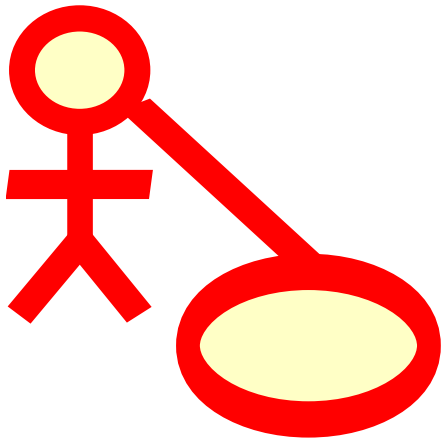
Handbok Umbrello UML Modeller



Innehåll

Sammanfattning

Umbrello UML Modeller hjälper till med utvecklingsprocessen för programvara genom att använda industristandarden Unified Modelling Language (UML) för att göra det möjligt att skapa diagram för att konstruera och dokumentera system.

Kapitel 1

Inledning

Umbrello UML Modeller är ett UML-diagramverktyg som stöder dig i utvecklingsprocessen av programvara. I synnerhet under analys- och konstruktionsfaserna av processen, hjälper Umbrello UML Modeller dig att skapa en produkt med hög kvalitet. UML kan också användas för att dokumentera programvarukonstruktioner för att hjälpa dig och dina medutvecklare.

Att ha en bra modell av programvaran är det bästa sättet att kommunicera med andra utvecklare som arbetar med projektet och med kunder. En bra modell är ytterst viktig för medelstora och stora projekt, men är också mycket användbar för små. Även om du arbetar på ett litet enmansprojekt, har du nytta av en bra modell, eftersom den ger dig en överblick, som hjälper dig att koda rätt från början.

UML är ett diagramspråk som används för att beskriva sådana modeller. Du kan representera dina idéer i UML med olika sorters diagram. Umbrello UML Modeller 2.11 stöder följande typer:

- Klassdiagram
- Sekvensdiagram
- Samarbetsdiagram
- Användningsfallsdiagram
- Tillståndsdiagram
- Aktivitetsdiagram
- Komponentdiagram
- Utplaceringsdiagram
- Objektsambandsdiagram

Mer information om UML finns på OMG:s webbplats, http://www.omg.org, de som skapade UML-standarden.

Vi hoppas att du trivs med Umbrello UML Modeller, och att det hjälper dig att skapa programvara med hög kvalitet. Umbrello UML Modeller är ett fritt verktyg, och det enda som vi ber dig är att rapportera eventuella fel, problem eller förslag till Umbrello UML Modellers utvecklare på umbrello-devel@kde.org eller https://bugs.kde.org.

Kapitel 2

Grundläggande UML

2.1 Om UML

Det här kapitlet ger en snabb översikt av grunderna i UML. Kom ihåg att det här inte är en heltäckande UML-handledning, utan bara en kortfattad introduktion till UML som kan läsas som UML-handledning. Om du skulle vilja lära dig mer om Unified Modelling Language, eller om allmän analys och konstruktion av programvara, hänvisas du till några av de många böcker som är tillgängliga i ämnet. Det finns också många handledningar på Internet, som du kan använda som startpunkt.

Unified Modelling Language (UML) är ett diagrambaserat språk eller notation för att specificera, visualisera och dokumentera modeller av objektorienterad programvara. UML är inte en utvecklingsmetod, vilket betyder att det inte talar om för dig vad du ska göra först och vad du ska göra därefter, eller hur du ska konstruera system, men det hjälper till att visualisera konstruktionen och kommunicera med andra. UML styrs av Object Management Group (OMG), och är industristandard för att grafiskt beskriva programvara.

UML är konstruerat för design av objektorienterad programvara, och har begränsad användning för andra programmeringsparadigmer.

UML är uppbyggt av många modelleringselement som representerar olika delar av programsystemet. UML-elementen används för att skapa diagram, som representerar en viss del, eller en synvinkel av systemet. Följande sorters diagram stöds av Umbrello UML Modeller:

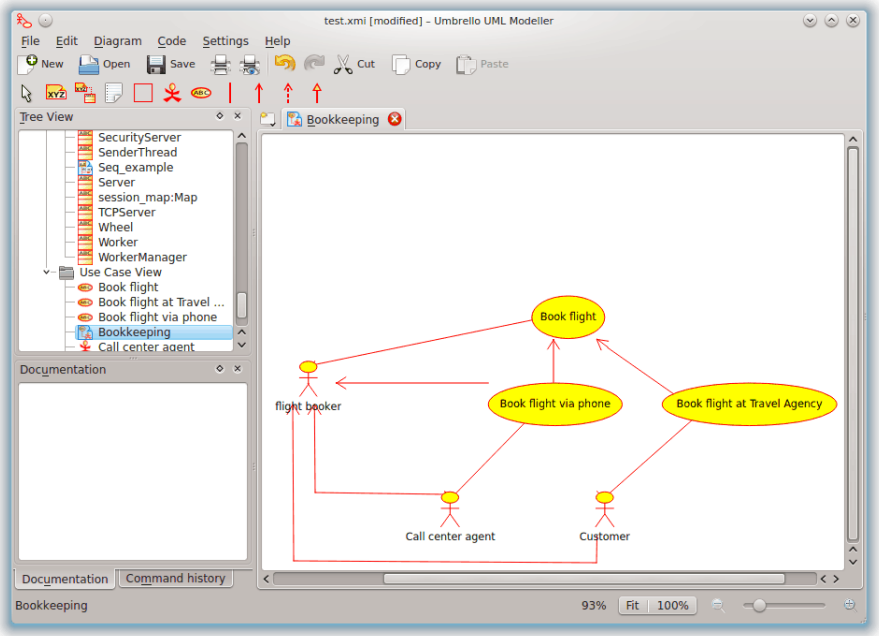
- *Användningsfallsdiagram* visar aktörer (människor eller andra användare av systemet), användningsfall (scenarion då de använder systemet), och deras samband
- *Klassdiagram* visar klasser, och sambanden mellan dem
- *Sekvensdiagram* visar objekt och deras samband, med betoning på utbyte av meddelanden mellan objekt i kronologisk ordning
- *Samarbetsdiagram* visar objekt och deras samband, med betoning på objekten som deltar i utbytet av meddelanden
- *Tillståndsdiagram* visar tillstånd, tillståndsändringar och händelser för ett objekt eller en del av systemet
- *Aktivitetsdiagram* visar aktiviteter, tillstånd och tillståndsändringar hos objekt och händelser som sker i någon del av systemet
- *Komponentdiagram* visar programmeringskomponenter på hög nivå (som Kparts eller Java Beans).
- *Utplaceringsdiagram* visar komponenternas instanser och deras inbördes förhållanden.
- *Objektsambandsdiagram* visar data, samt sambanden och begränsningarna mellan dataobjekt.

2.2 UML-element

2.2.1 Användningsfallsdiagram

Användningsfallsdiagram beskriver samband och beroenden mellan en grupp *användningsfall* och aktören som deltar i processen.

Det är viktigt att observera att användningsfallsdiagram inte är lämpade att representera konstruktionen, och kan inte beskriva systemets innanmäte. Användningsfallsdiagram är avsedda att möjliggöra kommunikation med framtida användare av systemet, och med kunden. De är till särskild hjälp för att avgöra vilka funktioner som krävs att systemet ska ha. Med andra ord talar användningsfallsdiagram om *vad* systemet ska göra, men de anger inte — och kan inte ange — *hur* detta ska åstadkommas.



Umbrello UML Modeller som visar ett användningsfallsdiagram

2.2.1.1 Användningsfall

Ett *användningsfall* beskriver — från aktörernas synvinkel — en samling aktiviteter i ett system, som ger upphov till ett konkret, påtagligt resultat.

Användningsfall är beskrivningar av typisk växelverkan mellan användarna av ett system och systemet själv. De representerar systemets yttre gränssnitt, och anger en sorts krav på vad systemet ska göra (kom ihåg, bara vad, inte hur).

Vid arbete med användningsfall, är det viktigt att komma ihåg några enkla regler:

- Varje användningsfall hör ihop med minst en aktör
- Varje användningsfall har ett ursprung (dvs. en aktör)
- Varje användningfall leder till ett relevant resultat (ett resultat med 'affärsvärde').

Användningfall kan också ha samband med andra användningfall. De tre mest typiska sorters samband mellan användningfall är:

- «*include*» (innehåller), vilket anger att användningsfallet äger rum *inne i* ett annat användningsfall
- «*extends*» (utökar), vilket anger att i vissa fall, eller vid något tillfälle (som kallas en utökningspunkt), kommer ett användningsfall att utökas av ett annat.
- *Generalisering* anger att ett användningfall ärver egenskaperna för 'super'-användningsfallet, och kan överskrida några av dem, eller lägga till nya på samma sätt som arv mellan klasser.

2.2.1.2 Aktör

En aktör är en extern enhet (utanför systemet) som växelverkar med systemet genom att delta i (och ofta inleda) ett användningsfall. Aktörer kan i verkligheten vara människor (till exempel användare av systemet), andra datorsystem eller yttre händelser.

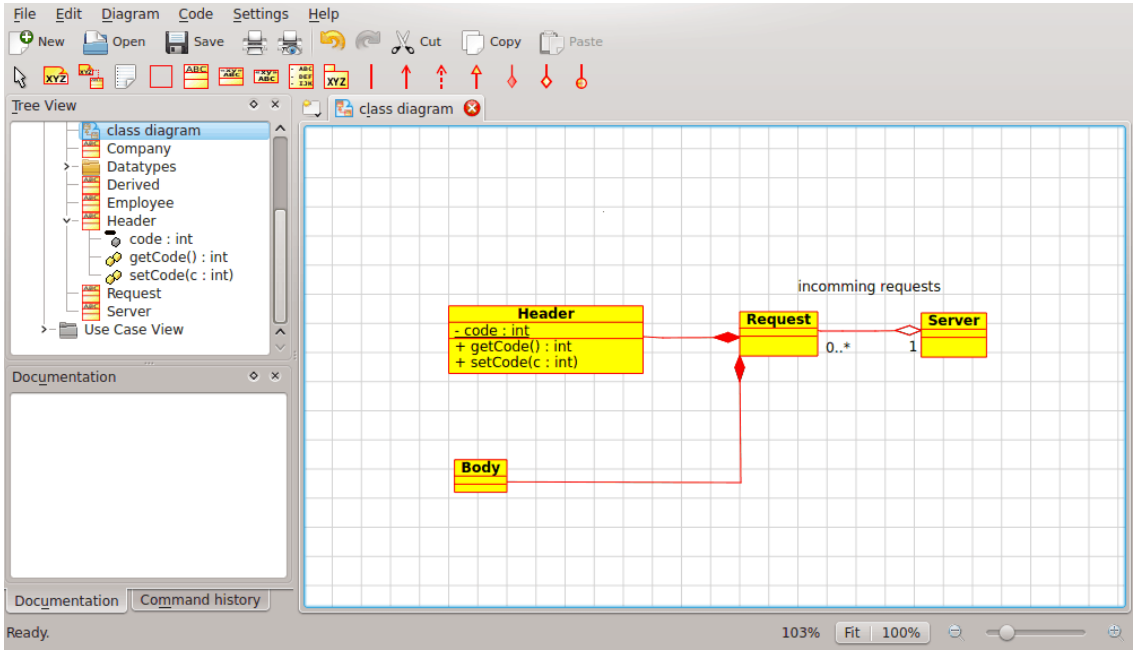
Aktörer representerar inte *fysiska* människor eller system, utan deras *roll*. Det betyder att när en person växelverkar med systemet på olika sätt (antar olika roller) representeras han med flera aktörer. En person som till exempel ger kundstöd via telefon och tar emot beställningar från kunden till systemet, skulle representeras av aktören 'kundstödspersonal' och aktören 'försäljningsassistens'.

2.2.1.3 Beskrivning av användningsfall

En beskrivning av ett användningsfall är en textbaserad berättelse om användningsfallet. Det är ofta i form av en anteckning eller ett dokument som på något sätt är länkat till användningsfallet, och förklarar processerna eller aktiviteterna som äger rum i användningsfallet.

2.2.2 Klassdiagram

Klassdiagram visar de olika klasserna som bygger upp ett system och hur de relateras till varandra. Klassdiagram sägs vara 'statiska' diagram, eftersom de visar klasserna, tillsammans med deras metoder och attribut, samt det statiska förhållandet mellan dem: vilka klasser som 'känner till' andra klasser, eller vilka klasser som 'är en del' av andra klasser, men visar inte metodanrop mellan dem.

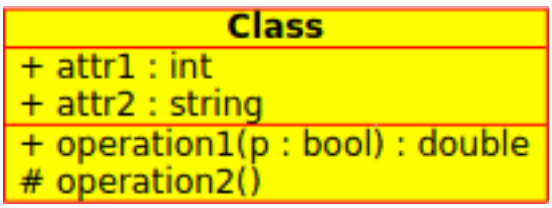


Umbrello UML Modeller som visar ett klassdiagram

2.2.2.1 Klass

En klass definierar attributen och metoderna för en mängd objekt. Alla objekt av klassen (instanser av klassen) delar samma beteende, och har samma mängd attribut (varje objekt har sin egen uppsättning). Termen 'typ' används ibland istället för klass, men det är viktigt att nämna att de två inte är samma sak, och att typ är en mer generell term.

Klasser i UML representeras av rektanglar, med klassens namn, och kan också visa klassens attribut och operationer i två 'fack' inne i rektangeln.



