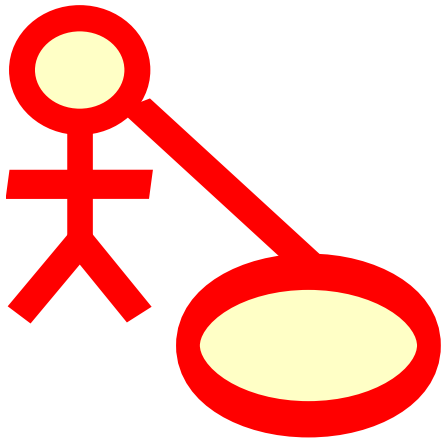
Das Handbuch zu Umbrello UML Modeller



Das Handbuch zu Umbrello UML Modeller

Inhaltsverzeichnis

Zusammenfassung

Umbrello UML Modeller unterstützt den Software Entwicklungsprozess durch die Anwendung des Industriestandards Unified Modelling Language (UML). Dadurch kann man mithilfe von Diagrammen das System entwickeln und dokumentieren.

Kapitel 1

Einführung

Umbrello UML Modeller ist eine Anwendung für UML-Diagramme, welches den Entwicklungsprozess von Software unterstützen kann. Hauptsächlich während der Analyse und des Design hilft Umbrello UML Modeller dabei, ein qualitativ hochwertiges Produkt zu erzeugen. Die UML kann weiterhin zur Dokumentation des Softwaredesigns genutzt werden, um sie und ihre Kollegen zu unterstützen.

Ein gutes Modell des zukünftigen Softwaresystems ist die beste Grundlage, um mit anderen daran arbeitenden Entwicklern oder mit dem Kunden zu kommunizieren. Ein gutes Modell ist extrem wichtig für mittlere und große Projekte, aber selbst in kleinen Projekten ist es hilfreich. So kann man es selbst in einem Ein-Mann-Projekt einsetzen. Kleine Projekte profitieren von einem guten Modell ebenso, da man einen Überblick erhält, was dabei hilft die Sache gleich beim ersten Versuch richtig umzusetzen.

Die UML ist eine Diagrammnotationssprache um solche Modelle zu beschreiben. Man kann seine Ideen mithilfe verschiedener Diagramme in der UML ausdrücken. Umbrello UML Modeller 2.11 unterstützt folgende Typen:

- Klassendiagramm
- Sequenzdiagramm
- Kollaborationsdiagramm
- Anwendungsfalldiagramm
- Zustandsdiagramm
- Aktivitätsdiagramm
- Komponentendiagramm
- Verteilungsdiagramm
- Entitäten-Beziehungs-Diagramm

Mehr Informationen über UML kann man auf der OMG Homepage http://www.omg.org/ finden. Der UML-Standard wurde von dieser Organisation geschaffen.

Wir hoffen sie haben Spaß mit Umbrello UML Modeller und das es ihnen bei der Erstellung qualitativ hochwertiger Software hilft. Umbrello UML Modeller ist ein freies Werkzeug. Die einzige Sache, um die wir sie bitten, ist es, Fehler, Probleme oder Vorschläge an die Umbrello UML Modeller-Entwickler zu berichten über umbrello-devel@kde.org oder https://bugs.kde.org!

Kapitel 2

UML-Grundlagen

2.1 Über UML

Dieses Kapitel gibt einen kurzen Überblick über die UML-Grundlagen. Man sollte sich aber bewusst sein, dass es keine vollständige UML-Anleitung ist. Wenn man mehr über die Unified Modelling Language, oder allgemein über Software Analyse und Design, lernen will, dann sollte man eines der vielen zu diesem Thema publizierten Bücher lesen. Man kann natürlich die vielen Einführungen im Internet zum Thema als Ausgangspunkt wählen.

Die Unified Modelling Language (UML) ist eine Diagrammnotation zum spezifizieren, visualisieren und dokumentieren von Modellen objektorientierter Softwaresysteme. UML ist kein Vorgehensmodell, d.h. es sagt nichts über die einzelnen Schritte aus, die nötig sind, um ein System zu gestalten. Aber UML hilft ihnen ihr Design zu visualisieren und mit anderen zu kommunizieren. Der UML-Standard wird von der Object Management Group (OMG) gepflegt und ist der Industriestandard zur Beschreibung von Softwaremodellen.

UML wurde für das objektorientierte Softwaredesign entwickelt und ist daher nur von begrenztem Nutzen bei anderen Programmierparadigmen.

UML setzt sich zusammen aus vielen Modellelementen, die für sich einen bestimmten Sachverhalt des Softwaresystems repräsentieren. Diese Elemente werden zu Diagrammen kombiniert, die einen Ausschnitt oder einen bestimmten Blickpunkt auf das System darstellen. Folgende Diagrammtypen werden von Umbrello UML Modeller unterstützt:

- *Anwendungsfalldiagramm* , stellt Akteure (Menschen oder andere Benutzer des Systems), Anwendungsfälle (Szenario, wie die Akteure das System nutzen) und die Beziehungen dazwischen dar
- *Klassendiagramm*, stellt Klassen und ihre Beziehungen untereinander dar
- *Sequenzdiagramm*, stellt Objekte und ihre Beziehungen dar, wobei der Schwerpunkt auf dem chronologischen Nachrichtenaustausch zwischen den Objekten liegt
- *Kollaborationsdiagramm* , stellt Objekte und ihre Beziehungen dar, wobei der Schwerpunkt auf Objekten liegt, die am Nachrichtenaustausch beteiligt sind
- *Zustandsdiagramm* , stellt Zustände, Zustandsänderungen und Ereignisse in einem Objekt oder Teilsystem dar
- *Aktivitätsdiagram* , stellt die Aktivitäten, Zustände und Zustandsänderungen von Objekten und die Ereignisse in Teilsystemen dar
- *Komponentendiagramm* , stellt die höheren Programmierkomponenten (wie KParts und Java Beans) dar.

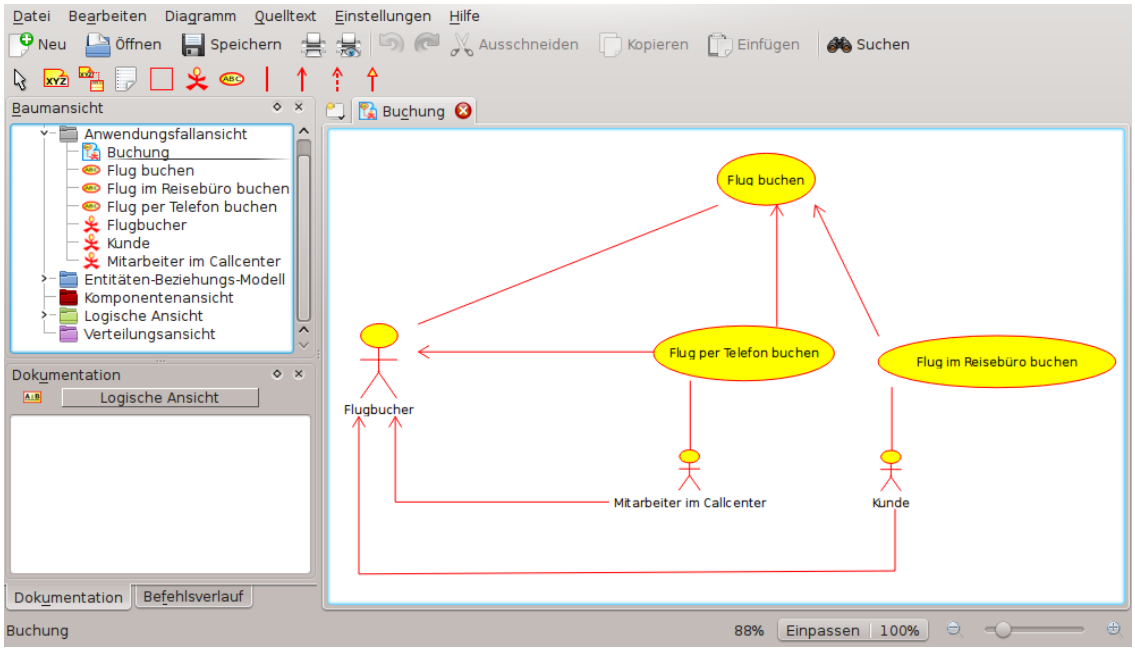
- *Verteilungsdiagramm* , stellt die Instanzen der Komponenten und ihre Beziehungen dar.
- *Entitäten-Beziehungs-Diagramm*, stellt Daten und ihre Beziehungen beziehungsweise Einschränkungen untereinander dar.

2.2 UML-Elemente

2.2.1 Anwendungsfalldiagramm

Anwendungsfalldiagramme beschreiben die Beziehungen und Abhängigkeiten zwischen einer Gruppe von *Anwendungsfällen* und den teilnehmenden Akteuren in einem Prozess.

Dabei ist zu beachten, dass ein Anwendungsfalldiagramm nicht das Systemdesign widerspiegelt und damit keine Aussage über die Systeminterna trifft. Anwendungsfalldiagramme werden zur Vereinfachung der Kommunikation zwischen Entwickler und zukünftigen Benutzer bzw. Kunde erstellt. Sie sind vor allem bei der Festlegung der benötigten Kriterien des zukünftigen Systems hilfreich. Somit treffen Anwendungsfalldiagramme eine Aussage, *was* zu tun ist, aber nicht *wie* wie das erreicht wird.



Umbrello UML Modeller bei der Darstellung eines Anwendungfalldiagramms

2.2.1.1 Anwendungsfall

Ein *Anwendungsfall* beschreibt, aus der Sicht des Akteurs eine Reihe von Aktivitäten im System, die ein konkret fassbares Ergebnis liefern.

Anwendungsfälle dienen als Beschreibung von typischen Interaktionen zwischen Benutzer und dem System. Sie stellen die externe Schnittstelle dar und spezifizieren damit die Anforderungen, was das System zu tun hat (nur das was, aber nicht das wie!).

Bei der Arbeit mit Anwendungsfällen, sollte man folgende einfache Regeln beachten:

- jeder Anwendungsfall ist mindestens mit einem Akteur verbunden

- jeder Anwendungsfall hat einen Auslöser (also einen Akteur)
- jeder Anwendungsfall führt zu einem relevanten Ergebnis („messbar und wirtschaftlich relevant")

Anwendungsfälle können untereinander in Verbindung stehen. Die 3 gebräuchlichsten Beziehungen zwischen Anwendungsfällen sind:

- *«include»*, wodurch ausgesagt wird, dass ein Anwendungsfall *in* einem anderen Anwendungsfall stattfindet.
- *«extends»*, wodurch ausgesagt wird, dass in einer bestimmten Situation (oder einem bestimmten Erweiterungspunkt) der Anwendungsfall durch einen anderen erweitert wird.
- *Verallgemeinerung*, wodurch ausgesagt wird, dass ein Anwendungsfall die Eigenschaften eines „Super"anwendungsfalls (übergeordneten) erbt und diese überschreiben oder erweitern kann, ganz ähnlich wie bei der Vererbung zwischen Klassen.

2.2.1.2 Akteur

Ein Akteur ist ein externes Objekt (außerhalb des Systems), welches mit dem System durch die Teilnahme und Auslösung von Anwendungsfällen in Kontakt tritt. Akteure können echte Menschen (z. B. der Benutzer des Systems), Computersysteme oder externe Ereignisse sein.

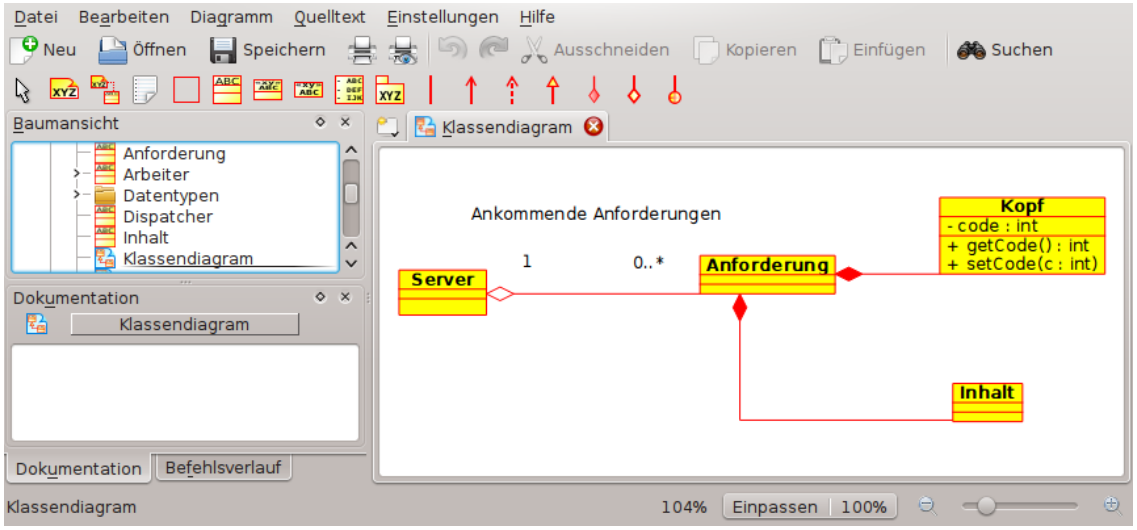
Akteure stellen somit nicht die *physische* Person oder System dar, sondern vielmehr die *Rollen* dieser Objekte. Steht eine physische Person auf verschiedenen Wegen (z. B. durch verschiedene Rollen) mit dem System in Kontakt, dann wird sie durch verschiedene Akteure dargestellt. So würde eine Person, die im Kundensupport genauso wie in der Auftragsannahme tätig ist, einmal als Akteur „Supportmitarbeiter" und einmal als Akteur „Vertriebsmitarbeiter" dargestellt werden.

2.2.1.3 Anwendungsfallbeschreibung

Eine Anwendungsfallbeschreibung legt in Textform den Anwendungsfall dar. Normalerweise werden dazu Notizen oder sonst wie mit dem Anwendungsfall verlinkte Dokumente dargestellt und beschreiben die Prozesse oder Aktivitäten, die im Anwendungsfall stattfinden.

2.2.2 Klassendiagramm

Klassendiagramme zeigen die verschiedenen Klassen, aus denen das System besteht, und wie diese untereinander in Abhängigkeit stehen. Die Klassendiagramme werden als „statisch" bezeichnet, da sie lediglich die Klassen mit ihren Methoden und Attributen sowie die statischen Verbindungen zwischen ihnen darstellen. Dabei wird gezeigt, welche Klassen von anderen Klassen „etwas wissen" und welche Klassen „ein Teil" von anderen Klassen sind. Es wird aber nicht der Nachrichtenaustausch (Methodenaufrufe) zwischen den Klassen dargestellt.

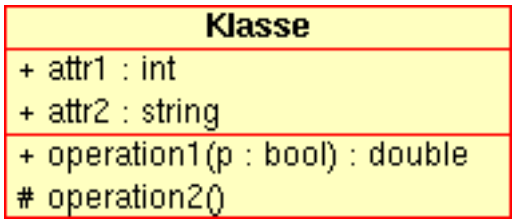


Umbrello UML Modeller bei der Darstellung eines Klassendiagramms

2.2.2.1 Klasse

Eine Klasse definiert die Attribute und Methoden einer Menge von Objekten. Alle Objekte dieser Klasse (die Instanzen) haben das gleiche Verhalten und die gleichen Attribute (aber mit unterschiedlichen Werten). Der Begriff „Typ" wird manchmal synonym für Klasse verwendet, aber es muss beachtet werden, dass Typ allgemeiner ist und daher die beiden Begriffe nicht identisch in ihrer Bedeutung sind.

