

Programação Orientada a Objetos

Aula 03 – Linguagem UML

Hugo Marcondes

Departamento Acadêmico de Eletrônica
DAELN

hugo.marcondes@ifsc.edu.br



UML – Unified Modeling Language

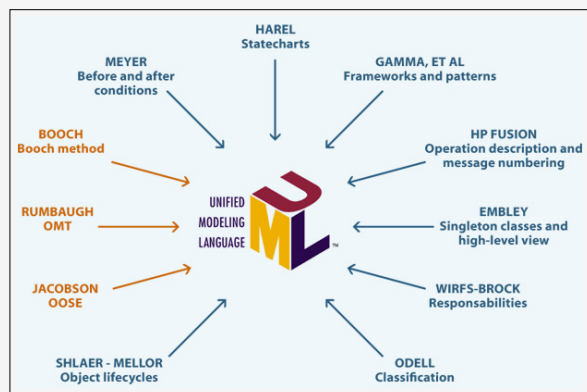
- A UML (Unified Modeling Language) é uma notação para descrição de sistemas orientados a objetos:
 - “The Unified Modeling Language for Object-Oriented Development” de Grady Booch, James Rumbaugh e Ivar Jacobson
- Baseia-se na experiência dos principais autores dos 3 principais métodos orientação a objetos
- Padronizada pela OMG (Object Management Group) em 1997



Notes

Notes

Origens da UML



Notes

Uso da UML

- A linguagem UML pode ser utilizada em diversas situações
- Esboço e discussão sobre a estrutura de um sistema
 - Melhor entendimento na análise
 - Compreensão do que se está projetando
- Documentação do projeto / sistema
 - Base para a codificação do sistema e elaboração de testes de funcionalidades
- Documentação de estruturas já existentes
 - Engenharia reversa



Notes

- **Diagramas Estruturais**
 - Descrição estática de estruturas de um sistema
 - classes, atributos, operações e relacionamentos
 - Diagrama de Classe, Componentes, Pacotes
- **Diagramas Comportamentais**
 - Detalham o funcionamento (comportamento)
 - Diagrama de Casos de Uso, Atividades, Transição de Estados
- **Diagramas de Interação**
 - Subgrupo dos diagramas comportamentais
 - Interações entre objetos de uma aplicação
 - Diagramas de Sequência, Interatividade, Colaboração e Tempo

Diagramas Estruturais

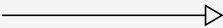
- Diagrama de **Classes**
 - Conjunto de classes, interfaces e colaborações e seus relacionamentos
 - **Diagrama mais comum!**
 - Ilustram a visão estática do sistema
- Diagrama de **Componentes**
 - Partes internas, os **conectores** e **portas** que implementam um componente
- Diagrama de **Objetos**
 - Conjunto de objetos e seus relacionamentos
 - Visão estática do sistema, contundo considerando casos reais (objetos instanciados)
- Outros diagramas: **estrutura composta, artefatos, implantação**

Diagrama de Classe

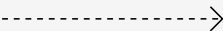
- Cada classe é representada por um retângulo dividido em três partes
 - Nome
 - Atributos (Estado)
 - Formato: visibilidade nome: tipo
 - Exemplo: **- nome: string**
 - Operações (Comportamento)
 - Formato: visibilidade nome(lista de argumentos): tipo de retorno
 - Exemplo: **+ calcularMedia(nota1: float, nota2: float): float**
- Modificadores são utilizados para indicar a visibilidade dos atributos e operações
 - **‘+’**: visibilidade Pública
 - **‘#’**: visibilidade Protegida
 - **‘-’**: visibilidade Privada
- Por padrão, atributos são privados e operações são públicas

Diagrama de Classe — Relações de Classes

- **Herança**: linha sólida com triângulo não preenchido na classe pai



- **Dependência**: linha pontilhada com seta



Notes

Notes

Notes

Notes

- Tipos de Relações
 - **Associação**: linha sólida entre classes (seta se direcional)
 - **Agregação**: linha sólida com losango vazio na classe “todo”
 - **Composição**: linha sólida com losango preenchido na classe “todo”
- Multiplicidade
 - direta: expressada pelo numeral
 - intervalo: expresso por dois números separados por ‘..’
 - *: expressa qualquer quantidade
 - 0: expressa zero — associação pode não existir
- Papéis
 - Expressão do papel desempenhado por um objeto em uma associação



