**http://docs.kde.org/stable/common/top-kde.jpg Elementos UML**

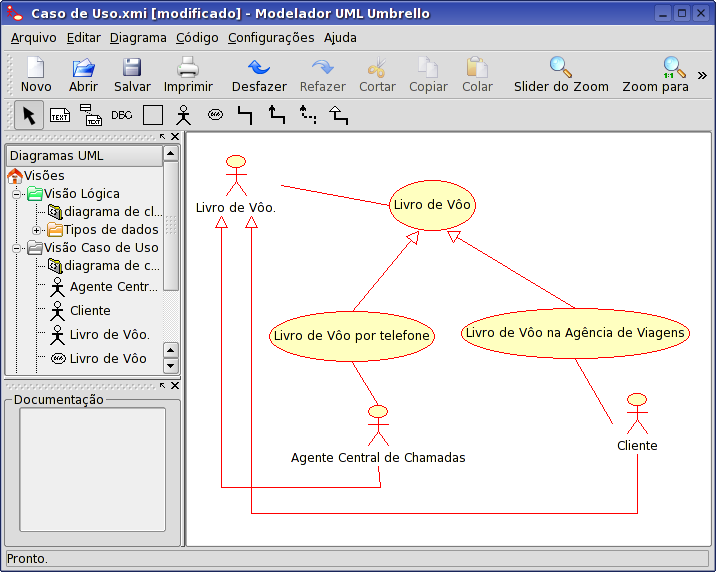
|  |  |  |
| --- | --- | --- |
| [**http://docs.kde.org/**](http://docs.kde.org/) | | |
| [**Anterior**](http://docs.kde.org/stable/pt_BR/kdesdk/umbrello/uml-basics.html) | Fundamentos do UML | [**Próxima**](http://docs.kde.org/stable/pt_BR/kdesdk/umbrello/working-with-umbrello.html) |

**Elementos UML**

**Diagrama de Caso de Uso**

Diagramas de Caso de Uso descrevem relacionamentos e dependências entre um grupo de *Caso de Uso* e os Atores participantes no processo.

É importante observar que Diagramas de Caso de Uso não são adequados para representar o desenho, e não podem descrever os mecanismos internos de um sistema. Diagramas de Caso de Uso são feitos para facilitar a comunicação com os futuros usuários do sistema, e com o cliente, e são especialmente úteis para determinar os recursos necessários que o sistema deve ter. Diagramas de Caso de Uso dizem *o quê* o sistema deve fazer, mas não fazem — e não podem — especificar *como* isto será conseguido.



O Umbrello UML Modeller mostrando um Diagrama de Caso de Uso

**Caso de Uso**

Um *Caso de Uso* descreve — do ponto de vista dos atores — um grupo de atividades num sistema que produz um resultado concreto e tangível.

Casos de Uso são descrições de interações típicas entre os usuários de um sistema e o sistema propriamente dito. Eles representam a interface externa do sistema e especificam um conjunto de exigências do que o sistema deve fazer (lembre-se: somente o quê, não como).

Quando trabalhar com Casos de Uso, é importante lembrar-se de algumas regras simples:

* Cada Caso de Uso está relacionado com no mínimo um ator
* Cada Caso de Uso possui um iniciador (isto é um ator)
* Cada Caso de Uso liga-se a um resultado relevante (um resultado com “valor de negócio”)

Casos de Uso também podem ter relacionamentos com outros Casos de Uso. Os três tipos mais comuns de relacionamento entre Casos de Uso são:

* *<<inclui-se>>* que especifica que um Caso de Uso toma lugar *dentro* de outro Caso de Uso
* *<<estende>>* que especifica que em determinadas situações, ou em algum ponto (chamado um ponto de extensão) um Caso de Uso será estendido por outro.
* *Generalização* especifica que um Caso de Uso herda as características do “Super” Caso de Uso, e pode sobrepor algumas delas ou adicionar novas de maneira semelhante a herança entre classes.

**Ator**

Um ator é uma entidade externa (fora do sistema) que interage com o sistema participando (e frequentemente iniciando) um Caso de Uso. Atores podem ser pessoas reais (por exemplo usuários do sistema), outro sistema de computador ou eventos externos.

