
Newton
Quem se prepara, não para.

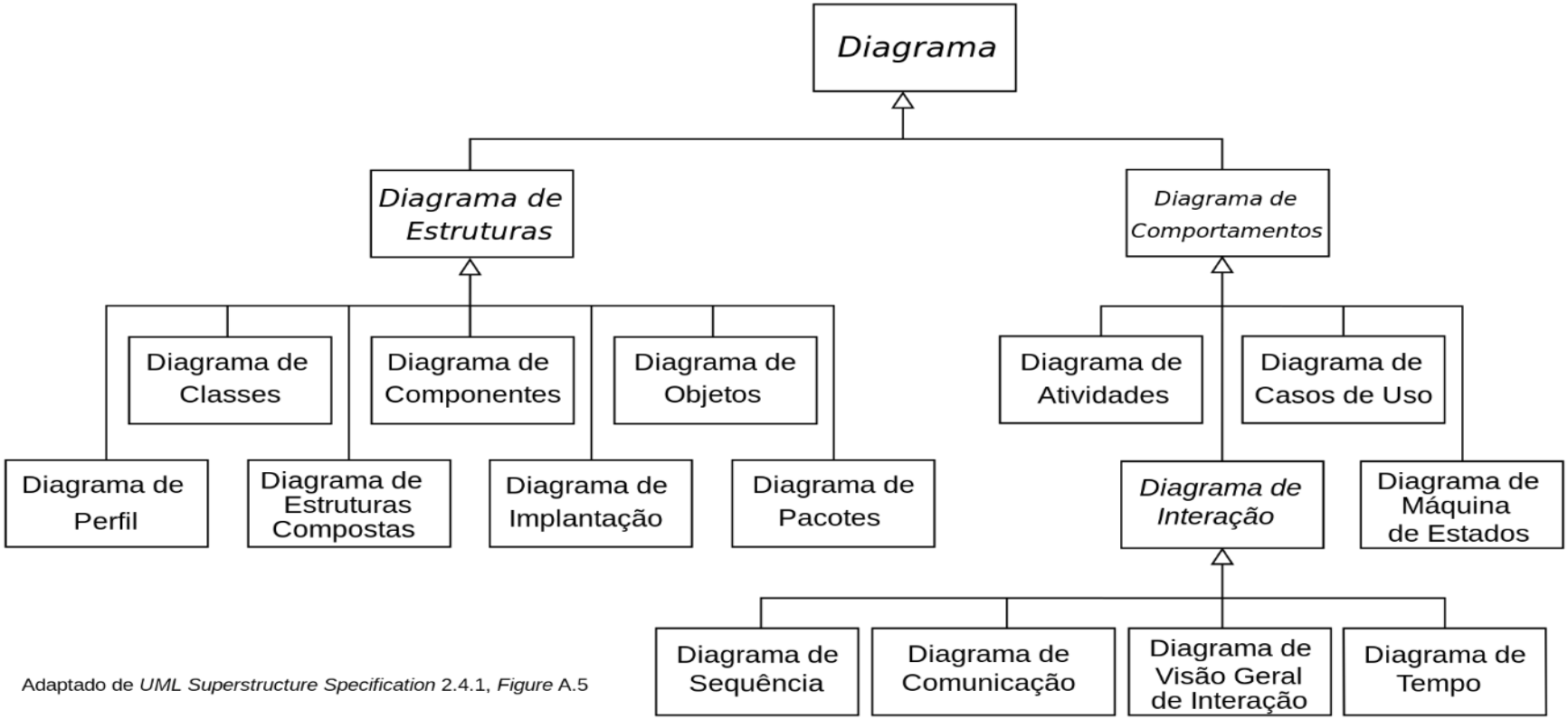
Análise de Sistemas

3º período

Professora: Michelle Hanne

Modelagem de Sistemas em UML

Newton
Quem se prepara, não para.



Adaptado de *UML Superstructure Specification* 2.4.1, *Figure* A.5

Diagrama de Máquina de Estado

Os **diagramas de máquina de estados** são uma técnica conhecida para descrever o comportamento de um sistema.

Nas estratégias orientadas a objetos, você desenha um diagrama de máquina de estados para uma única classe, para mostrar o comportamento do ciclo de vida de um único objeto.

Fonte: Martin, 2011.

Diagrama de Máquina de Estado

Newton
Quem se prepara, não para.

SUPERESTADOS

Vários estados compartilham transições e atividades internas comuns. Neste caso, é possível transformá-los em subestados e mover o comportamento compartilhado para um superestado.

