgoUML hat ein Gitter, in das Objekte beim Zeichnen einrasten. Die Größe dieses Gitters und seine Auswirkungen können durch das `Ansicht`-Menü (verwenden Sie `Raster einstellen` und `Einrasten einstellen`) verändert werden. Dies ist im Referenzhandbuch vollständig beschrieben (siehe Kapitel 10, *Die Menüzeile*).

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 die beides ist: Wartbar und Robust.

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, das 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
- *Computerbetrieb*. Laurel, Stanley

- *Geschäftsführer*. Hun, Atilla von
- *Marketing*. Harry, Oil Can

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 Planes ausgeführt, oder wenn der Zentralcomputer feststellt, dass der Geldvorrat zu gering oder die Einlagenkassette fast voll ist.

Es wird ein Nachweis von allen Aktivitäten erzeugt und periodisch an den Zentralcomputer der Banken gesandt. Es soll dem Wartungsingenieur möglich sein, eine Kopie des Nachweislog auf einer Diskette für den Transport zum Zentralcomputer zu speichern.

Es wird Wähl- und Standleitungssupport benötigt. Der Geldautomat soll auch weiterarbeiten können, wenn die Kommunikation mit dem Zentralcomputer nicht verfügbar ist.

4.5.1.6. Randbedingungen

Das Projekt muss innerhalb von 9 Monaten abgeschlossen sein. Es darf nicht mehr als 1.750.000 USD ausschliesslich der Produktionskosten kosten. Komponenten können ausserhalb produziert werden aber die Basisarchitektur als auch die Infrastruktur wird im Haus entworfen. Eine enge Zusammenarbeit muss zwischen der Softwareentwicklung und dem Design, der Entwicklung und der Produktion der Hardware sichergestellt werden. Weder die Hardware noch die Software darf als unabhänige Variable betrachtet werden. Sie müssen immer gleichzeitig betrachtet werden.

4.5.1.7. Anhang

Im Folgenden finden Sie Akteure, die diese Vision direkt unterstützen. Zusätzliche Akteure können später identifiziert werden, sofern Sie zur Unterstützung dieser Vision oder der Technologie erforderlich sind. Sie sollten nicht zu dieser Liste hinzugefügt werden, es sei denn, sie sind zur direkten Unterstützung dieser Vision, wie in diesem Dokument beschrieben, erforderlich.

- 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

Für die Geldautomaten-Fallstudie werden wir die Beispiele in Abschnitt 4.3, „ Ergebnis des Anforderungs-Erfassungs-Prozesses “, Abbildung 4.4, „ Anwendungsfalldiagramm für einen Geldautomaten mit include-Beziehungen. “ und Abbildung 4.5, „ Anwendungsfalldiagramm für einen Geldautomaten. Zeigt eine extend-Beziehung. “, und fortschreiben, um zusätzliche Akteure und Anwendungsfälle zu identifizieren, die unser Geldautomatenmodell enthält. Abbildung 4.4, „ Anwendungsfalldiagramm für einen Geldautomaten mit include-Beziehungen. “ und Abbildung 4.5, „ Anwendungsfalldiagramm für einen Geldautomaten. Zeigt eine extend-Beziehung. “ erläutert die grundsätzlichen Konzepte und Komponenten des Anwendungsfalldiagrammes wie Anwendungsfälle, Akteure, Kardinalität und Include-/Extend-Beziehungen. Sie zeigen die Beziehungen zwischen den Akteuren und den Anwendungsfällen und demonstieren wie diese Akteure und Anwendungsfälle miteinander interagieren.

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 Phänomenalist 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 initiiert werden können.

CRC-Karten und die entsprechende Methode sind detailliert in Anhang G, *Die CRC-Karten Methode* beschrieben. Sie werden in der Designphase erneut verwendet und weiter diskutiert in Kapitel 6, *Design* .

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 das Ausrollen des Systems verantwortlich sind.

Wenn man das erste Mal in die Analysephase eintritt, tritt ein spezieller Fall der CRC-Sitzung auf, weil es keine Klassen oder Szenarien gibt, die man in der CRC-Sitzung für die Definition auswählen könnte. Zu diesem Zeitpunkt tritt ein spezieller Sitzungstyp, Brainstorming genannt, auf. Während dieser Phase identifizieren Sie den grundlegenden Satz von Klassen des Problembereiches, indem Sie die Problembeschreibung oder das Anforderungsdokument oder was immer Sie auch über das gewünschte Ergebnis wissen als Startpunkt verwenden. Die Hauptwörter, die Sie finden, sind gute Ansätze für einen ersten Satz von Klassen in dem System. Während der Brainstorming-Sitzung sollten die Ideen wenig bis gar nicht diskutiert werden. Schreiben Sie diese auf und filtern Sie die Ergebnisse nach dem Brainstorming. Zu diesem Zeitpunkt ist die Unterscheidung zwischen Klasse und Objekt unscharf.

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)

Noch zu beschreiben...

5.1.3. System-Sequenzdiagramm (Noch zu beschreiben)

Noch zu beschreiben...

5.1.4. System-Zustandsdiagramm (Noch zu beschreiben)

Noch zu beschreiben...

5.1.5. Anwendungsfalldiagramm realisieren (Noch zu beschreiben)

Noch zu beschreiben...

5.1.6. Dokumente (Noch zu beschreiben)

Anwendungsfall- und Ergänzende Anforderungs-Spezifikationen sind in die Sprache der Lösung umzuwandeln. Noch zu beschreiben...

5.2. Klassendiagramme (Noch zu beschreiben)

Noch zu beschreiben...

5.2.1. Das Klassendiagramm (Noch zu beschreiben)

Noch zu beschreiben...

5.2.2. Erweiterte Klassendiagramme (Noch zu beschreiben)

Noch zu beschreiben...

5.2.2.1. Assoziationsklassen (Noch zu beschreiben)

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)

Noch zu beschreiben...

5.3.2.1. Aggregation (Noch zu beschreiben)

Noch zu beschreiben...

5.3.3. Klassenattribute und Operationen (Noch zu beschreiben)

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 bischen 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)

Noch zu beschreiben...

5.3.3.3. Klassenoperationen (Noch zu beschreiben)

Noch zu beschreiben...

5.3.4. Erweiterte Klasseneigenschaften (Noch zu beschreiben)

5.3.4.1. Assoziationsklassen (Noch zu beschreiben)

Noch zu beschreiben...

5.3.4.2. Stereotypen (Noch zu beschreiben)

Noch zu beschreiben...

5.4. Sequenzdiagramme (Noch zu beschreiben)

Noch zu beschreiben...

5.4.1. Das Sequenzdiagramm (Noch zu beschreiben)

Noch zu beschreiben...

5.4.2. Aktionen identifizieren (Noch zu beschreiben)

Noch zu beschreiben...

5.4.3. Erweiterte Sequenzdiagramme (Noch zu beschreiben)

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)

Noch zu beschreiben...

5.5.3. Erweiterte Sequenzdiagramme (Noch zu beschreiben)

Noch zu beschreiben...

5.6. Zustandsdiagramme (Noch zu beschreiben)

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 Zustands?

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)

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 Problembeschreibung 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 erfüllt werden müssen (auf Geschäftsebene) und alles weglassen müssen, was nach "wie müssen wir es tun" aussieht. Als generelle Regel diese 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)

Noch zu beschreiben...

5.10.2.1. Klassen identifizieren (Noch zu beschreiben)

Noch zu beschreiben...

5.10.2.2. Assoziationen identifizieren (Noch zu beschreiben)

Noch zu beschreiben...

5.10.3. System-Sequenzdiagramme (Noch zu beschreiben)

Noch zu beschreiben...

5.10.3.1. Aktionen identifizieren (Noch zu beschreiben)

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.

Die verwischten Grenzen zwischen der Analyse und dem Design zeigt sich hier auch durch die gemeinsame Verwendung derselben UML-Werkzeuge. In diesem Kapitel werden wir sehr häufig die UML-Technologie verwenden, die wir bereits kennengelernt haben. Der grosse Schritt ist das Umwandeln in konkrete Ausdrücke. Wir bewegen uns von den abstrakten Konzepten der Analyse hin zu deren konkreten Realisierung.

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)

Der Designprozess erweitert den Modellierungsaufwand jenseits der Geschäftsanforderungen in Richtung des Lösungsraumes. Während dieser Arbeit entscheiden Sie, ob Sie Java, C++, J3EE, CORBA, SOAP, Wählleitungen, Internetverbindungen, Standleitungen, XML, usw. verwenden werden. Viele dieser Entscheidungen werden das PSM-Modell direkt beeinflussen, andere widerum werden sich nur in den erzeugten Dokumenten wiederspiegeln.

...

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.

Der Fokus der Gruppe bewegt sich von der Frage was zu tun ist hin zu der Frage wie es zu tun ist. Der Klassen aus dem Lösungsbereich werden denen der Analysephase hinzugefügt. Es wird darüber nachgedacht, welche Klassen werden benötigt, damit das System arbeitet. Benötigen Sie eine Listenklasse, die Objekte beinhaltet? Benötigen Sie Klassen, die Ausnahmen behandeln? Benötigen Sie Wrapperklassen für andere Subsysteme? In diesem Abschnitt wird nach neuen Klassen gesucht, nach Klassen, die die Implementierung des Systems unterstützen.

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)

Noch zu beschreiben...

6.1.3. Klassendiagramme realisieren (Noch zu beschreiben)

Noch zu beschreiben...

6.1.4. Sequenz- und Kollaborationsdiagramme (Noch zu beschreiben)

Noch zu beschreiben...

6.1.5. Zustands- und Aktivitätsdiagramme (Noch zu beschreiben)

Noch zu beschreiben...

6.1.6. Verteilungsdiagramme (Noch zu beschreiben)

Noch zu beschreiben...

6.1.7. Dokumente (Noch zu beschreiben)

System Architektur. Noch zu beschreiben...

6.2. Paketdiagramme (Noch zu beschreiben)

Noch zu beschreiben...

6.2.1. Das Paketdiagramm (Noch zu beschreiben)

Noch zu beschreiben...

6.2.2. Erweiterte Paketdiagramme (Noch zu beschreiben)

Noch zu beschreiben...

6.2.2.1. Subpakete (Noch zu beschreiben)

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)

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

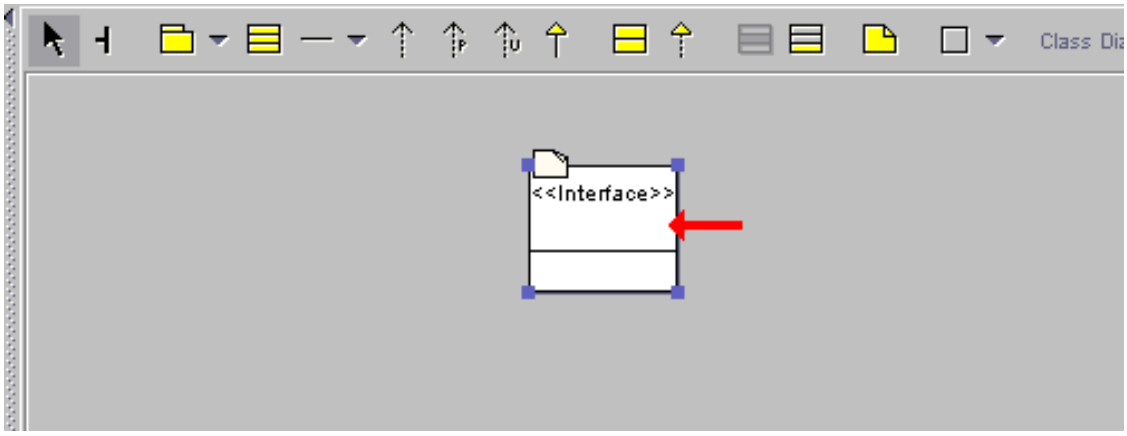
6.5.3.1.1. Schnittstellen, die Schnittstellen erweitern

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.

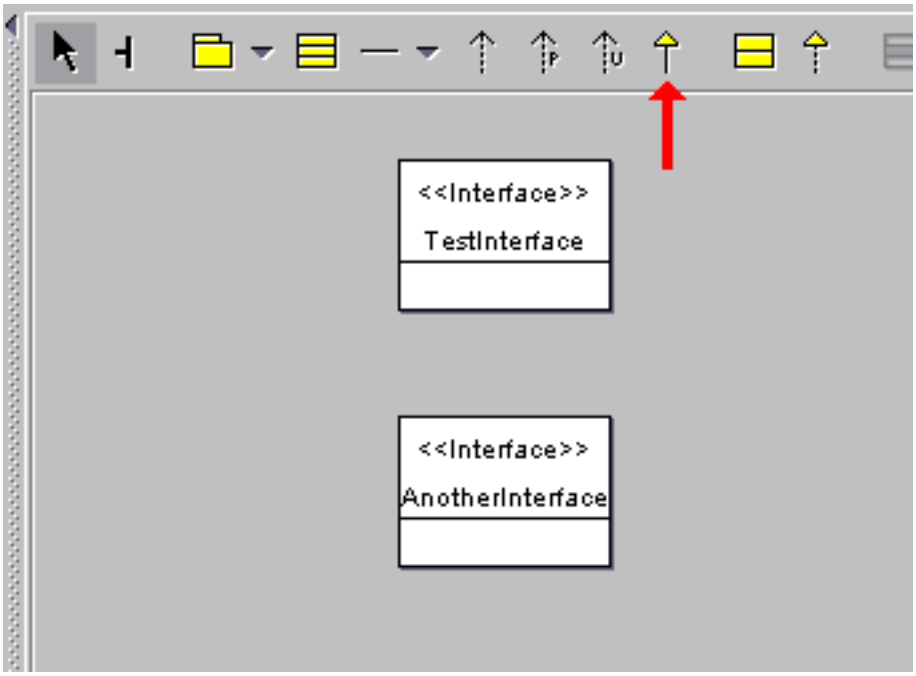
Abbildung 6.2. Modellelement Schnittstelle im Klassendiagramm



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.)

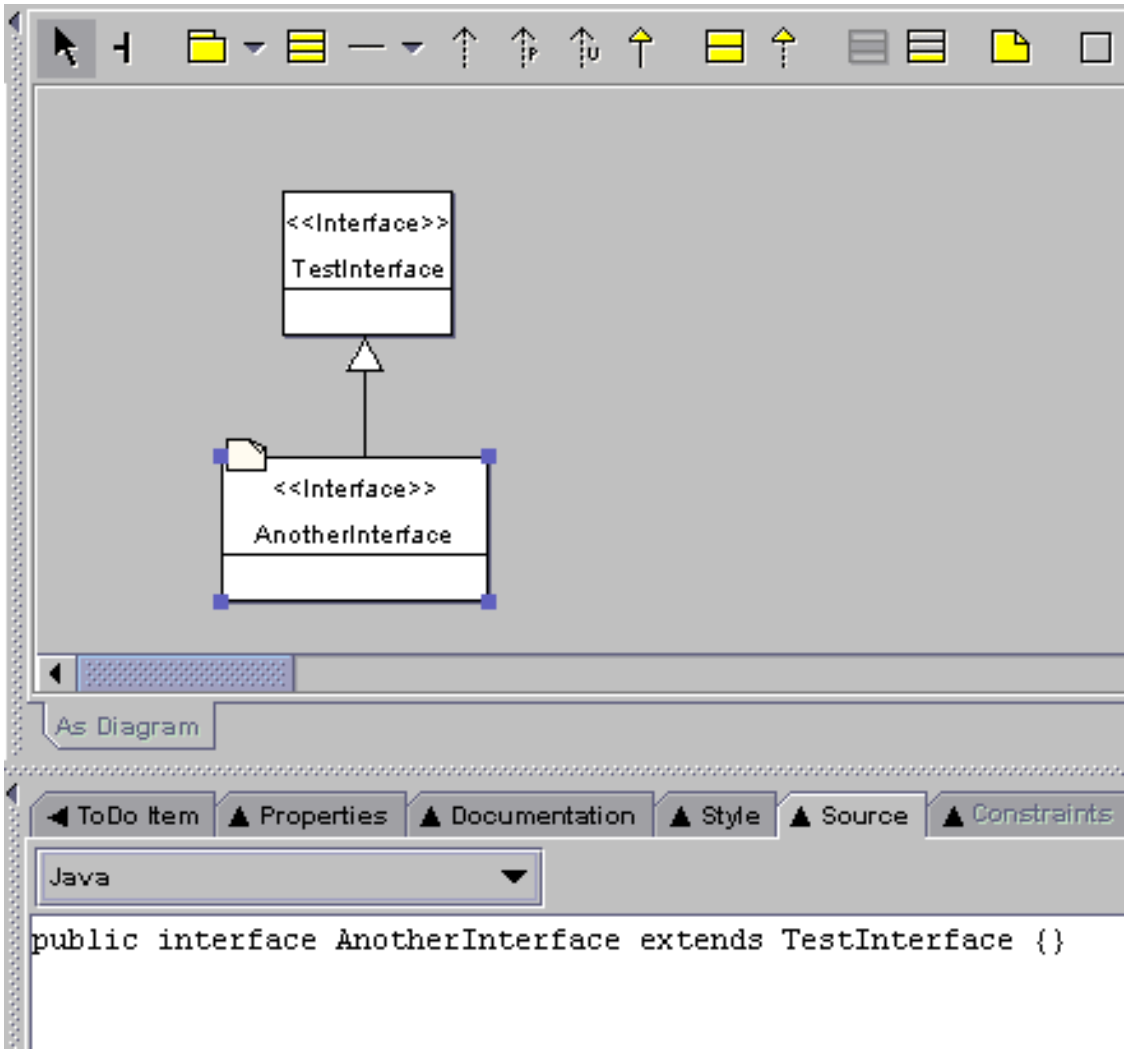
Fügen Sie eine weitere Schnittstelle hinzu, indem Sie die letzten beiden Schritte wiederholen. Dann klicken Sie auf das Generalisierungssymbol in der Werkzeugleiste wie in Bild Abbildung 6.3, „ Generalisierung über die Symbolleiste des Klassendiagrammes “ dargestellt.

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...

6.14. System-Architektur (Noch zu

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

Wir haben jetzt unser Design vollständig spezifiziert. Mit dem richtigen Simulator können wir das Design aktuell ausführen und sehen, wie es arbeitet. (ArgoUML enthält keine solche Funktionalität, aber diese Funktionalität wird in anderen Tools angeboten).

ArgoUML erlaubt es Ihnen, Code aus dem Design in vielen verschiedenen Programmiersprachen auszuführen. Wir haben bereits sehr wahrscheinlich im Design eine Programmiersprache ausgewählt, weil einige der Designanforderungen fordern, eine spezielle Sprache auszuwählen.

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.

Der Prozess des vor- und zurückgehens zwischen den Änderungen im Design gefolgt von einer Codegenerierung und anschliessender Änderungen im Code gefolgt von einem Reverse Engineering bei jeder Änderung, die bestmögliche Perspektive, wird Round-trip Engineering genannt.

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 Membervariablen.
- 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.

  In C++ werden dies referenzierende nicht-konstante Parameter. Bei Javaklassen ist dies der Standard. Einfache Typen (int, boolean) müssen in Java in ein Objekt einer korrespondierenden Klasse (Integer, Boolean) konvertiert werden.
- 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

Der grösste Teil der Generierung wird automatisch durch die ausgewählten Sprachmodule erledigt. Dateien, die benötigt werden, um den aktuellen Code aufzunehmen, werden in Verzeichnis- hierarchien generiert.

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.

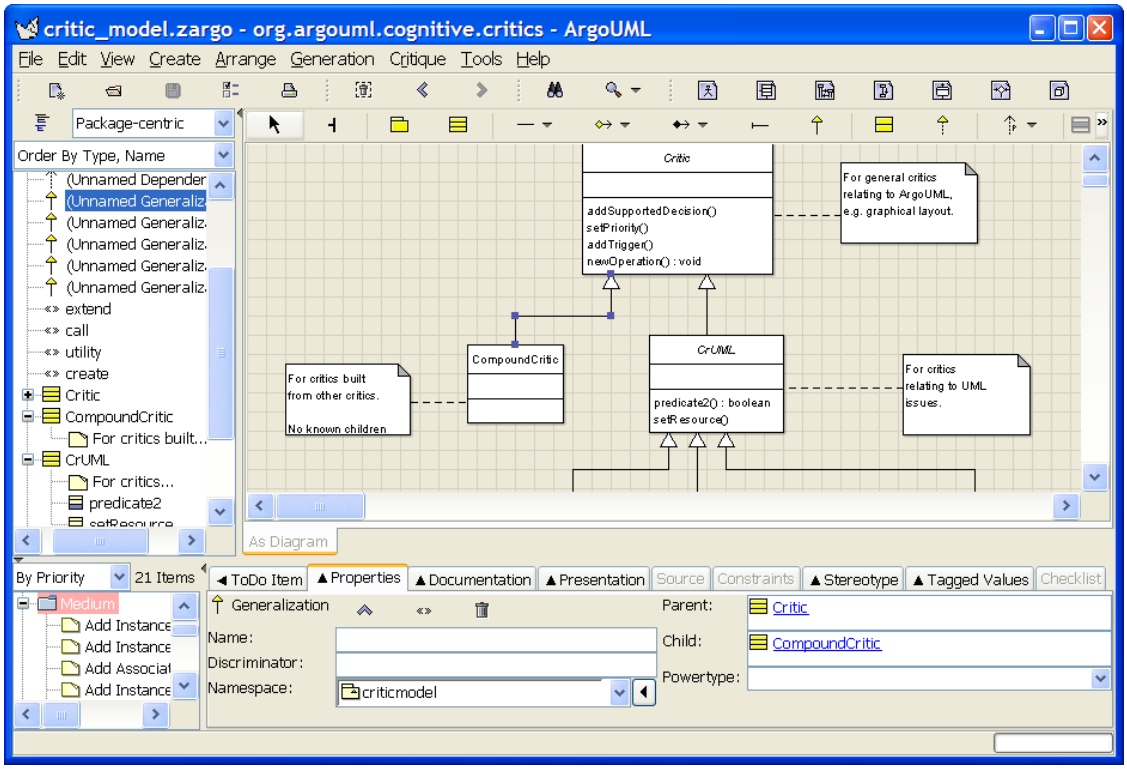
8.1. Überblick über das Fenster

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 duch einen Bindestrich.

- Den aktuellen Dateinamen. Ist noch kein Dateiname für das Projekt vergeben, dann wird in der Titelzeile „Unbenannt“ angezeigt.
- Der Name des aktuell aktiven Diagrammes.
- Der Name „ArgoUML“.
- Ein Stern (*). Dieses Symbol ist nur vorhanden, wenn die aktuelle Projektdatei „ungesichert“ ist. Z.B. sie wurde verändert, aber noch nicht gespeichert. Mit anderen Worten, fehlt der Stern, dann wurde die aktuelle Datei nicht verändert.

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

ArgoUML unterstellt eine Maus mit zwei Tasten. Wir beziehen uns auf die Tasten mit den Bezeichnungen „Taste 1" und „Taste 2". Taste 1 ist die linke Taste auf einer Rechtshänder-Maus und wird manchmal als *Auswahl*-Taste bezeichnet. Taste 2 ist die rechte Taste auf einer Rechtshänder-Maus und wird manchmal als *Einstell*--Taste bezeichnet.

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

Das Klicken auf ein Objekt der Anwenderschnittstelle oder auf ein Modellelement eines Diagrammes kann unterschiedliche Dinge auslösen. Der größte Teil des Verhaltens ist für den Anwender erfahrungsgemäß vollständig intuitiv. Hauptsächlich wegen des hohen Grades an Standardisierung, auch plattformübergreifend (Macintosh, PC, UNIX,...). ArgoUML befolgt die *Java Look and Feel Design-Richtlinien* von Sun. Siehe http://java.sun.com/products/jlf/. Daraus folgt, dass das Verhalten von allgemeinen Komponenten der Anwenderschnittstelle in diesem Dokument generell nicht diskutiert wird.

Auf der anderen Seite können Mausaktionen in einem Diagramm für den Anwender nicht so intuitiv erscheinen, da sie spezifisch für ArgoUML sind. Deshalb werden sie hier erläutert. In Kürze: klicken markiert oder aktiviert das Objekt nahe des Mauszeigers und verschiebt den Fokus (z.B. Navigation).

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

angewendet werden sollen. Mehrere Modellelemente können durch eine Umschalt- und/oder Strg-Kombination mit der Taste 1 ausgewählt werden, siehe Abschnitt 8.2.5, „ Umschalt- und Strg- und die Taste 1 “. Die Markierung wird immer durch einen eingefärbten Hintergrund klar angezeigt.

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.
- Menüfunktionen und Tastenkürzel. Viele Menüoperationen verändern die Markierung als Seiteneffekt. Zum Beispiel beim erstellen eines neuen Diagrammes. Viele Tastenkürzel für Menüoperationen ändern die Markierung, z. B. Strg-A, die für die Funktion `Markiere alles` steht.

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 (hart 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

Anwender von Microsoft Windows könnten mit dem Gebrauch des Umschalt-Strg-Klick vertraut sein (z.B. Umschalt- und Strg-Taste während des klickens gedrückt halten), um der existierenden Markierung Sublisten hinzuzufügen. ArgoUML unterstützt dies nicht. Umschalt-Strg-Klicks wirken wie Strg-Klicks.

8.2.5.2. Generelles Verhalten beim Editieren von Text

In verschiedenen Fällen kann Text in ArgoUML direkt editiert werden (zum Beispiel beim Benennen eines Modellelementes im Eigenschaftsregister oder bei der Eingabe eines UML-Hinweises/ -Kommentares. Hier wird Umschalt-Taste 1 verwendet, um ausgehend von einem vorher markierten Punkt einen Textbereich zu markieren. Nachfolgende Operationen (Text einfügen und löschen) werden den markierten Text ersetzen.

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 verschieben. 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

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.

Darüber hinaus gibt es zwei kleine nach links zeigende Pfeile in den vertikalen Trennbalken, einer im oberen Bereich des Trennbalkens, zwischen dem Explorer und dem Editierfenster und einen im oberen Bereich des vertikalen Trennbalkens zwischen dem „Zu bearbeiten“- und dem Detailfenster. Ein Taste 1-Klick auf den ersten von ihnen erweitert das Editierfenster bis zur vollen Breite des Fensters, der Taste 1-Klick auf den zweiten erweitert das Detailfenster auf die volle Breite des Fensters.

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 Pfeile 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

Die Statuszeile befindet sich am unteren Rand des ArgoUML-Fensters und wird dazu verwendet, kurzes Hinweisnachrichten anzuzeigen. Normalerweise sind diese Nachrichten selbsterklärend. Sie wird z. B. für die Anzeige von Parsing-Fehlermeldungen verwendet, wenn ein in ein Diagramm eingegebener Text nicht interpretiert werden kann.

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 unterAbschnitt 10.3.3, „  Projekt speichern “.
-  `Projekt-Eigenschaften` Siehe vollständige Beschreibung unter Abschnitt 10.3.14, „  Projekteinstellungen... “.
-  `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... “.

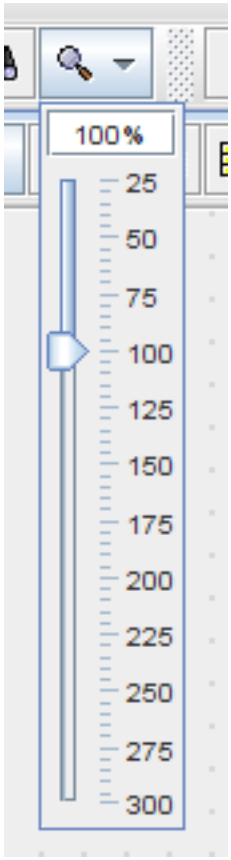
9.3. Ansicht-Operationen

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

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

 `Zoom` 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

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.

Es ist auch möglich, das Menü von der Tastatur aus zu bedienen. Jede Ebene eines Menüs wird durch einen Buchstaben (im Menü unterstrichen dargestellt oder durch Eingabe des Namens im Moment des drückens der ALT-Taste) identifiziert. Diese Buchstabensequenz, während die ALT-Taste gedrückt wird, wählt den Eintrag aus.

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 *Bearbeiten* ist hauptsächlich dafür gedacht, das Modell zu editieren oder den Inhalt eines Diagrammes zu verändern. Es enthält auch Funktionen, die das Editieren erst ermöglichen, wie z.B. das Markieren. Dieses Menü ist nicht für Diagramm-Layout-Funktionen gedacht. Die meisten Funktionen hier, tun irgendetwas mit dem markierten Modellelement und Diagramm. Die Elemente „Konfiguriere Perspektiven...“ und „Einstellungen...“ sind ein bißchen unterschiedlich, da sie einstellen, wie ArgoUML arbeitet - aber sie gehören nicht in das Dateimenü, da ihre Einstellungen nicht im Projekt gespeichert 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. Dies Funktionen sind kontextabhänig, 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 *Hinweise* ist speziell für die Einstellungen der Hinweise, 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.

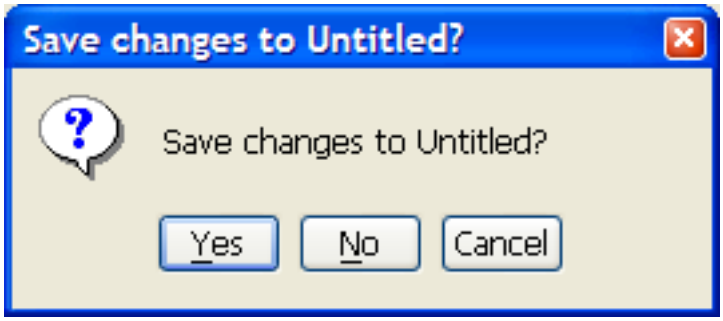
Dies initialisiert ein neues Projekt in ArgoUML. Das Projekt wird ohne Name erstellt. Es enthält ein (oberste Ebene) `unbenanntesModell` bezeichnetes `Modell` und zwei leere Diagramme: Ein Klassendiagramm und ein Anwendungsfalldiagramm.

Achtung

`unbenanntesModell` ist kein Modellname, der der Konvention entspricht (die meisten Prozesse erwarten, dass Modelle aus Kleinbuchstaben gebildet werden sollten). ArgoUML erlaubt es Ihnen Groß- und Kleinbuchstaben zu verwenden, aber ein Hinweis wird Sie warnen, dass dies nicht der Konvention entspricht. Siehe Abschnitt 16.2, „ Das Modell “.

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`.

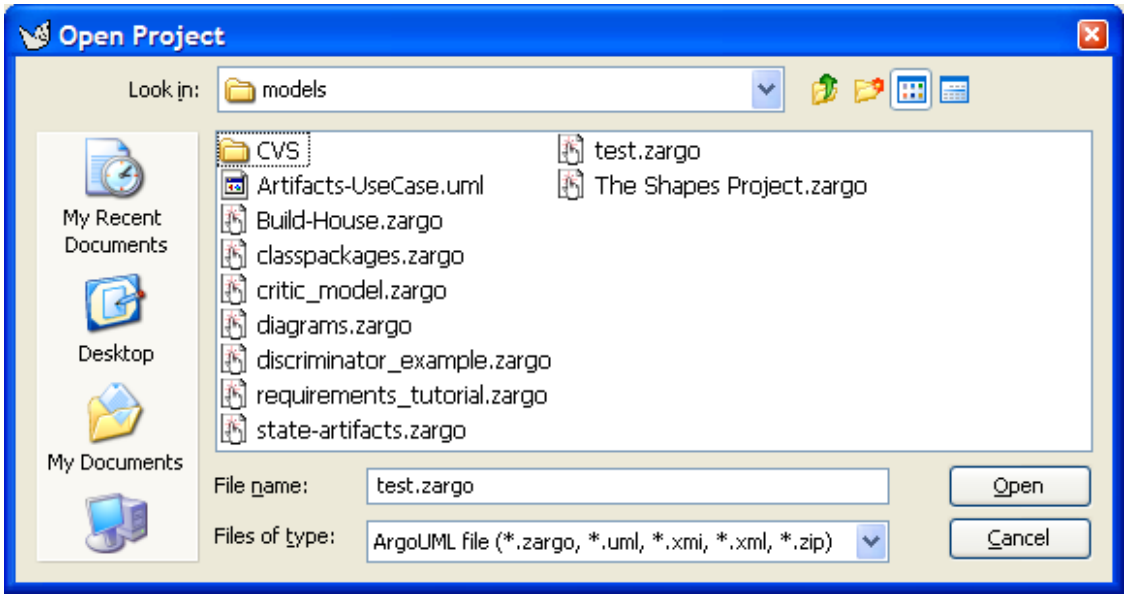


10.3.2. Projekt öffnen...

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.

Darunter befindet sich eine mit `Dateityp:` bezeichnete DropDown-Auswahl, um einen Filter für die Dateien auszuwählen, die in der Verzeichnisliste angezeigt werden sollen. Nur die Dateien werden angezeigt, die mit diesem Filter übereinstimmen. Die verfügbaren Filter sind nachstehend aufgeführt. Der Standardfilter ist der erste, der alle verfügbaren Formate kombiniert.

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

Unter bestimmten Umständen gibt es nichts zu speichern. Dann ist diese Menüoption deaktiviert. Z.B. wenn der Anwender das geladene Projekt nicht verändert hat. Die Präsenz eines „*“ in der Titelzeile von ArgoUML zeigt an, dass das aktuelle Projekt „ungesichert“ ist (geändert wurde) und gespeichert werden kann.

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.

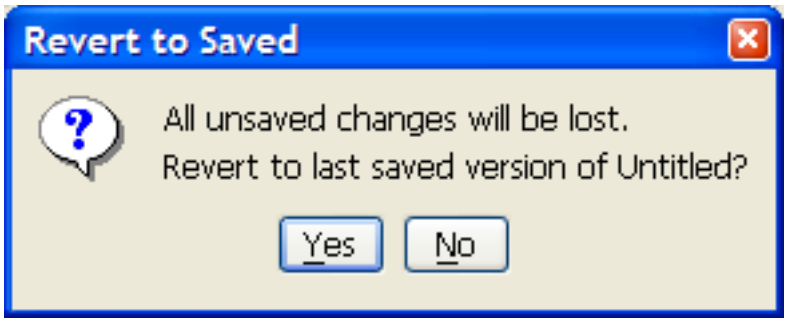
10.3.5. Projekt speichern rückgängig machen

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.

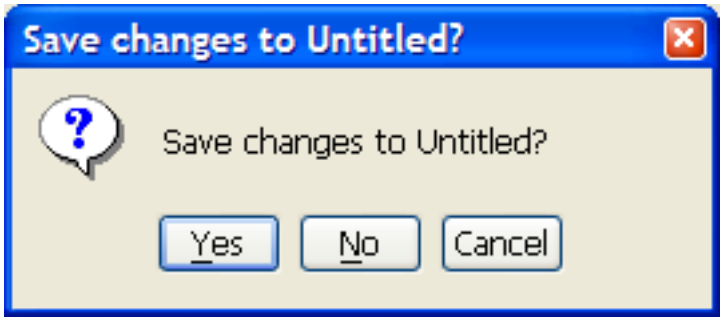


10.3.6. XMI importieren...

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 XMI-V1.0-, V1.1- oder V1.2-Standard exportiert wurde. Die Erweiterung einer solchen Datei sollte `.xmi` lauten.

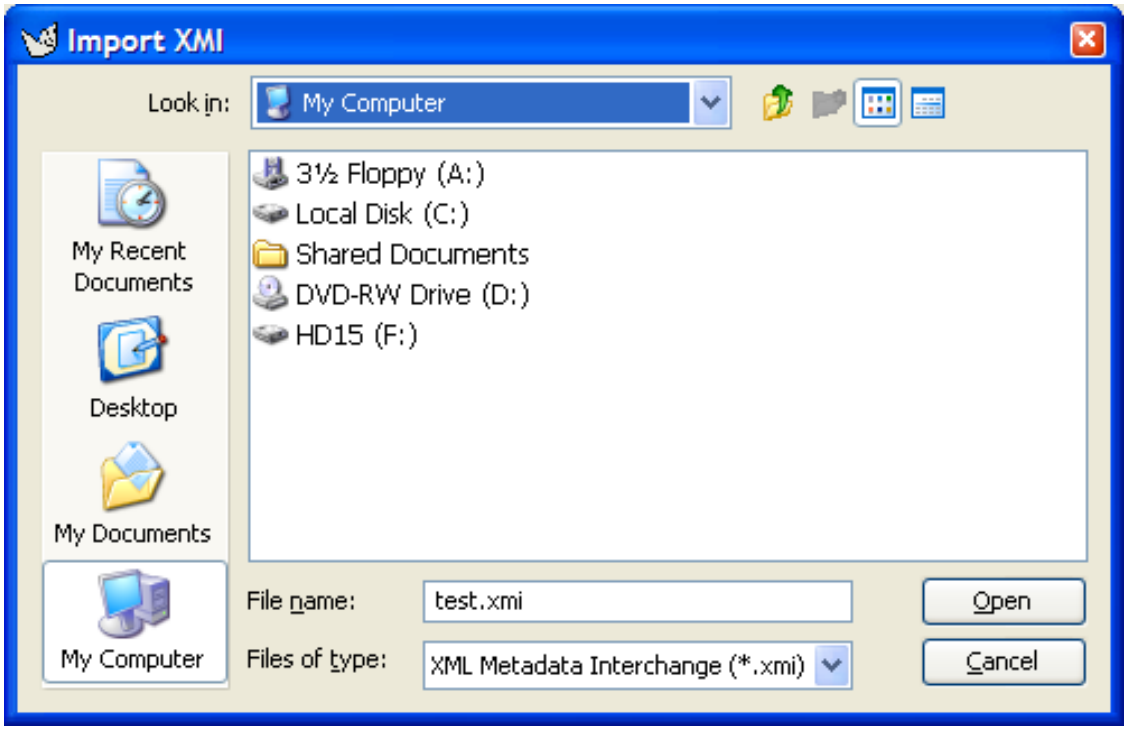
Wenn das Modell geändert wurde (angezeigt durch einen „* “ in der Titelzeile von ArgoUML), dann ist die Aktivierung der „XMI importieren...“-Funktion wahrscheinlich nicht die Absicht des Anwenders, da dies die Änderungen löscht. Aus diesem Grund erscheint die Dialog, der es dem Anwender erlaubt, zuerst 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...`

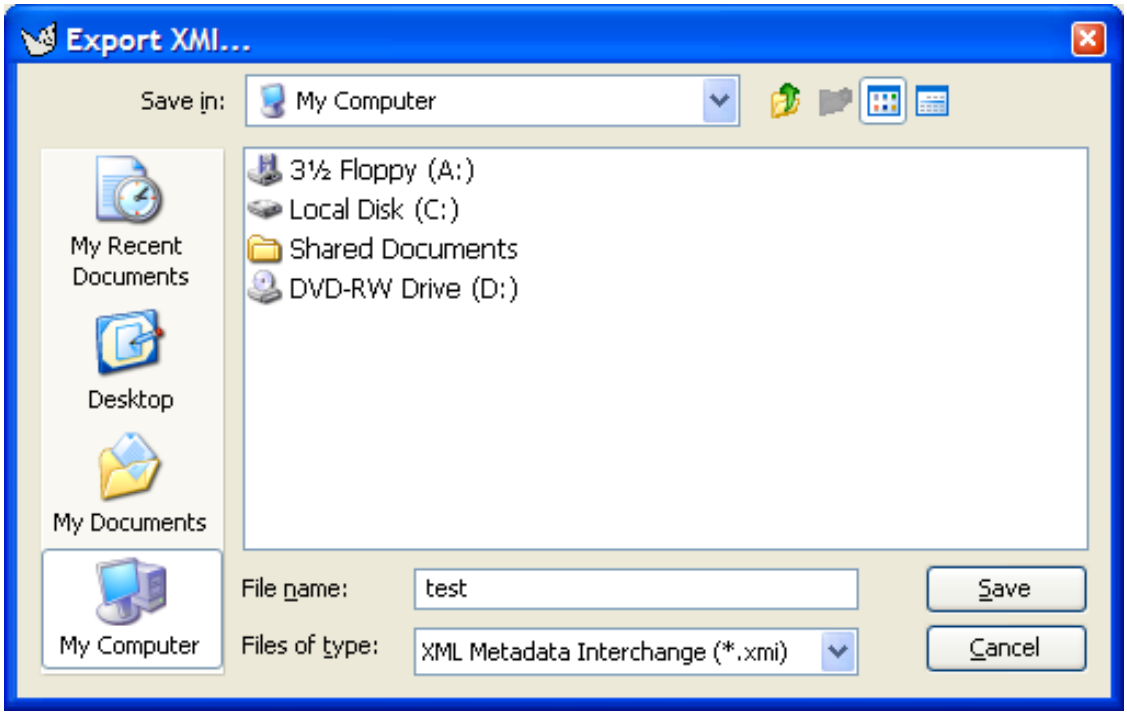


10.3.7. Exportiere als XMI...

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 enthalten wird, 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...`

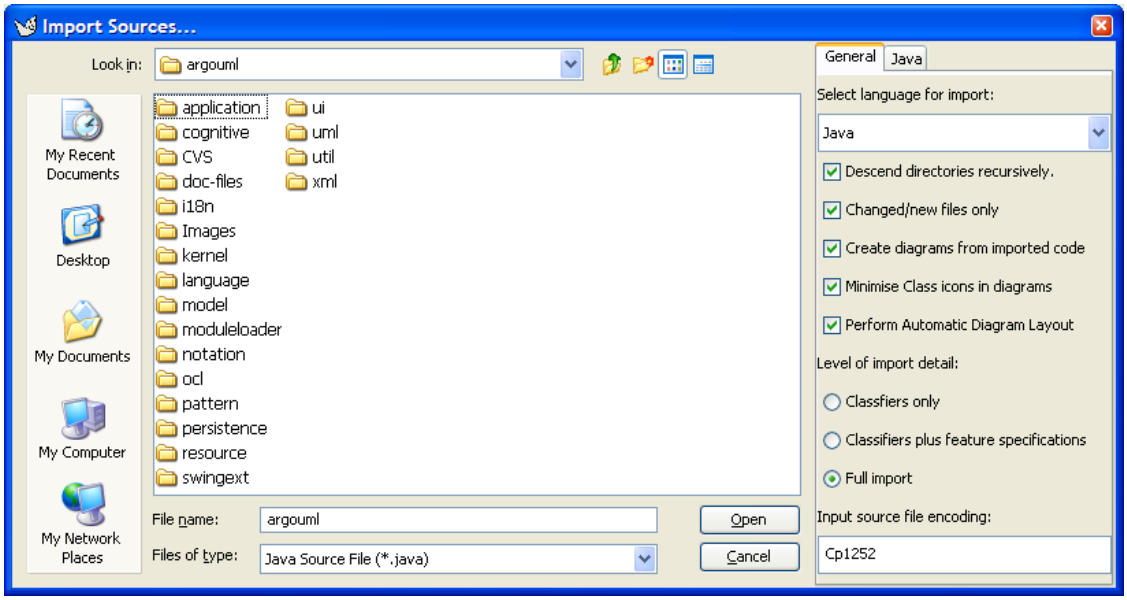


10.3.8. Quellcode importieren...

Eine sehr leistungsfähige Eigenschaft von ArgoUML ist es, das es Java-Code „reengineeren“ kann, um ein Klassendiagramm zu erhalten. Dieser Untermenüeintrag spezifiert den Java-Code, der zum reengineeren importiert werden soll.

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.
- `Minimiere Klassensymbole in den Diagrammen` . Wenn aktiviert, dann werden die Attribut- und Operations- Trenner in den Klassen des generierten Klassendiagrammes nicht angezeigt. Achtung: Dieses Element ist standardmäßig markiert und wird durch viele Anwender übersehen, die dann von dem Ergebnis überrascht sind.
- `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.
- `Codierung der Eingabedatei:`. Der Wert `Cp1252` ist sehr oft der Standard. Diese Zeichenkette repräsentiert den `codierten Zeichensatz- Bezeichnung (CCSID)`.

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.

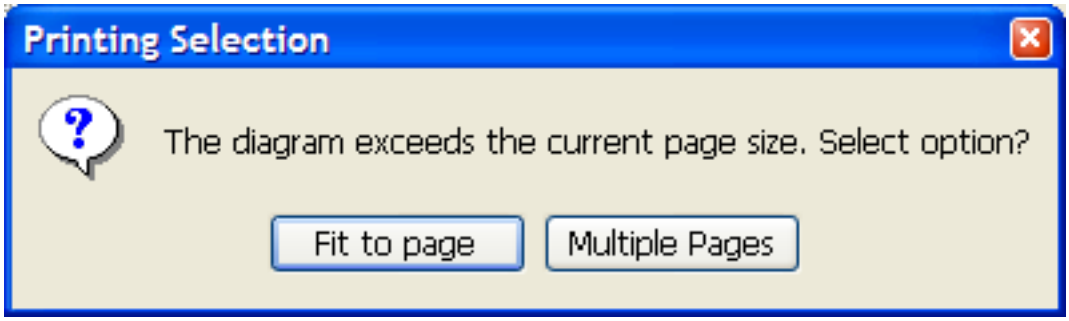
10.3.10. Drucken...

Tastenkürzel Strg-P.

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

In einigen Fällen erscheint der Dialog Abbildung 10.8, „ Der Dialog: Das Diagramm überschreitet die Seitengrösse. “, wenn der Druckvorgang gestartet wurde. Die Auswahl der Schaltfläche „An Seite anpassen“ druckt das gesamte Diagramm auf eine Seite, indem es das Diagramm herunterskaliert. Dies kann dazu führen, dass der gesamte Text bei grossen Diagrammen zu klein zum lesen wird. Aber es ist ein schneller und einfacher Weg einen verwendbaren Ausdruck zu bekommen. Die Auswahl der Option „Mehrere Seiten“ druckt unskaliert, indem es das Diagramm in so viele Einzelteile aufteilt, wie nötig. Drücken der Schliessen-Schaltfläche führt zum Schliessen des Dialoges.

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.

Der Dialog ist mit dem in `Projekt öffnen` verwendeten identisch (siehe Abbildung 10.2, „ Der Dateiauswahldialog für `Projekt öffnen...` . “), mit Ausnahme des Feldes `Dateityp:`. Der ausgewählte Dateityp bestimmt das beim Speichern verwendete Grafikformat. Der Dateiname wird automatisch mit der entsprechenden Dateierweiterung erweitert (wenn er nicht bereits eingegeben wurde). Auf Basis des Diagrammnamens wird ein Standarddateiname generiert.

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, „ 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:

- 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.

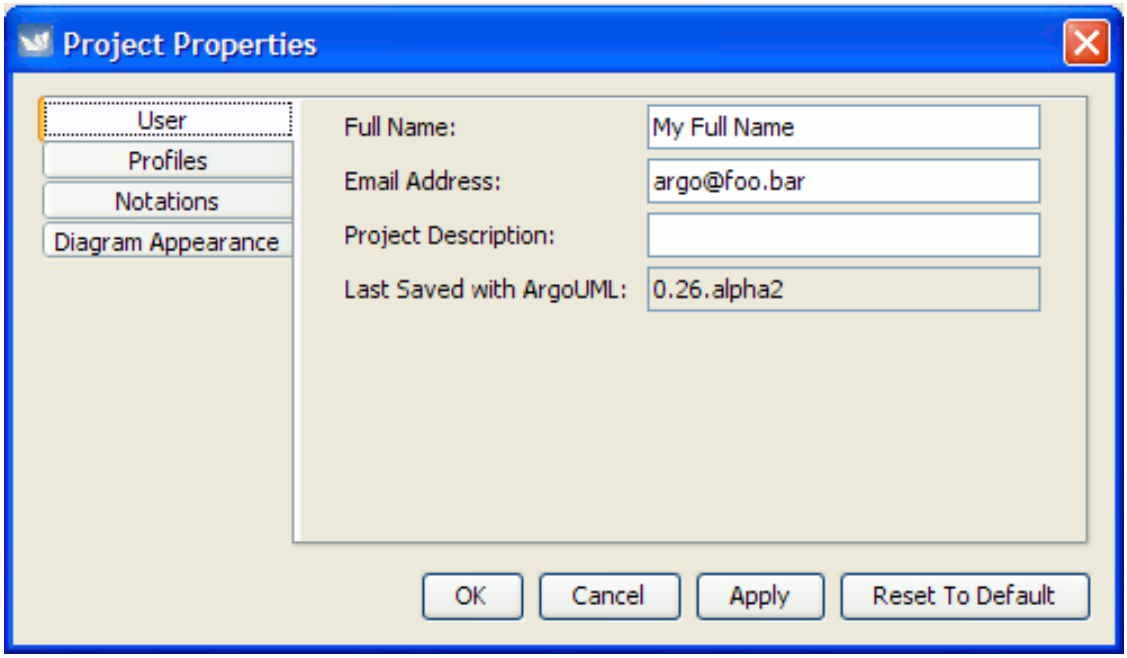
Neben UML ist in V0.22 von ArgoUML nur Java teilweise implementiert.

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.



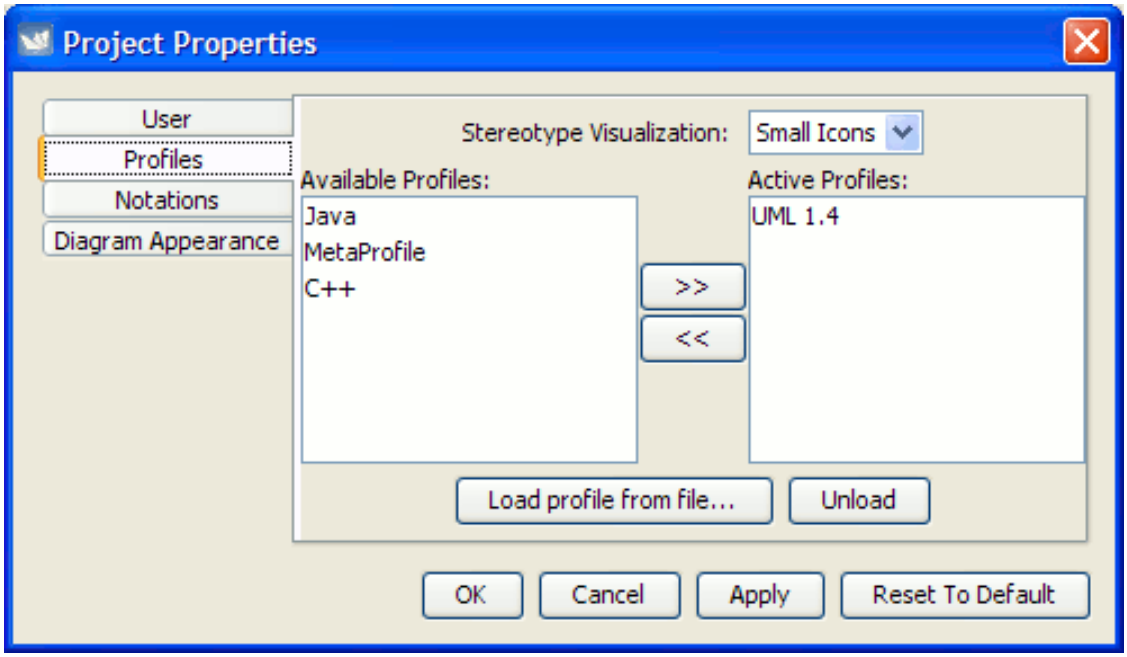
Im Register Benutzer können Sie folgende Felder einstellen:

- 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.
- Das Feld "Zuletzt gespeichert mit ArgUML" gibt die Version von ArgoUML an, mit der dieses Projekt gespeichert wurde (zum letzten Mal gespeichert wurde). Dies kann hilfreich sein, wenn mehrere Designer mit unterschiedlichen Versionen von ArgoUML arbeiten, die nicht immer rückwärtskompatibel sind.

