

Briefing Sobre a Unified Modeling Language (UML)

Resumo Executivo

Este documento sintetiza os conceitos, a estrutura e a aplicação da Unified Modeling Language (UML), uma linguagem gráfica padrão para a modelagem de sistemas. Nascida da união de metodologias proeminentes na década de 1990 por James Rumbaugh, Grady Booch e Ivar Jacobson ("The Three Amigos") na Rational Software, a UML foi padronizada pela Object Management Group (OMG) para unificar a modelagem orientada a objetos.

A UML é uma ferramenta versátil, independente de processo e de linguagem de programação, utilizada para especificar, visualizar, construir e documentar artefatos de sistemas. Embora primariamente associada ao software, sua expressividade permite a modelagem de processos de negócio e sistemas em diversas áreas, como finanças e aeroespacial. Sua principal força, segundo Martin Fowler, reside em facilitar a comunicação e o entendimento entre as equipes, oferecendo um meio-termo entre a ambiguidade da linguagem natural e o detalhamento excessivo do código.

A linguagem é organizada em 14 tipos de diagramas, divididos em duas categorias principais:

1. Diagramas Estruturais: Representam os aspectos estáticos de um sistema, como classes, componentes e a distribuição física do hardware. Os diagramas mais importantes desta categoria são o Diagrama de Classes, que serve como espinha dorsal da modelagem orientada a objetos, e o Diagrama de Componentes.
2. Diagramas Comportamentais: Descrevem os aspectos dinâmicos, mostrando como o sistema se comporta e interage ao longo do tempo. Dentro desta categoria, destacam-se:
 - Diagrama de Casos de Uso: Essencial para o levantamento de requisitos funcionais a partir da perspectiva do usuário.
 - Diagrama de Atividades: Modela fluxos de trabalho e processos de negócio, suportando comportamento paralelo.
 - Diagrama de Máquina de Estados: Descreve o ciclo de vida de um objeto através de suas mudanças de estado.

- Diagrama de Sequência e de Comunicação: Detalham a interação entre objetos, com foco na ordem temporal (Sequência) ou na organização estrutural (Comunicação).

Mecanismos como Estereótipos permitem a extensão da semântica da UML, adaptando-a a domínios ou plataformas específicas, o que confere grande flexibilidade à linguagem.

1. Fundamentos e Origem da UML

1.1. Histórico e Criação

A Unified Modeling Language (UML) surgiu em meados da década de 1990 como uma solução para a proliferação de linguagens de modelagem que dificultava a adoção da tecnologia de Orientação a Objetos. A empresa Rational Software Corporation reuniu três dos mais proeminentes pesquisadores da área, conhecidos como "The Three Amigos":

- James Rumbaugh: Criador da Técnica de Modelagem de Objetos (TMO).
- Grady Booch: Criador do método de Projeto Orientado a Objetos (POO).
- Ivar Jacobson: Criador do método de Engenharia de Software Orientada a Objetos (ESOO).

O objetivo era consolidar seus respectivos métodos em uma linguagem de modelagem unificada e não proprietária.

1.2. Padronização e Evolução

Para evitar que um padrão controlado pela Rational gerasse vantagens competitivas desleais, outros fornecedores (IBM, HP, Oracle, etc.) incitaram a Object Management Group (OMG) — um consórcio industrial responsável por padrões de interoperabilidade — a assumir o processo de padronização.

- UML 1.1: Publicada em agosto de 1997 e adotada como padrão oficial pela OMG. Posteriormente, tornou-se o padrão internacional ISO/IEC 19501:2005.
- UML 2.0: Lançada em 2005, representou uma grande maturação da linguagem, corrigindo inconsistências e integrando novos conceitos.
- UML 2.5: É a versão atual, resultado de diversas pequenas atualizações desde a versão 2.0.

1.3. Definição e Propósito

A UML é definida como uma linguagem gráfica para especificar, visualizar e documentar artefatos de um sistema, primariamente de software. No entanto, sua aplicabilidade transcende essa área, sendo utilizada em telecomunicações, defesa, setor bancário e até para modelar sistemas não computacionais, como fluxogramas do sistema judiciário.