In der UML werden Klassen als Rechtecke mit dem Klassennamen dargestellt. Sie können die Attribute und Operationen der Klasse in 2 extra „abgetrennten Bereichen" innerhalb des Rechteck enthalten.



visuelle Darstellung einer Klasse in der UML

2.2.2.1.1 Attribut

In der UML werden Attribute mindestens mit ihrem Namen dargestellt, es können aber noch der Typ, der Anfangswert und weitere Eigenschaften mit angezeigt werden. So kann man z. B. die Sichtbarkeit von Attributen mit darstellen:

- +, steht für *public* (öffentliche) Attribute
- #, steht für *protected* (geschützte) Attribute
- –, steht für *private* Attribute

2.2.2.1.2 Operation

Operationen (Methoden) müssen ebenfalls mindestens mit ihrem Namen dargestellt werden und können weiterhin die Parameter und die Rückgabewerte in der Darstellung enthalten. Wie bei Attributen, ist die Sichtbarkeit ebenfalls darstellbar:

- +, steht für *public* (öffentliche) Operationen
- #, steht für *protected* (geschützte) Operationen
- -, steht für *private* Operationen

2.2.2.1.3 Klassen-Templates

Klassen können als Klassen-Templates genutzt werden um zu zeigen, dass es sich um eine unspezifizierte Klasse oder Typ handelt. Der Klassentyp wird während der Instanzbildung (also Objekt wird angelegt) bestimmt. Template-Klassen gibt es in modernen Sprachen wie C++ und ab Java 1.5, wo sie Generics genannt werden.

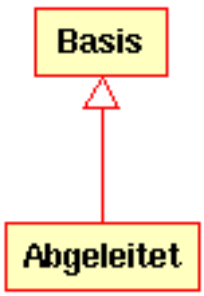
2.2.2.2 Klassenassoziation

Klassen können sich auf unterschiedlichen Wegen aufeinander beziehen (assoziieren):

2.2.2.2.1 Verallgemeinerung

Die Vererbung ist ein Basiskonzept der objektorientierten Programmierung. Ein Klasse „erhält" dabei alle Attribute und Operationen der Klasse, von der sie abgeleitet wird. Die Klasse kann diese Operationen/Attribute überschreiben und ändern, sowie neue hinzufügen.

In der UML wird durch die Assoziation *Verallgemeinerung* zwischen 2 Klassen eine Hierarchie aufgebaut, die das Konzept von abgeleiteter Klasse und Basisklasse verkörpert. Die Verallgemeinerung zwischen 2 Klassen wird in der UML durch eine Linie zwischen den 2 Klassen dargestellt, wobei sich ein Pfeil auf der Seite der Basisklasse befindet.



visuelle Darstellung der Verallgemeinerung in UML

2.2.2.2.2 Assoziation

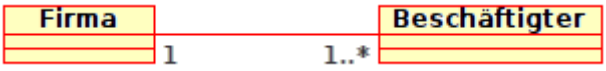
Eine Assoziation stellt den Zusammenhang zwischen Klassen dar und beschreibt damit die allgemeine Bedeutung und Struktur verschiedenster Typen von „Verbindungen" zwischen Objekten.

Assoziationen sind der Mechanismus, der es den Objekten erlaubt untereinander zu kommunizieren. Sie beschreiben die Verbindungen zwischen verschiedenen Klassen (die Verbindung zwischen den eigentlichen Objekten werden als Objektverbindungen oder als *Link* bezeichnet).

Assoziationen können Rollen haben, die den Zweck der Verbindung beschreiben und entweder uni- oder bidirektional sind (ob die Verbindung zwischen den Objekten ein- oder zweiseitig ist).

Beide Enden einer Assoziation verfügen über einen Multiplizitätswert, der angibt, wieviele Objekte auf der einen Seite mit wieviel Objekten auf der anderen Seite verbunden sein können.

In der UML wird die Assoziation durch eine Linie zwischen den in der Beziehung teilnehmenden Klassen dargestellt. Dabei kann die Rolle und die Multiplizität ebenfalls angezeigt werden. Multiplizität wird als Bereich [min..max] von nicht negativen Werten dargestellt, wobei der Stern (*) auf der Maximumseite einen unendlichen Wert repräsentiert.



visuelle Darstellung einer UML-Assoziation

2.2.2.2.3 Aggregation

Aggregationen sind eine Spezialart von Assoziationen. Dabei haben die verbundenen Klassen nicht den gleichen Status, sondern es wird eine „Teil eines Ganzes" Beziehung dargestellt. Ein Aggregation beschreibt, wie die Klasse, die die Rolle des Ganzen vertritt, sich aus anderen Klassen, die die Rollen des Teil innehaben, zusammensetzt. Bei Aggregationen hat die Klasse mit der Rolle des Ganzen immer eine Multiplizität von 1.

In der UML werden Aggregationen durch Assoziationen dargestellt, wobei auf der Seite des Ganzen ein Rhombus ist.

visuelle Darstellung einer Aggregationsbeziehung in der UML

2.2.2.2.4 Komposition

Kompositionen sind Assoziationen, die eine *sehr starke* Aggregation darstellen. Das bedeutet, dass Kompositionen ebenfalls eine Teil eines Ganzen Beziehung darstellen, wobei die Beziehung aber so stark ist, dass die Teile nicht allein existieren können. Sie bestehen nur innerhalb des Ganzen und werden zerstört, wenn das Ganze zerstört wird.

In der UML werden Kompositionen durch ein ausgefülltes Rhombus auf der Seite des Ganzen dargestellt.

2.2.2.3 Andere Klassendiagramm-Elemente

Neben Klassen können weitere Elemente in einem Klassendiagramm dargestellt werden.

2.2.2.3.1 Schnittstelle

Schnittstellen (Interfaces) sind abstrakte Klassen. Das bedeutet, von der Klasse können keine Instanzen direkt gebildet werden. Schnittstellen können Operationen aber keine Attribute beinhalten. Klassen können durch eine Realisierungsassoziation von Schnittstellen abgeleitet werden und von der Klasse können dann Instanzen gebildet werden.

2.2.2.3.2 Datentypen

Datentypen sind Primitive, die normalerweise in den Programmiersprachen verfügbar sind wie Integer und Boolean. Sie können keine Beziehung zu Klassen haben, aber Klassen zu ihnen.

2.2.2.3.3 Aufzählungen

Aufzählungen sind eine einfache Liste von Werten. Ein typisches Beispiel für eine Aufzählung sind die Wochentage. Die einzelnen Einträge in der Aufzählung werden als Aufzählungswert bezeichnet. Wie Datentypen können sie keine Beziehung zu Klassen haben, aber Klassen können Beziehungen zu ihnen haben.

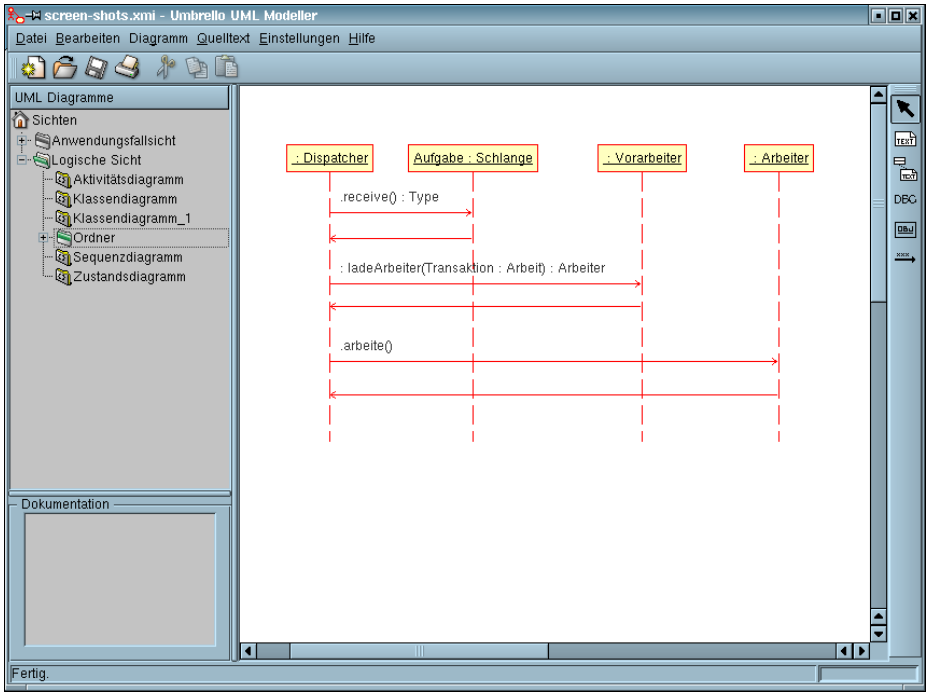
2.2.2.3.4 Pakete

Pakete stehen für Namensräume (Namespace) in Programmiersprachen. In einem Diagramm stehen sie stellvertretend für Teilsysteme, die aus mehr als einer Klasse, manchmal hunderte, bestehen können.

2.2.3 Sequenzdiagramm

Sequenzdiagramme zeigen den Nachrichtenaustausch (also Methodenaufruf) zwischen Objekten in einem spezifischen Zeitrahmen. Dabei wird besondere Betonung auf die Reihenfolge und die Zeiten gelegt, in denen die Nachrichten an die Objekte gesendet werden.

Im Sequenzdiagramm wird das Objekt durch eine vertikal verlaufende gestrichelte Linie gekennzeichnet. Der Objektname befindet sich am oberen Ende. Die Zeitachse verläuft ebenfalls vertikal, wobei sie sich nach unten hin erhöht. Nachrichten zwischen den Objekten werden als Pfeile mit den Operationen- und Parameternamen gekennzeichnet.



Umbrello UML Modeller bei der Darstellung eines Sequenzdiagramms

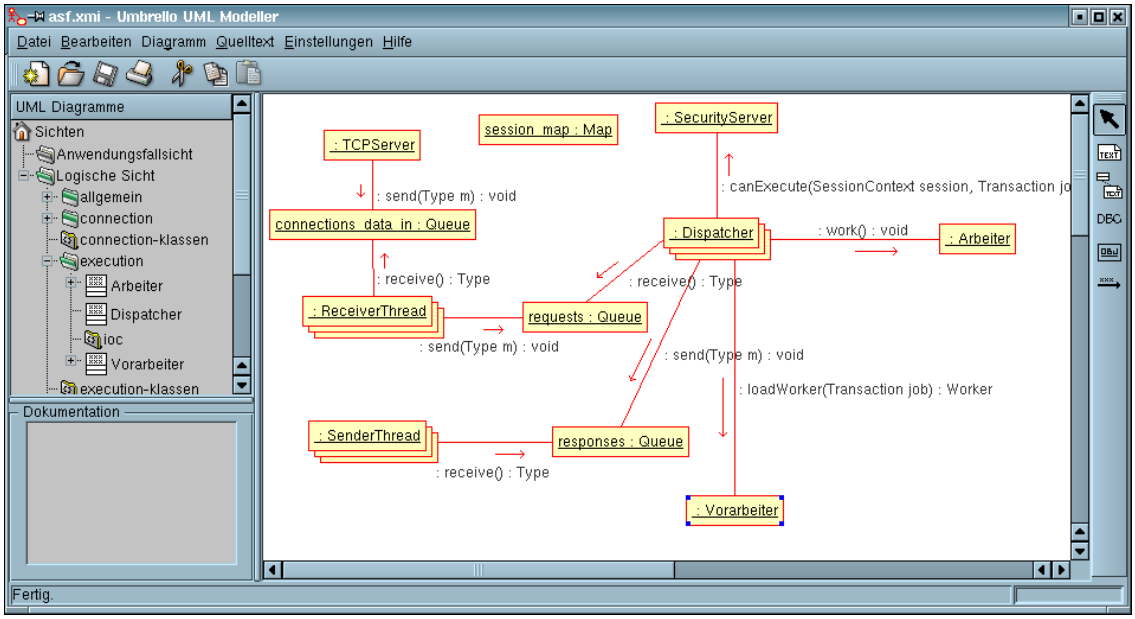
Nachrichten können synchron oder asynchron sein. Bei synchronen Nachrichten erfolgt ein normaler Nachrichtenaufruf und die Kontrolle wird an das gerufene Objekt übergeben. Asynchrone

Aufrufe geben sofort die Kontrolle an das rufende Objekt zurück. Synchrone Nachrichten haben eine vertikale Box auf der Seite des gerufenen Objektes um den Programmverlauf darzustellen.

2.2.4 Kollaborationsdiagramm

Kollaborationsdiagramme zeigen die Interaktionen zwischen Objekten, die an einer spezifischen Situation teilnehmen. Dies entspricht im Wesentlichen den Informationen aus dem Sequenzdiagramm, wobei bei diesen der Schwerpunkt auf dem zeitlichen Auftreten liegt. Bei Kollaborationsdiagramm wird hingegen vordergründig die Verbindung zwischen Objekten und ihrer Topologie gelegt.

Die Nachrichten zwischen den Objekten werden in Kollaborationsdiagrammen durch Pfeile dargestellt, an denen Name, Parameter und die Nachrichtensequenz steht. Die Kollaborationsdiagramme sind hervorragend dafür geeignet eine spezielle Programmabfolge oder Situation zu zeigen. Man kann damit sehr leicht und schnell einen Teil der Programmlogik demonstrieren und erklären.



Umbrello UML Modeller bei der Darstellung eines Kollaborationdiagramms

2.2.5 Zustandsdiagramm

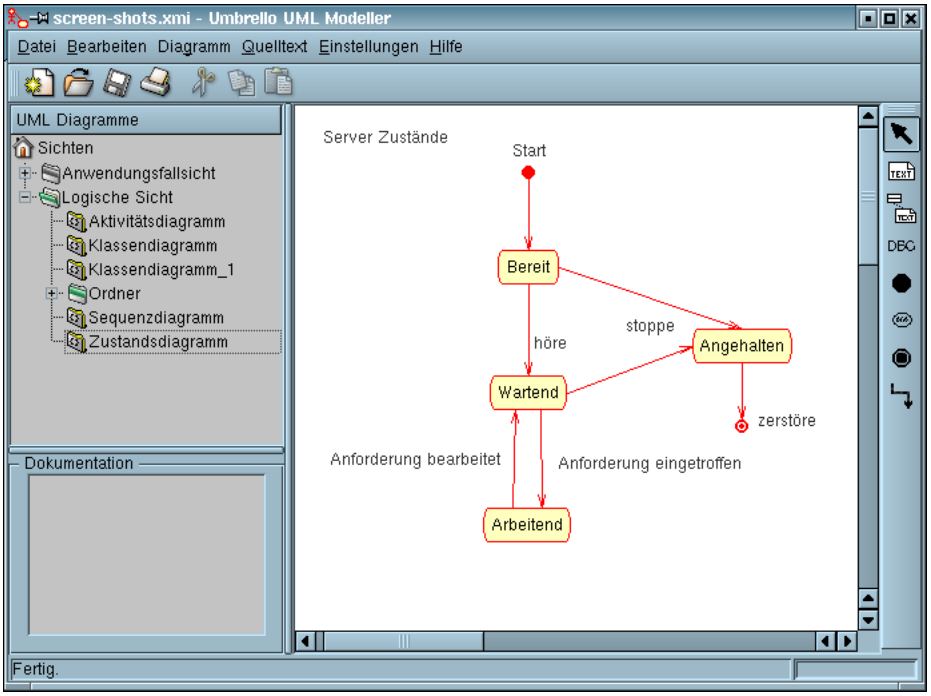
Zustandsdiagramme zeigen Objekte in ihren verschiedenen Zuständen während ihres Lebens und sie zeigen die Einflüsse, die die Objekte in einen anderen Zustand bringen.