Visuell representation av en klass i UML

2.2.2.1.1 Attribut

Attribut i UML visas åtminstone med sina namn, och kan också visas med typ, ursprungligt värde och andra egenskaper. Attribut kan också visas med synlighet:

- \+ Betyder *öppna* (public) attribut
- # Betyder *skyddade* (protected) attribut
- – Betyder *privata* (private) attribut

2.2.2.1.2 Operationer

Operationer (metoder) visas också åtminstone med sina namn, och kan också visas med parametrar och returtyper. Operationer, precis som attribut, kan visas med sin synlighet:

- + Betyder *öppna* (public) operationer
- # Betyder *skyddade* (protected) operationer
- - Betyder *privata* (private) operationer

2.2.2.1.3 Mallar

Klasser kan ha mallar, ett värde som används för en ospecificerad klass eller typ. Malltypen anges när klassen initieras (dvs. ett objekt skapas). Mallar finns i modern C++ och kommer att introduceras i Java 1.5, där de kallas Generics.

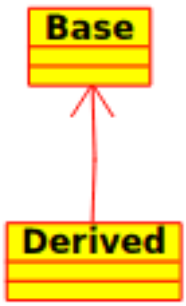
2.2.2.2 Klassassociationer

Klasser kan relateras till (associeras med) varandra på olika sätt:

2.2.2.2.1 Generalisering

Arv är ett av de grundläggande koncepten i objektorienterad programmering, där en klass 'erhåller' alla attribut och operationer från klassen den ärver från, och kan överskrida/ändra några av dem, samt lägga till fler egna attribut och operationer.

En *generalisering* mellan två klasser i UML, placerar dem i en hierarki som representerar arvkonceptet för en härledd klass från en basklass. Generaliseringar i UML representeras med en linje som binder samman de två klasserna, med en pil på basklassens sida.



Visuell representation av en generalisering i UML

2.2.2.2.2 Associationer

En association representerar ett samband mellan klasser, och ger den allmänna semantiken och strukturen för många typer av 'förbindelse' mellan objekt.

Associationer är mekanismen som tillåter att objekt kommunicerar med varandra. De beskriver förbindelsen mellan olika klasser (förbindelsen mellan de verkliga objekten kallas objektförbindelse, eller *länk*).

Associationer kan ha en roll, som anger associationens syfte, och kan vara enkelriktade eller ömsesidiga (anger om två objekt som deltar i sambandet kan skicka meddelanden till det andra,

eller om bara ett av dem känner till det andra). Varje ända av associationen har också ett mångfaldsvärde, som bestämmer hur många objekt på denna sida av associationen som kan relatera till ett objekt på andra sidan.

Associationer i UML representeras som linjer som binder samman klasserna som deltar i sambandet, och kan också visa rollen och mångfalden för var och en av deltagarna. Mångfald visas som ett intervall [minimum..maximum] med icke-negativa värden, med en asterisk (*) på maximumsidan som representerar oändlighet.

Visuell representation av en association i UML

2.2.2.2.3 Aggregering

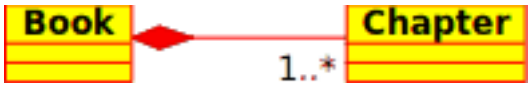
Aggregeringar är särskilda sorters associationer, där de två deltagande klasserna inte har en likvärdig status, utan utgör ett 'helhet-del' samband. En aggregering beskriver hur klassen som intar rollen som helhet, är sammansatt av (har) andra klasser, som intar rollerna som delar. Klassen som fungerar som helhet har alltid mångfalden ett, för aggregeringar.

Aggregeringar i UML representeras av en association som visar en romb på sidan som hör till helheten.

Visuell representation av en aggregeringsrelation i UML

2.2.2.2.4 Sammansättning

Sammansättningar är associationer som representerar *mycket starka* aggregeringar. Det betyder att sammansättningar också formar helhet-del samband, men att sambandet är så starkt att delarna inte kan existera för sig själv. De finns bara inne i helheten, och om helheten förstörs, försvinner också delarna.
Sammansättning i UML representeras av en ifylld romb på sidan som hör till helheten.



2.2.2.3 Andra objekt i klassdiagram

Klassdiagram kan innehålla flera andra objekt förutom klasser.

2.2.2.3.1 Gränssnitt

Gränssnitt är abstrakta klasser vilket betyder att instanser inte direkt kan skapas från dem. De kan innehålla operationer men inga attribut. Klasser kan ärva från gränssnitt (via en realisationsassociation) och instanser kan därefter skapas av klasserna.

2.2.2.3.2 Datatyper

Datatyper är primitiver som typiskt är inbyggda i ett programspråk. Vanliga exempel omfattar heltal och en boolesk typ. De kan inte ha samband med klasser, men klasser kan ha samband med dem.

2.2.2.3.3 Uppräkningstyper

Uppräkningstyper är enkla listor med värden. Ett typiskt exempel är en uppräkningstyp av veckodagar. Medlemmar i en uppräkningstyp kallas uppräkningsvärden. Som datatyper kan de inte ha samband med klasser, men klasser kan ha samband med dem.

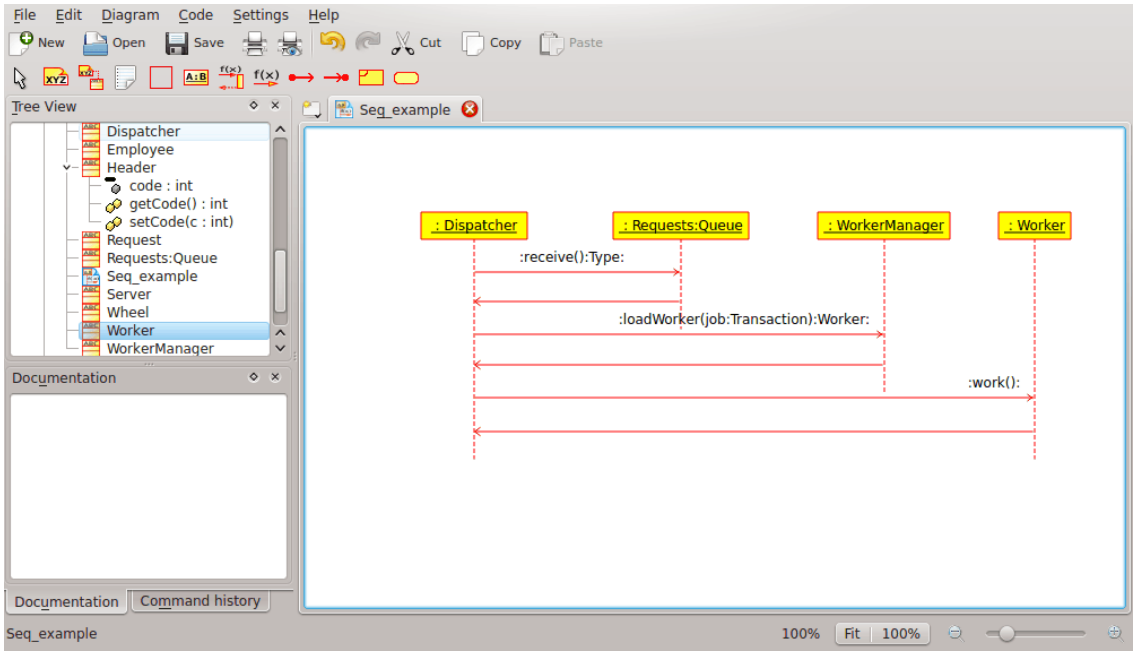
2.2.2.3.4 Paket

Paket representerar namnrymder i ett programspråk. I ett diagram används de för att representera delar i ett system som innehåller mer än en klass, kanske hundratals klasser.

2.2.3 Sekvensdiagram

Sekvensdiagram visar utbyte av meddelanden (dvs. metodanrop) mellan flera objekt, i en specifik, tidsbegränsad situation. Sekvensdiagram lägger särskild vikt vid ordningen och tiden då meddelanden till objekt skickas.

Objekt representeras av vertikala streckade linjer i sekvensdiagram, med objektets namn överst. Tidsaxeln är också vertikal, och ökar neråt, så att meddelanden skickas från ett objekt till ett annat i form av pilar med operationer och parameternamn.



Umbrello UML Modeller som visar ett sekvensdiagram

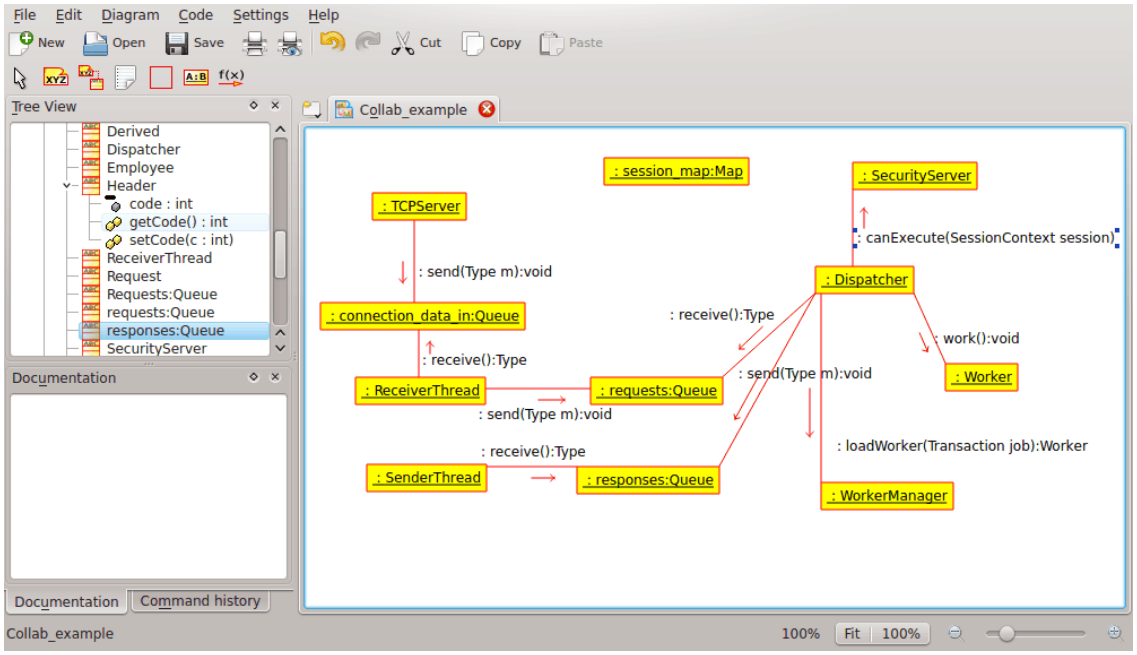
Meddelanden kan antingen vara synkrona, den normala typen för meddelandeanrop där kontrollen övergår till det anropade objektet till metoden har kört färdigt, eller asynkront där kontrollen direkt återgår till anropande objekt. Synkrona meddelanden har en vertikal ruta vid sidan om det anropade objektet, för att visa programflödet.

2.2.4 Samarbetsdiagram

Samarbetsdiagram visar växelverkan mellan objekt som deltar i en speciell situation. Det här är mer eller mindre samma information som visas i sekvensdiagram, men där läggs vikten vid hur

växelverkan sker i tiden, medan samarbetsdiagram lägger vikten vid sambanden mellan objekten och deras topologi.

I samarbetsdiagram representeras meddelanden från ett objekt till ett annat med pilar, som visar meddelandets namn, parametrar och meddelandesekvensen. Samarbetsdiagram är särskilt lämpade att visa ett särskilt programflöde eller situation, och är bland de bästa diagramtyperna för att snabbt demonstrera eller förklara en process i programmets logik.



Umbrello UML Modeller som visar ett samarbetsdiagram

2.2.5 Tillståndsdiagram

Tillståndsdiagram visar de olika tillstånd ett objekt har under sin livstid, och de stimuli som orsakar att objektet ändrar sitt tillstånd.

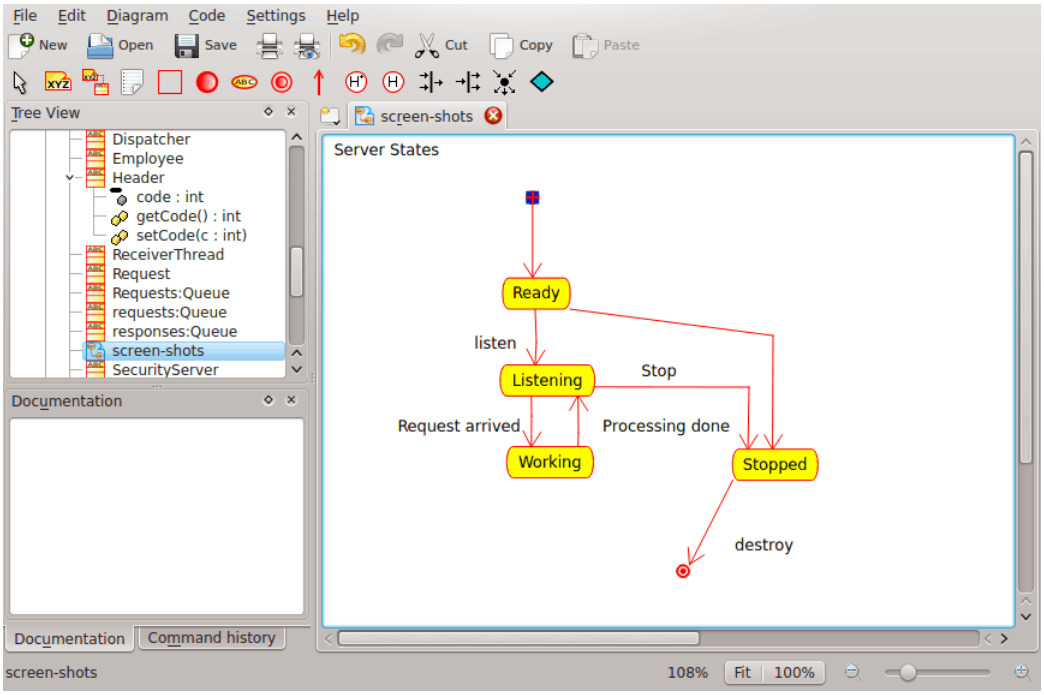
Tillståndsdiagram ser objekt som *tillståndsmaskiner* eller finita automater, som kan vara i något av en mängd begränsade tillstånd och som kan ändra sina tillstånd via något av ett begränsat antal stimuli. Ett objekt av typen *Nätserver*, kan till exempel vara i något av följande tillstånd under sin livstid:

- Klar
- Lyssnar
- Arbetar
- Stoppad

och händelserna som kan göra att ett objekt byter tillstånd är

- Objektet skapas
- Objektet tar emot meddelandet att lyssna
- En klient begär en anslutning via nätverket
- En klient avslutar en begäran

- En begäran körs och avslutas
- Objektet tar emot meddelandet att stoppa
- etc



Umbrello UML Modeller som visar ett tillståndsdiagram

2.2.5.1 Tillstånd

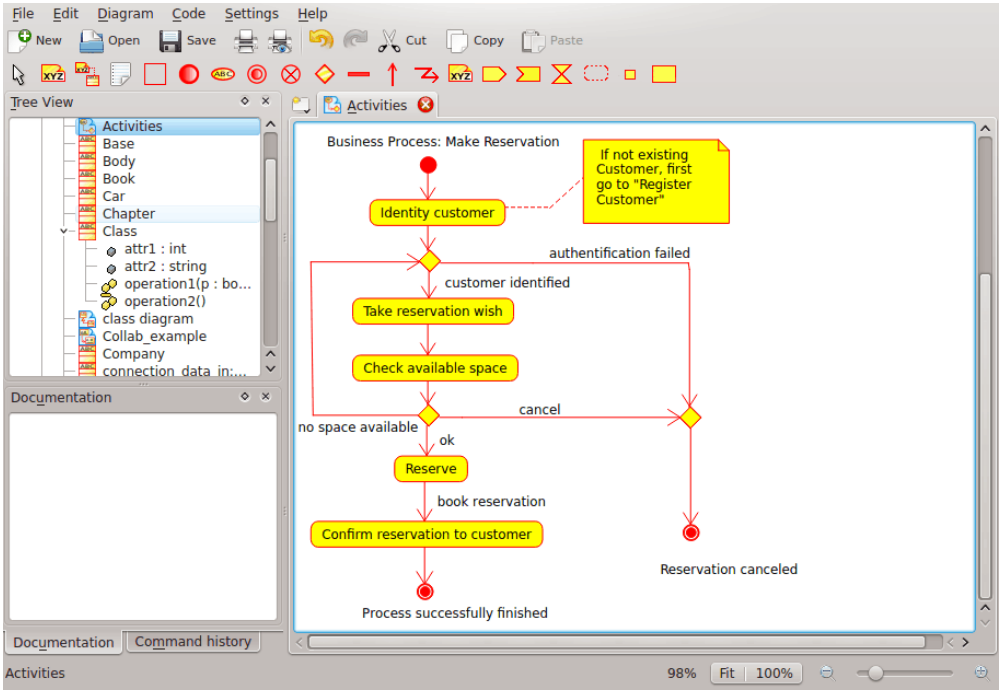
Tillstånd är byggblocken i tillståndsdiagram. Ett tillstånd hör till exakt en klass, och representerar en summering av de värden klassens attribut kan inta. Ett UML-tillstånd beskriver det interna tillståndet för ett objekt av en viss klass.

Observera att inte varje ändring av något av ett objekts attribut ska representeras som ett tillstånd, utan bara de ändringar som väsentligt kan påverka objektets arbete.

Det finns två speciella typer av tillstånd: start och slut. De är speciella på det sättet att det inte finns någon händelse som kan göra att ett objekt återgår till sitt starttillstånd, och på samma sätt finns det ingen händelse som gör det möjligt för ett objekt att lämna sitt sluttillstånd när det väl har nåtts.

2.2.6 Aktivitetsdiagram

Aktivitetsdiagram beskriver en följd av händelser i ett system, med hjälp av aktiviteter. Aktivitetsdiagram är en speciell form av tillståndsdiagram, som bara (eller i huvudsak) innehåller aktiviteter.



Umbrello UML Modeller som visar ett aktivitetsdiagram

Aktivitetsdiagram liknar procedurella flödesdiagram, med skillnaden att alla aktiviteter är klart länkade till objekt.

Aktivitetsdiagram hör alltid ihop med en *klass*, en *operation* eller ett *användningsfall*.

Aktivitetsdiagram stöder sekvens- samt parallella aktiviteter. Parallell körning representeras med ikonen Dela upp/samla ihop, och det är inte viktigt för aktiviteter som kör parallellt i vilken ordning de utförs (de kan köras samtidigt eller en i taget).

2.2.6.1 Aktivitet

En aktivitet är ett enda steg i en process. En aktivitet är ett tillstånd i systemet med intern aktivitet och åtminstone en utgående övergång. Aktiviteter kan också ha mer än en utgående övergång, om de har olika villkor.

Aktiviteter kan bygga upp hierarkier, vilket betyder att en aktivitet kan bestå av flera 'detaljaktiviteter', där inkommande och utgående övergångar måste passa ihop med de inkommande och utgående övergångarna i detaljdiagrammet.

2.2.7 Hjälpelement

Det finns några få element i UML som inte har något verkligt semantiskt värde för modellen, men som hjälper till att klargöra delar av diagrammen. Dessa element är

- Textrader
- Anteckningar och ankare
- Rutor

Textrader är användbara för att lägga till kort textinformation i ett diagram. Det är fristående text, och har ingen betydelse i själva modellen.

Anteckningar är användbara för att lägga till mer detaljerad information om ett objekt eller en särskild situation. De har den stora fördelen att anteckningar kan ankras vid UML-element för att visa att anteckningen 'hör till' ett särskilt objekt eller situation.

Rutor är fristående rektanglar som kan användas för att gruppera objekt tillsammans, för att göra diagram mer läsbara. De har ingen logisk mening i modellen.

2.2.8 Komponentdiagram

Komponentdiagram visar programkomponenter (antingen komponentteknologier som Kparts, CORBA-komponenter eller Java Beans eller bara delar av systemet som är klart urskiljbara) och artefakterna de består av, som källkodsfiler, programbibliotek eller relationsdatabastabeller.

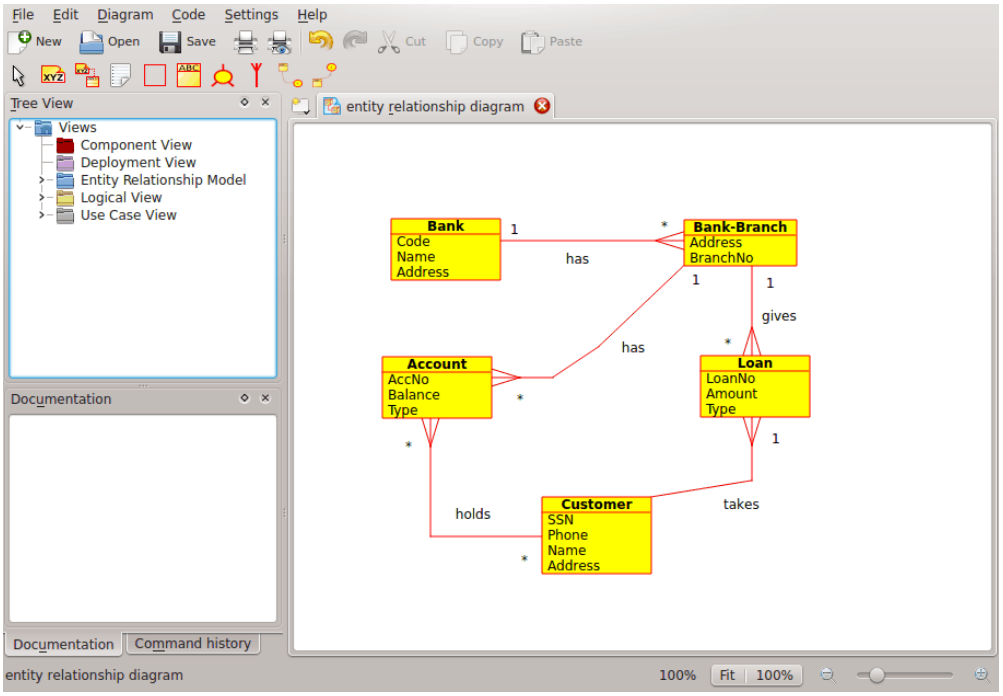
Komponenter kan ha gränssnitt (dvs. abstrakta klasser med operationer) som tillåter association mellan komponenter.

2.2.9 Utplaceringsdiagram

Utplaceringsdiagram visar komponentinstanserna vid körning och deras associationer. De omfattar noder, som är fysiska resurser, typiskt en enskild dator. De visar också gränssnitt och objekt (klassinstanser).

2.2.10 Objektsambandsdiagram

Objektsambandsdiagram visar begreppsmässig konstruktion av databasprogram. De avbildar de olika objekten (koncepten) i informationssystemet och befintliga samband och begränsningar mellan dem. En utökning av objektsambandsdiagram som kallas 'utökade objektsambandsdiagram' eller 'förbättade objektsambandsdiagram' används för att införliva objektorienterade konstruktionstekniker i diagrammen.



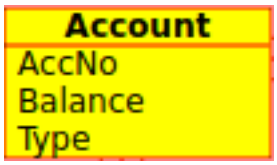
Umbrello UML Modeller som visar ett objektsambandsdiagram

2.2.10.1 Objekt

Ett *objekt* är vilket koncept i verkligheten som helst med en oberoende existens. Det kan vara ett objekt som existerar fysiskt (t.ex. dator, robot), eller det kan vara ett objekt som existerar som koncept (t.ex. universitetskurs). Varje objekt har en uppsättning egenskaper, som beskriver objektet.

Observera: Det finns ingen standardnotation för att avbilda objektsambandsdiagram. Olika skrifter om ämnet använder olika notation. Koncepten och notationen för utökade objektsambandsdiagram i Umbrello kommer från följande bok: *Elmasri R. and Navathe S. (2004), Fundamentals of Database Systems, 4:e utgåvan, Addison Wesley*.

I objektsambandsdiagram representeras objekt av rektanglar, med objektets namn längst upp, och kan också visa objektets attribut i ett annat 'fack' inne i rektangeln.



Visuell representation av ett objekt i ett objektsambandsdiagram

2.2.10.1.1 Objektattribut

I objektsambandsdiagram visas objektattribut med sina namn i ett annat fack hos objektet de tillhör.

2.2.10.1.2 Begränsningar

Begränsningar i objektsambandsdiagram anger restriktioner hos data i informationsschemat.

Det finns fyra typer av begränsningar som stöds i Umbrello:

- *Primär nyckel:* Egenskaperna som deklareras som *primär nyckel* är unika för objektet. Det kan bara finnas en primär nyckel i ett objekt, och ingen av de ingående egenskaperna kan vara tom.
- *Unik nyckel:* Uppsättningen egenskaper som deklareras som *unika* är unika för objektet. Det kan finnas många unika begränsningar för ett objekt. Dess ingående egenskaper kan vara tomma. Unika nycklar och primära nycklar identifierar en rad i en tabell (objekt) på ett unikt sätt.
- *Främmande nyckel:* En främmande nyckel är en referensbegränsning mellan två tabeller. Den främmande nyckeln identifierar en kolumn eller en uppsättning kolumner i en (den refererade) tabell. Kolumnerna i den refererade tabellen måste utgöra en primär nyckel eller en unik nyckel.
- *Kontrollbegränsning:* En kontrollbegränsning (också känd som tabellkontrollbegränsning) är ett villkor som definierar giltig data när ett objekt i en tabell läggs till eller uppdateras i en relationsdatabas. En kontrollbegränsning tilldelas till varje rad i tabellen. Begränsningen måste vara ett predikat. Det kan hänvisa till en eller flera kolumner i tabellen.

  Exempel: pris >= 0

2.2.11 Koncept hos utökade objektsambandsdiagram

2.2.11.1 Specialisering

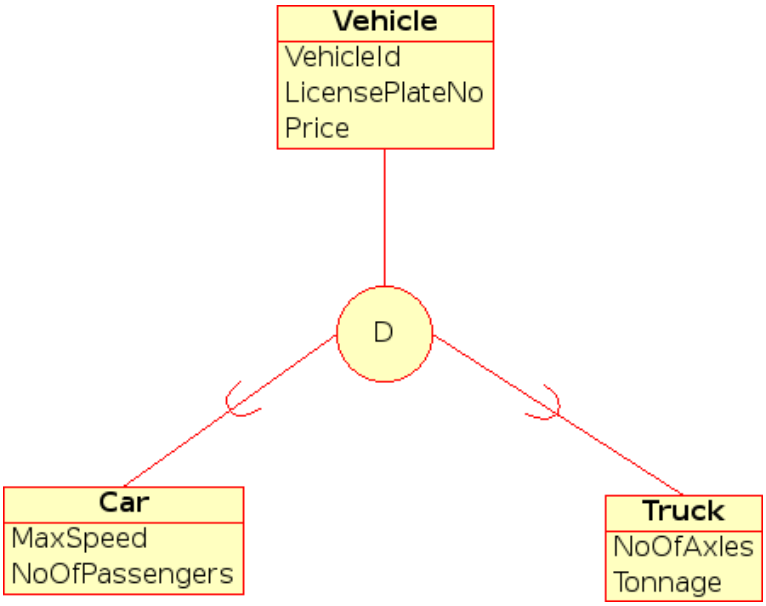
Specialisering är ett sätt att skapa nya objekt med användning av objekt som redan har definierats. De nya objekten, som kallas härledda objekt, tar över (eller ärver) egenskaperna hos de

befintliga objekten, som kallas basobjekt. Detta har avsikten att hjälpa till att använda befintlig data med inga eller små förändringar.