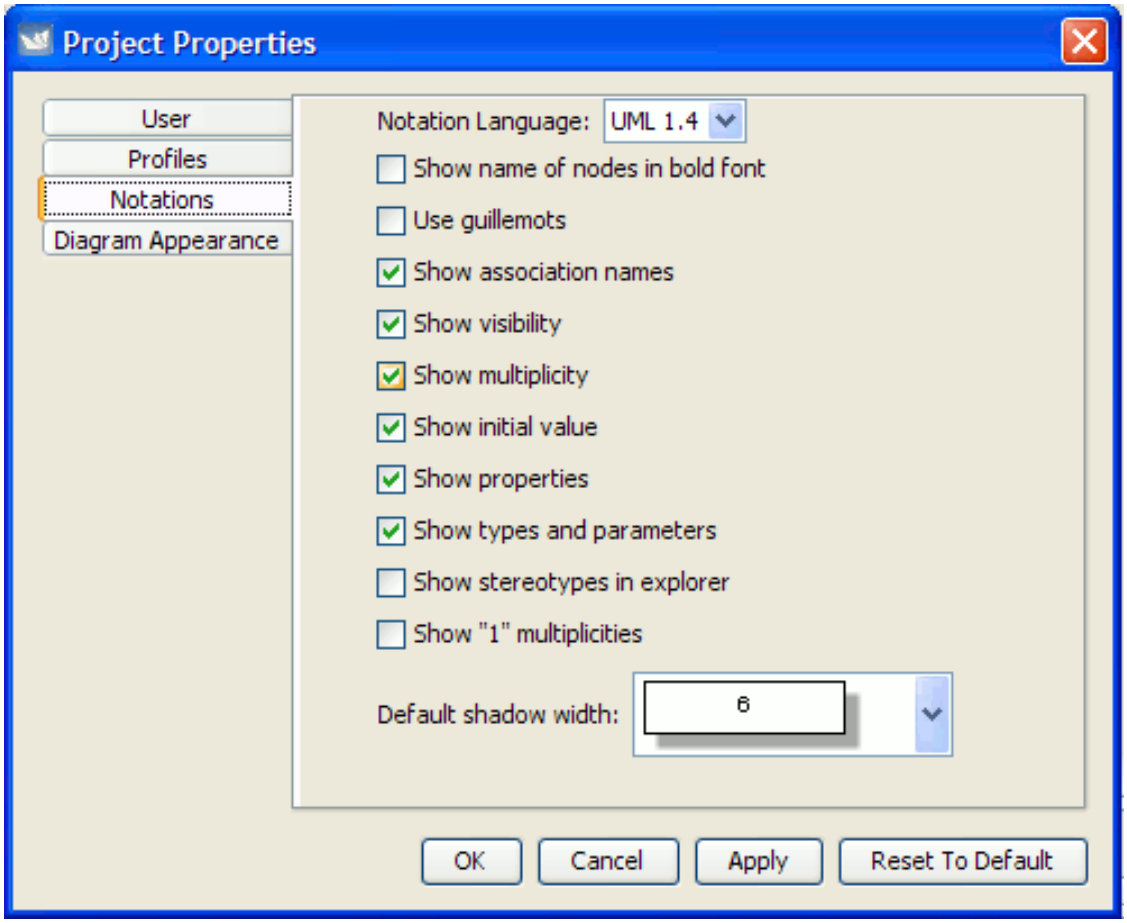
Abbildung 10.10. Der Dialog Projekteinstellungen - Das Register Profile.



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 (standarmäßig 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 (« »).

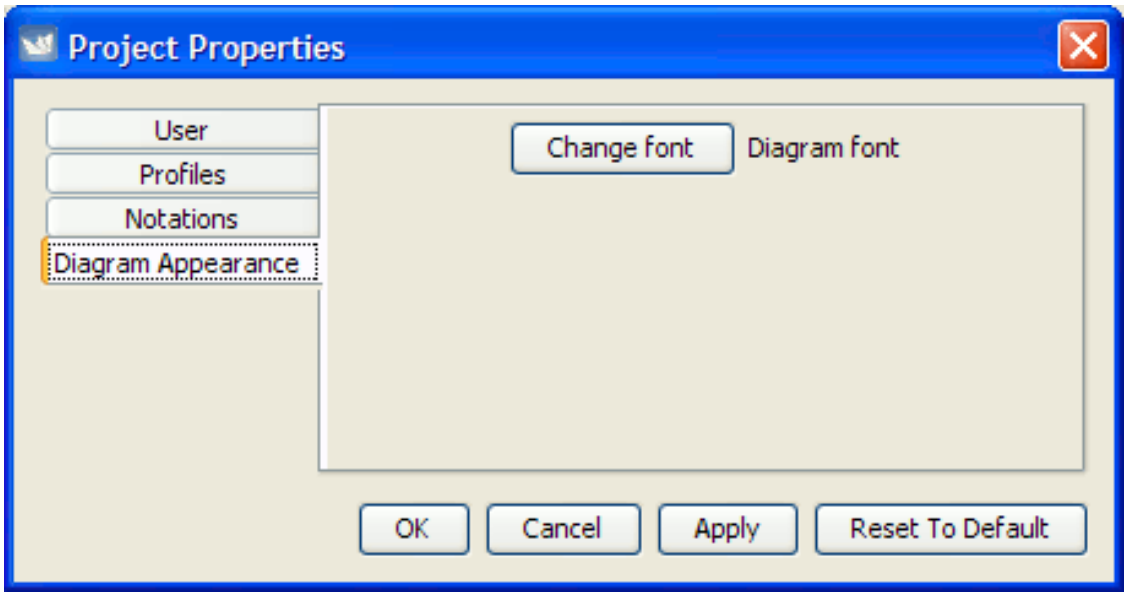
  Diese Eigenschaft wurde ArgoUML vermutlich hinzugefügt, weil französische Anführungszeichen durch die verschiedensten Schriften nur sehr schlecht unterstützt werden und wenn Sie vorhanden sind, dann sind sie sehr klein und schlecht sichtbar.

- `Sichtbarkeit anzeigen` (standardmäßig leer). Ist dies markiert, dann wird ArgoUML die Sichtbarkeitssymbole vor jedem z.B. Attribut im Diagramm anzeigen. In der UML-Notation ist dies das "+" für public, "-" für private, "#" für protected und "~" für Paket. Für ein Attribut könnte es z.B. anzeigen: `+neuesAttr : int` .

- `Kardinalität anzeigen` (standardmäßig leer). Ist dies markiert, wird ArgoUML die Kardinalität von z.B. Attributen im Diagramm anzeigen. In der UML-Notation wird die Kardinalität zwischen [] angezeigt. Wie z.B.: `+neuesAttr [0..*] : int`. Diese Einstellung hat keinen Einfluss auf die Anzeige der Kardinalität an Assoziationsenden.

- `Anfangswerte anzeigen` (standardmäßig leer). Ist dies markiert, wird ArgoUML den Anfangswert eines z.B. Attributes im Diagramm anzeigen. In der UML-Notation wird der Anfangswert wie folgt dargestellt: `+neuesAttr : int = 1`.
- `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 Modellelementsymboles im Explorer anzeigen. Z.B. in der Baumstruktur auf der linken Seite.
- `Standard-Schattenbreite` (standardmäßig auf 1 eingestellt). ArgoUML ist in der Lage, aus ästhetischen Gründen alle Elemente in einem Diagramm mit einem Schatten zu zeichnen. Verwenden Sie diese Einstellung, um die Größe des Schattens einzustellen. Diese Einstellung wird beim Erstellen des Modellelementes verwendet. Das Register "Darstellung" im Detailfenster erlaubt es Ihnen, den Schatten nach dem Erstellen je Modellelement einzustellen. ArgoUML V0.22 behält diese Änderungen allerdings nach dem Speichern und Laden nicht.

Abbildung 10.12. Der Dialog Projekteinstellungen - Das Register Diagramm-Darstellung.



Im Register Diagramm-Darstellung 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.

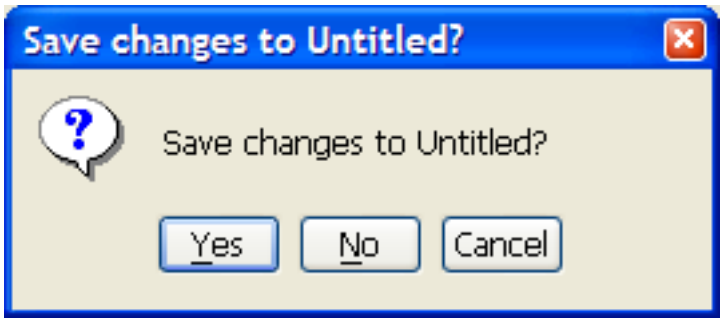
10.3.16. Beenden

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".

Abbildung 10.13. Der Dialog Änderungen speichern.



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.

- `Vorheriges Element`. ArgoUML merkt sich einen Satz von Modellelementen, den Sie während der Navigation durch das Modell markiert haben. Diese Menüoption bringt Sie zu dem zuvor markierten Modellelement. Gibt es keine zuvor markierten Modellelemente, dann ist diese Menüoption deaktiviert.
- `Nächstes Element`. ArgoUML merkt sich einen Satz von Modellelementen, den Sie während der Navigation durch das Modell markiert haben. Diese Menüoption bringt Sie zum nächsten markierten Modellelement (nachdem Sie die Menüoption Zurück verwendet haben). Gibt es keine nächsten Modellelemente, dann ist diese Menüoption deaktiviert.
- `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.

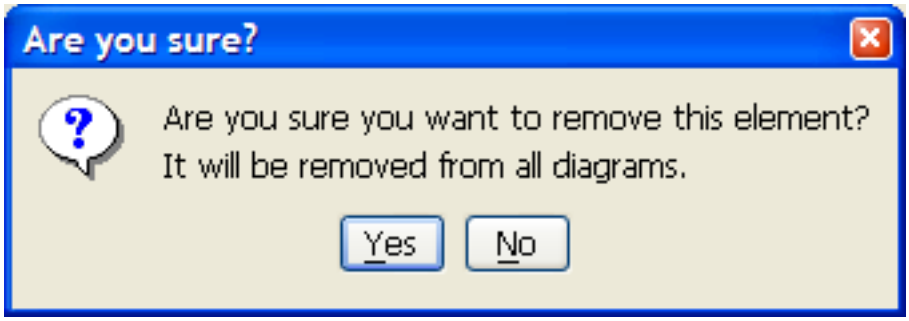
10.4.3. Aus Modell entfernen

Tastenkürzel Strg-Entf.

Diese Funktion löscht die markierten Elemente vollständig aus dem Modell.

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

Abbildung 10.14. Der Bestätigungsdialog zu `Aus Modell entfernen`.



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`. “).

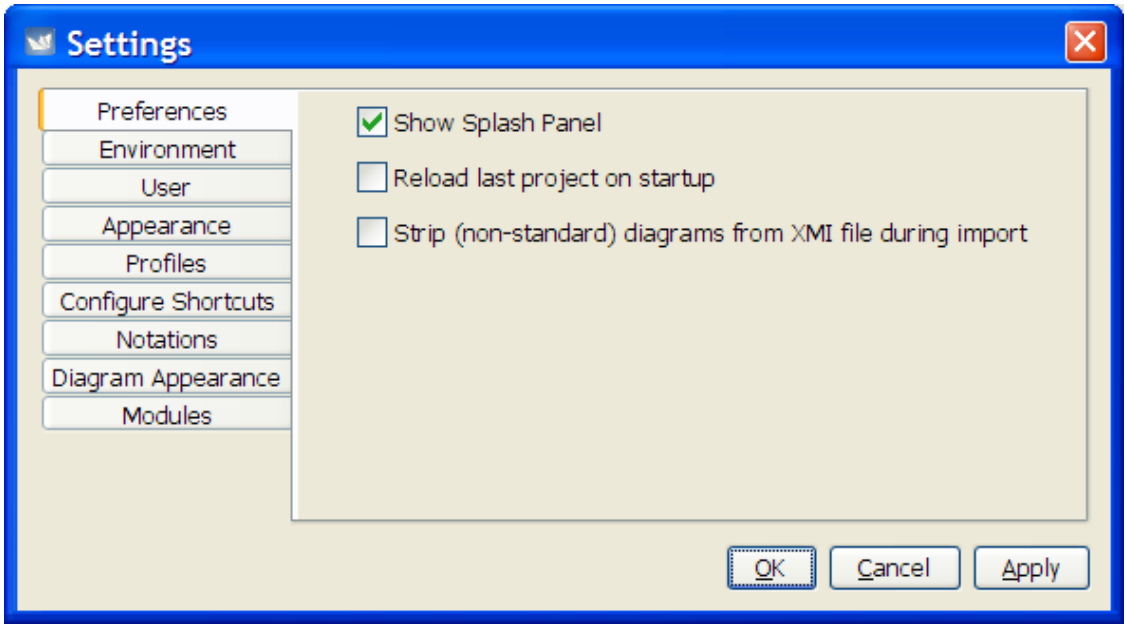
Diese Einstellungen werden persistent für die Nutzung durch nachfolgende ArgoUML-Sitzungen gespeichert.

ArgoUML hat verschiedene benutzerspezifische Konfigurationen, die in diesem Dialog oder direkt in den verschiedenen Fenstern eingestellt werden können. Auch die Lage und Größe des Hauptfensters ist eine solche Einstellung. Die Aktivierung dieser Menüoption veranlasst, dass die Informationen in der Datei `argo.user.properties` gespeichert werden. Der Speicherort dieser Datei ist das "Benutzer-Homeverzeichnis ", welches als `${user.home}` definiert ist und wie in Abschnitt 10.4.5.2, „ Register Umgebung “ beschrieben, bestimmt werden kann.

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.

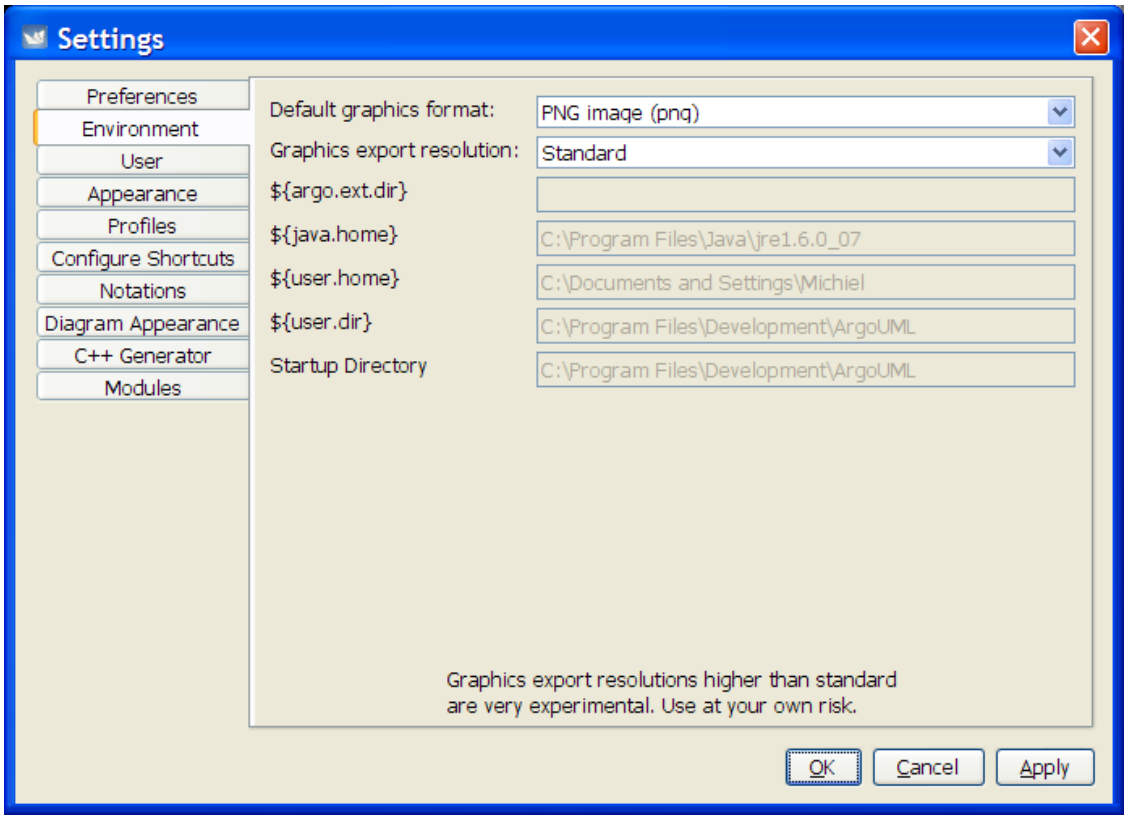
- `Entferne (Nicht-Standard)-Diagramme während des Importes` (standardmäßig leer). Das Markieren dieses Elementes weist ArgoUML an, die "Diagrammelemente" während des importieren der XMI-Dateien zu ignorieren.

  Sie müssen diese Einstellung nur verwenden, wenn ArgoUML einen Fehler während des importierens Ihrer XMI-Datei ausgibt, die besagt, das unbekannte Elemente mit der Bezeichnung "Diagramm" aufgetreten sind. Einige Versioen von Poseidon sind bekannt dafür, dass sie diesen Dateityp standardmäßig erstellen, obwohl es gewöhnlich eine Exportoption gibt, die Erstellung von Standard-XMI-Dateien zu erzwingen.

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.

Abbildung 10.16. Der Dialog `Einstellungen - Umgebung`.



- `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

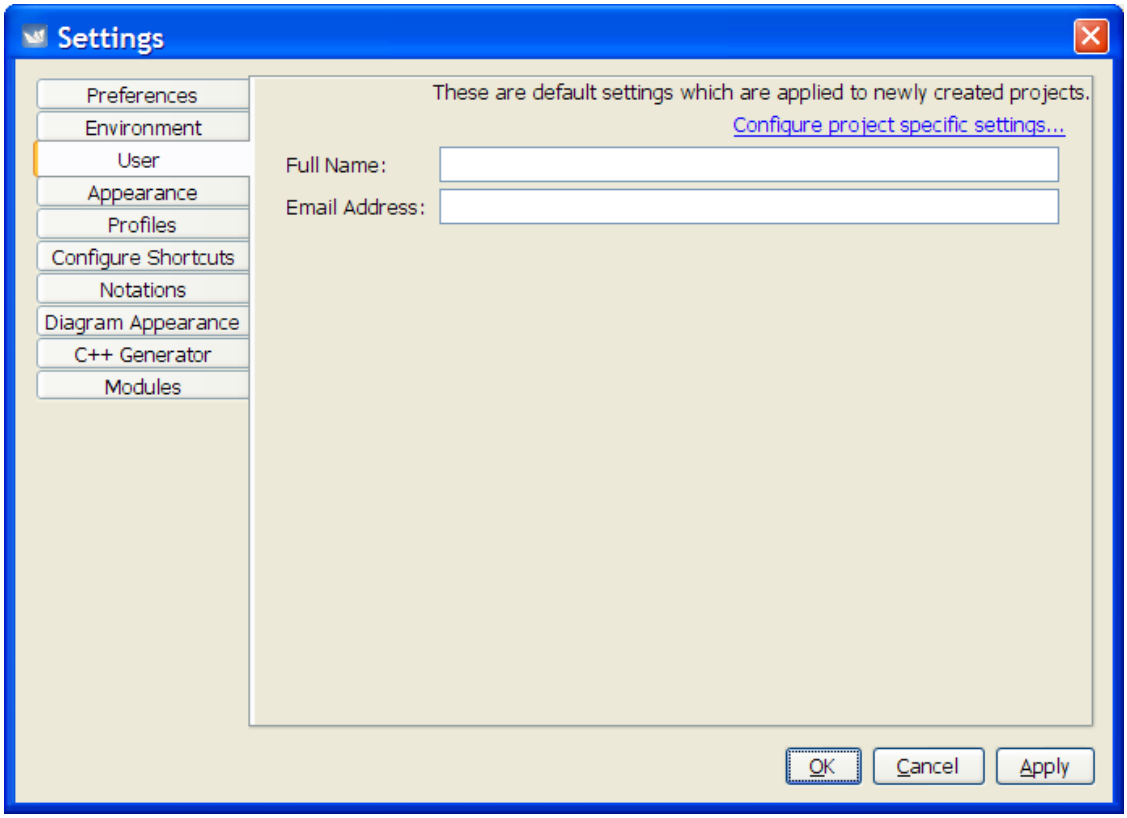
künstlich zu erhöhen. Die vorgegebene Einstellung ist "Standard". Um in der Lage zu sein "Hoch" oder "Extra hoch" einzustellen, müssen Sie gewöhnlich die virtuelle Maschine von Java mit zusätzlich reserviertem Speicher starten.

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

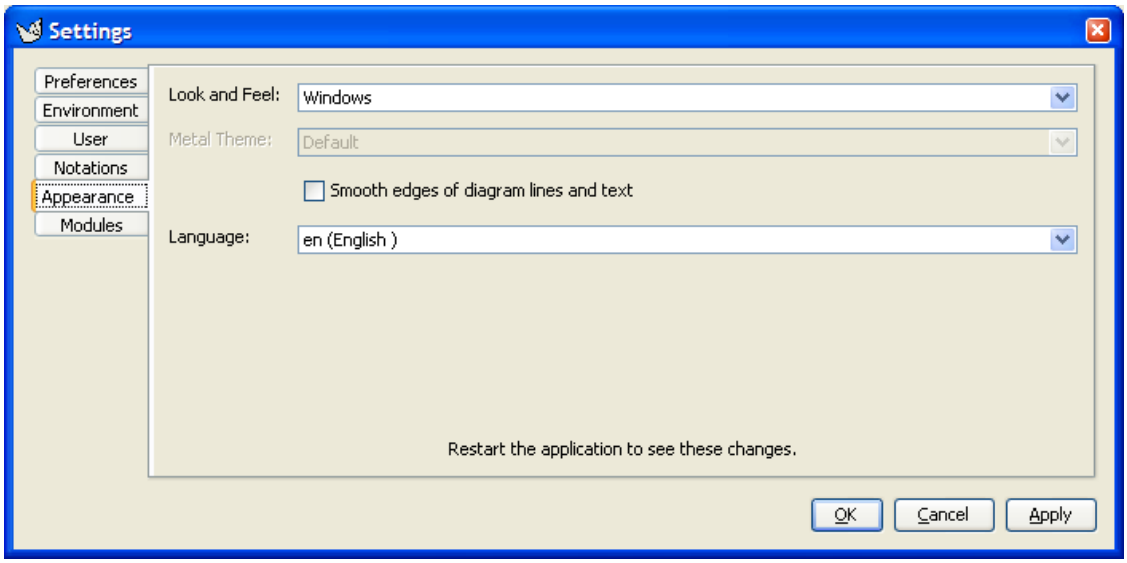
- `Email-Adresse`. Erlaubt es dem Benutzer, seine EMail-Adresse einzugeben.

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 .`

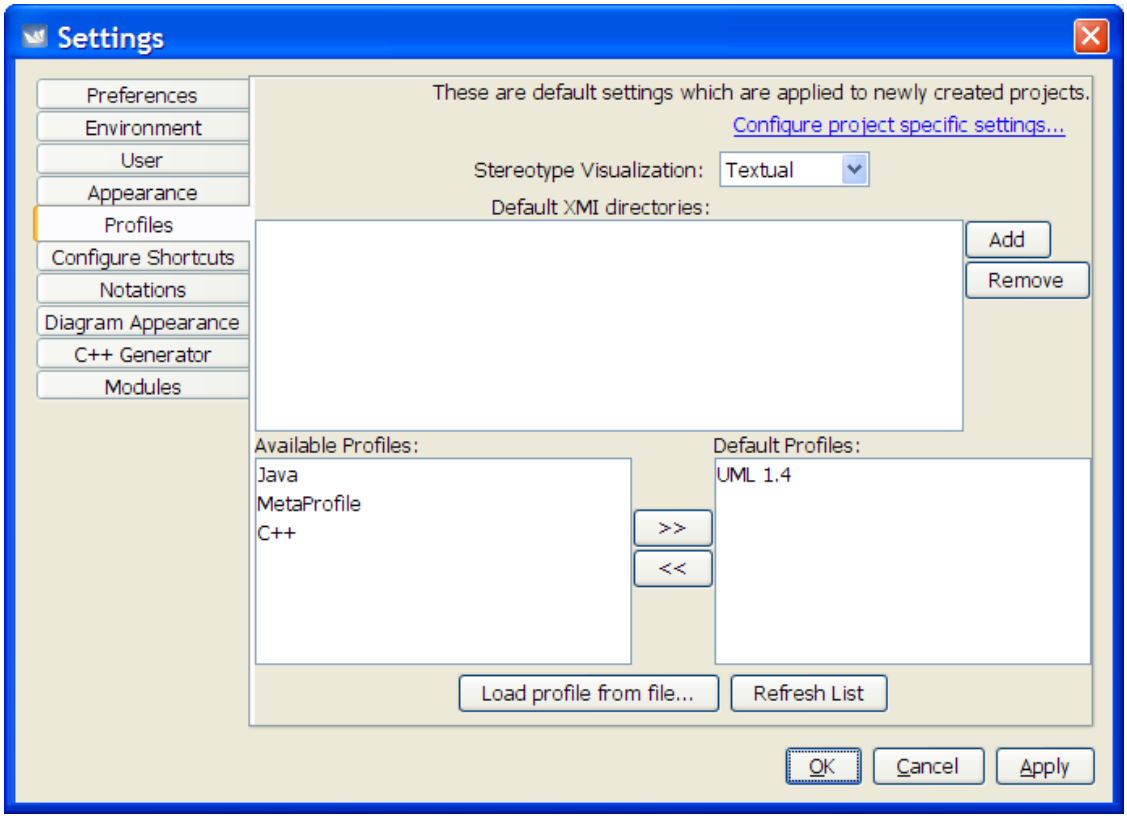


- `Look and Feel`. Die hier gemachte Auswahl beeinflusst die gesamte Anwenderschnittstelle. Die Änderung wird nur wirksam, wenn ArgoUML beendet und neu gestartet wird.
- `Metall-Thema`. Dieses Element ist deaktivert, wenn das Aussehen Metall nicht ausgwählt wurde. Die Auswahl hier beeinflusst die gesamte Anwenderschnittstelle. Die Änderung wird nur wirksam, wenn ArgoUML beendet und neu gestartet wird.
- `Ränder von Diagrammlinien und Text glätten.` Diese Funktion ist in bestimmten Plattformen als „Anti-aliasing“ bekannt. Sie bewirkt, dass diagonale Linien durch die Nutzung von verschiedenen Grauschattierungen nicht so stark gezackt aussehen. Diese Funktion arbeitet nur, wenn es das Betriebssystem unterstützt.

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`.

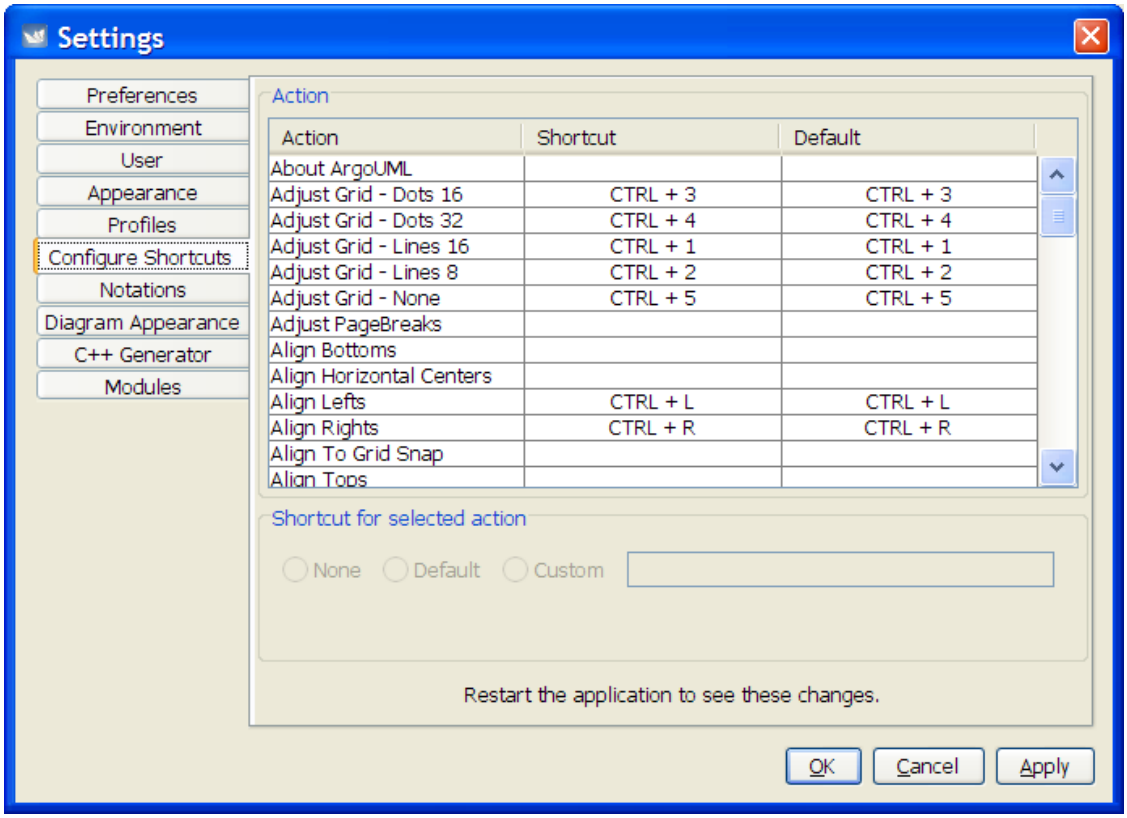


- `Stereotyp-Darstellung` - Auswahl, um Stereotypen als Text, kleine oder große Symbole darzustellen.
- `Standard-XMI-Verzeichnisse` - erlaubt dem Anwender, die Verzeichnisse zu konfigurieren, in denen ArgoUML die benutzerdefinierten Profile finden kann.
- `Standard-Profile` - Auswahl, welches Profil 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`.

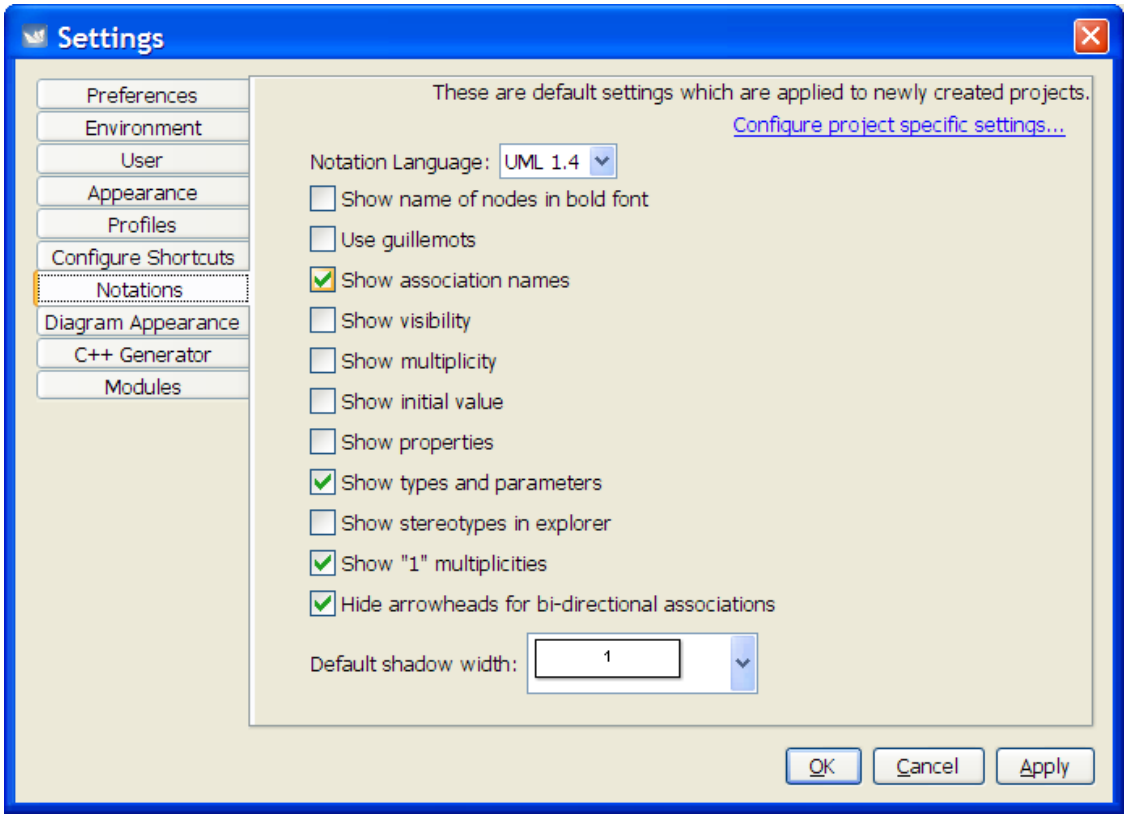


10.4.5.7. Register Notation

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.

Abbildung 10.21. Der Dialog für `Einstellungen - Notationen`.



- `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 irgendjemand 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.

  Unabhhängig 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`.

- `Kardinalitäten anzeigen` (standardmäßig leer). Wenn dies markiert ist, dann wird ArgoUML die Kardinalität z.B. jeden Attributes im Diagramm darstellen. In der UML-Notation wird die Kardinalität zwischen [] gestellt: `+neuesAttr [0..*] : int`. Diese Einstellung hat keine Auswirkung auf die Darstellung der Kardinalität von Assoziationsenden.

- `Anfangswerte anzeigen` (standardmäßig leer). Wenn dies markiert ist, dann wird ArgoUML den Anfangswert z.B. eines Attributes im Diagramm darstellen. In der UML-Notation wird der Anfangswert wie folgt dargestellt: `+neuesAttr : int = 1`.

- `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.

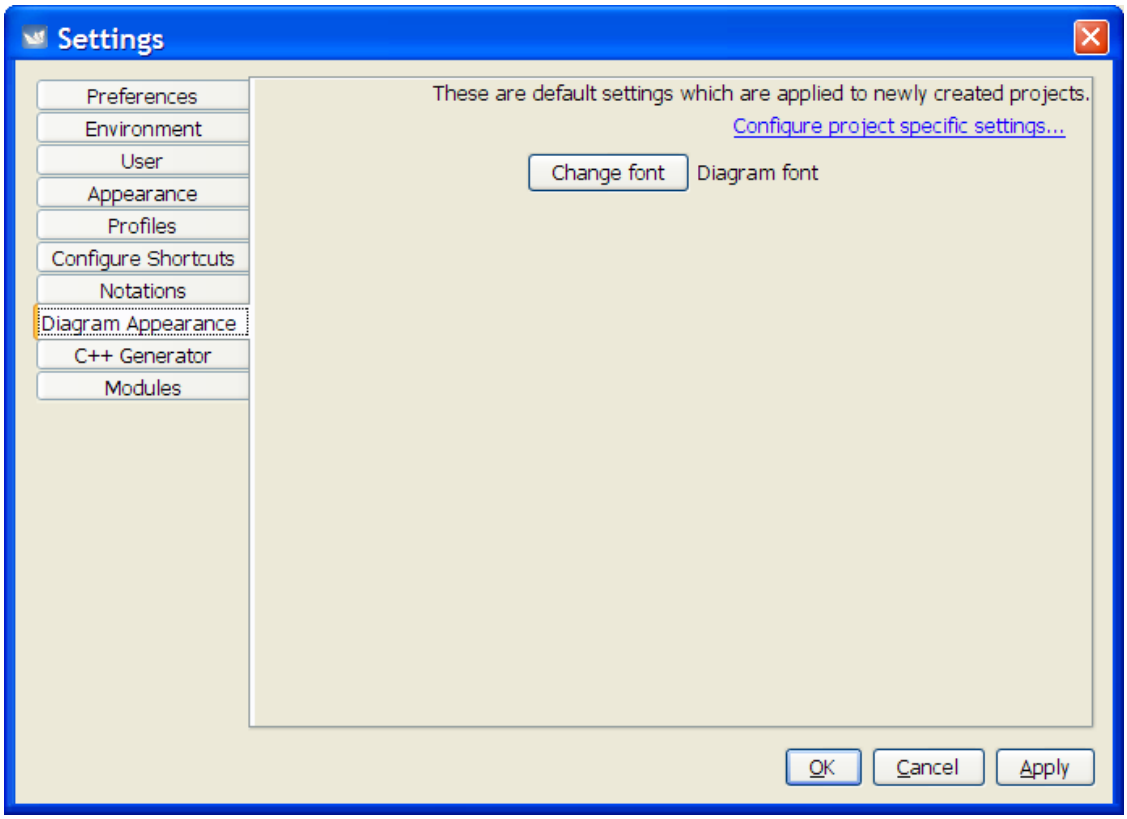
- `Standard-Schattenbreite` (standardmäßig auf 1 eingestellt). ArgoUML ist in der Lage, alle Elemente in einem Diagramm mit einem Schatten zu versehen. Verwenden Sie diese Einstellung, um die Größe des Schattens einzustellen, der beim Erzeugen des Modellelementes verwendet wird.

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)

Abbildung 10.22. Der Dialog für `Einstellungen` - `Diagramm-Darstellung`.



10.4.5.9. Das Register Module

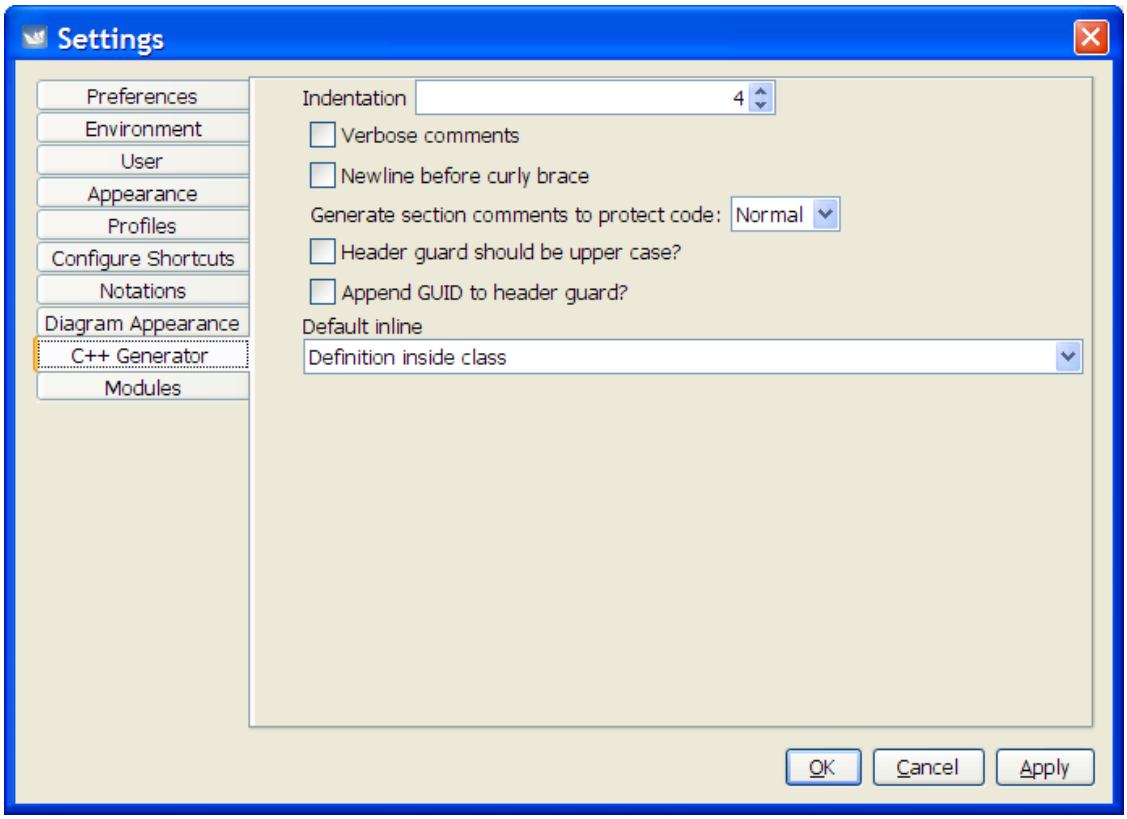
Dieses Register zeigt eine Liste der installierten Module an, die aktiviert oder deaktiviert werden können. Seitdem dies ein neues Konzept in ArgoUML ist, enthält es derzeit eine Liste von Modulen, die nicht entfernt werden können und eine Schaltfläche, um das Konzept zu testen. Das Drücken dieser Schaltfläche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

Abbildung 10.23. Der Dialog für `Einstellungen - C++`.



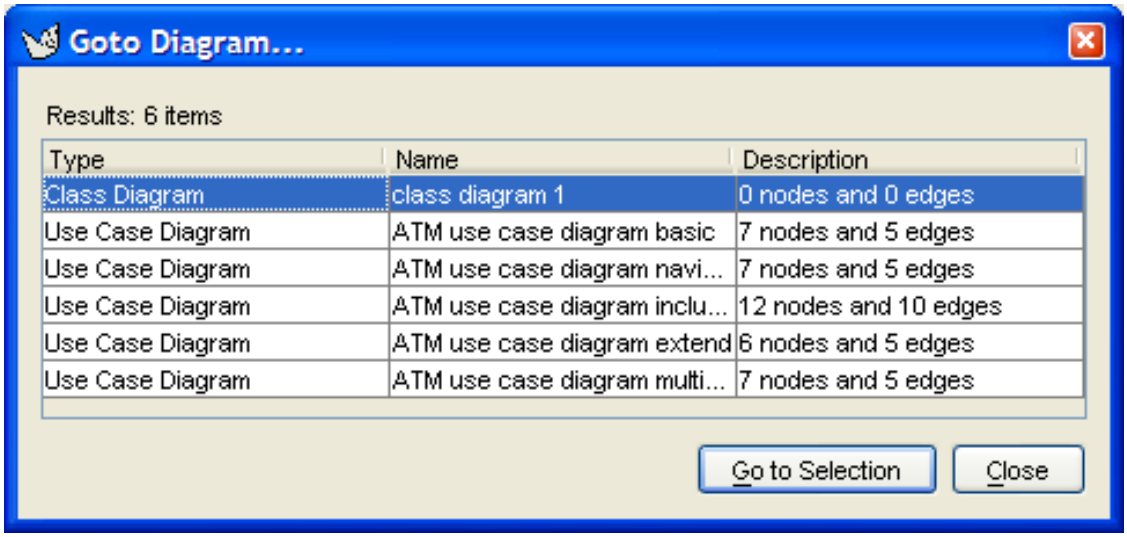
10.5. Das Menü Ansicht

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

10.5.1. Gehezu 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.
- `Beschreibung`. Stellt dar, wieviele Knoten und Kanten sich im Diagramm befinden. Ein Knoten ist ein „2-D “-Modellelement und eine Kante ist eine Verküpfung zu einem Modellelement.

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

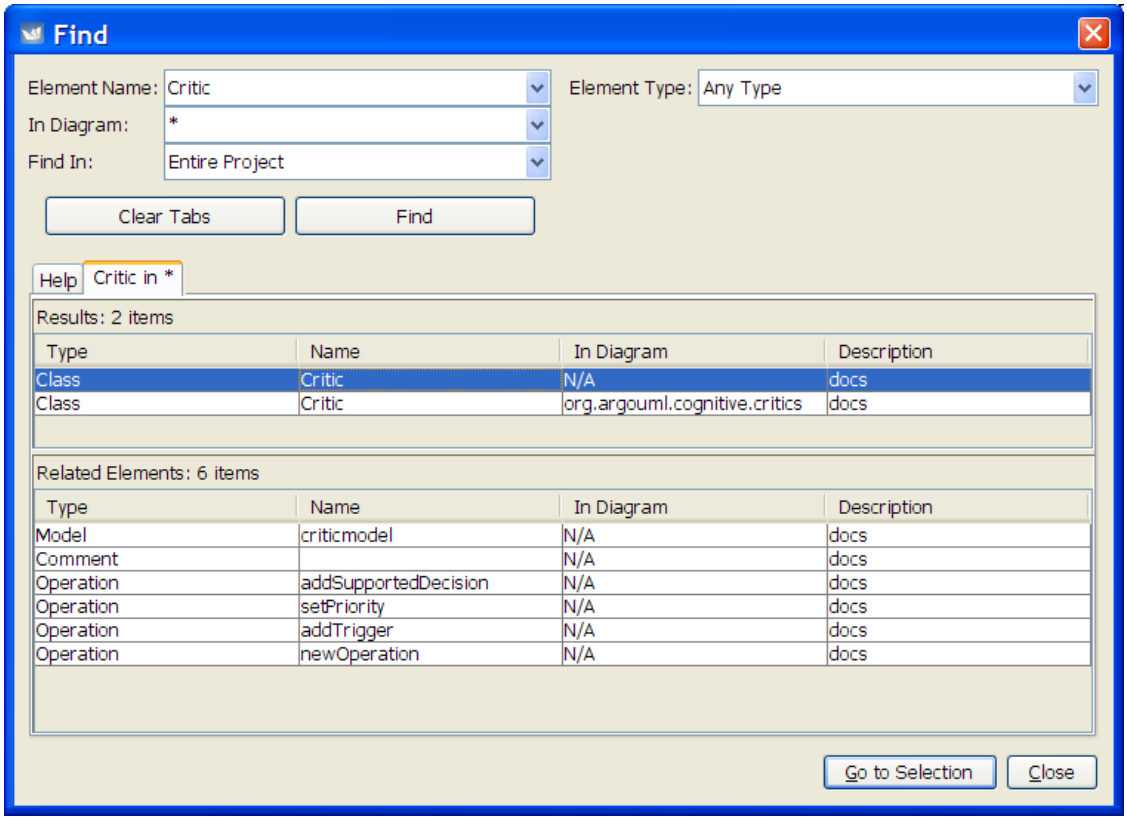
Warnung

Die Version 0.26 von ArgoUML aktualisiert den Dialog nicht sofort, wenn Änderungen in den Diagrammen vorgenommen werden: Änderung des Namens, hinzufügen von Diagrammen, löschen von Diagrammen.

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:

- Ein Textfeld mit der Bezeichnung `Elementname:`, in dem der Name des zu suchenden Modellelementes spezifiziert wird. Die Ersetzungszeichen (`*` und `?` können hier verwendet werden. Ein Pull-down-Menü gewährt den Zugriff auf vorher verwendete Ausdrücke.

- Ein Textfeld mit der Bezeichnung `Im Diagramm:` spezifiziert, welche Diagramme durchsucht werden sollen. Erneut können Ersetzungszeichen verwendet werden. Beide Textfelder haben den Standardeintrag `*`. Z.B. Alles suchen.

- 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.

- Und abschliessend gibt es die Schaltfläche `Suchen` . Diese löst die Suche anhand der in den Text- und Auswahlfeldern spezifizierten Suchkriterien aus. Die Ergebnisse werden in einem Register angezeigt, welche die unteren beiden Drittel der Seite einnehmen.

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.

Die obere Hälfte ist mit `Ergebnis:` und der Anzahl der gefundenen Elemente bezeichnet. Es zeigt eine Tabelle mit einer Zeile mit vier Spalten für jedes Modellelement. Die Breite der Spalten kann eingestellt werden.

- `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

Wenn Sie den Dialog vertikal vergrössern, zeigt es sich, dass der Teil "Zugehörige Elemente" ebenfalls seine Größe ändert, aber nicht der Teil mit den Suchergebnissen. Zwischen diesen Teilen befindet sich jedoch eine Trennlinie. Wenn Sie die Maus darüber bewegen, verwandelt sich der Mauszeiger in das Grössenveränderungs-Symbol und die Begrenzung zwischen diesen beiden Bereichen kann nach oben oder nach unten bewegt werden, um den Platz im Fenster aufzuteilen.

Warnung

Dieser Dialog ist nicht modal, was es erlaubt, dass er während des editierens des Modelles geöffnet 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 16`: vollständiges Gitter, mit einem Zwischenraum von 16 Pixeln.
- `Zeilen 8`: vollständiges Gitter, mit einem Zwischenraum von 8 Pixeln.
- `Punkte 16`: Punkte, mit einem Zwischenraum von 16 Pixeln (der Standard).
- `Punkte 32`: Punkte, mit einem Zwischenraum von 32 Pixeln.
- `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

Dieser Menüeintrag erstellt ein leeres Anwendungsfalldiagramm und markiert das Diagramm im Editierfenster. Ist ein Paket aktuell markiert, dann wird das Anwendungsfalldiagramm innerhalb dieses Paketes erstellt. Das bedeutet, dass es in der Explorerhierarchie (Ansicht: Nach Paketen) als Teil des Paketes dargestellt wird. Im Diagramm erstellte Modellelemente werden im Namensraum des Paketes erzeugt. Dies wirkt sich nicht nur auf das Paket aus, sondern auch auf eine Klasse, Schnittstelle, Anwendungsfall, usw..

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

Dieser Menüeintrag erstellt ein leeres Klassendiagramm und markiert das Diagramm im Editierfenster. Ist ein Paket aktuell markiert, dann wird das Klassendiagramm innerhalb dieses Paketes erstellt. Das bedeutet, dass es in der Explorerhierarchie (Ansicht: Nach Paketen) als Teil des Paketes dargestellt wird. Im Diagramm erstellte Modellelemente werden innerhalb des Namensraumes des Paketes erstellt. Dies wirkt sich nicht nur auf das Paket aus, sondern auch auf eine Klasse, Schnittstelle, Anwendungsfall, usw..

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

Dieser Menüeintrag erstellt ein leeres Sequenzdiagramm und markiert das Diagramm im Editierfenster. Er erzeugt auch ein UML-Element `Kollaboration`, das ein Container für die im neuen Diagramm dargestellten Elemente ist. Wenn eine Klasse aktuell markiert ist, wird ein Sequenzdiagramm und eine Kollaboration erstellt, die das Verhalten dieser Klasse repräsentieren. Das bedeutet, dass die erstellten Elemente in der Explorerhierarchie (Ansicht: Nach Paketen) als Teil der Klasse dargestellt werden. Im Diagramm erstellte Modellelemente werden innerhalb des Namensraumes der Kollaboration erzeugt. Ein Sequenzdiagramm muss nicht nur das Verhalten einer Klasse repräsentieren, sondern auch jeden anderen Klassifizierer, wie zum Beispiel eine Schnittstelle, einen Anwendungsfall, usw.. Es ist auch möglich, Sequenzdiagramme für eine Operation zu erstellen.

10.6.4. Kollaborationsdiagramm

Dieser Menüeintrag erstellt ein leeres Kollaborationsdiagramm und markiert das Diagramm. Es erstellt auch ein UML-Element `Kollaboration`, das ein Container für die im neuen Diagramm dargestellten Elemente ist. Wenn ein Paket markiert ist, wenn dieser Menüeintrag aktiviert wird, wird das Kollaborationsdiagramm unterhalb einer Kollaboration innerhalb dieses Paketes erstellt. Das bedeutet, dass es in der Explorerhierarchie (Ansicht: Nach Paketen) als Teil der Kollaboration innerhalb des Paketes dargestellt wird. Im Diagramm erstellte Modellelemente werden im Namensraum der Kollaboration des Paketes erstellt.

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.

10.6.5. Zustandsübergangsdiagramm

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

Zustandsübergangsdiagramme sind mit einem Modellelement mit dynamischem Verhalten verknüpft, wie z.B. einem Klassifizierer oder einer Verhaltenseigenschaft, welche den Kontext für den zu repräsentierenden Zustandsautomaten enthält. Passende Modellelemente sind zum Beispiel eine Klasse, eine Operation und ein Anwendungsfall. Wenn kein solches Elemente zum Zeitpunkt des aktivierens des Menüs `Neues Zustandsübergangsdiagramm` markiert ist, dann wird eine ungebundener Zustandsautomat erstellt. Um ein korrektes UML-Modell zu erhalten, müssen Sie den Kontext des Zustandsautomaten im Detailfenster einstellen.

10.6.6. Aktivitätsdiagramm

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

Aktivitätsdiagramm sind mit einem Modellelement mit dynamischem Verhalten verknüpft, wie z.B. Pakete, Klassifizierer (einschliesslich Anwendungsfällen) und Verhaltenseigenschaften. Passende Modellelemente sind z.B. eine Klasse, ein Anwendungsfall, eine Operation und ein Paket. Wenn ein solches Element zum Zeitpunkt des aktivieren des Menüs `Neues Aktivitätsdiagramm` nicht

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 Verteilungsdiagramm 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 Gittereinrastgrenzen aus (siehe Abschnitt 10.5.5, „ Einrasten einrichten “).

Tipp

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

Auswirkungen bis Sie entweder die Gittereinrasteinstellungen auf einen größeren Wert eingestellt haben oder einen der anderen `Anordnen`-Menüeinträge verwendet haben, um die Elemente aus ihren ursprünglichen Positionen zu bewegen.

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 Modellelementes zur linken Kante des rechten Modellelementes) 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 Modellelementes zur oberen Kante des unteren Modellelementes) 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.

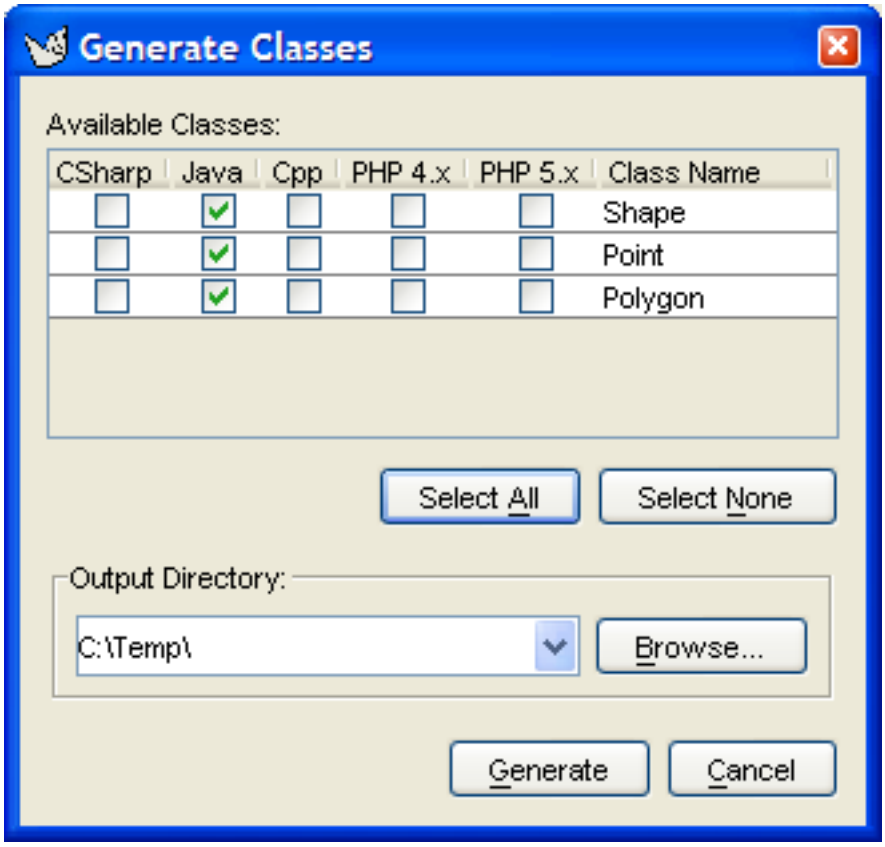
Warnung

Codegenerierung ist natürlich eine sehr fortschrittsbezogene Arbeit. Die aktuelle Version von ArgoUML wird ein strukturelles Template für Ihren Code generieren, aber es ist nicht in der Lage mit der Verhaltens-Spezifikationen umzugehen, um Code für das dynamische Verhalten des Modelles zu generieren.

