PSI 2591

PROJETO DE FORMATURA I

8ª Aula

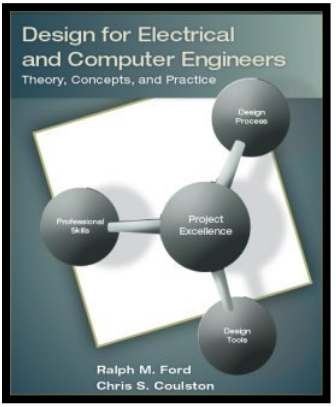
Modelos Comportamentais

2015

Elaboração

- Prof. Sergio Takeo Kofuji
- Prof. Marcelo K. Zuffo
- Prof. Antonio C. Seabra
- Dra. Ramona M. Straube

- Livro Texto:



Modelos Comportamentais Motivação:

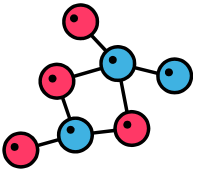
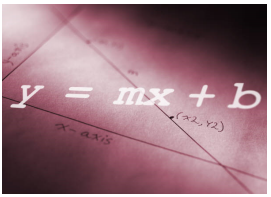
- Projeto funcional / Decomposição Funcional
  - Apropriado para sistemas orientados por funções: entradas, saídas e algumas transformações entre elas
- Existem outros comportamentos de sistemas que os projetistas precisam também compreender e explicar
  - Comportamento de Estado
  - Lógica e fluxo
  - Fluxo de dados
  - Bases de dados relacionais
  - …

Razões para estudar Modelos Comportamentais

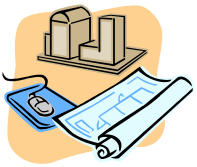
- Familiarizar-se com as seguintes ferramentas para descrever comportamentos de sistemas típicos de Eng. Elétrica e Computação:
  - diagrama de estados
  - fluxogramas
  - diagramas de fluxo de dados
  - diagramas de entidade-relação
  - linguagem de modelagem unificada
- Compreender a intenção e o poder de expressão dos diferentes modelos.
- Compreender os domínios aos quais cada modelo se aplica.
- Ser capaz de conduzir a análise e o projeto com modelos.
- Compreender que tipos de modelos escolher para cada problema

Como você entende “Modelos”?





Propriedades dos Modelos

Um bom modelo deve ser

- Abstrato: permite múltiplas implementações e é independente delas
- Não ambíguo: descreve claramente o comportamento esperado
- Permitir inovações: encoraja soluções alternativas
- Padronizado: segue procedimentos e padrões estabelecidos
- Uma boa ferramenta de Comunicação: facilita comunicação
- Modificável: modificações devem ser relativamente fáceis de executar
- Remover detalhes desnecessários & destacar as características importantes: facilita o entendimento e deixa os detalhes para a implement.
- Dividir o Sistema em subsistemas: explora o conceito de hierarquia
- Substituir sequências de ações por uma ação única: permite a compreensão do todo
- Auxiliar na verificação: permite mostrar que o projeto segue os requisitos de Engenharia
- Auxiliar na validação: permite mostrar que as necessidades do usuário estão contempladas. Deve facilitar a discussão com os clientes

Definições

Modelo: um modelo parte de uma abstração top-down que captura os aspectos essenciais do projeto em um primeiro nível e vai detalhando-a paulatinamente. Considera-se que esse tipo de abstração é um modelo quando é construída através de uma linguagem padronizada e reconhecida pelos interessados. Em outras palavras, um modelo é uma representação padronizada de um Sistema, processo, ou objeto que captura seus detalhes essenciais sem especificar como ele será implementado fisicamente.

Linguagem de modelagem: uma linguagem de modelagem é um conjunto organizado de representações que expressa as características do projeto. Ela não precise ser uma linguagem na forma de letras e palavras, pode ser exemplo ser contituída de representações gráficas, como um esquemático de circuito ou um diagram de blocos de um programa.

Definições...