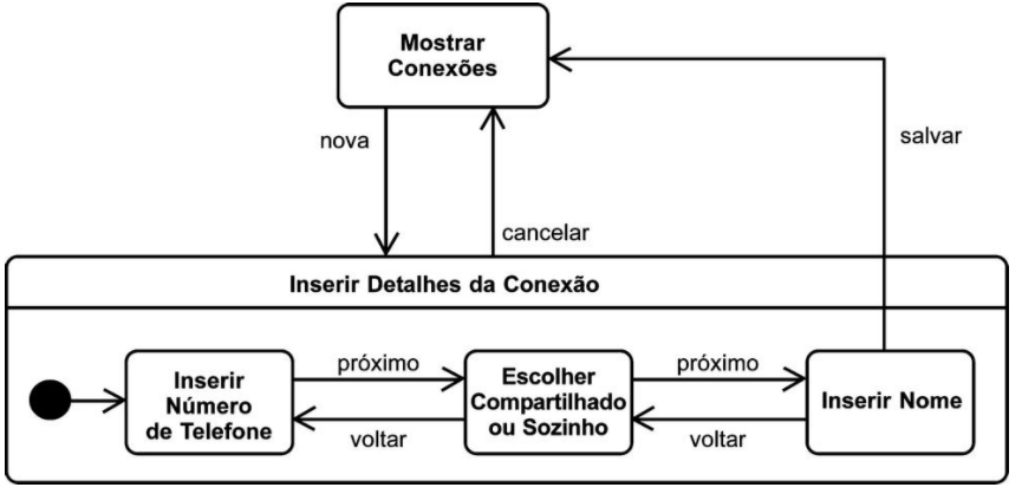


FIGURA 10.4 Superestado com subestados aninhados

Fonte: Martin, 2011.

Diagrama de Máquina de Estado

ESTADOS CONCORRENTES

Os estados podem ser divididos em vários diagramas de estados ortogonais que executam concorrentemente. A Figura abaixo mostra um despertador simples, que pode tocar CDs ou ligar o rádio e mostrar a hora atual ou a hora de alarme.