Características Centrais:

- Independente de Processo: Pode ser aplicada em diferentes contextos de desenvolvimento, como o RUP (Rational Unified Process).
 - Independente de Tecnologia: Não está atrelada a uma linguagem de programação específica, podendo ser usada com linguagens estruturadas (C, Cobol) ou orientadas a objetos.
 - Foco em Comunicação: Conforme destacado por Martin Fowler, seu principal valor é melhorar a comunicação e o entendimento dentro de uma equipe. A notação gráfica atua como um intermediário eficaz entre a imprecisão da linguagem natural e a complexidade do código-fonte.
-

1. Arquitetura e Mecanismos da UML

A UML é definida por quatro especificações inter-relacionadas e possui mecanismos gerais que permitem sua extensão e clareza.

2.1. Especificações Fundamentais

| Especificação | Descrição |
|----------------------------------|--|
| Infraestrutura | Contém o núcleo da arquitetura, perfis e estereótipos. |
| Superestrutura | Contém os elementos de modelagem estáticos e dinâmicos (os diagramas). |
| Object Constraint Language (OCL) | Linguagem formal e declarativa para descrever regras e restrições nos modelos UML. |

| Especificação | Descrição |
|--------------------------|---|
| Intercâmbio de Diagramas | Define os formatos para o intercâmbio de diagramas entre ferramentas. |

2.2. Mecanismos de Uso Geral

| Mecanismo | Descrição |
|--------------------|--|
| Estereótipo | Estende o significado de um elemento do modelo, permitindo a adaptação da UML a domínios específicos. Pode ser predefinido (<>) ou definido pelo usuário (<>), e representado textualmente (<>) ou graficamente (ícone). |
| Notas Explicativas | Comenta ou esclarece uma parte de um diagrama sem alterar a semântica do modelo. Graficamente, é um retângulo com uma "orelha", ligado ao elemento por uma linha tracejada. |
| Tagged Values | Define propriedades adicionais (metadados) para elementos de um modelo. A partir da UML 2.0, seu uso está associado a estereótipos. |
| Restrições | Especifica condições ou regras que um ou mais elementos do modelo devem satisfazer. São delimitadas por chaves {{restrição}} e podem ser escritas em OCL (formal) ou texto livre (informal). |
| Pacotes | Agrupa elementos semanticamente relacionados (classes, casos de uso, etc.) em unidades de mais alto nível, ajudando a organizar modelos complexos. |

2.3. Visões Arquiteturais (Modelo 4+1)

A arquitetura de um software pode ser descrita a partir de cinco visões concorrentes, cada uma focada em um conjunto específico de interesses.

| Visão | Perspectiva | Foco | Diagramas Principais |
|---------------------------------------|------------------------|--|--|
| Lógica (ou de Projeto) | Usuário Final | Requisitos funcionais e estrutura do sistema (classes, objetos). | Classe, Objetos, Pacotes |
| Desenvolvimento (ou de Implementação) | Programador | Organização estática dos módulos de software. | Componentes |
| Processo | Integrador | Requisitos não-funcionais (desempenho, escalabilidade) e concorrência. | Sequência, Estrutura Composta, Máquina de Estados, Atividade |
| Física (ou de Implantação) | Engenheiro de Sistemas | Topologia física do sistema (hardware e distribuição de software). | Implantação, Componentes |
| Casos de Uso (ou de Cenários) | Todos os Interessados | Consistência e validação do sistema através de cenários de uso. | Casos de Uso |

1. Classificação dos Diagramas UML

A UML 2.x define 14 tipos de diagramas, agrupados em três categorias (sendo a terceira um subconjunto da segunda).

3.1. Diagramas Estruturais

Representam os aspectos estáticos do sistema, ou seja, sua estrutura em um determinado momento, sem considerar a passagem do tempo.