Tipos de objeto: Um modelo necessita de objetos (símbolos) capazes de encapsular o conjunto de components necessário para realizar o Sistema. A fim de capturer a dependência entre objetos, os modelos tipicamente possuem tipos de relacionamentos. Exemplos:

Intenção: Modelos possuem uma intenção, que é a class e de comportamento que ele descreve. Por exemplo a intenção de um esquemático de circuito é diferente da intenção do esquemático de um jogo de futebol.
Alerta: é possível escolher o modelo errado para analisar um Sistema em particular.

Tipos de Modelos que analisaremos

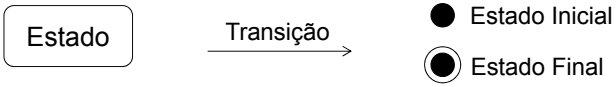
- Diagrama de estados (State Diagrams)
- Fluxogramas (Flowcharts)
- Diagramas de fluxo de dados (Data Flow Diagrams)
- Diagramas de entidade-relação (Entity Relationship Diagrams)
- Linguagem de modelagem unificada (UML – Unified Modeling Language)

Diagramas de Estados
(State Diagrams)

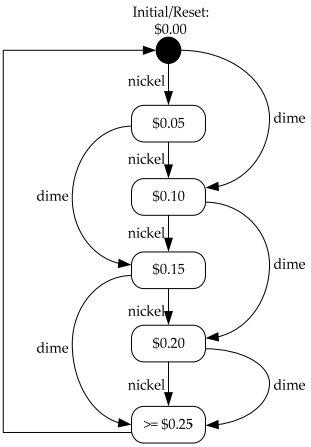
A intenção é? Descrever o comportamento de sistemas com memória. O estado de um Sistema representa o efeito global de todas as entradas anteriores no sistema. *Estado* é um sinônimo de *memória*.

Como você sabe se deve utilizá-lo? Para determinar se o sistema tem memoria, pergunte-se: "A mesma entrada pode produzir respostas diferentes?". Se sim, o sistema pode ser modelado a partir de um Diagrama de Estados.

Símbolos utilizados em diagramas de estados:



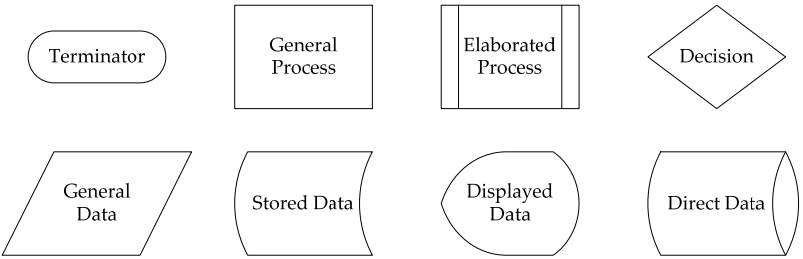
Exemplo: Máquina de Vendas



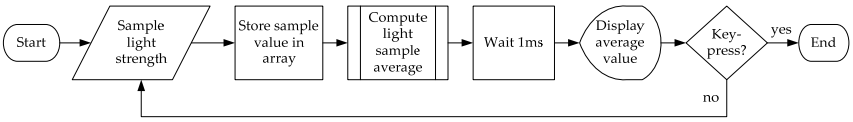
Fluxogramas
(Flowcharts)

A intenção é? Descrever visualmente um processo ou algoritmo, incluindo passos e controle. São considerados muito simples e ultrapassados, mas na verdade essas características são sua força. Por serem simples são adequados para diversos públicos, inclusive pra descrever processos de negócios.