Zustandsdiagramme betrachten Objekte als *Zustandsmaschinen* oder endliche Automaten, die in einer endlichen Anzahl von Zuständen sein können und die ihren Zustand durch eine endliche Anzahl von Einflüssen verändern. So kann sich zum Beispiel das Objekt *NetServer* während seines Lebens in folgenden Zuständen befinden:

- bereit
- wartend
- arbeitend
- angehalten

und die Ereignisse, die eine Zustandsänderung des Objektes auslösen, könnten sein:

- Objekt ist angelegt
- Objekt erhält Nachricht zu warten
- ein Client fordert eine Verbindung über ein Netzwerk an
- ein Client beendet eine Anforderung
- eine Anforderung wird bearbeitet und beendet
- Objekt empfängt Nachricht anhalten
- usw.



Umbrello UML Modeller bei der Darstellung eines Zustandsdiagramms

2.2.5.1 Zustand

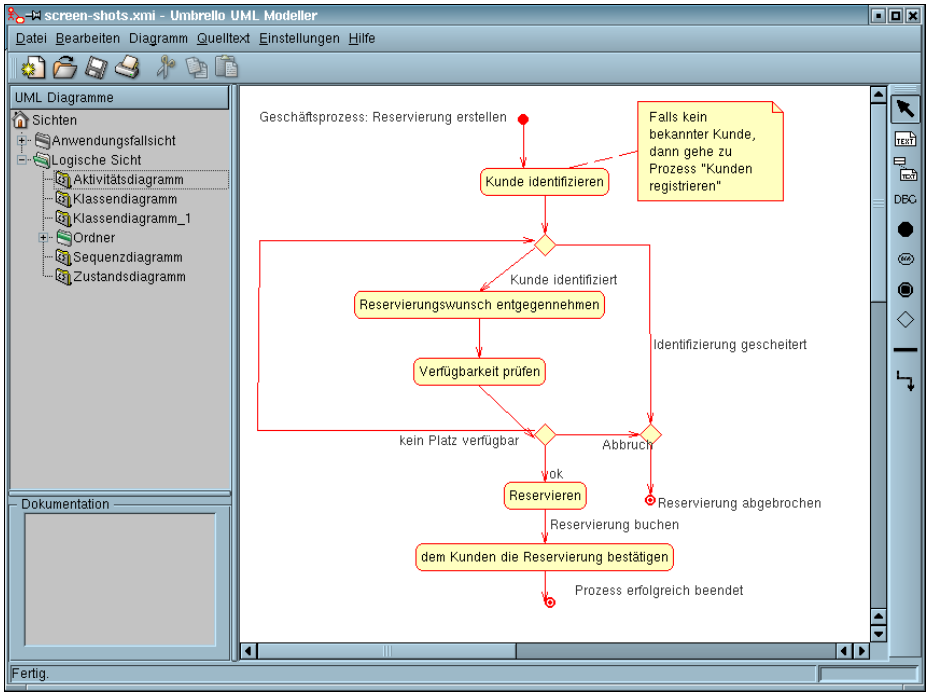
Zustände sind die Bausteine des Zustandsdiagramms. Ein Zustand gehört zu genau einer Klasse und steht für eine bestimmte Konstellation der Attributwerte dieser Klasse. Somit beschreibt ein Zustand in der UML den internen Zustand eines Objektes einer bestimmten Klasse.

Wichtig ist dabei, dass nicht jede Attributveränderung im Objekt zu einem neuen Zustand führen soll, sondern lediglich Attributwerte mit spürbaren Auswirkungen als Zustand des Objektes zu vermerken sind.

Zwei besondere Zustände sind Start und Ende. So kann es kein Ereignis geben, dass ein Objekt in seinen Startzustand zurückversetzt und kein Ereignis kann ein Objekt in einen anderen Zustand versetzen, wenn dieses bereits seinen Endzustand erreicht hat.

2.2.6 Aktivitätsdiagramm

Aktivitätsdiagramme beschreiben die Abfolge von Aktivitäten in einem System. Sie sind dabei eine spezielle Form der Zustandsdiagramme, indem sie fast ausschließlich Aktivitäten beinhalten.



Umbrello UML Modeller bei der Darstellung eines Aktivitätsdiagramms

Aktivitätsdiagramme sind den prozeduralen Flussdiagrammen sehr ähnlich. Der Unterschied ist, dass Aktivitäten klar an Objekte gekoppelt sind.

Aktivitätsdiagramme gehören immer eindeutig zu *Klassen*, *Operationen* oder *Anwendungsfällen*.

In Aktivitätsdiagrammen können sowohl sequenzielle wie auch parallele Aktivitäten dargestellt werden. Parallele Bearbeitung wird mittels der Fork/Wait Symbole umgesetzt. Für parallel laufende Aktivitäten ist es dabei unerheblich, in welcher Reihenfolge sie ablaufen. Sie können zum gleichen Zeitpunkt aber auch nacheinander ausgeführt werden.

2.2.6.1 Aktivität

Eine Aktivität ist ein einzelner Schritt in einem Prozess. Somit ist eine Aktivität ein Zustand des Systems mit einer internen Aktivität und mindestens einem Übergang. Es sind allerdings mehrere Übergänge möglich, wenn die Aktivität unterschiedliche Bedingungen enthält.

Aktivitäten können eine Hierarchie bilden, indem man eine Aktivität aus anderen zusammensetzen kann. Dabei müssen die eingehenden und ausgehenden Übergänge mit den entsprechenden Übergängen in der Verfeinerung übereinstimmen.

2.2.7 Hilfselemente

In der UML sind einige Elemente, die keine semantische Bedeutung für das Modell haben, aber Diagramme verständlicher machen können. Die Elemente sind:

- Textzeile

- Textnotiz und entsprechende Verbindung
- Rahmen

Mit der Textzeile kann eine Kurzinformation in das Diagramm eingefügt werden. Der Text ist freistehend und hat keine Bedeutung für das Modell.

Mittels der Textnotiz können detaillierte Informationen über ein Objekt oder eine Situation eingefügt werden. Der große Vorteil der Textnotiz ist, dass sie mit einem UML-Element verbunden werden kann und somit die Notiz zu diesem speziellen Objekt oder der speziellen Situation „gehört".

Rahmen sind frei schwebende Rechtecke, die zur visuellen Gruppierung von Objekten in Diagrammen genutzt werden können. Sie machen ein Diagramm besser lesbar, haben aber keine logische Bedeutung für das Modell.

2.2.8 Komponentendiagramm

Komponentendiagramme stellen die Software Komponenten (entweder Komponententechnologien wie KParts, CORBA Komponenten oder Java Beans oder klar abgrenzbare Systemeinheiten) dar und die Resultate wie Quelltextdateien, Programmbibliotheken oder relationale Datenbanktabellen.

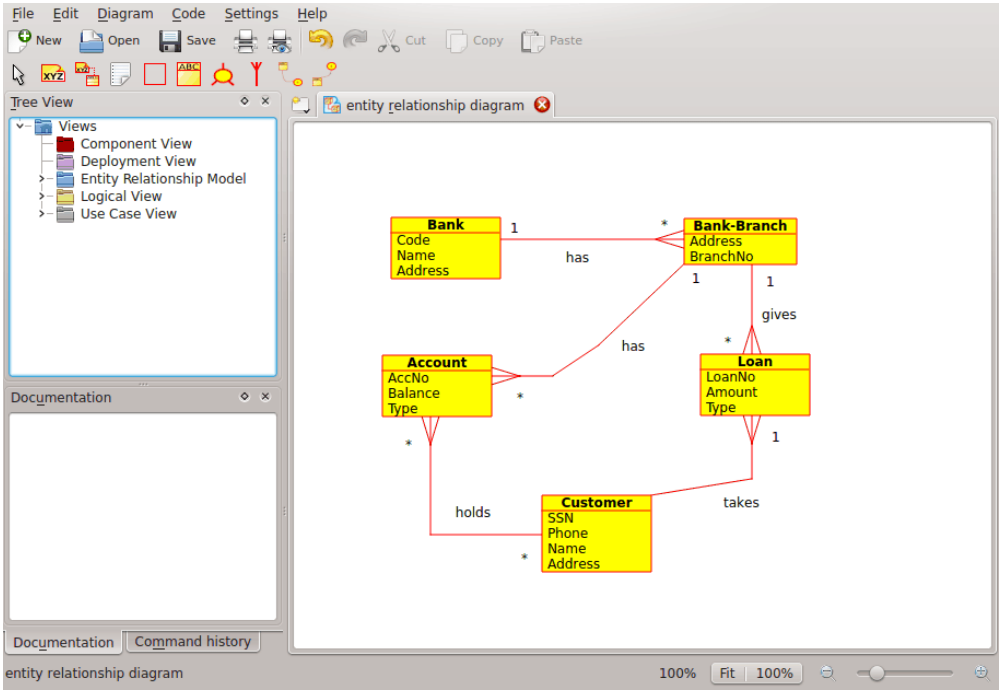
Komponenten können Schnittstellen (also abstrakte Klasse mit Operationen) enthalten, mit denen Verbindungen zwischen Komponenten möglich werden.

2.2.9 Verteilungsdiagramm

Verteilungsdiagramme zeigen die Laufzeitobjekte der Komponenten und ihre Beziehungen. Sie beinhalten Knoten als physische Ressourcen, typischerweise ein Computer. Weiterhin zeigen sie Schnittstellen und Objekte (Klasseninstanzen).

2.2.10 Entitäten-Beziehungs-Diagramm

Entitäten-Beziehungs-Diagramme (ER-Diagramme) stellen das konzeptuelle Design von Datenbankanwendungen dar. Sie beschreiben die verschiedenen Entitäten (Konzepte) im Informationssystem und die vorhandenen Beziehungen sowie Einschränkungen untereinander. Eine Weiterentwicklung der Entitäten-Beziehungs-Diagramme sind 'Extended Entity Relationship Diagrams' oder 'Enhanced Entity Relationship Diagrams' (EER). Damit werden Objektorientierte-Entwurfstechniken für ER-Diagramme eingeführt.



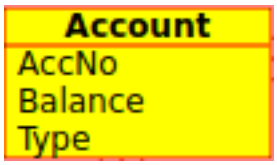
Umbrello UML Modeller bei der Darstellung eines Entitäten-Beziehungs-Diagramms

2.2.10.1 Entität

Eine *Entität* ist jedes Konzept in der realen Welt mit einer unabhängigen Existenz. Das kann ein physikalisch existierendes Objekt wie ein Rechner oder Roboter sein, oder ein konzeptionell existierendes Objekt wie zum Beispiel ein Kurs an der Universität. Jede Entität hat Attribute, die ihre Eigenschaften beschreiben.

Hinweis: Es gibt keine Standard-Schreibweise zur Darstellung von ER-Diagrammen. In der Literatur werden verschiedene System verwendet. Die in Umbrello benutzen Konzepte und Schreibweisen orientieren sich an folgendem Buch: *Elmasri R. and Navathe S. (2004). Fundamentals of Database Systems 4th edn. Addison Wesley.*

In einem Entitäten-Beziehungs-Diagramm (ER-Diagramm) werden Entitäten als Rechtecke mit dem Namen der Entität oben im Rechteck dargestellt. Die Attribute der Entität können in einem extra „abgetrennten Bereich" innerhalb des Rechteck angezeigt werden.



Visuelle Darstellung einer Entität in einem ER-Diagramm

2.2.10.1.1 Entitätsattribute

In ER-Diagrammen werden Attribute der Entität mit ihrem Namen in einem abgetrennten Bereich innerhalb der Entität angezeigt werden.

2.2.10.1.2 Einschränkungen

Einschränkungen in ER-Diagrammen bestimmen die Beschränkung der Daten im Informationsschema.

In Umbrello werden vier Arten von Einschränkungen unterstützt:

- *Primärschlüssel:* Die Menge der als *Primärschlüssel* deklarierten Attribute müssen für eine Entität eindeutig sein. Es darf nur einen Primärschlüssel in einer Entität geben und keiner seiner konstituierende Attribute darf den Wert NULL haben.
- *Eindeutiger Schlüssel:* Die Menge der als *eindeutig* deklarierten Attribute müssen für eine Entität eindeutig sein. Es darf mehrere eindeutige Einschränkungen in einer Entität geben. Die konstituierende Attribute dürfen den Wert NULL haben. Eindeutige Schlüssel und Primärschlüssel bestimmen eindeutig eine Zeile in einer Tabelle (Entität).
- *Fremdschlüssel:* Ein Fremdschlüssel ist eine referentielle Einschränkung zwischen zwei Tabellen. Der Fremdschlüssel bestimmt eine oder mehrere Spalten in einer Tabelle als Referenz auf eine oder mehrere Spalten in einer anderen Tabelle. Die Spalten in der referenzierten Tabelle müssen einen Primärschlüssel oder eindeutigen Schlüssel bilden.
- *Prüf-Einschränkung:* Eine Prüf-Einschränkung ist eine Bedingung, die die Gültigkeit von Daten definiert, wenn ein Eintrag in einer Tabelle einer relationalen Datenbank hinzugefügt oder aktualisiert wrd. Eine Prüf-Einschränkung wird auf jede Zeile einer Tabelle angewendet. Die Einschränkung muss eine Eigenschaft sein. Sie kann sich auf eine oder mehrere Spalten einer Tabelle beziehen.
  Beispiel: Preis >= 0

2.2.11 Konzepte der erweiterten Entitäten-Beziehungs-Diagramme (EER-Diagramme)

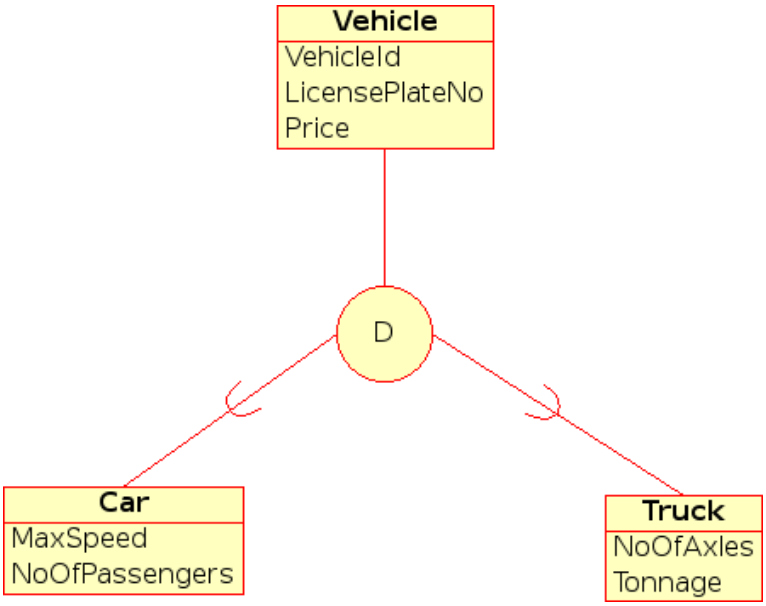
2.2.11.1 Spezialisierung

Spezialisierung ist eine Möglichkeit, neue Entitäten unter Verwendung bereits definierter Entitäten zu erstellen. Die neuen Entitäten - auch abgeleitete Entitäten genannt - übernehmen oder erben die Attribute bereits existierender Entitäten, die auch als Basis-Entitäten bezeichnet werden. Dies ermöglicht es, bereits vorhandene Daten mit keiner oder nur geringen Änderungen wiederzuverwenden.

In Umbrello können Zerlegungs- und Überschneidungs-Spezialisierungen definiert werden.