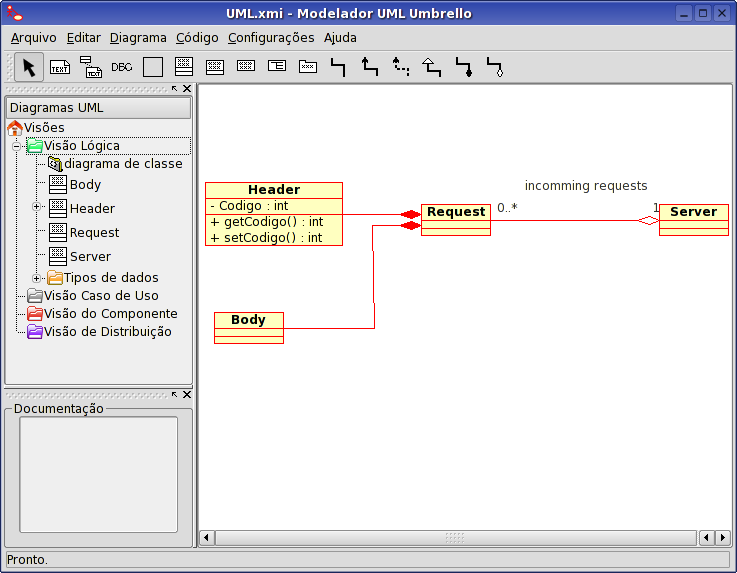
Atores não representam as pessoa *física* ou sistemas, mas sua *regra*. Isto significa que quando uma pessoa interage com o sistema de diferentes maneiras (assumindo diferentes regras) ela será representada por diversos atores. Por exemplo um pessoa que fornece suporte ao cliente por telefone e recebe ordens do cliente para o sistema pode ser representado por um ator da “Equipe de Suporte” e um ator “Representante de Vendas”

**Descrição do Caso de Uso**

Descrição do Caso de Uso são narrativas de texto do Caso de Uso. Elas usualmente tomam a forma de uma nota ou um documento que é de alguma maneira ligado ao Caso de Uso, e explana o processo ou atividades que tomarão lugar no Caso de Uso.

**Diagrama de Classe**

Diagramas de Classe mostram as diferentes classes que fazem um sistema e como elas se relacionam. Os Diagramas de Classe são chamados diagramas “estáticos” porque mostram as classes, com seus métodos e atributos bem como os relacionamentos estáticos entre elas: quais classes “conhecem” quais classes ou quais classes “são parte” de outras classes, mas não mostram a troca de mensagens entre elas.

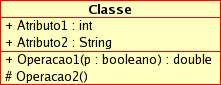


O Umbrello UML Modeller mostrando um Diagrama de Classe

**Classe**

Um Classe define os atributos e os métodos de um conjunto de objetos. Todos os objetos desta classe (instâncias desta classe) compartilham o mesmo comportamento, e possuem o mesmo conjunto de atributos (cada objeto possui seu próprio conjunto). O termo “Tipo” é algumas vezes usado ao invés de Classe, mas é importante mencionar que estes dois termos não são a mesma coisa, e Tipo é um termo mais genérico.

Em UML Classes são representadas por retângulos, com o nome da classe, e podem também mostrar os atributos e operações da classe em dois outros “compartimentos” dentro do retângulo.



Representação visual de uma Classe em UML

**Atributos**

Na UML, atributos são mostrados com pelo menos seu nome, e podem também mostrar seu tipo, valor inicial e outras propriedades. Atributos podem também ser exibidos com sua visibilidade:

* + indica atributos *públicos*
* # indica atributos *protegidos*
* - indica atributos *privados*

**Operações**

Operações (métodos) também são exibidos com pelo menos seu nome, e podem também mostrar seus parâmetros e valores de retorno. Operações podem, como os Atributos, mostras sua visibilidade:

* + indica operações *públicas*
* # indica operações *protegidas*
* - indica operações *privadas*

**Modelos**

Classes podem ter modelos, um valor que é usado para uma classe ou tipo não especificado. O tipo de modelo é especificado quando uma classe é iniciada (isto é um objeto é criado). Modelos existem no C++ moderno e foram introduzidos no Java 1.5 onde eles são chamados de Genéricos.