Símbolos:



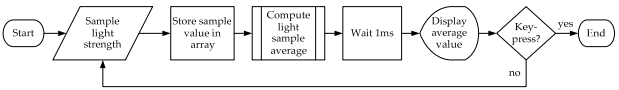
Exemplo: Sistema de Iluminação



- Você pode descrever o funcionamento do sistema a partir do fluxograma acima?
- Esta é a razão pela qual fluxogramas são intuitivos e elegantes

Exemplo: Sistema de Iluminação

- Mais Formalmente:

Módulo	Registrador de intensidade luminosa
Entradas	- Intensidade luminosa - Botão de parade: necessário caso o usuário deseja parar o registro
Saídas	- Terminal: Apresenta a intensidade luminosa media - Disco: Armazena o hostórico de intensidade luminosa
Funcionalidade	

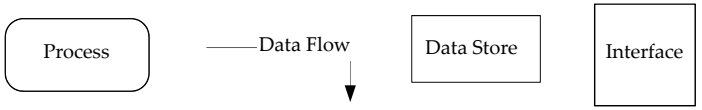
Diagramas de Fluxo de Dados
(Data Flow Diagrams - DFD)

A intenção é? A intenção de um DFD é modelar o processamento e fluxo de dados dentro de um sistema. É uma aboradgem orientada a funções, muito similar à decomposição funcional (aula anterior). A decomposição funcional que vimos anteriormente costuma ser muito próxima da implementação física, enquanto o DFD foca nos dados.

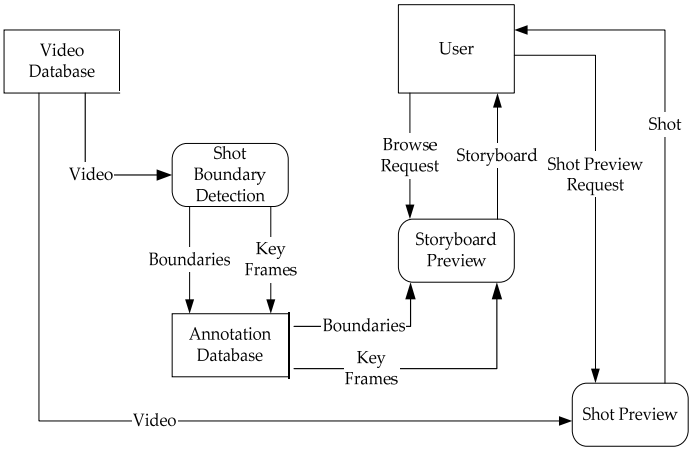
O DFD é fundamentalmente diferente de um fluxograma pois não encapsula o sequenciamento das informações e o controle (tomadas de decisão) e permite múltiplos processos sendo executados concomintantemente.

DFDs podem ter níveis de refinamento (nível 0, 1, ...)

Símbolos de um DFD:



Exemplo DFD
Sistema de Manipulação de Vídeo

Módulo	Sistema de Manipulação de Vídeo
Entradas	- Vídeo: video externo ao sistema que é adicionado à base de videos - Requisição de manipulação de video: Usuário solicita manipular determinado vídeo - Requisição de visualização de quadro: Usuário solicita ver um quadro(s) de um video
Saídas	- Storyboard: uma sequência de quadros (frames) que resume o vídeo - Vinheta: O video complete que corresponde às imagens (quadros) que estão no storyboard
Funcionalidade	

DFD: Tabela de Eventos

É importante ressaltar alguns pontos sobvre DFDs:

- São independentes da solução. Informações específicas sobre o fluxo de dados são definidas em uma linguagem formal conhecida como *dicionário de dados*.
- Pode haver processos ocorrendo concomintamente em um DFD. No exemplo do video, várias pessoas podem usar o sistema ao mesmo tempo. É claro que deve haver uma **tabela de eventos** possíveis.

Event	Trigger	Process	Source
Annotate Video	New Video Arrival	Shot Boundary Detection	System
View Storyboard	Browse Request	Storyboard Preview	User
View Shot	Shot Preview Request	Shot Preview	User

Diagramas Entidade-Relação
(Entity Relationship Diagrams - ERD)

- A intenção é? Catalogar um conjunto de objetos (entidades) relacionados, seus atributos e as relações entre eles. As entidades e seus relacionamentos são coisas reais e distintas que possuem características que precisam ser capturadas. O projeto de uma base de dados começa por descrever as entidades, seus atributos e as relações entre as entidades de um ERD. As entidades precisam estar ligadas umas às outras. Por exemplo, uma lista de estudantes e uma lista de cursos tem pouca utilidade, mas se ligarmos as duas entidades, podemos extrair diversas informações.