Diagrama de Classe

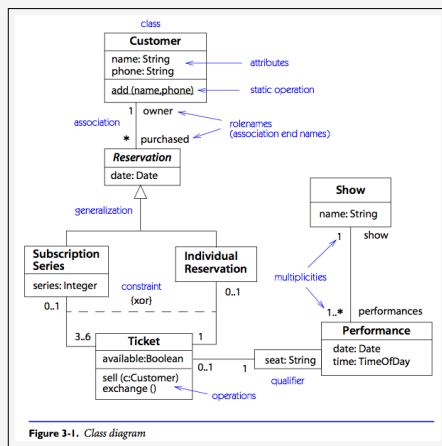


Figure 3-1. Class diagram



Diagrama de Componente

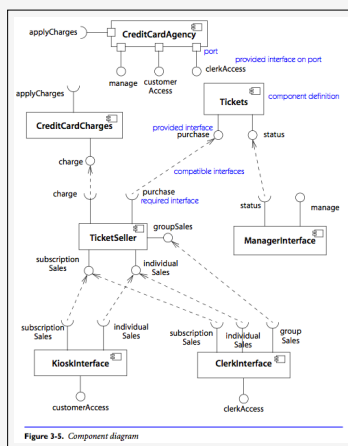


Figure 3-5. Component diagram



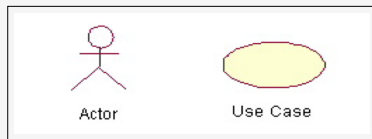
Diagramas Comportamentais

- Diagrama de **Casos de Uso**
 - Apresentação de funcionalidades e características do sistema
 - Relacionamento entre o sistema e os usuários/entidades
- Diagrama de **Atividades**
 - Contempla as diversas tarefas desempenhadas na execução de uma atividade, sendo utilizado geralmente na representação de processos dentro de uma empresa/organização.
- Diagrama de **Transição de Estados**
 - Detalha os diferentes estados pelos quais pode passar um objeto, tomando por base a execução de um processo dentro do sistema que se está considerando.



Diagramas de Caso de Uso

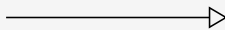
- **Atores:** Um papel que um usuário “interpreta” em relação ao sistema, incluindo pessoas reais ou como outros sistemas (ex. um robô, um sistema externo que utiliza uma informação do sistema modelado)
- **Caso de Uso:** Um conjunto de cenários descrevendo a interação entre um usuário e o sistema, incluindo cenários alternativos



Notes

Diagramas de Caso de Uso

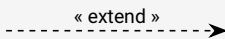
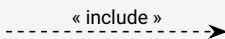
- **Associação:** comunicação entre um ator e um caso de uso
 - Representado por uma linha sólida
- **Generalização:** relação entre um caso de uso genérico e um caso especial de uso (especificação de alternativas)
 - Representado por uma linha com uma flecha triangular apontando para o caso de uso “pai”.



Notes

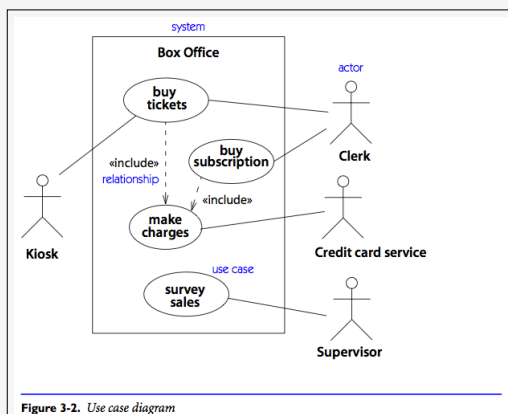
Diagramas de Caso de Uso

- **Inclusão:** Uma linha pontilhada rotulada com “«include»” iniciando no caso de uso “base” e terminando com uma flecha apontando para o caso de uso incluído. Inclusão pode ser utilizado para refatorar partes de especificação comum a diversos casos de uso.
- **Extensão:** Uma linha pontilhada rotulada com “«extend»” com uma flecha apontando para o caso de uso “base”. Serve para determinar pontos de extensão do caso de uso (execução não obrigatória)