2.2.11.1.1 Zerlegungs-Spezialisierung

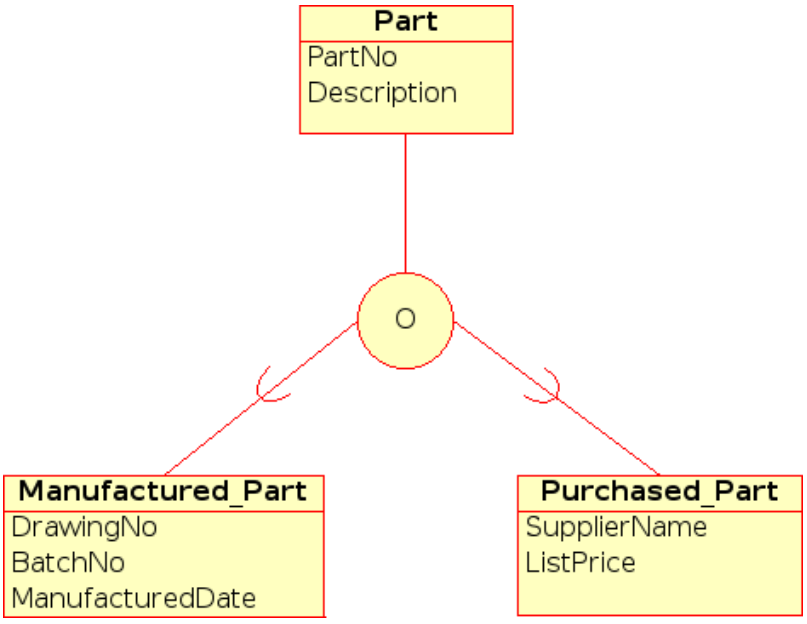
Die Zerlegungs-Spezialisierung legt fest, dass die Unterklassen der Spezialisierung getrennt sein müssen. Das bedeutet, dass eine Entität nur ein Mitglied höchstens einer der abgeleiteten Entitäten der Spezialisierung sein kann.



Visuelle Darstellung einer Zerlegungs-Spezialisierung in einem EER-Diagramm

2.2.11.1.2 Überschneidungs-Spezialisierung

Wenn die abgeleitete Entitäten nicht durch eine Zerlegung eingeschränkt sind, wird die Menge der Entitäten als Überschneidungs-Spezialisierung bezeichnet. Das bedeutet, dass eine Entität ein Mitglied von mehr als einer abgeleiteten Entität der Spezialisierung sein kann.

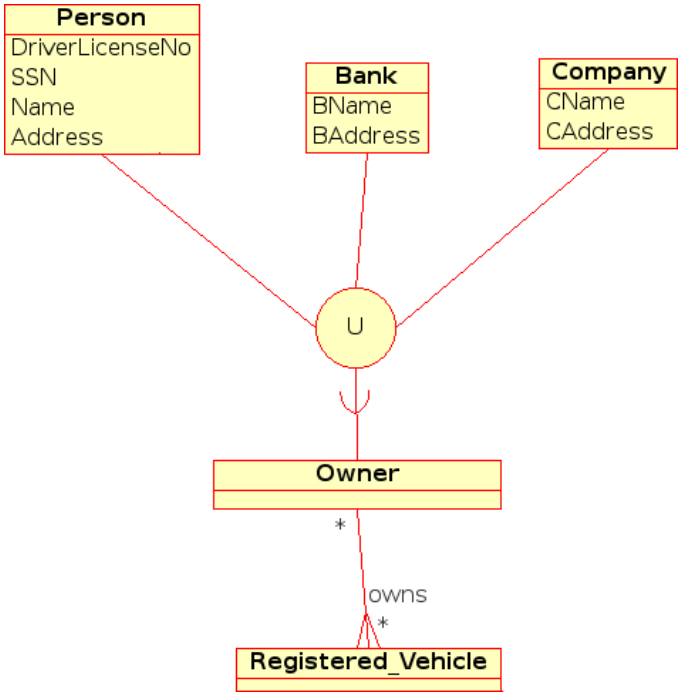


Visuelle Darstellung einer Überschneidungs-Spezialisierung in einem EER-Diagramm

2.2.11.1.3 Kategorie

Eine abgeleitete Entität wird als *Kategorie* bezeichnet, wenn sie eine Sammlung von Objekten darstellt, die aus einer Teilmenge der Vereinigung von verschiedenen Entitätstypen besteht. Eine

Kategorie wird dann gebildet, wenn eine Beziehung zwischen mehreren Ober- und einer Unterklasse benötigt wird, die Oberklassen bestehen dabei unterschiedliche Entitätstypen. Das entspricht etwa der Mehrfachvererbung in der Objektorientierten Programmierung.



Visuelle Darstellung einer Kategorie in einem EER-Diagramm

Kapitel 3

Mit Umbrello UML Modeller arbeiten

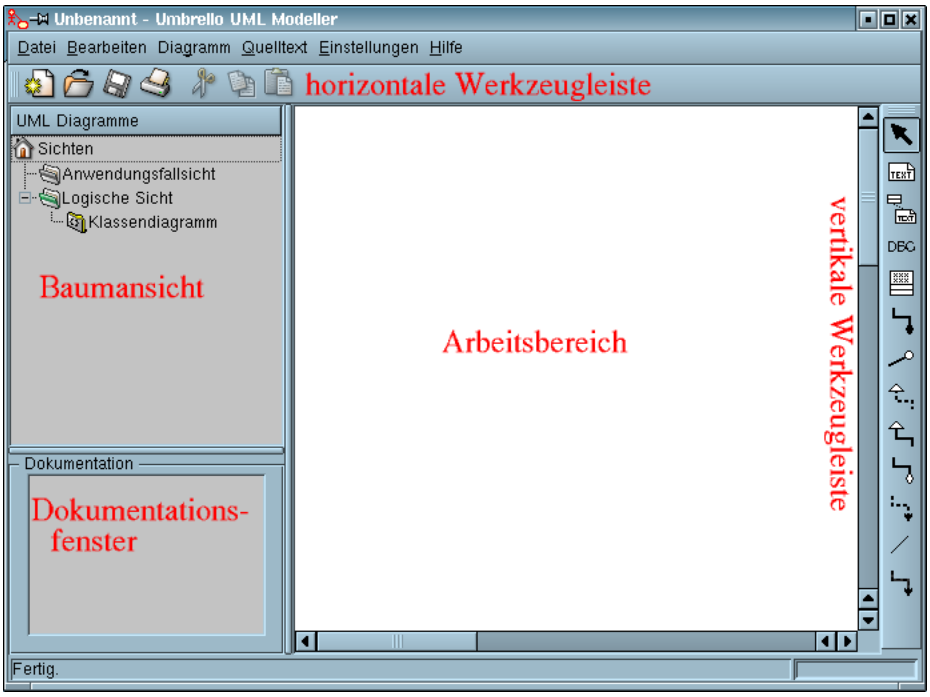
Dieses Kapitel gibt eine Einführung in die Oberfläche von Umbrello UML Modeller und vermittelt alle Kenntnisse, um sofort mit der Modellierung anzufangen. Wie normalerweise üblich, sind (fast) alle Aktionen über Umbrello UML Modellers Menüs und Werkzeugleisten erreichbar. Allerdings nutzt Umbrello UML Modeller ebenfalls sehr stark die über rechte Maustaste erreichbaren Kontextmenüs. Man kann fast auf jedes Element in Umbrello UML Modellers Arbeitsbereich oder Baumansicht mit der rechten Maustaste klicken, um für das gewählte Element sinnvoll erscheinende Funktionen zu erreichen. Für einige Benutzer ist dies am Anfang sehr verwirrend, da sie meist die Arbeit über Menüs und Werkzeugleisten gewöhnt sind. Aber hat man sich erst einmal an den Rechtsklick gewöhnt, kann man sein Arbeitstempo wesentlich erhöhen.

3.1 Benutzeroberfläche

Umbrello UML Modellers Hauptfenster unterteilt sich in 3 Bereiche, die dabei helfen den Überblick über das gesamte System zu wahren, verschiedene Diagramme schnell aufzurufen und letztendlich am Modell zu arbeiten.

Diese Bereiche werden folgendermaßen bezeichnet:

- Baumansicht
- Arbeitsbereich
- Dokumentation und Befehlsverlauf



Umbrello UML Modellers Benutzeroberfläche

3.1.1 Baumansicht

Die Baumansicht befindet sich am oberen linken Rand des Fensters und zeigt alle Diagramme, Klassen, Akteure und Anwendungsfälle, aus denen das Modell besteht. Mit der Baumansicht bekommt man einen schnellen Überblick über die Elemente, die das System formen. Weiterhin kann man mit der Baumansicht sehr elegant zwischen den einzelnen Diagrammen wechseln und Elemente aus dem Modell in das aktuelle Diagramm einfügen.

Besteht das Modell aus vielen Klassen und Diagrammen, hilft die Baumansicht bei der Wahrung der Übersicht, indem man das Modell mittels Ordnern besser organisiert. Man kann Ordner anlegen, indem man vom Kontextmenü (Klick mit rechter Maustaste auf ein Ordnerelement in der Baumansicht) den entsprechenden Befehl wählt. Die Elemente kann man einfach in die gewünschten Ordner mittels Drag'n Drop (Ziehen und Ablegen) verschieben.

3.1.2 Dokumentation und Befehlsverlauf

Das Dokumentation und Befehlsverlauf wird am unteren linken Rand des Umbrello UML Modeller-Fensters. Es zeigt eine Überblick der für das aktuell gewählte Objekt hinterlegten Dokumentation und den Befehlsverlauf der Arbeitssitzung. Das Dokumentationsfenster ist sehr klein, da es nur einen kurzen Einblick in die hinterlegte Dokumentation und einen Überblick des Befehlsverlaufs geben soll. Die komplette Dokumentation ist natürlich über die Eigenschaften des Elementes verfügbar.

3.1.3 Arbeitsbereich

Der Arbeitsbereich ist der Hauptteil des Fensters. Hier findet die eigentliche Arbeit statt. Er wird zum bearbeiten und anschauen der im Modell hinterlegten Diagramme genutzt. Der Arbeitsbereich zeigt das aktuell aktive Diagramm. Momentan kann immer nur 1 Diagramm im Arbeitsbereich zu einem Zeitpunkt dargestellt werden.

3.2 Erstellen, Laden und Speichern von Modellen

Um etwas Sinnvolles mit Umbrello UML Modeller machen zu können, muss man zuerst ein Modell anlegen, an dem man arbeiten kann. Während Umbrello UML Modellers Start wird entweder das zuletzt bearbeitete Modell geladen oder ein neues leeres Modell angelegt. Die hängt von den Einstellungen im Einstellungsfenster ab. Dadurch kann man sofort mit der gewünschten Arbeit beginnen.

3.2.1 Neues Modell

Um zu irgendeinem Zeitpunkt ein neues Modell anzulegen, wählt man **Neu** aus dem Menü **Datei** oder klickt auf das Symbol **Neu** in der horizontalen Werkzeugleiste. Arbeitet man gerade an einem Modell, fragt Umbrello UML Modeller zuerst nach, ob man das aktuelle Modell nicht speichern will, bevor man ein neues Element anlegt.

3.2.2 Modell speichern

Man kann zu jedem Zeitpunkt das Modell speichern indem man den Eintrag **Speichern** aus dem Menü **Datei** wählt, oder auf das entsprechende Symbol in der horizontalen Werkzeugleiste klickt. Soll das Modell unter einem anderen Dateinamen gespeichert werden, wählt man den Eintrag **Speichern unter ...** aus dem Menü **Datei**.

Zur Absicherung bietet Umbrello UML Modeller die Möglichkeit, das Modell automatisch nach einem bestimmten Zeitabschnitt zu speichern. Man kann diese Möglichkeit und das Zeitintervall in Umbrello UML Modellers **Einstellungen** definieren.

3.2.3 Modell laden

Um ein bereits existierendes Modell zu laden, muss man **Öffnen** aus dem Menü **Datei** oder das entsprechende Symbol aus der horizontalen Werkzeugleiste wählen. Die zuletzt bearbeiteten Modelle sind ebenfalls über das Untermenü **Zuletzt geöffnete Dateien** im Menü **Datei** verfügbar. Dadurch kann man häufig verwendete Modelle wesentlich schneller aufrufen.

Mit Umbrello UML Modeller kann man immer nur an einem Modell gleichzeitig arbeiten. Fordert man das Programm zum Laden eines anderen Modells auf und wurde das momentan bearbeitete Modell seit dem letzten Speichern verändert, fragt Umbrello UML Modeller nach, ob man die Änderungen nicht zuerst speichern will. Dadurch wird ein möglicher Datenverlust verhindert. Man kann aber mehrere Instanzen von Umbrello UML Modeller starten und auch zwischen den Instanzen Objekte kopieren und einfügen.

3.3 Modelle bearbeiten

In Umbrello UML Modeller gibt es prinzipiell 2 Möglichkeiten die Elemente des Modells zu verändern.

- Man kann die Elemente direkt in der Baumansicht bearbeiten.
- Man kann die Elemente in den Diagrammen bearbeiten.

Über das mit der rechten Maustaste erreichbare Kontextmenü kann man in der Baumansicht fast alle Elemente des Modells hinzufügen, entfernen oder verändern. Klickt man zum Beispiel mit der rechten Maustaste auf einen Ordner in der Baumansicht, kann man eines der verschiedenen Diagramme sowie Akteure, Klassen und Anwendungsfälle anlegen, je nachdem ob der Ordner der *logischen Sicht* oder der *Anwendungsfallsicht* untergeordnet ist.

Nachdem man ein Element dem Modell hinzugefügt hat, kann man seine Eigenschaften über den Eigenschaftendialog ändern. Diesen erreicht man über den Punkt *Eigenschaften* aus dem Kontextmenü des jeweiligen Elements.

Weiterhin kann man das Modell durch das Anlegen und Ändern von Elementen in den Diagrammen bearbeiten. Weitere Details, wie das geht, sind in den folgenden Abschnitten zu finden.

3.4 Diagramme hinzufügen und entfernen

Das UML-Modell besteht aus UML-Elementen und den Assoziationen zwischen diesen. Man kann aber das Modell nicht direkt sehen, sondern man nutzt *Diagramme*, um es zu betrachten.

3.4.1 Diagramme anlegen

Um ein neues Diagramm dem Modell hinzufügen, muss man lediglich den Diagrammtyp aus dem Untermenü **Neu** aus dem Menü **Diagramm** wählen und dem Kind einen Namen geben. Das Diagramm wird angelegt und als aktives Diagramm für den Arbeitsplatz ausgewählt. Es taucht sofort in der Baumansicht auf.

Man sollte sich daran erinnern, dass Umbrello UML Modeller sehr starken Gebrauch von den Kontextmenüs, erreichbar über die rechte Maustaste, macht. So kann man auf einen Ordner in der Baumansicht mit der rechten Maustaste klicken und aus dem Untermenü **Neu** den gewünschten Typ auswählen. Es ist zu beachten, dass man in der Anwendungsfallsicht lediglich Anwendungsfalldiagramme hinzufügen kann und in der logischen Sicht alle anderen Diagrammtypen.

3.4.2 Diagramme entfernen

Will man ein Diagramm aus dem Modell entfernen, muss man es aktivieren und dann **Löschen** aus dem Menü **Diagramm** aufrufen. Man kann dies ebenfalls über den Eintrag **Löschen** im Kontextmenü des entsprechenden Diagramms in der Baumansicht machen.

Da das Löschen eines Diagramms ein ernster Eingriff ist, der Datenverlust verursachen kann, falls es unbeabsichtigt passiert, fragt Umbrello UML Modeller um Bestätigung, bevor die Löschoperation ausgeführt wird. Wurde das Diagramm einmal gelöscht und die Datei gespeichert, gibt es keine Möglichkeit, die Aktion rückgängig zu machen!