Diagramas Entidade-Relação
(Entity Relationship Diagrams - ERD)

- Em uma linguagem de modelagem ERD temos:
  - Entidades: Em geral na forma de objetos tangíveis, funções dentro de uma estrutura, unidades de uma organização, dispositivos, localizações geográficas, etc. Uma instância é uma manifestação particular de uma entidade (por ex., uma entidade pode ser `Estudante` enquanto uma instância pode ser `Carolina`.
  - Relações: São descritores de uma relação entre entidades
  - Attributos: São características utilizadas para diferenciar entre instâncias das entidades

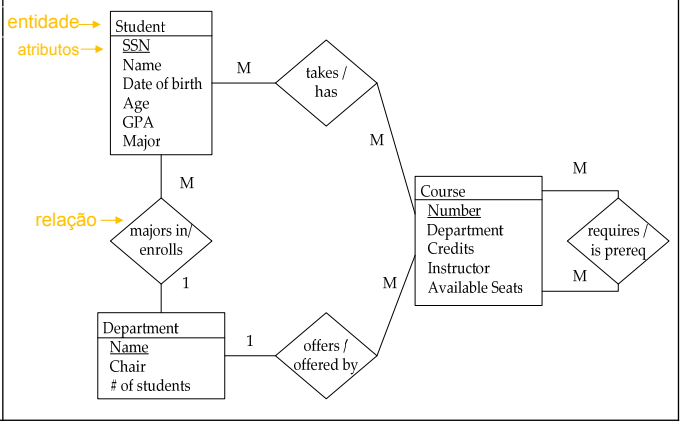
Exemplo: Base de Dados Escolar

- Tipicamente o processo se inicia entrevistado os usuários e identificando as entidades e seus atributos.
- O processo de construir o ERD se inicia determinando as relações entre as entidades. Isso pode ser feito, p.ex., criando-se uma matriz entidade-relação.

Exemplo: Base de Dados Escolar

entidade → / ↓	Estudante	Disciplina	Curso razão cardinal
Estudante	relação nenhuma	faz muitos M:M	se diploma em um M:1
Disciplina	tem muitos M:M	pode ser prerequisito/ necessitar de muitas (M:M)	é oferecida por um M:1
Curso	matricula muitos 1:M	oferece muitas (1:M)	

Exemplo ERD
Base de Dados de Graduação

Módulo	Base de Dados de Graduação
Entradas	- Estudantes: Dados sobre estudantes - Departamentos: Dados sobre departamentos - Cursos: Dados sobre cursos
Saídas	- Relações entre departamentos e alunos matriculados - Relações entre estudantes e as disciplinas que eles fazem - Relações entre os departamentos e os cursos que eles oferecem - Relações entre disciplinas e seus cursos pré-requisitos
Funcionalidade	

Tipos de Modelos que estamos analisando

- Diagrama de estados (State Diagrams)
- Fluxogramas (Flowcharts)
- Diagramas de fluxo de dados (Data Flow Diagrams)
- Diagramas de entidade-relação (Entity Relationship Diagrams)
- **Linguagem de modelagem unificada (UML – Unified Modeling Language)**

Linguagem Unificada de Modelagem (UML)

- Criada para o projeto de software orientado por objetos
- Possui características interessantes para outras aplicações em EE&C
- Possui 6 visões diferenciadas de sistemas (unificadas!):
  - Estática (Static View)
  - Caso de Uso (Use-Case View)
  - Máquina de Estados (State Machine View)
  - Atividades (Activity View)
  - Interações (Interaction View)
  - Física (Physical View)
- Vamos apresentar apenas uma visão geral. Para isso vamos estudar a UML aplicando-a diretamente a um exemplo
- Detalhes do UML em: http://www.uml.org/ e http://pt.wikipedia.org/wiki/UML