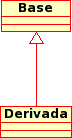
**Associações de Classe**

Classes podem relacionar-se (ser associada com) com outras de diferentes maneiras:

**Generalização**

A herança é um dos conceitos fundamentais da programação orientada por objetos, nos quais uma classe “ganha” todos os atributos e operações da classe que herda, podendo sobrepor ou modificar algumas delas, assim como adicionar mais atributos ou operações próprias.

EM UML, uma associação *Generalização* entre duas classes coloca-as numa hierarquia representando o conceito de herança de uma classe derivada de uma classe base. Em UML, Generalizações são representadas por uma linha conectando duas classes, com uma seta no lado da classe base.



Representação visual de uma generalização em UML

**Associações**

Um associação representa um relacionamento entre classes, e fornece a semântica comum e a estrutura para muitos tipos de “conexões” entre objetos.

Associações são o mecanismo que permite objetos comunicarem-se entre si. Elas descrevem a conexão entre diferentes classes (a conexão entre os objetos atuais é chamada conexão do objeto, ou *link*.

Associações podem ter um regra que especifica o propósito da associação e pode ser uni ou bidirecional (indicando se os dois objetos participantes do relacionamento podem mandar mensagens para o outro, ou se apenas um deles sabe sobre o outro). Cada ponta da associação também possui uma valor de multiplicidade, que dita como muitos objetos neste lado da associação pode relacionar-se com o outro lado.

Em UML, associações são representadas como linhas conectando as classes participantes do relacionamento, e podem também mostrar a regra e a multiplicidade de cada um dos participantes. A multiplicidade é exibida como um intervalo [min...máx] de valores não negativos, com uma estrela (\*) no lado máximo representando infinito.

Representação visual de uma Associação em UML

Representação visual de uma Associação em UML

**Agregação**

Agregações são um tipo especial de associação no qual as duas classes participantes não possuem em nível igual, mas fazem um relacionamento “todo-parte”. Uma Agregação descreve como a classe que possui a regra do todo, é composta (tem) de outras classes, que possuem a regra das partes. Para Agregações, a classe que age como o todo sempre tem uma multiplicidade de um.

Em UML, Agregações são representadas por uma associação que mostra um romboide no lado do todo.

Representação visual de um relacionamento Agregação em UML

Representação visual de um relacionamento Agregação em UML

**Composição**

Composições são associações que representam agregações *muito fortes*. Isto significa que Composições formam relacionamentos todo-parte também, mas o relacionamento é tão forte que as partes não pode existir independentes. Elas existem somente dentro do todo, e se o todo é destruído as partes morrem também.

Em UML, Composições são representadas por um romboide sólido no lado do todo.

Representação visual de um relacionamento Composição em UML

**Outros Itens do Diagrama de Classe**

Diagramas de Classe podem conter diversos outros itens além das classes.

**Interfaces**

Interfaces são classes abstratas que significam instâncias que não podem ser diretamente criadas delas. Elas podem conter operações mas não podem conter atributos. Classes podem derivar de interfaces (através da realização de uma associação) e instâncias podem então ser feitas destes diagramas.

**Tipos de dados**

Tipos de dados são primitivos uma vez que são tipicamente construídos numa linguagem de programação. Exemplos comuns são inteiros e lógicos. Eles não podem ser relacionados à classes mas classes pode se relacionar com eles.

**Enumerações**

Enumerações são uma lista simples de valores. Um exemplo típico é uma enumeração para dias da semana. As opções de uma enumeração são chamadas Literais de Enumeração. Como tipos de dados, elas não podem ter relacionamentos para classes mas classes podem relacionar-se com elas.

