Processo Unificado

A **UML** (Unified Modeling Language), que significa Linguagem Unificada de Modelagem é uma linguagem padrão para modelagem orientada a objetos. Ela surgiu da fusão de três grandes métodos, do BOOCH, OMT (Rumbaugh) e OOSE (Jacobson). Esta linguagem de modelagem não proprietária de terceira geração, não é um método de desenvolvimento. Têm como papel auxiliar a visualizar o desenho e a comunicação entre objetos. Ela permite que desenvolvedores visualizem os produtos de seu trabalho em **diagramas** padronizados, e é muito usada para criar modelos de sistemas de software.

Além de fornecer a tecnologia necessária para apoiar a prática de **engenharia de software**orientada a objetos, a UML poderá ser a linguagem de modelagem padrão para modelar sistemas concorrentes e distribuídos. Utiliza-se de um conjunto de técnicas de notação gráfica para criar modelos visuais de software de sistemas intensivos, combinando as melhores técnicas de modelagem de dados, negócios, objetos e componentes. É uma linguagem de modelagem única, comum e amplamente utilizável.

Embora com a UML seja possível representar o software através de modelos orientados a objetos, ela não demonstra que tipo de trabalho deve ser feito, ou seja, não possui um processo que define como o trabalho tem que ser desenvolvido. O objetivo então é descrever "o que fazer", "como fazer", "quando fazer" e "porque deve ser feito". É necessária a elaboração completa de um dicionário de dados, para descrever todas as entidades envolvidas, refinando, com isso, os requisitos funcionais do software.

A Linguagem Unificada de Modelagem possui diagramas (representações gráficas do modelo parcial de um sistema) que são usados em combinação, com a finalidade de obter todas as visões e aspectos do sistema.

Os Diagramas da UML estão divididos em Estruturais e Comportamentais.

Diagramas Estruturais

De Classe: Este diagrama é fundamental e o mais utilizado na UML e serve de apoio aos outros diagramas. O Diagrama de Classe mostra o conjunto de classes com seus atributos e métodos e os relacionamentos entre classes.

De Objeto: O diagrama de objeto esta relacionado com o diagrama de classes e, é praticamente um complemento dele. Fornece uma visão dos valores armazenados pelos objetos de um Diagrama de Classe em um determinado momento da execução do processo do software.

De Componentes: Está associado à linguagem de programação e tem por finalidade indicar os componentes do software e seus relacionamentos.

De implantação: Determina as necessidades de hardware e características físicas do Sistema.

De Pacotes: Representa os subsistemas englobados de forma a determinar partes que o compõem.

De Estrutura: Descreve a estrutura interna de um classificador.

Diagramas Comportamentais

De Caso de Uso (Use Case): Geral e informal para fases de levantamento e análise de Requisitos do Sistema.

De Máquina de Estados: Procura acompanhar as mudanças sofridas por um objeto dentro de um processo.

De Atividades: Descreve os passos a serem percorridos para a conclusão de uma atividade.

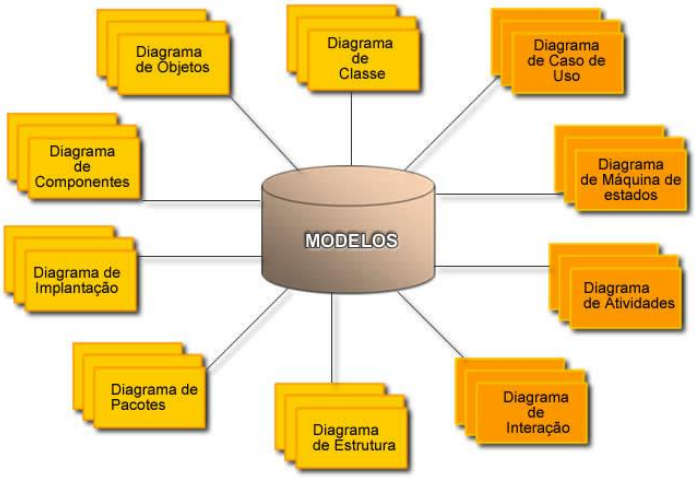
De Interação: Dividem-se em:

1. De Sequência: Descreve a ordem temporal em que as mensagens são trocadas entre os objetos.

2. Geral interação: Variação dos diagramas de atividades que fornece visão geral dentro do sistema ou processo do negócio.