UML aplicado a um exemplo: A v-Doces (Doces virtuais)

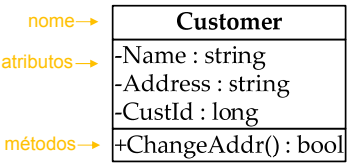
- Aplicação popular: pedidos web de doces incluindo a entrega em domicílio
- Chamaremos o system de “v-Doces”
- O usuário recebe um leitor de códigos de barras conectado ao seu computador, aonde é instalado um sw
- Quando o usuário fica sem estoque de um item ele passa o scanner no código de barras (UPC) daquele item e um pedido é armazenado imediatamente no computador
- Os pedidos e entregas podem ser organizados e enviados pela web, sendo que as entregas são feitas no horário que o cliente estipular

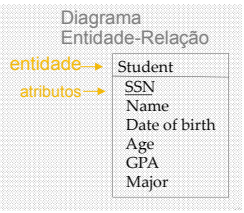
Projetos Orientados por Objetos

- Projetos orientados a objetos: Diferentes do projeto de softwares funcionais pois enfatiza objetos como dados (atributos) e métodos (funções) que podem agir sobre os dados.
- Um objeto representa uma instância particular de uma classe, que define os atributos e métodos.
- Um objeto encapsula essa informação que estará disponível apenas a esses objetos: Outros objetos não podem mudra o estado de outro objeto a não ser que tenham permissão para tal.
- Isso melhora a confiabilidade e a acessibilidade de sistemas de software pois mudanças internas de uma classe não são visíveis fora dessa classe

UML: Visão Estática

- A intenção é? Mostrar as Classes em um sistema e as relações entre elas.
- Caracterizada por um *diagrama de classes* *(class diagrams)*





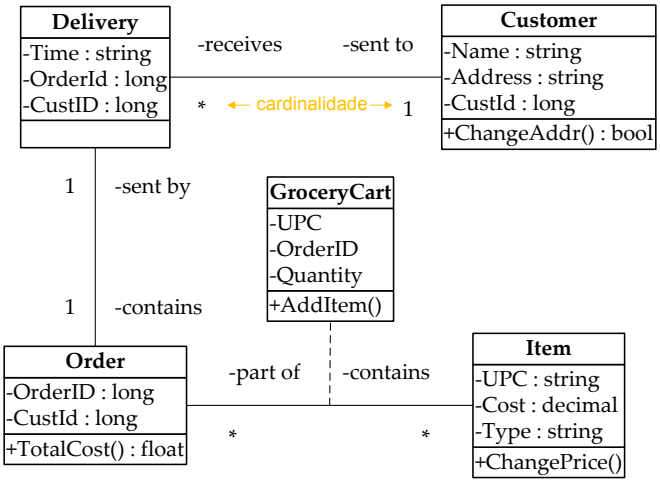
- Permite diversos níveis de segurança

UML: Visão Estática

- Classes interagem entre si por relações (relationships)
  - P. ex., se uma classe pertence a outra classe (um tipo de relação) então a subclasse (subclass) herda os atributos e métodos da superclasse (superclass)
- Em UML existem diversos tipos possíveis de relações, tais como:
  - Generalização: quando uma classe é subclasse de outra
  - Composição: quando uma classe é composta de membros de outra
  - Associação: quando classes precisam trocar mensagens entre si

UML: Visão Estática e Diagrama de Classes

- Relações são feitas por linhas ligando duas classes: As pontas das linhas tem formatos diversos dependendo do tipo de relação. Como em ERD as relações tem cardinalidade.