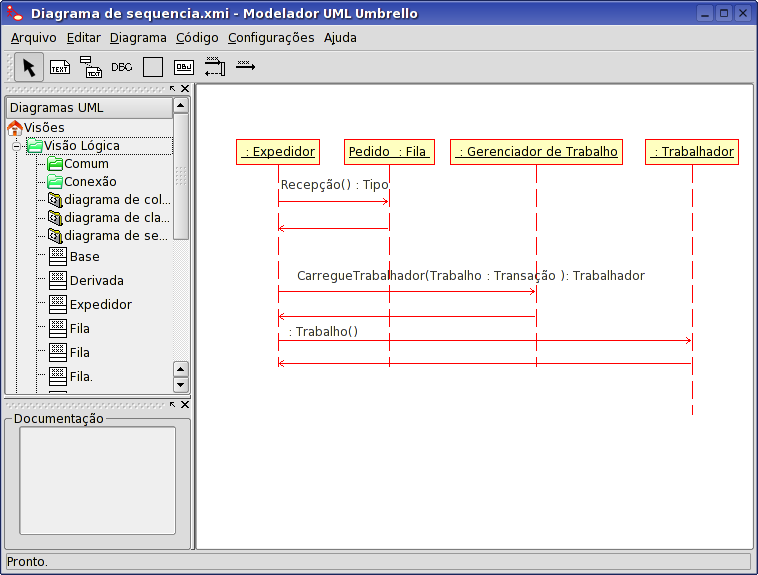
**Pacotes**

Pacotes representam um espaço de nomes numa linguagem de programação. Num diagrama eles são usados para representar partes de um sistema que contém mais de uma classe, talvez centenas de classes.

**Diagramas de Sequência**

Diagramas de Sequência mostram a troca de mensagens (isto é chamada de método) entre diversos Objetos, numa situação específica e delimitada no tempo. Objetos são instâncias de classes. Diagramas de Sequência colocam ênfase especial na ordem e nos momentos nos quais mensagens para os objetos são enviadas.

Em Diagramas de Sequência objetos são representados através de linhas verticais tracejadas, com o nome do Objeto no topo. O eixo do tempo é também vertical, aumentando para baixo, de modo que as mensagens são enviadas de um Objeto para outro na forma de setas com a operação e os nomes dos parâmetros.



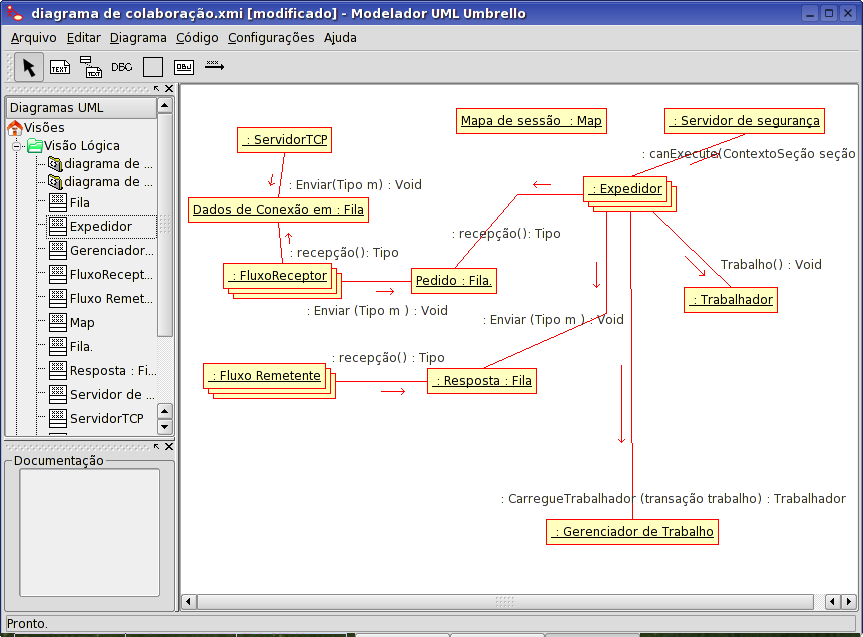
O Umbrello UML Modeller mostrando um Diagrama de Sequência

Mensagens pode ser síncronas, o tipo normal de mensagem de chamada onde o controle é passado para o objeto chamado até o método ter terminado sua execução, ou assíncronas onde o controle é passado diretamente para o objeto chamado. Mensagens síncronas possui uma caixa vertical no lado do objeto chamado para mostrar o controle do fluxo do programa.

**Diagramas de Colaboração**

Diagramas de Colaboração mostram as interações que ocorrem entre os objetos participantes numa situação específica. Isto é mais ou menos a mesma informação mostrada pelos Diagramas de Sequência, mas neste a ênfase é colocada em como as interações ocorrem no tempo, enquanto os Diagramas de Colaboração colocam os relacionamentos entre os objetos e sua topologia em destaque.

Em Diagramas de Colaboração as mensagens enviadas de um objeto para outro são representadas por setas, mostrando o nome da mensagem, parâmetros, e a sequência da mensagem. Diagramas de Colaboração são especialmente indicados para mostrar um fluxo ou situação específica do programa e são um dos melhores tipos de diagrama para rapidamente demonstrar ou explanar um processo na lógica do programa.