I Umbrello kan man definiera separerad och överlappande specialisering.

2.2.11.1.1 Separerad specialisering

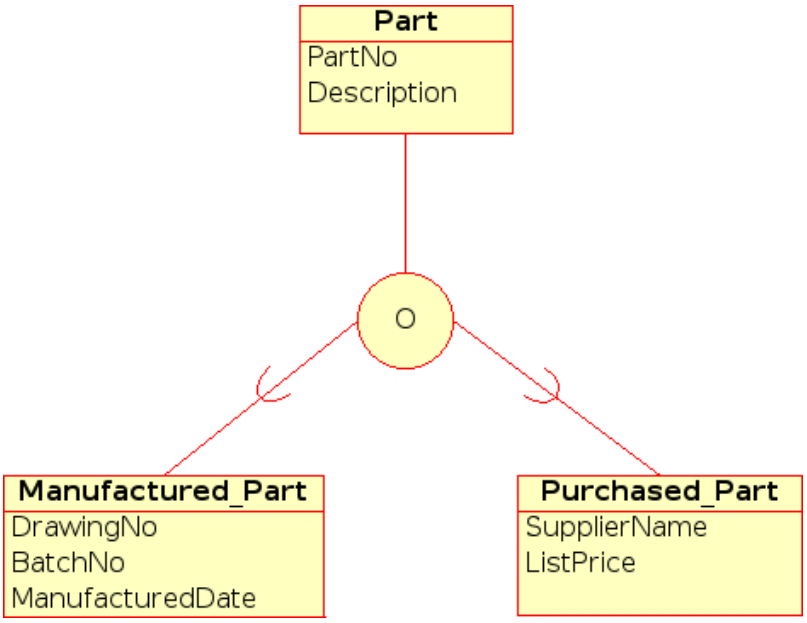
Separerad specialisering anger att specialiseringens delklasser måste vara separerade. Det betyder att ett objekt som mest kan vara medlem av ett av objekten härledda från specialiseringen.



Visuell representation av separerad specialisering i utökade objektsambandsdiagram

2.2.11.1.2 Överlappande specialisering

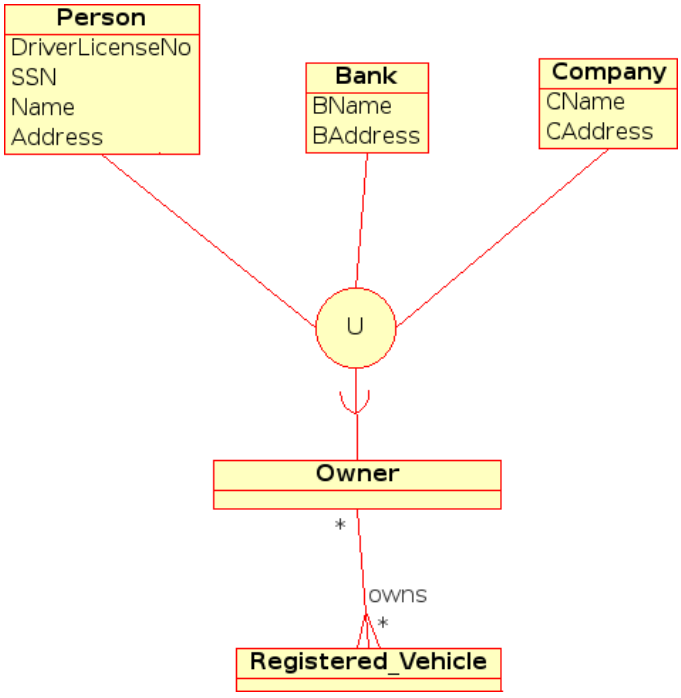
När härledda objekt inte är begränsade att vara separerade, sägs deras objektmängd befinna sig i överlappande specialisering. Det betyder att samma verkliga objekt kan vara medlem i mer än ett av objekten härledda från specialiseringen.



Visuell representation av överlappande specialisering i utökade objektsambandsdiagram

2.2.11.1.3 Kategori

Ett härlett objekt sägs vara en *kategori* när det representerar en objektsamling som är en delmängd av unionen av de skilda objekttyperna. En kategori modelleras när behovet av ett enda superklass/delklass-förhållande med mer än en superklass, där superklasserna representerar olika objekttyper (i likhet med multipla arv i objektorienterad programmering).



Visuell representation av en kategori i utökade objektsambandsdiagram

Kapitel 3

Att arbeta med Umbrello UML Modeller

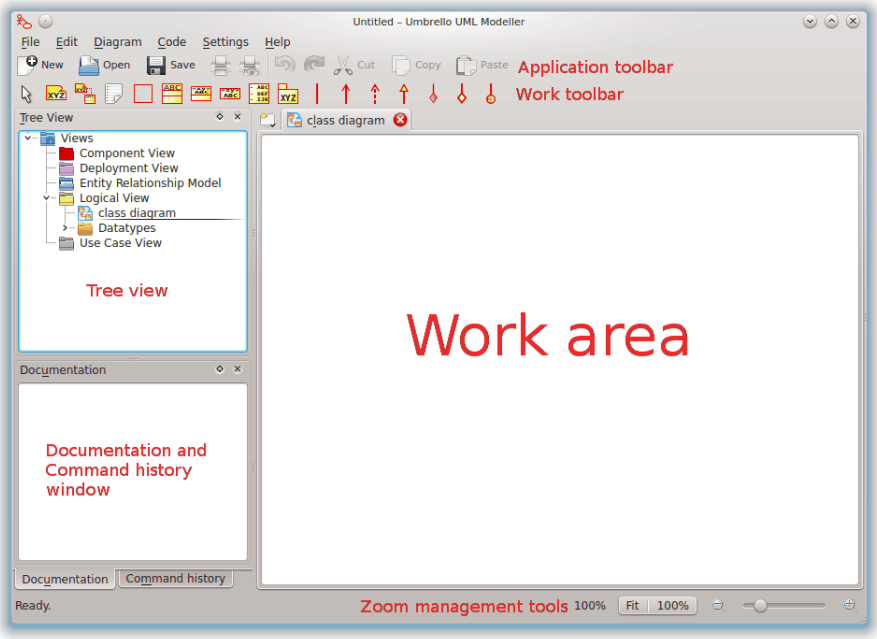
Det här kapitlet ger en introduktion till Umbrello UML Modellers användargränssnitt och berättar allt du måste veta för att börja med modellering. Alla åtgärder i Umbrello UML Modeller är tillgängliga via menyer och verktygsrader, men Umbrello UML Modeller använder också i stor utsträckning sammanhangsberoende menyer som visas med höger musknapp. Du kan högerklicka på nästan alla element på Umbrello UML Modellers arbetsyta eller i trädvyn för att få en meny med de mest användbara funktionerna som kan tillämpas på just det särskilda elementet som du arbetar med. Vissa användare tycker att detta är lite förvirrande i början (eftersom de är mer vana att arbeta med menyn eller verktygsrader), men när man väl har vant sig att högerklicka, snabbar det upp arbetet en hel del.

3.1 Användargränssnitt

Umbrello UML Modellers huvudfönster är uppdelat i tre områden som hjälper till att få en överblick över hela systemet och att komma åt de olika diagrammen snabbt, under arbetet med modellen.

Dessa områden kallas:

- Trädvy
- Arbetsyta
- Dokumentations- och kommandohistorikfönstret



Umbrello UML Modellers användargränssnitt

3.1.1 Trädvy

Trädvyn är oftast placerad längst upp till vänster i fönstret, och visar alla diagram, klasser, aktörer och användningsfall som bygger upp modellen. Trädvyn låter dig få en snabb överblick över elementen som modellen består av. Trädvyn ger också ett snabbt sätt att byta mellan de olika diagrammen i modellen, och att infoga element från modellen i det nuvarande diagrammet.

Om du arbetar med en model som har mer än ett fåtal klasser och diagram, kan trädvyn hjälpa dig att klara av det hela genom att organisera modellen i mappar. Du kan skapa mappar genom att välja lämpligt alternativ i den sammanhangsberoende menyn (högerklicka på en av mapparna i trädvyn) och du kan organisera element genom att flytta dem till lämpliga mappar (drag och släpp).

3.1.2 Dokumentations- och kommandohistorikfönstret

Dokumentations- och kommandohistorikfönstret är det lilla fönstret placerat längst ner till vänster i Umbrello UML Modeller, som ger en snabb förhandsgranskning av dokumentationen för objektet som för närvarande är markerat. Dokumentationsfönstret är ganska litet, eftersom det är avsett att ge ett snabbt utdrag ur elementets dokumentation, medan det tar så lite plats som möjligt på skärmen. Om du behöver titta på dokumentationen i mer detalj, kan du alltid öppna elementets egenskaper.

3.1.3 Arbetsyta

Arbetsytan är huvudfönstret i Umbrello UML Modeller, och är platsen där alla verkliga åtgärder sker. Man använder arbetsytan för att redigera och visa diagrammen i en modell. Arbetsytan visar diagrammet som för tillfället är aktivt. För närvarande kan bara ett diagram åt gången visas på arbetsytan.

3.2 Skapa, ladda och spara modeller

Det första du behöver göra, för att börja utföra något användbart med Umbrello UML Modeller, är att skapa en modell att arbeta med. När du startar Umbrello UML Modeller laddar det alltid den senast använda modellen, eller skapar en ny, tom, modell (beroende på alternativ som du ställer in i inställningsdialogrutan). Det gör det möjligt att börja arbeta direkt.

3.2.1 Ny modell

Om du vid något tillfälle behöver skapa en ny modell, kan du göra det genom att välja alternativet **Ny** i menyn **Arkiv**, eller genom att klicka på ikonen **Ny** i programverktygsraden. Om du för ögonblicket arbetar med en modell som har ändrats, frågar Umbrello UML Modeller om dina ändringar ska sparas, innan den nya modellen skapas.

3.2.2 Spara modell

Du kan spara modellen när som helst, genom att välja alternativet **Spara** i menyn **Arkiv**, eller genom att klicka på knappen **Spara** i programverktygsraden. Om du behöver spara modellen med ett annat namn, kan du använda alternativet **Spara som** i menyn **Arkiv**.

Av bekvämlighetsskäl, erbjuder Umbrello UML Modeller också möjligheten att automatiskt spara arbetet efter en viss tidsperiod. Du kan anpassa om du vill aktivera den här funktionen, samt tidsintervallet, i Umbrello UML Modellers **inställningar**.

3.2.3 Ladda modell

Du kan välja alternativet **Öppna** i menyn **Arkiv** för att ladda en befintlig modell, eller klicka på ikonen **Öppna** i programverktygsraden. De senast använda modellerna är också tillgängliga i undermenyn **Öppna senaste** i menyn **Arkiv**, för att snabba upp åtkomst till de oftast använda modellerna.
Umbrello UML Modeller kan bara arbeta med en modell åt gången, så om du ber programmet ladda en modell åt dig, och den nuvarande modellen har ändrats sedan du senast sparade den, frågar Umbrello UML Modeller om ändringarna ska sparas för att förhindra att arbetet går förlorat. Du kan starta två eller flera instanser av Umbrello UML Modeller när som helst. Du kan också kopiera och klistra in mellan instanser.

3.3 Redigera modeller

I Umbrello UML Modeller finns det två grundläggande sätt att redigera elementen i modellen.

- Redigera modellelement direkt via trädvyn
- Redigera modellelement direkt via ett diagram

Med användning av den sammanhangsberoende menyn i trädvyn, kan du lägga till, ta bort, och ändra nästan alla element i modellen. Högerklicka på mapparna i trädvyn för att visa alternativ för att skapa olika sorters diagram, samt - beroende på om mappen är en *Användningsfallsvy* eller en *Logisk vy* - aktörer, användningsfall, klasser etc.

När du väl har lagt till element i modellen, kan du också redigera dem genom användning av deras egenskapsdialogrutor, som du hittar genom att välja alternativet *Egenskaper* i den sammanhangsberoende menyn som visas vid ett *höger*klick på elementen i trädvyn.

Du kan också redigera modellen genom att skapa eller ändra element via diagram. Mer information om hur detta görs, får du i följande avsnitt.

3.4 Lägga till och ta bort diagram

UML-modellen består av en uppsättning UML-element och samband mellan dem. Man kan dock inte se modellen direkt, utan man använder *diagram* för att titta på den.

3.4.1 Skapa diagram

För att skapa ett nytt diagram i modellen, välj helt enkelt diagramtypen du behöver i undermenyn **Ny** från menyn **Diagram**, och ge den ett namn. Diagrammet skapas, och görs aktivt, och du ser det omedelbart i trädvyn.

Kom ihåg att Umbrello UML Modeller i stor utsträckning använder sammanhangsberoende menyer: du kan också högerklicka på en mapp i trädvyn, och välja lämplig diagramtyp i undermenyn **Ny** från den sammanhangsberoende menyn. Observera att du kan bara skapa användningsfallsdiagram i användningsfallsmappar, och att övriga typer av diagram bara kan skapas i mappar för logiska vyer.

3.4.2 Ta bort diagram

Skulle du behöva ta bort ett diagram från modellen, kan du göra det genom att göra det aktivt och välja **Ta bort** i menyn **Diagram**. Du kan också åstadkomma detta genom att välja **Ta bort** i den sammanhangsberoende menyn för diagrammet i trädvyn.