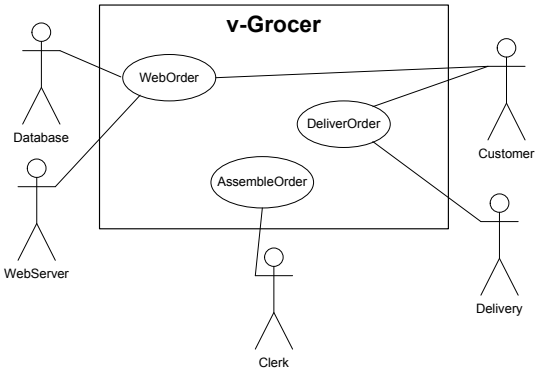


UML: Visão Caso de Uso (Use-Case)

- A intenção é? Capturar o comportamento global do sistema do ponto de vista do usuário e descrever casos onde o sistema será utilizado
- Caracterizado pelo diagrama de caso de uso (Use-Case Diagram)
- O diagrama de caso de uso utiliza apenas dois símbolos: atores (actors) e casos de uso (use-case)
  - **Atores** são pessoas ou sistemas idealizados que interagem com o sistema
  - **Casos de uso** são desenhados como ovais em no diagrama e são uma situação particular quando os atores utilizam o sistema
- Em geral essa sequência de ações representa a funcionalidade em alto nível do sistema

UML: Visão Caso de Uso e Diagrama de caso de uso

- Do diagrama é claro que Database, Webserver e Customer interagem quando é criada uma Weborder
- Dado seu alto nível de abstração, pode ser facilmente incorporada em uma variedade de Projetos de EE&C.
- São também fáceis de entender em apresentações e para audiências leigas



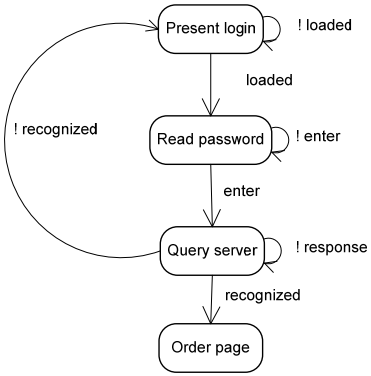
UML: Visão Caso de Uso e Descrição de caso de uso

- Casos de uso são frequentemente descritos na forma de tabelas

Use-Case	`WebOrder`
Actors	`Customer`, `Database`, and `WebServer`
Description	This use-case occurs when a customer submits an order via the `WebServer`. If it is a new customer, the `WebServer` prompts them to establish an account and their customer information is stored in the `Database` as a new entry. If they are an existing customer, they have the opportunity to update their personal information.
Stimulus	Customer order via the `GroceryCart`.
Response	Verify payment, availability of order items, and if successful trigger the `AssembleOrder` use-case.

UML: Visão de Máquina de Estados (State Machine)

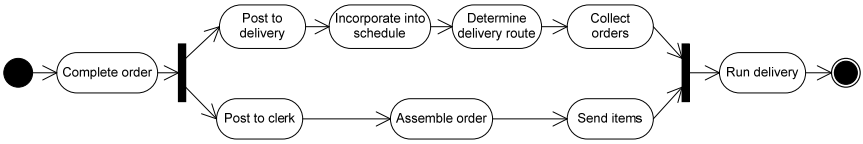
- Caracterizada por um Diagramas de Estados (slide 10)
- Novamente, a intenção é um sistema com memória



UML: Visão de Atividade (Activity View)

- A intenção é? Descrever a sequência de atividades necessária para completer a tarefa
- Caracterizada por um Diagrama de Atividades (que descreve a sequência de atividades para realizar uma tarefa)
- Em UML as tarefas são os casos de uso individuais descritos no Diagrama de casos de uso
- Como mais de um ator pode estar envolvido em uma tarefa, o diagrama de atividades pode expressar a simultaneidade das tarefas
- É composto de estados, transições, bifurcações e juntas

UML: Visão de Atividade
Explo do v-Doces



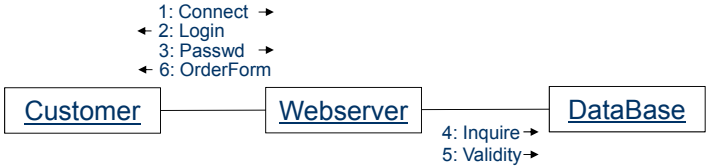
- Visão clara das atividades a serem executadas
  - Há uma bifurcação para processos que correm simultaneamente
  - Um desses processos é para preparar o pedido e outro para definir a rota
  - Quando completados, há uma junção e em seguida é feita a entrega