O Umbrello UML Modeller mostrando um Diagrama de Colaboração

**Diagrama de Estado**

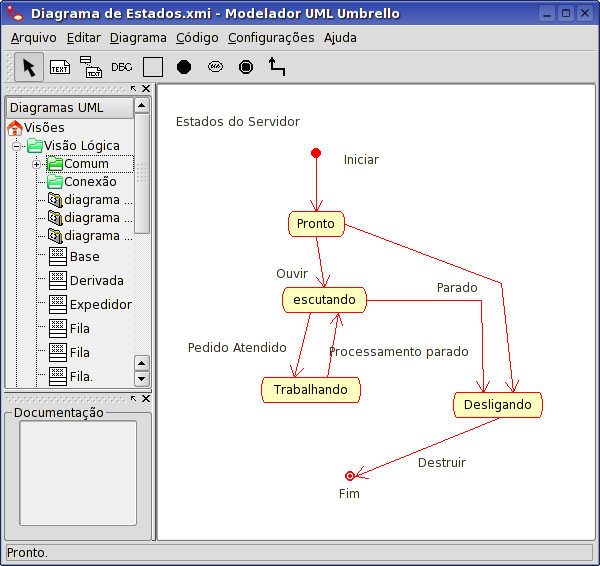
Diagramas de Estado mostram os diferentes estados de um Objeto durante sua vida, e o estímulo que faz com que o Objeto mude seu estado.

Diagramas de Estado veem Objetos como *máquinas de estado* ou automatismos finitos que podem ser um de um conjunto de estados finitos e que podem mudar seu estado através de um de um conjunto finito de estímulos. Por exemplo um tipo de Objeto *ServidorRede* pode estar em um dos seguintes estados durante sua vida:

* Pronto
* Ouvindo
* Trabalhando
* Parado

e os eventos que podem fazer com que o Objeto mude de estado são

* Objeto é criado
* Objeto recebe mensagem ouvir
* Um Cliente solicita uma conexão através da rede
* Um Cliente termina um pedido
* O pedido é executado e terminado
* Objeto recebe mensagem parar
* etc



O Umbrello UML Modeller mostrando um Diagrama de Estado

**Estado**

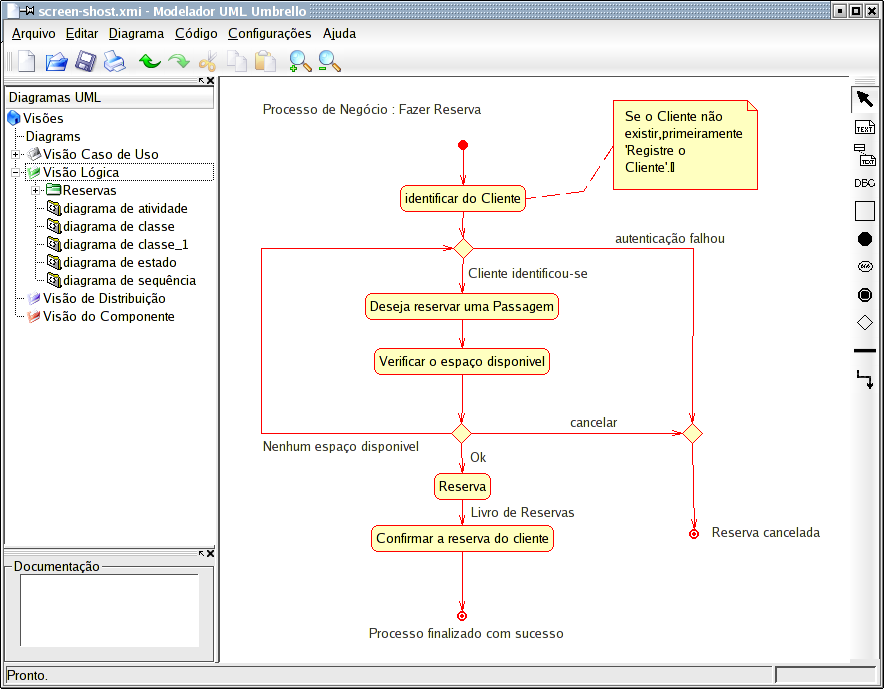
Estados são os blocos construídos dos Diagramas de Estado. Um Estado pertence a exatamente uma classe e representa um resumo dos valores dos atributos que uma classe pode tomar. Um Estado UML descreve o estado interno de um objeto para uma classe em particular