3. De comunicação: Associado ao diagrama de Seqüência, complementando-o e concentrando-se em como os objetos estão vinculados.

4. De tempo: Descreve a mudança de estado ou condição de uma instância de uma classe ou seu papel durante o tempo.



Introdução a UML e seus Diagramas

A Unified Modelling Language (UML) é uma linguagem ou notação de diagramas para especificar, visualizar e documentar modelos de 'software' orientados por objetos. O UML não é um método de desenvolvimento, o que significa que não lhe diz o que fazer primeiro ou o que fazer depois ou como desenhar o seu sistema, mas ajuda-o a visualizar o seu desenho e a comunicar com os outros.
O UML é controlado pelo Object Management Group (OMG) e é a norma da indústria para descrever graficamente o 'software'.

O UML está desenhado para o desenho de 'software' orientado por objetos e tem uma utilização limitada para outros paradigmas de programação.

A UML é composta por muitos elementos de modelo que representam as diferentes partes de um sistema de software. Os elementos UML são usados para criar diagramas, que representam uma determinada parte, ou um ponto de vista do sistema.

Objetivos

Os objetivos da UML são: especificação, documentação, estruturação para sub-visualização e maior visualização lógica do desenvolvimento completo de um sistema de informação. A UML é um modo de padronizar as formas de modelagem.

Para que usar UML?

Ajudar a conceber nossas idéias, em relação ao sistema que estivermos projetando – Pensar antes de codificar;

Apresentar nossas idéias ao grupo de forma que todos possam interagir e discutir um determinado ponto – Aumentar a participação e envolvimento do time;

Documentar nossas idéias quando elas já estiverem bem consolidadas para que novos integrantes e novos colaboradores possam acelerar sua compreensão dos sistemas desenvolvidos pelo grupo;

Tipos de Diagramas

Diagramas servem para capturar diferentes visões do sistema:

Estrutural: estática

Diagrama de Classes

Diagrama de Objetos

Diagrama de Componentes

Diagrama de Implantação

Comportamental: dinâmica

Diagrama de Casos de Uso

Diagrama de Seqüência

Diagrama de Atividades

Diagrama de Estados

Diagrama de Colaboração

Diagramas Estáticos

Diagrama de Classes

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.

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:

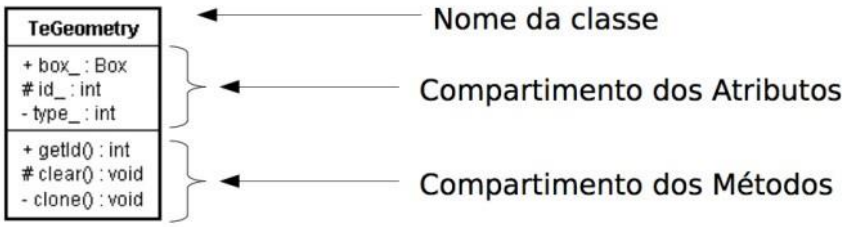
+ indica atributos ou métodos públicos (significa que ele pode ser acessado de qualquer lugar da classe ou sub-classe);

indica atributos ou métodos protegidos (significa que ele só pode ser acessado de dentro da própria classe ou em suas classes-filha);

– indica atributos ou métodos privados (significa que ele só pode ser acessado de dentro da própria classe.);

~ pacote.

As operações também são exibidas com pelo menos seu nome, e podem também mostrar seus parâmetros e valores de retorno.



Uma observação a ser feita é que quando se tem um atributo ou método estáticos, eles irão aparecer sublinhados no diagrama. Já os métodos abstratos irão aparecer em itálico.

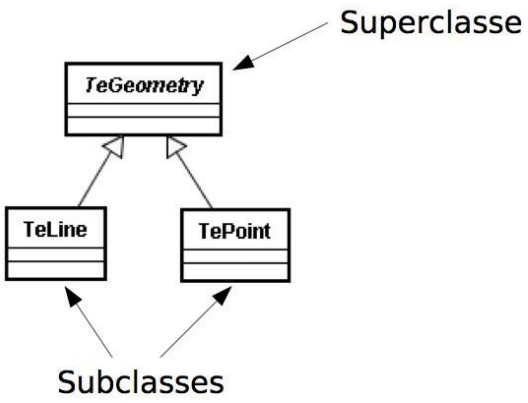
Classes podem ter modelos, um valor que é usado para uma classe ou tipo não especificado. O tipo de modelo é especificado quando uma classe é iniciada (isto é um objeto é criado). Modelos existem no C ++ moderno e foram introduzidos no Java 1.5 onde eles são chamados de genéricos.