UML: Visão de Interação

- A intenção é? mostrar a interação entre objetos (quando eles devem cooperar para fazer algo)
- Em um sistema orientado por objetos:
  - as tarefas (que são usalmente casos de uso) são cumpridas passando mensagens entre objetos
  - A visão de interação mostra como as mensagens são trocadas, indicando em que ordem isso ocorre
- É caracterizada por **diagramas de colaborações** e de **sequência**

UML: Visão de Interações
Diagrama de Colaborações

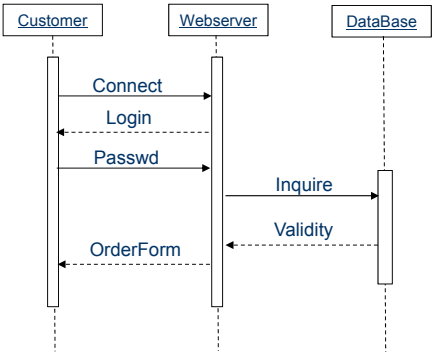
- Dois objetos colaboram entre si para produzir algum resultado
- O **Diagrama de colaborações** mostra a sequência de mensagens trocadas entre classes para completer uma tarefa
- As mensagens são desenhadas como flexas. Elas recebem rótulos com o nome da mensagem e sua posição numérica na sequência de mensagens



- O diagrama de colabrações é similar em forma a um diagrama de classes (slide 29), sendo que as diferenças nas relações são anotadas com as mensagens que são trocadas, auxiliando na compreensão e implementação dos métodos utilizados entre as classes.
- Para enfatizar a ordem em que as mensagens são trocadas pode-se utilizer um **Diagrama de Sequência**

UML: Visão de Interações
Diagrama de Sequência

- Um **Diagrama de Sequência** contém a mesma informação que o Diagrama de Colaborações
- Enquanto o Diagrama de Colaborações enfatiza os objetos que interagem para produzir um comportamento, o Diagrama de Sequência enfatiza a ordem das mensagens
- No diagrama de sequência:
  - As classes são listadas no topo
  - De cada classe sai uma linha vertical pontilhada representando a duração da classe
  - Quando um objeto requisite ou aguarda uma mensagem desenha -se um retângulo sobre a linha
  - A mensagem é desenhada como uma linha horizontal de envio/rece-bimento com o nome da mensagem

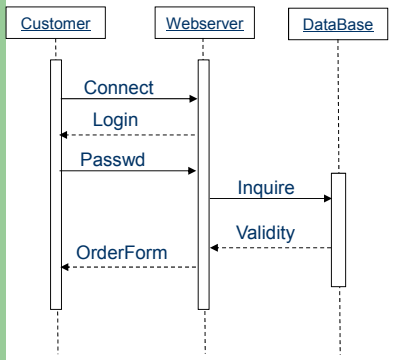


- Diagramas de atividade são largamente empregados em EE&C

UML: Visão de Interações
Explo: Diagrama de Sequência no Protocolo 3GPP

- Compare

UML



3GPP A GLOBAL INITIATIVE — The Mobile Broadband Standard

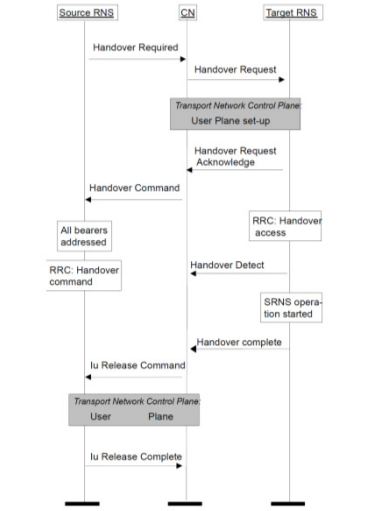


Figure 2. An example RANAP protocol message flow at Iu interface related to Inter RNS Hard Handover. A successful case.