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.

Der untere Teil des Dialoges ist ein editierbares Kombinationsfeld mit der Beschriftung `Ausgabeverzeichnis`, um das Verzeichnis festzulegen, in das der Code generiert wird. Innerhalb dieses Verzeichnisses wird ein Oberverzeichnis mit dem Namen des Modelles erstellt. Weitere Unterverzeichnisse werden erzeugt, welche die Hierarchie der Pakete/Namensräume des Modelles reflektieren. Ein Pull-down-Menü erlaubt den Zugriff auf vorher verwendete Ausgabeverzeichnisse.

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 Hinweise

Dieses Menü steuert eines von ArgoUML's Alleinstellungsmerkmalen; die Verwendung von Hinweisen, um den Designer anzuleiten. Die dahinterstehende Theorie ist ausführlich in Jason Robbins' PhD-Dissertation beschrieben http://argouml.tigris.org/docs/robbins_dissertation/ [http://argouml.tigris.org/docs/robbins_dissertation/].

Anmerkung

Ein Wort zur Terminilogie: Die *Hinweise* sind Hintergrundprozesse, die das aktuelle Modell anhand verschiedener „guter“ Designkriterien überprüfen. Es gibt jeweils einen Hinweis für jedes Designkriterium.

Die Ausgabe eines Hinweises ist eine *kritische Beschreibung* - eine Ausführung zu einigen Aspekten des Modelles, 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 Hinweise sind asynchrone Prozesse die parallel zum Hauptprozess von ArgoUML ablaufen. Änderungen benötigen üblicherweise eine oder zwei Sekunden bis die Hinweise verfügbar sind.

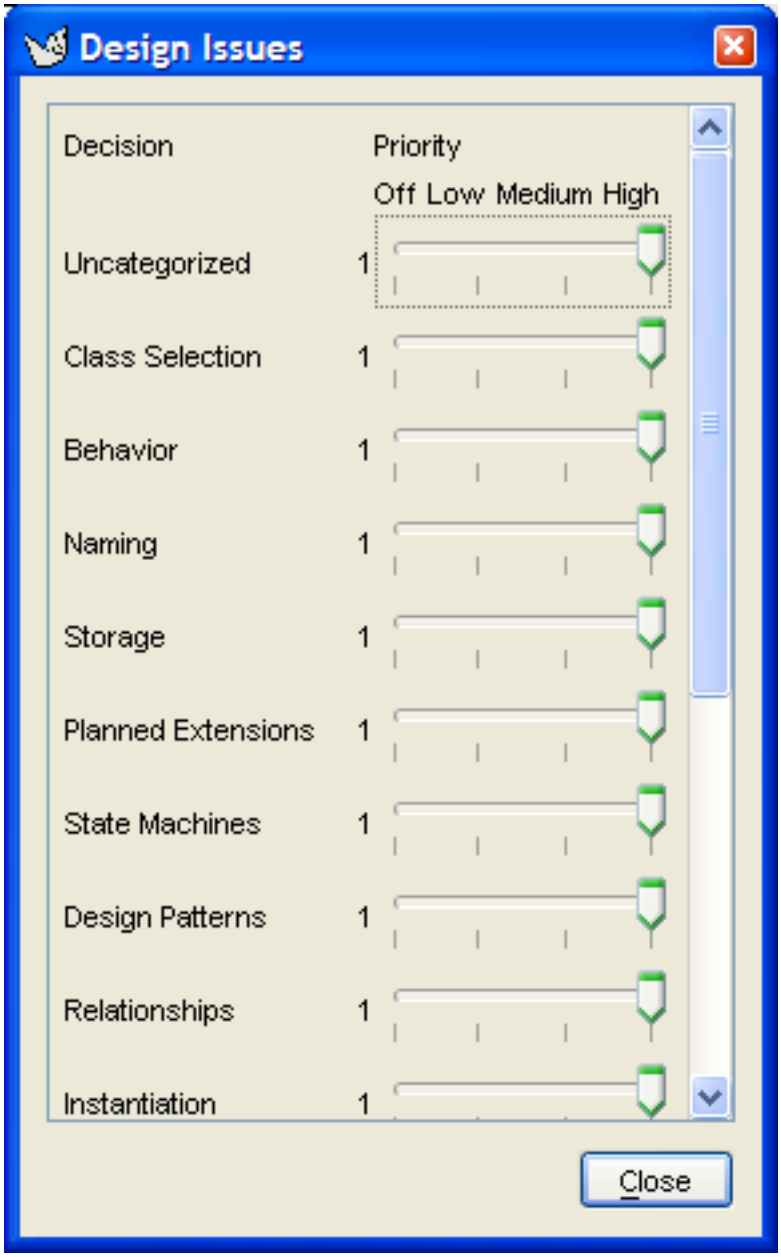
10.9.1. Hinweise ein-/ausschalten

Dies ist ein Markierfeld, welches steuert, ob die Hinweise eingeschaltet sind. Standardmäßig ist das Feld markiert. Ist es nicht markiert, sind alle Hinweise ausgeschaltet und jedes, durch die Hinweise 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 Hinweise angewendet werden (siehe Abbildung 10.27, „ Der Dialog für `Design-Wichtungen...` “).

Abbildung 10.27. Der Dialog für `Design-Wichtungen...`



ArgoUML kategorisiert Hinweise je nachdem wie die Designwichtungen diese adressieren. Es gibt 16 solcher Kategorien. Die Hinweise in jeder Kategorie werden detailliert im Kapitel über Hinweise (Kapitel 15, *Die Hinweise*) diskutiert.

Die Schieberegler können für jede Kategorie eingestellt werden, um die Hinweise zu steuern, die in dieser Kategorie ausgelöst werden. Das Verschieben eines Reglers auf `Aus` schaltet alle Hinweise 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 Hinweise innerhalb der Design-Hinweiskategorie freischalten (`Aus` ist die niedrigste Priorität)

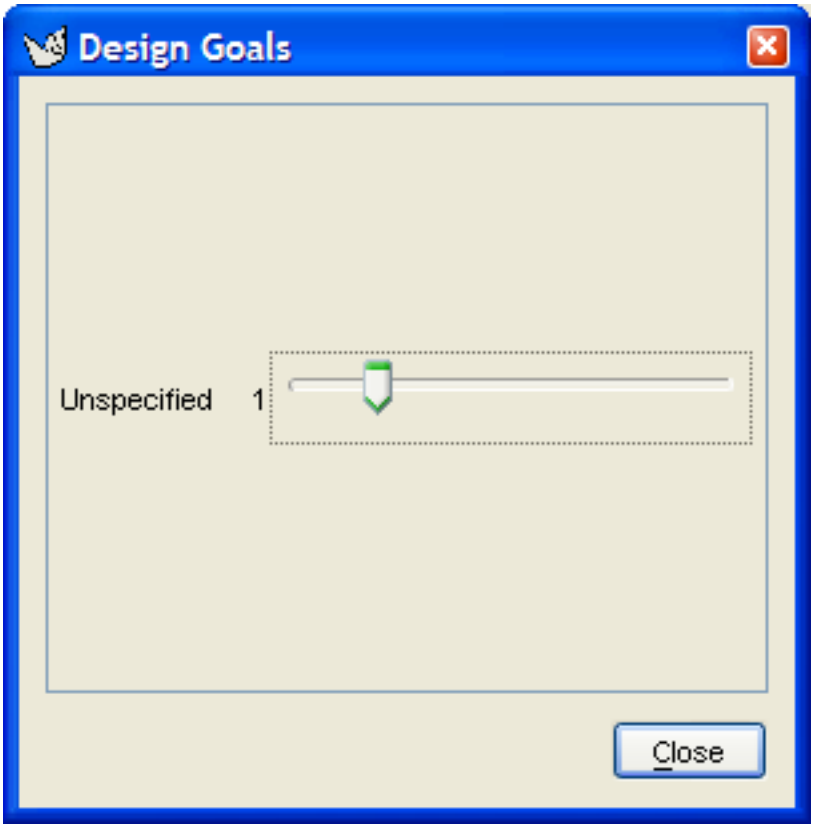
Anmerkung

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.). Hinweise sind mit einem oder mehreren Zielen verknüpft.

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

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

Die Einstellung eines Reglers auf einen höheren Wert wird alle Hinweise 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 Hinweise entsprechend dem Ergebnis der OOA&D und nicht mit der Gruppierung entsprechend der Struktur von UML.

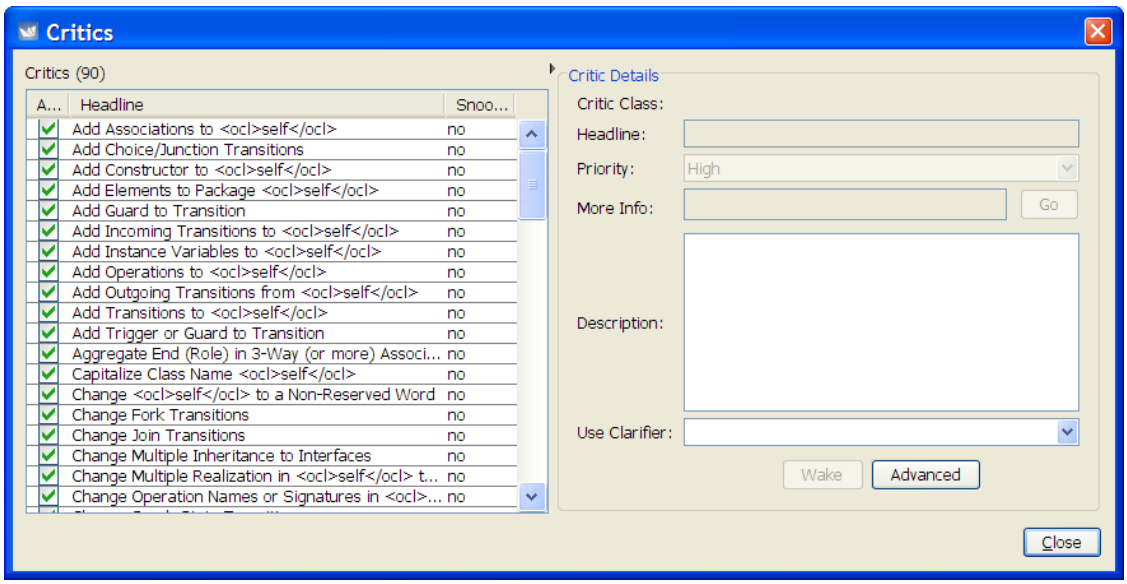
Warnung

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 Hinweise und hat somit keine Auswirkungen.

10.9.4. Hinweise anzeigen...

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

Abbildung 10.29. Der Dialog für `Hinweise anzeigen...`



Dieser Dialog steuert das Verhalten der einzelnen Hinweise. Links befindet sich eine Liste aller Hinweise, um diese individuell ein- oder ausschalten zu können. Für jeden Hinweis gibt es drei Spalten, beschriftet mit `Aktiv`, `Titel` und `deaktiviert`. Die erste davon ist ein Markierfeld, das mit Taste 1-Klicks verändert werden kann. Die zweite ist der Titel des Hinweises und die dritte zeigt an, wenn der Hinweis im "Zu bearbeiten"-Fenster deaktiviert wurde (siehe Kapitel 14, *Der Bereich Zu-Bearbeiten* . Ein Hinweis ist nur dann wirklich aktiv, wenn das Markierfeld in der ersten Spalte markiert ist *und* der Hinweis nicht deaktiviert wurde.

Jeder Hinweis, bei dem das Markierfeld in der ersten Spalte nicht markiert ist, ist inaktiv und wird nicht ausgelöst. Zusätzlich wird jedes, mit diesem Hinweis verknüpfte "Zu bearbeiten"-Element aus dem "Zu bearbeiten"-Fenster entfernt.

Die Version 0.26 von ArgoUML umfasst 90 Hinweise, einige davon sind unvollständig implementiert. Sie sind je Designwichtungskategorie im Kapitel Hinweise detailliert beschrieben (siehe Kapitel 15, *Die Hinweise*).

Rechts von der Liste gibt es eine Reihe von Feldern, mit `Details zum Hinweis` bezeichnet, die eine detaillierte Kontrolle über die einzelnen Hinweise gibt. Das Markieren eines Hinweises in der linken Liste wird die Felder für diesen Hinweis füllen.

Das erste Feld rechts ist mit `Klasse:` bezeichnet. Darauf folgt der vollständige Name der Klasse in ArgoUML, welche den Hinweis implementiert. Dieser Name kann aus eindeutiger Bezeichner für diesen Hinweis verwendet werden, z.B. bei der Kommunikation über diesen Hinweis.

Das erste Feld danach ist ein Textfeld mit der Beschriftung `Titel:`. Dieses Textfeld beinhaltet den vollständigen Titel des Hinweises (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 der Hinweis ausgelöst wird.

Das nächste Feld ist ein mit `Priorität` beschriftetes Pull-down-Menü. Die drei verfügbaren Optionen sind `Hoch`, `Mittel` und `Niedrig` und spezifizieren die Prioritätskategorie eines jeden "Zu bearbeiten"-Elementes dieser Hinweise. Dies ändert nich die Priorität bereits existierender "Zu bearbeiten"-Elemente. Nur die neu generierten. Die Änderung der Priorität eines Hinweises wird nicht dauerhaft gespeichert.

Das nächste Feld ist mit `Mehr Informationen:` beschriftet und enthält eine URL, die auf weitergehende Informationen zeigt. Mit der rechts befindlichen Schaltfläche `Gehe zu` 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 der Hinweis 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 der Hinweis ausgelöst wird.

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

Kennzeichen sind Symbole und rote Wellenlinien in aktuellen Diagrammen, um den Artefakt zu kennzeichnen, auf den sich der Hinweis bezieht. Die ursprüngliche Absicht war es, die Verbindung zwischen den Hinweisen und den Kennzeichen etwas flexibler zu machen.

Ein Benutzer möchte z.B. den Hinweis `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. Hinweise, 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, einen Hinweis im "Zu bearbeiten"-Fenster zu deaktivieren (siehe Kapitel 14, *Der Bereich Zu-Bearbeiten*), was den Hinweis für eine bestimmte Zeit ausschaltet. Wenn der Hinweis deaktiviert wurde, wird diese Schaltfläche aktiviert und der Hinweis wieder aktiviert. Ansonsten ist sie deaktiviert.

Tipp

Sie können einen deaktivierten Hinweis erkennen, da dies in der linken Liste in der dritten Spalte angezeigt wird.

- `Erweitert`. Diese Schaltfläche veranlasst ArgoUML einige zusätzliche Spalten in der Tabelle der Hinweise anzuzeigen. Sie erlauben eine detailliertere Untersuchung der Eigenschaften eines Hinweises.

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 generieschen Menüerweiterungspunkt für die in ArgoUML enthaltenen Plugins. Das Standardsystem hat kein Plugin und dieser Menüeintrag ist standardmäßig leer.

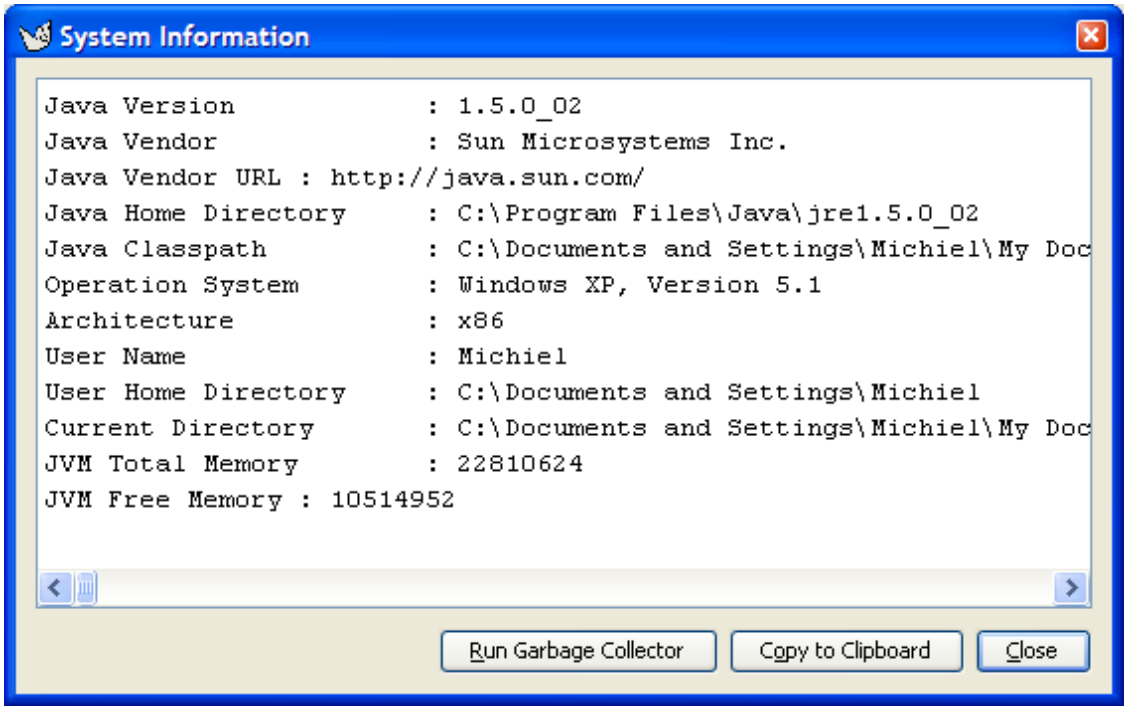
10.11. Das Menü Hilfe

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`.



Verwenden Sie dieses Menü, um das System für den Systemmanager oder Entwickler zu beschreiben auf dem ArgoUML läuft. Das Drücken der Schaltfläche `Starte Speicherbereinigung (GC)` startet nicht nur den Java Garbage Collector sondern aktualisiert auch die dargestellten Informationen. Um das Kopieren und Einfügen in (z.B.) eine E-Mail zu unterstützen, dafür ist die Schaltfläche `In Zwischenablage kopieren` vorgesehen. Die Schaltfläche `Schliessen` schließt den Dialog.

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*).

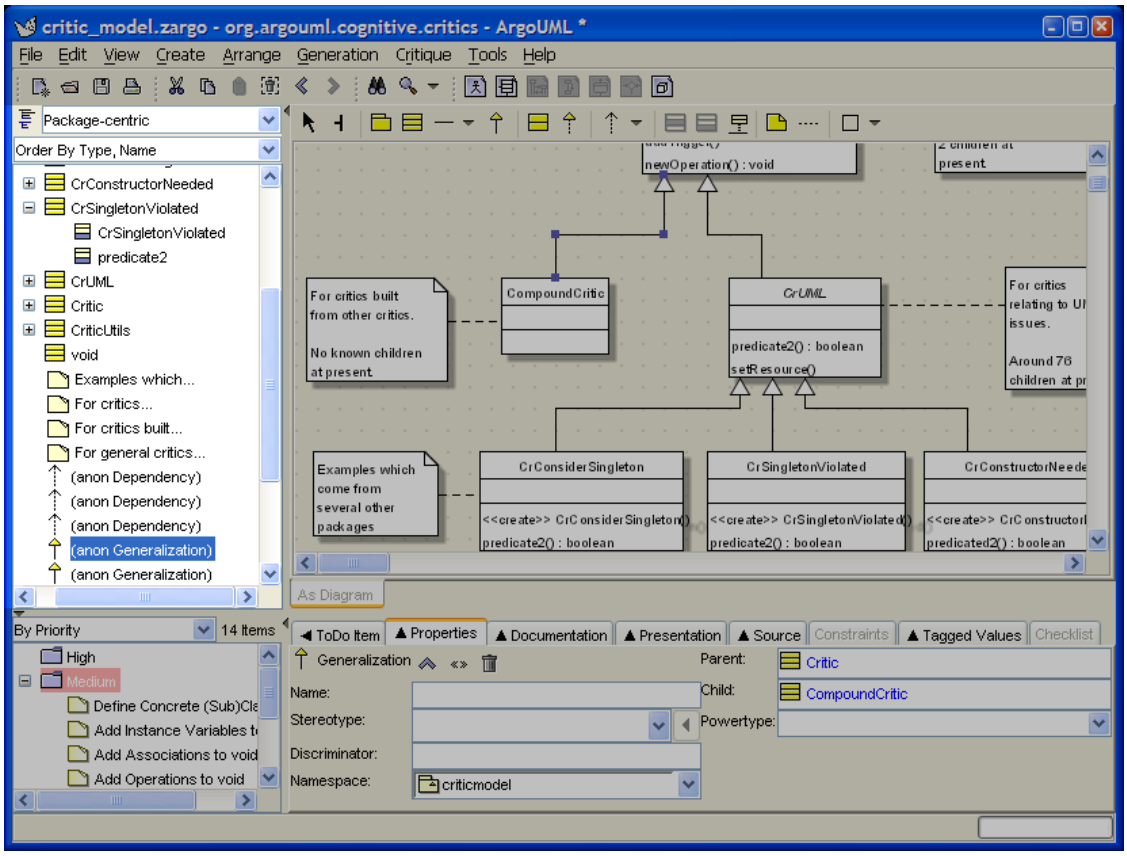
Kapitel 11. Der Explorer

Der Explorer wurde vorher Navigationsfenster/-baum genannt oder machmal Navigatorfester/-baum.

11.1. Einleitung

Abbildung 11.1, „ Überblick über den Explorer “ zeigt das ArgoUML-Fenster mit dem hervorgehobenen Explorer.

Abbildung 11.1. Überblick über den Explorer



Der Explorer erlaubt es dem Anwender, die Struktur des Modelles in einer Anzahl von vordefinierten Perspektiven zu betrachten. Er erlaubt es den Anwendern auch, ihre eigenen Perspektiven für das benutzerspezifische Erkunden des Modelles zu definieren.

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 Modelles 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.

Ein Taste 1-Klick über dem Namen eines Diagramm-Modellelementes veranlasst, dass das Diagramm markiert und im Editierfenster angezeigt wird. Zusätzlich werden seine Details im Detailfenster dargestellt.

Ein Taste 1-Klick im Hauptbereich des Explorers über dem Namen eines Modellelementes, welches kein Diagramm ist, wird markiert und seine Details im Detailfenster angezeigt. Ist das Modellelement Teil eines aktuell im Editierfenster dargestellten Diagrammes, wird das Modellelement dort hervorgehoben.

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 enspricht 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 Modellelemente im ArgoUML-Modell können für die Darstellung in der Baumansicht für eine Anzahl von Perspektiven konfiguriert werden. Zu diesem Zweck erlaubt ein Pull-down-Menü im oberen Bereich, die Auswahl der Explorerperspektive.

Nachfolgend gibt es ein Pull-down-Menü, um die Reihenfolge der Elemente innerhalb der Hierarchie auszuwählen. Die zwei Möglichkeiten sind: "Nach Typ und Name" und "Nach Name". Der vorherige gruppiert alle Elemente nach ihrem Typ und sortiert diese alfabetisch nach ihrem Namen. Der letztere sortiert einfach nur nach dem Namen.

Die folgenden Explorerperspektiven können in dem obigen Pull-down- Menü ausgewählt werden:

- `Paketorientiert` (der Standard). Die Hierarchie ist anhand der Pakethierarchie organisiert. Die oberste Ebene zeigt das Modell. Darunter befinden sich alle auf oberster Ebene befindlichen Pakete des Modelles und alle Modellelemente, die sich direkt im Namensraum des Modelles befinden.

  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 verknüpften 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.

- `Klassenassoziationsorientiert`. 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 Modellelemente 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.

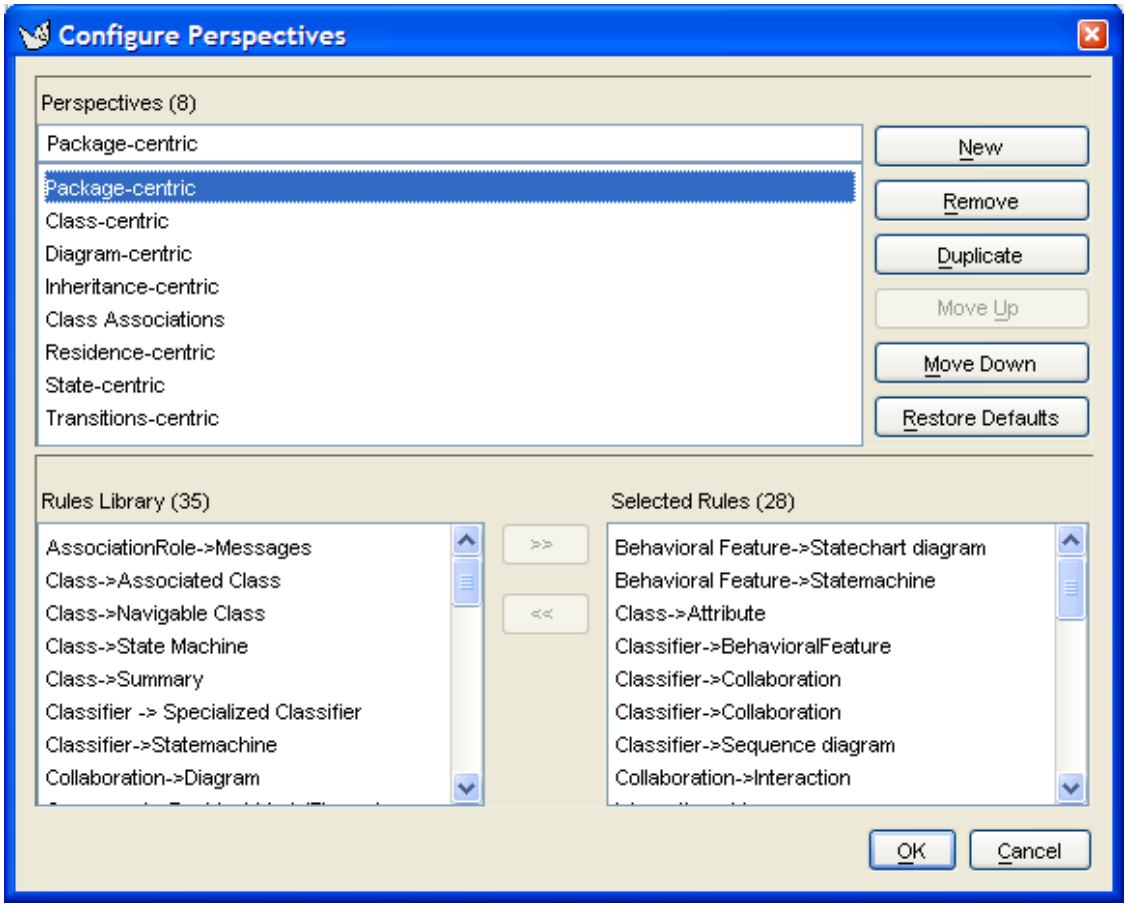
11.5. Perspektiven konfigurieren

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.

Die untere Hälfte des Dialoges enthält zwei Listen. Die eine links, mit `Regelbibliothek` beschriftet, enthält eine Liste der verfügbaren Regeln, die für das Erstellen einer Perspektive verwendet werden können. Die eine rechts, mit `Ausgewählte Regeln` beschriftet, enthält die, für die in der obigen Liste der Perspektiven markierte Perspektive aktuell ausgewählten Regeln. In beiden Listen können Sie nur eine Regel gleichzeitig auswählen.

Als Trennung der beiden Bereiche in der unteren Hälfte des Dialoges befinden sich Schaltflächen, die mit >> und << beschriftet sind. Die erste davon überträgt die in der Biblothek 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.

Ganz unten rechts befindet sich eine Schaltfläche mit der Beschriftung `OK`, die verwendet wird, wenn alle Änderungen vervollständigt sind. Ein Taste 1-Klick auf diese Schaltfläche wird den Dialog schliessen. Die Änderungen werden in der Datei `argo.user.properties` gespeichert, wenn Sie ArgoUML beenden.

Dann gibt es noch die Schaltfläche `Abbrechen`, die alle im Dialog durchgeführten Änderungen aufhebt. Das Betätigen des Dialog-Schliessen-Symboles (normalerweise in der oberen rechten Ecke) hat den gleichen Effekt, wie das Betätigen der Schaltfläche Abbrechen.

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

Dieser Eintrag im Popup-Menü öffnet eines Auswahl von Untermenüs. Eines für jedes 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

Dieser Eintrag im Popup-Menü erstellt eine grafische Datei im Standard-Grafikformat und bringt es in die Zwischenablage Ihres PC. Die Grafik kann unmittelbar darauf in z.B. ein Anforderungsdokument in OpenOffice.Org kopiert werden.

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 löscht 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 löschen, die in dem Diagramm angezeigt werden. Um die Unterschiede zu illustrieren, betrachten Sie die folgenden Beispiele:

- Das Löschen eines Klassendiagrammes löscht nicht jedes Modellelement, das darin angezeigt wird. Alle Modellelemente die im Diagramm angezeigt werden, bleiben im

Modell erhalten. Dies ist so, weil ein Klassendiagramm nicht auf jedes Modellelement entsprechend dem UML-Standard V1.4 " abgebildet" wird.

- Das Löschen eines Zustandsdiagrammes löscht auch den Zustandsautomaten, den es repräsentiert und behandelt auch alle Modellelemente des Zustandsautomaten auf die gleiche Weise. Dies ist so, weil ein Zustandsdiagramm entsprechend dem UML-Standard V1.4 nicht auf einen Zustandsautomaten "abgebildet" wird.

11.6.6. Einstellen des Quellpfades... (Noch zu beschreiben)

Dieser Eintrag im Popup-Menü ...

11.6.7. Paket hinzufügen

Dieser Eintrag im Popup-Menü ist verfügbar, wannimmer ein Modellelement markiert ist, das ein Paket enthalten darf, z.B. ein Paket. Nach dem Aktivieren dieses Menüs wird das Modellelement ein neues Paket enthalten.

11.6.8. Neuer Stereotyp

Dieser Eintrag im Popup-Menü ist verfügbar ... (Noch zu beschreiben)

11.6.9. Alle Klassen im Namensraum hinzufügen

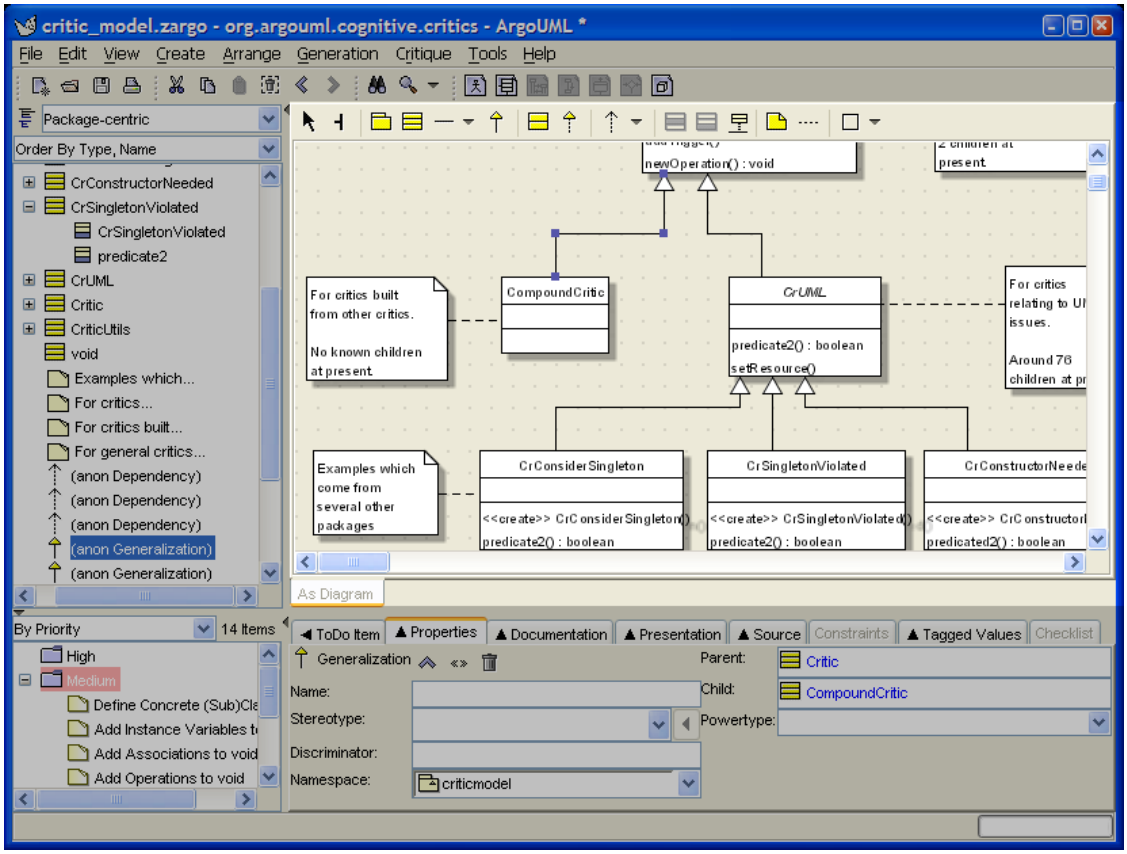
Dieser Eintrag im Popup-Menü ist nur bei Klassendiagrammen verfügbar. Das Aktivieren dieses Menüeintrages wird alle Klassen des aktuellen Namensraumes zum Diagramm hinzufügen. Sie werden in der oberen linken Ecke angeordnet; offensichtlich eine günstige Gelegenheit, die im Menü befindliche „Anordnen->Layout“-Funktion 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



Darin werden alle Diagramme gezeichnet. In früheren Versionen von ArgoUML firmierte dieses Fenster unter verschiedenen Namen. Sie werden die Begriffe „Zeichenfenster“, „ Diagrammfenster“ oder „Multi-Editorfenster“ in anderen, noch nicht aktualisierten Dokumentationen vorfinden.

Das Fenster hat oben eine Werkzeugleiste 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.

Viele Modellelemente (z.B. Akteur, Klasse) zeigen spezielle Verhaltensweisen, wenn sie markiert sind und die Maus darüber fährt. Diese werden „Auswahl-Aktionsschaltflächen“ genannt, siehe Abschnitt 12.6, „ Auswahl-Aktionsschaltflächen “. Sie erscheinen an den Seiten, oben und unten und geben einen Beziehungstyp an. Klicken auf eine Auswahl-Aktionsschaltfläche erstellt eine neues Modellelement mit einer Beziehung des angezeigten Typs. Wenn die Umschalttaste gedrückt ist, wenn die Maus über ein markiertes Modellelement fährt, werden manchmal unterschiedliche Griffe angezeigt. Sie stehen für unterschiedliche Beziehungstypen.

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

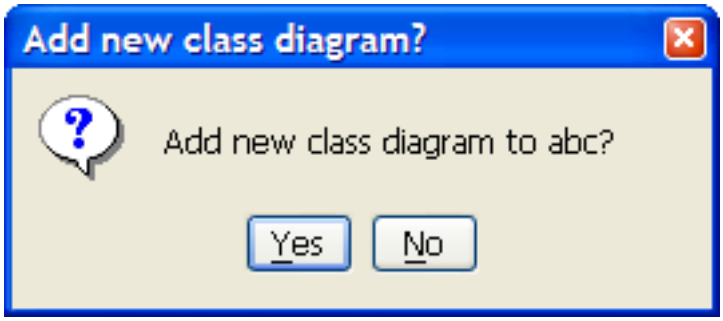
Wenn dies in der Werkzeugleiste mit einem Werkzeug zum Hinzufügen eines Modellelementes verwendet wird, wird das markierte Modellelement dem Zeichenbereich mehrmals hinzugefügt. Einmal für jeden weiteren Tastenklick, bis das Werkzeug erneut markiert oder ein anderes Werkzeug ausgewählt wird.

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 Modellelementen wird die Taste 1-Bewegung das Modellelement an eine neue Position bewegen.

Grafische, markierte Modellelemente zeigen Griffe an den Ecken oder Enden an. Diese können für Größenänderungen verwendet werden.

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.

Wo das Modellelement eine Form von Verbinder zwischen anderen Elementen ist, verursacht die Taste 1-Bewegung neben den Griffen die Erstellung eines neuen Griffes, der es dem Verbinder erlaubt, sich mit diesem Punkt zu verbinden. Dies funktioniert nur, wenn die verbindende Linie nicht gerade rechtwinklig ist. Solche neuen Griffe können durch das Bewegen auf das Ende des Verbinders entfernt werden.

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 Editierfenster verwendet wird, wird ein kontextabhäniges Popup-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 Popup-Menüs siehe Abschnitt 12.10, „ Pop-Up Menü's “.

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 Symbolleiste im oberen Bereich des Editierfensters enthält die Hauptfunktionen des Fensters. Das Standardsymbol ist das `Auswahl`-Werkzeug (). Generell wählt ein Taste 1-Klick auf irgendein Symbol dieses für die einmalige Verwendung aus, bevor es zum Standardsymbol zurückkehrt. Und ein Taste 1-Doppelklick wählt ein Symbol für die wiederholte Nutzung aus.

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.

- *Zeichen-Symbole.* Werden zum Hinzufügen grafischer Objekte in Diagrammen verwendet.
- *Diagrammspezifische Symbole.* Werden zum Hinzufügen von diagrammspezifischen UML-Modellelementen in das Diagramm verwendet.

Einige Werkzeuge, die normalerweise nicht so oft verwendet werden, sind in ein Pulldown-Menü aufgenommen worden, um mehr Platz in der Werkzeugleiste zu erhalten. Siehe z.B. Abbildung 12.3, „ Die Symbolauswahl Zeichen. “. Das Drücken des Symboles auf der rechten Seite des Werkzeuges öffnet das Pulldown-Menü. Diese Pulldown-Werkzeuge erinnern sich dauerhaft, welches Werkzeug zuletzt verwendet wurde. Das heißt, wenn ArgoUML startet zeigen sie das zuletzt aktivierte Werkzeug an.

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 Griffes 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 Modellelementelayout sind über das Menü `Anordnen` verfügbar (siehe Abschnitt 10.7, „ Das Menü Anordnen “).

12.4.2. Kommentierungs-Symbole

Das Kommentierungs-Symbol `Kommentar` () wird dazu verwendt, 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

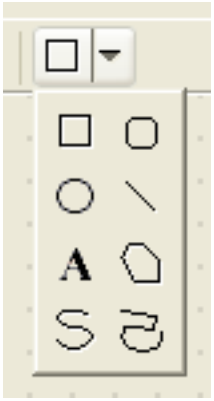
Dies sind eine Reihe von Symbolen, die grafische Zusätze zu Diagrammen liefern. Obwohl sie keine UML-Modellelemente sind, erlaubt der UML-Standard solche Ausschmückungen, um die Lesbarkeit der Diagramme zu verbessern.

Tipp

Diese Zeichen-Symbole stellen einen nützlichen Weg dar, einige der in der aktuellen Release von ArgoUML fehlenden UML-Eigenschaften (wie z.B. generelle Zweck-Hinweise) teilweise zu unterstützen.

Acht Symbole sind vorhanden, alle in einem Pulldown-Menü gruppiert. Siehe Abbildung 12.3, „ Die Symbolauswahl Zeichen. “. Ein Taste 1-Klick auf das Diagramm wird eine Instanz des grafischen Elementes in der gleichen Größe wie das letzte plazieren. Die Größe kann durch eine Taste 1-Bewegung während des Plazierens eingestellt werden. Eine Seite oder Ende des Elementes wird durch Taste 1-gedrückt die andere Seite oder Ende durch Taste 1- loslassen beeinflusst. Nachdem sie im Diagramm plaziert wurden, können die grafischen Elemente mit dem Werkzeug `Auswählen` und der Taste 1 verschoben und in der Größe durch eine Taste 1-Bewegung über die Griffe nach dem markieren verändert werden.

Abbildung 12.3. Die Symbolauswahl Zeichen.



-  `Rechteck`. Erzeugt ein Rechteck.
-  `Abgerundetes Rechteck`. Erzeugt ein Rechteck mit abgerundeten Ecken. Der Grad der Rundung kann nicht eingestellt werden.
-

 `Kreis`. Erzeugt einen Kreis.

-  `Gerade`. Erzeugt eine Gerade.

-  `Text`. Erzeugt ein Textfeld. Der Text wird durch markieren des Feldes und schreiben eingegeben. Der Text wird nach der Eingabe horizontal zentriert. Das Feld passt sich an die Größe des Textes an. Es kann jedoch in der Größe durch ziehen an den Ecken verändert werden.

-  `Polygon`. Erzeugt ein Polygon. Die Punkte des Polygons werden durch Taste 1-Klicks markiert. Das Polygon wird durch einen Taste 1-Doppelklick abgeschlossen (dies verbindet den letzten mit dem ersten Punkt).

-  `Segmentierte Linie`. Erzeugt eine segmentierte Linie. Die Punkte der segmentierten Linie werden mit Taste 1-Klicks markiert und der letzte Punkt mit einem Taste 1-Doppelklick ausgewählt.

-  `Punktierte Linie`. Erzeugt eine punktierte Linie. Die Punkte werden durch eine Taste 1-Bewegung erzeugt.

12.4.4. Anwendungsfalldiagrammspezifische Symbole

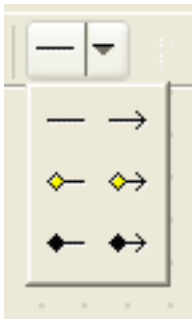
Verschiedene Symbole sind speziell für UML-Modellelemente in Anwendungsfalldiagrammen vorhanden. Die detaillierten Eigenschaften dieser Modellelemente sind im Abschnitt über Modellelemente von Anwendungsfalldiagrammen beschrieben (siehe Kapitel 17, *Referenz der Modellelemente für Anwendungsfalldiagramme*).

-  `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.

-  `Anwendungsfall`. Fügt dem Diagramm einen Anwendungsfall hinzu. Wenn sich die Maus über einem markierten Anwendungsfall befindet, werden aus Gründen des Komforts links und rechts zwei Griffe angezeigt, die gezogen werden können, um Assoziationsbeziehungen herzustellen und zwei Griffe oben und unten, die gezogen werden können, um Generalisierungen und Spezialisierungen herzustellen.

- `Assoziation`. Fügt zwischen zwei Modellelementen eine Assoziation mit Hilfe einer Taste 1- Bewegung ein (vom ersten bis zum zweiten Modellelement). Es werden hier 6 Assoziationstypen angeboten, siehe Abbildung 12.4, „ Die Symbolauswahl Assoziation. “: `Assoziation`, `Aggregation` und `Komposition`, und all diese drei können `bidirektional` oder `unidirektional` sein.

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).
-  `Erweiterungspunkt hinzufügen`. Einen Erweiterungspunkt zu einem markierten Anwendungsfall hinzufügen. Dem Erweiterungspunkt wird der Standardname `neuerEP` gegeben und der Ort `ort` . Wo ein Erweiterungspunktbereich angezeigt wird, kann der Erweiterungspunkt mit Hilfe eines Taste 1- Doppelklicks und der Tastatur, oder durch markieren mit einem Taste 1-Klick (nachdem der Anwendungsfall markiert wurde) und Verwendung des Registers Eigenschaften editiert werden. Ansonsten kann es über sein Register Eigenschaften editiert werden, welches über das Register Eigenschaften des besitzenden Anwendungsfalles ausgewählt wird.

Anmerkung

Dieses Werkzeug ist deaktiviert, es sei denn, ein Anwendungsfall ist markiert.

12.4.5. Klassendiagrammspezifische Symbole

Einige Symbole sind speziell für UML-Modellelemente in Klassendiagrammen gedacht. Die detaillierten Eigenschaften dieser Modellelemente sind im Abschnitt über Klassendiagramm-Modellelemente beschrieben (siehe Kapitel 18, *Modellelement-Referenz Klassendiagramm*).

-  `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).
- `Datentyp`. Fügt in das Diagramm einen Datentyp ein. Wenn sich eine Maus über einem markierten Datentyp befindet werden oben und unten Griffe angezeigt, die angeklickt oder gezogen werden können, um eine Generalisierungs-Beziehung herzustellen (das Ziel kann ein anderer Datentyp sein). Es sind hier 2 andere Elemente verfügbar, `Aufzählung` und «» `Stereotyp`. Diese haben identische Griffe, außer dem oben an einem Stereotyp befindlichen: wenn er angeklickt wird, erstellt er eine Metaklasse, die über eine mit «stereotype» markierte Abhängigkeit verknüpft ist. Dies erleichtert die Erstellung von "Stereotyp-Deklarations"-Diagrammen - näheres entnehmen Sie bitte der Literatur.

12.4.6. Sequenzdiagrammspezifische Symbole

Sieben Symbole sind speziell für UML-Modellelemente in Sequenzdiagrammen vorhanden. Die detaillierten Eigenschaften dieser Modellelemente sind im Abschnitt über Sequenzdiagramm-Modellelemente beschrieben (siehe Kapitel 19, *Modellelement-Referenz Sequenzdiagramm*).

- `Klassifizierte Rolle`. Fügt dem Diagramm eine klassifizierte Rolle hinzu.
- `Nachricht mit Aufruf einer Aktion`. Fügt eine Aufruf-Nachricht zwischen zwei klassifizierten Rollen mit Hilfe einer Taste 1-Bewegung ein (von der verursachenden klassifizierten Rolle zur empfangenden klassifizierten Rolle).
- `Nachricht mit Antwort-Aktion`. Fügt eine Antwort-Nachricht mit Hilfe einer Taste 1-Bewegung zwischen zwei klassifizierten Rollen ein (von der verursachenden klassifizierten Rolle zur empfangenden klassifizierten Rolle).
- `Nachricht mit einer Erzeugen-Aktion`. Fügt eine Erzeugungs-Nachricht mit Hilfe einer Taste 1-Bewegung zwischen zwei klassifizierten Rollen ein (von der verursachenden klassifizierten Rolle zur empfangenden klassifizierten Rolle).
- `Nachricht mit Destruktions-Aktion`. Fügt eine Destruktions-Nachricht mit Hilfe einer Taste 1-Bewegung zwischen zwei klassifizierten Rollen ein (von der verursachenden klassifizierten Rolle zur empfangenden klassifizierten Rolle).
- `Vertikalen Zwischenraum in Diagramm einfügen`. Fügen Sie vertikalen

Zwischenraum in ein Diagramm ein, indem Sie alle nachfolgenden Nachrichten nach unten verschieben. Klicken Sie mit der Maus auf den Punkt, wo Sie den Zwischenraum einfügen wollen und ziehen Sie auf dem Bildschirm vertikal um den Abstand nach unten, der der Höhe des gewünschten hinzuzufügenden Zwischenraumes entspricht.

-  `Vertikalen Zwischenraum im Diagramm entfernen` . Entfernt den vertikalen Zwischenraum im Diagramm und bewegt alle nachfolgenden Elemente vertikal nach oben. Klicken und ziehen Sie die Maus vertikal über den Zwischenraum, den Sie löschen wollen.

12.4.7. Kollaborationsdiagrammspezifische Symbole

Drei Symbole sind speziell für UML-Modellelemente in Kollaborationsdiagrammen vorhanden. Die detaillierten Eigenschaften dieser Modellelemente sind im Abschnitt über Kollaborationsdiagramm-Modellelemente beschrieben (siehe Kapitel 21, *Modellelement-Referenz Kollaborationsdiagramm*).

-  `Klassifizierte Rolle`. Fügt dem Diagramm eine klassifizierte Rolle hinzu.
- `Assoziation`. Fügt eine Assoziation mit Hilfe einer Taste 1-Bewegung zwischen zwei klassifizierten Rollen ein (von der verursachenden Rolle zur empfangenden Rolle). Es werden hier 6 Typen von Assoziationen angeboten, siehe Abbildung 12.4, „ Die Symbolauswahl Assoziation. “: `Assoziation`, `Aggregation` und `Komposition`, und all diese drei können `bidirektional` oderr `unidirektional` sein.
-  `Generalisierung`. Fügt eine Generalisierung zwischen zwei Modellelementen ein, die mit der Taste 1 markiert wurden (von Kind zum Vater).
-  `Abhängigkeit`. Fügt eine Abhängigkeit zwischen zwei Modellelementen ein, die mit einer Taste 1- Bewegung markiert wurden (vom abhängigen Modellelement).
-  `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

Elf Symbole speziell für UML-Modellelemente in Zustandsdiagrammen sind vorhanden. Die detaillierten Eigenschaften dieser Modellelemente sind im Abschnitt Zustandsdiagramm-Modellelemente beschrieben (siehe Kapitel 20, *Modellelement-Referenz Zustandsdiagramm*).

-  `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 ein Hinweis 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 ein Hinweis 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 ein Hinweis 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 ein Hinweis 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 ein Hinweis 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.

-  `Aufruf-Ereignis`. Fügt einer Transition ein Aufruf-Ereignis als Trigger hinzu. Es werden hier 4 Ereignisarten angeboten: `Aufruf-Ereignis`, `Änderungs-Ereignis`, `Signal-Ereignis` und `Zeit-Ereignis`.

-  `Wächter`. Fügt einer Transition einen Wächter hinzu.

-  `Aufruf-Aktion`. Fügt einer Transition (z.B. einem Effekt) eine Aufruf-Aktion hinzu. Es werden hier 7 Arten von Aktionen angeboten: `Aufruf-Aktion`, `Erzeugungs-Aktion`, `Zerstören-Aktion`, `Antwort-Aktion`, `Sende-Aktion`, `Beenden-Aktion`, `Uninterpretierte Aktion` und `Aktionsfolge`.

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 bedeutunglos und es wird einer der Hinweise 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 ein ArgoUML-Hinweis 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 ein ArgoUML- Hinweis 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 ein ArgoUML- Hinweis bei jeder Vereinigung ausgelöst, die dieser Regel nicht entspricht.

-  `Aufrufknoten`. Sie fügen dem Diagramm einen Aufrufknoten hinzu. Ein Aufrufknoten ist eine Aktion, die eine einzelne Operation aufruft. Folglich wird der Name der aufzurufenden Operation, zusammen mit dem in Klammern stehenden Namen des Klassifizierers, der die Operation ausführt, in das Symbol geschrieben.

-  `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

Zehn Symbole sind speziell für UML-Modellelemente in Verteilungsdiagrammen vorhanden. Die detaillierten Eigenschaften dieser Modellelemente sind im Abschnitt über Modellelemente in Verteilungsdiagrammen beschrieben (siehe Kapitel 23, *Modellelement-Referenz Verteilungsdiagramm*).

Anmerkung

Erinnern Sie sich daran, dass ArgoUML's Verteilungsdiagramme auch als Komponentendiagramme verwendet werden.

-  `Knoten`. Sie fügen dem Diagramm einen Knoten hinzu. Befindet sich die Maus über einem markierten Knoten, zeigt dieser vier Griffe, jeweils einer links, rechts, oben und unten. Diese Griffe können auf andere Objekte gezogen werden, um Assoziationen einzurichten.

-  `Knoteninstanz`. Sie fügen dem Diagramm eine Knoteninstanz hinzu. Befindet sich die Maus über einer markierten Knoteninstanz, zeigt diese vier Griffe, jeweils einer links, rechts, oben und unten. Diese Griffe können auf andere Objekte gezogen werden, um Verknüpfungen einzurichten.

-  `Komponente`. Sie fügen dem Diagramm eine Komponente hinzu. Befindet sich die Maus über einer markierten Komponente, zeigt diese vier Griffe, jeweils einer links, rechts, oben und unten. Diese Griffe können auf andere Objekte gezogen werden, um Abhänigkeiten einzurichten.

-  `Komponenteninstanz`. Sie fügen dem Diagramm eine Komponenteninstanz hinzu. Befindet sich die Maus über einer markierten Komponenteninstanz, zeigt diese vier Griffe, jeweils einer links, rechts, oben und unten. Diese Griffe können auf andere Objekte gezogen werden, um Abhängigkeiten einzurichten.

-  `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).