Eftersom att ta bort ett diagram är något allvarligt, som kunde orsaka att arbete går förlorat, om det görs av misstag, ber Umbrello UML Modeller att du bekräftar en borttagningsåtgärd innan diagrammet verkligen tas bort. Så fort ett diagram har tagits bort, och filen har sparats, finns det inget sätt att ångra åtgärden.

3.4.3 Byta namn på diagram

Om du vill byta namn på ett befintligt diagram, kan du lätt göra det genom att välja alternativet Byt namn i den sammanhangsberoende menyn i trädvyn.

Ett annat sätt att byta namn på ett diagram är via dess egenskapsdialogruta, som du erhåller genom att välja Egenskaper från den sammanhangsberoende menyn, eller genom att dubbelklicka på det i trädvyn.

3.5 Redigera diagram

Medan du arbetar med ett diagram, försöker Umbrello UML Modeller leda dig rätt genom att tillämpa några enkla regler om vilka element som är giltiga i olika sorters diagram, samt vilka förhållanden som kan finnas mellan dem. Om du är expert på UML, kommer du förmodligen inte ens märka det, men det är till hjälp för nybörjare för att skapa diagram som följer standarden.

Så fort du har skapat diagrammen är det dags att börja redigera dem. Observera här (den för nybörjare subtila) skillnaden mellan att redigera ett diagram, och att redigera *modellen*. Som du redan känner till, är diagram *vyer* av modellen. Om du till exempel skapar en klass genom att redigera ett klassdiagram, redigerar du i själva verket både diagrammet och modellen. Om du ändrar färg eller andra visningsalternativ för en klass i klassdiagrammet, redigerar du bara diagrammet, men ingenting ändras i modellen.

3.5.1 Infoga element

En av de första sakerna som du gör när du redigerar ett nytt diagram, är att infoga element i det (klasser, aktörer, användningsfall, etc.). Det finns två grundläggande sätt att göra det:

- Dra befintliga element till modellen från trädvyn
- Skapa nya element i modellen, och samtidigt lägga till dem i diagrammet, genom att använda ett av redigeringsverktygen i arbetsverktygsraden.

För att infoga element som redan finns i modellen, dra dem bara från trädvyn och släpp dem där du vill att de ska vara i diagrammet. Du kan alltid flytta omkring element i diagrammet med markeringsverktyget.

Det andra sättet att lägga till element i diagrammet är att använda arbetsverktygsradens redigeringsverktyg (observera att detta också lägger till elementen i modellen).

Arbetsverktygsraden var normalt placerad längst upp i fönstret. Verktygen som är tillgängliga på den här verktygsraden (knapparna du ser på den) ändras beroende på vilket diagram du arbetar med för ögonblicket. Knappen för verktyget som just nu är valt är aktiverad i verktygsraden. Du kan byta till **markeringsverktyget** genom att trycka på **Esc**-tangenten.

När du har valt ett redigeringsverktyg i arbetsverktygsraden (till exempel verktyget för att infoga klasser), ändras muspekaren till ett kors, och du kan infoga element i modellen genom att enkelklicka i diagrammet. Observera att element i UML måste ha ett *unikt namn*. Så om du har en klass i ett diagram som heter 'KlassA', och sedan använder verktyget för att infoga klasser för att infoga en klass i ett annat diagram, kan du inte också ge den nya klassen namnet 'KlassA'. Om det är meningen att de två ska vara olika element, måste du ge dem unika namn. Om du försöker lägga till *samma* element i diagrammet, är inte verktyget för att infoga klasser rätt verktyg för detta. Du ska istället dra och släppa klassen från trädvyn.

3.5.2 Ta bort element

Du kan ta bort vilket element som helst, genom att välja alternativet **Ta bort** i dess sammanhangsberoende meny.

Återigen är det en *stor* skillnad mellan att ta bort ett objekt från diagrammet, och att ta bort ett objekt från modellen. Om du tar bort ett objekt inifrån ett diagram, tar du bara bort det från just det diagrammet: elementet är fortfarande en del av modellen och om det finns andra diagram som använder samma element, råkar de inte ut för någon ändring. Å andra sidan, om du tar bort elementet i trädvyn, tar du i själva verket bort elementet från *modellen*. Eftersom elementet inte längre existerar i modellen, tas det också automatiskt bort från alla diagram det visas i.

3.5.3 Redigera element

Du kan redigera de flesta UML-element i modellen och diagram genom att öppna dess egenskapsdialogruta och välja lämpliga alternativ. För att redigera egenskaperna hos ett objekt, välj **Egenskaper** i dess sammanhangsberoende meny (högerklicka). Varje element har en dialogruta som består av flera sidor där du kan anpassa alternativen som har med det elementet att göra. För vissa element, som aktörer, kan du bara ange ett fåtal alternativ, som objektnamn och dokumentation, medan för andra element, som klasser, kan du redigera dess attribut och operationer, välja vad du vill visa i diagram (hela operationssignaturen eller bara operationsnamn, etc.) och till och med färgerna du vill använda för linjer och ifyllnad av klassens representation i ett diagram.

För UML-element kan du också öppna egenskapsdialogrutan genom att dubbelklicka på det, om du använder markeringsverktyget (pilen).

Observera att du också kan välja alternativet egenskaper i den sammanhangsberoende menyn för elementen i trädvyn. Detta låter dig också redigera egenskaper för diagram, som att ställa in om rutnätet ska visas eller inte.

3.5.4 Redigera klasser

Även om redigering av egenskaper för alla objekt redan har täckts av föregående avsnitt, förtjänar klasser ett särskilt avsnitt, eftersom de är något mer komplicerade, och har fler alternativ än de flesta andra UML-element.

I klassens egenskapsdialogruta kan du ställa in allting, från färgen den använder till operationerna och attributen den har.

3.5.4.1 Allmänna klassinställningar

Sidan med allmänna klassinställningar i egenskapsdialogrutan är självförklarande. Här kan du ändra klassens namn, synlighet, dokumentation, etc. Den här sidan är alltid tillgänglig.

3.5.4.2 Inställningar av klassattribut

På sidan för inställningar av attribut, kan du lägga till, redigera eller ta bort attribut (variabler) för klassen. Du kan flytta attribut upp och ner i listan genom att trycka på piltangenterna längs kanten. Den här sidan är alltid tillgänglig.

3.5.4.3 Inställningar av klassoperationer

På liknande sätt som för inställningar av klassattribut, kan du lägga till, redigera eller ta bort operationer för klassen på sidan för inställningar av klassoperationer. När du lägger till eller redigerar en klassoperation, skriver du in grundläggande data i dialogrutan *Operationsegenskaper*. Om du vill lägga till parametrar till operationerna, måste du klicka på knappen **Ny parameter**, som visar dialogrutan *Parameteregenskaper*. Den här sidan är alltid tillgänglig.

3.5.4.4 Klassmallsinställningar

Den här sidan låter dig lägga till klassmallar som är ospecificerade klasser eller datatyper. I Java 1.5 kommer de att kallas Generics.

3.5.4.5 Sidan för klassassociationer

Sidan **Klassassociationer** visar alla klassens associationer i det nuvarande diagrammet. Ett dubbelklick på en association visar dess egenskaper, och beroende på typ av association, kan du ändra vissa parametrar här som att ställa in mångfald och rollnamn. Om associationen inte tillåter att sådana alternativ ändras, är dialogrutan för associationsegenskaper bara läsbar, och du kan endast ändra dokumentationen som hör ihop med associationen.

Den här sidan är bara tillgänglig om du öppnar klassegenskaperna inne i ett diagram. Om du väljer klassegenskaper från den sammanhangsberoende menyn i trädvyn, är den här sidan inte tillgänglig.

3.5.4.6 Sidan för klassvisning

På sidan **Visningsalternativ**, kan du ställa in vad som ska visas i diagrammet. En klass kan visas som bara en rektangel med klassnamnet i (användbart om du har många klasser i diagrammet, eller för tillfället inte är intresserad av detaljerna för varje klass), eller så fullständiga att paket, stereotyper, attribut och operationer visas med fullständig signatur och synlighet.

Beroende på mängden information som du vill se, kan du välja motsvarande alternativ på sidan. Ändringarna du gör här gäller bara *visningsalternativen* för diagrammet. Det betyder att 'dölja' klassens operationer bara gör att de inte visas i diagrammet, men operationerna är forfarande där som en del av modellen. Det här alternativet är bara tillgängligt om du väljer klassegenskaperna inne i ett diagram. Om du öppnar klassegenskaper från trädvyn, saknas den här sidan, eftersom sådana visningsegenskaper inte är vettiga i detta fall.

3.5.4.7 Sidan för klasstil

På sidan **Komponentstil** kan du anpassa färgerna som du vill ha för linjer och ifyllnad av komponenten. Det här alternativet är naturligtvis bara vettigt för klasser som visas i diagram, och saknas om du öppnar klassens egenskapsdialogruta i trädvyn.

3.5.5 Associationer

Associationer relaterar två UML-objekt med varandra. Normalt definieras associationer mellan två klasser, men vissa typer av associationer kan också finnas mellan användningsfall och aktörer.

För att skapa en association, välj lämpligt verktyg i arbetsverktygsraden (generell association, generalisering, aggregering, etc.), och enkelklicka på det första elementet som ingår i associationen. Enkelklicka sedan på det andra elementet som ingår. Observera att detta är två klick, ett på vart och ett av elementen som ingår i associationen. Det är *inte* att dra från ett objekt till ett annat.

Om du försöker använda associationer på ett sätt som inte tillåts av UML-specifikationen, vägrar Umbrello skapa associationen och du får ett felmeddelande. Det skulle inträffa, om till exempel en generalisering finns från klass A till klass B, och du därefter försöker skapa en ny generalisering från klass B till klass A.

Ett högerklick på en association visar en sammanhangsberoende meny med åtgärder som du kan genomföra med den. Om du behöver ta bort en association, välj helt enkelt alternativet **Ta bort** i den sammanhangsberoende menyn. Du kan också välja alternativet **Egenskaper**, och beroende på associationens typ, redigera attribut som roller och mångfald.

3.5.5.1 Ankringspunkter

Associationer ritas normalt som en rak linje som förbinder de två objekten i diagrammet.

Du kan lägga till ankringspunkter för att forma en association genom att dubbelklicka någonstans längs associationslinjen. Då infogas en ankringspunkt (som visas som en blå punkt där associationslinjen är markerad), som du kan flytta omkring för att ge associationen sin form.

Om du behöver ta bort en ankringspunkt, dubbelklicka på den igen för att ta bort den.

Observera att det enda sättet att redigera en associations egenskaper är via den sammanhangsberoende menyn. Om du försöker att dubbelklicka på den som med andra UML-objekt, infogas bara en ankringspunkt.

3.5.6 Anteckningar, text och rutor

Anteckningar, textrader och rutor är element som kan finnas i alla sorters diagram, och har inget verkligt semantiskt värde, men är mycket hjälpsamma för att lägga till extra kommentarer eller förklaringar, som kan göra diagrammet lättare att förstå.

För att lägga till en anteckning eller textrad, välj motsvarande verktyg i arbetsverktygsraden, och enkelklicka på diagrammet där du vill placera kommentaren. Du kan redigera texten genom att öppna elementet via dess sammanhangsberoende meny, eller för anteckningar, också genom att dubbelklicka på dem.

3.5.6.1 Ankare

Ankare används för att länka ihop anteckningar och ett annat UML-element. Normalt använder du till exempel en anteckning för att förklara eller ge en kommentar om en klass eller en viss association, och i så fall kan du använda ankaret för att klargöra att anteckningen 'hör till' just det elementet.

Använd ankarverktyget i arbetsverktygsraden, för att lägga till ett ankare mellan en anteckning och ett annat UML-element. Först måste du klicka på anteckningen, och sedan klicka på UML-elementet som du vill att anteckningen ska länkas till.

Kapitel 4

Kodimport och kodgenerering

Umbrello UML Modeller är ett UML-modelleringsverktyg, och som sådant är dess huvudsakliga syfte att hjälpa dig med *analys och konstruktion* av system. För att åstadkomma övergången från konstruktion till *implementering*, tillåter Umbrello UML Modeller dock att generera källkod i olika programspråk för att komma igång. Om du dessutom vill börja använda UML i ett projekt som redan har startat, kan Umbrello UML Modeller hjälpa till att skapa en modell av systemet från källkoden genom att analysera den och importera klasserna som hittas i den.

4.1 Kodgenerering

Umbrello UML Modeller kan generera källkod för diverse programspråk, baserad på din UML-modell för att hjälpa dig komma igång med implementeringen av projektet. Koden som skapas består av klassdeklarationer, med metoder och attribut, så att du kan 'fylla i tomrummen' genom att tillhandahålla funktionerna i klassernas operationer.

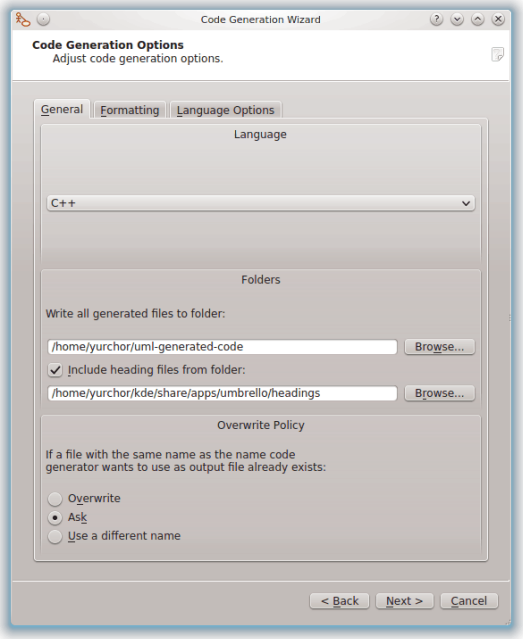
Umbrello UML Modeller 2 levereras med kodgenereringsstöd för ActionScript, Ada, C++, C#, D, IDL, Java™, Javascript, MySQL, Pascal, Perl, PHP, PHP5, PostgreSQL, Python, Ruby, Tcl, Vala och XMLSchema.