Existem alguns tipos de relacionamentos no diagrama de classes que veremos abaixo:

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.



Generalização

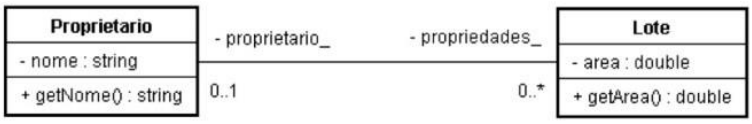
Associação

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.



Associação

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.

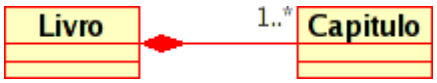
Em UML, Agregações são representadas por uma associação que mostra um romboide no lado do todo.

Agregação

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.



Composição

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.

Diagrama de Objetos

O diagrama de objetos é uma variação do diagrama de classes e utiliza quase a mesma notação. A diferença é que o diagrama de objetos mostra os objetos que foram instanciados das classes. O diagrama de objetos é como se fosse o perfil do sistema em um certo momento de sua execução.

A mesma notação do diagrama de classes é utilizada com duas exceções: os objetos são escritos com seus nomes sublinhados e todas as instâncias num relacionamento são mostradas. Os diagramas de objetos não são tão importantes como os diagramas de classes, mas eles são muito úteis para exemplificar diagramas complexos de classes ajudando muito em sua compreensão. Diagramas de objetos também são usados como parte dos diagramas de colaboração(passou a se

chamar comunicação na uml 2.0), onde a colaboração dinâmica entre os objetos do sistema são mostrados.

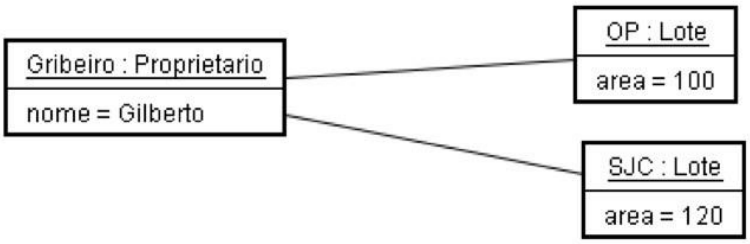


Diagrama de Objetos

Diagrama de Componentes

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

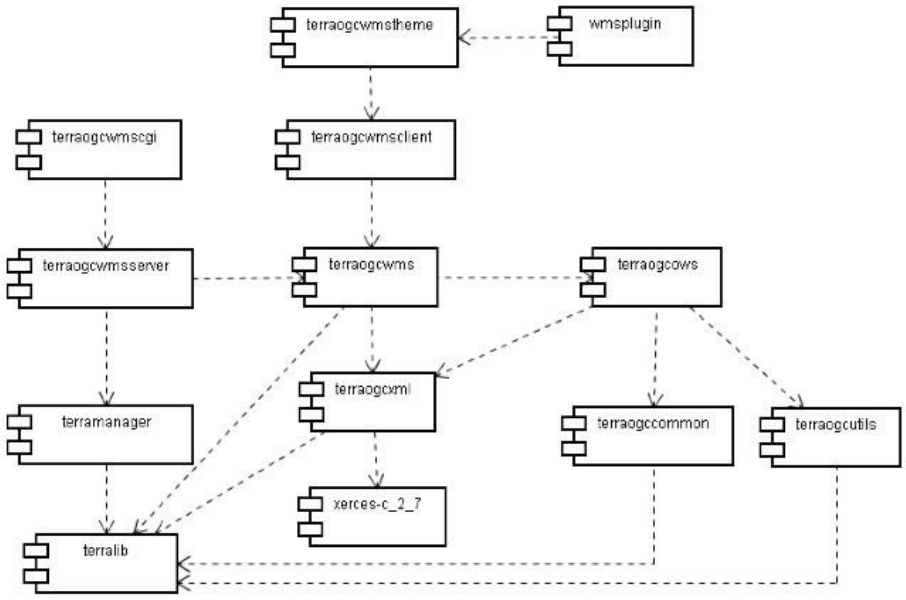


Diagrama de Componentes

Diagrama de Implantação

O Diagrama de instalação/implantação é definido pela Linguagem de Modelagem Unificada (Unified Modeling Language – UML), descreve os componentes de hardware e software e sua interação com outros elementos de suporte ao processamento. Representa a configuração e a arquitetura de um sistema em que estarão ligados seus respectivos componentes, sendo representado pela arquitetura física de hardware, processadores etc.