3.4.3 Diagramme umbenennen

Will man ein Diagramm umbenennen, kann man dies durch den Eintrag **Umbenennen** im Kontextmenü des Diagramms in der Baumansicht erreichen.

Ein anderer Weg ist der Eigenschaftendialog des Diagramms, der durch den Eintrag Eigenschaften aus dem über die rechte Maustaste erreichbaren Kontextmenüs des Diagramms aufgerufen werden kann. Weiterhin kann man in der Baumansicht mit einem Doppelklick auf das Diagramm den Eigenschaftendialog ebenfalls aufrufen.

3.5 Diagramme bearbeiten

Während der Arbeit mit Diagrammen versucht Umbrello UML Modeller einen durch die Anwendung einfacher Regeln zu unterstützen, indem nur die möglichen Elemente in einem Diagramm zur Verfügung stehen und nur die sinnvollen Beziehungen zwischen diesen angelegt werden

können. Als UML-Experte wird man dies vielleicht gar nicht bemerken, für Anfänger ist es eine große Unterstützung bei der Erstellung von standardkonformen Diagrammen.

Nachdem man ein Diagramm angelegt hat, kann man es bearbeiten. Es ist dabei der Unterschied zwischen Diagramm- und Modellbearbeitung, für Anfänger schwieriger verständlich, zu beachten. Wie bereits dargelegt wurde, sind Diagramme eine *Sicht* auf das *Modell*. Legt man zum Beispiel in einem Klassendiagramm eine Klasse an, dann bearbeitet man eigentlich sowohl Modell wie auch Diagramm. Ändert man hingegen lediglich die Farbe oder andere Darstellungsoptionen einer Klasse in einem Klassendiagramm, dann wird dadurch lediglich das Diagramm, aber nicht das Modell verändert.

3.5.1 Elemente einfügen

Eines der ersten Dinge mit einem neuen Diagramm ist es, Elemente wie Klassen, Akteure oder Anwendungsfälle einzufügen. Es gibt prinzipiell 2 Möglichkeiten, wie dies geschehen kann.

- Existierende Elemente aus der Baumansicht in das Diagramm schieben.
- Ein neues Element mithilfe der Werkzeuge in der vertikalen Werkzeugleiste anlegen und gleichzeitig dem aktiven Diagramm hinzuzufügen.

Um ein im Modell bereits existierendes Element in das aktuelle Diagramm einzufügen, muss man es lediglich von der Baumansicht in das Diagramm an die gewünschte Stelle ziehen. Man kann das Element jederzeit mit dem Auswahlwerkzeug im Diagramm verschieben.

Die zweite Möglichkeit Elemente hinzuzufügen ist es, eines der Werkzeuge aus der vertikalen Werkzeugleiste am rechten Bildschirmrand zu benutzen. Dabei wird das Element ebenfalls dem Modell hinzugefügt.

Die Diagramm-Werkzeugleiste befindet sich standardmäßig am oberen Fenster des Anwendungsfensters, man kann sie allerdings an andere Stellen verschieben oder sie über allen anderen Bereichen schwebend positionieren. Die in dieser Werkzeugleiste verfügbaren Werkzeuge, dargestellt durch die verschiedenen Knöpfe, ändern sich je nach momentan bearbeiteten Diagrammtyp. Das momentan gewählte Werkzeug wird hervorgehoben in der Werkzeugleiste dargestellt. Über die Taste **Esc** kann man das Auswahl-Werkzeug auswählen.

Hat man ein Bearbeitungswerkzeug aus der Werkzeugleiste ausgewählt, zum Beispiel um eine Klasse einzufügen, ändert sich der Mauszeiger in ein Kreuz. Man kann nun das Element in das Diagramm über einen einzelnen Klick mit der linken Maustaste im Diagramm einfügen. UML-Elemente müssen immer *eindeutige Namen* haben. Gibt es zum Beispiel in einem Diagramm die Klasse „KlasseA", dann kann man in einem anderen Diagramm keine neue Klasse mit dem gleichen Namen einfügen. Soll es sich bei den beiden Klassen um unterschiedliche Elemente handeln, dann müssen diese auch unterschiedliche Namen haben. Um das *gleiche* Element nochmals einzufügen, ist das Werkzeug Klasse einfügen nicht das richtige Hilfsmitteln. Man muss in solch einem Fall lediglich die gewünschte Klasse aus der Baumansicht in das Diagramm ziehen.

3.5.2 Elemente löschen

Man kann jedes Element über den Eintrag **Löschen** aus dem Kontextmenü entfernen.

Hier zeigt sich wiederum der *große* Unterschied zwischen einem Element aus einem Diagramm und einem Element aus dem Modell zu entfernen. Löscht man ein Element in einem Diagramm, dann wird es nur aus diesem speziellen Diagramm entfernt. Das Element ist aber weiterhin Teil des Modells und falls es in anderen Diagrammen genutzt wird, werden diese nicht verändert. Löscht man hingegen das Element in der Baumansicht, dann löscht man das Element im ganzen *Modell*. Da das Element dann nicht länger im Modell existiert, wird es ebenfalls in allen Diagrammen gelöscht, in denen es verwendet wird.

3.5.3 Elemente bearbeiten

Man kann die meisten UML-Elemente des Modells ändern, indem man den Eigenschaftendialog öffnet und die gewünschten Änderungen vornimmt. Um zum Beispiel ein Objekt zu verändern, ist es auszuwählen und dann **Eigenschaften** aus seinem Kontextmenü (rechte Maustaste) zu wählen. Jedes Element verfügt über solch einen Dialog, der aus mehreren Seiten besteht, abhängig vom Typ des zu bearbeitenden Elementes. Für einige Elemente, wie Akteure, kann man lediglich einige wenige Parameter wie den Namen oder die Dokumentation ändern. Für andere Elemente hingegen, wie Klassen, kann man viele Einstellungen vornehmen wie Attribute und Operationen, Sichtbarkeit und die Darstellung im Diagramm (nur Operationen oder komplette Signatur der Operationen). Man kann sogar die Linienfarben und die Füllfarbe für die Visualisierung der Klasse im Diagramm einstellen.

Für UML-Elemente kann man diesen Eigenschaftendialog durch einen Doppelklick auf das entsprechende Element mit dem Auswahlwerkzeug (Pfeil) aufrufen.

Man kann den Eigenschaftendialog auch über das Kontextmenü in der Baumansicht erreichen. Dadurch kann man die Eigenschaften für Diagramme bearbeiten, zum Beispiel ob der Raster anzuzeigen ist oder nicht.

3.5.4 Klassen bearbeiten

Obwohl das Bearbeiten der Eigenschaften von Objekten bereits im letzten Abschnitt erörtert wurde, folgt ein weiterer Abschnitt zu Klassen, da sie komplizierter als alle anderen UML-Elemente sind.

Im Eigenschaftendialog der Klassen kann man alles von der Farbe bis zu den Operationen und Attributen bearbeiten.

3.5.4.1 Allgemeine Klasseneinstellungen

Die Seite im Eigenschaftendialog für die allgemeinen Einstellungen ist selbsterklärend. Dort kann man den Klassennamen, die Sichtbarkeit, die Dokumentation usw. ändern. Diese Seite ist immer verfügbar.

3.5.4.2 Attributeinstellungen von Klassen

Auf der Seite für die Attributeinstellungen kann man Attribute der Klasse hinzufügen, bearbeiten und löschen. Man kann weiterhin die Attribute in der Liste nach oben und nach unten über die Pfeile verschieben. Diese Seite ist ebenfalls immer verfügbar.

3.5.4.3 Operationseinstellungen von Klassen

Wie in der Seite für die Attributeinstellungen, kann man auf der Seite für die Operationseinstellungen Operationen hinzufügen, bearbeiten und aus der Klasse entfernen. Fügt man eine Operation hinzu oder verändert eine bestehende, dann geschieht dies über den Dialog *Operationseigenschaften*. Um einen neuen Parameter zu einer Operation hinzuzufügen, muss man den Knopf **Neuer Parameter** benutzen, der den Dialog *Parametereigenschaften* aufruft. Diese Seite ist immer verfügbar.

3.5.4.4 Einstellung für parametrisierbare Klasse

Auf dieser Seite kann man Klassen-Vorlagen, das sind unspezifizierte Klassen und Datentypen, einfügen. In Java 1.5 werden sie unter der Bezeichnung Generics eingeführt.

3.5.4.5 Seite der Klassenassoziationen

Die Seite **Assoziationen** zeigt alle mit der Klasse verbundenen Assoziationen im aktuellen Diagramm. Ein Doppelklick auf eine Assoziation zeigt den entsprechenden Eigenschaftsdialog. Je nach Assoziationstyp kann man einige Parameter wie Multiplizität und Rollennamen ändern. Verfügt die gewählte Assoziation nicht über solche Parameter, dann ist der Eigenschaftendialog nicht änderbar und man kann lediglich die Dokumentation der Assoziation ändern.

Diese Seite ist nur verfügbar, wenn man die Klasseneigenschaften aus einem Diagramm heraus aufruft. Erfolgt hingegen der Aufruf aus der Baumansicht, ist diese Seite nicht verfügbar.

3.5.4.6 Seite Anzeige

In der Seite **Anzeige** kann man einstellen, wie die Klasse im Diagramm dargestellt wird. Eine Klasse kann zum Beispiel als einfaches Rechteck mit dem Namen dargestellt werden. Dies ist besonders nützlich, wenn man sehr viele Klassen im Diagramm hat, oder sich noch nicht für Details interessiert. Man kann aber auch die komplette Klasse mit Paketen, Stereotypen, Attributen und Operationen mit Signatur und Sichtbarkeit anzeigen lassen.

Je nach gewünschtem Informationsumfang, kann man die entsprechenden Optionen auf dieser Seite wählen. Die hier gemachten Einstellungen sind lediglich *Anzeigeeinstellungen* für das aktuelle Diagramm. Das bedeutet, „ausgeblendete" Operationen im Diagramm sind trotzdem weiterhin Teil des gesamten Modells. Diese Seite ist nur verfügbar, wenn der Eigenschaftendialog der Klasse aus einem Diagramm heraus aufgerufen wurde. Wird der Eigenschaftendialog lediglich aus der Baumansicht aufgerufen, fehlen die Darstellungseinstellungen logischerweise.

3.5.4.7 Seite Klassenstil

Auf der Seite **Stil der Bedienelemente** kann man die Linienfarben und die Füllfarbe einstellen. Diese Seite ist nur verfügbar, wenn die Eigenschaften der Klasse aus einem Diagramm aufgerufen wurden und nicht aus der Baumansicht.

3.5.5 Assoziationen

Assoziationen bringen zwei UML-Elemente miteinander in Verbindung. Normalerweise werden Assoziationen zwischen Klassen definiert, aber einige können auch zwischen Anwendungsfällen und Akteuren angelegt werden.

Um eine Assoziation anzulegen, ist das entsprechende Werkzeug aus der vertikalen Werkzeugleiste auszuwählen (zum Beispiel allgemeine Assoziation, Verallgemeinerung, Aggregation usw.) und auf das Ausgangselement zu klicken. Danach muss man auf den 2. Partner der Assoziation klicken. Es handelt sich dabei um 2 Klicks und *nicht* um einen Klick mit anschließendem Ziehen von einem Element zum anderen!

Versucht man eine Assoziation anzulegen, die nicht mit den UML-Spezifikationen vereinbar ist, dann verhindert dies Umbrello UML Modeller und gibt einen entsprechenden Hinweis aus. Dies ist zum Beispiel der Fall, wenn zwischen Klasse A und B eine Verallgemeinerung existiert und nun versucht wird, eine Verallgemeinerung von B nach A anzulegen.

Durch einen Rechtsklick auf eine Assoziation erscheint ein Menü mit den Einträgen, die man auf die Assoziation anwenden kann. Um zum Beispiel eine Assoziation zu löschen, muss man den Eintrag **Löschen** aus dem Kontextmenü auswählen. Man kann über den Eintrag **Eigenschaften** je nach Typ der Assoziation in einem Dialog die Attribute der Assoziation wie Rollenname und Multiplizität ändern.

3.5.5.1 Ankerpunkte

Assoziationen werden standardmäßig als durchgezogene Linie zwischen den beiden zu verbindenden Objekten im Diagramm gezeichnet.

Man kann solch einen zusätzlichen Ankerpunkt durch einen Doppelklick auf die Assoziation an einer beliebigen Stelle einfügen. Diesen eingefügten Ankerpunkt (dargestellt durch einen blauen Punkt, wenn die Assoziation ausgewählt ist) kann man nun verschieben, um der Assoziation die gewünschte Form zu geben.

Um einen Ankerpunkt zu entfernen, muss man diesen lediglich doppelt mit der linken Maustaste anklicken.

Es ist zu beachten, dass man den Eigenschaftsdialog einer Assoziation lediglich über ihr Kontextmenü aufrufen kann. Ein Doppelklick, wie bei anderen UML-Elementen, fügt einen Ankerpunkt ein.

3.5.6 Textnotizen, Anmerkungen und Rahmen

Textnotizen, Textzeilen und Rahmen sind Elemente, die in allen Diagrammtypen verfügbar sind. Sie haben keine inhaltliche Bedeutung für das Modell, sie können aber hilfreiche Kommentare und Erklärungen enthalten, die ein Diagramm leichter lesbar machen.

Um eine Textnotiz oder eine Textzeile einzufügen, wählt man das entsprechende Werkzeug aus der vertikalen Werkzeugleiste und klickt im Diagramm an die Stelle, wo das Kommentar erscheinen soll. Man kann den Text über das Kontextmenü ändern oder im Fall von Textnotizen durch einen Doppelklick auf den Text.

3.5.6.1 Anker

Mit Ankern kann man ein UML-Element mit einer Textnotiz verbinden. Möchte man zum Beispiel mit einer Textnotiz einige Hinweise über eine Klasse angeben, dann verbindet man diese Textnotiz mit der Klasse. Dadurch ist ersichtlich, dass die Textnotiz sich auf das entsprechende Element „bezieht".

Um einen Anker zwischen einer Textnotiz und einem UML-Element einzufügen, muss man das entsprechende Werkzeug von der vertikalen Werkzeugleiste wählen. Zuerst klickt man auf die gewünschte Textnotiz und mit dem zweiten Klick auf das zu verknüpfende UML-Element.

Kapitel 4

Quelltextimport und Quelltexterzeugung

Umbrello UML Modeller ist ein UML Modellierungswerkzeug und sein Schwerpunkt liegt deshalb auf der *Analyse und des Designs* ihres Systems. Um den Übergang zwischen Design und *Implementierung* zu erleichtern, kann Umbrello UML Modeller Quelltext in verschiedenen Programmiersprachen erzeugen. Möchte man hingegen die UML in einem bereits existierendem C++ Projekt einsetzen, unterstützt Umbrello UML Modeller einem bei der Erstellung eines Modells des vorhandenen Systems, indem es den Quelltext einliest und die gefundenen Klassen erstellt.

4.1 Quelltexterzeugung

Umbrello UML Modeller kann Quelltext in verschiedenen Programmiersprachen auf Basis ihres UML Modells erzeugen und hilft dabei einen Anfang für die Implementierung zu schaffen. Der erzeugte Quelltext besteht aus den Klassendeklarationen, den Methoden und den Attributen. Man muss diese Hüllen „lediglich ausfüllen", um die Klassenoperationen mit Funktionalität zu füllen.