4.1.1 Generera kod

För att generera kod med Umbrello UML Modeller, måste du först skapa eller ladda en modell som innehåller minst en klass. När du är klar att börja skriva lite kod, välj då alternativet **Kodgenereringsguide** i menyn **Kod**, för att starta guiden som leder dig igenom kodgenereringsprocessen.

Det första steget är att välja klasser, som du vill skapa källkod för. Normalt väljs alla klasser i modellen, och du kan ta bort de som du inte vill generera kod för, genom att flytta dem till listan på vänster sida.

Nästa steg i guiden låter dig ändra parametrar som kodgeneratorn använder när den skriver ut koden. Följande alternativ är tillgängliga:



Alternativ för kodgenereringen i Umbrello UML Modeller

4.1.1.1 Kodgenereringsalternativ

4.1.1.1.1 Kommentarnivå

Alternativet **Skriv dokumenteringskommentarer även om tomma** instruerar kodgeneratorn att skriva ut kommentarar med stilen /** blaha */, även om kommentarblocken är tomma. Om du lagt till dokumentation i klasser, metoder eller attribut i modellen, skriver kodgeneratorn ut kommentarerna som Doxygen-dokumentation, oberoende av vad du anger här, men om du väljer det här alternativet, skriver Umbrello UML Modeller ut kommentarblock för alla klasser, metoder och attribut även om det inte finns någon dokumentation i modellen, då detta är fallet bör du dokumentera klasserna senare direkt i källkoden.

Skriv kommentarer för sektioner även om sektionen är tom: Umbrello UML Modeller skriver kommentarer i källkoden för att avdela de olika sektionerna i en klass. Till exempel 'Public methods' eller 'Attributes' innan motsvarande sektioner. Om du väljer det här alternativet, så skriver Umbrello UML Modeller kommentarer för alla sektioner i klassen, även om sektionen är tom. Det skulle till exempel skriva en kommentar som lyder 'Protected methods', även om det inte finns några sådana i klassen.

4.1.1.1.2 Kataloger

Skriv alla filer som skapas till katalog: Här ska du välja katalogen där du vill att Umbrello UML Modeller ska lägga källkoden som skapas.

Alternativet **Infoga huvudfiler från katalog**, låter dig infoga ett huvud i början av varje fil som genereras. Huvudfiler kan innehålla upphovsrätts- eller licensinformation, och kan innehålla variabler som utvärderas när genereringen sker. Du kan ta en titt på mallar för huvudfiler som levereras med Umbrello UML Modeller, för att se hur man använder variablerna för att ersätta ditt namn eller dagens datum när genereringen sker.

4.1.1.1.3 Överskrivningspolicy

Det här alternativet talar om för Umbrello UML Modeller vad som ska ske om filen som ska skapas redan finns i destinationskatalogen. Umbrello UML Modeller 1.1 *kan inte ändra befintliga källkodsfiler*, så du måste välja mellan att skriva över den befintliga filen, hoppa över att skapa just den filen, eller låta Umbrello UML Modeller välja ett annat filnamn. Om du väljer alternativet att använda ett annat filnamn, lägger Umbrello UML Modeller till ett suffix till filnamnet.

4.1.1.1.4 Språk

Umbrello UML Modeller genererar normalt kod för språket som du har valt som aktivt språk, men du har möjlighet att ändra detta till ett annat språk med kodgenereringsguiden.

4.1.1.2 Generering med kodgenereringsguiden

Det tredje och sista steget i guiden visar status för kodgenereringsprocessen. Du behöver bara klicka på knappen Generera för att få klasserna utskrivna åt dig.

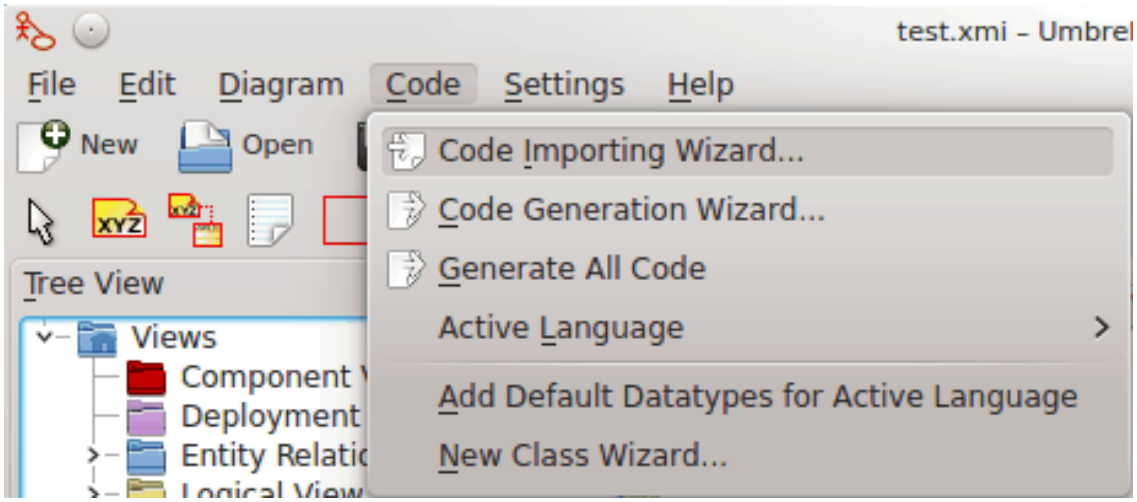
Observera att de alternativ som du väljer med kodgenereringsguiden bara gäller för aktuell generering. Nästa gång du kör guiden, måste du välja alla alternativ igen (din huvudkatalog, överskrivningspolicy, och så vidare). Du kan ställa in standardvärden som används av Umbrello UML Modeller i avdelningen **Kodgenerering** i Umbrello UML Modellers inställningar, tillgängliga via **Inställningar** → **Anpassa Umbrello UML Modeller...**.

Om du har ställt in kodgenereringsalternativ till riktiga inställningar, och vill skapa lite kod direkt utan att gå via guiden, kan du välja **Generera all kod** i menyn **Kod**. Det genererar kod för alla klasser i modellen med nuvarande inställningar (inklusive utdatakatalog och överskrivningspolicy, så använd det med försiktighet).

4.2 Kodimport

Umbrello UML Modeller kan importera källkod från befintliga projekt för att hjälpa dig bygga modeller av system. Umbrello UML Modeller 2 stöder ActionScript, Ada, C++, C#, D, IDL, Java™, Javascript, MySQL, Pascal, PHP och Vala källkod.

För att importera klasser till modellen, välj alternativet **Kodimportguide...** i menyn **Kod**. Välj filerna som innehåller klassdeklarationer i fildialogrutan och tryck på **Nästa** och därefter **Starta import** och **Slutför**. Klasserna importeras, och du hittar dem som en del av modellen i trädvyn. Observera att Umbrello UML Modeller inte skapar något sorts diagram för att visa klasserna, de importeras bara till modellen så att du senare kan använda dem i valfritt diagram.



Meny för att importera källkod till Umbrello UML Modeller

Kapitel 5

Övriga funktioner

5.1 Övriga funktioner i Umbrello UML Modeller

Det här kapitlet förklarar kortfattat några andra funktioner som Umbrello UML Modeller erbjuder.

5.1.1 Kopiera objekt som PNG-bilder

Förutom att erbjuda de normala funktionerna för att kopiera, klippa ut och klistra in, som man kan förvänta sig för att kopiera objekt mellan olika diagram, kan Umbrello UML Modeller kopiera objekt som PNG-bilder, så att man kan infoga dem i vilket annat typ av dokument som helst. Man behöver inte göra något särskilt för att använda den här funktionen, markera bara ett objekt i ett diagram (klass, aktör, etc.) och kopiera det (**Ctrl-C**, eller använd menyn), öppna sedan ett dokument med Calligra Words (eller något annat program där bilder kan klistras in) och välj **Klistra in**. Detta är en utmärkt funktion för att exportera delar av diagram som enkla bilder.

5.1.2 Exportera till en bild

Man kan också exportera ett fullständigt diagram som en bild. Det enda man måste göra är att välja diagrammet som ska exporteras, och därefter alternativet **Exportera som bild...** i menyn **Diagram**.

Man kan exportera flera diagram samtidigt genom att använda alternativet **Exportera diagram som bilder** i menyn **Arkiv**. Med den kan också bildernas upplösning ställas in, så att bilderna inte blir så suddiga.

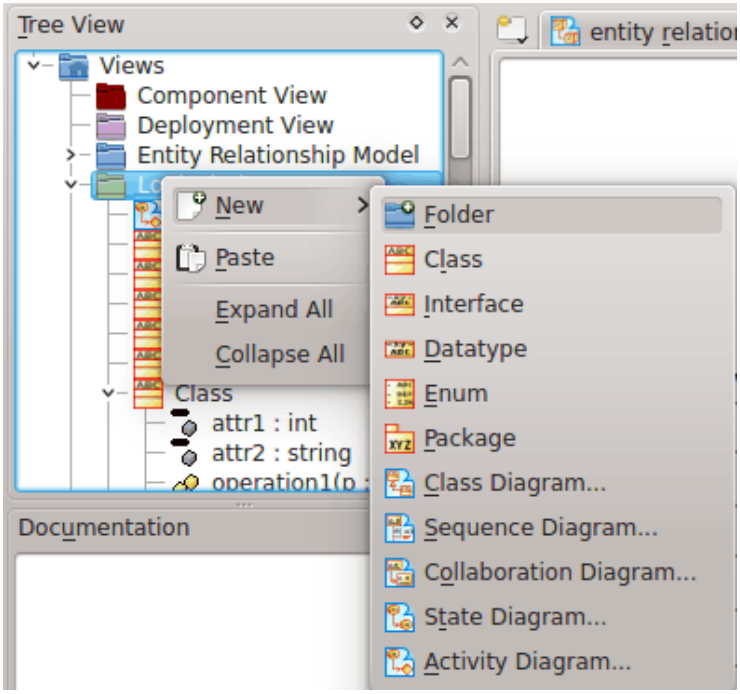
5.1.3 Skriva ut

Umbrello UML Modeller tillåter att enskilda diagram skrivs ut. Tryck på knappen **Skriv ut** i programverktygsraden eller välj alternativet **Skriv ut** i menyn **Arkiv**, så visas KDE:s standardutskriftsdialogruta där diagram kan skrivas ut.

5.1.4 Logiska mappar

För att organisera en modell på ett bättre sätt, särskilt för större projekt, kan man skapa logiska mappar i trädvyn. Välj bara alternativet **Nytt** → **Mapp** i den sammanhangsberoende menyn i

standardmappen under trädvyn, för att skapa dem. Mappar kan finnas i varandra, och man kan flytta omkring objekt genom att dra dem från en mapp till och släppa dem i en annan.

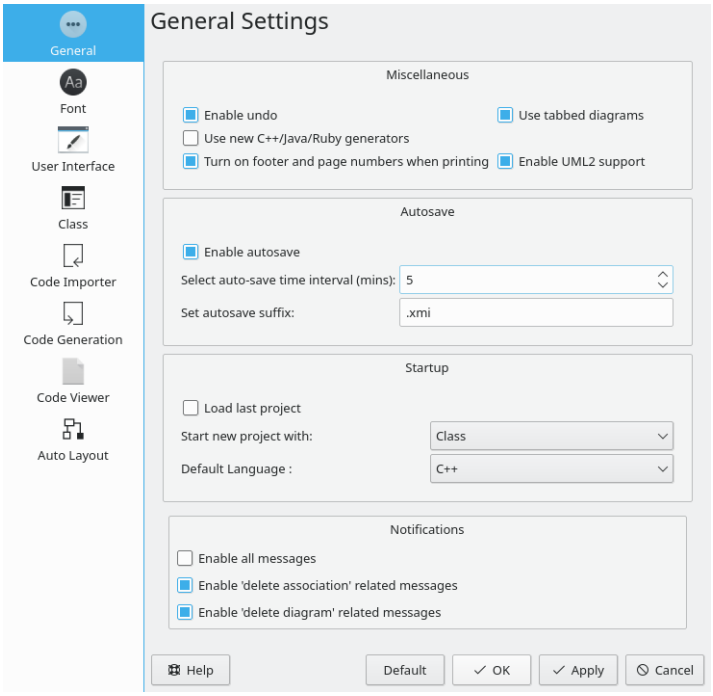


Organisera en modell med logiska mappar i Umbrello UML Modeller

Kapitel 6

Inställningar

6.1 Allmänna inställningar



Alternativ i de allmänna inställningarna i Umbrello UML Modeller

6.1.1 Diverse

- Alternativet **Aktivera ångra** tillåter att en tidigare åtgärd ångras.
- **Använd nya C++, Java och Ruby kodgeneratorer** låter användaren antingen välja de gamla eller de nya kodgeneratorerna
- När **Använd sidfot och sidnummer vid utskrift** är markerad, skrivs diagraminformation och sidnummer ut för diagrammet som skrivs ut.
- **Använd flikdiagram** ger möjlighet att ha flera diagramfönster under flikar öppna samtidigt.

6.1.2 Spara automatiskt

- **Aktivera spara automatiskt** ger valet att spara filen automatiskt.
- **Välj intervall för spara automatiskt (minuter):** tillåter att tiden innan filen sparas automatiskt ställs in.
- **Ange ändelse för spara automatiskt** har standardvärdet .xmi, men tillåter att en annan filändelse ställs in.