Nó: Representa uma peça física de equipamento na qual o sistema será implantado.

Artefatos: Qualquer pedaço físico de informação usada ou produzida por um sistema.

Especificação de implantação: Especifica um conjunto de propriedades que determina os parâmetros de execução de um artefato que está instalado em um nó.

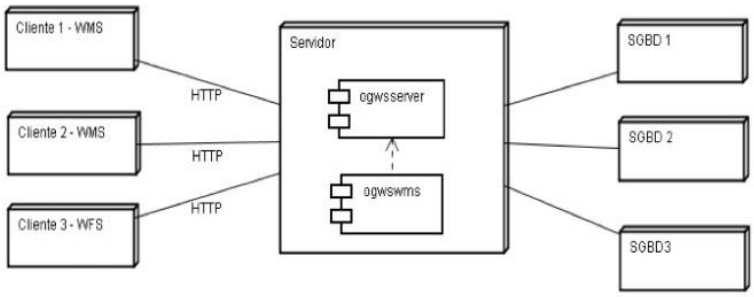


Diagrama de Implantação

Diagramas Dinâmicos

Diagrama de Casos 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.

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

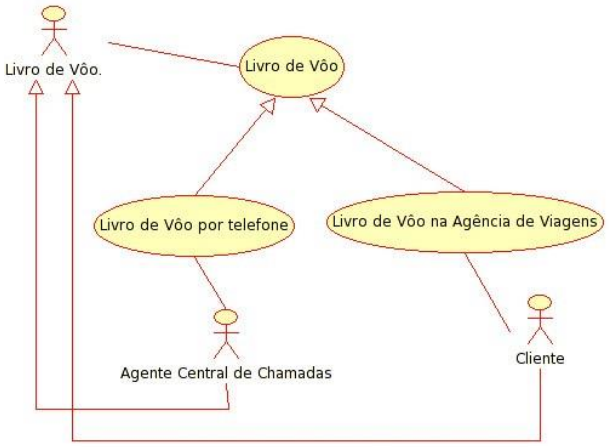
Um ator é uma entidade externa (fora do sistema) que interage com o sistema participando (e frequentemente iniciando) um Caso de Uso. Atores podem ser pessoas reais (por exemplo usuários do sistema), outro sistema de computador ou eventos externos.

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.



Caso de Uso

Diagrama de Sequência

É usado para mostrar uma seqüência de atividades. Mostra o fluxo de trabalho (workflow) a partir de um ponto inicial até um ponto final, detalhando as decisões do caminho tomado durante a execução das tarefas. Este diagrama possui várias aplicações, desde a definição do fluxo básico de um programa até a definição de um processo com as suas tomadas de decisões e ações.

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.

Diagrama de Sequência

Diagramas 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.

Os diagramas de atividade não são importantes somente para a modelagem de aspectos dinâmicos de um sistema ou um fluxograma, mas também para a construção de sistemas executáveis por meio de engenharia de produção reversa.

Alguns conceitos:

Atividades: Comportamento a ser realizado.

Sub-atividade: Execução de uma sequência não atómica de atividades.

Transição: Fluxo de uma atividade para outra.

Ação: Transformação.

Decisão: Dependendo de uma condição, mostra as diferentes transições.

Raia: Diferenciação de unidades organizacionais.

Bifurcação (Fork): Separa uma transição em várias transições executadas ao mesmo tempo.

Sincronização (Join): Concatenação de transições vindas do Fork.

Objecto: O objecto da atividade.

Envio de sinal: Transição pra um meio externo, por exemplo, um hardware.

Recepção de sinal: Recepção do envio.

Região: Agrupamento de uma ou mais atividades.

Exceção: Atividades que ocorrerem em decorrência de uma excepção.