- Diagrama de Classes

- Diagrama de Objetos
- Diagrama de Componentes
- Diagrama de Implantação
- Diagrama de Pacotes
- Diagrama de Estrutura Composta
- Diagrama de Perfil

3.2. Diagramas Comportamentais

Representam os aspectos dinâmicos do sistema, descrevendo como os processos e funcionalidades se relacionam e evoluem ao longo do tempo.

- Diagrama de Casos de Uso
- Diagrama de Atividades
- Diagrama de Máquina de Estados
- Diagrama de Sequência
- Diagrama de Comunicação
- Diagrama de Tempo
- Diagrama de Interação Geral

3.3. Diagramas de Interação

São um subconjunto dos diagramas comportamentais que focam na troca de informações e no fluxo de controle entre os objetos do sistema.

- Diagrama de Sequência
- Diagrama de Comunicação
- Diagrama de Tempo
- Diagrama de Interação Geral

1. Análise Detalhada dos Diagramas Estruturais

4.1. Diagrama de Classes

É o diagrama mais utilizado e cobrado. Descreve a estrutura estática do sistema em termos de classes, interfaces, seus atributos, operações e os relacionamentos entre elas.

- Classe: Abstração de um conjunto de objetos com características similares. Pode ser representada com diferentes níveis de detalhe (apenas nome; nome e atributos; nome, atributos e operações). Uma

Classe Ativa (com borda dupla) representa objetos que possuem seus próprios fluxos de controle (threads).

- Atributos e Operações:
- Um atributo/operação estático é sublinhado.
- Uma operação abstrata é escrita em itálico.
- Visibilidade: Define o nível de acesso a atributos e operações.

| Modificador | Símbolo UML | Acesso na UML |
|-------------|-------------|--|
| public | + | Acessível por qualquer elemento. |
| protected | # | Acessível pela própria classe e suas subclasses. |
| package | ~ | Acessível por elementos no mesmo pacote. |
| private | - | Acessível apenas pela própria classe. |

- Relacionamentos:
- Dependência: Um relacionamento "usa-um", onde uma mudança em um elemento (independente) pode afetar outro (dependente). Representado por uma seta tracejada apontando para o elemento independente.
- Generalização (Herança): Um relacionamento "é-um", onde uma subclasse herda características de uma superclasse. Representado por uma linha sólida com uma seta triangular vazia apontando para a superclasse.
- Realização: Um relacionamento onde um elemento (ex: uma classe) implementa o comportamento especificado por outro (ex: uma interface). Representado por uma linha tracejada com uma seta triangular vazia apontando para o elemento que especifica o contrato.
- Associação: Relacionamento estrutural que descreve conexões entre instâncias de classes.
- Simples: Conexão geral entre objetos. Representada por uma linha sólida.
- Agregação: Associação "tem-um" onde as partes podem existir independentemente do todo (ex: um carro e suas rodas).

Representada por uma linha sólida com um losango vazio no lado do todo.

- Composição: Uma agregação forte onde as partes não podem existir sem o todo (ex: um carro e sua pintura). A existência da parte é dependente do todo. Representada por uma linha sólida com um losango preenchido no lado do todo.

4.2. Outros Diagramas Estruturais

- Diagrama de Objetos: Mostra uma "fotografia" do sistema em um momento específico, representando instâncias concretas de classes (joao:Pessoa) e seus relacionamentos. Útil para exemplificar estruturas complexas do diagrama de classes.
- Diagrama de Componentes: Apresenta a organização e as dependências entre os componentes de software (módulos, bibliotecas, arquivos executáveis). Foca na visão de implementação e utiliza interfaces fornecidas (o que o componente oferece) e requeridas (o que ele precisa).
- Diagrama de Pacotes: Organiza os elementos do modelo em grupos lógicos (pacotes), mostrando as dependências entre eles. É crucial para gerenciar a complexidade de sistemas grandes.
- Diagrama de Implantação: Descreve a arquitetura física do sistema, mostrando como os artefatos de software são distribuídos nos nós de hardware (servidores, dispositivos).
- Diagrama de Perfil: Opera no nível de metamodelo e é usado para criar customizações da UML para domínios específicos através da definição de um conjunto de estereótipos, tagged values e restrições.
- Diagrama de Estrutura Composta: Detalha a estrutura interna de um classificador (como uma classe ou componente), mostrando suas partes internas e como elas colaboram para realizar uma funcionalidade.