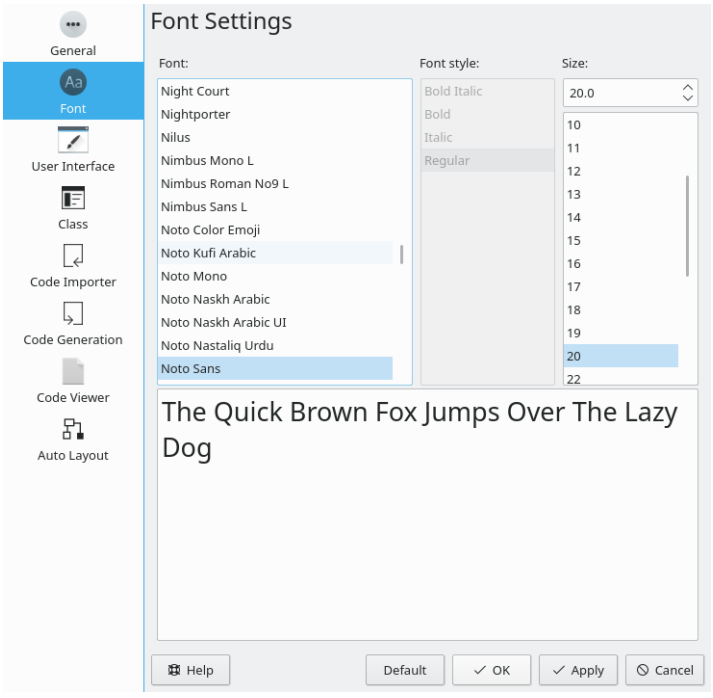
6.1.3 Start

- **Ladda senaste projekt** om markerad, laddar alltid det senaste arbetsprojektet när programmet startas.
- **Starta nytt projekt med:** ger ett val av vilken UML-diagramtyp som ett nytt projekt startas med.
- **Standardspråk:** är en inställning för förvalt programspråk som används.

6.1.4 Underrättelser

- **Aktivera alla meddelanden** är ett alternativ för att antingen se alla underrättelser eller en reducerad mängd av underrättelser.
- **Aktivera meddelanden relaterade till 'ta bort association'** säkerställer att du tar emot alla meddelanden av den här typen om markerat.
- **Aktivera meddelanden relaterade till 'ta bort diagram'** aktiverar alla meddelanden av den här typen om markerat.

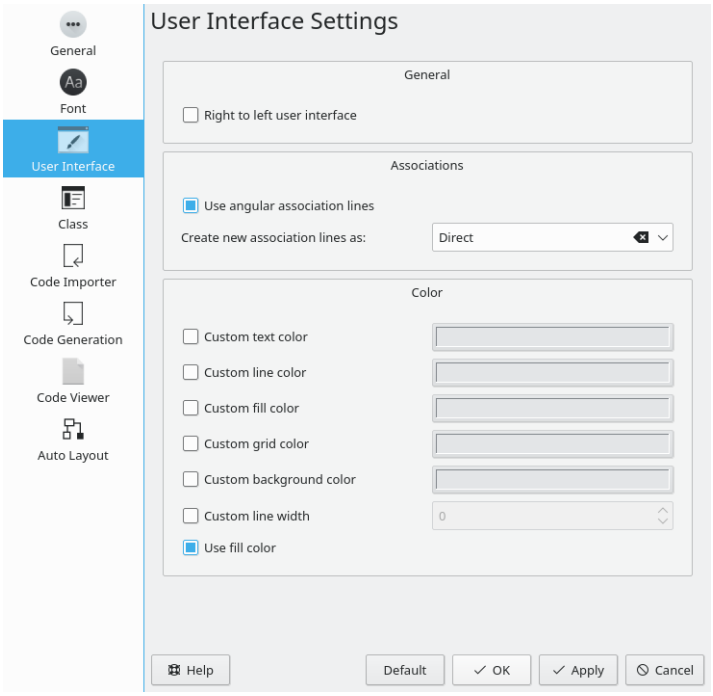
6.2 Teckensnittsinställningar



Alternativ i inställningarna av diagramteckensnitt i Umbrello UML Modeller

Teckensnittsinställningarna ställer in textkarakteristiken i diagrammen. Teckenstil och storlek är de enda valbara alternativen.

6.3 Inställning av användargränssnitt



Alternativ i inställningarna av användargränssnitt i Umbrello UML Modeller

6.3.1 Allmänt

Användargränssnitt från höger till vänster anpassar gränssnittet för höger till vänster språk.

6.3.2 Associationer

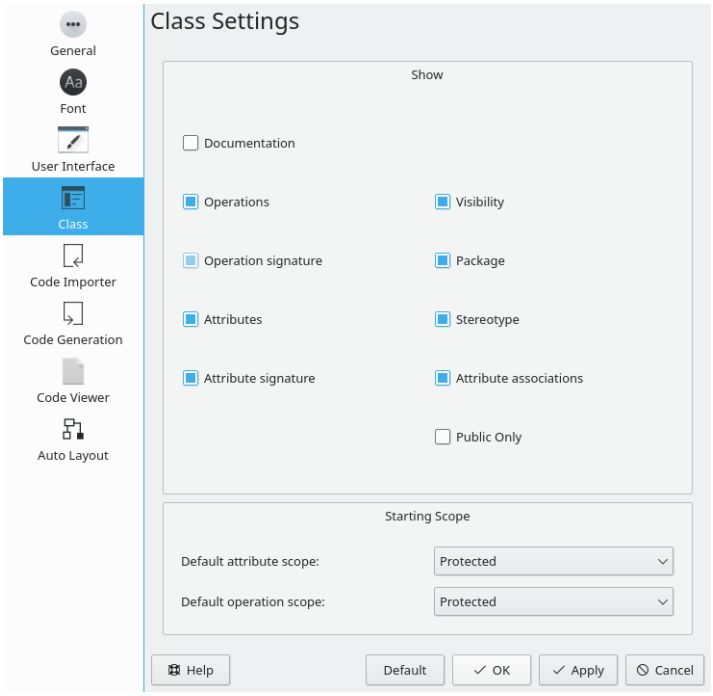
Använd vinklade associationslinjer tillåter att associationslinjer varierar med godtycklig vinkel.

Skapa nya associationslinjer som: ger möjlighet att ändra linjestil för associationer.

6.3.3 Färg

Färgvalet ger flera alternativ för att ändra text-, linje-, ifyllnads-, rutnäts-, och bakgrundsfärger samt linjebredden.

6.4 Klassinställningar



Alternativ i klassinställningarna i Umbrello UML Modeller

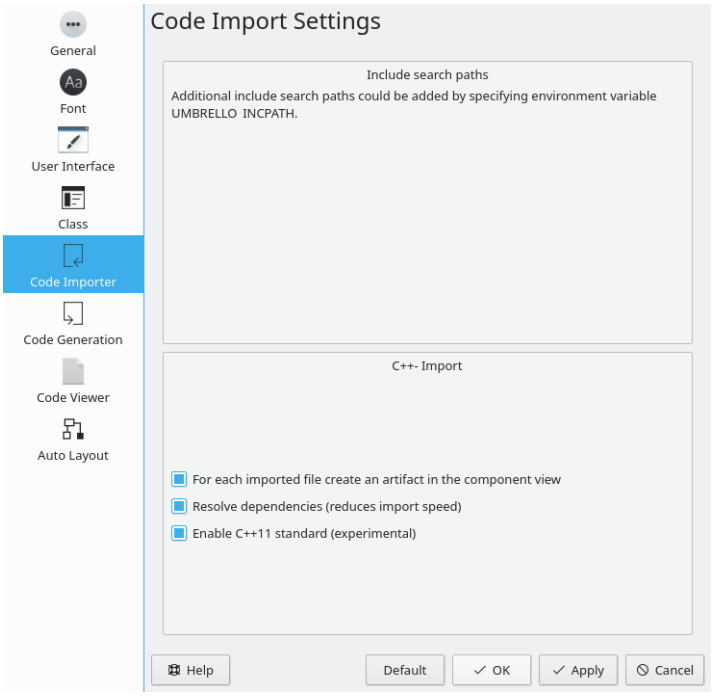
6.4.1 Visa

Sektionen Visa har flertalet inställningar som bestämmer vilken klasskarakteristik som visas i klassdiagrammen.

6.4.2 Startområde

Val av standardinställningar för attribut och operationer, öppna, privata eller skyddade.

6.5 Inställningar av kodimport



Alternativ i inställningarna av kodimport i Umbrello UML Modeller

6.5.1 Inkludera sökvägar

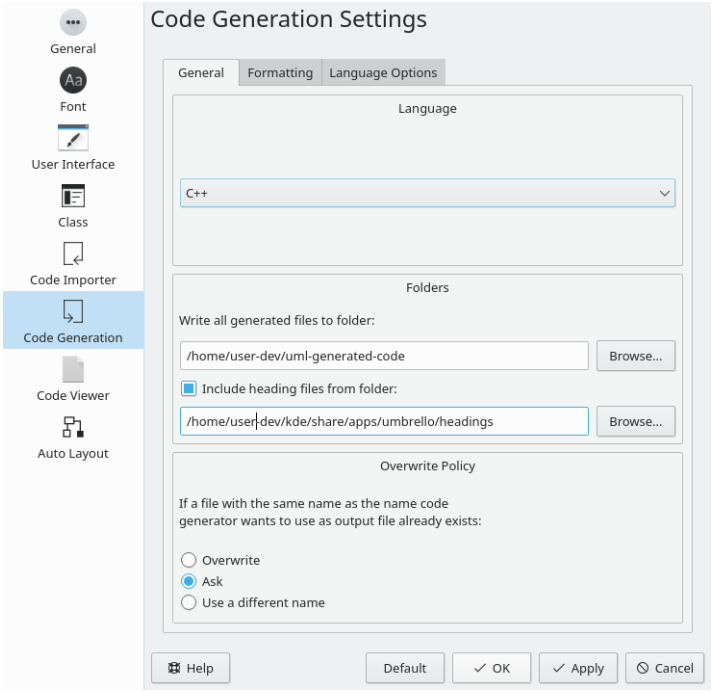
En allmän rekommendation ges för att förbättra sökning genom att inkludera UMBRELLO INCPATH som en miljövariabel.

6.5.2 Import av C++

- **Skapa en artefakt i komponentvyn för varje importerad fil** Artefakten som skapas kan sedan dras till klassdiagramvyn där beroenden enkelt kan observeras tillsammans med attributen och funktionerna i varje fil.
- **Lös upp beroenden (minskar importhastighet)** säkerställer att alla filberoenden löses upp vilket sedan dyker upp i klassberoenden i klassdiagrammen.
- **Aktivera C++ 11-standard (experimentellt** En experimentell funktion för att överensstämma med C++ 11, inaktivera om det inte behövs.

6.6 Inställningar av kodgenerering

6.6.1 Allmän flik för inställningar av kodgenerering



Alternativ i de allmänna inställningarna av kodgenerering i Umbrello UML Modeller

Umbrello UML-modellering kan generera källkod för diverse programspråk, baserad på din UML-modell för att hjälpa dig komma igång med implementeringen av projektet. Koden som skapas består av klassdeklarationer, med metoder och attribut, så att du kan ″fylla i tomrummen″ genom att tillhandahålla funktionerna i klassernas operationer.

6.6.1.1 Språk

Välj programspråk att använda för projekt. Valen som erbjuds är ActionScript, Ada, C++, C#, D, IDL, Java, JavaScript, MYSQL, Pascal, Perl, PHP, PHP5, PastgreSQL, Python, Ruby, SQL, Tcl, Vala och XMLSchema

6.6.1.2 Kataloger

Skriv alla filer som skapas till katalog: har ett redigerbart fält för den önskade sökvägen för genererade filer eller en valfri bläddringsknapp för att välja sökvägen.

Infoga huvudfiler från katalog: Om markerad, låt användaren ange en sökväg i ett redigerbart fält eller välja den med en bläddringsknapp.

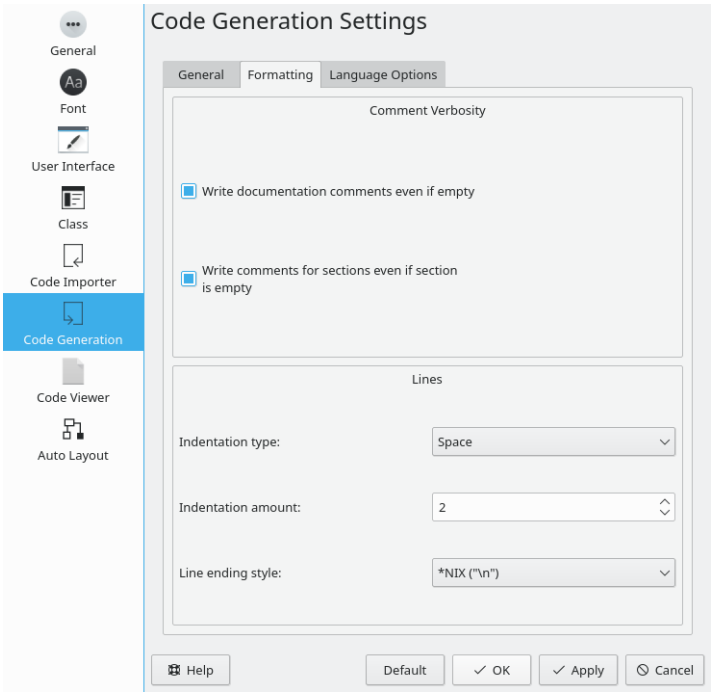
6.6.1.3 Överskrivningspolicy

När koden genereras i den angivna katalogen, bestämmer den här inställningen vad som händer när en fil med samma namn påträffas.

- **Skriv över** filen utan en varning eller alternativ.

- **Fråga** om filen ska skrivas över eller om namnet ska ändras.
- **Använd ett annat namn** när en fil redan finns genom att byta namn på det med ett suffix.