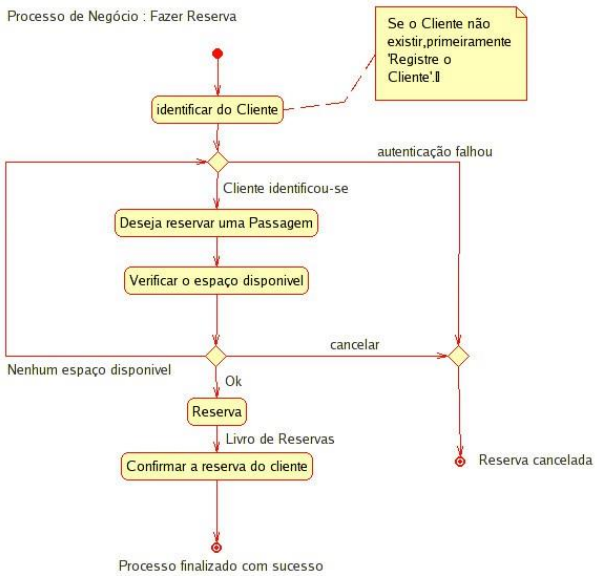


Diagrama de Atividades

Diagramas de Estado

Modela o comportamento de um objeto individual. Especifica as seqüências de estados pelos quais um objeto passa durante seu tempo de vida em resposta a eventos.

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

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.

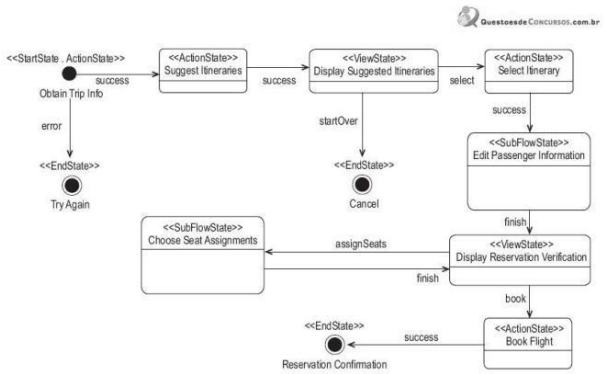
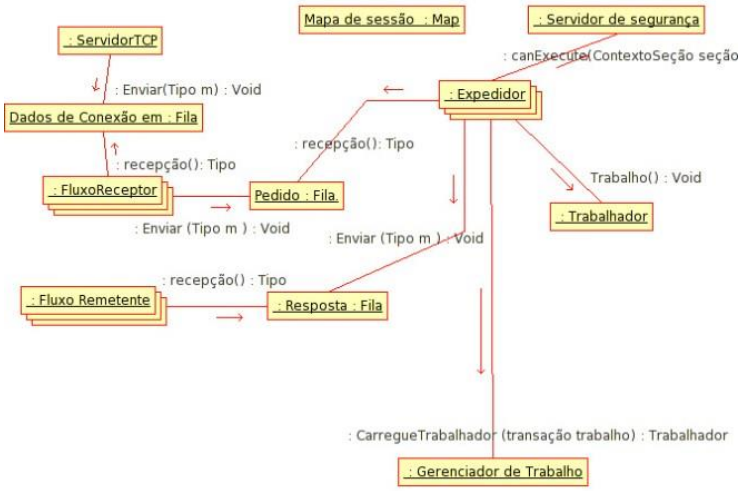


Diagrama de Estados

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 que é UML?

A Linguagem de modelagem unificada (UML) foi criada para estabelecer uma linguagem de modelagem visual comum, semanticamente e sintaticamente rica, para arquitetura, design e implementação de sistemas de software complexos, tanto estruturalmente quanto para comportamentos. Além do desenvolvimento de software, a UML tem aplicações em fluxos do processo na fabricação.

É análoga aos modelos utilizados em outros campos, e é composta por diferentes tipos de diagramas. De modo geral, diagramas UML descrevem o limite, a estrutura e o comportamento do sistema e os objetos nele contidos.

A UML não é uma linguagem de programação, mas existem ferramentas que podem ser usadas para gerar código em várias linguagens por meio de diagramas UML. A UML tem uma relação direta com a análise e o design orientados a objetos.

A UML e seu Papel na Modelagem e no Design Orientados a Objetos

Existem muitos paradigmas ou modelos de resolução de problemas na ciência da computação, que é o estudo de algoritmos e dados. Há quatro categorias modelo de resolução de problemas: linguagens imperativas, funcionais, declarativas e orientadas a objetos (POO). Em linguagens orientadas a objetos, os algoritmos são expressos através da definição de ‘objetos’, e por meio da interação dos objetos uns com os outros. Esses objetos são coisas a serem manipulados, e eles existem no mundo real. Eles podem ser edifícios, widgets em um desktop ou seres humanos.