Observe que nem toda mudança em um dos atributos de um objeto pode ser representada por um Estado mas somente aquelas mudanças que podem afetar significativamente o trabalho do objeto

Existem dois tipos especiais de Estados: Inicial e Final. Eles são especiais porque nenhum evento pode fazer com que um Objeto retorne para seu estado Inicial, e da mesma maneira nenhum evento pode tirar um Objeto de seu estado Final uma vez que ele já o tenha alcançado.

**Diagrama de Atividade**

O Diagrama de Atividade descreve a sequência de atividades num sistema com a ajuda as Atividades. Diagramas de Atividade são uma forma especial de Diagramas de Estado, que somente (ou principalmente) contém Atividades.



O Umbrello UML Modeller mostrando um Diagrama de Atividade

Diagramas de Atividade são similares as Diagramas de Fluxo de procedimentos, com a diferença de que todas as Atividades são claramente anexas aos Objetos.

Diagramas de Atividade são sempre associados a um *Classe*, uma *Operação* ou um *Caso de Uso*.

Diagramas de Atividade suportam Atividades sequenciais bem como paralelas. A execução paralela é representada pelos ícones Forquilha/Esperar, e para as Atividades executadas em paralelo, não é importante a ordem na qual elas se executam (elas podem ser executadas ao mesmo tempo ou uma após a outra).

**Atividade**

Uma Atividade é um passo simples num processo. Uma Atividade é um estado no sistema com atividade interna e, pelo menos, uma transição de saída. Atividades podem também ter mais de uma transição de saída se elas possuem condições diferentes.

Atividades podem formar hierarquias, isto significa que uma Atividade pode ser composta por diversas Atividades em “detalhe”, na qual as transições de entrada e saída devem corresponder às transições de entrada e saída do diagrama de detalhe.

**Elementos Auxiliares**

Existem dois elementos em UML que não possuem nenhum valor real semântico para o modelo, mas auxiliam a elucidar partes do diagrama. Estes elementos são

* Linhas de texto
* Notas de Texto e âncoras
* Caixas

Linhas de texto são úteis para adicionar informações curtas de texto ao diagrama. São textos livres e não possuem nenhum significado para o Modelo propriamente dito.

Notas são úteis para adicionar informações mais detalhadas sobre um objeto ou situação específica. Elas possuem a grande vantagem de poderem ser ancoradas a Elementos UMLpara mostrar que a nota “pertence” a um objeto específico ou situação.

Caixas são retângulos de forma livre que podem ser usados para agrupar itens tornando os diagramas mais legíveis. Eles não possuem nenhum significado lógico no modelo.

**Diagramas de Componente**

Diagramas de Componente mostram os componentes do software (sejam componentes de tecnologias como KParts, componentes CORBA ou Java Beans ou apenas seções do sistema que são claramente distintas) e os artefatos de que eles são feitos como arquivos de código-fonte, bibliotecas de programação ou tabelas de bancos de dados relacionais.