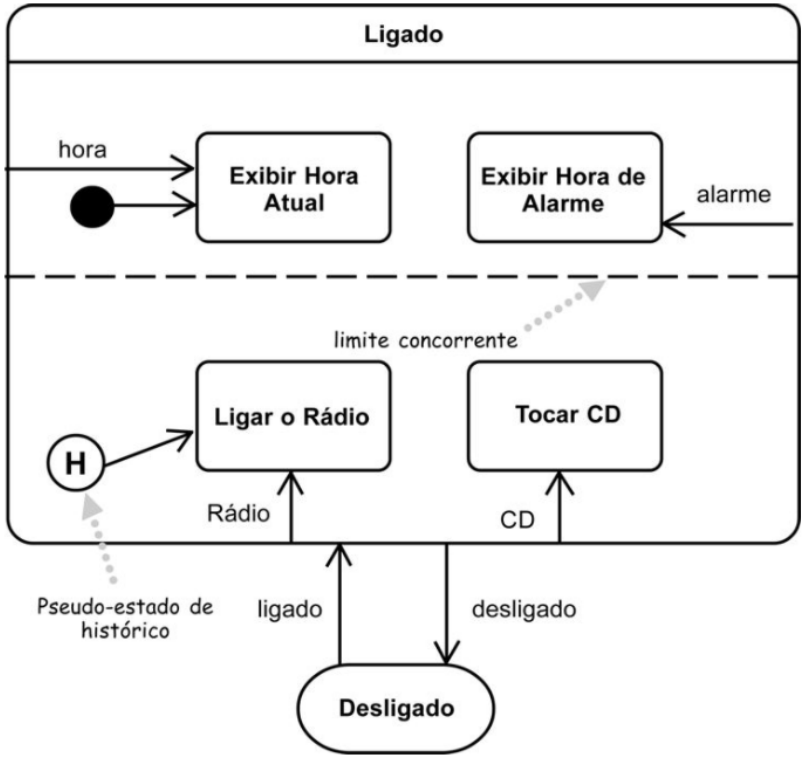


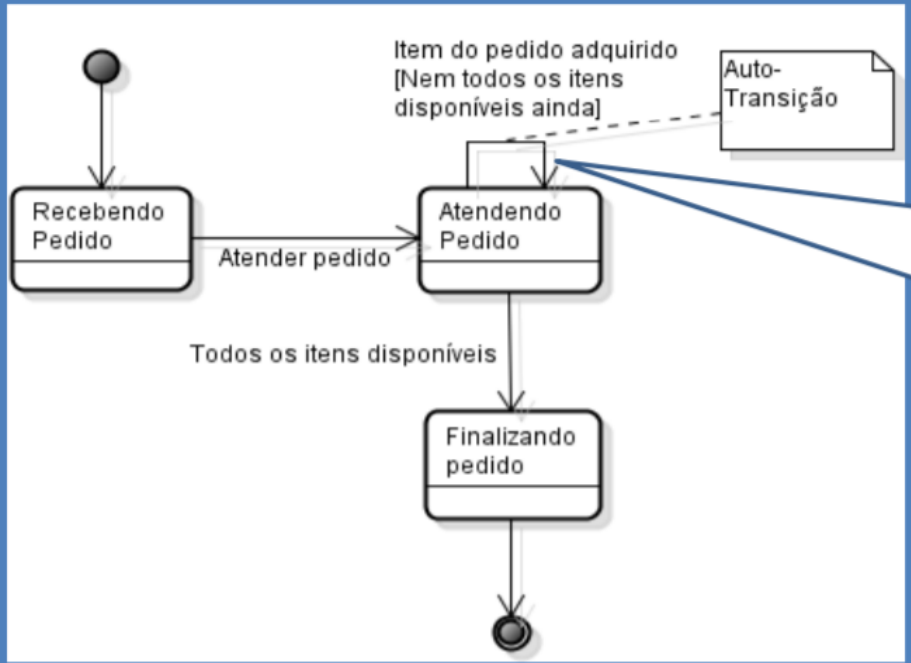
FIGURA 10.5 Estados concorrentes ortogonais.

Fonte: Martin, 2011.

Exemplo 1

Newton
Quem se prepara, não para.

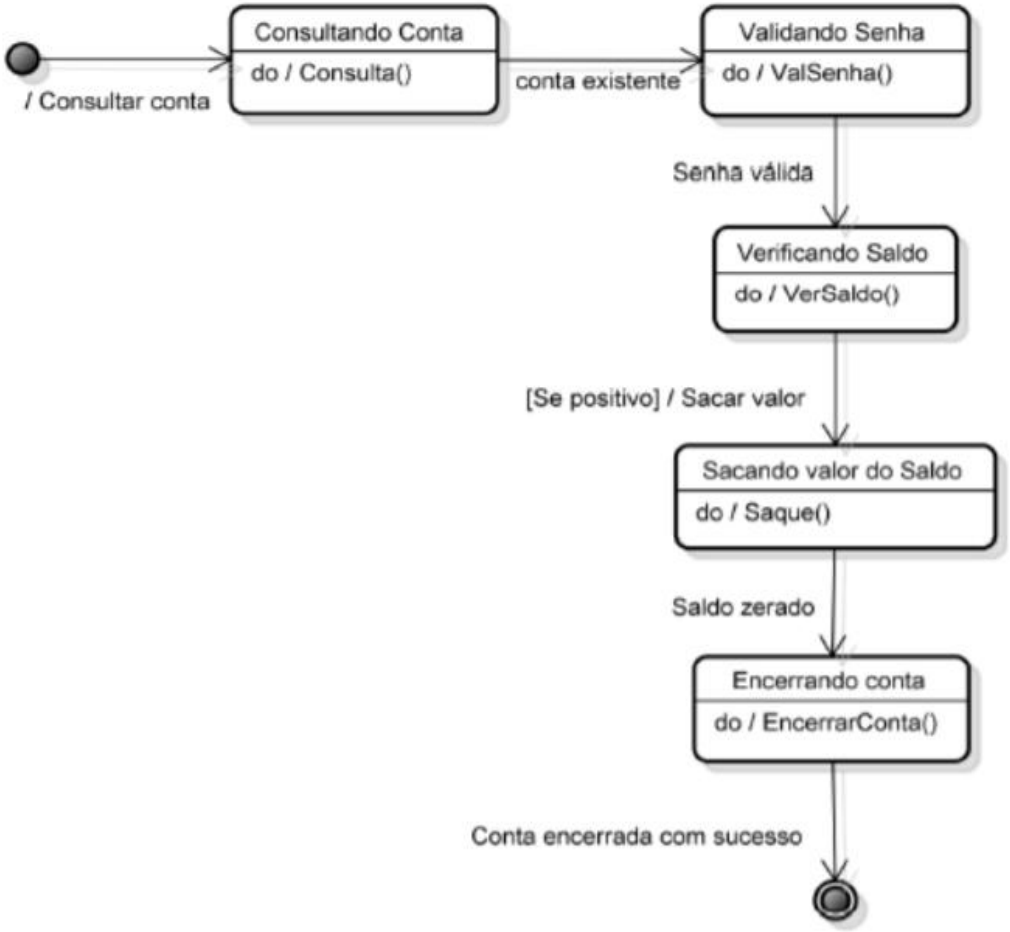
Exemplo:



Sempre que um item venha a ser adquirido, causará uma Auto-Transição no Estado do objeto pedido, mas o Estado permanecerá o mesmo, até que todos os itens tenham sido adquiridos.