- `Assoziation`. Sie fügen eine Assoziation zwischen zwei Modellelementen (Knoten, Komponente, Klasse oder Schnittstelle) ein (vom ersten Modellelement zum zweiten Modellelement) Es gibt 6 Arten von Assoziationen, die hier angeboten werden. Siehe Abbildung 12.4, „ Die Symbolauswahl Assoziation. “: `Assoziation`, `Aggregation` und `Komposition`, und alle diese drei können `bidirektional` oder `unidirektional` sein.

  Achtung

  Die Randbedingung, dass Assoziationen zwischen Klassen und Schnittstellen *von* der Schnittstelle nicht navigierbar sein dürfen, gilt auch in Verteilungsdiagrammen.

-  `Objekt`. Sie fügen dem Diagramm ein Objekt hinzu. Befindet sich die Maus über einem markierten Objekt, zeigt dieses vier Griffe, jeweils einer links, rechts, oben und unten. Diese Griffe können auf andere Objekte gezogen werden, um Verknüpfungen einzurichten.

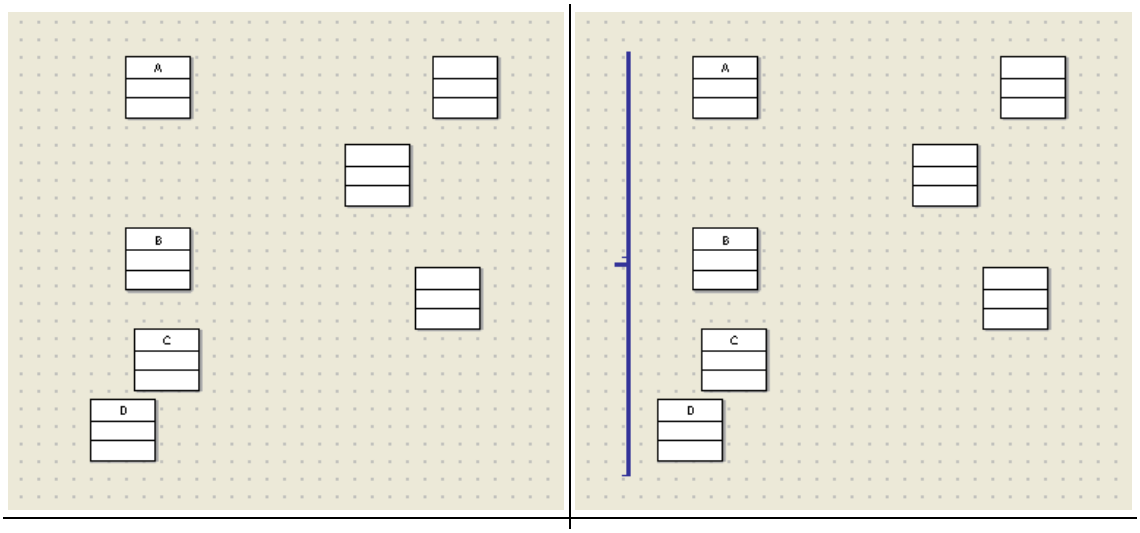
-  `Verknüpfung`. Sie fügen eine Verknüpfung zwischen zwei Modellelementen (Knoteninstanz, Komponenteninstanz oder Objekt) ein.

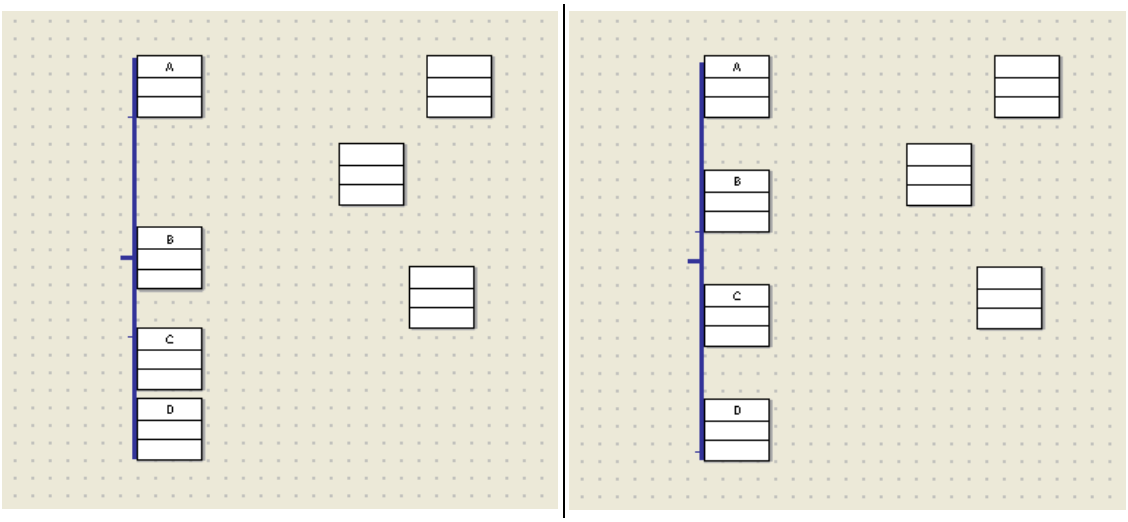
12.5. Der Besen

ArgoUML's Besen-Ausrichtungswerkzeug ist spezialisiert darauf, die Bedürfnisse von Designern hinsichtlich der Ausrichtung von Modellelementen in UML-Diagrammen zu unterstützen. Häufig richten Designer Objekte beim Erzeugen grob aus oder tun dies über einfache Bewegungskommandos. Der Besen ist ein einfacher Weg, grob ausgerichtete Objekte präzise auszurichten. Darüber hinaus sind die Verteilungsoptionen des Besen's auf die Bedürfnisse von Designern zugeschnitten: zusammengehörende Objekte haben einen gleich grossen Zwischenraum, bündelt Objekte, um Diagrammplatz einzusparen, und schafft Platz für neue Objekte. Der Besen macht es auch einfach, von der horizontalen in die vertikale Ausrichtung oder von linksbündiger zu rechtsbündiger Ausrichtung zu wechseln.

Das T-förmige Symbol in ArgoUML's Diagrammwerkzeugleiste ruft das Besen-Ausrichtungswerkzeug auf. Wird die Maustaste 1 im Besenmodus gedrückt, wird die erste Mausbewegung des Designers den Besen in einer der vier Richtungen: Norden, Süden, Osten oder Westen ausrichten. Danach verursachen Mausbewegungen, dass der Besen sich in die gewählte Richtung vorwärts, rückwärts oder seitwärts bewegt. Wie ein Schieber in der realen Welt, verschiebt das Besen- Werkzeug Diagrammelemente, die mit ihm in Kontakt kommen. Dies hat zur Folge, dass die Objekte entsprechend der Besenvorderseite ausgerichtet werden und dies unmittelbar visualisiert wird (siehe nachfolgendes Bild). Im Gegensatz zu einem Besen in der realen Welt, erlaubt die Rückwärtsbewegung, dass Diagrammelemente in ihre ursprüngliche Position zurückkehren können. Das Vergrößern des Besens macht es möglich, Objekte, die nicht nahe beieinander sind, auszurichten. Wird die Maustaste losgelassen, verschwindet der Besen , die Objekte bleiben markiert, um diese weiterhin verändern zu können.

Abbildung 12.5. Der Besen.





Wenn der Designer die *Leertaste* während der Nutzung des Besens drückt, werden die Objekte an der Vorderseite des Besens gleichmäßig verteilt (z.B. gleichmäßiger Zwischenraum). ArgoUML's Besen unterstützt drei Verteilungsmodi: Objekte können gleichmäßig auf den Raum, den sie nutzen, verteilt werden, Objekte können gepackt werden, nur mit einem kleinen Zwischenraum dazwischen, oder Objekte können gleichmäßig über die gesamte Länge der Besenvorderseite verteilt werden. Wiederholtes Drücken der Leertaste wechselt zwischen diesen drei Verteilungsmodi und gibt eine Kurzinformation aus, welche Operation gerade ausgeführt wird: Gleichmäßiger Zwischenraum, komprimieren, spreizen und Ursprung.

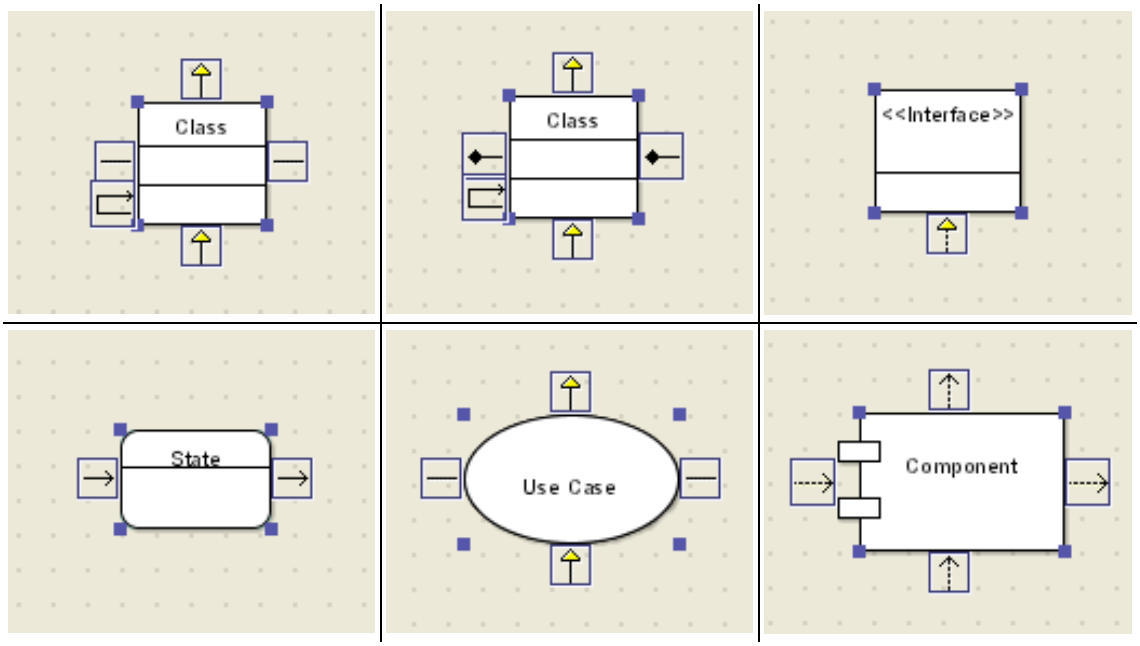
Wenn der Designer die Taste *Enter* während der Nutzung des Besens drückt, wird der Besen rot (anstelle des normalen blau) und es werden keine Objekte während der Vorwärtsbewegung des Besens mitgenommen. Dies wirkt wie das Anheben des Besens. Durch erneutes Drücken der Taste Enter kehrt man in den normalen Modus zurück.

Das Betätigen der Taste *Tab* arbeitet genauso wie die Taste Enter.

12.6. Auswahl-Aktionsschaltflächen

Wenn der Anwender ein Modellelement in einem UML-Diagramm markiert, dann werden mehrere Griffe dargestellt, um anzuzeigen, dass es markiert ist und um die Anwenderschnittstelle mit der Fähigkeit zu versehen, die Größe des Knotens zu verändern. ArgoUML zeigt auch einige „Auswahl-Aktionsschaltflächen“ entlang des markierten Modellelementes an. In den nachfolgenden Bildern sehen Sie einige Beispiele von Griffen und „Auswahl- Aktionsschaltflächen“. Die beiden Bilder für eine Klasse unterscheiden sich, weil beim Erzeugen der zweiten Klasse die Umschalt-Taste gedrückt wurde.

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 hinzufügen zu können; eine bei 6 Uhr, um eine Subklasse hinzuzufügen und Schaltflächen bei der 3 Uhr- und 9 Uhr-Position, um Assoziationen hinzufügen 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.

Auswahl-Aktionsschaltflächen sind transparent. Sie haben einen sichtbaren rechteckigen Rahmen und enthalten ein Symbol, welches dem Symbol des entsprechenden Designelementes in der Standard-Symbolleiste entspricht. Diese Symbole sind jedoch ungefüllte Zeilen-Symbole mit vielen transparenten Pixeln. Dies erlaubt es, daß Auswahl-Aktionsschaltflächen die Zeichenfläche überlappen, ohne das Diagramm zu verdecken. Aus diesem Grund werden die Schaltflächen nur angezeigt, wenn sich die Maus über dem Symbol des markierten Modellelementes befindet; wenn ein Teil des Diagrammes verdeckt ist, kann die Maus einfach fortbewegt werden, um eine klarere Sicht auf das Diagramm zu erhalten.

12.7. Erläuterungen (Clarifiers)

Eine Schlüsseleigenschaft von ArgoUML sind die Hinweise, die parallel zum ArgoUML-Werkzeug arbeiten. Wenn diese ein Problem finden, erzeugen sie ein Zu-Bearbeiten-Element und heben das Problem im Editierfenster hervor. Die für das Hervorheben verwendete grafische Technik wird *Erläuterungen* genannt.

- Hinweis-Symbol (). Wird in der oberen linken Ecke eines Modellelementes angezeigt und signalisiert einen Hinweis zu diesem Modellelement. Das Bewegen der Maus über dieses Symbol blendet die Überschrift des Hinweises ein.

- Farbige Wellen-Linie ( ). Wird für Hinweise verwendet, die spezifisch für Subkomponenten von grafischen Modellelementen sind. Zum Beispiel unterstreichen sie die Attribute einer Klasse mit einem Problem.

- Durchgehende farbige Linie ( ). Nicht sichtbar beim normalen editieren, wird aber verwendet, wenn ein Zu-Bearbeiten-Element im Zu-Bearbeiten-Fenster (siehe Kapitel 14, *Der Bereich Zu-Bearbeiten*) durch einen Taste 1 Doppelklick hervorgehoben wird. Die durchgehende Linie kennzeichnet alle von dem Hinweis betroffenen Modellelemente. Zum Beispiel alle Stimuli, die sich außerhalb der Reihenfolge befinden.

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

Unterhalb des Editierfensters befindet sich ein kleiner Reiter, der mit `Als Diagramm` beschriftet ist. Das Konzept ist, daß ein UML-Diagramm auf unterschiedliche Art und Weise dargestellt werden kann. Zum Beispiel als grafisches Diagramm oder als Tabelle. Jede Darstellung hätte seinen eigenen Reiter und kann durch einen Taste 1-Klick auf den Reiter ausgewählt werden.

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. Hinweise

Dieses Untermenü gibt eine Liste aller Hinweise aus, die von diesem Modellelement ausgelöst wurden. Die Auswahl eines der Menüeinträge bewirkt, daß der Eintrag im Zu-Bearbeiten-Fenster hervorgehoben und die ausführliche Erläuterung im Detailfenster des `Zu-Bearbeiten`-Reiters plaziert wird. Eine durchgezogene farbige Linie markiert das entsprechende Element.

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.

-  `Nach hinten`. Das markierte Modellelement wird hinsichtlich der es überlappenden Modellelemente in der Reihenfolgenhierarchie eine Ebene nach unten bewegt.
-  `In den Vordergrund`. Das markierte Modellelement wird hinsichtlich der es überlappenden anderen Modellelemente an die vorderste Stelle bewegt.
-  `In den Hintergrund`. Das markierte Modellelement wird hinsichtlich der es überlappenden anderen Modellelemente an die hinterste Stelle 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.

- `Erweiterungspunkte ausblenden`. Erscheint nur, wenn der Erweiterungspunkt eines Anwendungsfalles eingeblendet ist. Blendet den Erweiterungspunkt aus.
- `Erweiterungspunkte einblenden`. Erscheint nur, wenn der Erweiterungspunkt eines Anwendungsfalles ausgeblendet ist. Blendet den Erweiterungspunkt ein.
- `Alle Bereiche ausblenden`. Erscheint nur, wenn Attribut- und Methoden-Bereiche einer Klasse oder eines Objektes angezeigt werden. Verbirgt beide Bereiche.
- `Alle Bereiche einblenden`. Erscheint nur, wenn Attribut- und Methoden-Bereiche einer Klasse oder eines Objektes ausgeblendet sind. Blendet beide Bereiche ein.
- `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 am 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).
- `Komposition`. Macht dieses Ende zu einer untrennbaren Aggregation (gewöhnlich als „Komposition“ 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 einen Hinweis auslösen), so dass dies nur von theoretischem Interesse ist.

Anmerkung

UML erlaubt keine Navigierbarkeit von einer Schnittstelle zu einer Klasse. ArgoUML verhindert dies nicht.

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

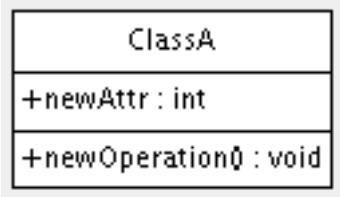
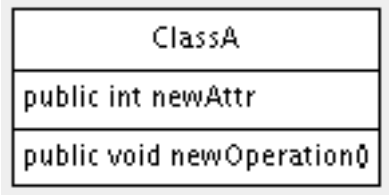


Abbildung 12.8. Eine Klasse in Java-Notation



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.

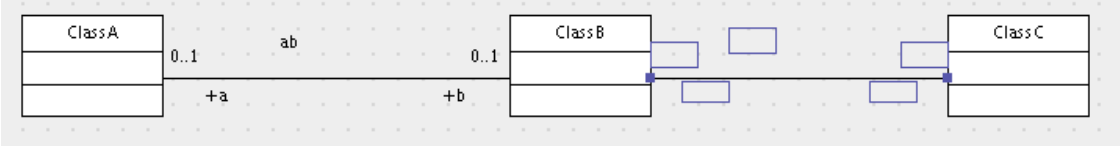
Zusätzlich zeigt die Statuszeile von ArgoUML (z.B. der kleine Bereich unten im ArgoUML-Fenster) einen Hilfetext an, der die Syntax des einzugebenden Textes beschreibt. Die Texteingabe kann durch Drücken der Taste F2, oder bei einzeiligen Felder durch Drücken der Taste Return abgeschlossen werden. Zusätzlich kann das Editieren durch Anklicken eines Bereiches ausserhalb des Diagrammes beendet werden.

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, Gleichzeitigkeit) auf einmal einzugeben:

```
+ «interactive» auftrag(kundenID : Integer, positionen : List) : Boolean
```

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 1 ermöglicht das Editieren.

Die Sichtbarkeit (das +, -, # oder ~) wird im Zusammenhang mit dem Namen des Assoziationsendes angezeigt. Bei unbenannten Assoziationsenden 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.

Das Beispiel-Bild demonstriert dies nicht, weil Stereotypen einer Assoziation im Diagramm angezeigt werden, aber nicht editierbar sind. Und Stereotypen von Assoziationsenden werden zusammen mit dem Namen des Assoziationsendes angezeigt.

12.11.3. Notation Parsen

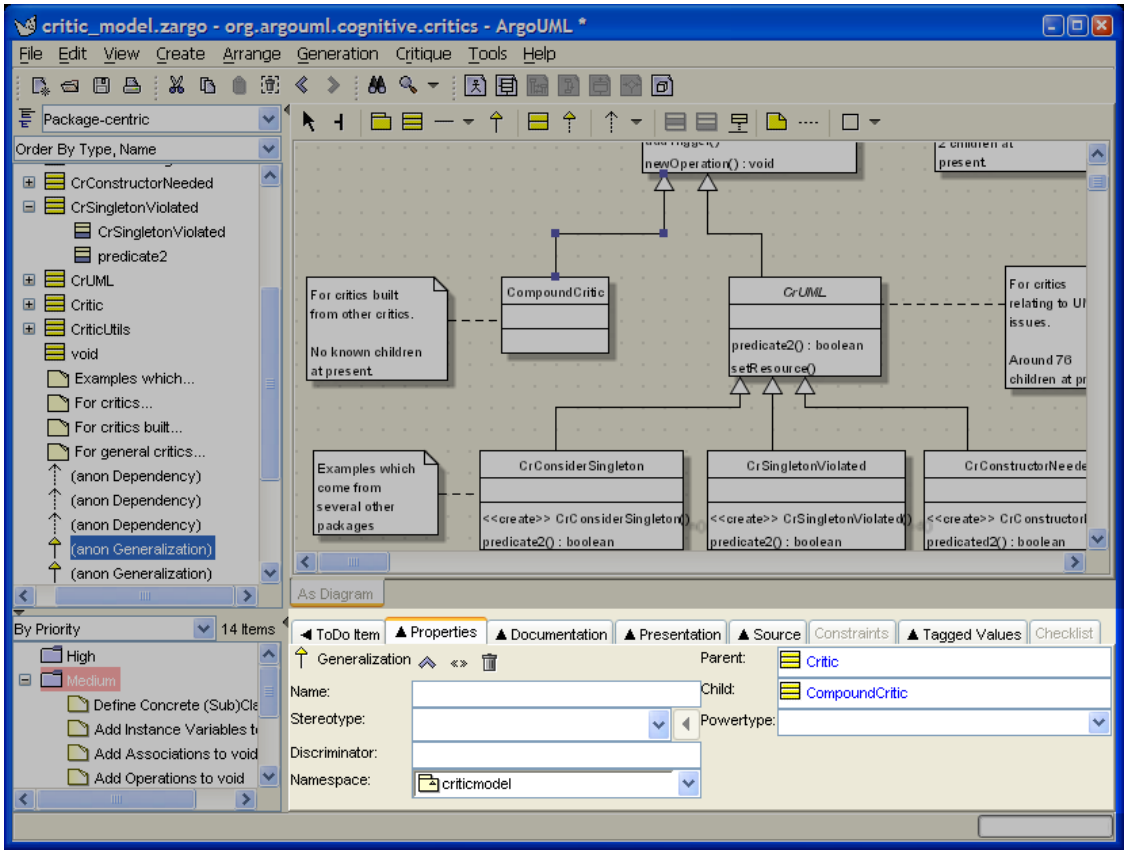
(noch zu beschreiben)

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 Modellelement innerhalb des Systems, werden alle mit ihm verknüpften Daten innerhalb dieses Bereiches angezeigt und eingegeben.

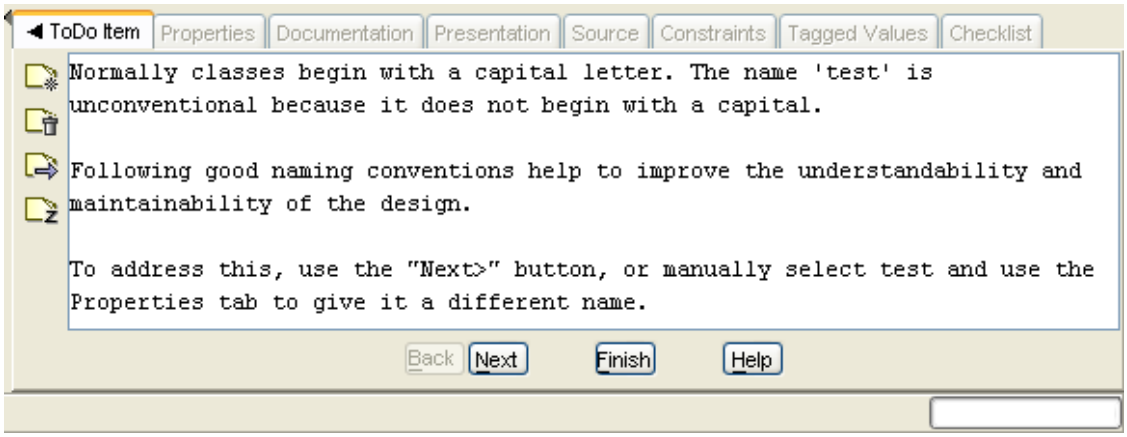
Der Bereich enthält eine Reihe von Registern, die mit einem Taste 1-Klick ausgewählt werden. Der Rumpf des Reiters ist ein Menü von Einträgen, die je nach ausgewähltem Register markiert, ausgewählt oder eingegeben werden, .

Von diesen, ist das Register `Eigenschaften` das komplexeste. Mit unterschiedlichen Darstellungen für jedes Modellelement im System. Die detaillierte Beschreibung des Registers Eigenschaften für jedes Modellelement ist Inhalt separater Kapitel, welche die Modellelemente beschreiben, die in den verschiedenen Diagrammen erscheinen können (siehe Kapitel 16, *Modellreferenz auf höchster Ebene* bis Kapitel 23, *Modellelement-Referenz Verteilungsdiagramm*).

13.2. Das Register "Zu Bearbeiten"

Dieses Register erlaubt die Steuerung über die verschiedenen, vom Anwender erzeugten Zu-Bearbeiten-Einträge oder automatisch von ArgoUML generierten Hinweisen (wird detaillierter im Abschnitt `Hinweise`-Menü diskutiert. Siehe Abschnitt 10.9, „ Das Menü Hinweise “). Abbildung 13.2, „ Beispiel eines `Zu-Bearbeiten-Elementes` im Eigenschafts-Bereich “ zeigt einen typischen Ausschnitt. Das Zu-Bearbeiten-Element wird mit der Taste 1 im Detailbereich markiert (siehe Kapitel 14, *Der Bereich Zu-Bearbeiten*) oder mit Hilfe des kontextsensitiven Popup-Menüs `Hinweise` im Editierfenster.

Abbildung 13.2. Beispiel eines `Zu-Bearbeiten-Elementes` im Eigenschafts-Bereich

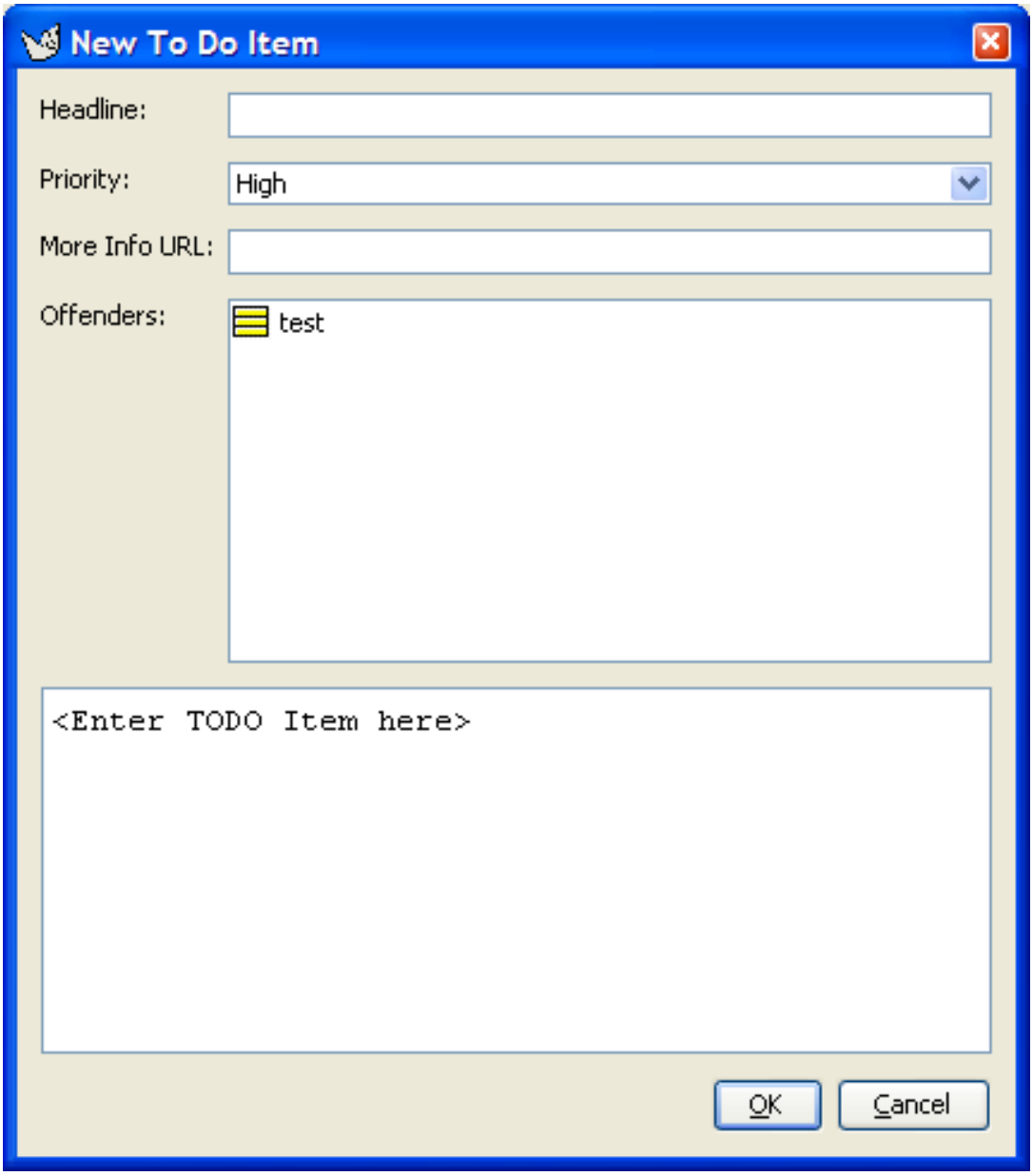


Benutzerspezifische Anpassungen des Verhaltens von Hinweisen ist über das `Hinweise anzeigen...`-Menü möglich (siehe Abschnitt 10.9.4, „ Hinweise anzeigen... “).

Der Rumpf des Registers beschreibt das Problem und beschreibt, wie es gelöst werden kann. Links befinden sich vier Schaltflächen.

- `Neuer Hinweis...` Öffnet ein Dialogfenster (siehe Abbildung 13.3, „ Dialogfenster für `Neuer Hinweis` “), das es Ihnen erlaubt, Ihren eigenen Hinweis 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.

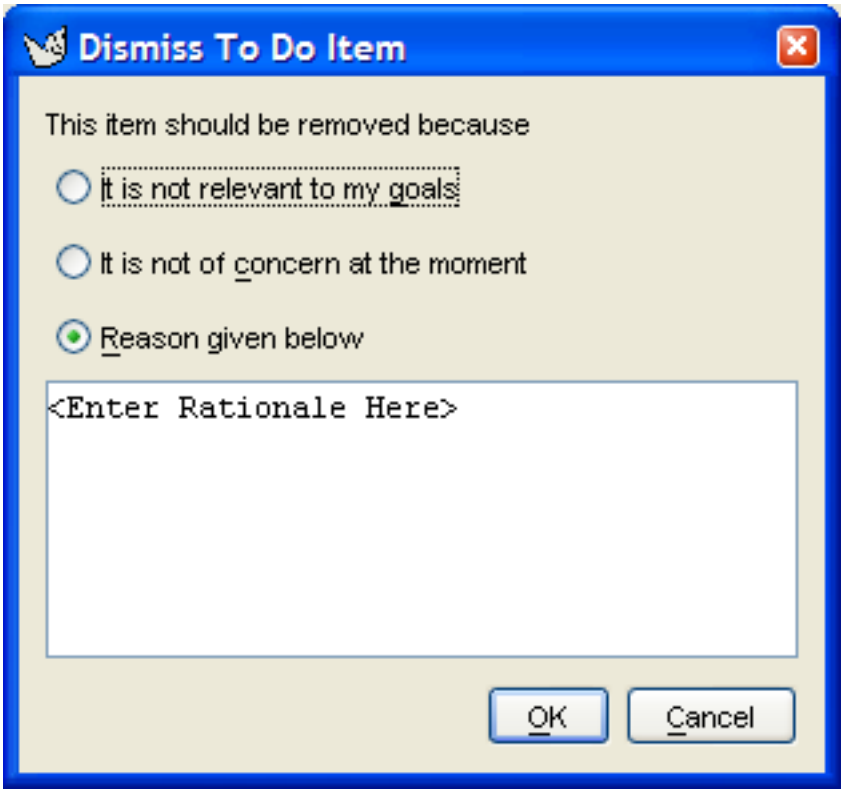
Abbildung 13.3. Dialogfenster für `Neuer Hinweis`



- `Hinweis löschen...` Öffnet einen Dialog, der es dem Anwender ermöglicht, das markierte Zu-Bearbeiten-Element zu löschen (siehe Abbildung 13.4, „ Dialogfenster für `Hinweis löschen` “). 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).

  Dieses Dialogfenster ist für folgende Fälle gedacht: Löschen von Zu-Bearbeiten-Elementen, die manuell erzeugt wurden, verhindern, dass sich ein Hinweis nur auf ein Objekt bezieht und auflösen von ganzen Kategorien von Zu-Bearbeiten-Elementen durch herabstufen von Designbedingungen oder Designzielen.

Abbildung 13.4. Dialogfenster für `Hinweis löschen`



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 löschen wollen (das durch einen Hinweis 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

Die Version V0.20 der ArgoUML-Implementierung ist unvollständig: Die angegebenen Gründe werden beim Speichern des Projektes nicht mitgespeichert. Und es gibt keinen Weg, die aufgelösten Zu-Bearbeiten-Elemente wiederherzustellen. Aus diesem Grund ist es aktuell nicht ratsam eine Begründung einzugeben.

Wenn eine generiertes Zu-Bearbeiten-Element aufgelöst wird, dann gibt es keinen Weg dies ungeschehen zu machen (es sei denn, durch erneutes Erzeugen des Hinweis-auslösenden Objektes).

- `Hinweis deaktivieren` Dies unterdrückt die Aktivität des Hinweises, die das aktuelle

Zu-Bearbeiten-Element generiert. Das Zu-Bearbeiten-Element (und alle anderen von diesem Hinweis generierten Zu-Bearbeiten-Elemente) werden aus dem Bereich Zu-Bearbeiten verschwinden.

Der Hinweis 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. Der Hinweis kann ausdrücklich durch `Hinweise > Hinweise anzeigen...` wieder zum Leben erweckt werden (siehe Abschnitt 10.9.4, „ Hinweise anzeigen... “).

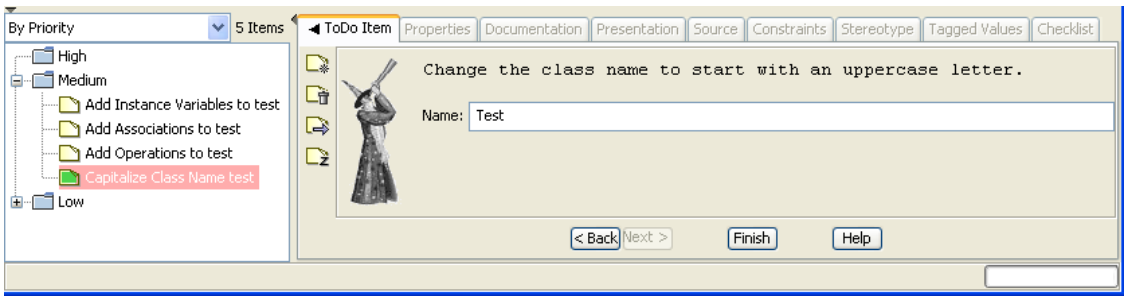
Tipp

Einige Hinweise können während der gesamten Erstellungszeit eines grossen Diagrammes erscheinen. Einige Anwender finden es daher nützlich, diese Hinweise zu deaktivieren bis das Diagramm vollständig ist.

13.2.1. Assistenten

Einige der am häufigsten vorkommenden Hinweise 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.

- `<< Zurück`. Dies bringt Sie in der vorhergehenden Schritt des Assistenten zurück. Die Schaltfläche ist deaktiviert, wenn es sich um den 1. Schritt handelt.
- `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

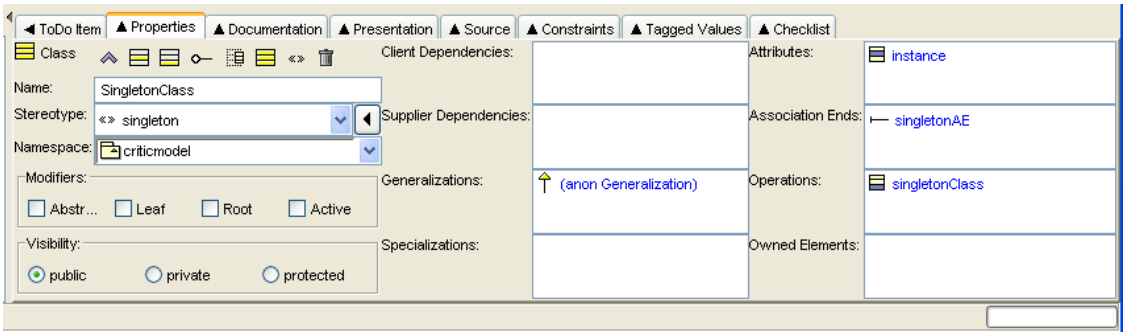
Über dieses Register werden die Eigenschaften eines im Explorer oder im Editierbereich markierten Modellelementes eingestellt. Die Eigenschaften eines Modellelementes können auf folgende Art und Weise angezeigt werden:

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 Modellelement aus dem Modell zu löschen. Die Symbole dazwischen, hängen vom jeweiligen Modellelement ab.

Der Rest des Registers enthält Felder, die in zwei oder drei Spalten angeordnet sind. Jedes Feld ist links davon bezeichnet. Die Felder können Textfelder, Textbereiche, DropDown-Felder, Auswahlfelder oder Markierfelder sein. In den meisten (aber nicht in allen Fällen) Fällen können die Werte geändert werden. Bei Textfeldern ist dies manchmal nur das Eingeben des erforderlichen Wertes.

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 Modellelementes eingegeben werden kann. Die verbleibenden Felder variieren, je nach markiertem Modellelement.

Die detaillierten Eigenschaftstabellen für alle ArgoUML-Modellelemente werden in separaten Kapiteln für jeden Diagrammtyp diskutiert (Anwendungsfalldiagramm (Kapitel 17, *Referenz der Modellelemente für Anwendungsfalldiagramme* , Klassendiagramm (Kapitel 18, *Modellelement-Referenz Klassendiagramm* , Sequenzdiagramm (Kapitel 19, *Modellelement-Referenz Sequenzdiagramm* , Zustandsdiagramm (Kapitel 20, *Modellelement-Referenz Zustandsdiagramm* , Kollaborationsdiagramm (Kapitel 21, *Modellelement-Referenz Kollaborationsdiagramm* , Aktivitätsdiagramm (Kapitel 22, *Modellelement-Referenz Aktivitätsdiagramm* , Verteilungsdiagramm (Kapitel 23, *Modellelement-Referenz Verteilungsdiagramm*). Eigenschaftstabellen für Modellelemente, die sehr häufig in allen Diagrammtypen erscheinen, haben ihr eigenes Kapitel (Kapitel 16, *Modellreferenz auf höchster Ebene*).

Achtung

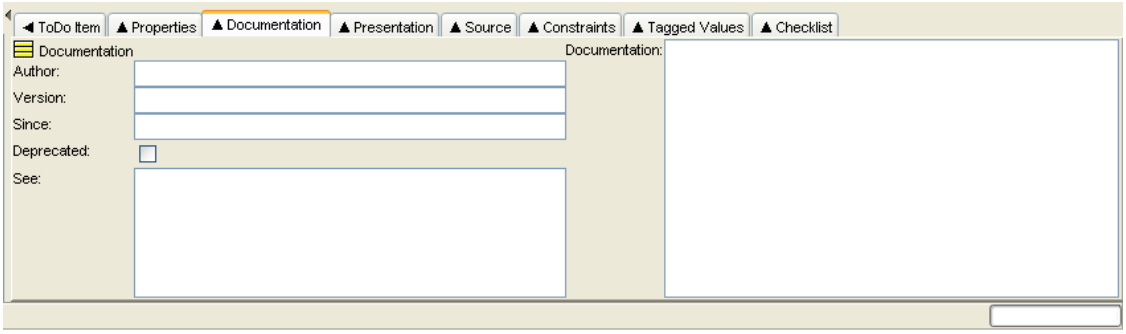
ArgoUML wird immer versuchen, alle Felder auf dem Register Eigenschaften auszugeben. Ist die Größe des Registers Eigenschaften zu klein, wird es unhandlich. Die Lösung ist, entweder das Register Eigenschaften durch vergrößern des Hauptfensters oder durch verschieben der Bereichsteiler nach links und nach oben ebenfalls zu vergrößern

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 Modellelement 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 Modellelement in ArgoUML.

Abbildung 13.7. Ein typisches Register `Dokumentation` im Bereich Details



Wie Sie sehen können, wurden sehr viel mehr Felder dem Dokumentationsfeld hinzugefügt. Die anderen Felder speichern Ihre Informationen gewöhnlich unter Eigenschaftswerte: `Autor`, `Version`, `Seit`, `Veraltet`, `Siehe`.

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 Designmodelles 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.

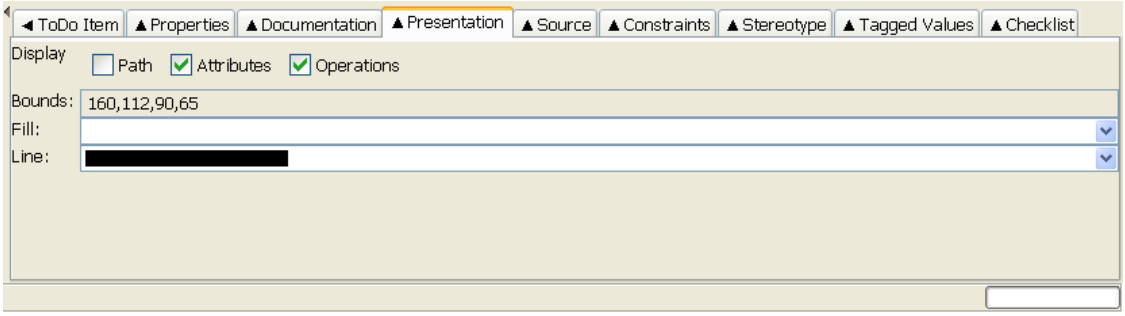
13.5. Das Register Darstellung

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 Editierbereiches.

  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`

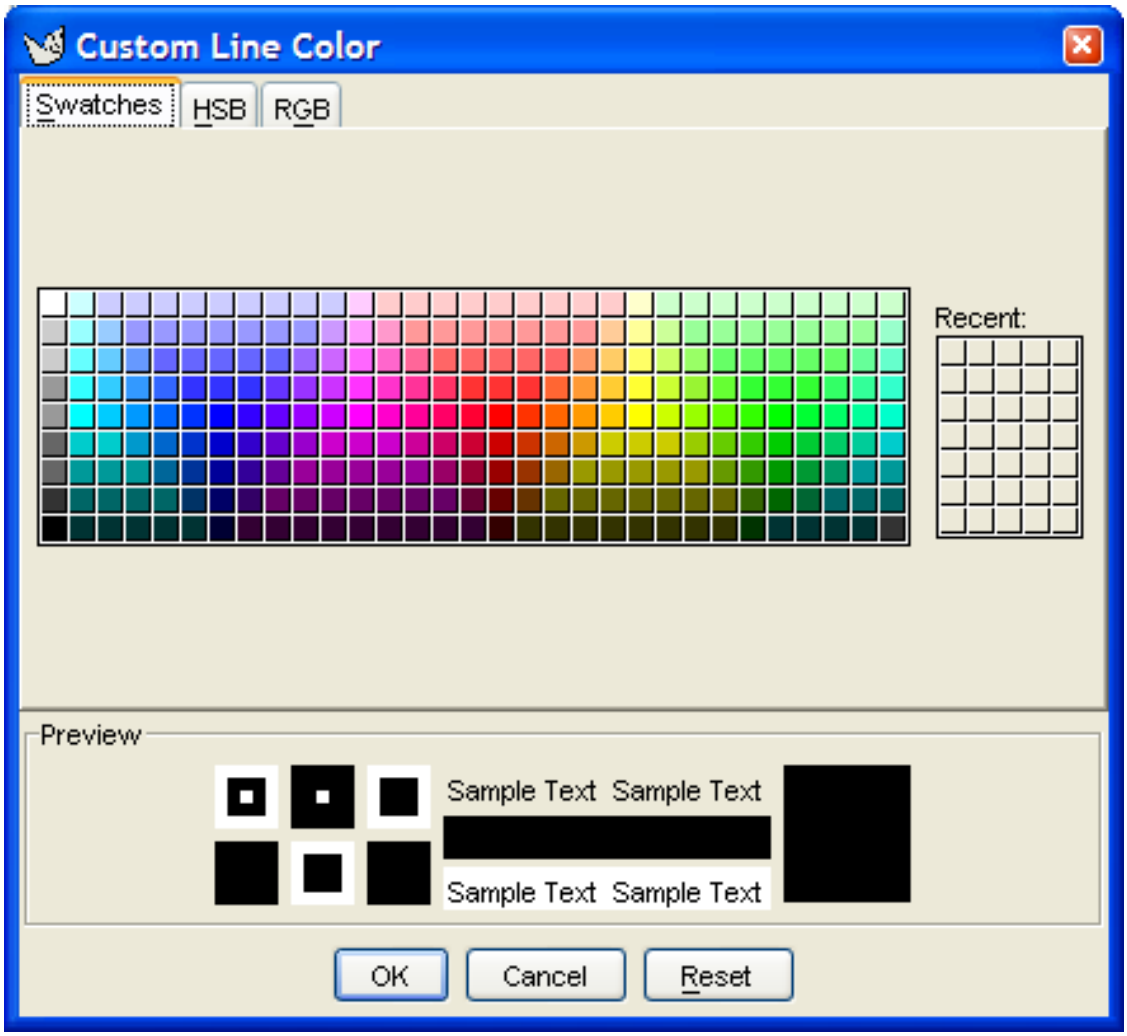


Abbildung 13.10. Der Dialog `Benutzerdefinierte Füll-/Linienfarbe`

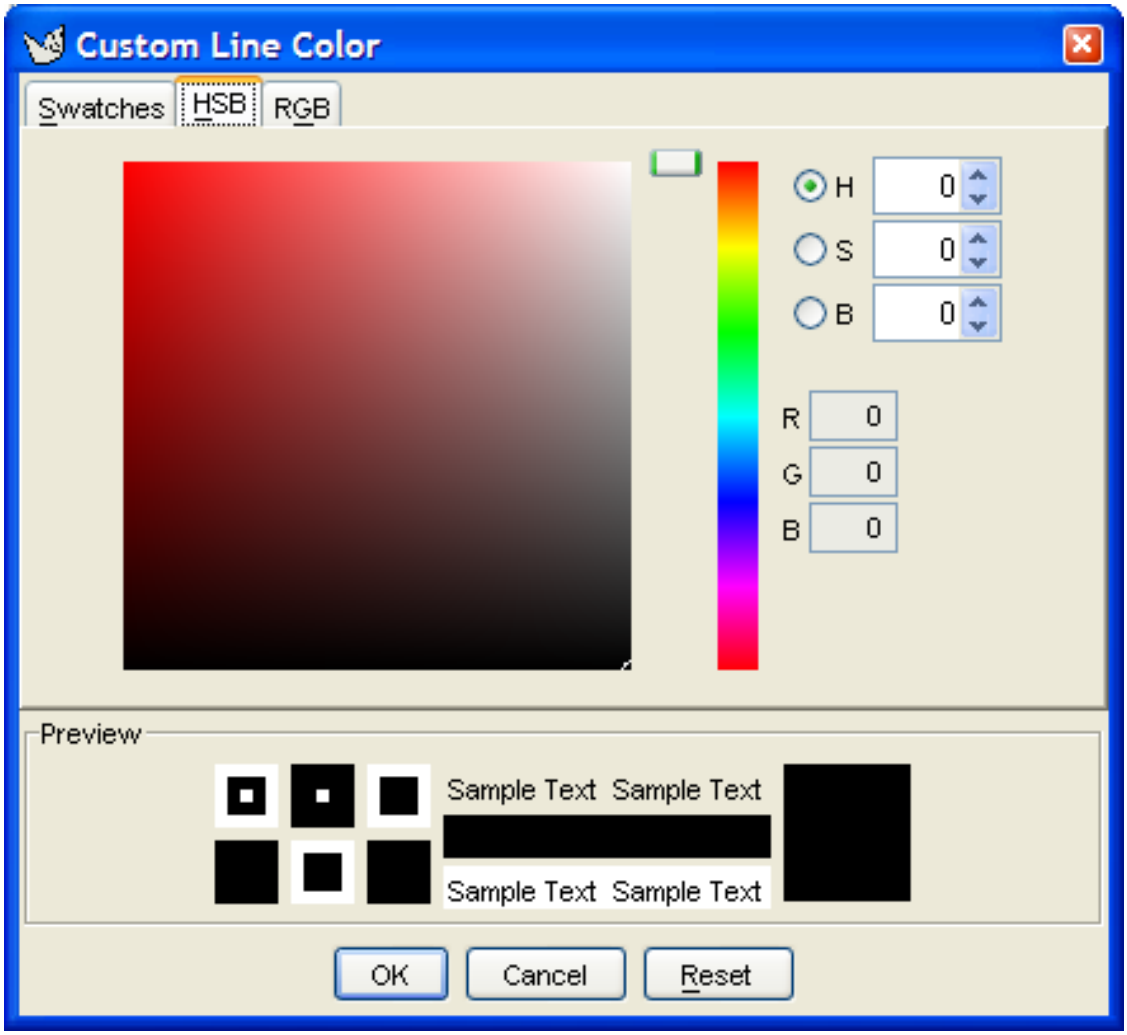
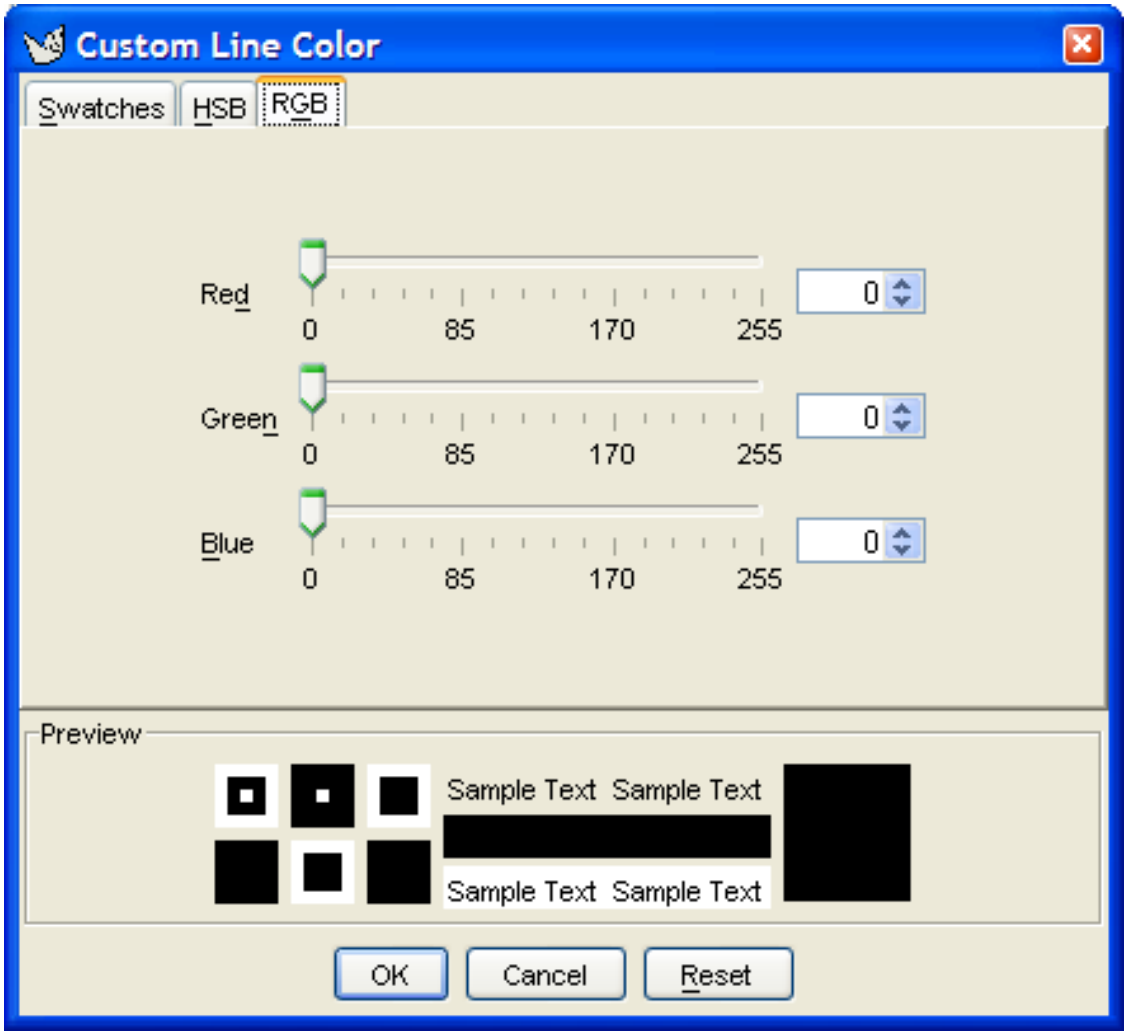


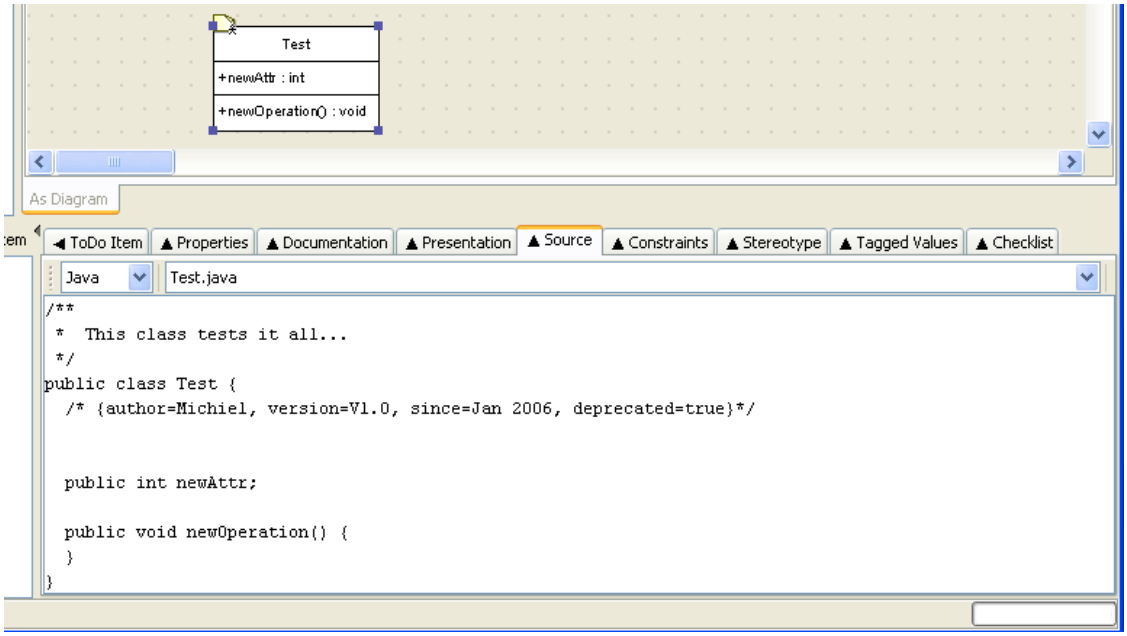
Abbildung 13.11. Der Dialog `Benutzerdefinierte Füll-/Linienfarbe`



13.6. Das Register Quellcode

Dieses Register zeigt den Quellcode, der für dieses Modellelement in der ausgewählten Sprache generiert wurde. ArgoUML generiert den Code zum Beispiel für Klassen und Schnittstellen. Der hier gezeigte Code kann in den angezeigten Dateien mit Hilfe der Funktionen im Menü Generieren gespeichert werden.