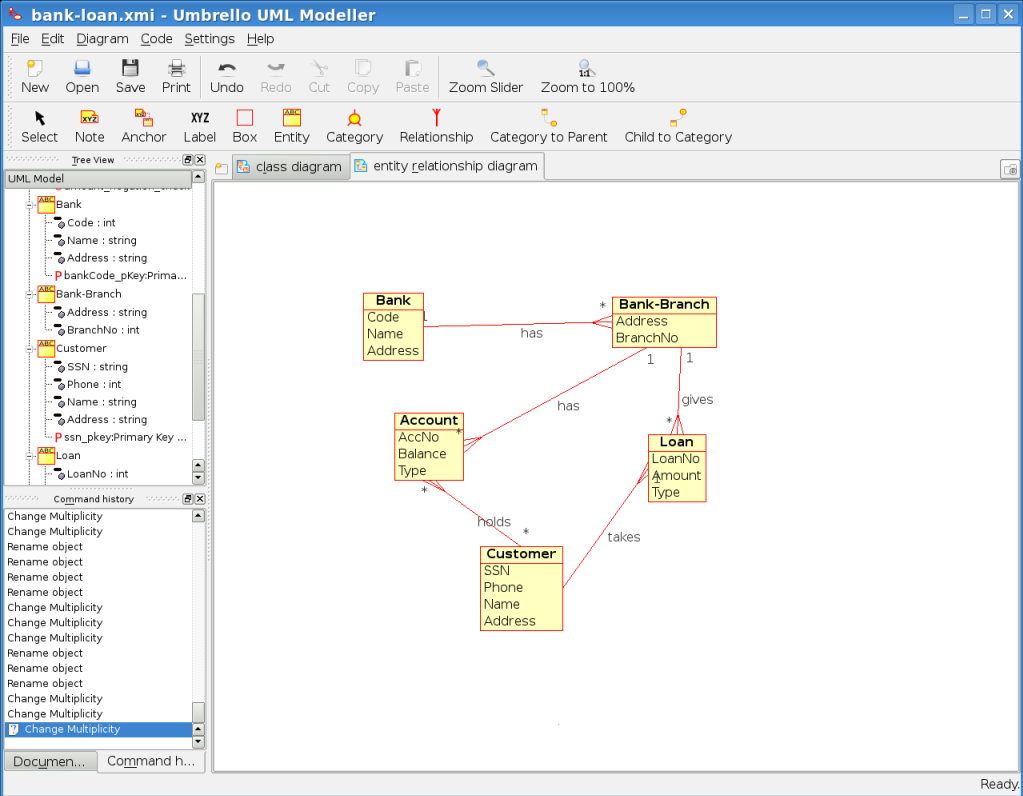
Componentes pode possui interfaces (isto é classes abstratas com operações) que permitem associações entre componentes.

**Diagramas de Distribuição**

Diagramas de distribuição mostram as instâncias dos componentes de tempo de execução e suas associações. Eles incluem Nós que são recursos físicos, tipicamente um computador simples. Eles também mostram interfaces e objetos (instâncias da classe).

**Diagramas de Entidade-Associação**

Os Diagramas de Entidade-Associação (Diagramas ER) mostram o desenho conceitual dos aplicativos de bancos de dados. Eles definem as várias entidades (conceitos) no sistema de informação e as relações e restrições entre eles. Uma extensão dos Diagramas Entidade-Associação, chamada 'Diagramas Extendidos Entidade-Associação' (EER) são usados para incorporar as técnicas de desenho Orientado por Objetos nos diagramas ER.



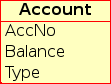
O Umbrello mostrando um Diagrama de Entidade-Associação

**Entidade**

Uma *Entidade* é qualquer conceito no mundo real com uma existência independente. Poderá ser um objeto com uma existência física (exemplo, Computador, Robô) ou poderá ser um objeto com uma existência conceitual ( p.ex.: Curso Universitário). Cada entidade possui um conjunto de atributos que descrevem as propriedades da Entidade.

*Nota:* Não existem ainda notações normalizadas para desenhar Diagramas ER. Os diferentes textos sobre este assunto usam notações diferentes. Os conceitos e notações para os diagramas ER usados no Umbrello são do seguinte livro: *Elmasri R. e Navathe S. (2004). Fundamentals of Database Systems 4ª ed. Addison Wesley*

Num Diagrama ER, as Entidades são representadas por retângulos com o nome da classe, e poderão também mostrar os atributos de uma entidade noutro “compartimento” dentro do retângulo.



Representação visual de uma entidade num Diagrama ER

**Atributos da Entidade**

Nos Diagramas ER, os Atributos das Entidades aparecem com o seu nome num compartimento diferente da Entidade a que pertencem.

**Restrições**

As restrições nos diagramas ER definem as limitações sobre os dados no esquema de informação.