1. Análise Detalhada dos Diagramas Comportamentais

5.1. Diagrama de Casos de Uso

Captura os requisitos funcionais do sistema a partir da perspectiva do

usuário. Descreve as interações entre atores e o sistema para atingir um objetivo.

- Ator: Representa um papel desempenhado por uma entidade externa (humano, outro sistema, hardware) que interage com o sistema.
- Caso de Uso: Descreve uma sequência de ações que o sistema executa para produzir um resultado de valor observável para um ator.
- Relacionamentos:
- Comunicação (Associação): Liga um ator a um caso de uso, indicando interação.
- Inclusão (<>): Um caso de uso base obrigatoriamente inclui a funcionalidade de outro. Usado para reutilizar comportamento comum. A seta aponta do caso de uso base para o incluído.
- Extensão (<>): Um caso de uso opcionalmente estende o comportamento de outro em um ponto específico (ponto de extensão). Usado para modelar fluxos alternativos ou excepcionais. A seta aponta do caso de uso extensor para o estendido.
- Herança (Generalização): Um ator/caso de uso especializado herda o comportamento de um mais genérico.

5.2. Diagrama de Atividades

Modela fluxos de trabalho e processos de negócio, descrevendo a sequência de ações. É semelhante a um fluxograma, mas com suporte nativo a paralelismo.

- Ação: Um passo único dentro de uma atividade.
- Nô de Decisão (Ramificação): Representado por um losango, indica um ponto onde o fluxo se divide com base em uma condição.
- Nô de Fork/Join (Bifurcação/União): Representado por uma barra, divide o fluxo em múltiplos caminhos paralelos (fork) ou sincroniza múltiplos caminhos em um único fluxo (join).
- Partições (Swimlanes): Organiza as ações em colunas ou linhas que representam as responsabilidades de diferentes atores ou componentes.

5.3. Diagrama de Máquina de Estados

Descreve o ciclo de vida de um único objeto, mostrando os diferentes

estados em que ele pode se encontrar e as transições entre esses estados em resposta a eventos.

- Estado: Uma condição ou situação na vida de um objeto.
- Transição: A passagem de um estado para outro, geralmente disparada por um evento.
- Ação: Uma atividade executada durante uma transição.

5.4. Diagrama de Sequência

É um diagrama de interação que enfatiza a ordem temporal das mensagens trocadas entre objetos.

- Eixo Vertical: Representa o tempo (passa de cima para baixo).
- Eixo Horizontal: Representa os objetos participantes.
- Linha de Vida (Lifeline): Uma linha tracejada vertical que representa a existência de um objeto ao longo do tempo.
- Mensagem: Uma comunicação entre objetos, representada por uma seta.

5.5. Diagrama de Comunicação

Anteriormente chamado de Diagrama de Colaboração, é um diagrama de interação que enfatiza a organização estrutural dos objetos e os links entre eles. Mostra as mesmas informações que um diagrama de sequência, mas foca nos relacionamentos em vez da cronologia. A ordem das mensagens é indicada por numeração sequencial.

5.6. Outros Diagramas Comportamentais

- Diagrama de Tempo: Foca em restrições de tempo, mostrando como os estados de um ou mais objetos mudam ao longo de uma linha do tempo e a duração em que permanecem em cada estado. É muito utilizado em sistemas de tempo real e por engenheiros de hardware.
- Diagrama de Interação Geral: Combina elementos do Diagrama de Atividades e do Diagrama de Sequência para fornecer uma visão geral do fluxo de controle entre interações complexas, mostrando como diferentes fragmentos de interação se encaixam.