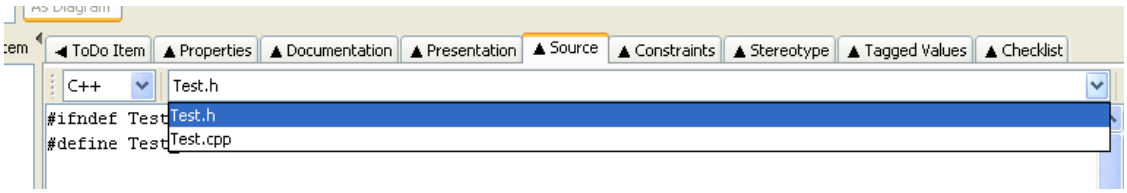
Abbildung 13.12. Das Register Quellcode einer Klasse.



Jeder von Ihnen hinzugefügte Code geht verloren - das ist nicht die Absicht von ArgoUML - verwenden Sie stattdessen eine IDE.

Das rechte Auswahlfeld erlaubt die Auswahl der Ausgabedatei. Diese Funktion ist nicht sehr sinnvoll für Sprachen, die den gesamten Code einer Klasse in einer Datei speichern, aber sie erfüllt seinen Zweck für z.B. C++, wo eine .h und .cpp-Datei generiert wird. Sie nachfolgendes Bild.

Abbildung 13.13. Ein C++ Beispiel.



13.7. Das Register Randbedingungen

Randbedingungen sind einer der in UML enthaltenen Erweiterungsmechanismen. ArgoUML ist mit einem leistungsfähigen Randbedingungseditor ausgerüstet, der auf der im UML 1.4-Standard definierten *Objekt Randbedingungssprache (Object Constraint Language; OCL)* basiert.

Achtung

Die OCL-Editorimplementierung für ArgoUML V0.24 unterstützt keine OCL-Randbedingungen außer Klassen und Eigenschaften.

Dies ist machmal 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.

> Vor OCL 2.0 wird keine generellere Definition von erlaubten Kontexten eingeführt. Der Schlüssel ist, dass Sie für jede benötigte Kontextdefinition einen kontextspezifischen Klassifizierer definieren müssen. Zum Beispiel einen Klassifizierer, der mit dem gleichen Schlüsselwort verknüpft wird. Die Ersteller der OCL-Spezifikation führen aus, dass dies kein Fehler der OCL-Spezifikation ist, aber selbst für UML einige Integrationsschritte erfordert. Es sieht so aus, das die Personen, die UML spezifizierten dachten, das dies in OCL spezifiziert wird (aus diesem Grund tun wir einen ersten Schritt in Richtung OCL 2.0).
>
> 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.

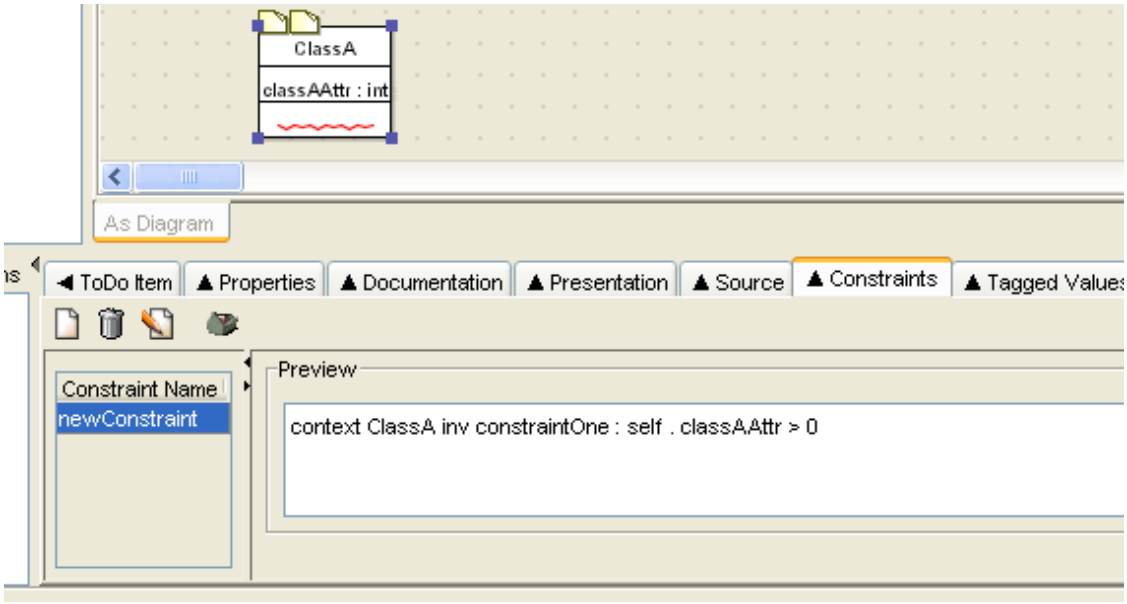
Der Standard definierte eine kleine Anzahl von Randbedingungen vor (zum Beispiel die Bedingung `xor` über einen Satz von Assoziationen, was anzeigt, dass nur einer in einer bestimmten Instanz bekannt ist).

Der Standard hat auch eine Anzahl von Umständen im Auge, in denen generelle Randbedingungen hilfreich sein können:

- Um zu kennzeichenen, 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 Randbedingung` 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

  Es scheint logisch, dass eine neue erstellte Randbedingung editiert werden muss. Aber ArgoUML V0.24 läuft beim Starten des OCL-Editors auf einen Fehler; sie müssen dies wie folgt tun: Erstens: die neue Randbedingung zuerst markieren, Zweitens: diese umbenennen und Drittens: die Schaltfläche `Bearbeiten Randbedingung` drücken. Es ist wesentlich für das erfolgreiche Erzeugen einer Randbedingung diesen vier Schritt akkurat zu folgen: erzeugen, markieren, umbenennen und editieren. Der Schritt des Umbenennens ist notwendig, weil die Gültigkeitsprüfung diese Randbedingung ablehnt, weil ihr Name von dem im Randbedingungstext benannten Namen abweicht. Aus dem gleichen Grund ist eine spätere Umbenennung einer Bedingung nicht möglich.

- `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.

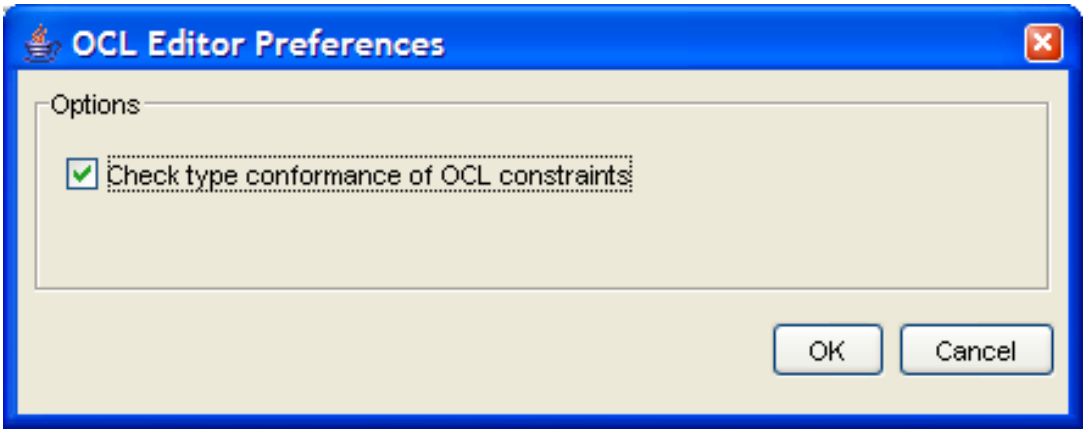
- `Bedingung editieren`. Dies startet den Bedingungseditor im Register `Randbedingungen` (siehe Abschnitt 13.7.1, „ Der Bedingungs-Editor “). Der Editor wird mit dem aktuell im Feld `Name Bedingung` markierten Randbedingung aufgerufen.

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.

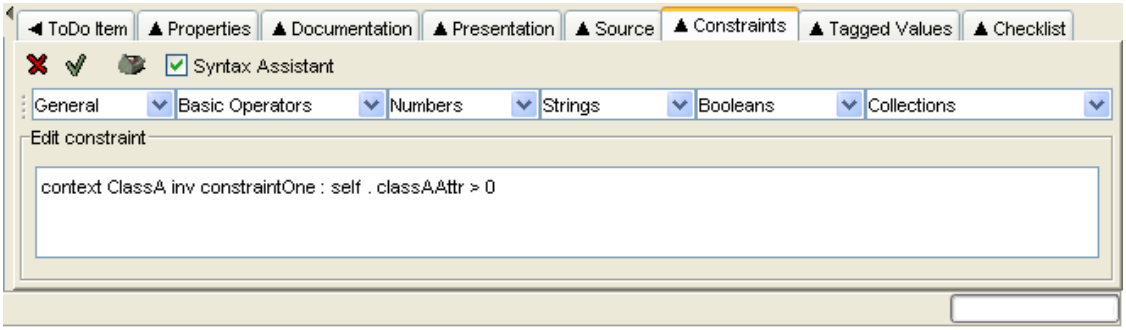
Das linke Feld ist mit `Name Bedingung` bezeichnet und listet alle Bedingungen auf (wenn es welche gibt) so weit sie für das markierte Modellelement definiert wurden. Eine Bedingung kann durch einen Taste 1-Klick markiert werden.

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 Bedingungseditors 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: =; <>; <; >; <=; >=; und ().
- Nummern. Arithmetische Operatoren und Funktionen. Einträge: +; -; *; /; mod; div; abs; max; min; round; und floor.
- Zeichenketten. Funktionen für Zeichenketten. Einträge: concat; size; toLower; toUpper; und 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.

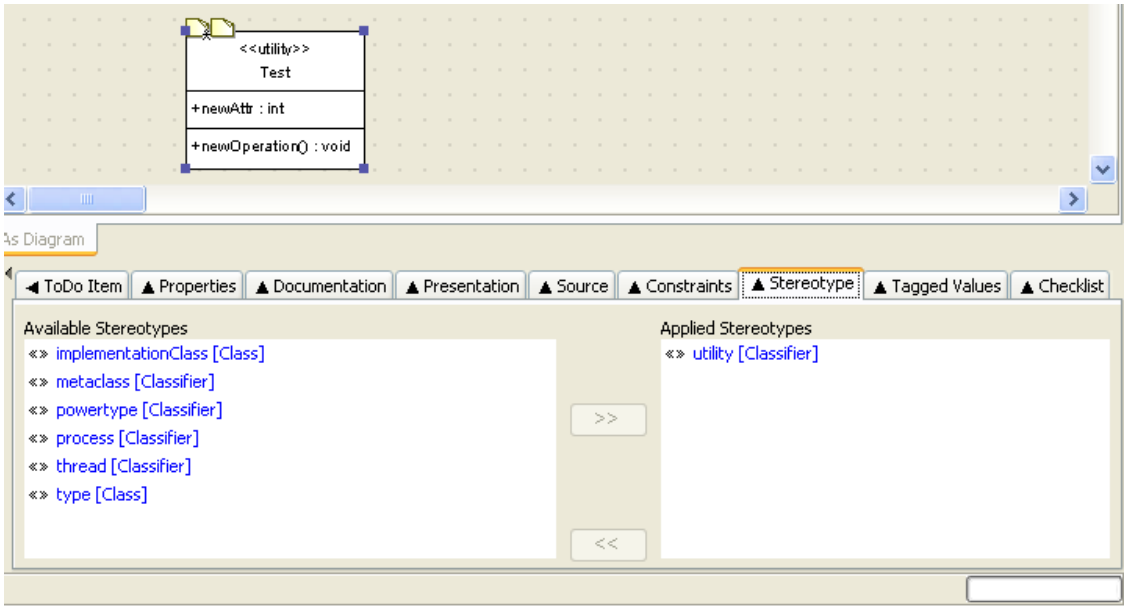
Der Rest des Registers beinhaltet ein beschreibbares Textfeld, welches den zu editierenden Text enthält. Die Maus-Schaltflächen weisen innerhalb des editierbaren Textfeldes ihr Standardverhalten auf (siehe Abschnitt 8.2, „ Generelles Verhalten der Maus in ArgoUML “).

Zusätzlich können Ausschneiden-, Kopieren- und Einfügen-Operationen über die Tastenkürzel Strg-X, Strg-C und Strg-V aufgerufen werden.

13.8. Das Register Stereotypen

Dieses Register zeigt die verfügbaren und anwendbaren Stereotypen für das aktuell markierte Modellelement. Es besteht aus zwei Feldern und zwei Schaltflächen. Die Schaltflächen erlauben es, die Stereotypen von einer Liste in die andere zu bewegen.

Abbildung 13.17. Ein Beispiel eines Registers Stereotypen für eine Klasse.



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 Modelles 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.

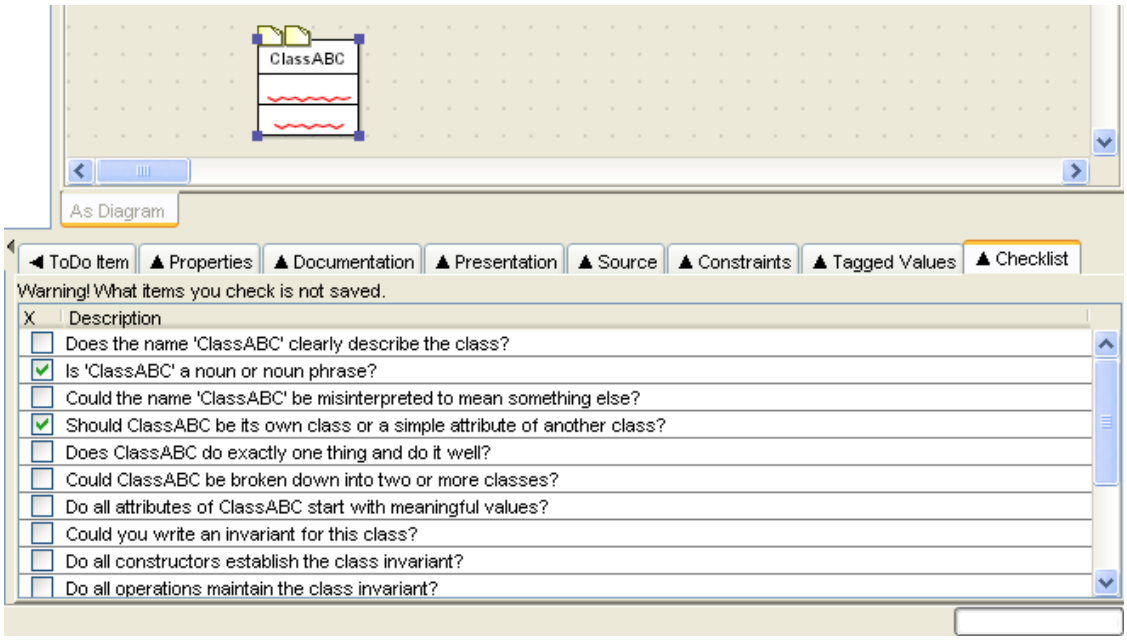
13.10. Das Register Checkliste

Das Durchführen von Designreviews und Inspektionen ist eine der effektivsten Arten Fehler während der Softwareentwicklung zu entdecken. Ein Designreview besteht typischerweise aus einer kleinen Zahl von Designern, Implementierern oder anderen Projektbeteiligter, die eine Besprechung durchführen, um ein Stück Softwareentwicklung zu reviewen. Viele Entwicklungsorganisationen haben Checklisten von häufigen Designproblemen für solche Reviews entwickelt. Die Vergangenheit zeigt auf, dass die Reviewer den Code ohne Besprechung mit Hilfe dieser Checklisten überprüfen und dies genauso effektiv ist, wie die Design- Reviews in Form einer Besprechung.

Aus diesem Grund wurden ArgoUML Checklisten hinzugefügt, die den Gedanken von der Design-Review-Checkliste unterstützen. Jedoch, sind die ArgoUML-Checklisten in die Anwenderschnittstelle des Design- werkzeuges und in die Designarbeit integriert.

Ein Softwaredesigner, der ArgoUML verwendet, kann für jedes Designelement eine Checkliste sehen. Das Register „Checkliste “ präsentiert eine Liste von unmarkierten Elementen, die zum aktuell markierten Designelement gehören. Zum Beispiel, wenn eine Klasse in einem Designdiagramm markiert ist, zeigt das Register Checklisten Elemente, die zum kritischen Nachdenken über diese Klasse anregen. Siehe nachfolgendes Bild. Designer können Elemente als betrachtet markieren. Markierte Elemente verbleiben in der Liste, um darzustellen, dass Sie bereits betrachtet wurden, während unmarkierte Elemente den Designer auffordern, über diesen neuen Designaspekt nachzudenken. ArgoUML unterstützt unterschiedliche Checklisten mit vielen möglichen Elementen.

Abbildung 13.18. Ein Beispiel für eine Checkliste einer Klasse.



Achtung

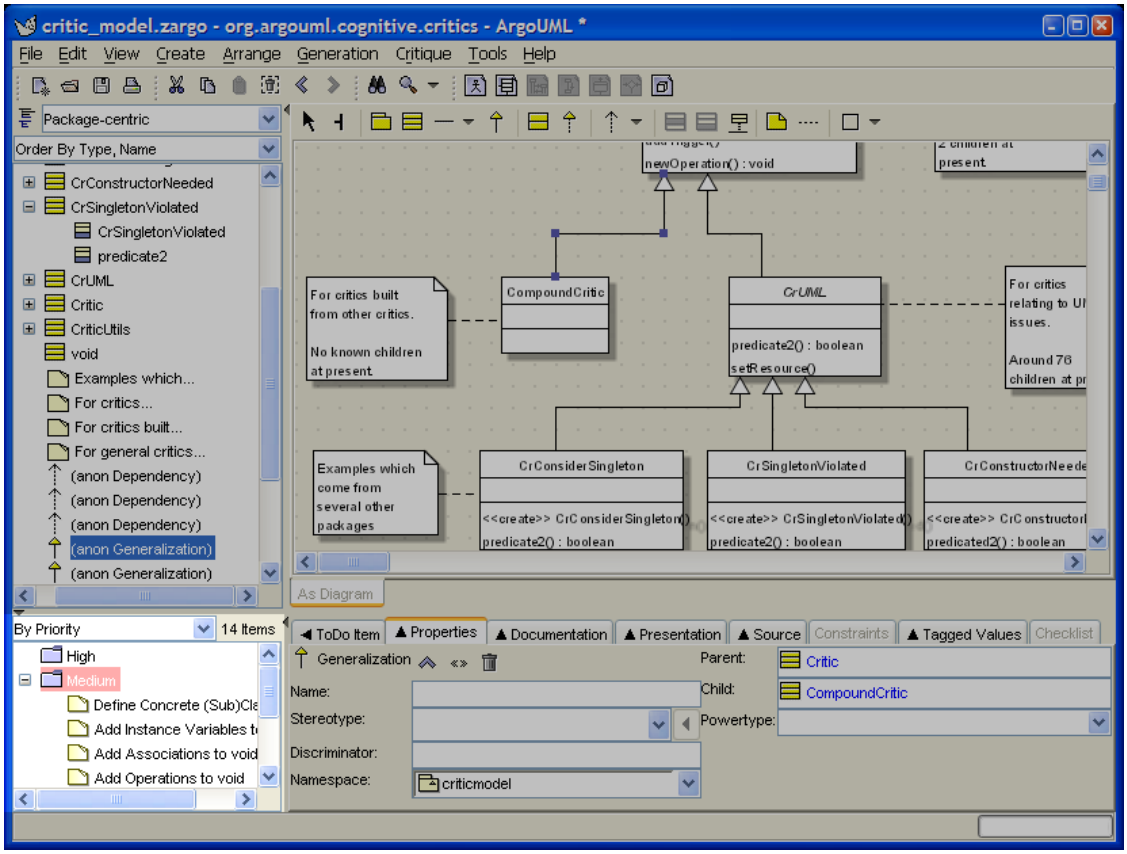
In der Release V0.22 von ArgoUML ist dieses Register nicht vollständig implementiert. Zum Beispiel werden die Markierungen nicht gespeichert.

Kapitel 14. Der Bereich Zu-Bearbeiten

14.1. Einleitung

Abbildung 14.1, „ Überblick über den Bereich Zu-Bearbeiten “ zeigt das ArgoUML-Fenster mit dem hervorgehobenen Bereich Zu-Bearbeiten.

Abbildung 14.1. Überblick über den Bereich Zu-Bearbeiten



Dieser Bereich hat Zugriff auf die Ergebnisse des Hinweisprozesses, 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 Hinweise können Sie in der Diskussion des `Hinweise`-Menüs (siehe Abschnitt 10.9, „ Das Menü Hinweise “) 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 farbigem 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 dem die durch eine bestimmten Hinweis generierten Zu-Bearbeiten-Elementen verknüpft sind, können über das Menü `Hinweise > Hinweise anzeigen...` geändert werden (siehe Abschnitt 10.9.4, „ Hinweise anzeigen... “).

- `Nach Entscheidung`. Die Zu-Bearbeiten-Elemente sind in 17 Hierarchien anhand von Designmerkmalen organisiert: `Unkategorisiert`, `Klassenauswahl`, `Verhalten`, `Benennung`, `Speicher`, `Vererbung`, `Containment`, `Geplante Erweiterungen`, `Zustandsautomaten`, `Design Patterns`, `Beziehungen`, `Instantiation`, `Modularität`, `Expected Usage`, `Methoden`, `Codegenerierung` und `Stereotypen`.

Die Details der Hinweise in jeder Kategorie werden in Abschnitt 10.9.2, „ Design-Wichtungen... “ diskutiert.

- `Nach Ziel`. ArgoUML verfolgt das Konzept, dass Hinweise entsprechend der sie beeinflussenden Benutzerziele gruppiert werden. Diese Darstellung gruppiert die Zu-Bearbeiten- Elemente in einer Hierarchie entsprechend der Ziele.

  ### Achtung

  In der aktuellen Release von ArgoUML gibt es nur ein Ziel `Unspezifiziert`. Alle Zu-Bearbeiten-Elemente werden unter dieser Überschrift erscheinen.

- `Nach Problemen`. Die Zu-Bearbeiten-Elemente werden hierarchisch entsprechend des Modellelementes organisiert, welches das Problem verursachte. Zu-Bearbeiten-Elemente, die manuell mit der Schaltfläche "Neues Zu-Bearbeiten-Element" erzeugt wurden (z.B. nicht durch einen Hinweis) werden hier nicht aufgelistet.

- `Nach Kurzbezeichnung`. Die Zu-Bearbeiten-Elemente werden danach hierarchisch organisiert, welcher Hinweis das Zu-Bearbeiten-Element generierte. Der Klassenname des Hinweises wird anstelle seiner Überschrift aufgelistet.

- `Nach Wissensgebiet`. ArgoUML hat das Konzept, dass Hinweise einen Ausschnitt eines Wissensgebietes reflektieren. Diese Darstellungsoptionen gruppieren die Hinweise entsprechend der Kategorie des Wissensgebietes: `Entwurf`, `Korrektheit`, `Vollständigkeit`, `Konsistenz`, `Syntax`, `Semantik`, `Optimierung`, `Darstellbarkeit`, `Organisierbarkeit`, `Experiencial` und `Werkzeug`. Die Hauptkategorie (Entwürfe) enthält die manuell eingegebenen Zu-Bearbeiten-Elemente.

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 Hinweise

15.1. Einleitung

Die Schlüsselfunktion, die ArgoUML von anderen UML-Werzeugen unterscheidet, ist die Verwendung von Konzepten der kognitiven Psychologie. Die dahinter stehende Theroie ist in Jason Robbins' PhD Dissertation http://argouml.tigris.org/docs/robbins_dissertation/ [http://argouml.tigris.org/docs/robbins_dissertation/] beschrieben.

Hinweise 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 Hinweise sind asynchrone Prozesse, die parallel zu ArgoUML ablaufen. Änderungen benötigen eine oder zwei Sekunden, bis die Hinweise erneut erzeugt wurden.

15.1.1. Terminologie

Die *Hinweise* sind Hintergrundprozesse, die das aktuelle Modell anhand verschiedener „guter“ Designkriterien überprüfen. Es gibt einen Hinweis für jedes Designkriterium.

Die Ausgabe eines Hinweises ist ein *Hinweis* - eine Anweisung über einige Aspekte des Modelles, die nicht der guten Designpraxis folgen.

Natürlich wird ein Hinweis nur vorschlagen, wie der gefundene Designmangel behoben werden kann, in dem es ein *Zu-Bearbeiten-Element* erzeugt.

15.1.2. Design-Mangel

ArgoUML kategorisiert Hinweise entsprechend der von ihnen adressierten Designmängel (einige Hinweise können in mehr als einer Kategorie erscheinen). Aktuell gibt es 16 solcher Kategorien.

In diesen Handbuch sind die Beschreibungen der Hinweise entsprechend ihres Designmangels in Abschnitten gruppiert.

15.2. Unkategorisiert

Diese Hinweise passen zu keiner der anderen Kategorien.

ArgoUML hat keinen Hinweis mit dieser Kategorie. Vielleicht werden einige in späteren Versionen hinzugefügt.

15.3. Klassenauswahl

Dies sind Hinweise, die sich darauf beziehen, wie Klassen ausgewählt und verwendet wurden.

ArgoUML hat die folgenden Hinweise 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).

Datentypen können nicht mit Klassen assoziiert werden, es sei denn, der Datentyp ist Teil einer Komposition (schwarzer Diamant). So eine Assoziation reflektiert die feste Verbindung einer Sammlung von Datentyp-Instanzen zu einer Klasseninstanz. In der Konsequenz ist so ein Datentyp ein Attribut der Klasse mit einer Kardinalität.

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. Veringere die Anzahl der Klassen im Namensraum <Namensraum>

Ein Vorschlag, die Verständlichkeit durch weniger Klassen in einem Namensraum zu verbessern. Wenn ein Namensraum (wie ein Modell, ein Paket, oder eine Klasse) zu viele Klassen aufweist, wird es für Menschen sehr schwierig dies zu verstehen. Einen verständlichen Satz von Namensräumen zu definieren ist ein wichtiger Teil Ihres Designs.

Der Assistent dieses Hinweises erlaubt das Setzen eines Schwellwertes, z.B. die maximale Anzahl von Klassen, ab der der Hinweis 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 Klassen, bis der Hinweis erneut ausgelöst wird. Der Neustart von ArgoUML setzt dieses Anzahl wieder auf seinen Standardwert: 20.

15.3.3. Diagramm aufräumen

Vorschlag, dass das Diagramm verbessert werden sollte, indem man sich überlappende Modellelemente auseinander zieht.

15.4. Benennung

Dies sind Hinweise, die sich auf die Benennung von Modellelementen beziehen. Die aktuelle Version von ArgoUML hat 18 Hinweise 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.

Wo es Signaturkonflikte gibt, kann kein korrekter Code in den hauptsächlich verwendeten OO-Sprachen generiert werden. Es führt auch zu einer sehr unklaren Semantik des Designs.

Beim Vergleichen von Signaturen betrachtet der Hinweis:

1. den Namen;
2. die Liste der Ein-, Aus- und Ein-Ausgabeparamtertypen in ihrer *Reihenfolge*;

Nur wenn diese sowohl in Typ und Reihenfolge übereinstimmen, wird die Signatur als gleich betrachtet.

Dies folgt der Linie von Java/C++, welche die Rückgabeparameter bei der Signatur ignorieren. Dies *kann* bei einigen funktionalen OO-Sprachen unvorteilhaft sein.

Anmerkung

Einige Puristen würden argumentieren, dass der Vergleich zwischen Ein-, Aus- und Ein-Ausgabeparametern differenzieren sollte. Jedoch kann keine praktische Programmiersprache so etwas tun, wenn sie einen überladenen Methodenaufruf auflöst. Aus diese Grund behandelt dieser Hinweis sie alle pauschal.

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 betrachteten 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)

Das betrachtete Zustand hat keinen Namen erhalten (es wird im Modell als `(Unbenannter Zustand)` erscheinen). Hinweis, dass gutes Design fordert, dass alle Zustände benannt werden.

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

Zeichen voneinander). Hinweis, dass dies potentiell zu einer Konfusion führt.

Achtung

Dieser Hinweis kann manchmal störend sein, da es manchmal nützlich und gutes Design ist, eine Serie von Modellelemente `var1`, `var2` usw. zu haben.

Es ist wichtig sich daran zu erinnern, dass Hinweise 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

Alle Modellelementnamen in ArgoUML dürfen nur aus Buchstaben, Ziffern und Unterstrichen bestehen. Dieser Hinweis weist Sie darauf hin, dass eine Entität nicht dieser Anforderung entspricht.

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 gebinnt. In ArgoUML wird dies diesen Hinweis auslösen, es sei denn, der Konstruktor erhält den Stereotyp «`create`».

Es ist wichtig sich daran zu erinnern, dass Hinweise 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.

Anmerkung

Obwohl sie diesen Hinweis 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

Hinweise, die sich auf die Attribute von Klassen beziehen.

Die aktuelle Version von ArgoUML hat in dieser Kategorie die folgenden Hinweise.

15.5.1. Überarbeiten Sie die Attributnamen, um einen Konflikt zu vermeiden

Dieser Hinweis 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

Sie haben bis jetzt keinen Konstruktor für die Klasse *Klasse* definiert. Konstruktoren initialisieren neue Instanzen, sodaß deren Attribute gültige Werte aufweisen. Diese Klasse benötigt wahrscheinlich einen Konstruktor, weil nicht alle seiner Attribute initiale Werte aufweisen.

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 Designengpaß in sich birgt.

Der Assistent dieses Hinweises erlaubt das Einstellen eines Schwellwertes, z.B. die maximal erlaubte Anzahl von Attributen bevor dieser Hinweis 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 der Hinweis erneut ausgelöst wird. Der Neustart von ArgoUML setzt dieses Anzahl wieder auf seinen Standardwert: 7.

15.6. Geplante Erweiterungen

Hinweise, 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 Hinweise 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

Hinweise, die sich auf Zustandsautomaten beziehen.

ArgoUML hat die folgenden Hinweise in dieser Kategorie.

15.7.1. Reduzieren Sie die Anzahl der Transitionen im <Zustand>

Hinweis, das der angegebene Zustand so viele Transitionen aufweist, dass er zu einem Leistungsengpass werden dürfte.

Der Assistenz dieses Hinweises erlaubt das Einstellen eines Schwellwertes, z.B. die maximale Anzahl von Transitionen bevor dieser Hinweis 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 Transitionen, bis der Hinweis erneut ausgelöst wird. Der Neustart von ArgoUML setzt dieses Anzahl wieder auf seinen Standardwert: 10.

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 dieses Hinweises erlaubt die Einstellung eines Schwellwertes, z.B. die maximale Anzahl von erlaubten Zuständen bevor dieser Hinweis 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 Zustände, bis der Hinweis erneut ausgelöst wird. Der Neustart von ArgoUML setzt dieses Anzahl wieder auf seinen Standardwert: 20.

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 initialien 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 ein gültiger Hinweis ist. Es ist sicherlich akzeptabel, eine Transistion ohne Wächter zu haben - die Transition wird immer genommen, wenn das Signal ausgelöst wird.

15.7.13. Das Diagramm aufräumen

Dieser Hinweis 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 (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

Hinweise über die Nutzung von Designmustern in ArgoUML.

Diese beziehen sich auf die Nutzung von Mustern, wie sie durch die sogenannte „Viererbande“ beschrieben wurden. ArgoUML nutzt diese Kategorie auch für Hinweise, die sich auf Verteilungs- und Sequenzdiagramme beziehen. Die aktuelle Version von ArgoUML hat die folgenden Hinweise in dieser Kategorie.

15.8.1. Ziehen Sie Nutzung des Singleton-Musters für eine <class> in Betracht.

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.

Unter diesen Umständen sollten sie in Betracht ziehen, dass Sie genau eine Instanz dieser Klsse haben, indem Sie das Singleton-Muster verwenden. Die Nutzung des Singleton-Musters kann Zeit und Speicherplatz einsparen. Innerhalb von ArgoUML kann dies durch Nutzung des Stereotyps «singleton» auf diese Klasse umgesetzt werden.

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.

1. Sie müssen ein statisches Attribut definieren (eine Klassenvariable), welche die Instanz aufnimmt. Diese muss aus diesem Grund den Typ *Klasse* aufweisen.

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 einmals 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>

Diese Klasse ist mit dem «singleton»-Stereotyp versehen, stimmt aber nicht mit den für Singletons geltenden Randbedingungen überein (ArgoUML akzeptiert auch den Stereotyp «Singleton » beim Definieren eines Singletons). Eine Singletonklasse kann maximal eine Instanz haben. Das bedeutet, dass die Klasse den Designkriterien für ein Singleton entsprechen muss (siehe Abschnitt 15.8.1, „ Ziehen Sie Nutzung des Singleton-Musters für eine <class> in Betracht. “).

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 konsitenten 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.

Bei der 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 autonomes 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 autonomes 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

Ein Hinweis, dass eine Verknüpfung (z.B. eine Assoziation), die Objekte in einem Verteilungsdiagramm verbindet, ein Ende in einer Komponente und das andere Ende in einer Komponenteninstanz (da Objekte in beiden auftreten können) hat. Dies macht keinen Sinn.

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

Das Auslösen dieses Hinweises weist auf schwerwiegendes Probleme hin, da ArgoUML keine Mechanismen für das Erzeugen einer Verknüpfung ohne einen Auslöseimpuls enthält. Es weist wahrscheinlich darauf hin, dass das Diagramm durch Laden eines defekten Projektes mit einer XMI-Datei erzeugt wurde, welche eine Verknüpfung ohne Auslöseimpuls beschreibt. Diese wurde möglicherweise durch ein anderes Werkzeug erzeugt.

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

Hinweise, die sich in ArgoUML auf Assoziationen beziehen.

Die aktuelle Version von ArgoUML hat die folgenden Hinweise 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

Dieser Hinweis ist in der V0.14 von ArgoUML bedeutungslos, da sie Assoziationsklassen nicht unterstützt.

15.9.2. <Assoziation> navigierbar machen

Hinweis, dass die betreffende Assoziation in keine Richtung navigierbar ist. Dies ist im UML-Standard erlaubt, aber es hat in einem praktischen Design keinerlei Bedeutung.

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.

Dieser Teil des Designs sollte geändert werden, bevor Sie Code von diesem Design generieren. Wenn Sie den Code generieren, bevor Sie dieses Problem lösen, wird der Code nicht mit dem Design übereinstimmen.

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. Entfernen Sie die Markierung `Navigierbar` für jedes dieser Enden.

Die Assoziation sollte dann mit einem Pfeil in Richtung der Schnittstelle erscheinen.

Wenn eine Assoziation zwischen einer Klasse und einer Schnittstelle in ArgoUML erzeugt wird, ist diese standardmäßig nur von der Klasse zur Schnittstelle navigierbar. Jedoch verhindert es ArgoUML nicht, wenn die Navigierbarkeit danach fehlerhaft geändert wird. Was diesen Hinweis auslösen würde.

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

Dieser Hinweis 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 dieses Hinweises erlaubt das Einstellen eines Schwellwertes, z.B. die maximale Anzahl von erlaubten Assoziationen bevor diese Kritk ausgelöst wird.

Achtung

Diese Zahl wird nicht dauerhaft gespeichert. Es gibt keinen Weg sie zu reduzieren, nachdem sie nach oben gesetzt wurde. Es sei denn, man erzeugt mehr Assoziationen bis der Hinweis erneut ausgelöst wird. Ein Restart von ArgoUML setzt diese Zahl auf seinen Standard zurück: 7.

15.9.7. Kante sichtbarer machen

Dieser Hinweis wird in einer früheren Designkategorie diskutiert (siehe Abschnitt 15.7.14, „ Eine Kante sichtbarer machen “).

15.10. Instanzen bilden

Hinweise, die sich auf das Bilden von Instanzen bei Klassifizierern in ArgoUML beziehen.

Die aktuelle Version von ArgoUML hat keine Hinweise in dieser Kategorie.

15.11. Modularität

Hinweise, die sich auf die modulare Entwicklung in ArgoUML beziehen.

Die aktuelle Version von ArgoUML hat die folgenden Hinweise 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.

Dieser Hinweis 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

Dieser Hinweis kann in der aktuellen Implementierung von ArgoUML keine hierarchischen Namensräume verarbeiten. Als Konsequenz daraus, wird der Hinweis für Assoziationen ausgelöst, bei denen der sich unmittelbare Namensraum der zugewiesenen Klassifizierer von dem der Assoziation unterscheidet, auch wenn sie Teil der gleichen Namensraumhierarchie sind.

15.11.2. Fügen Sie Elemente zum Paket <Paket> hinzu.

Hinweis, dass das angegebene Paket keinen Inhalt hat. Gutes Desgin weist Pakete auf, die erzeugt wurden, Dinge hinein zu tun.

Anmerkung

Dieser Hinweis wird immer ausgelöst, wenn Sie erstmalig ein Paket erzeugen, da Sie kein Paket erzeugen können, das nicht leer ist.

15.12. Erwartete Verwendung

Hinweise, die sich auf eine generell akzeptierte, gute Praxis beziehen.

Die aktuelle Version von ArgoUML hat einen Hinweis in dieser Kategorie.

15.12.1. Diagramm aufräumen

Dieser Hinweis wird in einer früheren Designkategorie diskutiert (siehe Abschnitt 15.3.3, „ Diagramm aufräumen “).

15.13. Methoden

Hinweise, die sich auf Operationen in ArgoUML beziehen.

Die aktuelle Version von ArgoUML hat die folgenden Hinweise in dieser Kategorie.

15.13.1. Ändere Namen oder Signaturen im <Modellelement>

Dieser Hinweis 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-Wartungsengpaß zu werden.

Der Assistent dieses Hinweises erlaubt das Setzen eines Schwellwertes, z.B. die maximale Anzahl von erlaubten Operationen bevor dieser Hinweis ausgelöst wird.

Achtung

Diese Anzahl wird nicht dauerhaft gespeichert. Es gibt keinen Weg diese Anzahl zu reduzieren, nachdem sei einmal hochgesetzt wurde. Außer, Sie erzeugen weitere Operationen bis dieser Hinweis erneut ausgelöst wird. Der Neustart von ArgoUML setzt diese Anzahl auf den Standardwert: 20 zurück.

15.14. Code-Generierung

Hinweise, die sich auf die Code-Generierung in ArgoUML beziehen.

Die aktuelle Version von ArgoUML hat einen Hinweis 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

Hinweise, die sich auf Stereotypen in ArgoUML beziehen.

Die aktuelle Version von ArgoUML hat keine Hinweise in dieser Kategorie.

15.16. Vererbung

Hinweise, die sich mit der Generalisierung und Spezialisierung in ArgoUML befassen.

Die aktuelle Version von ArgoUML hat die folgenden Hinweise in dieser Kategorie.

15.16.1. Überprüfen Sie die Attributnamen, um einen Konflikt zu vermeiden.

Dieser Hinweis 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 erbt, was nicht erlaubt ist.

Achtung

Dieser Hinweis ist in der aktuellen Release von ArgoUML standardmäßig als inaktiv markiert (der Einzige so markierte Hinweis). Sie wird nicht ausgelöst, bis sie aktiviert wird.

15.16.3. Die Klasse muß abstrakt sein

Dieser Hinweis 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. Enferne 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

Dieser Hinweis 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

Dieser Hinweis wird unter einer früheren Designkategorie diskutiert (siehe Abschnitt 15.7.14, „ Eine Kante sichtbarer machen “).

15.17. Containment

Hinweise, 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 Hinweise in dieser Kategorie.

15.17.1. Entferne zirkuläre Komposition

Hinweis, das 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 Hinweisassistenten die Schaltfläche `Weiter >`, setzen seine Aggregation manuell mit Hilfe des Taste 2-Popup-Menüs oder des Eigenschaftsregisters auf `none`.

Eine Komposition (korrekter verbundene Aggregation genannt) wird verwendet, wenn es eine Ganzes-Teil-Beziehung gibt, die eins-zu-eins oder eins-zu-viele sind und die Lebensdauer des Teils unauflösbar von der Lebensdauer des Ganzen abhängt. Instanzen des Ganzen sind für das Erzeugen und Löschen der Instanzen der verknpften Teile verantwortlich. Das bedeutet auch, dass eine Klasse nur Teil einer verbundenen Aggregation sein kann.

Ein Beispiel einer verbundenen Aggregation könnte eine Datenbank von Autos mit deren Reifen sein. Dies ist eine eins-zu-vier- Beziehung und der Datenbankeintrag für einen Reifen ist mit seinem Auto verknüpft. Wenn ein Auto in der Datenbank aufhört zu existieren, dann trifft dies auch auf seine Reifen zu.

Die Aggregation (korrekter aufgeteilte Aggregation genannt) wird verwendet, wenn es eine Ganzes-Teil-Beziehung gibt, die nicht den Kriterien für eine verbunden Aggregation übereinstimmt. Ein Beispiel könnte eine Datenbank von Universitätskursen und den Studenten sein, die diese belegen. Es gibt eine Ganzes-Teil-Beziehung zwischen den Kursen und den Studenten. Jedoch gibt es keine Lebensdauerbeziehung zwischen den Studenten und den Kursen (ein Student existiert weiter, nachdem er den Kurs absolviert hat) und die Beziehung lautet viele-zu-viele.

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

Dieser Hinweis 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 Sub-Pakete 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`
: Siehe Abschnitt 16.2.2, „ Symbolleiste Modelleigenschaften “ und Abschnitt 16.2.3, „ Eigenschaftsfelder des Modelles “ unten.

`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`).

  Werte `true`, bedeutet: die Klasse ist redundant - sie kann formal von anderen Elementen abgeleitet werden oder, `false` bedeutet: sie kann nicht von anderen Elementen abgeleitet werden.

  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 Modelles, wie alle anderen Pakete, wird per Konvention in Kleinbuchstaben geschrieben.

Anmerkung

Der Standardname, der einem neuen Modell in ArgoUML zugwiesen wird ist `unbenanntesModell`. Er ist daher fehlerfrei und garantiert, dass ArgoUML immer mit mindestens einem Problem startet, was durch die Design- Hinweise 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`

Textfeld. Nimmt den Namensraum des Modelles auf. Dies ist die Pakethierarchie. Da sich das Modell an der Spitze der Hierarchie befindet, ist dieses Feld immer leer.

`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`.

- `abstract` wird verwendet, um zu deklarieren, dass dieses Modell nicht instantiiert werden kann. Aber es muß immer spezialisiert werden.

  Die Bedeutung von `abstract`, wenn es auf ein Modell angewendet wird ist nicht klar. Es könnte bedeuten, dass das Modell Schnittstellen oder abstrakteKlassen ohne Realiserung enthält. Da ArgoUML nur ein Modell erlaubt, ist das markieren dieses Feldes bedeutungslos.

- `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 Modelles 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`

Textbereich. Eine Liste der Pakte, Klassen, Schnittstellen, Datentypen, Akteure, Anwendungsfälle, Assoziationen, Generalisierungen und Stereotypen innerhalb des Modelles auf oberster Ebene.

Ein Taste 1-Klick auf jedes dieser Modellelemente veranlasst, dass zu diesem Modellelement gesprungen wird.

16.3. Datatyp

Datentypen kann man sich als einfache Klassen denken. Sie haben keine Attribute und jede Operation von Ihnen darf keine Seiteneffekte aufweisen. Eine nützliche Analogie sind primitive Datentypen in einer Sprache wie Java. Die Integerzahl „3“ steht für sich selbst - sie hat keine innere Struktur. Es gibt Operationen (z.B. Addition) auf Integerzahlen, aber wenn sie `3 + 4` ausführen, ist das Ergebnis eine neue Zahl, „3“ und „4“ bleiben in diesem Beispiel unverändert.

Innerhalb von UML 1.4 ist ein `Datentyp` 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.

Innerhalb von ArgoUML können neue Datentypen mit Hilfe der Schaltfläche `Neuer Datentyp` im Register Eigenschaften des Modelles und Paketen (in diesem Fall ist der neue Datentyp auf den Scope des Paktes beschränkt), als auch im Register Eigenschaften für Datentypen erzeugt werden. Datentypen können auch mit dem Symbol in der Diagrammsymbolleiste eines Klassendiagrammes erzeugt werden.

Der UML 1.4-Standard erlaubt das Plazieren von benutzerdefinierten Datentypen in Klassendiagrammen, um deren Vererbungsstruktur zu definieren. Dies ist auch in ArgoUML möglich. Er wird im Diagramm durch einen Rahmen mit zwei Bereichen dargestellt, wobei der oberste mit «`Datentyp`» gekennzeichnet ist und den Namen enthält. Der untere enthält die Operationen.

16.3.1. Register Datentypdetails

Folgende Detail-Register sind für Datentypen aktiv:

`Zu-Bearbeiten-Element`
Standard-Register.

`Eigenschaften`

Siehe Abschnitt 16.3.2, „ Symbolleiste Datentypeigenschaften “ und Abschnitt 16.3.3, „ Eigenschaftsfelder für den Datentyp “ unten.

`Dokumentation`
Standard-Register. Siehe Abschnitt 13.4, „ Das Register Dokumentation “.

`Quellcode`
Standard-Register. Nicht im Einsatz. Es würde eine Klassendeklaration für den neuen Datentyp erwarten, um die Code-Generierung zu ermöglichen.

`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 Modelles, 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 neue 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`

Textfeld. Der Name des Datentyps. Die primitiven Datentypen haben alle kleingeschriebene Namen. Es gibt allerdings keine formale Konvention hierfür.

Anmerkung

Der Standardname neu erzeugter Datentypen ist ein leerer String „“. Datentypen mit leeren String-Namen erscheinen im Explorer unter `(Unbenannte Datentypen)` .

`Namensraum`
DropDown-Kombinationsfeld. Erlaubt die Änderung des Namensraumes für den Datentyp. Dies ist die Pakethierarchie.

`Modifizierer`

Markierfeld, mit den Einträgen `Abstract`, `Leaf` und `Root`.

- `Abstract` deklariert, dass dieser Datentyp nicht instantiiert werden kann und daher immer

spezialisiert 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`
Textbereich. Listet alle Operationen auf, die zu diesem Datentyp definiert sind. Ein Taste 1-Doppelklick springt zu der markierten Operation. Ein Taste 2-Klick öffnet ein Popup-Menü mit zwei Einträgen.

- `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)

Eine Enumeration ist ein primitiver Datentyp, der eine begrenzten Anzahl von Werten in einer kurzen Liste aufweist. Sie hat keine Attribute und jede Operation darf keine Seiteneffekte haben. Eine nützliche Analogie dazu ist der primitive Datentyp boolean in einer Sprache wie Java. Boolean steht für sich selbst - und hat keine innere Struktur. Es gibt Operationen (z.B. das logische xor) für die Booleans, aber wenn ich `true xor true` ausführe, ist das Ergebnis ein neuer Boolean und die beiden Original-Boolean bleiben unverändert.

Innerhalb von UML 1.4 ist `Enumeration` eine Subklasse der Metaklasse `Datentyp`.

Der große Unterschied zu anderen Datentypen ist, dass eine Enumeration `Enumeration-Literale` aufweist. Z.B. die Enumeration „boolean“ ist mit zwei Enumeration- Literalen „true“ und „false“ definiert.

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.

Der UML 1.4-Standard erlaubt es, benutzerdefinierte Enumerationen in Klassendiagrammen zu plazieren, um deren Vererbungsstruktur zu definieren. Dies ist auch in ArgoUML möglich. Sie wird im Diagramm durch einen Rahmen mit drei Bereichen dargestellt. Der oberste Bereich ist mit «`enumeration`» markiert und enthält den Namen. Der mittlere Bereich zeigt die Enumerations-Literale. Der untere enthält die Operationen.

16.4.1. Die Detail-Register Enumeration

Folgende Detail-Register sind für Enumerationen aktiv.

`Zu-Bearbeiten-Element`
: Standard-Register.

`Eigenschaften`

: Siehe Abschnitt 16.4.2, „ Eigenschaftssymbolleiste Enumeration “ und Abschnitt 16.4.3, „ Eigenschaftsfelder für Enumerationen “ unten.

`Dokumentation`
: Standard-Register. Siehe Abschnitt 13.4, „ Das Register Dokumentation “.

`Darstellung`
: Standard-Register.

`Quellcode`
Standard-Register.

`Stereotyp`
Standard-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`

Standard-Register. Im UML-Metamodell sind für die `Enumeration` keine Standard-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`

Textfeld. Der Name der Enumeration. Die primitiven Enumerationen haben alle Namen in Kleinbuchstaben. Hierfür gibt es allerdings keine formale Konvention.

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`
Ein Kombinationsfeld. Erlaubt das Ändern des Namensraumes der Enumeration. Dies ist die zusammengesetzte Hierarchie.

`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`
Textbereich. Listet jedes Element auf, dass von dieser Enumeration abhängig ist. Taste 1-Doppelklick geht zum markierten Modellelement. Taste 2-Klick öffnet ein Popup-Menü mit den folgenden Einträgen.

- `Hinzufügen...`. Öffnet ein Dialogfenster, das es erlaubt, Abhängigkeiten von anderen Modellelementen zu erzeugen.

`Notwendig für`
Textbereich. Listet jedes Element auf, von dem diese Enumeration abhängt. Ein Taste 1-Doppelklick geht zum markierten Modellelement. Ein Taste 2-Klick öffnet ein Popup-Menü mit den folgenden Einträgen..

- `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`
Textbereich. Listet alle Operationen auf, die für diese Enumeration definiert sind. Ein Taste 1-Doppelklick geht zu der markierten Operation. Ein Taste 2-Klick öffnet ein Popup-Menü mit zwei Einträgen.

- `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

ArgoUML behandelt alle Operationen gleich. Jede, hier erzeugte Operation wird den gleichen Mechanismus wie Operationen für Klassen verwenden. Erinnern Sie sich, dass Operationen für Enumerationen keine Seiteneffekte haben dürfen (sie sind read-only). Das heißt, der Modifizierer `query` *muss* für alle Operationen markiert sein.

`Literale`

Textbereich. Listet alle Enumerations-Literale auf, die für diese Enumeration definiert sind. Ein Taste 1- Doppelklick geht zu dem markierten Literal, ein Taste 2- Klick öffnet ein Popup-Menü mit zwei Einträgen.

- `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

Stereotypen sind der Haupterweiterungsmechanismus von UML. Er enthält den Weg, um Spezialisierungen von den Standard-Metaklassen abzuleiten. `Stereotyp` ist eine Subklasse von `GeneralisierbaresElement` im UML-Metamodell. Stereotypen werden durch *Randbedingungen* und *Eigenschaftswerte* ergänzt.

Neue Stereotypen werden von fast jedem Modellelement über das Register Eigenschaften hinzugefügt. Eigenschaften existierender Stereotypen können durch Markieren im Explorer erreicht werden.