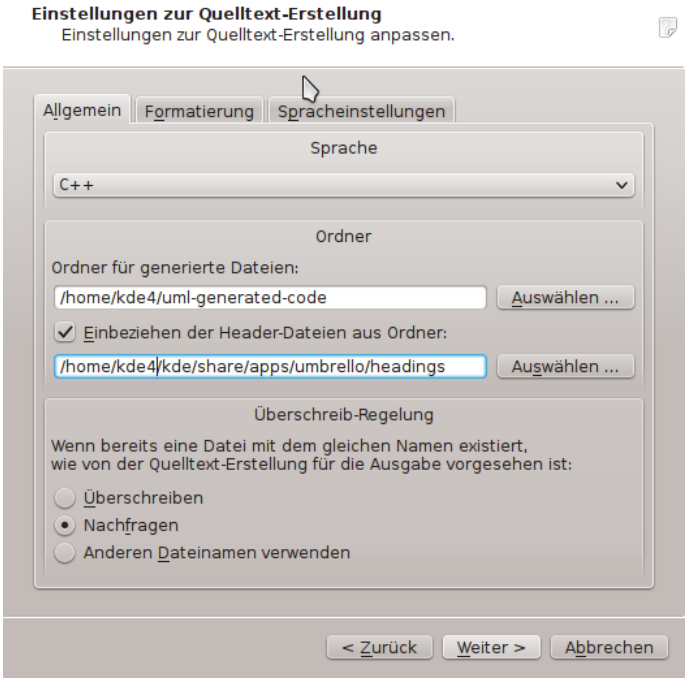
Umbrello UML Modeller 2 bietet Quelltexterzeugung für ActionScript, Ada, C++, C#, D, IDL, Java™, JavaScript, MySQL, Pascal, Perl, PHP, PHP5, PostgreSQL, Python, Ruby, Tcl, Vala, und XMLSchema.

4.1.1 Quelltext erzeugen

Um Quelltext mit Umbrello UML Modeller zu erzeugen, muss man zuerst ein Modell laden, das mindestens eine Klasse enthält. Wenn man Quelltext erzeugen will, muss man den **Assistent für Quelltext-Erstellung ...** aus dem Menü **Quelltext** auswählen. Dadurch wird der Assistent gestartet, der dann durch den Prozess zur Quelltexterzeugung führt.

Im ersten Schritt muss man die Klassen auswählen, für die Quelltext erzeugt werden soll. Am Anfang sind alle Klassen des Modells ausgewählt und man kann nun einzelne entfernen. Dazu muss man sie aus der rechten Liste in die linke Liste verschieben.

Im folgenden Schritt des Assistenten kann man die Parameter des Quelltextgenerators verändern. Folgende Parameter sind verfügbar:



Parameter für die Quelltexterzeugung in Umbrello UML Modeller

4.1.1.1 Generierungsoptionen

4.1.1.1.1 Umfang Quelltextkommentare

Der Parameter **Erzeugt Dokumentations-Kommentare, selbst wenn diese leer sind.** weist den Quelltextgenerator an, Kommentare der Form /** bla */ einzufügen, selbst wenn diese leer sind. Hat man die Klassen, Methoden und Attribute im Modell dokumentiert, fügt die Quelltexterzeugung diese Kommentare im Doxygen Format mit ein, egal was an dieser Stelle ausgewählt wurde. Ist dieser Parameter aktiviert, werden im Unterschied allerdings für alle Klassen, Methoden und Attribute Platzhalter eingefügt, selbst wenn diese nicht im Modell dokumentiert wurden. Man sollte diese dann direkt im Quelltext in den bereits vorhandenen Platzhaltern dokumentieren.

Erzeugt Kommentare für Abschnitte, selbst wenn diese leer sind.: Umbrello UML Modeller schreibt Kommentare in den Quelltext um die verschiedenen Bereiche einer Klasse zu trennen. So würde zum Beispiel „public methods" oder „Attributes" vor den entsprechenden Abschnitten eingefügt werden. Wurde dieser Parameter aktiviert, wird für jeden Abschnitt ein entsprechendes Kommentar eingefügt, selbst dann, wenn der Abschnitt leer ist. So würde zum Beispiel das Kommentar „protected methods " eingefügt werden, selbst wenn keine geschützten Methoden in der Klasse existieren.

4.1.1.1.2 Ordner

Verzeichnis für alle zu erzeugenden Dateien: Hier wählt man das Verzeichnis aus, in dem der erzeugte Quelltext abgelegt werden soll.

Der Parameter **Einbeziehung der Header-Dateien aus Verzeichnis** erlaubt es einen Kopf an den Anfang jeder erzeugten Datei einzufügen. Diese Dateiköpfe können zum Beispiel Urheberhinweise oder Lizenzinformationen enthalten, sowie Variablen, die während der Quelltexterzeugung entsprechend ersetzt werden. Man sollte einen Blick auf die Vorlagedateien für Dateiköpfe werfen, die mit Umbrello UML Modeller ausgeliefert werden. Dort kann man sehen, wie man zum Beispiel mit den Variablen das aktuelle Datum oder einen Namen einfügen kann.

4.1.1.1.3 Vorgaben für Überschreibung

Dieser Parameter legt das Verhalten von Umbrello UML Modeller fest, wenn es eine Datei während der Quelltexterzeugung anlegen will, die im Zielverzeichnis bereits existiert. Umbrello UML Modeller kann vorhandene Dateien *nicht modifizieren*. Man kann wählen zwischen dem Überschreiben der existierenden Datei, dem Nachfragen, was passieren soll, und der Auswahl eines anderen Dateinamens durch Umbrello UML Modeller. Soll Umbrello UML Modeller einen anderen Dateinamen finden, dann hängt Umbrello UML Modeller ein Suffix an die entsprechende Datei an.

4.1.1.1.4 Sprache

Umbrello UML Modeller wählt als Sprache für die Quelltexterzeugung die momentan als aktive Sprache gewählte aus. Man kann allerdings im Quelltextassistenten eine andere Sprache auswählen.

4.1.1.2 Quelltexterzeugung

Der dritte und letzte Schritt des Assistenten zeigt den Status der eigentlichen Quelltexterzeugung. Man muss lediglich auf die Schaltfläche Erzeugen klicken, damit die entsprechenden Dateien mit den Klassen angelegt werden.

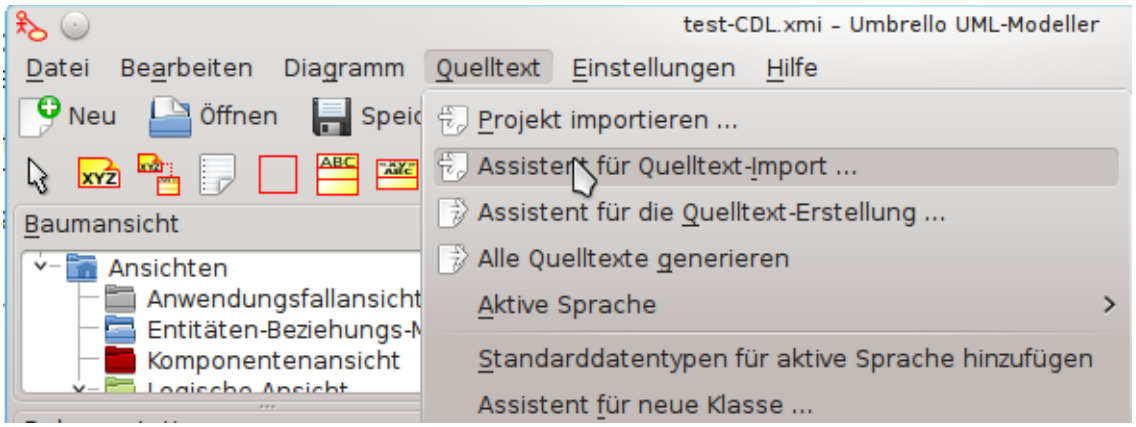
Es ist zu beachten, dass die gesetzten Parameter nur für die aktuelle Quelltexterzeugung gültig sind. Beim nächsten Aufruf des Assistenten muss man alle Parameter, wie Header-Datei Verzeichnis und Vorgaben für Überschreibung, neu einstellen. Man kann allerdings die Voreinstellungen dauerhaft über den **Quelltexterzeugung** Abschnitt in den Umbrello UML Modeller Einstellungen verändern. Diese Einstellungen erreicht man über **Einstellungen** → **Umbrello UML Modeller einrichten ...**.

Hat man die Standardeinstellungen für die Quelltexterzeugung bereits richtig eingestellt, kann man die Quelltexterzeugung ohne den entsprechenden Assistenten direkt starten. Dazu wählt man **Erzeuge alle Quelltexte** aus dem Menü **Quelltext**. Dies erzeugt den Quelltext aller Klassen des Modells mit den aktuellen Einstellungen wie Ausgabeverzeichnis und Vorgaben für Überschreiben. Man sollte deshalb vorsichtig damit umgehen.

4.2 Quelltext einlesen

Umbrello UML Modeller kann bereits vorhandenen Quelltext eines bestehenden Projektes einlesen, um Sie beim Aufbau des Systemmodells zu unterstützen. Umbrello UML Modeller 2 unterstützt Quelltext von ActionScript, Ada, C++, C#, D, IDL, Java™, Javascript, MySQL, Pascal, PHP und Vala.

Um Klassen in das aktuelle Modell zu importieren, muss man **Assistent für Quelltext-Import ...** aus dem Menü**Quelltext** auswählen. Im erscheinenden Dateidialog sind die Dateien auszuwählen, die die Deklarationen der Klassen enthalten und dann muss **Weiter**, **Import starten** und dann **Fertigstellen** gedrückt werden. Die Klassen werden importiert und man findet sie danach in der Baumansicht des Modells. Es ist zu beachten, dass Umbrello UML Modeller beim Einlesen von Quelltext keine Diagramme anlegt, sondern lediglich die Klassen. Man kann diese nun in beliebigen Diagrammen verwenden.



Dialog für das Einlesen von Quelltext in Umbrello UML Modeller

Kapitel 5

Weitere Funktionen

5.1 Weitere Funktionen in Umbrello UML Modeller

In diesem Kapitel werden weitere Funktionen von Umbrello UML Modeller genau erläutert.

5.1.1 Objekte als PNG-Bilder kopieren

Neben dem normalen Funktionen wie Ausschneiden und Einfügen für das Kopieren von Objekten zwischen einzelnen Diagrammen, beherrscht Umbrello UML Modeller auch das Kopieren von Objekten als PNG-Bilder, sodass man diese in andere Dokumente einfügen kann. Damit dies funktioniert, muss man keine speziellen Handlungen vornehmen. Einfach das Objekt (Klasse, Akteur, usw.) auswählen und kopieren (**Strg-C** oder über das Menü), dann ein Textverarbeitungsprogramm öffnen (oder jedes andere Programm, indem man Bilder einfügen kann) und **Einfügen** auswählen. Mit dieser tollen Funktion kann man sehr leicht einzelne Teile von Diagrammen als einfache Bilder exportieren.

5.1.2 Export als ein Bild

Man kann natürlich ebenfalls ein komplettes Diagramm als PNG-Bild exportieren. Dazu wählt man das gewünschte Diagramm und ruft dann aus dem Menü **Diagramm** den Eintrag **Als Bild exportieren ...** auf.

Sie können mehrere Diagramme auf einmal exportieren, indem Sie **Diagramme in Bilder exportieren ...** aus dem Menü **Datei** aufrufen. Hierbei können Sie auch die Bildauflösung einstellen, sodass die Bilder nicht zu unschaft werden.

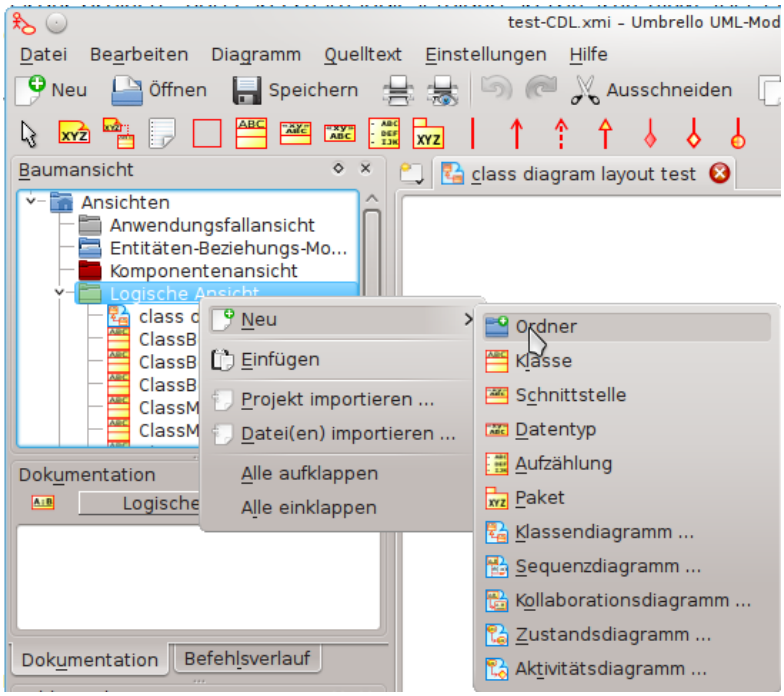
5.1.3 Drucken

Umbrello UML Modeller ermöglicht es einzelne Diagramme zu drucken. Man kann dies entweder über den **Druckknopf** aus der Programmwerkzeugleiste oder über den Menüeintrag **Druck** aus dem Menü **Datei** aufrufen. Es erscheint der Standard-KDE-Druckdialog, womit man den Ausdruck des Diagramms steuern kann.

5.1.4 Logische Ordner

Für eine bessere Organisation des Modells, besonders bei größeren Projekten, kann man logische Ordner in der Baumansicht anlegen. Dazu einfach **Neu** → **Ordner** aus dem Kontextmenü der

Standardordner in der Baumansicht wählen und einen neuen Ordner anlegen. Ordner können verschachtelt werden und man kann Objekte in diese hinein ziehen oder kopieren.

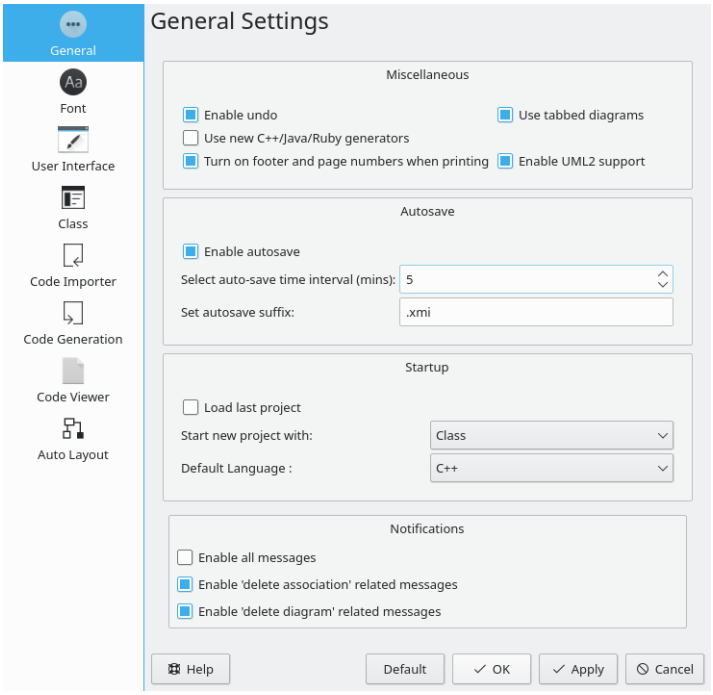


ein Modell mit logischen Ordnern in Umbrello UML Modeller organisieren

Kapitel 6

Einstellungen

6.1 Allgemeine Einstellungen



Optionen für Allgemeine Einstellungen in Umbrello UML Modeller

6.1.1 Verschiedenes

- Die Option **Rückgängig/Wiederherstellen** ermöglicht es, vorherige Aktionen rückgängig zu machen und sie wiederherzustellen.
- Mit **Neue Quelltext-Erstellung für C++/Java/Ruby verwenden** kann zwischenablagealten und neuen Quelltext-Generatoren gewählt werden.
- Ist **Fußzeile und Seitennummer beim Drucken verwenden** ausgewählt, werden beim Drucken die Seitennummern mit gedruckt.
- **Unterfenster verwenden** ermöglicht gleichzeitig mehrere Diagrammfenster in Unterfenstern.