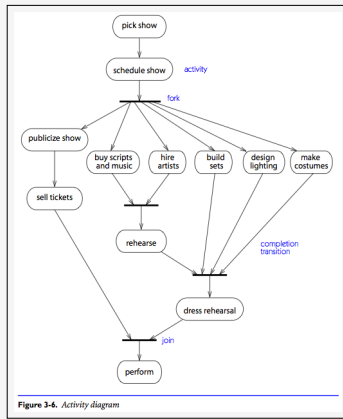
Notes

Diagrama de Casos de Uso



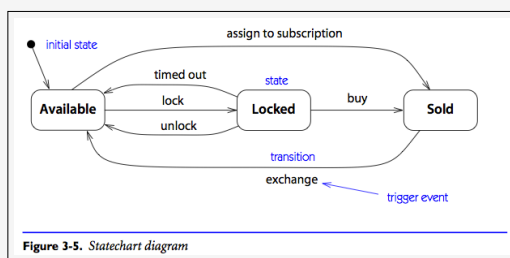
Notes

Diagrama de Atividades



Notes

Diagrama de Transição de Estados



Notes

Diagramas de Interação

■ Diagrama de Sequência

- Interações entre diferentes objetos na execução de uma operação
- Ordem em que tais ações acontecem em um intervalo de tempo

■ Diagrama de Colaboração ou Comunicação

- Similar a diagramas de sequência
- Não apresenta estrutura rígida, geralmente derivado do diagrama de objetos



Notes

Diagrama de Sequência

- Representa uma linha de tempo
- Começa no topo e desce gradualmente
 - Marca a sequência de interações
- Cada objeto tem uma coluna
- Mensagens trocadas entre eles são representadas por setas

■ Notações

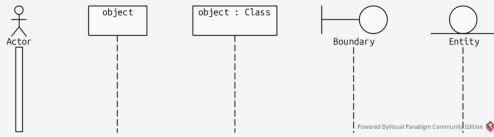
- Notação de linha de vida
- Barras de Ativação
- Setas de Mensagem
- Fragmentos de sequência
 - alternativas (alt)
 - opcionais (opt)
 - loops (loop)
 - referência (ref)



Notes

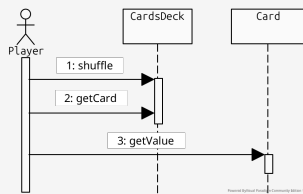
Notação linha de vida

- Um diagrama de sequência é composto por várias dessas notações
- Devem ser organizadas horizontalmente na parte superior do diagrama
- Duas notações de linha de vida não devem se sobrepor
- Eles representam os diferentes objetos ou partes que interagem entre si no sistema durante a sequência
- Podem representar entidades específicas (Atores, Boundaries, etc)



Barras de Ativação

- A barra de ativação é a caixa colocada na linha de vida.
- Indica que o objeto está ativo
- O comprimento da barra indica a duração da ativação (relativa)
- A interação entre dois objetos ocorre quando um objeto envia uma mensagem para outro
- A barra de ativação do objeto que envia a mensagem e do objeto que recebe a mensagem indica que ambos estão ativos durante a troca da mensagem

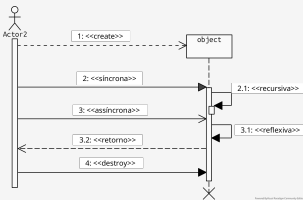


Notes

Notes

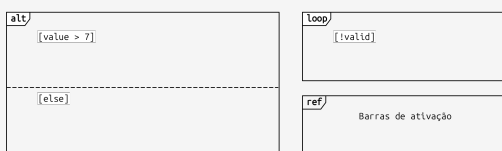
Setas de Mensagem

- Uma mensagem pode fluir em qualquer direção
- Diferentes tipos de setas indicam propriedades das mensagens
 - Síncronas, Assíncronas e Reflexivas
 - Retorno
 - Construção / Destruição
 - Reflexiva
- Formato da mensagem (apena o nome é obrigatório)
 - atributo = nome_da_mensagem (argumentos): tipo_de_retorno



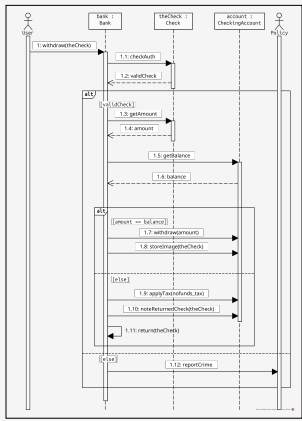
Fragmentos de sequência

- Um fragmento de sequência é representado como uma caixa que emoldura uma seção de interações entre objetos
- Utilizado para emoldurar interações complexas, como fluxos e loops alternativos
- Canto superior direito indica o tipo de fragmento