UML: Visão Física

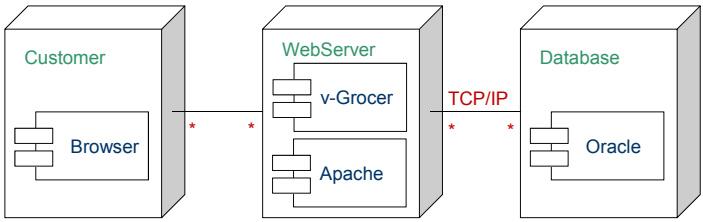
- A intenção é? Mostrar os components físicos do sistema e como as visões lógicas (anteriores) estão mapeadas nele.
- A visão física é caracterizada por
  - um Diagrama de componentes mostra os arquivos de software (retângulos) e as interrelações (linhas) que compõem o sistema
  - um Diagrama de instalação (deployment) mostra os componentes de hardware (cubos rotulados) e de Comunicação
- É uma forma de apresentação muito mais ampla que o próprio UML

Exemplo Doceria Virtual

Arquivos de software

Hardware

Comunicação



Aplicação em Projetos: Selecionando Modelos

Model	Intention
Functional Decomposition	To describe the input, output, and functional transformations applied to information (electrical signals, bits, energy, etc.) in a system. It is broad in application. This often provides a view that is close to the actual system implementation. See Chapter 5.
State Diagram	To describe the behavior of systems with memory. It is very flexible when it comes to application. It is often applied to digital design, but state diagrams can also be used to describe the high-level behavior of complex systems. The only prerequisite on their use is that the system has memory. See Section 6.2.
Flowchart	To describe a process or algorithm, including its steps and control. It is applicable to many problem domains from software to describing business practices. See Section 6.3.
Data Flow Diagram	To model the processing, transformation, and flow of data inside a system. It is typically supplemented by an event table describing all possible events and the resulting actions. Creating a DFD requires the designer to carefully think about the uses of the system and how the system is to react to external users and events. See Section 6.4.
Entity Relationship Diagram	To catalog a set of related objects (entities), their attributes, and the relationships between them. ERDs capture the entities and relationships of a portion of the world into a formal data model. The graphical language describes the attributes of the entities and the cardinality of the relationships. ERDs are formal enough to be unambiguously translated into a complete description of a database system. See Section 6.5.

Aplicação em Projetos: Selecionando Modelos

Model	Intention
The Unified Modeling Language. The intention of UML is to describe complex software systems. However, certain views are well suited to describing systems at a high level and are applicable to many domains. The process of viewing a system from the six perspectives listed below decreases the chances that crucial details of the design will be overlooked.	
Class Diagram	To describe classes and their relationship in an object-oriented software system. Class diagrams are primarily for software design and are not easily accessible to a nontechnical audience. See Section 6.6.1.
Use-Case Diagram	To capture the overall behavior of the system from the user's point of view and describe cases in which the system will be used. See Section 6.6.2.
State Machine	This is essentially the same as the state diagram, but is also a formal UML view. See Sections 6.2 and 6.6.3.
Activity Diagram	To describe the sequencing of processes required to complete a task. Composed of states, transitions, forks, and joins. Can show concurrent processes. See Section 6.6.4.
Interaction Diagram	To show the interaction and passing of messages between entities in a system. Characterized by both collaboration and sequence diagrams. See Section 6.6.5.
Physical Diagram	To describe the arrangement and connections of the physical components that constitute a system. In the case of UML, it is characterized by a component and deployment diagram. See Section 6.6.6.

Resumo

- Modelos são uma abstração do sistema
- Modelos podem ser considerados como uma especificação do projeto
- Modelos têm intençoes diferentes e descrevem comportamentos específicos
- Modelos devem encorajar inovação e possibilitar uma documentação clara