16.6.1. Detail-Register Stereotyp

Die folgenden Detail-Register sind für Stereotypen aktiv.

`Zu-Bearbeiten-Element`
Standard-Register.

`Eigenschaften`
Siehe Abschnitt 16.6.2, „ Eigenschaftssymbolleiste Stereotyp “ und Abschnitt 16.6.3, „ Eigenschaftsfelder für Stereotypen “ unten.

`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 Stereotypen an, dessen Eigenschaftswert entsprechend auf `derived` gesetzt wurde.

- dokumentation
- persistence
- persistent
- semantics
- usage

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.

`Löschen`

Entfernt diesen Stereotypen aus dem Modell.

16.6.3. Eigenschaftsfelder für Stereotypen

`Name`

Textfeld. Der Name des Stereotypen. Es gibt keine Konvention für das Benennen von Stereotypen, außer dass sie mit einem Kleinbuchstaben beginnen. Auch die Standard-UML-Stereotypen variieren zwischen alles in Kleinbuchstaben (z.B. `metamodel`), verbundeneZeichen (z.B. `systemModel`) und leerzeichenseparierte (z.B. `object model`).

Anmerkung

ArgoUML erzwingt keine Namenskonvention für Stereotypen.

`Namensraum`

Kombinationsfeld. Gibt den Namensraum für diesen Stereotypen wieder. Dies ist die Pakethierarchie.

`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*.

`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`

Kombinationsfeld. Jeder Stereotyp muss von einer der Metaklassen des UML-Metamodelles oder der Modellelementklassen abgeleitet werden, die von diesen abgeleitet sind. Der Stereotyp wird dann für die Modellelemente verfügbar sein, die von der gleichen Metaklasse oder von diesem Modellelement abgeleitet wurden.

16.7. Eigenschaftsdefinition

(Noch zu beschreiben)

16.8. Diagramm

Der UML-Standard spezifiziert acht prinzipielle Diagramme, alle werden von ArgoUML unterstützt.

- *Anwendungsfalldiagramm*. Wird für das Erfassen und Analysieren der Anforderungen für ein OOA&D- Projekt verwendet. Details zum ArgoUML-Anwendungsfalldiagramm und den von ihm unterstützten Modellelementen, siehe Kapitel 17, *Referenz der Modellelemente für Anwendungsfalldiagramme* .

- *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.

  Das ArgoUML-Klassendiagramm erlaubt die Unterstützung von Klassen- und Paketdiagrammen. Details über die unterstützten Modellelemente, siehe Kapitel 18, *Modellelement-Referenz Klassendiagramm* . Das Objektdiagramm wird durch das Verteilungsdiagramm unterstützt.

- *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.

  - *Zustandsdiagramm*. Wird verwendet, um das dynamische Verhalten eines einzelnen Objektes darzustellen (Klasseninstanz). Dieses Diagramm ist in Systemen, die komplexe Kommunikationsprotokolle verwenden, wie in der Telekommunikation, von besonderer Bedeutung. Details über das ArgoUML-Zustandsdiagramm und der von ihm unterstützten Modellelemente, siehe Kapitel 20, *Modellelement-Referenz Zustandsdiagramm* .

  - *Aktivitätsdiagramm*. Wird verwendet, um das dynamische Verhalten von Gruppen von Objekten (Klasseninstanz) darzustellen. Dieses Diagramm ist eine Alternative zum Zustandsdiagramm und passt besser bei Systemen mit umfangreicher Benutzer-Interaktion. Details über das ArgoUML-Aktivitätsdiagramm und der von ihm unterstützten Modellelemente, siehe Kapitel 22, *Modellelement-Referenz Aktivitätsdiagramm* .

  - *Interaktionsdiagramme*. Es gibt zwei Diagramme in dieser Kategorie, die dazu verwendet werden, die dynamische Interaktion zwischen Objekten im System (Klasseninstanzen)

darzustellen.

- *Sequenzdiagramm*. Zeigt die Interaktionen (typischerweise Nachrichten oder Prozeduraufrufe) zwischen Instanzen von Klassen (Objekten) und Akteuren über die Zeit. Besonders nützlich, wo die zeitlichen Beziehungen zwischen Interaktionen wichtig sind. Details über das ArgoUML-Sequenzdiagramm und den von ihm unterstützten Modellelementen, siehe Kapitel 19, *Modellelement-Referenz Sequenzdiagramm* .

- *Kollaborationsdiagramm*. Zeigt die Interaktionen (typischerweise Nachrichten oder Prozeduraufrufe) zwischen Instanzen von Klassen (Objekten) und Akteuren gegenüber den strukturellen Beziehungen zwischen diesen Instanzen. Besonders passend dort, wo es hilfreich ist, eine Beziehung zwischen Interaktionen und der statischen Struktur des Systems herstellen zu müssen. Details über das ArgoUML-Kollaborationsdiagramm und der von ihm unterstützten Modellelemente, siehe Kapitel 21, *Modellelement-Referenz Kollaborationsdiagramm* .

- *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.

  Das ArgoUML-Verteilungsdiagramm enthält die Unterstützung für das Komponenten- und das Verteilungsdiagramm und zusätzlich für die Objektdiagramme. Details über das Diagramm und die von ihm unterstützten Modellelemente, siehe Kapitel 23, *Modellelement-Referenz Verteilungsdiagramm* .

Diagramme werden mit Hilfe des `Neues Diagramm`- DropDown-Menüs (siehe Abschnitt 10.6, „ Das Menü "Neues Diagramm" “), oder mit den Werkzeugen in der Symbolleiste (siehe Abschnitt 9.4, „ Neues Diagramm “), oder mit den Popup-Menüs im Explorer erzeugt.

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), mit denen das Verhalten modelliert wird. Dies wird als Kontext des Verhaltensdiagrammes bezeichnet. Wenn diese Diagramme mit dem markierten Modellelement erzeugt werden, dann wird deren Kontext automatisch gesetzt. Ansonsten können Sie den Kontext manuell setzen.

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`
Siehe Abschnitt 16.8.3, „ Eigenschaftsfelder für Diagramme “ unten.

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`

Das Ausgangsmodell eines Diagrammes ist nicht in der UML- Spezifikation definiert. Das Ausgangsmodell ist das durch das Diagramm repräsentierte Modellelement. Von da an, hängt sein Typ vom Typ des Diagrammes ab: z.B. es ist der von einem Klassendiagramm repäsentierte Namensraum, oder der Zustandautomat im Falle eines Zustandsdiagrammes.

Kapitel 17. Referenz der Modellelemente für Anwendungsfalldiagramme

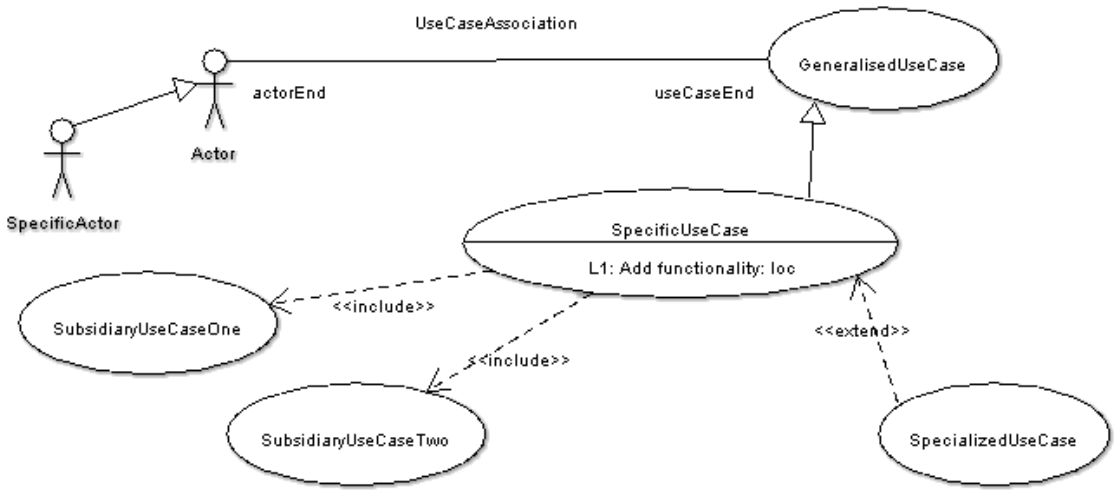
17.1. Einleitung

Dieses Kapitel beschreibt jedes Modellelement, das innerhalb eines Anwendungsfalldiagrammes erzeugt werden kann. Beachten Sie, dass einige Sub-Modellelemente der Modellelemente nicht immer im Diagramm 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`
: Siehe Abschnitt 17.2.2, „ Eigenschaftssymbolleiste Akteur “ und Abschnitt 17.2.3, „ Eigenschaftsfelder für einen Akteur “ unten.

`Dokumentation`
: Standard-Register. Siehe Abschnitt 13.4, „ Das Register Dokumentation “.

`Darstellung`
: Standard-Register. Die Füllfarbe wird für den Kopf des Strichmännchens verwendet.

`Quellcode`
: Standard-Register. Normalerweise ist für einen Akteur kein Code erforderlich, da er hinsichtlich des Systems extern ist.

`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 Eigenschaftwert 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`

Textfeld. Der Name des Akteurs. Das Diagramm zeigt diesen Namen unterhalb des Strichmännchens an. Da ein Akteur ein Classifier ist, entspräche es der Konvention den ersten Buchstaben als Großbuchstabe (und die ersten Buchstaben der zusammengesetzten Wörter) zu schreiben. Z.B. `RemoteSensor`.

Anmerkung

ArgoUML erzwingt keine Namenskonventionen für Akteure.

`Namensraum`

Textfeld mit Navigationsschaltfläche. Nimmt den Namensraum für einen Akteur auf. Dies ist die Pakethierarchie.

`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.

`Assoziationsenden`

Textbereich. Listet jedes Assoziationsende der mit diesem Akteur verbundenen Assoziationen auf.

Ein Taste 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-Metamodelles 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`
Standard-Register. Es wäre ungewöhnlich, wenn dieses Code für den Anwendungsfall enthielte, da der Anwendungsfall primär ein Vehikel für die Erfassung von Anforderungen über das zu entwickelnde System ist und keine Lösung erzeugt.

`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 Hauptaspekt 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 Modelles einen neuen Anwendungsfall (aber nicht innerhalb des Diagrammes) und zeigt unmittelbar das Register Eigenschaften dieses Anwendungsfalles.

Tipp

Diese Methode, einen Anwendungsfall zu erzeugen kann irritierend sein. Es ist viel besser, einen neuen Anwendungsfall im Diagramm Ihrer Wahl zu erzeugen.


`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.


`Neues Attribut`

Erzeugt innerhalb des aktuellen Anwendungsfalles ein neues Attribut und springt unmittelbar in das Register Eigenschaften dieses Attributes.


`Neue Operation`

Erzeugt innerhalb des aktuellen Anwendungsfalles eine neue Operation und springt unmittelbar in das Register Eigenschaften dieser Operation.


`Neuer Signaleingang`

Erzeugt innerhalb des aktuellen Anwendungsfalles einen neuen Signaleingang und springt unmittelbar in das Register Eigenschaften dieses Signaleinganges.


`Neuer Stereotyp`

Erzeugt innerhalb des aktuellen Anwendungsfalles einen neuen Stereotyp und springt unmittelbar in das Register Eigenschaften dieses Stereotyps.


`Löschen`

Entfernt den markierten Anwendungsfall aus dem Modell.

Warnung

Dies ist eine Löschung aus dem Modell, *nicht* nur aus dem Diagramm. Um einen Anwendungsfall aus einem Diagramm zu löschen, ihn aber im Modell zu erhalten, verwenden Sie das Hauptmenü `Aus Diagramm entfernen` (oder drücken Sie die Taste Entf).

17.3.3. Eigenschaftsfelder für einen Anwendungsfall

`Name`

Textfeld. Der Name des Anwendungsfalles. Da ein Anwendungsfall ein Klassifizierer ist, entspräche es der Konvention den ersten Buchstaben als Großbuchstabe (und die ersten Buchstaben der zusammengesetzten Wörter) zu schreiben. Z.B. `RemoteSensor`. Der Name wird innerhalb der ovalen Darstellung des Anwendungsfalle im Diagramm angezeigt.

Anmerkung

ArgoUML erzwingt keine Namenskonventionen für Anwendungsfälle.

`Namensraum`

Textfeld mit Navigationsschaltfläche. Nimmt den Namensraum des Anwendungsfalles auf. Dies ist die Pakethierarchie.

`Modifizierer`

Markierfeld mit den Einträgen `Abstract` `Leaf` und `Root`.

- `Abstract` wird verwendet, um zu deklarieren, dass der Anwendungsfall nicht instanziiert

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.

Ein Taste 2-Klick gibt ein Popup-Menü mit einem Eintrag `...hinzufügen` aus, das ein Dialogfenster öffnet, in dem Sie abhängige Modellelemente hinzufügen oder entfernen können.

`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`

Textbereich. Listet alle Anwendungsfälle auf, die Generalisierungen dieses Anwendungsfalles sind. Wird immer gesetzt, wenn eine Generalisierung von diesem Anwendungsfall erzeugt wurde. Ein Taste 1-Doppelklick auf eine Generalisierung wird einen Sprung zu dieser Generalisierung auslösen.

`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`
Textbereich. Listet alle für diesen Anwendungsfall definierten Attribute auf (siehe Abschnitt 18.7, „ Attribute “). Ein Taste 1- Doppelklick geht zu dem markierten Attribut. Ein Taste 2- Klick öffnet ein Popup-Menü mit zwei Einträgen, die ein Ändern der Attributreihenfolge erlauben.

- `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`
Textbereich. Listet alle Operationen auf (siehe Abschnitt 18.8, „ Operation “), die für diesen Anwendungsfall definiert wurden. Ein Taste 1-Klick springt zu der ausgewählten Operation. Taste 2 öffnet ein Popup-Menü mit zwei Einträgen, die es erlauben die Reihenfolge der Operationen zu ändern.

- `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 oben`. Nur verfügbar, wenn zwei oder mehr Erweiterungspunkte aufgelistet sind und der sich der ausgewählte Erweiterungspunkt nicht ganz oben befindet. Er bewegt den Erweiterungspunkt um eine Position nach oben.

- `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 Bezahlen 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.

Innerhalb des UML-Metamodelles ist der `Erweiterungspunkt` eine Subklasse des `Modellelementes`. Ein Anwendungsfall kann einen Erweiterungspunktbereich anzeigen (Details, siehe Abschnitt 17.3, „ Anwendungsfall “), in dem die Erweiterungspunkte in der folgenden Syntax dargestellt werden.

Name : *Ort* .

17.4.1. Detail-Register Erweiterungpunkt

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.

Anmerkung

Es ist nicht klar, wie abgeleitete Erweiterungspunkte einen Wert in der Analyse aufweisen können.

17.4.2. Eigenschaftssymbolleiste Erweiterungspunkt

 `Nach oben`

Springt nach oben zu dem Anwendungsfall, der Eigentümer dieses Erweiterungspunktes ist.

 `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

Textfeld. Eine Beschreibung des Erstellungsortes dieses Erweiterungspunktes innerhalb des Eigentümer-Anwendungsfalles.

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

Textfeld. Zeigt den Basis-Anwendungsfall in dem der Anwendungsfall definiert wurde. Ein Taste 1-Doppelklick springt zu dem Anwendungsfall.

Extend

Textfeld. Listet alle Anwendungsfälle auf, die den Basis- Anwendungsfall über diesen Erweiterungspunkt erweitern.

Dort, wo ein Erweiterungspunkte 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

Äbhängigkeiten haben in Anwendungsfalldiagrammen wenig Sinn. Sie sind Notwendig, weil frühere Versionen von ArgoUML sie nutzten (unkorrekterweise), um include- und extend-Beziehungen zu implementieren.

17.8. Generalisierung

Eine Generalisierung ist eine Beziehung zwischen zwei Anwendungsfällen oder zwei Akteuren. Wenn A eine Generalisierung von B ist, bedeutet das, dass A das generelle Verhalten beschreibt und B eine speziellere Version dieses Verhaltens darstellt.

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

Es geht sehr schnell, eine *extends*-Beziehung zwischen Anwendungsfällen mit einer Generalisierung zu verwechseln. Jedoch extends erweitert das Verhalten eines Anwendungsfalles in einem speziellen Punkt. Während eine Generalisierung das Verhalten über den ganzen Anwendungsfall spezialisiert.

Innerhalb des UML-Metamodelles 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`
Siehe Abschnitt 17.8.2, „ Eigenschaftssymbolleiste Generalisierung “ und Abschnitt 17.8.3, „ Eigenschaftsfelder der Generalisierung “ folgende.

`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.

`Löschen`

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

Der leere String „“ ist ein gültiger Eintrag (und der Standard) für dieses Feld. Der Diskriminator ist nur im Falle einer Mehrfachvererbung von praktischem Nutzen. Ein (Klassendiagramm) Beispiel wird in Abbildung 17.2, „ Beispiel eines Diskrimintators mit Generalisierung “ gezeigt. Hier sollte jeder Anwendertyp von zwei Anwenderarten erben. Eine, die zwischen lokalen und fernen Anwendern unterscheidet (die durch einen Diskriminator identifiziert werden können) und eine, welche die Funktion eines Anwenders angibt (durch einen anderen Diskriminator bezeichnet).

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`

Textfeld. Zeigt den Anwendungsfall oder Akteur, der die *Superklasse* dieser Beziehung darstellt. Z.B. das generellere Ende der Beziehung. Ein Taste 1- Doppelklick auf diesen Eintrag springt zu diesem Anwendungsfall oder Akteur.

`Subklasse`

Textfeld. Zeigt den Anwendungsfall oder Akteur, der die *Subklasse* dieser Beziehung darstellt. Z.B. das spezifischere Ende der Beziehung. Ein Taste 1- Doppelklick auf diesen Eintrag springt zu diesem Anwendungsfall oder Akteur.

`Powertype`

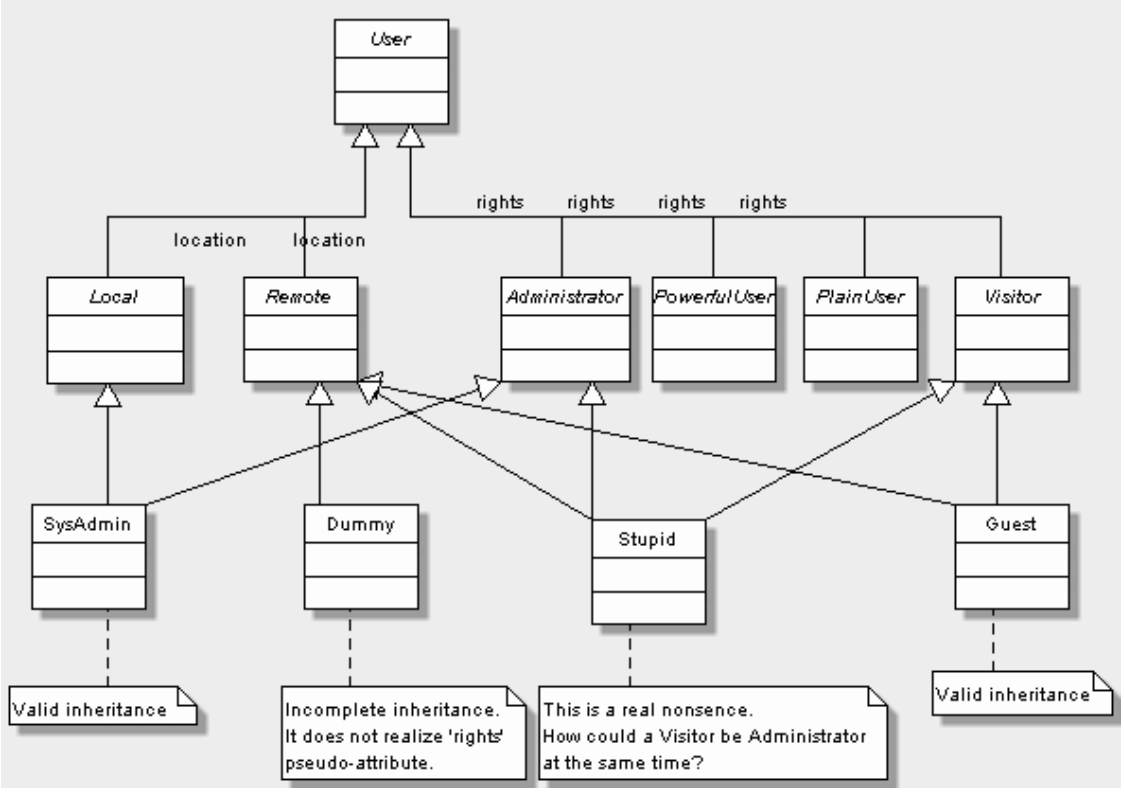
Drop-Down-Auswahl mit Zugriff auf alle Standard-UML-Typen, die durch ArgoUML unterstützt werden und allen neuen Klassen, die innerhalb des aktuellen Modelles erzeugt wurden.

Dies ist der Typ der Subklasse der Generalisierung.

Tipp

Dies kann in der Anwendungsfallanalyse ignoriert werden. Der einzige, einzugebende sensible Wert würde der Typ des Subklassen-Anwendungsfalles sein (wie ein Klassifizierer, der in der Drop-Down-Auswahlliste erscheint).

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-Metamodelles 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 Anwendungsfalldiagramm. “).

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`
Siehe Abschnitt 17.9.2, „ Extend Eigenschaftssymbolleiste “ und Abschnitt 17.9.3, „ Eigenschaftsfelder für Extend “ unten.

`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`

Textfeld. Nimmt den Namensraum für die Extend-Beziehung auf. Dies entspricht der Pakethierarchie.

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`

Textfeld. Zeigt den Anwendungsfall, der durch diese Extend- Beziehung erweitert wird. Ein Taste 1-Doppelklick auf diesen Eintrag navigiert zu diesem Basis-Anwendungsfall.

`Erweiterung`

Textfeld. Zeigt den Anwendungsfall, der die Erweiterung über diese Extend-Beziehung darstellt. Ein Taste 1- Doppelklick auf diesen Eintrag navigiert zu diesem Erweiterungs-Anwendungsfall.

`Erweiterungspunkte`

Textfeld. Listet die Erweiterungspunkte des Basis- Anwendungsfalles auf, bei denen die Erweiterung ausgeführt wird, wenn die Bedingung eintritt.

Anmerkung

Wenn die Bedingung erfüllt ist, wird die Sequenz der Anwendungsfall-Instanz durch das Einbinden der Sequenz des erweiternden Anwendungsfalles erweitert. Die unterschiedlichen Teile des erweiternden Anwendungsfalles werden an den durch die Abfolge der Erweiterungspunkte in der Beziehung bestimmten Stellen eingefügt - ein Teil an jedem referenzierten Erweiterungspunkt. Beachten Sie, dass die Bedingung nur einmal geprüft wird: nämlich am ersten referenzierten Erweiterungspunkt. Wenn sie erfüllt ist, werden alle erweiternden Anwendungsfälle in die Original-Sequenz eingefügt.

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`
Textbereich. Mehrzeilige textuelle Beschreibung jeder Bedingung der Extend-Beziehung.

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 beinhaltet.

Innerhalb des UML-Metamodelles 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 Incldue-Beziehungen können ihren Wert in der Analyse haben, um nützliche Namen oder Konzepte einzubinden.

17.10.2. Include Eigenschaftssymbolleiste

 `Nach oben`

Navigiert in der Paketstruktur des Modelles 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 Stereotyps.

 `Lösche`

Entfernt die markierte Include-Beziehung aus dem Modell.

Warnung

Diese Funktion löscht aus dem Modell und *nicht* nur aus dem Diagramm. Um eine Include-Beziehung aus dem Diagramm zu löschen, sie aber im Modell zu erhalten, verwenden Sie den Eintrag `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`

Kombinationsfeld. Gibt den Anwendungsfall wieder, der den anderen Anwendungsfall in dieser Include-Beziehung aufnimmt. Ein Taste 1-Klick auf diesen Eintrag öffnet ein DropDown-Menü aller verfügbaren Anwendungsfälle, die durch einen Taste 1-Klick ausgewählt werden können.

`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 18.1. Denkbare Modellelemente in einem Klassendiagramm.

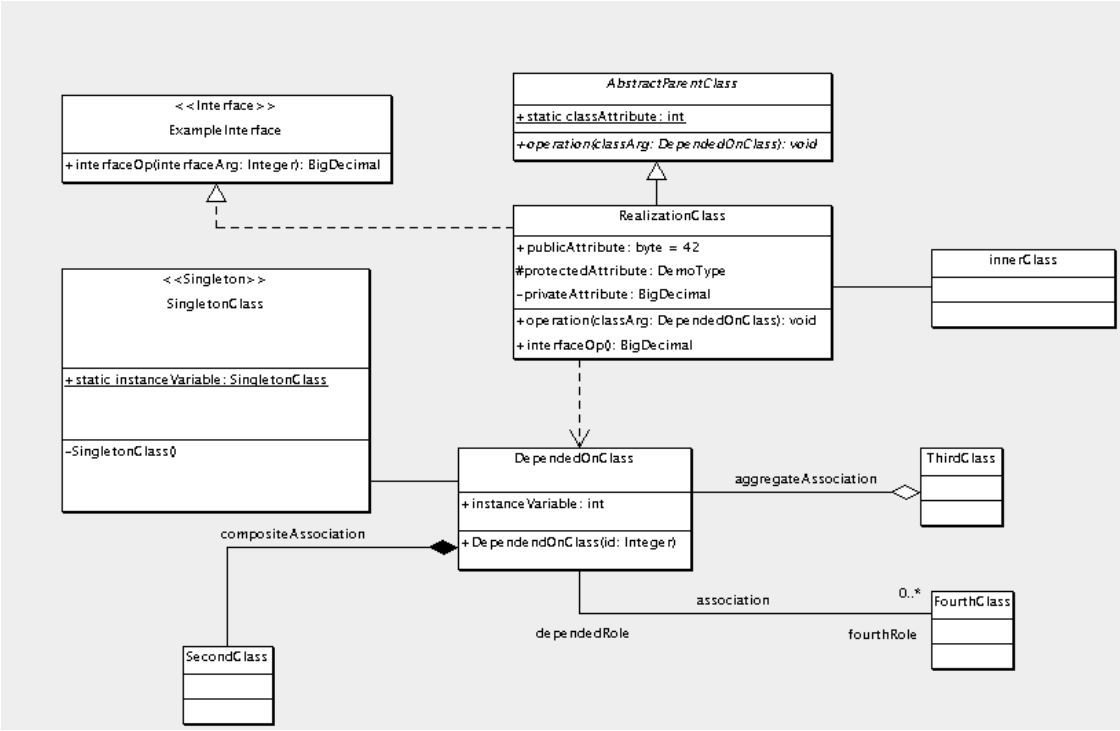


Abbildung 18.2, „ Denkbare Modellelemente in einem Paketdiagramm. “ zeigt ein Paketdiagramm mit

allen denkbaren Modellelementen.

Abbildung 18.2. Denkbare Modellelemente in einem Paketdiagramm.

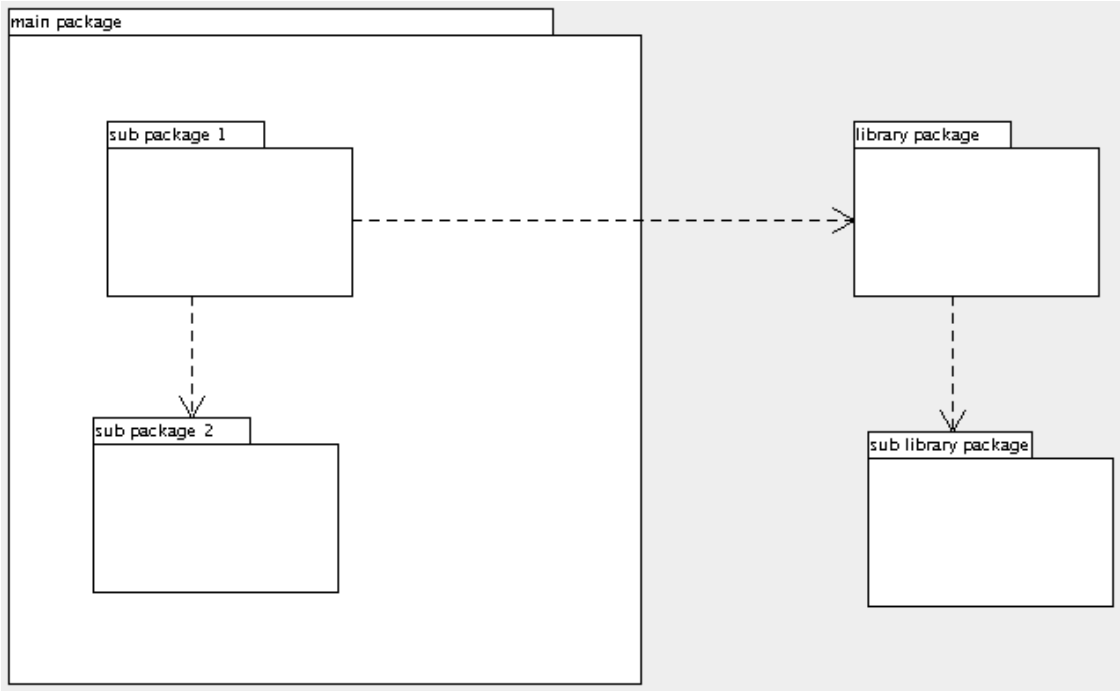


Abbildung 18.3, „ Denkbare Modellelemente in einem Datentypdiagramm. “ zeigt ein Datentyp-Diagramm mit einem Datentyp und einer Aufzählung.

Abbildung 18.3. Denkbare Modellelemente in einem Datentypdiagramm.

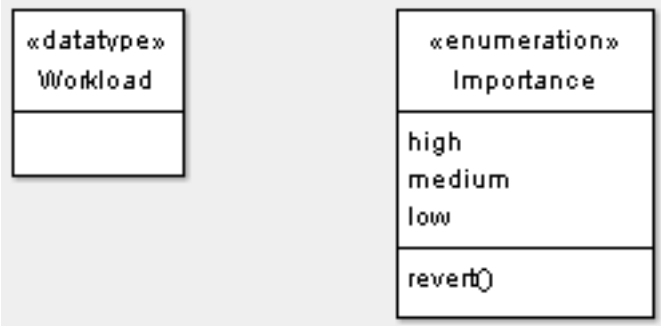
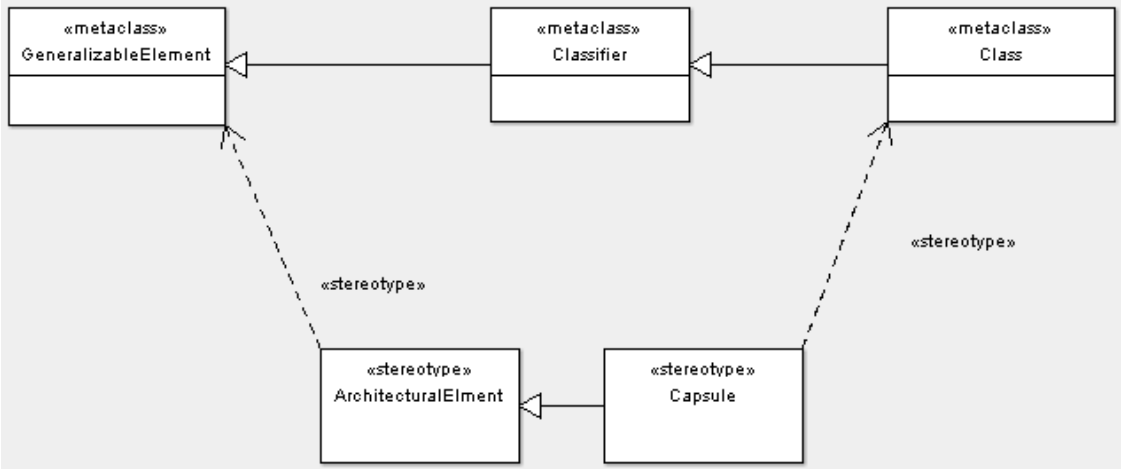


Abbildung 18.4, „ Denkbare Modellelemente in einem Stereotyp-Definitionsdiagramm. “ zeigt ein Stereotyp- Definitionsdiagramm mit allen denkbaren Modellelementen.

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.

Eine andere Variante des Klassendiagrammes ist innerhalb des UML- Standards das *Ojektdiagramm*. Es gibt aktuell keine Unterstützung für Objekte oder Verknüpfungen innerhalb von ArgoUML-Klassendiagrammen. Dafür hat das ArgoUML-Verteilungsdiagramm beides, Objekte und Verknüpfungen. Es kann für das Zeichnen von Objektdiagrammen verwendet werden.

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`
: Siehe Abschnitt 18.2.2, „ Paket Eigenschaftssymbolleiste “ und Abschnitt 18.2.3, „ Eigenschaftsfelder für ein Paket “ unten.

`Dokumentation`
Standard-Register. Siehe Abschnitt 13.4, „ Das Register Dokumentation “.

`Darstellung`
Standard-Register. Das editierbare Feld `Begrenzung` definiert die Größe des Rahmens des Paketes im Diagramm.

`Stereotyp`
Standard-Register.

`Eigenschaftswerte`
Standard-Register. Im UML-Metamodell hat das `Paket` die folgenden definierten Eigenschaftswerte.

- `derived` (von der Superklasse `ModelElement`). Der Wert `true` bedeutet, dass das Paket redundant ist - es kann formal von anderen Elementen abgeleitet werden. Oder, `false` bedeutet, dass es nicht von anderen Elementen abgeleitet werden kann.

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`

Erzeugt einen neuen Datentyp (siehe Abschnitt 16.3, „ Datatyp “) in dem markierten Paket und springt sofort in das Register Eigenschaften dieses Datentyps.


`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 Modelles 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`

Kombinationsfeld. Gibt den Namensraum des Paketes wieder. Dies entspricht der Pakethierarchie.

`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 instanziiert 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 Modelles befinden. Dies kann dazu verwendet werden, um zu betonen, dass sich das Modell auf oberste Ebene befindet.

`Generealisierung`

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`

Textbereich. Eine Liste aller Pakete, Klassen, Schnittstellen, Datentypen, Akteure, Anwendungsfälle, Assoziationen, Generalisierungen, Stereotypen usw. des Paketes.

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.

Ein Taste 1-Doppelklick auf eines dieser aufgelisteten Elemente navigiert zu diesem Modellelement. Ein Taste 2- Klick öffnet ein Popup-Menü mit den folgenden Einträgen.

- `Hinzufügen`. Das Fenster „ Hinzufügen/Entfernen importierter Elemente“ öffnet sich. In diesem Fenster ist es möglich, eine Liste von importierten Elementen zu erstellen.
- `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 Bereich können optional versteckt werden.

18.6.1. Detail-Register Klasse

Die aktiven Detail-Register für Klassen sind folgende.

`Zu-Bearbeiten-Element`
: Standard-Register.

`Eigenschaften`
: Siehe Abschnitt 18.6.2, „ Eigenschaftssymbolleiste Klasse “ und Abschnitt 18.6.3, „ Eigenschaftsfelder für eine Klasse “ unten.

`Dokumentation`
: Standard-Register. Siehe Abschnitt 13.4, „ Das Register Dokumentation “.

`Darstellung`
: Standard-Register. Die Markierfelder `Attribute` und `Operationen` erlauben es, die Attribut- und Operations-Bereiche anzuzeigen (der Standard) oder auszublenden. Dies ist eine Einstellung, die nur für das aktuelle Diagramm gültig ist, das die Klasse darstellt. Die editierbaren Felder `Begrenzung` definieren die Größe des Paketrahmens im Diagramm.

`Quellcode`
: Standard-Register. Enthält ein Template für die Klassendeklaration und Deklarationen der verküpften Klassen.

`Randbedingungen`
: Standard-Register. Es sind keine Standard-Randbedingungen für `Klassen` innerhalb des UML-Metamodelles definiert.

`Stereotypen`
: Standard-Register.

`Eigenschaftswerte`

: Standard-Register. Im UML-Metamodell hat die `Klasse` die folgenden Standard-Eigenschaftswerte definiert.

- `persistence` (von der Superklasse s`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`

Erzeugt innerhalb der Klasse eine neue innere Klasse (die in keinem Diagramm erscheint). Sie gehört zur Klasse und ist auf den Namensraum dieser Klasse beschränkt. Sie modelliert exakt das Java-Konzept einer inneren Klasse. Als innere Klasse benötigt sie keine Attribute oder Operationen, weil sie diese mit ihrem Eigentümer teilt.

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 entferen` 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-Hinweise werden sich über Klassennamen beschweren, die nicht mit einem Großbuchstaben beginnen.

`Namensraum`

Kombinationsfeld. Gibt den Namensraum einer Klasse wieder und erlaubt dessen Erstellung. Die ist die Pakethierarchie.

Ein Taste 1-Klick auf den Eintrag verschiebt die Klasse in den markierten Namensraum.

`Modifizierer`

Markierfelder mit den Einträgen `Abstract`, `Leaf`, `Root` und `Active`.

- `Abstract` wird verwendet, um zu deklarieren, dass diese Klasse nicht instanziiert 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 Klasser außerhalb des Namensraumes sichtbar ist.

`Abhängig von`
Textbereich. Listet alle „abhängigen“ Enden der Beziehung auf. Z.B. das Ende, welches Gebrauch vom anderen Ende macht.

Ein Taste 1-Doppelklick navigiert zu der Abhängigkeit und öffnet dessen Register Eigenschaften.

Ein Taste 2-Klick zeigt ein Popup-Menü mit einem Eintrag `Hinzufügen...`. Dieser Eintrag öffnet ein Dialogfenster, in dem Sie abhängige Modellelemente hinzufügen oder entfernen können.

`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.

Ein Taste 2-Klick zeigt ein Popup-Menü mit dem Eintrag `Hinzufügen...`. Dieser Eintrag öffnet ein Dialogfenster in dem Sie abhängige Modellelemente hinzufügen oder entfernen können.

`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`
Textbereich. Listet alle Attribute auf (siehe Abschnitt 18.7, „ Attribute “), die für diese Klasse definiert sind. Ein Taste 1-Doppelklick navigiert zu dem markierten Attribut. Ein Taste 2-Klick öffnet ein Popup-Menü mit zwei Einträgen, die eine Neuanordnung der Attribute erlauben.

- `Nach oben`. Nur verfügbar, wenn mehr als zwei Attribute gelistet sind und sich das markierte Attribut nicht ganz oben befindet. Bewegt das Attribut um eine Position nach oben.

- `Nach unten`. Nur verfügbar, wenn mehr als zwei Attribute gelistet sind und sich das markierte Attribut nicht ganz unten befindet. Bewegt das Attribut um eine Position nach unten.

`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`

Textbereich. Listet alle Operationen auf (siehe Abschnitt 18.8, „ Operation “), die für diese Klasse definiert sind. Ein Taste 1-Klick navigiert zu der markierten Operation. Ein Taste 2-Klick öffnet ein Popup-Menü mit zwei Einträgen, die eine Neuanordnung der Operationen erlauben.

- `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`
Textbereich. Eine Liste von Modellelementen, die sich im Namensraum der Klasse befinden. Hier wird jede innere Klasse (siehe Abschnitt 18.6.2, „ Eigenschaftssymbolleiste Klasse “) erscheinen.

Ein Taste 1-Doppelklick auf eines der Modellelemente navigiert zu diesem Modellelement.

Tipp

Die meisten Namensraumhierarchien sollten über den Paketmechanismus verwaltet werden. Namensraumhierarchien über Klassen sind am besten auf innere Klassen zu beschränken. Denkbare Datentypen, Signale und Schnittstellen können hier auch erscheinen, aber Akteure und Anwendungsfälle scheinen hier wertlos.

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:

Sichtbarkeit Attributname : *Typ* [= *Anfangswert*]

Die *Sichtbarkeit* ist +, #, - oder ~ entsprechend den Werten `public`, `protected`, `private`, oder `package`.

Attributname ist der aktuelle Name des deklarierten Attributes.

Typ ist der für das Attribut deklarierte Typ (UML-Datentyp, Klasse oder Schnittstelle).

Anfangswert ist der Anfangswert, der dem Attribut zugewiesen wird, wenn die Klasseninstanz erzeugt wird. Dieser kann von einer Konstruktoroperation überschrieben werden.

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`
Siehe Abschnitt 18.7.2, „ Eigenschaftssymbolleiste Attribut “ und Abschnitt 18.7.3, „ Eigenschaftsfelder für Attribute “ unten.

`Dokumentation`
Standard-Register. Siehe Abschnitt 13.4, „ Das Register Dokumentation “.

`Randbedingungen`
Standard-Register. Es sind im UML-Metamodell keine Standard-Randbedingungen für `Attribute` definiert.

`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

Dies ist ein Löschen aus dem Modell, *nicht* nur aus dem Diagramm. Wenn gewünscht, kann der ganze Attributbereich im Diagramm mit Hilfe des Registers *Darstellung* (siehe Abschnitt 18.7.2, „ Eigenschaftssymbolleiste Attribut “), oder mit Hilfe des Taste 2-Popup-Menüs für Klassen ausgeblendet werden.

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-Hinweise 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`.

- `Nur hinzufügen`. Nur dann von Bedeutung, wenn die Kardinalität nicht auf einen einzigen Wert festgelegt ist. Zusätzliche Werte können dem Satz von Werten hinzugefügt werden. Wenn der Wert einmal erzeugt wurde, darf er nicht entfernt oder geändert werden.
- `veränderlich`. Es gibt keine Einschränkungen hinsichtlich der Veränderbarkeit.
- `unveränderlich`. Auch „ eingefroren“ genannt. Der Wert des Attributes darf während der Lebensdauer der Klasse nicht geändert werden. Der Wert muss bei der Objekterzeugung gesetzt werden und darf danach niemals geändert werden. Dies hat zur Folge, dass es gewöhnlich ein Argument für diesen Wert in einem Konstruktor gibt und dass es keine Operation gibt, die diesen Wert aktualisiert.

`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`

Kombinationsfeld. Der Typ dieses Attributes. Dies kann jeder UML-`Klassifizierer` sein, obwohl in der Praxis nur `Klasse`, `Datentyp` oder `Schnittstelle` Sinn machen.

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:

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.

Ein *Stereotyp* kann zwischen zwei Elementen einer Zeile im Format: `<<stereotype>>` eingegeben werden.

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`
Siehe Abschnitt 18.8.2, „ Eigenschaftssymbolleiste Operation “ und Abschnitt 18.8.3, „ Eigenschaftsfelder für Operation “ unten.

`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`
Standard-Register. Es sind innerhalb des UML-Metamodelles keine Standard-Randbedingungen für `Operation` definiert.

`Eigenschaftswerte`

Standard-Register. Im UML-Metamodell sind für `Operation` die folgenden Standard-Eigenschaftswerte definiert.

- `semantics`. Der Wert ist eine Spezifikation der Semantik der Operation.
- `derived` (von der Superklasse `ModelElement`). Der Wert `true` bedeutet, dass die Operation redundant ist - sie kann formal von anderen Elementen abgeleitet werden. Oder `false` , wenn sie nicht abgeleitet werden kann.

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`

Erzeugt im Namensraum des Eigentümers dieser Operation einen neuen Datentyp (siehe Abschnitt 16.3, „ Datatyp “) und springt sofort in das Register Eigenschaften dieses Datentyps.


`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

Dies ist eine Löschung aus dem Modell und *nicht* nur aus dem Diagramm. Wenn gewünscht, kann der ganze Operationen-Bereich der Klasse im Diagramm mit Hilfe des Registers *Darstellung* (siehe Abschnitt 18.8.2, „ Eigenschaftssymbolleiste Operation “) oder dem Taste 2-Popup-Menü versteckt werden.

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-Hinweise 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 diesen Hinweis zum Schweigen, indem Sie den Stereotypen `create` für die Konstruktor-Operation vergeben.

`Stereotyp`

Kombinationsfeld. Es gibt zwei UML-Standard-Stereotypen für eine `Operation` (von der Eltern-Metaklasse `BehavioralFeature`) `create` und `destroy`.

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 zugehörgien 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 geblockt, 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`. Mehrer 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.

Ein Taste 1-Doppelklick auf einen dieser Parameter navigiert zu diesem Parameter. Ein Taste 2-Klick öffnet ein Popup-Menü mit zwei Einträgen.

- `Nach oben`. Nur verfügbar, wenn es mehr als zwei Parameter gibt und sich der markierte Parameter nicht ganz oben befindet. Er wird um eine Position nach oben bewegt.
- `Nach unten`. Nur verfügbar, wenn es mehr als zwei Parameter gibt und sich der markierte Parameter nicht ganz unten befindet. Er wird um eine Position nach unten bewegt.

`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`
Siehe Abschnitt 18.9.2, „ Eigenschaftssymbolleiste Parameter “ und Abschnitt 18.9.3, „ Eigenschaftsfelder für Parameter “ unten.

`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

Die UML-Metaklasse `Element`, von dem alle anderen Modellelemente abgeleitet werden, beinhaltet den Eigenschaftswert `documentation`. Dieser wird in ArgoUML durch das Register *Dokumentation* verwaltet.

18.9.2. Eigenschaftssymbolleiste Parameter


`Nach oben`

Navigiert in der Paketstruktur nach obene.


`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`

Erzeugt im Namensraum des Eigentümers der Operation dieses Parameters einen neuen Datentyp (siehe Abschnitt 16.3, „ Datatyp “) und springt sofort in das Register Eigenschaften dieses Datentyps.

 `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

Dies ist eine Löschung aus dem Modell und *nicht* nur aus dem Diagramm. Wenn gewünscht, kann der ganze Operationenbereich im Diagramm mit Hilfe des Registers `Darstellung` oder mit dem Taste 2- Popup-Menü der Klasse versteckt werden.

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-Hinweise 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`

Auswahlfeld. Der Typ dieses Parameters. Dies kann jeder UML-`Klassifizierer` sein, obwohl in der Praxis nur `Klasse`, `Datentyp` , oder `Schnittstelle` Sinn machen.

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 gewüscht, 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 asychrone Verhalten der Zustandsautomaten und

wenn Sie mit *Sende-Aktionen* in Zustandsautomaten und Nachrichten in Kollaborationsdiagrammen verknüpft werden.

Tipp

Generell hat es aktuell einen begrenzten Wert, Signale in ArgoUML zu definieren. Es kann sich als nützlich erweisen, Signale als Klassen mit einem (benutzerdefinierten) Stereotypen `«signal»` zu definieren, wie im UML 1.4-Standard empfohlen. Dies erlaubt eine Abhängigkeitsbeziehung zwischen den angezeigten Signalen.

18.10.1. Detail-Register Signal

Die aktiven Detail-Register für Signale sind die folgenden.

`Zu-Bearbeiten-Element`
: Standard-Register.

`Eigenschaften`
: Siehe Abschnitt 18.10.2, „ Eigenschaftssymbolleiste Signal “ und Abschnitt 18.10.3, „ Eigenschaftsfelder für ein Signal “ unten.

`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.
- `derived` (von der Superklasse `ModelElement`). Der Wert `true` bedeutet, dass das Signal redundant ist - es kann formal von anderen Elementen abgeleitet werden, oder `false`, wenn es nicht abgeleitet werden kann.

Anmerkung

Abgeleitete Signale haben Ihren Wert in der Analyse, um nützliche Namen oder Konzepte einzubinden und im Design, um eine Wiederverarbeitung zu vermeiden.

Anmerkung

Die UML-Metaklasse `Element`, von der alle anderen Modellelemente abgeleitet werden, beinhaltet einen Eigenschaftswert `documentation`. Dieser wird in ArgoUML im Register `Dokumentation` bearbeitet.

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`

Löscht 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-Hinweise 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 “).

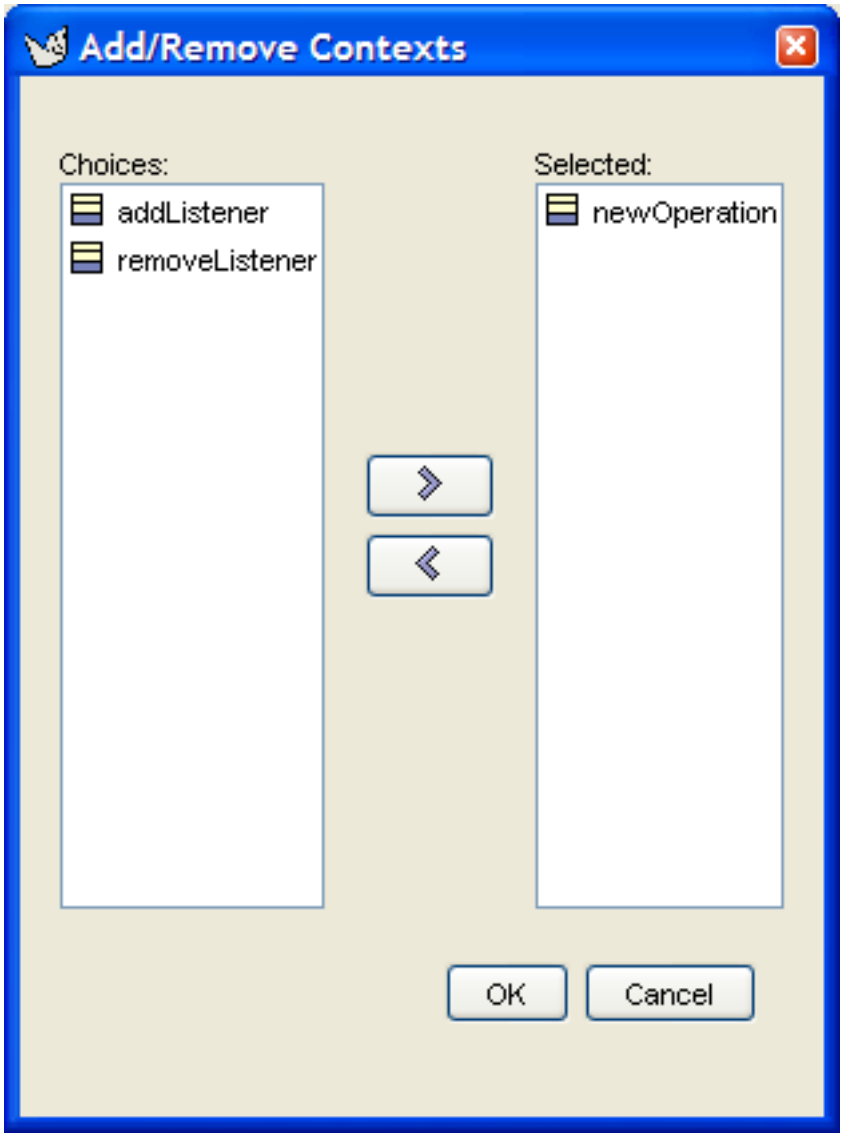
`Namensraum`
Auswahlfeld. Gibt den Namensraum für ein Signal wieder und erlaubt dessen Änderung. Dies ist die Pakethierarchie des Signals.

`Kontexte`

Textbereich. Listet alle Kontexte auf, die für dieses Signal definiert sind. Ein Taste 1-Doppelklick navigiert zu dem markierten Kontext, ein Taste 2-Klick öffnet ein Popup-Menü mit einem Eintrag.

- `Hinzufügen`. Für einen neuen Kontext hinzu. Dies öffnet das Dialogfenster *Kontexte hinzufügen/entfernen*, welches Ihnen erlaubt, zwischen den möglichen Operationen auszuwählen und diese der ausgewählten Liste hinzuzufügen.

Abbildung 18.5. Das Dialogfenster „Kontext hinzufügen/entfernen“



18.11. Empfangssignal (noch zu beschreiben)

Ein Empfangssignal ist ...

18.12. Assoziation

Eine Assoziation in einem Klassendiagramm stellt die Beziehung zwischen Klassen, oder zwischen einer Klasse und einer Schnittstelle dar. In einem Anwendungsfalldiagramm bindet eine Assoziation einen Akteur an einen Anwendungsfall.

Innerhalb des UML-Metamodelles 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

UML 1.3 enthält N-fach Assoziationen und Assoziationen, die durch eine dritte *assoziative Klasse* verwaltet werden. Beide werden von ArgoUML unterstützt.

N-fach Assoziationen werden erzeugt durch zeichnen einer Assoziation von einer existierenden Assoziation zu einer dritten Klasse. Die aktuelle Implementierung von ArgoUML erlaubt nicht die umgkehrte Vorgehensweise: Das Zeichnen von der 3. Klasse zu einer existierenden Assoziation ist nicht möglich.

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`
Siehe Abschnitt 18.12.3, „ Eigenschaftssymbolleiste Assoziation “ und Abschnitt 18.12.4, „ Eigenschaftsfelder für eine Assoziation “ unten.

`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`
Standard-Register. Sie würden nicht erwarten, dass Code für eine Assoziation generiert wird. Jeder hier eingegebene Code wird ignoriert (er wird verschwunden sein, wenn sie zu dieser Assoziation zurückkehren).