6.1.2 Automatisches Speichern

- Mit **Automatisches Speichern aktivieren** können Sie Ihre Datei automatisch speichern.
- In **Intervall für automatisches Speichern (in Minuten)** stellen Sie ein, nach wie viel Minuten die Datei automatisch gespeichert wird.
- Als **Dateierweiterung für automatische Speicherung** ist „.xmi“ voreingestellt, kann aber nach Wunsch hier angepasst werden.

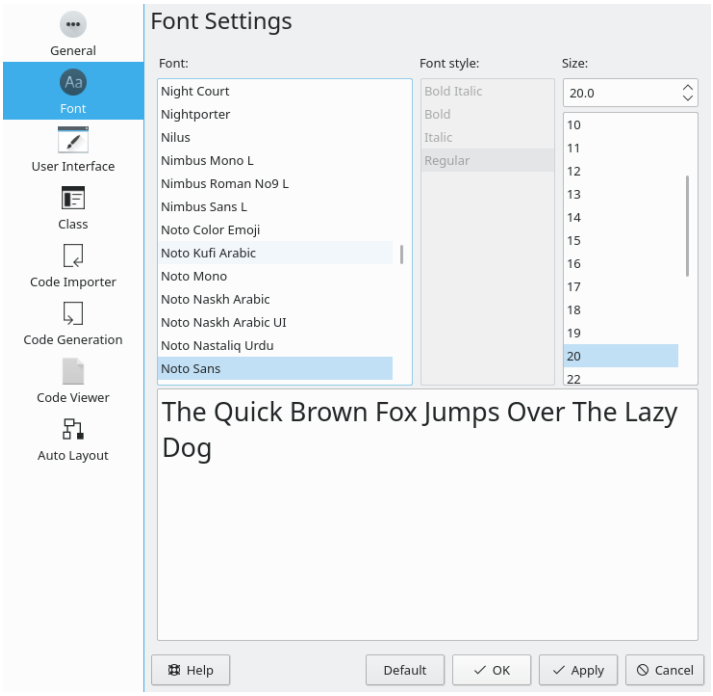
6.1.3 Start

- Ist **Letztes Projekt beim Start öffnen** gewählt, wird immer das letzte bearbeitete Projekt beim Start automatisch geöffnet.
- **Neues Projekt starten mit** legt fest, mit welchem UML-Diagrammtyp ein neues Projekt begonnen wird.
- **Standardsprache** ist eine Einstellung für die Programmiersprache, die als Voreinstellung verwendet wird.

6.1.4 Benachrichtigungen

- **Meldungen aktivieren** ist eine Einstellung, um alle oder nur eine beschränkte Anzahl von Meldungen anzuzeigen.
- Mit **Alle Meldungen zu „Assoziation löschen“ aktivieren** werden alle Meldungen dieser Art angezeigt.
- Mit **Alle Meldungen zu „Diagramm löschen“ aktivieren** werden alle Meldungen dieser Art angezeigt.

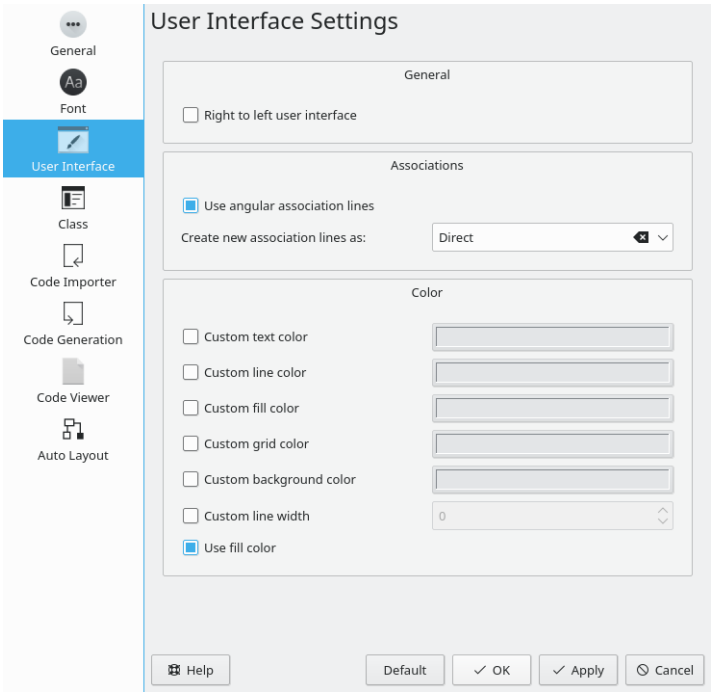
6.2 Schrift-Einstellungen



Optionen für Schrift-Einstellungen der Diagramme in Umbrello UML Modeller

Diese Einstellungen legen das Aussehen der Texte in Diagrammen fest. Sie können Schriftart, -stil und -Größe einstellen.

6.3 Einstellungen der Benutzeroberfläche



Optionen für die Einstellungen der Benutzeroberfläche in Umbrello UML Modeller

6.3.1 Allgemein

Bedienungsoberfläche „Rechts nach links“ stellt die Anwendung auf diese Schriftrichtung um.

6.3.2 Assoziationen

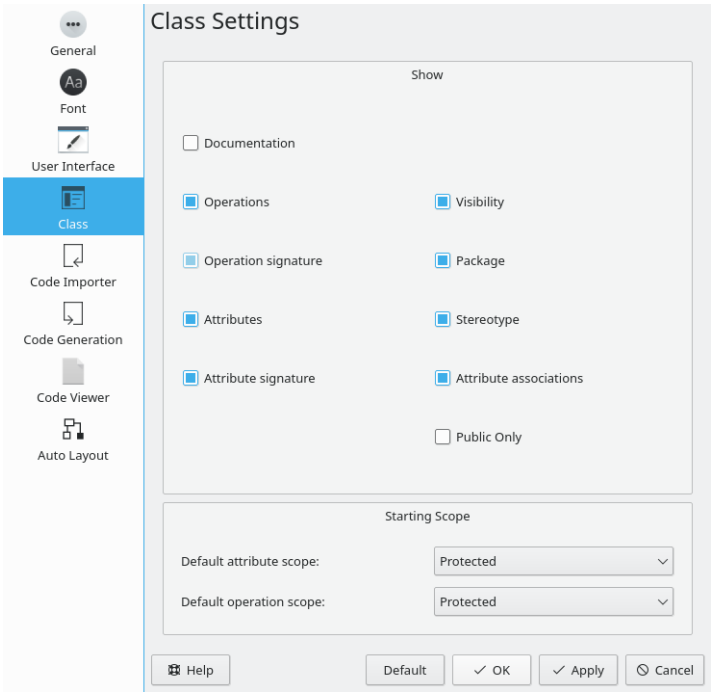
Mit **Eckige Assoziationslinien verwenden** können diese Linien beliebige Winkel haben.

Neue Assoziationslinien erstellen als ermöglicht die Auswahl von Linienstilen für Assoziationen.

6.3.3 Farbe

Hier können Sie die Farbe für Text, Linien, Füllung, Raster und Hintergrund wie auch die Linienstärke einstellen.

6.4 Klassen-Einstellungen



Optionen für Klassen-Einstellungen in Umbrello UML Modeller

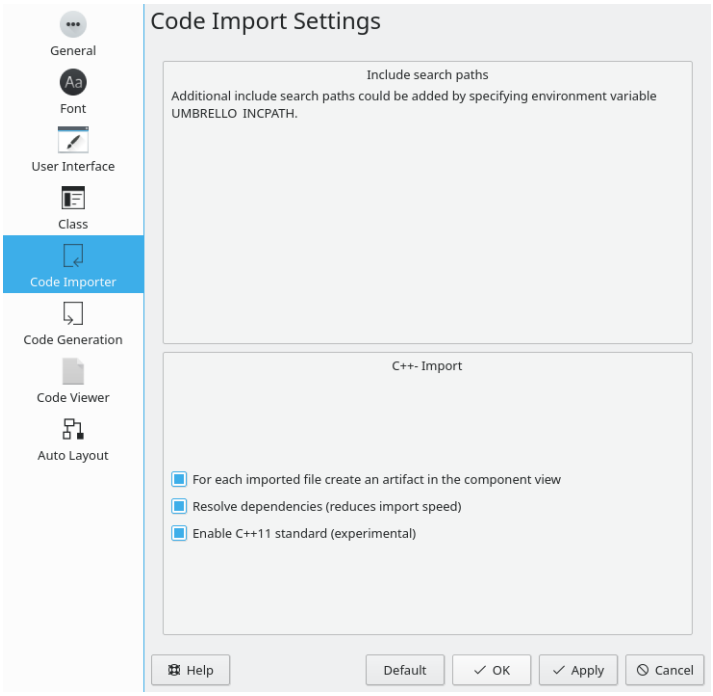
6.4.1 Anzeigen

In diesem Abschnitt legen Sie fest, welche Eigenschaften in einem Klassendiagramm angezeigt werden.

6.4.2 Startsichtbarkeit

Als Auswahl für die Sichtbarkeit von Attribut und Operation können „Public“, „Private“ oder „Protected“ eingestellt werden.

6.5 Einstellungen für Quelltext-Import



Optionen für Quelltext-Import in Umbrello UML Modeller

6.5.1 Suchpfade einbeziehen

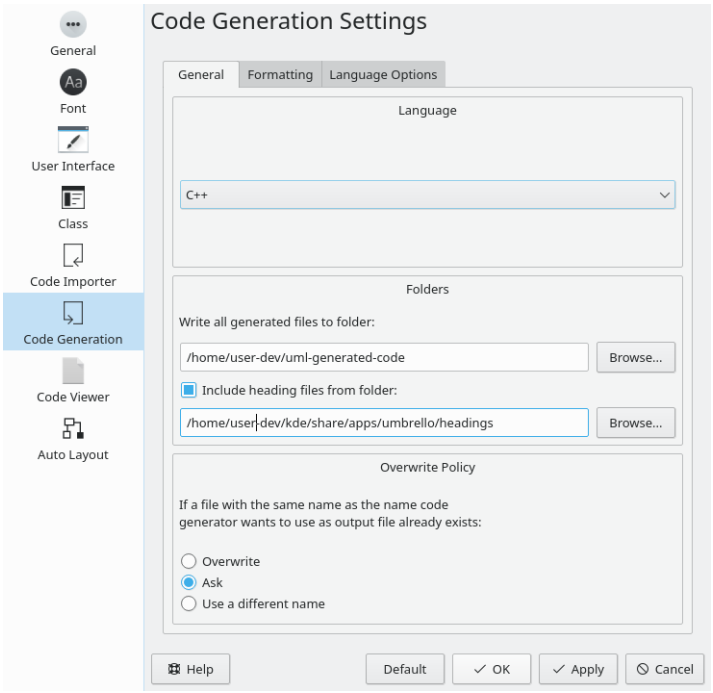
Es wird empfohlen, zur Verbesserung der Suche UMBRELLO_INCPATH als Umgebungsvariable zu verwenden.

6.5.2 C++-Import

- **Für jede importierte Datei ein Artefakt in der Komponentenansicht erstellen**: Das erstellte Artefakt kann dann in die Ansicht des Klassendiagramms gezogen werden, dann sind die Abhängigkeiten zusammen mit den Attributen und Methoden jeder Datei leicht zu erkennen.
- **Abhängigkeiten auflösen, verringert die Geschwindigkeit beim Import**: Stellt sicher, dass alle Dateiabhängigkeiten aufgelöst werden, die dann in den Klassenabhängigkeiten im Klassendiagramm angezeigt werden.
- **C++11-Standard aktivieren (Experimentell**: Einen experimentelle Funktion zur Anpassung an C++11, deaktivieren Sie sie, wenn nicht benötigt.

6.6 Einstellungen für Quelltext-Erstellung

6.6.1 Karteikarte „Allgemein“ der Einstellungen für Quelltext-Erstellung



Optionen für Allgemein-Einstellungen für Quelltext-Erstellung in Umbrello UML Modeller

Umbrello UML Modeller kann Quelltext in verschiedenen Programmiersprachen auf Basis ihres UML-Modells erzeugen und hilft dabei einen Anfang für die Implementierung zu schaffen. Der erzeugte Quelltext besteht aus den Klassendeklarationen, den Methoden und den Attributen. Man muss diese Hüllen lediglich ausfüllen, um die Klassenoperationen mit Funktionalität zu füllen.

6.6.1.1 Sprache

Wählen Sie die im Projekt verwendete Programmsprache. Zur Auswahl stehen ActionScript, Ada, C++, C#, D, IDL, Java, JavaScript, MYSQL, Pascal, Perl, PHP, PHP5, PastgreSQL, Python, Ruby, SQL, Tcl, Vala und XMLSchema.

6.6.1.2 Ordner

Ordner für generierte Dateien ist ein editierbares Feld für den gewünschten Pfad für generierte Dateien. Alternativ können Sie mit dem Symbol rechts daneben den Pfad auswählen.

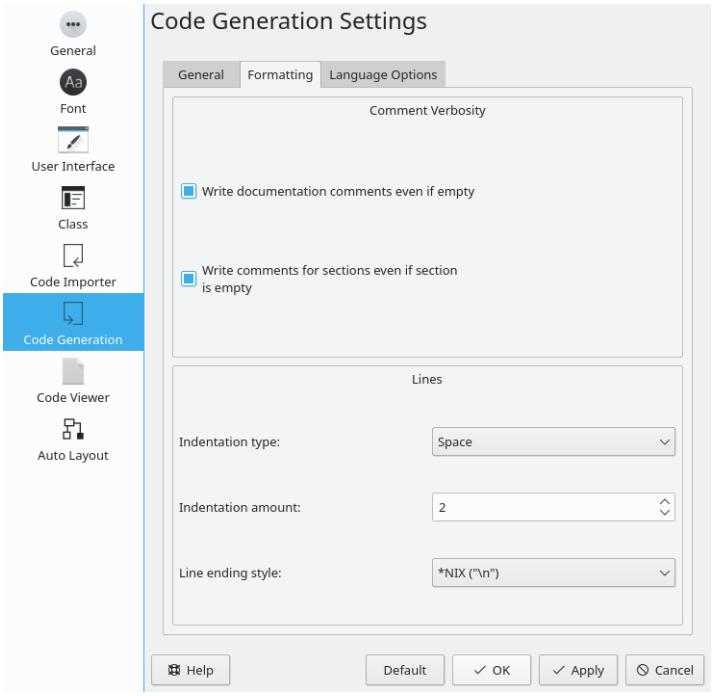
Ist **Einbeziehen der Header-Dateien aus Ordner** angekreuzt, kann der Pfad in einem editierbaren Feld eingeben oder mit dem Symbol rechts daneben ausgewählt werden.

6.6.1.3 Überschreib-Regelung

Wenn der Quelltext in den angegebenen Ordner generiert wird wenn eine Datei mit demselben Namen bereits vorhanden ist, bestimmt diese Einstellung, wie weiter verfahren wird.