6.6.2 Flik för inställningar av kodgenereringsformatering



Alternativ i inställningarna av kodgenereringsformatering i Umbrello UML Modeller

6.6.2.1 Kommentarnivå

Skriv dokumenteringskommentarer även om tomma Genererar kommentarer för klasser och funktioner även om de är tomma.

Skriv kommentarer för sektioner även om sektionen är tom Skriver kommentarer för privata, skyddade och öppna sektionerna även om de är tomma.

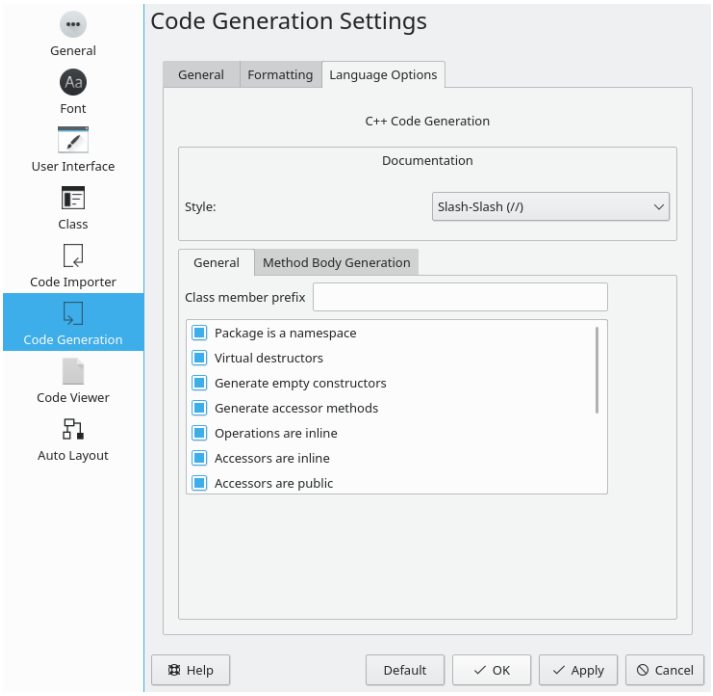
6.6.2.2 Rader

Intenteringstyp: erbjuder ett val mellan ingen indentering, tabulator eller mellanslag.

Indenteringssteg: låter användaren ange antal mellanslag för indenteringsvalen tabulator eller mellanslag.

Radslutstil: är ett val mellan radslutstilarna på *NIX, Windows och Mac.

6.6.3 Språkalternativ



Alternativ i inställningarna för kodgenereringsspråk i Umbrello UML Modeller

Sidan ändras för varje programspråk som väljes under fliken Allmänt. För närvarande är de enda alternativen som är tillgängliga för språket C++.

6.6.3.1 C++ kodgenerering

6.6.3.1.1 Dokumentation

Stil: ger ett val att antingen använda ″/** */″ eller ″//″ som dokumentationsstil

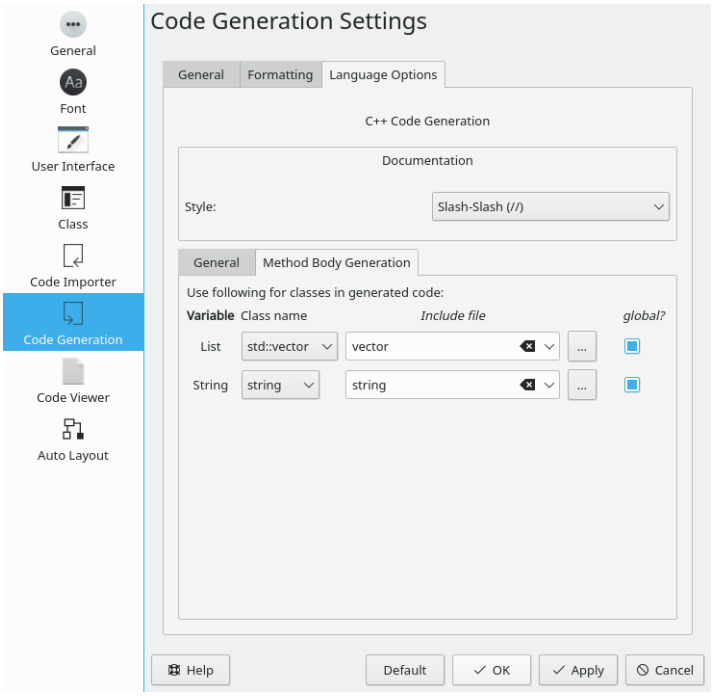
6.6.3.1.2 Allmänt

Under fliken **Allmänt** under fliken **Språkalternativ**, listas flera kodgenereringsalternativ.

- **Prefix för klassmedlem**
  Ett alternativ som tillåter ett prefix bestämt av användaren, för att läggas till i klassmedlemmar när kod genereras.
- **Paket är en namnrymd**
  Namnrymder tillhandahåller en metod för att förhindra namnkonflikter i stora projekt. Symboler deklarerade inne i ett namnrymdsblock är placerade i en namngiven omgivning som förhindrar att de misstas för symboler med identiska namn i andra omgivningar.
- **Virtuella destruktorer**
  Även om destruktorer inte ärvs, om en basklass deklarerar sin destruktor virtuell, överskrider de härledda destruktorerna alltid den. Det gör det möjligt att ta bort dynamiskt tilldelade objekten av polymorfiska typer via pekare till basen.
- **Skapa tomma konstruktormetoder**
  Det här skapar konstruktorer som har tomma klammerparenteser.

- **Skapa åtkomstmetoder**
  Skapar genereringsmetoder för att komma åt datatyper.
- **Operationer är infogade**
  Generera metoderna infogade, men kompilatorer är fria att välja att inte infoga metoden.
- **Åtkomstobjekt är infogade**
  Metoder som kommer åt klassens data genereras infogade, men kompilatorer är fria att välja att inte infoga metoden.
- **Åtkomstobjekt är öppna**
  Metoder som genereras som öppna kommer att vara tillgängliga för alla instansieringar av klassen.
- **Skapa hämtningsfunktioner med prefixet 'get'**
  Det här lägger till prefixet "get" för metoderna som hämtar eller returnerar klassdata.
- **Ta bort prefix '[a-zA-Z]_' från åtkomstmetodnamn**
  Om ett prefix matades in i **Prefix för klassmedlem**, tas det bort av det här.
- **Åtkomstmetoder börjar med stora bokstäver**
  Det här gör metodnamnets första bokstav stor.
- **Använd '\' som dokumentationstagg istället för @**
  Ett val av tagg att använda när en metods parametrar dokumenteras.

6.6.3.1.3 Skapa metodimplementering



Alternativ i inställningarna för metodimplementering i kodgenereringsspråk i Umbrello UML Modeller

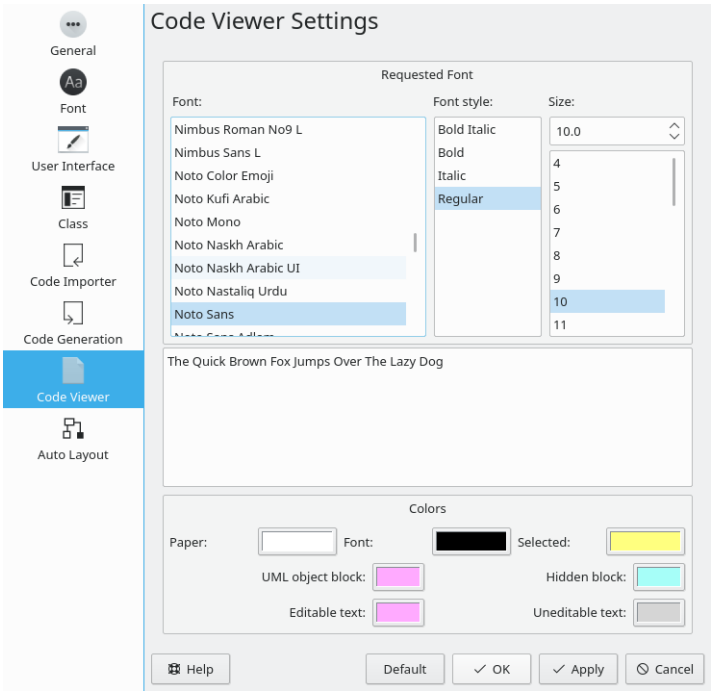
Lista

Har alternativen QPtrList, vector och std::vector för listtypen. En redigerbar eller valbart fält följer för att ange deklarationsfilen tillsammans med en bläddringsknapp för att hitta och välja deklarationsfilen. Det finns också ett alternativ för att göra listan global.

Sträng

Alternativen string eller QString för strängtypen. En redigerbar eller valbart fält följer för att ange deklarationsfilen tillsammans med en bläddringsknapp för att hitta och välja deklarationsfilen. Det finns också ett alternativ för att göra strängen global.

6.7 Inställningar av kodvisning

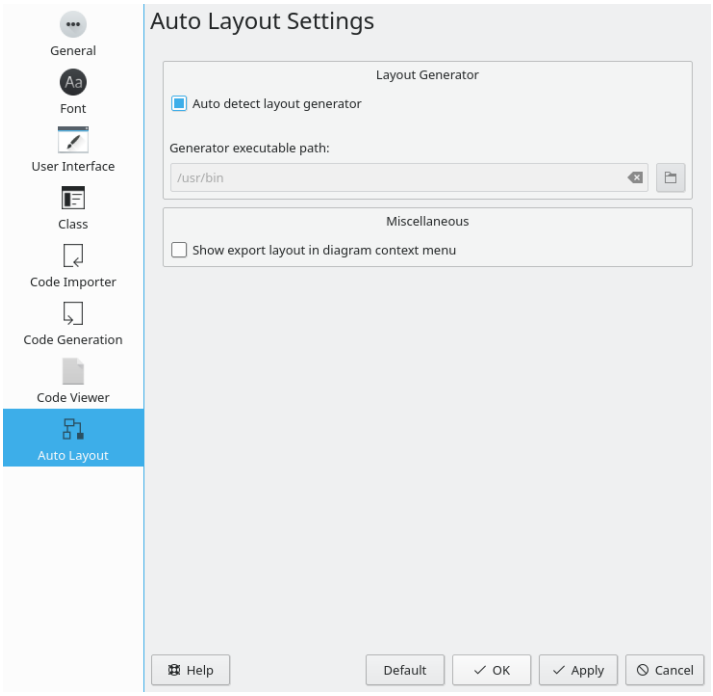


Alternativ i inställningarna av kodvisning i Umbrello UML Modeller

Tillåter anpassning av kodvisningen. Sektionen **Begärt teckensnitt** gör att teckensnitt, teckenstil och och teckenstorlek kan väljas. En representation av det du väljer visas nedanför valen.

I sektionen **Färger** kan ändringar göras av Papper, Teckensnitt, Markerade, UML-objektblock. Gömt block, Text som inte kan redigeras och Text som kan redigeras. Ändringar av färgerna kan göras genom att klicka på färgrutan vid respektive beteckning.

6.8 Inställningar för automatisk layout



Alternativ i inställningarna av automatisk layout i Umbrello UML Modeller

Detektera layoutgenerering automatiskt

Funktionen för automatisk layout beror på layout-generatorer som tillhandahålls av GraphViz-paketet, som normalt installeras tillsammans med Umbrello av en pakethanterare. Umbrello har inbyggt stöd för att detektera installationsplatsen av layout-generatorerna. I fall då beroenden inte är tillgänglig eller inte är lämpliga, kan en annan installationssökväg väljas.

Visa Exportera layout i diagrammets sammanhangsberoende meny

Export av dot-filer utförs genom att använda exportera layout. När alternativet är markerat läggs den exporterade layouten till i listan med tillgängliga diagramlayouter och gör det möjligt att se en snabb förhandsgranskning av dot-export.

Kapitel 7

Upphovsmän och historik

Detta projekt startades av Paul Hensgen som ett av hans universitetsprojekt. Det ursprungliga namnet på programmet var *UML Modeller*. Paul gjorde all utveckling till slutet av 2001, då programmet nådde version 1.0.

Version 1.0 erbjöd redan en hel del funktioner, men efter att projektet hade granskats av Pauls universitet, kunde andra utvecklare delta, och de började ge värdefulla bidrag till UML Modeller, som byte från ett binärt filformat till en XML-fil, stöd för flera sorters UML-diagram, kodgenerering och kodimport, för att bara nämna några få.

Paul var tvungen att avgå från utvecklingsgruppen sommaren 2002, men som fri och öppen programvara, fortsätter programmet förbättras och utvecklas, och underhålls av en grupp utvecklare från olika delar av världen. Projektet ändrade namn från UML Modeller till Umbrello UML Modeller i septemer 2002. Det finns flera skäl till namnändringen, den viktigaste att bara 'uml' som det var känt som, var ett alldeles för generellt namn och orsakade problem med vissa distributioner. En annan viktig orsak är att utvecklarna tycker att Umbrello är ett mycket häftigare namn.

Utvecklingen av Umbrello UML Modeller, samt diskussioner om i vilken riktning programmet ska utvecklas i framtida versioner, är öppen och äger rum via Internet. Om du skulle vilja bidra till projektet, tveka då inte att kontakta utvecklarna. Det finns många sätt som du kan hjälpa Umbrello UML Modeller på:

- Rapportera fel eller förbättringsförslag
- Rätta fel eller lägga till funktioner
- Skriva bra dokumentation eller översätta till andra språk
- Och förstås... koda med oss!

Som du ser, finns det många sätt som du kan bidra på. Samtliga är mycket viktiga och alla är välkomna att delta.

Umbrello UML Modeller-utvecklarna kan nås på umbrello-devel@kde.org.

Kapitel 8

Copyright