`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

Die UML-Metaklasse `Element`, von der alle anderen Modellelemente abgeleitet werden, beinhaltet einen Eigenschaftswert `documentation`. Dieser wird in ArgoUML im Register `Dokumentation` bearbeitet.

18.12.3. Eigenschaftssymbolleiste Assoziation

 `Nach oben`

Navigiert in der Paketstruktur des Modelles nach oben. Bei einer Assoziation ist dies das Paket, welches die Assoziation enthält.

 `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`

Löscht 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`

Textfeld. Der Name der Assoziation. Per Konvention beginnen Assoziationsnamen mit einem Kleinbuchstaben und mit Groß-/ Kleinschreibung, um Wörter innerhalb des Namens unterscheiden zu können. Beispiel: `verkaufsVorgehen` .

Anmerkung

ArgoUML erzwingt keine Namenskonvention für Assoziationen.

Tipp

Obwohl die Designhinweise dies fordern, ist es auf der anderen Seite normal, die Assoziationen in einem Klassendiagramm nicht zu benennen, da die Beziehung der Klassennamen (oder 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 (Anforderungsaufnahme) 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`

Textbereich. Listet die Enden dieser Assoziation auf. Eine Assoziation kann zwei oder mehr Enden haben. Bei mehr als zwei Assoziationsenden siehe Abschnitt 18.13, „ Assoziationsende “.

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 Anwendungsfallmodelles 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`
Siehe Abschnitt 18.13.2, „ Eigenschaftssymbolleiste Assoziationsende “ und Abschnitt 18.13.3, „ Eigenschaftsfelder für ein Assoziationsende “ unten.

`Dokumentation`
Standard-Register. Siehe Abschnitt 13.4, „ Das Register Dokumentation “.

`Darstellung`
Standard-Register.

`Quellcode`
Standard-Register. Dieses Register enthält eine Deklaration für ein Assoziationsende als Instanz des Modellelementes, mit der es verbunden ist.

`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

Diese Schaltfläche ist bei binären Assoziationen deaktiviert, weil eine Assoziation mindestens *zwei* Enden benötigt. Nur für N-fache Assoziationen ist diese Schaltfläche zugreifbar und löscht nur ein Ende aus der Assoziation.

18.13.3. Eigenschaftsfelder für ein Assoziationsende

`Name`

Textfeld. Der Name des Assoziationsendes, das einen *Rollenname* für dieses Ende der Assoziation enthält. Dieser Rollenname kann für die Navigation und im Kontext der Implementation, enthält es einen Namen, durch den das Quell-Ende einer Assoziation durch das Ziel-Ende referenziert wird.

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`

Textfeld. Gibt die Eltern-Assoziation für dieses Assoziationsende wieder. Ein Taste 1-Doppelklick auf diesen Eintrag navigiert zu der Assoziation.

`Typ`

Das Kombinationsfeld bietet Zugriff auf alle Standard-UML- Typen von ArgoUML und alle neuen Klasse an, die innerhalb des aktuelle Modelles erezgut wurden.

Dies ist der zu diesem Ende der Assoziation hinzugefügte Typ der Entität.

Tipp

Standardmäßig wird ArgoUML die Klasse des Modellelementes markieren, mit der die Verknüpfung verbunden ist. Jedoch kann eine Assoziation zu einer anderen Klasse durch markieren des anderen Eintrages bewegt werden. Modellelementen

`Kardinalität`

DropDown-Menü mit Editierfenster. Der Wert kann aus dem Auswahlfenster ausgewählt werden oder eine neue kann im Textfeld editiert werden. Nimmt die Kardinalität dieses Assoziationsendes wieder (unter Berücksichtigung des anderen Endes). Z.B. wie viele Instanzen dieses Endes können mit einer Instanz des anderen Endes verknüpft werden. Die Kardinalität wird im Diagramm am Ende der Assoziation angezeigt.

`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 Verknpfungen selbst nicht innewohnen.

Einstellungen für die Markierfelder sind: Nicht markiert - Die Verweise bilden einen Satz ohne innewohnende Reihenfolge. Markiert - Ein Satz geordneter Verweise in einer Reihenfolge.

- `Static` (Noch zu beschreiben)

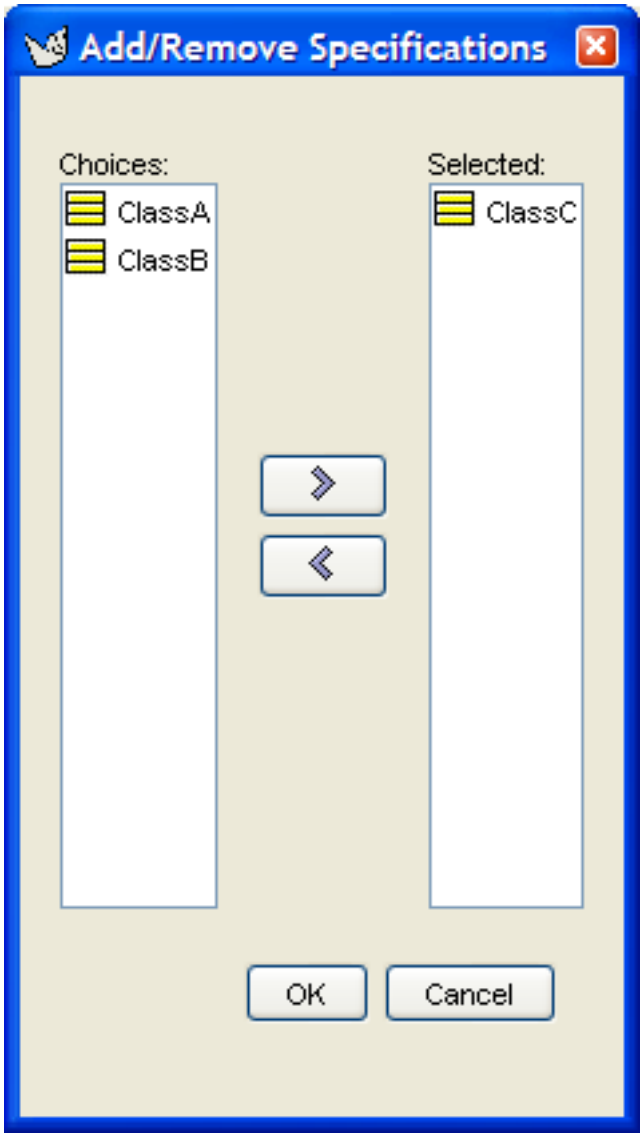
`Spezifikation`

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 realisert 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.

Ein Taste 1-Doppelklick navigiert zu dem markierten Klassifizierer. Ein Taste 2-Klick öffnet ein Popup-Menü mit einem Eintrag.

- `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`

Textfeld. Nimmt die Qualifizierer für dieses Assoziationsende auf. Ein Taste 1-Doppelklick auf diesen Eintrag navigiert zu diesem Qualifizierer. Ein Taste 2-Klick öffnet ein Popup-Menü, das zwei Einträge enthält: `Nach oben` und `Nach unten` , die ein Ändern der Reihenfolge der Qualifizierer ermöglichen.

`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.

18.14. Abhängigkeit

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`.

Ein Abhängigkeit wird durch eine gestrichelte Line mit einem offenen Pfeil vom abhängigen Modellelement zu dem Modellelement von dem es abhängt dargestellt.

18.14.1. Detail-Register Abhängigkeit

Die akviten Detail-Register für Abhängigkeiten sind die folgenden.

`Zu-Bearbeiten-Element`
Standard-Register.

`Eigenschaften`
Siehe Abschnitt 18.14.2, „ Eigenschaftssymbolleiste Abhängigkeit “ und Abschnitt 18.14.3, „ Eigenschaftsfelder für eine Abhängigkeit “ unten.

`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 Modelles 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`

Kombinationsfeld. Die Abhängigkeit hat unter UML 1.3 keine Standard-Stereotypen und daher bietet ArgoUML auch keine an. Der Stereotyp wird zwischen « und » über oder quer zur Generalisierung angezeigt.

`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 Schnittstellennamen 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`
: Siehe Abschnitt 18.16.2, „ Eigenschaftssymbolleiste Schnittstelle “ und Abschnitt 18.16.3, „ Eigenschaftsfelder einer Schnittstelle “ unten.

`Dokumentation`
: Standard-Register. Siehe Abschnitt 13.4, „ Das Register Dokumentation “.

`Darstellung`
: Standard-Register. Das Markierfeld `Operationen` erlaubt es, den Operationenbereich anzuzeigen (der Standard) oder zu verstecken. Diese Einstellung ist nur für das aktuelle Diagramm gültig. Das Feld `Begrenzung` definiert die Grenzen der Schnittstelle im Diagramm.

`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 Stereotyps.

`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-Hinweise 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`
Auswahlfeld mit den drei Einträgen `public`, `protected`, `private` und `package`. Gibt an, ob die Navigation zu diesem Ende sein darf durch: i) jeden Klassifizierer; ii) nur den Quellklassifizierer und dessen Kinder; oder iii) nur durch den Quellklassifizierer.

`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

dieser Schnittstelle verbunden sind.

Anmerkung

Assoziationen zwischen Klassen und Schnittstellen *dürfen nur* von der Klasse zur Schnittstelle navigierbar sein. ArgoUML wird Assoziationen zwischen Klassen und Schnittstellen mit der korrekten Navigierbarkeit erzeugen, verhindert aber nicht, wenn der Anwender versucht, dies zu ändern.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

`Operationen`
Textbereich. Listet alle Operationen auf (siehe Abschnitt 18.8, „ Operation “), die in dieser Schnittstelle definiert sind. Ein Taste 1-Doppelklick navigiert zu der markierten Operation. Ein Taste 2- Klick öffnet ein Popup-Menü mit zwei Einträgen: `Nach oben` und `Nach unten`, die ein Ändern der Reihenfolge erlauben.

Achtung

Alle Operationen in einer Schnittstelle *müssen* public sein. Die ArgoUML-Hinweise 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`
Siehe Abschnitt 18.17.2, „ Eigenschaftssymbolleiste Abstraktion “ und Abschnitt 18.17.3, „ Eigenschaftsfelder für eine Abstraktion “ unten.

`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

ArgoUML wählt den Stereotyp realize automatisch aus, wenn eine Abstraktion erzeugt wird. Dem Anwender steht es frei, diesen Stereotyp zu ändern, um über die Abstraktion zum Beispiel eine `trace`-Beziehung darzustellen. ArgoUML wird jedoch die Darstellung im Diagramm nicht entsprechend ändern.

`Stereotyp navigieren`
 Symbol. Wenn ein Stereotyp markiert wurde, navigiert es in das Eigenschaftsfenster des Stereotyps (siehe Abschnitt 16.6, „ Stereotyp “).

`Namensraum`
Kombinationsfeld. Nimmt den Namensraum der Abstraktion auf und erlaubt deren Änderung. Dies ist die Pakethierarchie.

`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

Dieses Kapitel beschreibt jedes Modellelement, das innerhalb eines Sequenzdiagrammes erzeugt werden kann. Beachten Sie, dass einige Sub-Modellelemente von Modellelementen des Diagrammes nicht selbst im Diagramm erscheinen.

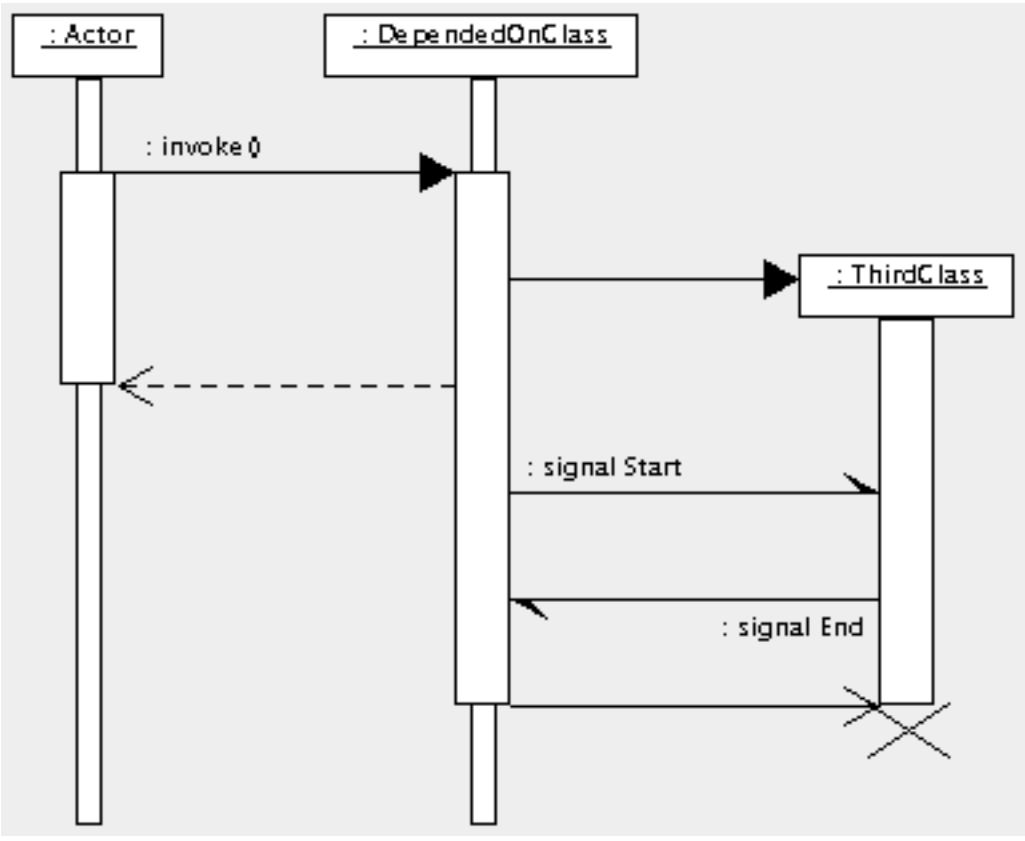
Es gibt eine enge Beziehung zwischen diesem Material und dem Register `Eigenschaften` des Detailfensters (siehe Abschnitt 13.3, „ Das Register Eigenschaften “). Dieser Abschnitt behandelt die Eigenschaften allgemein, in diesem Kapitel sind sie mit spezifischen Modellelementen verknüpft.

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.

Abbildung 19.1. Denkbare Modellelemente in einem Sequenzdiagramm.



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 es gibt keinen Weg, diese mit deren verknüpften Operationen oder Signalen zurückzuverfolgen.

19.2. Objekt

Ein Objekt ist eine Instanz einer Klasse. Im UML-Metamodell ist `Object` eine Subklasse von `Instance` . Innerhalb eines Sequenzdiagrammes könnten Objekte verwendet werden, um eine spezielle Instanz einer Klasse darzustellen. Wie in Kollaborationsdiagrammen (siehe Kapitel 21, *Modellelement-Referenz Kollaborationsdiagramm* können Sequenzdiagramme kein generelles Verhalten zwischen Klassifzierer-Rollen darstellen.

Ein Objekt wird in einem Sequenzdiagramm von ArgoUML als Rechteck mit dem Objektnamen (sofern vorhanden) und durch ein Semikolon abgetrennten (:) Klassennamen dargestellt. Als Verbindungen werden Impulse von und zu anderen Objekten hinzugefügt und eine Zeitline geht vom Objekt aus nach unten. Diese ist dünn, wenn das Objekt keine Steuerungsfunktion hat und dick, wenn es eine hat.

Achtung

Die aktuelle Release von ArgoUML zeigt Interaktionen zwischen Objekten, obwohl der UML-Standard für Sequenzdiagramme für Interaktionen zwischen Instanzen eines Klassifizierers gedacht ist.

Die aktuelle Implementierung in ArgoUML erlaubt jedoch die Anwendung jedes Klassifizierers mit dem Objekt. So kann das Diagramm zum Beispiel Instanzen von Akteuren wie auch Klassen erfolgreich darstellen.

19.2.1. Detail-Register Objekt

Die aktiven Detail-Register für Objekte sind die folgenden.

`Zu-Bearbeiten-Element`
: Standard-Register.

`Eigenschaften`
: Siehe Abschnitt 19.2.2, „ Eigenschaftssymbolleiste Objekt “ und Abschnitt 19.2.3, „ Eigenschaftsfelder für ein Objekt “ unten.

`Dokumentation`
: Standard-Register.

`Darstellung`
: Standard-Register. Die Werte für die Begrenzung des Objeketes 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).
- `derived` (von der Superklasse `ModelElement`). Der Wert `true` bedeutet, dass das Objekt redundant ist - es kann formal von anderen Elementen abgeleitet werden, oder `false`, wenn es das nicht kann.

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 Eigeschaften dieses Stereotyps.

 `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`
: Textfeld. Nimmt den Namensraum für das Objekt auf. Dies ist die Pakethierarchie.

`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

wenn ein Strichmännchen üblich wäre).

19.3. Impuls

Ein Impuls ist eine Kommunikation zwischen zwei Instanzen und wird durch eine Aktion generiert. In einem Sequenzdiagramm ist ein Impuls mit einer Verbindung verknüpft - eine Instanz einer Assoziation verbindet zwei Objektinstanzen. Im UML-Metamodell ist `Stimulus` eine Subklasse von `ModelElement`.

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:

- `Impulsaufruf`. Durch eine Aufrufaktion generiert, ist selbst das Ergebnis einer Operation der Klasse. Wird als durgehende Linie mit einem ausgemalten Pfeil auf der Zeitlinie der, den Impuls empfangenden Objektinstanz dargestellt.
- `Impuls erzeugen`. Von einer Create-Aktion der Klasse generiert, von der eine Instanz erzeugt wurde. Dargestellt als durchgehende Linie mit einem ausgemalten Pfeil auf den Objektkopf der erzeugten Objektinstanz.
- `Impuls löschen`. Von einer Lösch-Aktion des Ursprungsobjektes generiert. Dargestellt als durchgehende Linie mit einem offenen Pfeil. Die Zeitlinie der empfangenden (gelöschten) Objektinstanz wird mit einem diagonalen Kreuz abgeschlossen.
- `Impuls gesendet`. Von einer Sende-Aktion generiert, ist es das Ergebnis eines, durch eine Operation der sendenden Objektinstanz ausgelöstes Signal und wird durch die empfangende Objektinstanz bearbeitet. Dargestellt als durchgehende Linie mit einem halb offenen Pfeil.
- `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

In der aktuellen Release von ArgoUML gibt es keinen Weg, eine Beenden-Aktion anzuzeigen, womit sich eine Objektinstanz selbst zerstört. Ein Weg ist, eine Lösch-Aktion zu zeichnen, die in einer Schleife auf das Objekt selbst zeigt, dieser eine Aktion ohne Namen zu vergeben und mit Hilfe des Registers Darstellung eine unsichtbare Linie einzustellen. Dabei bleibt aber der Pfeil sichbar, der nicht unsichtbar gemacht werden kann. Es ist also immer sematisch falsch, eine Lösch-Aktion zu verwenden, die eine Beenden-Aktion dargstellt.

19.3.1. Detail-Register Impuls

Die aktiven Detail-Register für Impulse sind die folgenden.

`Zu-Bearbeiten-Element`
Standard-Register.

`Eigenschaften`
Siehe Abschnitt 19.3.2, „ Eigenschaftssymbolleiste Impuls “ und Abschnitt 19.3.3, „ Eigenschaftsfelder für einen Impuls “ unten.

`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

Abgeleitete Impulse 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.

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.

Als praktische Konvention wird empfohlen, dass Aufruf- Aktionen mit dem Namen der Operation dargestellt werden, welche die Aktion, einschliesslich aller, in Klammern gesetzten Argumente erzeugen und Sende-Aktionen mit dem Namen des Signales dargestellt werden, welche die Aktion, einschliesslich aller, in Klammern gesetzten Argumente generieren. Rückgabe-Aktionen sollten als Ausdruck für den Rückgabewert dargestellt werden, oder im anderen Fall leer sein. Erzeugen- und Lösch- Aktionen sollten leer bleiben, weil sie durch ihre Darstellung erklärt sind.

`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.

Ein Taste 1-Klick navigiert zu der Empfängerinstanz, Taste 2 öffnet ein Popup-Menü mit einem Eintrag.

- `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`

Textfeld. Nimmt den Namensraum für den Impuls auf. Dies ist die Pakethierarchie.

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.

Alle Detail-Register und Eigenschaften sind mit diesem Impuls im Allgemeinen identisch (siehe Abschnitt 19.3, „ Impuls “). Seine grafische Darstellung im Diagramm ist die eines, mit einer Aufrufaktion verknüpften Impulses, z.B. eine durchgehende Linie mit einem ausgefüllten Pfeil.

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.

Alle Detail-Register und Eigenschaften sind im Allgemeinen mit diesem Impuls identisch (siehe Abschnitt 19.3, „ Impuls “). Seine grafische Darstellung im Diagramm ist die eines, mit einer Erzeugen-Aktion verknüpften Impulses, z.B. eine durchgehende Linie mit einem ausgefüllten Pfeil, der am Kopf der erzeugten Instanz endet.

Anmerkung

Da die aktuelle Release von ArgoUML Aktionen nicht vollständig implementiert hat, gibt es keine Durchgängigkeit zwischen der Beziehung und der Erzeugen-Aktion.

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.

Alle Detail-Register und Eigenschaften sind im Allgemeinen mit diesem Impuls identisch (siehe Abschnitt 19.3, „ Impuls “). Seine grafische Darstellung im Diagramm ist die eines, mit einer Zerstören-Aktion verknüpften Impulses, z.B. eine durchgehende Linie mit einem ausgefüllten Pfeil, der am Ende der zu zerstörenden Instanz mit einem Kreuz abgeschlossen wird.

Anmerkung

Da die aktuelle Release von ArgoUML Aktionen nicht vollständig implementiert hat, gibt es keine Durchgängigkeit zwischen der Beziehung und der Zerstören-Aktion.

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.

Alle Detail-Register und Eigenschaften sind im Allgemeinen mit diesem Impuls identisch (siehe Abschnitt 19.3, „ Impuls “). Seine grafische Darstellung im Diagramm ist die eines, mit einer Senden-Aktion verknüpften Impulses, z.B. eine durchgehende Linie mit einem halb offenen Pfeil.

Anmerkung

Da die aktuelle Release von ArgoUML Aktionen nicht vollständig implementiert hat, gibt es keine Durchgängigkeit zwischen der Beziehung und der Senden-Aktion.

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.

Alle Detail-Register und Eigenschaften sind im Allgemeinen mit diesem Impuls identisch (siehe Abschnitt 19.3, „ Impuls “). Seine grafische Darstellung im Diagramm ist die eines, mit einer Rückgabe-Aktion verknüpften Impulses, z.B. eine gepunktete Linie mit einem halb offenen Pfeil.

Anmerkung

Da die aktuelle Release von ArgoUML Aktionen nicht vollständig implementiert hat, gibt es keine Durchgängigkeit zwischen der Beziehung und der Rückgabe-Aktion.

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`
: Siehe Abschnitt 19.9.2, „ Eigenschaftssymbolleiste Verknüpfung “ und Abschnitt 19.9.3, „ Eigenschaftsfelder für die Verknüpfung “ unten.

`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 Verküpfung 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

Abgeleitete Verknüpfungen 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 den Eigenschaftswert `documentation`. Dieser wird in ArgoUML über das Register *Dokumentation* bearbeitet.

`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 Stereotyps.

 `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`
Textfeld. Nimmt den Namensraum für diese Verknüpfung auf. Dies ist die Pakethierarchie.

`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.

Abbildung 20.1, „ Modellelemente Zustandsdiagramm 1. “ und Abbildung 20.2, „ Modellelemente Zustandsdiagramm 2. “ zeigen Zustandsdiagramme mit den denkbaren Modellelementen.

Abbildung 20.1. Modellelemente Zustandsdiagramm 1.

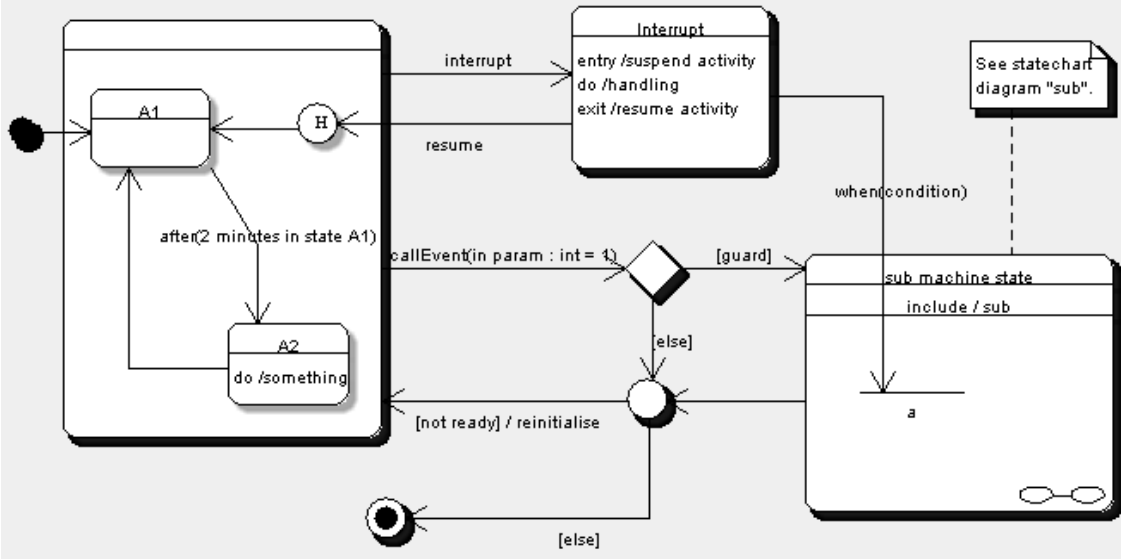
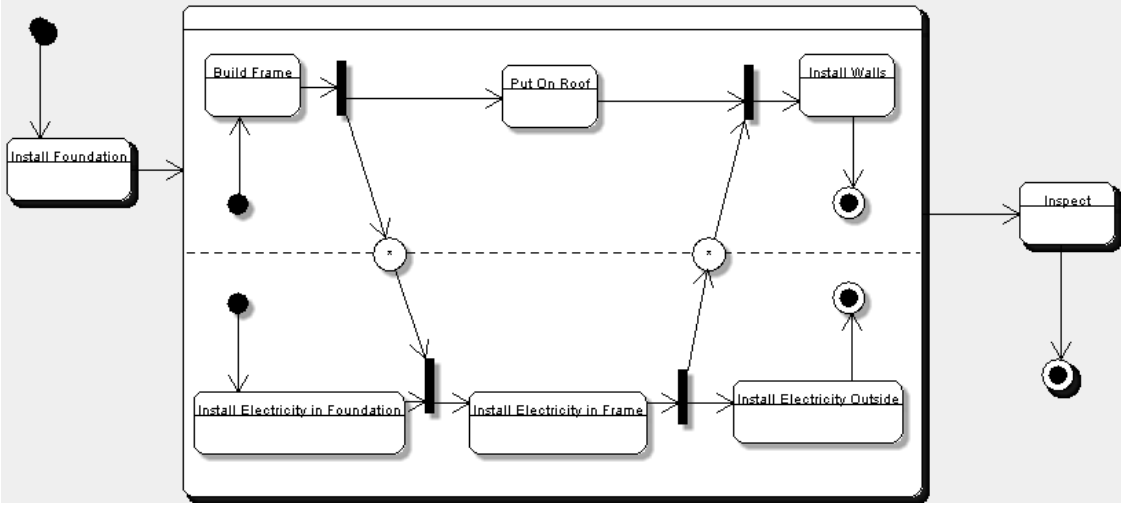


Abbildung 20.2. Modellelemente Zustandsdiagramm 2.



20.1.1. Einschränkungen, die Zustandsdiagramme in ArgoUML betreffend

Die Zustandsdiagramme untersützen 7 definierte Aktionstypen (CallAction, CreateAction, DestroyAction,ReturnAction, SendAction, TerminateAction und UninterpretedAction). Es gibt aber keinen Weg, die gleiche Aktion mehr als ein Mal zu verwenden. Daher ist es in einigen Fällen nicht möglich, die entsprechenden Elemente zu setzen oder auszuwählen. Z.B. gibt es keine Möglichkeit, ein Signal für eine Senden-Aktion auszuwählen.

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.

Ein Zustand wird in einem Zustandsdiagramm von ArgoUML als Rechteck mit abgerundeten Ecken dargestellt, mit einer horizontalen Linie, die den Namen im oberen Teil von der Beschreibung des Verhaltens im unteren Teil trennt. Die Beschreibung des Verhaltens schließt die Ein- und Ausgangs-Aktionen und jede interne Transition ein.

20.2.1. Detail-Register Zustand

Die aktiven Detail-Register für Zustände sind die folgenden.

`Zu-Bearbeiten-Element`
: Standard-Register.

`Eigenschaften`
: Siehe Abschnitt 20.2.2, „ Eigenschaftssymbolleiste Zustand “ und Abschnitt 20.2.3, „ Eigenschaftsfelder für einen Zustand “ unten.

`Dokumentation`
: Standard-Register.

`Darstellung`
Standard-Register. Die Werte der Begrenzung des Zustandes definieren den umgebenden Rahmen des Zustandes.

`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

Dies ist ein Löschen aus dem Modell, *nicht* nur aus dem Diagramm. Sie können einen Zustand aus dem Diagramm entfernen und ihn innerhalb des Modelles erhalten, wie dies in den anderen Diagrammen möglich ist.

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`
Textfeld. Zeigt den Container dieses Zustandes. Dies ist die Zustandshierarchie.

Ein Taste 1-Doppelklick auf diesen Eintrag navigiert zu dem zusammengesetzten Zustand, der diesen Zustand enthält. Alle Zustände sind mindestens in einem versteckten Zustand auf oberster Ebene enthalten („top“ genannt), der die Wurzel der Zustands-Container-Hierarchie bildet.

`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:

- `Neu`. Fügt eine neue Eintritt-Aktion einer bestimmten Art ein. Dieses Menü hat die folgenden Submenüs, um die Art der Aktion auszuwählen: Call Action, Create Action, Destroy Action, Return Action, Send Action, Terminate Action, Uninterpreted Action.
- `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.

- `Neu`. Fügt eine neue Austritt-Aktion einer bestimmten Art ein. Dieses Menü hat die folgenden Submenüs, um die Art der Aktion auszuwählen: Call Action, Create Action, Destroy Action, Return Action, Send Action, Terminate Action, Uninterpreted Action.
- `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.

- `Neu`. Fügt eine neue Ausführen-Aktion einer bestimmten Art ein. Dieses Menü hat die folgenden Submenüs, um die Art der Aktion auszuwählen: Call Action, Create Action, Destroy Action, Return Action, Send Action, Terminate Action, Uninterpreted Action.
- `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.
- `Neu`. Fügt ein neues Ereignis einer bestimmten Art hinzu. Dieses Menü hat die folgenden Submenüs, um die Art des Ereignisses auszuwählen: Call Event, Change Event, Signal Event, Time Event.

- `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`
Textbereich. Listet alle internen Transitionen des Zustandes auf. Solche Transitionen weder verlassen noch gehen in den Zustand hinein, so dass sie keinen Zustandswechsel verursachen. Das bedeutet, das die Eintritt- und Austritt- Aktionen nicht aufgerufen werden.

Anmerkung

Dieses Feld zeigt den Namen der Transition, während im Diagramm der Name des Auslösers, mit einem `/` getrennt vom Auswirkungsskript angezeigt wird.

Ein Taste 1-Doppelklick navigiert zu der markierten Transition, Taste 2 öffnet ein Popup-Menü mit einem Eintrag.

- `Neu`. Fügt eine neue interne Transition hinzu.

20.3. Aktion

Eine Aktion spezifiziert eine ausführbare Anweisung und ist eine Abstraktion einer verarbeitbaren Prozedur, die den Zustand des Modelles ändern kann. Im UML-Metamodell ist sie ein Kind von `ModelElement`. Weil im Metamodell eine ActionSequence selbst eine Aktion ist, die eine Aggregation von anderen Aktionen ist (z.B. das "zusammengesetzte"-Muster), kann eine ActionSequence überall dort verwendet werden, wo auch eine Aktion sein darf.

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 praktiable Konvention wird angenommen, dass Aufruf-Aktionen als Name der Operation dargestellt werden, die die Aktion mit in Klammer gesetzten Argumenten generiert und Sende-Aktionen 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`
Siehe Abschnitt 20.3.2, „ Eigenschaftssymbolleiste Aktion “ und Abschnitt 20.3.3, „ Eigenschaftsfelder für eine Aktion “ unten.

`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 Stereotyps.

 `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 eine 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 Programmiersprache, mit der dieser Ausdruck geschrieben wurde.

`Rekursion`
Doppeltes Textfeld. Ein Ausdruck, der ausdrückt, wie oft die Aktion ausgeführt werden soll. Das Feld besteht aus zwei Teilen: dem ersten für den Ausdruck und den zweiten für die Sprache, in der dieser Ausdruck geschrieben wurde.

`Argumente`
Textfeld. Dies ist eine geordnete Liste mit Argumenten der Aktion.

Ein Taste 1-Doppelklick 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.

`Instanziierung (nur für CreateAction)`
Textfeld. Dies zeigt den Klassifizierer, der durch die Erzeugen-Aktion instanziiert wird.

Ein Taste 1-Doppelklick 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.

Ein zusammengesetzter Zustand wird in einem Zustandsdiagramm von ArgoUML als ein grosses Rechteck mit abgerundeten Ecken, mit einer horizontalen Linie, der den Namen oben von der Beschreibung des Verhaltens und das Modell des Sub-Zustandsautomaten unten trennt, dargestellt. die Beschreibung des Verhaltens beinhaltet die Eintritt-, Austritt- und Ausführen-Aktionen und die internen Transitionen.

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, 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.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrittspunkt, Taste 2 öffnet ein Popup-Menü mit zwei Einträgen.

- `Neu`. Es öffnet sich ein Submenü mit einer Auswahl von 7 Zustandsarten, die dem Modell hinzugefügt werden können. Die 7 unterstützten Zustandsarten sind: Pseudo-Zustand, Synch-Zustand, Stub-Zustand, Zusammengesetzter Zustand, Einfacher Zustand, Abschliessender Zustand, Sub-Zustandsautomat Zustand.

Warnung

Dem Modell Zustände auf diese Art und Weise hinzuzufügen ist keine gute Idee, weil Sie den Zustand dem Diagramm später hinzufügen müssen. Dies können Sie tun, indem Sie ihn im Explorer markieren und das Popup-Menü aktivieren und dort `Zum Diagramm hinzufügen` auswählen. Es ist ratsam, stattdessen die Symbolleiste des Diagrammes zu nehmen.

- `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

Ein Subautomaten-Zustand ist eine syntaktische Bequemlichkeit, die die Wiederverwendung und die Modularisation erleichert. Es ist eine Kurzform die eine makroähnliche Erweiterung durch einen anderen Zustandsautomaten darstellt, der sematisch einem zusammengesetzten Zustand entspricht. Der eingefügte Zustandsautomat wird referenzierter Zustandsautomat genannt, während der Zustandsautomat, der den Subautomaten-Zustand enthält, der haltende Zustandsautomat genannt wird. Der gleiche Zustandsautomat kann mehr als einmal im Kontext eines einfachen haltenden Zustandsautomaten referenziert werden. In der Tat stellt ein Subautomaten-Zustand ein *Aufruf* einer *Subroutine* eines Zustandsautomaten dar, mit einem oder mehreren Eintritts- und Austrittspunkten. Die Eintritts- und Austrittspunkte werden durch flache Zustände spezifiziert.

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.

Ein Subautomatenzustand stellt den Aufruf eines irgendwo definierten Zustandsautomaten dar. Im allgmeinen Fall, kann ein aufgerufener Zustandsautomat an jedem seiner Subzustände oder an seinem Standard (initialen) Pseudozustand eingegeben werden. Ebenso läßt er sich an jedem Subzustand oder als Ergebnis des aufgerufenen Zustandsautomaten verlassen werden und erreicht seinen abschliessenden Zustand. Die nicht-standardmäßigen Eintritts- oder Austrittspunkte werden über *flache Zustände* spezifiziert. Im UML-Metamodell ist StubState ein Kind von State.

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 ist eine gerichtete Beziehung zwischen einem Ursprungszustand (jeder Art, z.B. ein zusammengesetzter Zustand) und einem Zielzustand (jeder Art, z.B. ein zusammengesetzter Zustand). Innerhalb des UML-Metamodelles ist `Transition` eine Subklasse von `ModelElement`.

Eine Transition wird in einem Zustandsdiagramm in ArgoUML als Linie mit einem Pfeil dargestellt, der

den Ursprungs- mit dem Zielzustand verbindet. In der Nähe dieser Linie befindet sich eine Zeichenkette, die die folgenden drei Teile enthält: Das Auslöseimpuls-Ereignis (z.B. ein Aufruf-Ereignis), das Parameter zwischen Klammern haben kann `()`. Als nächstes folgen (falls vorhanden) der Wächter in eckigen Klammern (`[]`). Abschliessend, wenn ein Effekt (z.B. Aufruf-Aktion) definiert ist, ein Schrägstrich (`/`), gefolgt durch einen Aktionsausdruck.

20.8.1. Detail-Register Transition

Die aktiven Detail-Register für Transitionen sind die folgenden.

`Zu-Bearbeiten-Element`
: Standard-Register.

`Eigenschaften`
: Siehe Abschnitt 20.8.2, „ Eigenschaftssymbolleiste Transition “ und Abschnitt 20.8.3, „ Eigenschaftsfelder für eine Transition “ unten.

`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 Stereotyps.

 `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 Hauptmenü (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`
Textfeld. Zeigt den Namen des Eltern-Zustandsautomaten der Transition.

Ein Taste 1-Doppelklick navigiert zu dem gezeigten Zustandsautomaten.

`Zustand`
Textfeld. Zeigt den Namen des Eltern-Zustandes im Fall einer internen Transition.

Ein Taste 1-Doppelklick navigiert zu dem gezeigten Zustand.

`Quellcode`
Textfeld. Zeigt den Ursprungszustand der Transition an.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag.

`Ziel`
Textfeld. Zeigt den Zielzustand der Transition.

Ein Taste 1-Doppelklick 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.

Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag, Taste 2 öffnet ein Popup-Menü mit drei Einträgen.

- `Auswählen - Hinzufügen...` Fügt ein existierendes Auslöseimpuls-Ereignis hinzu. Ein Untermenü öffnet mit 4 Einträgen: Aufruf-Ereignis, Änderungs- Ereignis, Signal-Ereignis, Zeit-Ereignis.

- `Neu`. Fügt ein neues Auslöseimpuls- Ereignis hinzu. Ein Untermenü mit 4 Einträgen öffnet: Aufruf-Ereignis, Änderungs-Ereignis, Signal-Ereignis, Zeit-Ereignis.

- `Aus Modell entfernen`. Löscht das Auslöseimpuls-Ereignis aus dem Modell. Diese Eigenschaft ist in der aktuellen Version von ArgoUML immer deaktiviert.

`Wächter`
: Textfeld. Zeigtden Namen eines Wächters (falls vorhanden). Der Ausdruck eines Wächters muss true sein, bevor diese Transition ausgeführt werden kann.

  Ein Taste 1-Doppelklick navigiert zu dem markierten Eintrag, Taste 2 öffnet ein Popup-Menü mit einem Eintrag.

  - `Neu`. Fügt einen neuen Wächter hinzu.

`Effekt`
: Textfeld. Zeigt die Aktion (falls vorhanden), die aufgerufen wird, wenn dies Transition ausgeführt wird.

  Ein Taste 1-Doppelklick navigiert zu der markierten Aktion, Taste 2 öffnet ein Popup-Menü mit zwei Einträgen.

  - `Neu`. Fügt einen neuen Effekt (Aktion) einer bestimmten Art hinzu. Dieses Menü hat die folgenden Submenüs, um die Art der Aktion auszuwählen: Aufruf-Aktion, Erzeugen-Aktion, Zerstören-Aktion, Rückgabe-Aktion, Sende-Aktion, Beenden-Aktion, uninterpretierte Aktion.

  - `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-Metamodelles 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`
Siehe Abschnitt 20.9.2, „ Eigenschaftssymbolleiste Ereignis “ und Abschnitt 20.9.3, „ Eigenschaftsfelder für ein Ereignis “ unten.

`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, das 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 einen 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 `[sig]` zu verwenden. Für Zeitereignisse verwenden Sie einen Zeitausdruck mit dem Präfix `[time]` und für Änderungsereignisse den Änderungsausdruck mit dem Präfix `[change]`.

`Namensraum`
Textfeld. Zeigt den Namensraum für ein Ereignis. Dies ist die zusammengesetzte Hierarchie.

`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`
Kombinationsfeld. Nur für Aufruf-Ereignisse vorhanden. Erlaubt das Spezifizieren der Operation, die von dem Ereignis ausgelöst wird.

`Signal`
Textfeld. Nur bei Signalereignissen vorhanden. Erlaubt das Spezifizieren eines Signals, welches das Ereignis auslöst.

Ein Taste 1-Doppelklick navigiert zu dem markierten Signal, Taste 2 öffnet ein Popup-Menü mit zwei Einträgen.

- `Hinzufügen...`. Öffnet ein Dialogfenster, das es erlaubt, ein bereits existierendes Signal auszuwählen.

- `Neues Signal`. Erzeugt ein neues Signal und navigiert dort hin.

`Wenn`
Doppeltes Textfeld. Nur bei Zeitereignissen vorhanden. Erlaubt es, die Zeit auszudrücken, die das Ereignis ausgelöst hat.

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 Änderungsereignisses ein Feld, um dem Änderungsausdruck einzugeben.

20.10. Wächeter

Ein Wächter ist mit einer Transition verknüpft. Zum Zeipunkt des Verteilens eines Ereignisses, wird der Wächter überprüft und, wenn false, dessen Transition ausgeschaltet. Im UML-Metamodell ist `Gurad` ein Kind von `ModelElement` .

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`
: Siehe Abschnitt 20.10.2, „ Eigenschaftssymbolleiste Wächter “ und Abschnitt 20.10.3, „ Eigenschaftsfelder für einen Wächter “ unten.

`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 Eignschaften 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 Taste 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

Ein Pseudozustand umfasst eine Reihe von verschiedenen transienten Eckpunkte in eine Zustandsdiagramm. Sie werden typischerweise verwendet, mehrere Transitionen in komplexeren Zustandstransitionspfaden miteinander zu verbinden. Zum Beispiel, durch kombinieren einer Transition durch Eingabe eines Entscheidungs- Pseudozustandes mit einer Reihe von Transitionen, welche die Entscheidung verlassen, erhalten wir ein zusammengesetzte Transition, die zu einer Reihe von nebenläufigen Zielzuständen führt. Pseudozustände haben keine Eigenschaften eines vollständigen Zustandes und dienen nur als Verbindungspunkt für Transaktionen (aber mit einem semantischen Wert). Innerhalb des UML-Metamodelles ist `Pseudostate` eine Subklasse von `StateVertex`.

Die Darstellung eines Pseudozustandes in einem Zustandsdiagramm von ArgoUML hängt von der Art des Pseudozustandes ab: initial, tiefe Historie, flache Historie, Verknüpfung, Entscheidung, Kreuzung, und Auswahl. ArgoUML läßt Sie jeden Pseudozustand durch die Werkzeuge für die spezifischen Typen des Pseudozustandes direkt plazieren. Diese sind in den nachfolgenden Abschnitten beschrieben (siehe Abschnitt 20.12, „ Start-Zustand “, Abschnitt 20.14, „ Kreuzung “, Abschnitt 20.15, „ Entscheidung “, Abschnitt 20.16, „ Gabelung “, Abschnitt 20.17, „ Vereinigung “, Abschnitt 20.18, „ Flache Historie “ und Abschnitt 20.19, „ Tiefe Historie “).

20.11.1. Detail-Register Pseudozustand

Die aktiven Detail-Register für Pseudozustände sind die folgenden.

`Zu-Bearbeiten-Element`
Standard-Register.

`Eigenschaften`
Siehe Abschnitt 20.11.2, „ Eigenschaftssymbolleiste Pseudozustand “ und Abschnitt 20.11.3, „ Eigenschaftsfelder für einen Pseudozustand “ unten.

`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 Eigenschaten 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`
Textfeld. Zeigt den Container des Pseudozustandes. Dies ist die Zustandshierarchie.

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

Der Startzustand ist ein Pseudozustand (siehe Abschnitt 20.11, „ Pseudozustand “), der den Ursprung für eine einzelne Transition zu dem *Standard*-Zustand eines zusammengesetzten Zustandes darstellt. Es ist der Zustand, von dem aus alle initialen Transition ausgehen.

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 ein Hinweis 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`
Siehe Abschnitt 20.13.2, „ Eigenschaftssymbolleiste Endzustand “ und Abschnitt 20.13.3, „ Eigenschaftsfelder für einen Endzustand “ unten.

`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 erzwingt diese Namenskonvention nicht.

Tipp

Endzustandnamen werden im Diagramm dargestellt, aber es ist normalerweise nicht notwendig, ihnen einen Namen zu geben.

`Container`
Textfeld. Zeigt den Container des Endezustandes. Dies ist die Zustandshierarchie.

Ein Taste 1-Doppelklick auf den Eintrag wird zu dem zusammengesetzten Zustand navigieren, der diesen Zustand enthält (oder den Zustand auf oberster Ebene, der 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:

- `Neu`. Fügt eine neue Eintrittsaktion einer bestimmten Art hinzu. Diese Menü hat die folgenden 7 Untermenüs, um die Art der Aktion auswählen zu können: Aufruf-Aktion, Erzeugen-Aktion, Zerstören-Aktion, Rückgabe-Aktion, Sende-Aktion, Beenden-Aktion, uninterpretierte Aktion.
- `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`
Textbereich. Listet alle internen Transitionen des Zustandes auf. Solche Transitionen treten werder aus noch treten in den Zustand ein, so dass sie keinen Statuswechsel verursachen. Das bedeutet, dass die Eintritts- und Austritts-Aktionen nicht aufgerufen werden.

Ein Taste 1-Doppelklick navigiert zu der markierten Transition.

20.14. Kreuzung

Eine Kreuzung ist ein Pseudozustand (siehe Abschnitt 20.11, „ Pseudozustand “), der verwendet wird, um ein ankommende Transition in mehrere abgehende Transitionssegmente mit unterschiedlichen Wächterbedingungen aufzuteilen. Eine Kreuzung wird auch Misch- oder Statischer Bedingungszweig genannt. Die gewählte Transition ist die, deren Wächter am Ende der Transition true ist.

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

Eine Entscheidung ist ein Pseudozustand (siehe Abschnitt 20.11, „ Pseudozustand “) der dazu verwendet wird, eine ankommende Transition in mehrere abgehende Transitionssegmente mit unterschiedlichen Wächterbedingungen aufzusplitten. Daher erlaubt eine Entscheidung eine dynamische Auswahl abgehender Transitionen. Die gewählte Transition ist die, deren Wächter während der Lebensdauer der Transition true ist (wenn mehr als eine true ist, wird nur eine zeitgleich ausgewählt).

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 nebenläufige 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 nebenläufige Transitionen zu einer Transition vereinigt.

Achtung

Die ankommenden Transitionen sollten keine Wächter aufweisen. ArgoUML erzwingt dies

jedoch nicht.

Eine Verzweigung wird im Diagrmm als ausgefüllte (standardmäßig schwarz) horizonaler 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

Eine flache Historie ist ein Pseudozustand (siehe Abschnitt 20.11, „ Pseudozustand “), der sich an den letzten Zustand seines aktiven Containers erinnern kann. Die Historien- Pseudozustand zeigt mit einem Transitionspfeil auf seinen Standardzustand, so wie es der Start-Pseudozustand tut. Diese Transition zeigt auf den Unterzustand, der aktiv wird, wenn es keine Historie gibt. Wenn der zusammengesetzte Zustand des Containers vorher aktiv wurde (z.B., wenn es eine Historie gibt), wird der Unterzustand, der aktiv war, wenn der Containerzustand verlassen wurde, erneut aktiv.

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

Eine tiefe Historie ist ein Pseudozustand (siehe Abschnitt 20.11, „ Pseudozustand “), der sich an den letzten Zustand seines aktiven Containers erinnern kann. Der Pseudozustand der Historie zeigt mit einem Transitionspfeil auf seinen Standardzustand, so wie es der Start-Pseudozustand tut. Diese Transition zeigt auf den Unterzustand, der aktiv wird, wenn es keine Historie gibt. Wenn der zusammengesetzte Containerzustand vorher aktiv war (z.B., wenn es eine Historie gibt), wird der Unterzustand, der aktiv war, wenn der Containerzustand verlassen wurde, erneut aktiv wird.

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 ist gedacht für die Synchronisation nebenläufiger Regionen von Zustandsautomaten. Sie wird in Verbindung mit Gabelungen und Vereinigungen verwendet, um sicherzustellen, dass eine Region einen bestimmten Zustand oder Zustände ausgibt, bevor eine andere Region in einen bestimmten Zustand oder Zustände eintreten kann. Das feuern abgehender Transitionen aus einem Synchronisationszustand kann durch die Angabe einer Grenze bezüglich der Differenz zwischen der Anzahl der abgehenden und ankommenden Transitionen begrenzt sein. Im UML-Metamodell ist `Synch` ein Kind von `StateVertex`.

Ein Synchronisationszustand wird als kleiner Kreis mit der oberen Grenze innerhalb des Kreises

dargestellt. Die Grenze ist entweder eine positive Integerzahl oder ein Stern ('*') für unbegrenzt. Snychronisationszustände werden, wenn möglich, auf der Grenze zwischen zwei Regionen gezeichnet.

20.20.1. Detail-Register Synchronisationszustand

Die aktiven Detail-Register für einen Synchronisationszustand sind die folgenden.

`Zu-Bearbeiten-Element`
: Standard-Register.

`Eigenschaften`
: Siehe Abschnitt 20.20.2, „ Eigenschaftssymbolleiste Synchronisationszustand “ und Abschnitt 20.20.3, „ Eigenschaftsfelder für einen Synchronisationszustand “ unten.

`Dokumentation`
: Standard-Register.

`Darstellung`
: Standard-Register.

`Stereotyp`
: Standard-Register, welches die Stereotypen des Synchronisationszustandes 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 Synchronisationszustand

 `Nach oben`

Navigiert in der Paketstruktur nach oben.

 `Neuer Stereotyp`

Erzeugt einen neuen Stereotyp (siehe Abschnitt 16.6, „ Stereotyp “) für den markierten Synchronisationszustand und springt sofort in das Register Eigenschaften dieses Stereotyps.

 `Aus Modell entfernen`

Löscht den Synchronisationszustand 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 verwendent 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`
Textfeld. Zeigt den Container des Synchronisationszustandes. Dies ist die Zustandshierarchie.

Ein Taste 1-Doppelklick auf diesen Eintrag wird zu dem zusammengesetzten Zustand navigieren, der diesen Zustand enthält (oder den Zustand auf oberster Ebene, der die Wurzel der Zustandshierarchie ist).

`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`
Textbereich. Listet alle ankommenden Transitionen des Endzustandes auf.

Ein Taste 1-Doppelklick navigiert zu der markierten Transition.

`Abgehende Transitionen`
Textbereich. Listet alle abgehenden Transitionen für den Endzustand auf.

Ein Taste 1-Doppelklick navigiert zu der markierten Transition.

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.

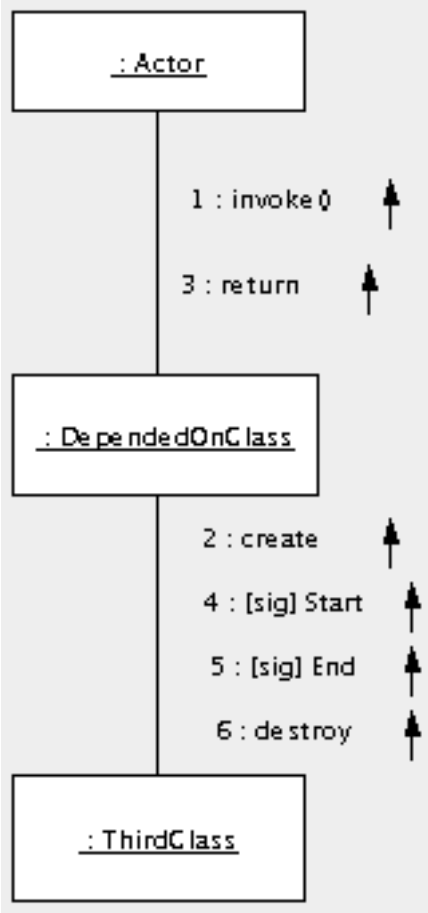
Es gibt eine enge Beziehung zwischen diesem Material und dem Register Eigenschaften des Detailfensters (siehe Abschnitt 13.3, „ Das Register Eigenschaften “). Dessen Abschnitt betrachtet die Eigenschaften im Allgemeinen. In diesem Kapitel werden sie mit spezifischen Modellelementen verküpft.

Achtung

Kollaborationsdiagramme sind aktuell nicht vollständig in ArgoUML implementiert. Viele Aspekte sind nicht vollständig implementiert oder Verhalten sich nicht so, wie erwartet. Im Einzelnen gibt es einige Probleme mit dem Layout der Kollaborationsrollen und -Meldungen.