- **Überschreiben** der Datei ohne Warnung oder Option.
- **Nachfragen**, ob die Datei überschrieben oder umbenannt wird.
- **Anderen Dateinamen verwenden**wenn die Datei bereits existiert, dann wird sie durch Anhängen eines Suffix umbenannt.

6.6.2 Karteikarte „Formatierung“ der Einstellungen für Quelltext-Erstellung



Optionen für Formatierung-Einstellungen für Quelltext-Erstellung in Umbrello UML Modeller

6.6.2.1 Umfang Quelltextkommentare

Erzeugt Dokumentations-Kommentare, selbst wenn diese leer sind: Kommentare für Klassen und Funktionen werden generiert.

Erzeugt Kommentare für Abschnitte, selbst wenn diese leer sind: Kommentare für die Abschnitte „private“,„protected“ und „public“ werden generiert.

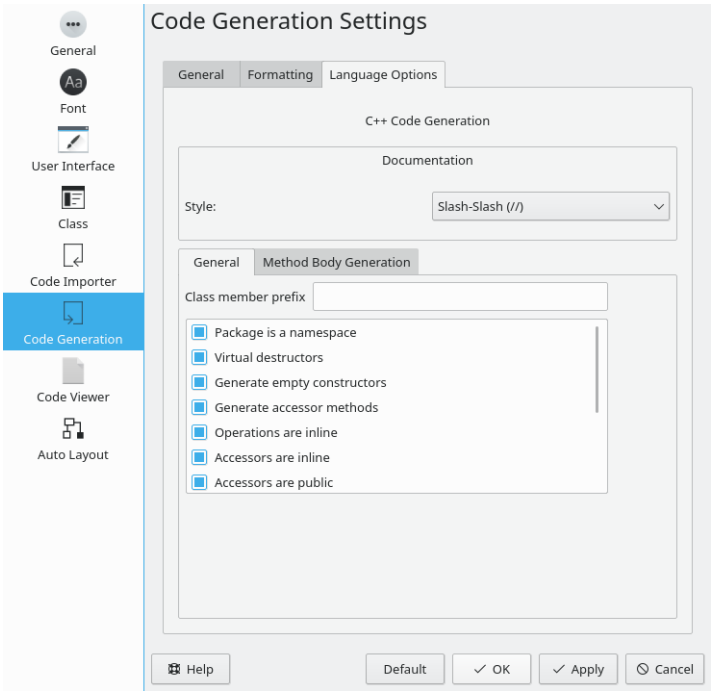
6.6.2.2 Zeilen

Quelltexteinrückung: Auswahl zwischen keine Quelltexteinrückung, Tabulator und Leerzeichen.

Umfang der Quelltexteinrückung: Geben Sie die Anzahl der Leerzeichen für einen Tabulator oder die Zahl der Leerzeichen selbst an.

Zeilenende-Stil ist eine Auswahl zwischen der Art in *NIX, Windows und Mac.

6.6.3 Spracheinstellungen



Optionen für Spracheinstellungen für Quelltext-Erstellung in Umbrello UML Modeller

Auf dieser Seite gibt es zurzeit nur Einstellungen für die Sprache C++.

6.6.3.1 C++-Quelltext-Erstellung

6.6.3.1.1 Dokumentation

Unter **Stil** können Sie „/** */“ oder „// “ für Kommentare zur Dokumentation wählen.

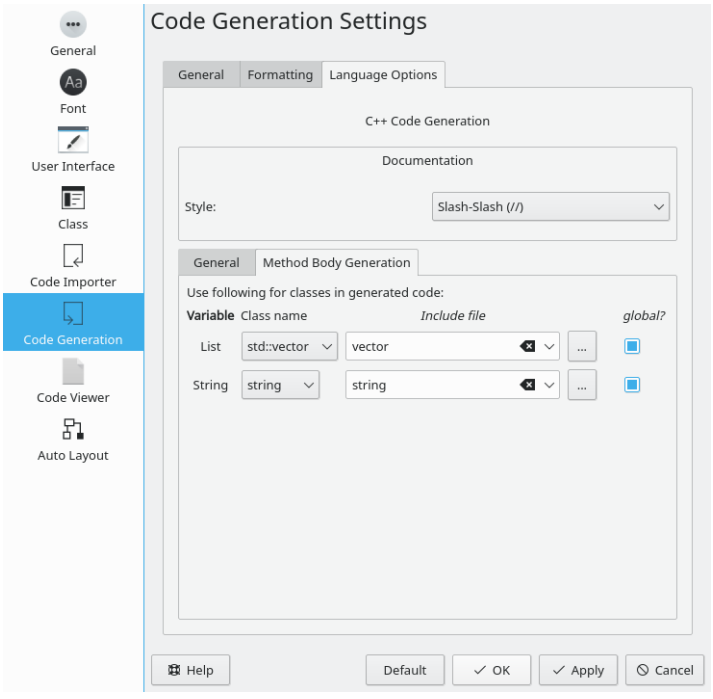
6.6.3.1.2 Allgemein

Auf der Karteikarte **Allgemein** der **Spracheinstellungen** finden Sie mehrere Einstellungen zur Generierung des Quelltextes.

- **Präfix für Klassen-Member**
  Hier können Sie ein Präfix eingeben, das beim Generieren von Klassen-Member den Namen vorangestellt wird.
- **Paket als Namensraum**
  Namensräume bieten eine Methode zur Vermeidung von Namenskonflikten in großen Projekten. Symbole, die innerhalb eines Namensraum-Blocks deklariert werden, befinden sich in einem benannten Bereich. Damit wird verhindert, dass sie mit gleichnamigen Symbolen in anderen Bereichen verwechselt werden.
- **Virtuelle Destruktoren**
  Auch wenn Destruktoren nicht vererbt werden, wenn eine Basisklasse ihren Destruktor virtuell deklariert, überschreibt der abgeleitete Destruktor ihn immer. Dies ermöglicht es möglich, dynamisch zugewiesene Objekte vom polymorphen Typ durch Zeiger auf die Basisklasse zu löschen.

- **Leere Konstruktoren erzeugen**
  Damit werden Konstruktoren mit leeren Klammern erzeugt.
- **Zugriffsmethoden erzeugen**
  Erstellt Methoden zum Zugriff auf Datentypen.
- **Operationen sind inline**
  Generiert die Methode als Inline, das kann aber von den Compilern ignoriert werden.
- **Zugriffsmethoden sind inline**
  Methoden, die auf die Daten der Klasse zugreifen, werden Inline generiert, das kann aber von den Compilern ignoriert werden.
- **Zugriffsmethoden sind public**
  Methoden, die als public generiert werden, sind für jede Instanz der Klasse verfügbar.
- **Getter-Methoden mit Präfix „get“ erstellen**
  Damit wird das Präfix „get“ für den Namen aller Methoden verwendet, die Daten der Klasse lesen/zurückgeben.
- **Präfix „[a-zA-Z]_“ von Namen der Zugriffsmethoden entfernen**
  Wurde ein Präfix in der Option **Präfix für Klassen-Member** eingegeben, wird es mit dieser Option entfernt.
- **Zugriffsmethoden beginnen mit Großbuchstaben**
  Damit wird das erste Zeichen des Methodennamens groß geschrieben.
- **„\“ anstatt „@“ als Dokumentations-Tag verwenden**
  Eine Tag-Auswahl zur Verwendung bei der Dokumentation von Parametern einer Methode.

6.6.3.1.3 Quelltext-Erstellung für Methoden



Optionen für Quelltext-Erstellung für Methoden in den Spracheinstellungen für Quelltext-Erstellung in Umbrello UML Modeller

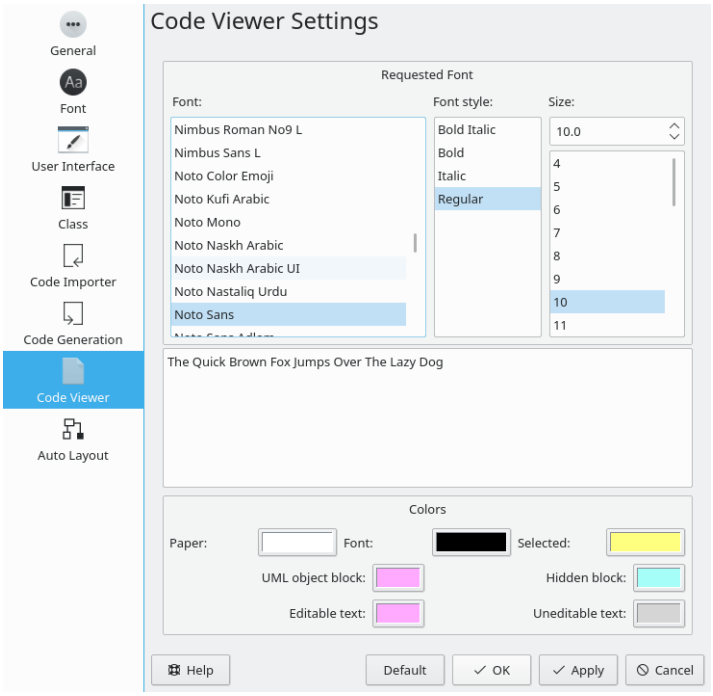
Liste

Einstellung von „QPtrList“. „vector“ oder „std::vector“ für den Listentyp. Es folgt ein editierbares oder auswählbares Feld, um die Include-Datei anzugeben, sowie ein Knopf zum Suchen und Auswählen der die Include-Datei. Es gibt auch eine Einstellung, um die Liste global zu machen.

Zeichenfolge

Einstellung von „string“ oder „QString“ für Zeichenfolgen. Es folgt ein editierbares oder auswählbares Feld, um die Include-Datei anzugeben, sowie ein Knopf zum Suchen und Auswählen der die Include-Datei. Es gibt auch eine Einstellung, um die Zeichenfolge global zu machen.

6.7 Einstellungen für Quelltext-Betrachter

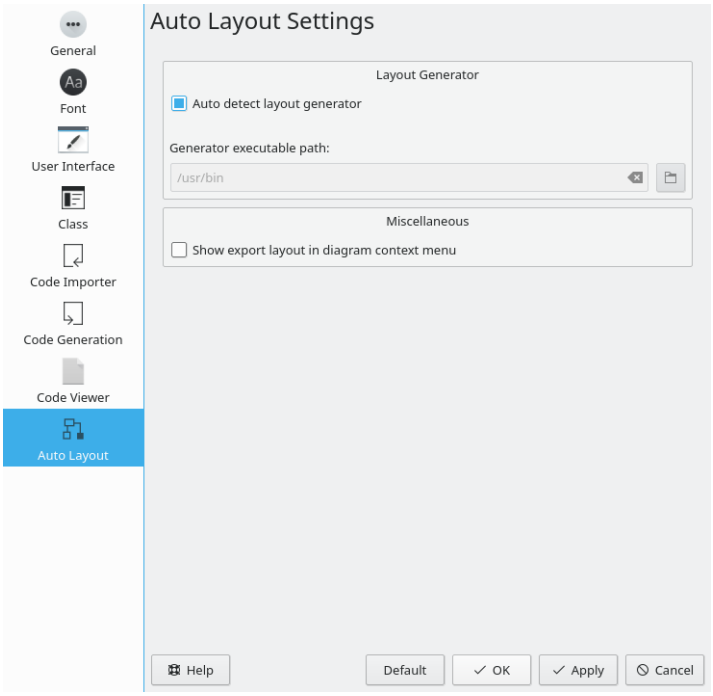


Optionen für Quelltext-Betrachter in Umbrello UML Modeller

Ermöglicht das Anpassen des Quelltext-Betrachters. Der Abschnitt **Gewünschte Schrift** ermöglicht die Auswahl der Schriftart, des Schriftstils und der Schriftgröße. Unter den Auswahlmöglichkeiten wird eine Vorschau Ihrer Einstellungen angezeigt.

im Abschnitt **Farben** können die Farben für Papier, Schrift, Auswahl, UML-Objekt-Block, versteckte Blöcke, änderbaren Text und nicht änderbaren Text geändert werden. Klicken Sie dazu auf das Feld neben der zugehörigen Beschriftung.

6.8 Automatische Layout-Einstellungen



Optionen für Automatische Layout-Einstellungen in Umbrello UML Modeller

Layout-Generator automatisch erkennen

Diese automatische Layout-Funktion hängt von den Layout-Generatoren aus dem Paket GraphViz ab, das normalerweise mit Umbrello durch das Paketverwaltungssystem installiert wird. Umbrello enthält eingebaute Funktionen zur Erkennung des Installationspfads dieser Layout-Generatoren. Wenn dieses erforderliche Programm an einem anderen Ort installiert ist, kann ein anderer Installationspfad angegeben werden.

Exportlayout im Diagramm-Kontextmenü anzeigen

Der Export von „Dot“-Dateien wird mithilfe des Export-Layouts durchgeführt.Ist dies angekreuzt, wird das Export-Layout zur Liste der verfügbaren Diagramm-Layouts hinzugefügt und ermöglicht die schnelle Exportvorschau von Dot-Grafiken.

Kapitel 7

Autoren und Geschichte

Dieses Projekt wurde von Paul Hensgen als eines seiner Universitätsprojekte initiiert. Der Originalname des Projektes lautete UML Modeller. Paul hatte die Entwicklung bis Ende 2001, als die Version 1.0 erschien, vorangetrieben.

Version 1.0 bot schon eine Vielzahl von Funktionen. Nachdem Paul bei seiner Universität die Arbeit eingereicht hatte, konnten andere Entwickler dem Projekt beitreten, um es weiter zu verbessern. So wurde das ursprünglich binäre Dateiformat zugunsten von XML-Dateien fallen gelassen, es wurde Unterstützung für weitere UML-Diagramme hinzugefügt, genauso wie die Quelltexterzeugung und das Einlesen von Quelltext, um nur ein paar wenige Verbesserungen zu nennen.

Paul musste sich im Sommer 2002 aus dem Entwicklungsteam leider zurückziehen, aber das Programm lebt weiter und entwickelt sich, wie andere Open Source Software. Es wird heute von einer Gruppe von Entwicklern von überall aus der Welt gepflegt. Im September 2002 wurde der Projektname von UML Modeller in Umbrello UML Modeller geändert. Dafür gab es gute Gründe, z. B. war „uml" — wie es allgemein genannt wurde — zu allgemein und es gab deswegen Probleme mit einigen Distributionen. Und außerdem sind die Entwickler der Meinung, dass Umbrello ein viel coolerer Name ist.

Umbrello UML Modellers Entwicklung, genauso wie die Diskussion wo in zukünftigen Versionen die Schwerpunkte liegen sollen, ist offen und wird über das Internet geführt. Falls sie zu dem Projekt beitragen wollen, zögern sie nicht die Entwickler zu kontaktieren! Es gibt viele Möglichkeiten, wie man zu Umbrello UML Modeller beitragen kann:

- Fehler berichten oder Vorschläge für Verbesserungen machen
- Fehler beheben oder neue Funktionen hinzufügen
- Eine gute Dokumentation schreiben oder Umbrello in eine andere Sprache übersetzen
- Und natürlich mit uns programmieren!

Es ist ersichtlich, dass es viele Möglichkeiten gibt, zum Projekt beizutragen. Alle Aufgaben dabei sind gleich wichtig und jeder ist herzlich willkommen uns zu helfen.

Die Entwickler von Umbrello UML Modeller sind per E-Mail erreichbar unter: umbrello-devel@kde.org.

Kapitel 8

Copyright

Copyright 2001, Paul Hensgen

Copyright 2002-2020 Die Umbrello UML Modeller-Autoren.

Diese Dokumentation ist unter den Bedingungen der GNU Free Documentation License veröffentlicht.

Dieses Programm ist unter den Bedingungen der GNU General Public License veröffentlicht.