Linguagens orientadas a objetos dominam o mundo da programação porque elas modelam objetos do mundo real. A UML é uma combinação de várias notações orientadas a objetos: design orientado a objetos, técnica de modelagem de objetos e engenharia de software orientada a objetos.

A UML usa os pontos fortes destas três abordagens para apresentar uma metodologia mais consistente e mais fácil de usar. A UML representa as melhores práticas para desenvolver e documentar aspectos diferentes da modelagem de software e sistemas de negócios.

A História e as Origens da UML

'Os Três Amigos' da engenharia de software, como eram conhecidos, desenvolveram ainda outras metodologias. Uniram-se para proporcionar maior clareza aos programadores por meio da criação de novos padrões. A colaboração entre Grady, Booch e Rumbaugh fortaleceu os três métodos e melhorou o produto final.

Em 1996, os grandes esforços destes pensadores resultaram no lançamento dos documentos UML 0.9 e 0.91. Logo ficou claro que muitas organizações, incluindo Microsoft, Oracle e IBM, passaram a considerar a UML como algo crucial para seu próprio desenvolvimento de negócios. Juntamente com muitos outros indivíduos e empresas, as organizações estabeleceram recursos que poderiam desenvolver uma linguagem de modelagem plena. Em 1999, Os Três Amigos publicaram "UML: guia do usuário", e uma atualização que inclui informações sobre UML 2.0, em sua segunda edição, de 2005.

OMG: tem outro significado

Segundo o site, The Object Management Group® (OMG®) (Grupo de gerenciamento de objetos, em tradução livre) é uma organização internacional voltada para aprovações de padrões tecnológicos, aberta a membros, sem fins lucrativos e fundada em 1989. O padrões OMG são implementados por fornecedores, utilizadores finais, instituições acadêmicas e agências governamentais. Forças-tarefas OMG desenvolvem padrões de integração empresarial para muitas tecnologias e um grande número de indústrias. Os padrões de modelagem OMG, incluindo UML e Model Driven Architecture® (MDA®), possibilitam uma criação de design visualmente poderoso e a execução e manutenção de software, entre outros processos.

A OMG supervisiona a definição e manutenção de especificações UML. Esta supervisão permite aos engenheiros e programadores usarem uma única linguagem para diversas finalidades durante todas as fases do ciclo de vida do software e para todos os tamanhos de sistemas.

O objetivo da UML de acordo com a OMG

O propósito da UML de acordo com a OMG:

Fornecer a arquitetos de sistemas, engenheiros de software e desenvolvedores de software ferramentas de análise, design e implementação para sistemas baseados em software, bem como para a modelagem de processos de negócios e similares.

Desenvolver as condições gerais da indústria ao permitir a interoperabilidade de ferramentas de modelagem visual de objetos. No entanto, para permitir a troca significativa de informações de modelos entre as ferramentas, é necessário um acordo sobre semântica e notação.

A UML atende aos seguintes requisitos:

Estabelecer uma definição formal de um metamodelo baseado em meta-objetos (MOF, em inglês) comuns que especifica a sintaxe abstrata da UML. A sintaxe abstrata define o conjunto de conceitos de modelagem UML, seus atributos e relacionamentos, bem como as regras para combinar estes conceitos para construir modelos UML parciais ou completos.

Fornecer uma explicação detalhada da semântica de cada conceito de modelagem UML. A semântica define, de forma independente da tecnologia, como os conceitos UML devem ser realizados por computadores.

Especificar os elementos de notação legíveis para humanos para representar os conceitos de modelagem UML individuais, bem como regras para combiná-los em uma variedade de tipos de diagramas correspondentes a diferentes aspectos dos sistemas modelados.

Definir maneiras pelas quais ferramentas UML podem entrar em conformidade com esta especificação. Isto é suportado (em uma especificação separada) por uma especificação baseada em XML de formatos de modelos de intercâmbio correspondentes (XMI) que devem ser realizados por ferramentas compatíveis.

UML e modelagem de dados

A UML é bastante usada pelos programadores, mas não é geralmente empregada por desenvolvedores de banco de dados. Um motivo é simplesmente que os criadores da UML não focaram banco de dados. Apesar disso, a UML é eficaz para níveis altos de modelagem de dados conceituais, e pode ser usada em diferentes tipos de diagramas UML. Você pode encontrar mais informações sobre como criar camadas de um modelo de classe orientado a objetos em um banco de dados relacional neste artigo sobre Modelagem de banco de dados em UML.