Abbildung 21.1, „ Denkbare Modellelemente in einem Kollaborationsdiagramm. “ zeigen ein Kollaborationsdiagramm mit allen möglichen Modellelementen.

Abbildung 21.1. Denkbare Modellelemente in einem Kollaborationsdiagramm.



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.

Der grössten Unterschiede gibt es bei den Nachrichten. Dort gibt es Probleme bei der Reihenfolge der Nachrichten und deren Darstellung im Diagramm. Die dahinter stehenden Aktionen sind rein textuell implementiert und es gibt keinen Weg, diese zu deren verküpften Operationen oder Signalen zurückzuverfolgen.

21.2. Klassifizierer Rolle

Eine Klassifzierer 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 Kollaborationsdiagrammes 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.

Eine Klassifizierer Rolle wird in einem Sequenzdiagramm von ArgoUML als leeres Rechteck, gekennzeichnet mit dem Rollenname des Klassifiziers (falls vorhanden) und dem durch ein Doppelpunkt (:) getrennten Klassifizierer dargestellt.

Achtung

Eine Klassifizierer Rolle sollte wahrscheinlich auch den Objektnamen anzeigen (falls vorhanden), dem der Rollenname des Klassifizierers, getrennt durch einen Schrägstrich (`/`) vorangestellt wird. Dies erlaubt es, das Klassifizier 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 Klassifzierer Rollen sind die folgenden.

`Zu-Bearbeiten-Element`
: Standard-Register.

`Eigenschaften`
: Siehe Abschnitt 21.2.2, „ Eigenschaftssymbolleiste Klassifierer Rolle “ znd Abschnitt 21.2.3, „ Eigenschaftsfelder für eine Klassifizierer Rolle “ unten.

`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 Klassifzierer 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

Abgeleitete Klassifizierer Rollen haben ihren Wert in 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 den Eigenschaftswert `documentation`, der in ArgoUML im Register `Dokumentation` bearbeitet wird.

21.2.2. Eigenschaftssymbolleiste Klassifierer 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.

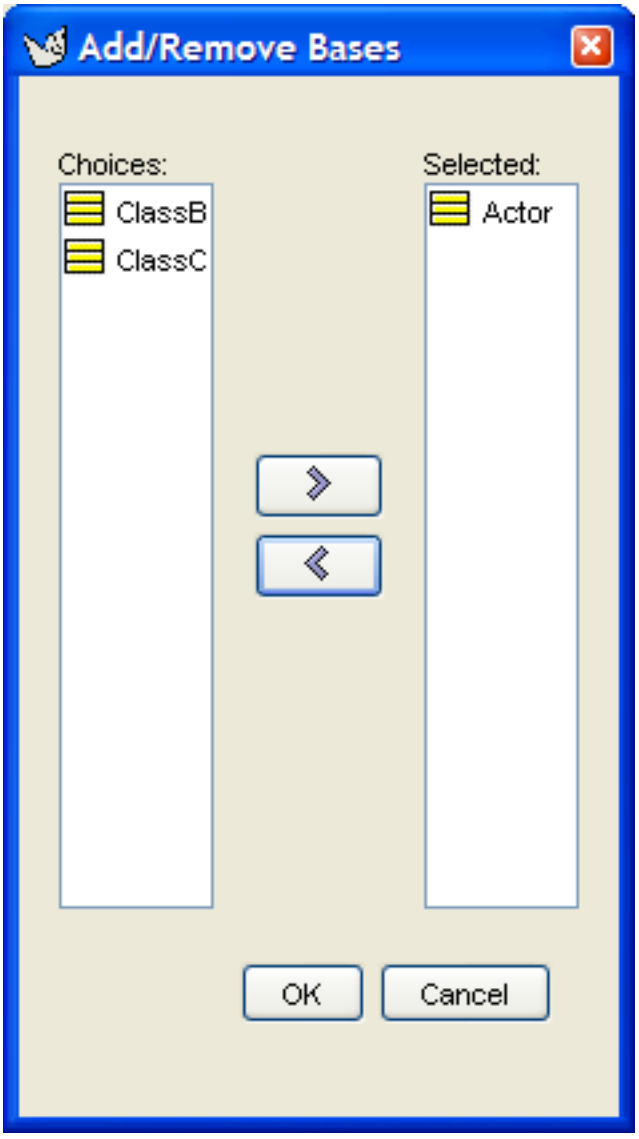
`Kardinalität`
Editierbares Kombinationsfeld. Der Standardwert ist `*`, was bedeutet, dass es beliebig viele Instanzen dieser Klassifizierer Rolle gibt, die in der Kollaboration eine Rolle spielen. Das Kombinationsfeld bietet einige andere Kardinalitäten an. Z.B. `1..1` würde bedeuten, dass nur eine Instanz in der Kollaboration eine Rolle spielen würde.

ArgoUML schränkt Sie hinsichtlich der vordefinierten Kardinalitäten nicht ein. Sie können diese Feld frei editieren.

`Basis`
Liste. Die Namen der Klassifizierer dieser Klassifizierer Rolle. Ein Taste 1-Doppelklick navigiert zu dem Klassfizierer. Taste 2 öffnet ein Popup-Menü mit den folgenden Einträgen.

- `Hinzufügen`. Erlaubt das Hinzufügen oder Entfernen von Klassifizierern in der Liste. An seinem Ende öffnet sich ein Dialogfenster, wie im Bild unten gezeigt.

Abbildung 21.2. Das Dialogfenster „Kontext hinzufügen“



- `Entfernen`. Erlaubt das Entfernen von Klassifzierern aus der Liste, ohne das Dialogfenster zu benutzen.

`Generalisierungen`
Textbereich. Listet jede Klassifizierer Rolle auf, die diese Klassifzierer Rolle *generalisiert*.

Ein Taste 1-Doppelklick navigiert zu der Generalisierung und öffnet dessen Register Eigenschaften.

`Spezialisierungen`
Textfeld. Listet jede spezialisierte Klassfizierer 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 Subsetz 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 ist eine Spezialisierung einer Assoziation, die dazu verwendet wird, ein Assoziationsverhalten in einem bestimmten Kontext zu beschreiben. Im UML-Metamodell ist `Association Role` eine Subklasse von `Association`.

Eine Assoziationsrolle wird in einem Kollaborationsdiagramm von ArgoUML als Linie dargestellt, welche die betreffenden Instanzen miteinander verbindet. In einem Sequenzdiagramm ist die Darstellung jedoch modizifiziert, 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:

/ Assoziationsrollenname : 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`
Siehe Abschnitt 21.3.2, „ Eigenschaftssymbolleiste Assoziationsrolle “ und Abschnitt 21.3.3, „ Eigenschaftsfelder für eine Assoziationsrolle “ unten.

`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 Desgin, um eine Wiederverarbeitung zu verhindern.

Anmerkung

Die UML-Metaklasse `Element`, aus der alle anderen Modellelemente 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`
Kombinationsfeld. Die Assoziationsrolle bietet standardmäßig den UML-Standard-Stereotyp der Superklasse Assoziation an: `implicit`.

`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`.

Zwei oder mehr Rollen Assoziationsende sind mit jeder Assoziationsrolle verknüpft (siehe Abschnitt 21.3, „ Assoziationsrolle “), obwohl in ArgoUML die Anzahl der Enden nur zwei sein kann.

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

Eine Rolle Assoziationsende sollte die gleichen, oder „ genauer“ Attributwerte haben, wie das Basis- Assoziationsende. Insbesondere seine Navigierbarkeit sollte nicht mehr Allgemein sein. Es gibt noch keinen Hinweis in ArgoUML, die eine Erläuterung zu dieser Regel bietet.

21.4.1. Detail-Register Rolle Assoziationsende

Die aktiven Detail-Register für eine Rolle Assoziationsende sind die folgenden.

`Zu-Bearbeiten-Element`
Standard-Register.

`Eigenschaften`
Siehe Abschnitt 21.4.2, „ Eigenschaftssymbolleiste Rolle Assoziationsende “ und Abschnitt 21.4.3, „ Eigenschaftsfelder für eine Rolle Assoziationsende “ unten.

`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

Abgeleitete Rollen Assoziationsende haben ihren Wert in der Analyse, um nützliche Namen oder Konzepte einzuführen und im Design, um eine Wiederverarbeitung zu verhindern. Jedoch macht das Markieren für eine Rolle Assoziationsende nur Sinn, wenn es auch auf die Eltern-Assoziationsrolle angewendet wird.

Anmerkung

Die UML-Metaklasse `Element`, von der alle anderen Modellelemente abgeleitet werden, beinhaltet einen Eigenschaftswert `documentation`, der in ArgoUML im Register *Dokumentation* bearbeitet wird.

21.4.2. 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 Qualifzierer 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

Diese Schaltfläche ist für binäre Assoziationsrollen deaktiviert, weil eine Assoziation mindestens *zwei* Enden benötigt. Nur für N-fach-Assoziationen ist diese Schaltfläche verfügbar und löscht ein Ende aus der Assoziation.

21.4.3. Eigenschaftsfelder für eine Rolle Assoziationsende

`Name`
Textfeld. Der Name einer Rolle Assoziationsende, die einen *Rollenname* 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`
Textfeld, das den Namen des entsprechenden Assoziationsendes anzeigt. Ein Taste 1-Doppelklick navigiert zu dem Assoziationsende.

`Assoziationsrolle`
Textfeld. Nimmt die Eltern-Assoziationsrolle für diese Rolle Assoziationsende auf. Ein Taste 1-Doppelklick navigiert zu den Assoziationsrolle.

`Typ`
Das Kombinationsfeld bietet Zugriff auf alle Standard-UML- Typen, die durch ArgoUML unterstützt werden und alle neuen, innerhalb des aktuellen Modelles erzeugten Klassen.

Dies ist der Entitätstyp, der diesem Ende der Assoziationsrolle zugewiesen wurde.

`Kardinalität`
Editierbares Kombinationsfeld. Erlaubt die Änderung der Kardinalität dieser Rolle Assoziationsende (unter Berücksichtigung des anderen Endes). Z.B. wie viele Instanzen dieses

Endes darf mit einer Instanz des anderen Endes verknüpft werden. Die Kardinalität wird im Diagramm am Ende der Assoziationsrolle angezeigt.

`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

Eine Nachricht ist eine Kommunikation zwischen zwei Instanzen einer Assoziationsrolle in einem Kollaborationsdiagramm auf Spezifikationsebene. Sie beschreibt eine Aktion, die den mit der Nachricht verküpften Impuls generiert. In einem Kollaborationsdiagramm ist eine Nachricht mit einer Assoziationsrolle verknüpft. Im UML-Metamodell ist `Message` eine Subklasse von `ModelElement`.

Die Nachricht wird in einem Kollaborationsdiagramm von ArgoUML durch seine durch einen Doppelpunkt vom Ausdruck, der die verküpfte Aktion definiert, getrennten Sequenznummer dargestellt. Dies wird begleitet durch einen Pfeil der in Richtung der Kommunikation zeigt, z.B. die Richtung der Assoziationsrolle. Per Konvention wird der Name einer Nachricht nicht im Diagramm angezeigt. Anstelle dessen zeigt das Diagramm die Sequenznummer der Nachricht an, entweder als eine Integerzahl oder als eine dezimale Zahl, um die Hierarchie darzustellen.

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`
Siehe Abschnitt 21.5.2, „ Eigenschaftssymbolleiste Nachricht “ und Abschnitt 21.5.3, „ Eigenschaftsfelder für eine Nachricht “ unten.

`Dokumentation`
Standard-Register.

`Darstellung`
Standard-Register. Die Werte für die Begrenzung der Nachricht definiert der umgebende Rahmen der Nachricht. Das Feld `Linienfarbe` definiert die Pfeilfarbe. Das Vergrößern der `Schatten`-Größe hat eine ästhetisch fragwürdigen Effekt.

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 Stereotyps.

 `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`
Kombinationsfeld. Nachrichten haben standardmäßig keine Stereotypen gemäß UML-Standard.

`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. Indentifiziert 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.

Ein Taste 1-Doppelklick navigiert zu der markierten Aktion, Taste 2 öffnet ein Popup-Menü mit den folgenden Einträgen.

- `Neu`. Für eine neue Aktion hinzu.

  Dieses Element ist deaktiviert, wenn eine Aktion bereits existiert.

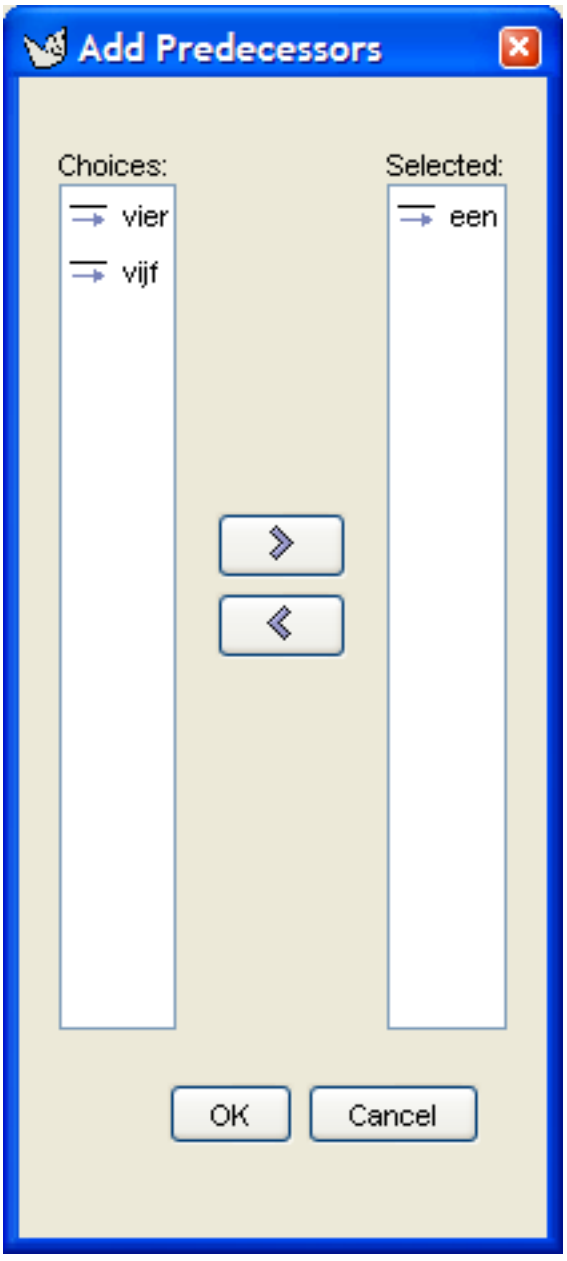
`Vorgänger`
Textbereich. Indentifiziert die Nachricht, deren Ausführungsabschluss diese Nachricht ermöglicht.

Ein Taste 1-Doppelklick navigiert zu der markierten Nachricht, Taste 2 öffnet ein Popup-Menü mit einem Eintrag.

- `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

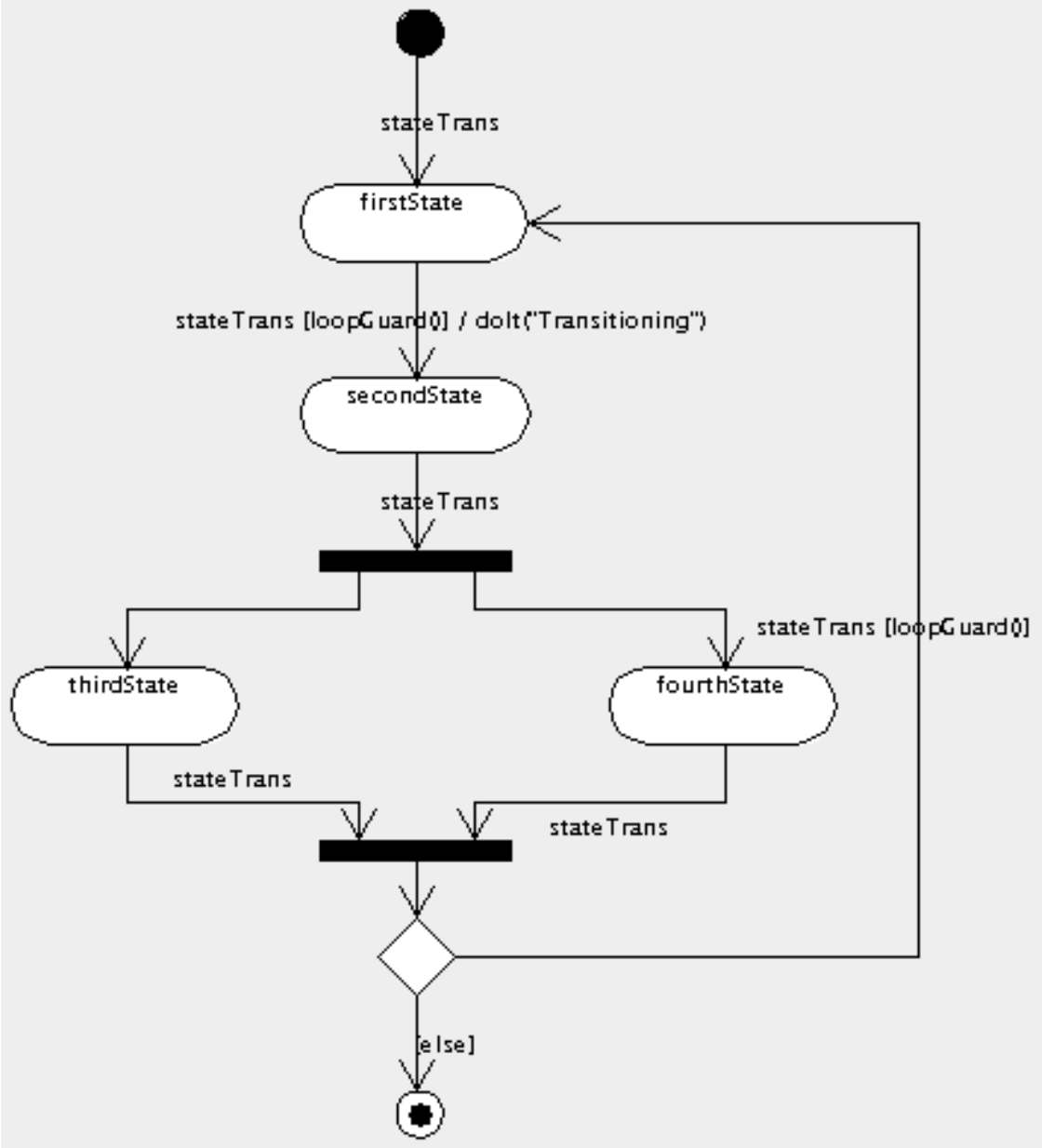
22.1. Einleitung

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

Aktion. Innerhalb des UML-Metamodelles ist `ActionState` eine Subklasse von `SimpleState`. Es ist ein spezialisierter einfacher Zustand, der nur eine Eintrittsaktion hat und mit einem impliziten Trigger, sobald die Aktion abgeschlossen ist.

Achtung

Als Konsequenz sollten für alle von einem Aktionszustand abgehenden Transitionen keine expliziten Trigger definiert sein (ArgoUML überprüft dies aktuell nicht). Sie dürfen Wächter haben, um eine Auswahl anzubieten, wenn es mehr als eine Transition gibt.

Anmerkung

Wie bei einem gewöhnlichen Zustand, einer internen Transition, sind bei Aktionszuständen eine Austrittaktion und eine Ausführenaktion nicht erlaubt.

Ein Aktionszustand wird in einem Aktivitätsdiagramm von ArgoUML als Rechteck mit abgerundeten Ecken dargestellt, welches den Namen des Aktionszustandes enthält.

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 ältern 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`
: Siehe Abschnitt 22.2.2, „ Eigenschaftssymbolleiste Aktionszustand “ und Abschnitt 22.2.3, „ Eigenschaftsfelder für einen Aktionszustand “ unten.

`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

Dies ist ein Löschen aus dem Modell, *nicht* nur aus dem Diagramm. Es ist nicht möglich, einen Aktionszustand aus dem Diagramm zu löschen, weil dieses Konzept nicht in den UML-Standard passt.

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`
Textfeld. Der Container des Aktionszustandes. Dies zeigt den sonst unsichtbaren zusammengesetzten Zustand ganz oben in der Umgebungshierarchie.

`Eintrittsaktion`
Textfeld. Zeigt den Namen der Aktion, die beim Eintritt in diese Aktion aufgerufen wird. Entsprechend dem UML-Standard ist ein Aktionszustand verpflichtet, eine Eintrittsaktion zu haben.

Ein Taste 1-Doppelklick navigiert zu dem gezeigten Eintrag, Taste 2 öffnet ein Popup-Menü mit zwei Einträgen.

- `Neu`. Fügt eine neue Eintrittsaktion eines bestimmten Typs hinzu. Dieses Menü hat die folgenden 7 Untermenüs, um den Aktionstyp auszuwählen: Aufrufaktion, Erzeugen-Aktion, Zerstören-Aktion, Rückgabe- Aktion, Sende-Aktion, Beenden-Aktion, Uninterpretierte Aktion.
- `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

Erinnern Sie sich, dass Aktionszustände keine expliziten Trigger haben. Die Transition ist implizit getriggert sobald das Eintrittsereignis des Aktionszustandes vollständig ist. Ein expliziter Trigger sollte daher nicht gesetzt werden.

Die aktuelle Release von ArgoUML überprüft derzeit nicht, ob diese Bedingung erfüllt ist.

Anmerkung

Transitionen zu und von Objektflusszuständen werden gepunktet, umd den *Objektfluss* vom *Steuerungsfluss* zu unterscheiden.

22.5. Wächter

Dieses Modellelement ist im Kontext des Zustandsdiagrammes beschrieben (siehe Abschnitt 20.10, „ Wächeter “).

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. Objektflusszustand

(Noch zu beschreiben)

Kapitel 23. Modellelement-Referenz Verteilungsdiagramm

23.1. Einleitung

Dieses Kapitel beschreibt jedes Modellelement, das innerhalb eines Verteilungsdiagrammes erzeugt werden kann. Beachten Sie, dass einige Submodellelemente von Modellelementen 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 Modellelementen verknüpft sind.

Innerhalb von ArgoUML werden Verteilungsdiagramme für Komponentendiagramme (z.B. ohne Instanzen, statische Abhängigkeiten von Komponenten darstellend) und Verteilungsdiagrammen (zeigen, wie Instanzen von Komponten durch Instanzen von Knoten zur Laufzeit gehandhabt werden).

Achtung

Verteilungsdiagramme sind in ArgoUML nicht vollständig entwickelt. Einige Aspekte sind nicht vollständig implementiert und können sich nicht wie erwartet verhalten. Beachtenswerte Auslassungen sind die Möglichkeit neue Schnittstellen zu zeichnen und richtiges stereotypisieren verschiedener Abhänigkeitsbeziehungen.

Abbildung 23.1, „ Denkbare Modellelemente in einem Komponentendiagramm. “ zeigt ein Komponentendiagramm mit allen denkbaren Modellelementen.

Abbildung 23.1. Denkbare Modellelemente in einem Komponentendiagramm.

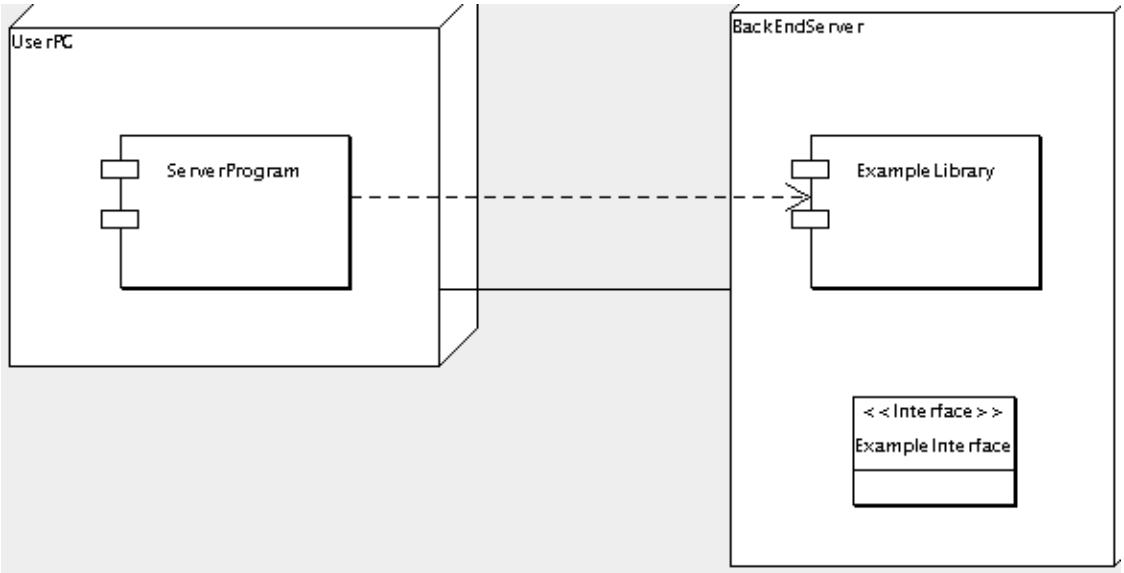
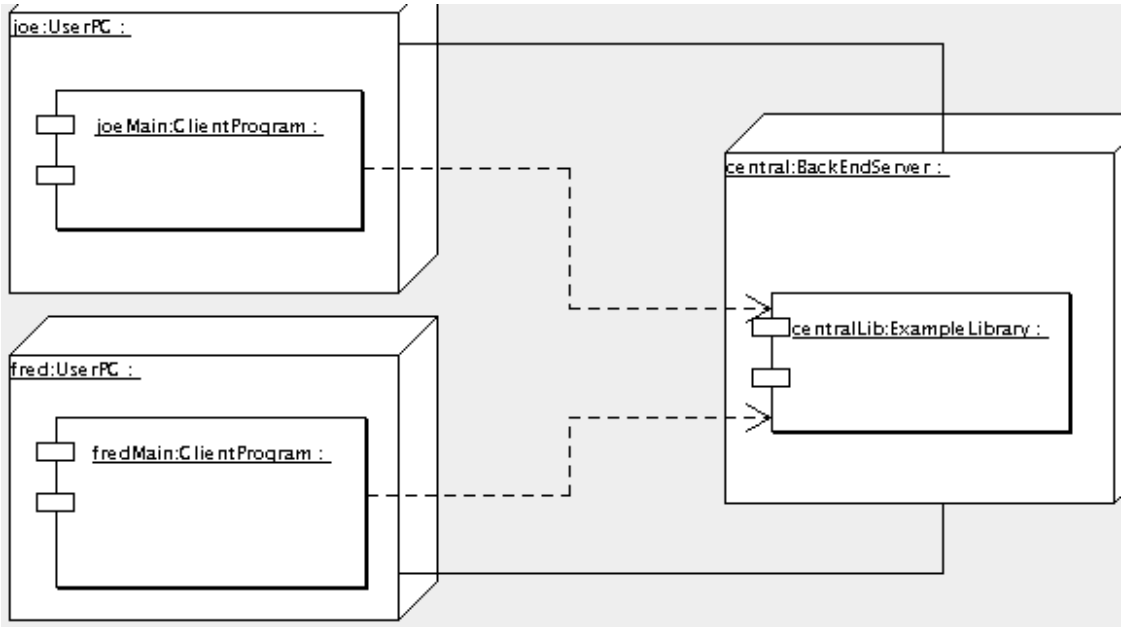


Abbildung 23.2, „ Denkbare Modellelemente in einem Verteilungsdiagramm. “ zeigt ein Verteilungsdiagramm mit allen denkbaren Modellelementen.

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`

Standard-Register.

`Eigenschaften`
Siehe Abschnitt 23.2.2, „ Eigenschaftssymbolleiste Knoten “ und Abschnitt 23.2.3, „ Eigenschaftsfelder für einen Knoten “ unten.

`Dokumentation`
Standard-Register.

`Darstellung`
Standard-Register. Das Feld `Begrenzung` definiert den umgebenden Kasten des Knoten im Diagramm.

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 Stereotyps.


`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`
Kombinationsfeld. Ein Knoten ist ein Klassifzierertyp und so sind weist er die Standard-Stereotypen eines Klassifzierers auf, wie im UML-Standard definiert. ArgoUML bietet die 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`
Kombinationsfeld. Erlaubt das Ändern des Namensraumes für diesen Knoten. Dies ist die Pakethierarchie.

`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.

Eine Knoteninstanz wird in einem Verteilungsdiagramm von ArgoUML als dreidimensionaler Kasten, der mit dem Knoteninstanznamen (falls vorhanden) und dem, durch einen Doppelpunkt (`:`) getrennten Knotentyp bezeichnet ist, dargestellt.

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`
Standard-Register. Das Feld `Begrenzung` definiert den umgebenden Kasten der Knoteninstanz im Diagramm.

Warnung

Vorsicht, in der Release 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, 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

Die UML-Metaklasse `Element`, von der alle anderen Modellelemente abgeleitet werden, beinhaltet den Eigenschaftswert `documentation`, der im Register *Dokumentation* von ArgoUML bearbeitet wird.

`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 Stereotyps.

 `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 verwenden die Groß-/Kleinschreibung, um Wörter innerhalb des Namens zu unterscheiden.

Anmerkung

ArgoUML erzwingt diese Namenskonvention nicht.

`Stereotyp`
Kombinationsfeld. Die Knoteninstanz hat standardmäßig im UML-Standard keine Stereotypen.

`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

Ein Komponententyp stellt eine ausführbares Stück Implemtierung eines Systems dar, einschliesslich Softwarecode (Quellcode, Binärcode oder ausführbare Datei), aber auch einschliesslich Geschäftsdokumente usw., in einem menschlichen System. Komponenten können dazu verwendet werden, Abhängigkeiten zu zeigen, solche wie Compiler und Laufzeit-Abhängigkeiten oder Informations-Abhängigkeiten in einer menschlichen Organisation. Im UML-Metamodell ist sie eine Subklasse von `Classifier`.

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`
: Siehe Abschnitt 23.4.2, „ Eigenschaftssymbolleiste Komponente “ und Abschnitt 23.4.3, „ Eigenschaftsfelder für eine Komponente “ unten.

`Dokumentation`
: Standard-Register.

`Darstellung`
: Standard-Register. Das Feld `Begrenzung` definiert den umgebenden Kasten der Komponente im Diagramm.

`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 Empfangssignales.


`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 erzwingt diese Namenskonvention nicht.

`Stereotyp`
Kombinationsfeld. Die Komponente bietet standardmäßig im UML-Standard die Stereotypen `document`, `executable`, `file`, `library` und `table`. ArgoUML bietet auch die Standard-Klassifzierer-Stereotypen `metaclass`, `powertype`, `process`, `thread` und `utility` an.

`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 Komponente auf und erlaubt des Ändern des Namensraumes. Dies ist die Pakethierarchie.

`Modifizierer`
Markierfelder mit den Einträgen `Abstract`, `Leaf` und `Root`.

- `Abstract` wird verwendet, um zu deklarieren, dass diese Komponente nicht instanziiert 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`
Textbereich. Listet abgehende Abhängigkeiten auf. Ein Taste 1-Doppelklick navigiert zu der Abhängigkeit.

`Notwendig für`
Textbereich. Listet ankommende Abhängigkeiten auf. Ein Taste 1-Doppelklick navigiert zu der Abhängigkeit.

`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.

Eine Komponenteninstanz wird in einem Sequenzdiagramm von ArgoUML als Kasten mit zwei schmalen Rechtecken, links hervorgehoben und mit dem Komponenteninstanznamen (falls vorhanden) und dem, durch einen Doppelpunkt (`:`) getrennten Komponententyp bezeichnet ist, 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`
Standard-Register. Das Feld `Begrenzung` definiert den umgebenden Kasten der Komponente im Diagramm.

`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

Die UML-Metaklasse `Element`, von der alle anderen Modellelemente abgeleitet werden, beinhaltet den Eigenschaftswert `documentation`, der im Register *Dokumentation* von ArgoUML bearbeitet wird.

`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 Stereotyps.

 `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 verwenden die Groß-/Kleinschreibung, um Wörter innerhalb des Namens zu unterscheiden.

Anmerkung

ArgoUML erzwingt diese Namenskonvention nicht.

`Stereotyp`
Kombinationsfeld. Die Komponenteninstanz hat standardmäßig im UML-Standard keine Stereotypen.

`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, einschliesslic 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 empfielt, 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 Klassifziererinstanzen 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 Hinweise enthalten.

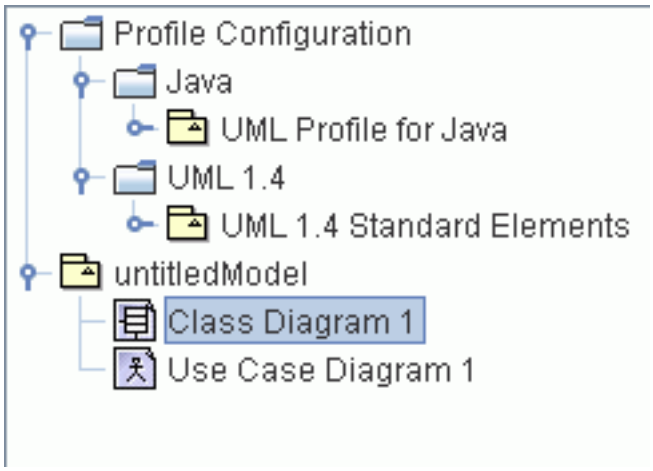
Datentypen, Aufzählungen, Klassen und Schnittstellen sind allgemein verfügbar und können überall genutzt werden, wo eine Klasse im Register Eigenschaften ausgewählt wurde.

Stereotypen und Tag-Definitionen werden häufig für applikationsspezifische Erweiterungen des Modelles verwendet, z.B., um Elemente zu unterscheiden, die unterschiedlich implementiert werden müssen.

Hinweise 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

Stereotyp	***Basiselement***
access	Permission
appliedProfile	Package
association	AssociationEnd
auxiliary	Class
become	Flow
call	Usage
copy	Flow
create	BehavioralFeature
create	CallEvent
create	Usage

Stereotyp	***Basiselement***
derive	Abstraction
destroy	BehavioralFeature
destroy	CallEvent
document	Abstraction
executable	Abstraction
facade	Package
file	Abstraction
focus	Class
framework	Package
friend	Permission
global	AssociationEnd
implementation	Class
implementation	Generalization
implicit	Association
import	Permission
instantiate	Usage
invariant	Constraint
library	Abstraction
local	AssociationEnd
metaclass	Class
metamodel	Package
modelLibrary	Package
parameter	AssociationEnd
postcondition	Constraint

Stereotyp	***Basiselement***
powertype	Class
precondition	Constraint
process	Classifier
profile	Package
realize	Abstraction
refine	Abstraction
requirement	Comment
responsibility	Comment
self	AssociationEnd
send	Usage
signalflow	ObjectFlowState
source	Abstraction
stateInvariant	Constraint
stub	Package
systemModel	Package
table	Abstraction
thread	Classifier
topLevel	Package
trace	Abstraction
type	Class

24.2.4. Tag-Definitionen

Dies sind die eingebauten Standard-Tag-Definitionen. Ihre Definition entnehmen Sie bitte dem UML-Standard.

- `derived`

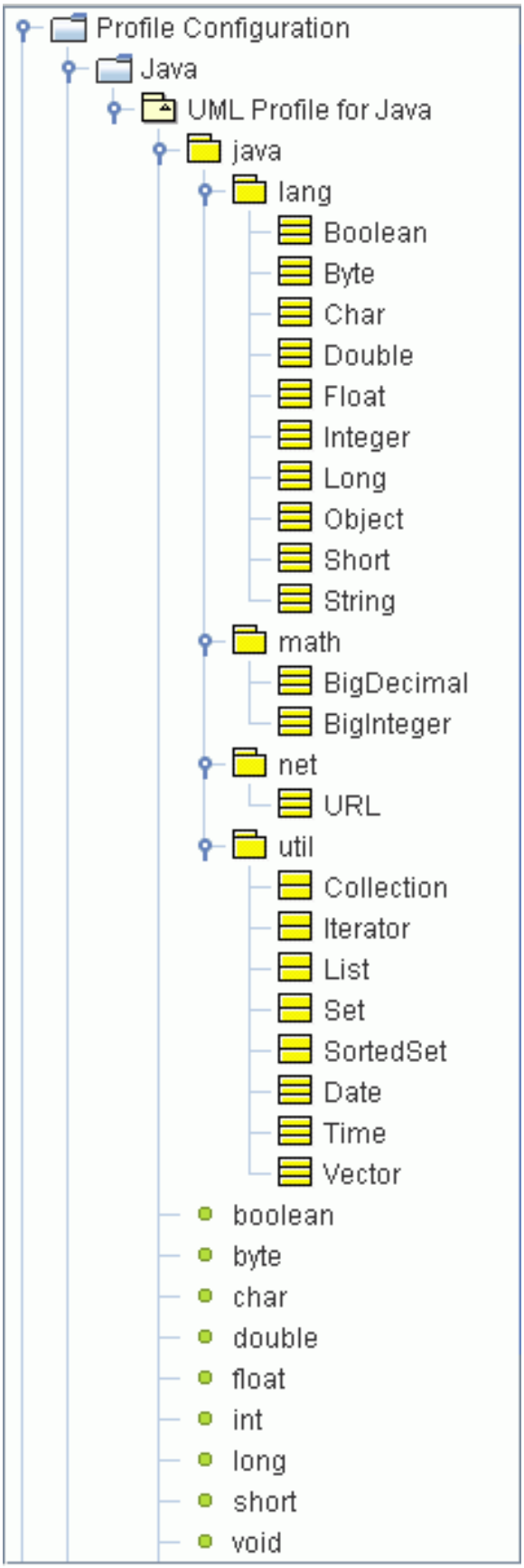
- documentation
- persistence
- persistent
- semantics
- usage

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.

Das Javaprofil ist als Hierarchie effektiv organisiert und in vier Pakete `lang`, `math`, `net` und `util` unterteilt und sind Subpakete von `java`. Dieses wiederum ist ein Subpaket des Modelles selbst. Abbildung 24.2, „ Hierarchie des Javaprofiles in ArgoUML “ zeigt diese Struktur.

Abbildung 24.2. Hierarchie des Javaprofiles in ArgoUML


Profile Configuration
Java
UML Profile for Java
java
lang
Boolean
Byte
Char
Double
Float
Integer
Long
Object
Short
String
math
BigDecimal
BigInteger
net
URL
util
Collection
Iterator
List
Set
SortedSet
Date
Time
Vector
boolean
byte
char
double
float
int
long
short
void

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.

- Big Decimal
- Big Integer

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

Aktivitätsdiagramm	Ein UML-Diagramm, welches das dynamische Verhalten eines Systems oder Subsystems beschreibt. Weitere Informationen, siehe Abschnitt 6.10, „ Aktivitätsdiagramme (Noch zu beschreiben) “.
Aktion	Verhalten, verknüpft mit *Zuständen* oder *Transitionen* in *Zustandsdiagrammen* . Diese Aktionen sind Aufrufe von *Methoden* und erscheinen in *Sequenz*- und *Kollaborationsdiagrammen*.
Akteur	Die Darstellung eines Agenten (animiert oder nicht animiert) in einem *Anwendungsfalldiagramm*, der ausserhalb des zu entwerfenden Systems steht.
Analyse	Analyse ist der Prozess die „Kunden“-Anforderungen aufzunehmen und diese aus der Perspektive der mutmaßlichen Lösung in die Sprache der mutmaßlichen Lösung umzuwandeln.
Anforderungserfassung	Die Anforderungserfassung ist der Prozess des Identifizierens was der „Kunde“ von dem gewünschten System wünscht. Eine ausführlichere Beschreibung, siehe Kapitel 4, *Erfassen der Anforderungen* .
Anwendungsfall	Eine UML-Notation, welche die Anforderungen an ein System oder Subsystem beinhaltet. Weitere Informationen, siehe Abschnitt 4.3, „ Ergebnis des Anforderungs-Erfassungs-Prozesses “.
Anwendungsfalldiagramm	Ein UML-Diagramm, welches die Beziehungen zwischen Akteuren und Anwendungsfällen darstellt. Weitere Informationen, siehe Abschnitt 4.3, „ Ergebnis des Anforderungs-Erfassungs-Prozesses “.
Anwendungsfallspezifikation	Das Dokument beinhaltet die detaillierten Anforderungen eines Anwendungsfalles.
Assoziationsklasse	Eine Klasse, die eine Assoziation zwischen zwei anderen Klassen charakterisiert.
Assoziation	Eine Beziehung zwischen zwei Klassen in einem Klassendiagramm oder zwischen Anwendungsfällen oder Anwendungsfällen und Akteuren in einem Anwendungsfalldiagramm.
Attribut (einer Klasse oder eines Objektes)	Ein Attribut einer Klasse oder eines Objektes ist eine Spezifikation eines Datenelementes, gekapselt durch dieses Objekt.

C

CASE

Computer Aided Software Engineering (Computergestützte Softwareentwicklung).

E

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.

F

Fenster

Ein Subfenster im Hauptfenster der ArgoUML-Anwenderschnittstelle.

G

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).

H

Hierarchisches Zustandsdiagramm

Ein *Zustandsdiagramm*, das untergeordnete Zustandsdiagramme mit individuellen *Zuständen* enthält.

I

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, „ Prozesstypen “.

J

Java

Eine vollständig objektorientierte Programmiersprache, die von Sun Microsystems eingeführt wurde. Strenger typisiert wie C++, übersetzt sie in einen interpretierenden Code, der Java Virtual Machine (JVM). Die JVM bedeutet, dass Javacode auf jeder Maschine ablaufen kann, auf der eine JVM implementiert ist.

Die wichtigsten Komponente von Java war die Integration der JVM in Webbrowsern, die es erlaubt, Code (Applets) herunterzulasen und über das Web auszuführen.

ArgoUML ist in Java geschrieben.

K

Klasse

Die Kapselung von Daten verbunden mit einem Modellelement (seine *Attribute*) und die Aktionen, die mit dem Modellelement verbunden sind (seine *Methoden*).

Eine Klasse spezifiziert die Charakteristika eines Modellelementes. Ein *Objekt* repräsentiert eine Instanz eines Modellelementes.

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, indem 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* .

Hinweis

Ein Prozess innerhalb von ArgoUML, der Hinweise anbietet, wie das Design verbessert werden kann. Die Hinweise basieren auf den Prinzipien der drei Theorien der kognitiven Psychologie *Reflektion während der Aktion*, *Opportunistisches Design* und *Verstehen und Problemlösung*.

M

Mealy Machine

Ein *Zustandsdiagramm*, in dem Aktionen mit *Zuständen* verküpft sind.

Methode (einer Klasse oder eines Objektes)

Eine Methode einer Klasse oder eines Objektes ist eine Spezifikation des Verhaltens, gekapselt durch das Objekt.

Moore Machine

Ein *Zustandsdiagramm*, in dem Aktionen mit *Transitionen* verknüpft sind.

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.

OMG

Die Object Management Group. Ein internationales Standardisierungsgremium der Industrie. Am Besten bekannt durch CORBA und UML.

OOA&D

Objektorientierte Analyse und Design. Ein Ansatz, um Softwareprobleme zu analysieren und basierend auf Objekten zu entwerfen, der sowohl die Daten als auch den Code kapselt. Siehe Abschnitt 1.1.1, „ Objektorientierte Analyse und Design “ oder jedes andere Buch über Software Engineering.

UML ist eine Notation, welche die OOA&D unterstützt.

Opportunistisches 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.

P

R

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

Eine Theorie innerhalb der kognitiven Psychologie, die beobachtete, dass Designer komplexer Systeme ein Design in seiner Gesamtheit nicht vollständig begreifen. Stattdessen müssen sie Teilentwürfe konstruieren, evaluieren, reflektieren und überarbeiten, bis sie in der Lage sind ihn zu erweitern. Als Entwickler arbeiten sie spielerisch mit dem Entwurf, dadurch verbessert sich ihr mentales Modell der Problemsituation und ihr Design.

S

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

Standard Graphical Markup Language. In ISO 8879:1986 definiert.

Simula 67

Eine prozedurale Programmiersprache für Simulationen. A procedural programming language intended for simulation. Aufgeführt wegen seiner Einführung in *Objekte* und *Co-Routinen*.

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.

Ein Stereotyp definiert einen Namensraum innerhalb des Entwurfes. Beispiele für Stereotypen sind `<<business>>` und `<<realization>>` für Anwendungsfälle, um zwischen Anwendungsfällen der Anforderungsphase, die in den Begrifflichkeiten des Problembereiches definiert sind und den Anwendungsfällen der Analysephase, die in den Begrifflichkeiten des Lösungsbereiches definiert wurden, zu unterscheiden.

SVG

Skalierbares Vektor-Grafik-Format. Eine Standarddarstellung von grafischen Diagrammen die Vektoren verwendet. ArgoUML kann Diagramme in SVG exportieren.

Systemsequenzdiagramm

Ein *Sequenzdiagramm* zeigt in der Analysephase das dynamische Verhalten des Systems als Ganzes. Weitere Informationen, siehe Kapitel 5, *Analyse* .

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

Universal Modeling Language. Eine grafische Notation für OOA&D- Prozesse, standardisiert durch die OMG. ArgoUML unterstützt UML 1.4. UML 2.0 befindet sich in den abschliessenden Runden der Standardisierung und sollte im Verlauf des Jahres 2006 vollständig sein.

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:

1. Mehrere Darstellungen wie die syntaktische Zerlegung eines Programmes, Zustandstransitionen, Steuerflüsse und Datenflüsse. Diese erlauben es dem Programmierer die Elemente und Beziehungen des Problemes und der Lösung besser zu identifizieren und können somit die Zuordnung zwischen ihren Situationsmodellen und dem ausführenden Systemmodellen leichter herstellen.

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

Das World Wide Web Konsortium, www.w3c.org [http://www.w3c.org]. Ein internationales Standardisierungsgremium für alle Dinge, die mit dem Word Wide Web zu tun haben.

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)

Noch zu beschreiben...

A.2.1. Visionsdokument (Noch zu beschreiben)

Noch zu beschreiben...

A.2.2. Anwendungsfall-Spezifikationen (Noch zu beschreiben)

Noch zu beschreiben...

A.2.2.1. UC Spezifikation 1 (Noch zu beschreiben)

Noch zu beschreiben...

A.2.3. Ergänzende Anforderungsspezifikation (Noch zu beschreiben)

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.

Weil UML und C++ objektorientiert sind, gibt es eine offensichtliche Korrespondenz zwischen den UML-Modellelementen und den strukturellen C++-Konstrukten, z.B. die UML-`Klasse` entspricht der C++-`Klasse`. Diese offensichtlichen Beziehungen werden hier nicht beschrieben, weil angenommen wird, dass ein ArgoUML- Anwender, der für C++ modellieren will, ein grundlegendes Wissen über C++ und UML hat.

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 die Modellelemente 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 freiem 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`
: `Directory` - sucht im angegebenen Verzeichnis nach den Templatedateien "header_template" und "cpp_template", die oben in die entsprechende Datei eingefügt werden. Die folgenden Tags der Templatedatei werden durch folgende Modellwerte ersetzt: |FILENAME|, |DATE|, |YEAR|, |AUTHOR|, |EMAIL|. Wenn kein solches Tag spezifiziert wurde, werden die Templates im Unterverzeichnis des Wurzelverzeichnisses für die Codegenerierung gesucht.

`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`
: `header` - wird zu Klassentypen mit einer Vordeklaration im Header führen, und zum Einfügen des Remote-Klassenheaders (-kopfes) in den Header (Kopf) der generierten Klasse.

`MultiplicityType`
: `list || slist || vector || map || stack || stringmap` - wird eine Kardinalität als korrespondierenden STL-Container definieren, wenn der `Kardinalitäts`-Bereich 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` auf 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.

Dieses Problem wird unter issue #3553 - handle multiple return parameters [http://argouml.tigris.org/issues/show_bug.cgi?id=3553] behandelt.

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:

```
 function Testclass::Testclass()
 // section -64--88-0-40-76f2e8:ec37965ae0:-7fff begin
{
}
 // section -64--88-0-40-76f2e8:ec37965ae0:-7fff end
```

Der gesamte Code, der von Ihnen zwischen den "begin"- und "end"-Zeilen eingefügt wird, bleibt erhalten, wenn Sie den Code erneut generieren. Bitte ändern Sie nichts innerhalb dieser Zeilen, weil die Abschnitte durch diese Kommentarsyntax erkannt werden. Da die, geschweiften Klammern innerhalb des geschützten Bereiches platziert wurden, bleibt auch die Attributinitalisierung in den Konstruktoren erhalten.

Die passiert auch, wenn Sie die Methodennamen nach dem Generieren ändern.

```
 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

Wie bei allen Produkten hat auch ArgoUML einige Beschränkungen. Die für den Anwender wichtigen, sind in diesem Abschnitt aufgelistet.

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:

> Copyright (c) <Jahr> von Name des <Autors> oder <Zeichnungsberechtigten>. Dieses Material darf nur unter den Voraussetzungen und den Bedingungen, die in der Open Publication Lizenz, vX.Y oder später (die letzte Version ist aktuell verfügbar unter http://www.opencontent.org/openpub/ [http://www.opencontent.org/openpub/]) beschrieben sind, verteilt werden.

Die kommerzielle Weiterverbreitung von Open Publication lizensiertem Material ist erlaubt.

Jegliche Publikation im Standard-(Papier-) Buch-Format erfordert das Zitieren der Original-Herausgeber und Autoren. Der Namen der Herausgeber und Autoren erscheinen auf allen äußeren Deckflächen des Buches. Auf allen äußeren Deckflächen des Buches soll der Name des Original-Herausgebers genauso groß sein, wie der Titel der Arbeit und so hervorgehoben benannt werden wie der Titel.

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.

Das Zusammenfassen von Open-Publication-Arbeiten oder Teilen einer Open-Publication-Arbeit mit anderen Arbeiten oder Programmen auf dem gleichen Medium, kann nicht dazu führen, dass diese Lizenz auf diese anderen Arbeiten anzuwenden ist. Die zusammengefaßte Arbeit sollte einen Hinweis enthalten, der auf die Einbeziehung von Open-Publication-Material hinweist und eine geeignete Copyright-Vermerk enthält.

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

eine Garantie der Nicht-Verletzung.

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.
3. Danksagungen der Original-Autoren und Herausgeber müssen ggf. beibehalten werden und den üblichen akademischen Zitat-Praktiken entsprechen.
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 modifzierten 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.
3. Schließlich, obwohl nicht erforderlich unter dieser Lizenz, wird es als guter Stil angesehen eine kostenlose Kopie einer unter Open Publication lizensierten Arbeit als Hardcopy- und CD-ROM an den Autor(en) zu senden.

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.

Kent Beck und Ward Cunningham führten die CRC-Karten in ihres Papiers "Ein Labor für die Lehre im Objektorientierten Denken" ein, die auf der OOPSLA-Konferenz (Objektorientierten Programmierung, Systeme, Sprachen (Languages) & Anwendungen) 1989 vorgestellt wurde. Ein Tutorial zum Thema finden Sie unter http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/crc_b/. Der CRC-Karten-Methode wurde ursprünglich als ein pädagogisches Instrument entworfen, aber es hat sich genauso als gutes Modellierungswerkzeug erwiesen.

Die drei Teile der CRC Abkürzung wurden von den Autoren des Papiers genutzt, um die wesentlichen Aspekte der objektorientierten Modellierung darzustellen. Der Begriff bezieht sich auf den Vertrag, den die in der Diskussion befindliche Klasse dem Rest der Welt (Interface-und Vertrag sind ähnliche Konzepte) bietet. Verantwortlichkeiten modellieren die Dinge, die eine Klasse tun kann. Dienstleistungen, Methoden oder Operationen ergeben sich aus diesen. Der Begriff Kollaboratoren bezieht sich Klassen, deren Dienste die in der Diskussion stehende Klasse nutzen wird. Kent Beck versuchte erfolglos den Begriff Helfer anstelle von Kollaboratoren zu verwenden, um Klassen darzustellen, die die in der Diskussion stehende Klasse unterstützt. Es ist allgemeine Ansicht, dass der Begriff gewählt wurde, weil CRC die Initialen von Ward Cunningham's Sohn sind.

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 ideale Gruppengröße für eine CRC-Karten-Sitzung beträgt fünf oder sechs Personen. Die Größe erlaubt es jedem, produktiv an der Sitzung teilzunehmen. In größeren Gruppen ist die Produktivität durch mehr Mißverständnisse eingeschränkt und die individuelle Teilnahme ist geringer. Wenn es mehr als sechs Personen sind, ist ein mögliche Lösung, die zusätzliche Person als Beobachter einzusetzen.

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.

Weisen Sie jede Klasse einem Gruppenmitglied zu. Jede Person sollte für mindestens eine Klasse verantwortlich sein. Sie sind der Eigentümer der Klasse in dieser Sitzung. Jede Person schreibt den Namen Ihrer Klasse auf die Karte. Eine Klasse pro Karte.

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

A

B

C

D

E

F

O

P

Q

R

S

T

U

V

W

X

Z