Existem quatro tipos de restrições suportados no Umbrello :

* *Chave Primária:* O conjunto de atributos declarado como *chave primária* é único para a entidade. Só poderá existir uma chave primária numa Entidade e nenhum dos seus atributos constituintes poderá ser nulo.
* *Chave Única:* O conjunto de atributos declarado como *chave única* é único para a entidade. Poderão existir várias restrições de chaves únicas. Os seus atributos constituintes poderão ser nulos. As Chaves Únicas e Chaves Primárias identificam de forma única uma linha numa tabela (entidade)
* *Chave Estrangeira:* Uma Chave Estrangeira é uma restrição de referência entre duas tabelas. A chave estrangeira identifica uma coluna ou conjunto de colunas numa tabela (de referência) que aponta para uma coluna ou conjunto de colunas noutra tabela (referenciada). As colunas da tabela referenciada deverão formar uma chave primária ou única.
* *Restrição de Verificação:* Uma restrição de verificação (também conhecida como restrição de verificação de tabelas) é uma condição que define os dados válidos ao adicionar ou atualizar um item numa tabela de um banco de dados relacional. Uma restrição de verificação é aplicada a cada linha na tabela. A tabela deverá ser um predicado. Poderá fazer uma referência a uma ou várias colunas da tabela.

Exemplo: preço >= 0

**Conceitos Extendidos dos Diagramas de Entidade-Associação (ER)**

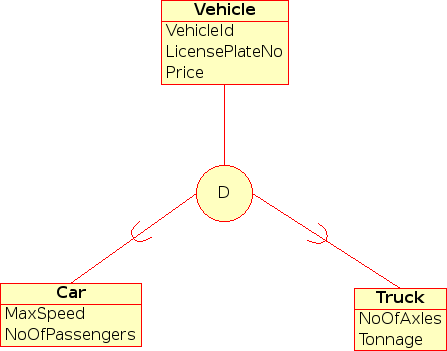
**Especialização**

A especialização é uma forma de criar novas entidades com base em entidades que já tenham sido definidas. As entidades novas, conhecidas como entidades derivadas, recebem (ou herdam) os atributos das entidades pré-existentes, as quais são referenciadas como entidades de base. Pretende ajudar a reutilizar os dados existentes com pouca ou nenhuma modificação.

No Umbrello, uma pessoa poderá definir Especializações Disjuntas e Sobrepostas

**Especialização Disjunta**

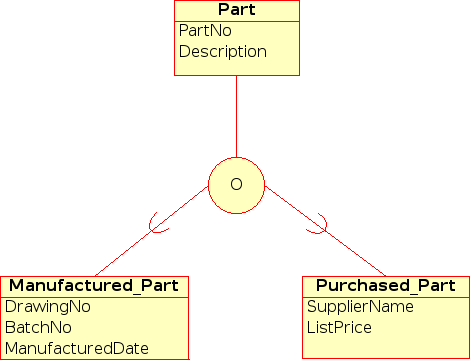
A Especialização Disjunta especifique que as subclasses de especialização devem ser disjuntas. Isto significa que uma entidade pode ser um membro de pelo menos uma das entidades derivadas da especialização



Representação visual de uma Especialização Disjunta num Diagrama EER

**Especialização Sobreposta**

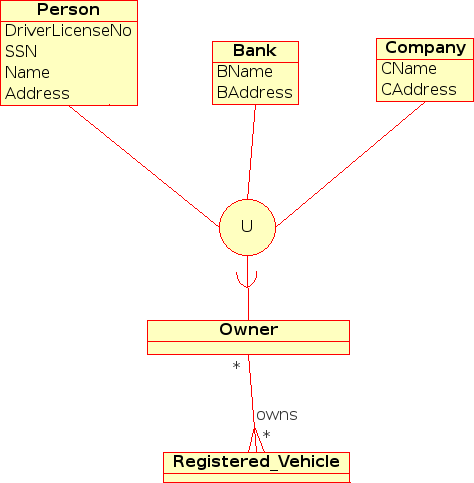
Quando as entidades derivadas não são forçadas a serem disjuntas, este conjunto de entidades são ditos como sendo de uma especialização sobreposta. Isto significa que a mesma entidade do mundo real pode ser membro de mais de uma entidade derivada de especialização.



Representação visual de uma Especialização Sobreposta num Diagrama EER

**Categoria**

Uma Entidade derivada é chamada de *Categoria* quando ela representa uma coleção de objetos que é um subconjunto da União de tipos de entidade distintos. Uma Categoria é modelada quando existe a necessidade de uma único relacionamento de superclasse/subclasse com mais de uma superclasse, onde as superclasses representam diferentes tipos de entidade. ( Como a herança múltipla em Programação Orientada a Objeto ).



Representação visual de uma Categoria num Diagrama EER