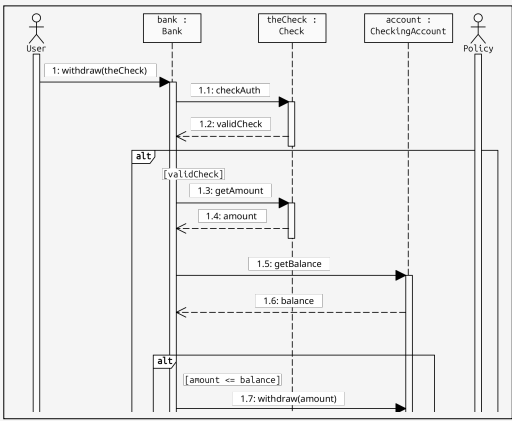
Notes

Diagrama de Sequência



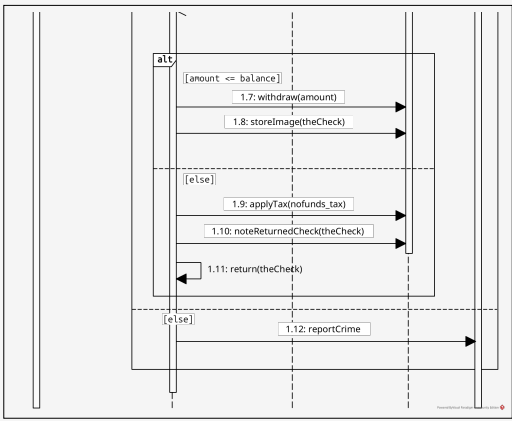
Notes

Diagrama de Sequência



Notes

Diagrama de Sequência



Notes

Diagrama de Colaboração

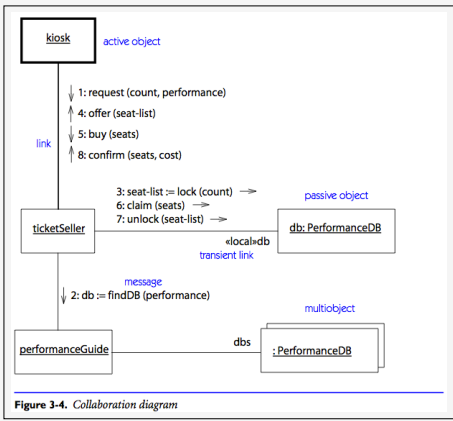


Figure 3-4. Collaboration diagram

Notes

Benefícios da UML

- A UML foca na representação visual de diferentes elementos e aspectos de um software
- Compreensão mais rápida, assim como abrangente, de componentes e funcionalidades que fazem parte de uma aplicação
 - **Simplifica** a apresentação dos relacionamentos complexos entre as diferentes partes que compõe um sistema complexo
- **Independente** de plataforma — melhor compreensão entre a equipe de um projeto complexo
- Excelente para a demonstração de conceitos de **orientação a objetos** (é a sua origem)
- Ênfase na **padronização** da linguagem, facilitando comunicação e transmissão de idéias



Cuidados no uso da UML

- Sincronização entre implementação e modelos UML
- Diagramas devem priorizar partes mais complexas ou críticas do sistema
 - Documentar funcionalidades e estruturas relativamente simples pode não agregar muito ao projeto!
- Cuidado com diagramas muito extensos!
 - Podem dificultar a compreensão
 - Solução? Diminuir escopo
 - Melhor entendimento



Ferramentas UML

- Gratuitas
 - ArgoUML — Java
 - Umbrella (KDE — Linux)
 - Papyrus (Eclipse) — <https://www.eclipse.org/papyrus/>
- Pagas
 - Visual Paradigm — <https://www.visual-paradigm.com/>
 - IBM Rational
 - Together
 - Poseidon

Unified Modeling Language tools -- Wikipedia¹

¹http://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools



Referências

- James Rumbaugh, Ivar Jacobson, and Grady Booch. 2004. **Unified Modeling Language Reference Manual, the (2nd Edition)**. Pearson Higher Education.
- **IBM Rational**²
- **Practical UML — A Hands-On Introduction for Developers**³
- **Tutorial do Diagrama de Sequência**⁴

²<http://www-306.ibm.com/software/rational/uml/>

³http://www.togethersoft.com/services/practical_guides/umlonlinecourse/

⁴<https://creately.com/blog/pt/diagrama/tutorial-do-diagrama-de-sequencia/>



Notes

Notes

Notes

Notes



Notes

Notes

Notes

Notes