Fonte: http://files.rosefib.webnode.com.br/200000331-077b4096ea/Aula09-Diagrama%20de%20Maquina%20de%20Estados.pdf

Exemplo 2



Newton
Quem se prepara, não para.

Fonte: http://files.rosefib.webnode.com.br/200000331-077b4096ea/Aula09-Diagrama%20de%20Maquina%20de%20Estados.pdf

Diagrama de Pacotes

Newton
Quem se prepara, não para.

Um diagrama de pacotes são frequentemente utilizados para representar **subsistemas, módulos ou parte da arquitetura de um sistema**. Seu elemento principal é o pacote.

Um pacote pode representar muitas coisas: uma camada, um módulo, um subsistema, um espaço de nomes, etc.

Um pacote UML pode agrupar qualquer coisa: classes, outros pacotes, casos de uso, etc.

O elemento pacote pode ser utilizado em diversos outros diagramas como o digrama de classes ou o diagrama de casos de uso.

Fonte: http://diatinf.ifrn.edu.br/prof/lib/exe/fetch.php?media=user:1542996:apoo_tads:diagrama_de_pacotes.pdf

Diagrama de Pacotes

Newton
Quem se prepara, não para.

As classes representam a forma básica de estruturação de um sistema orientado a objetos. Um **pacote** é uma construção de agrupamento que permite **agrupar seus elementos em unidades de nível mais alto.** Seu uso mais comum é o agrupamento de classes, mas lembre-se de que você também pode usar pacotes para todos os outros elementos da UML.

Fonte: Martin, 2011.

Diagrama de Pacotes

- Cada pacote representa um espaço de nomes, o que significa que toda classe deve ter um nome exclusivo dentro do pacote a que pertence.
- A UML permite que as classes de um pacote sejam públicas ou privadas. Uma classe pública faz parte da interface do pacote e pode ser usada por classes de outros pacotes; uma classe privada fica oculta.

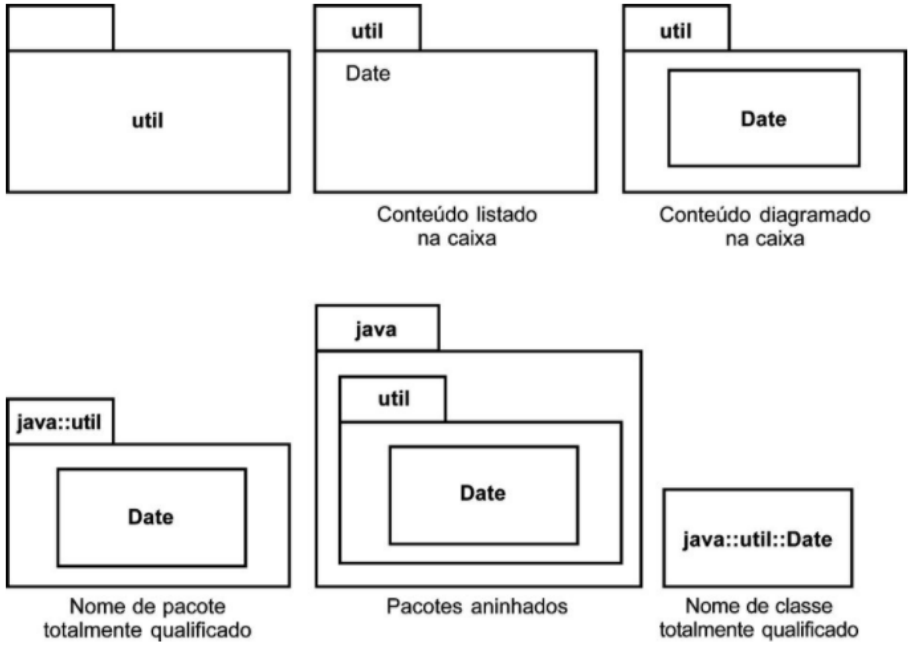


FIGURA 7.1 Maneiras de mostrar pacotes em diagramas.

Fonte: Martin, 2011.

Diagrama de Pacotes

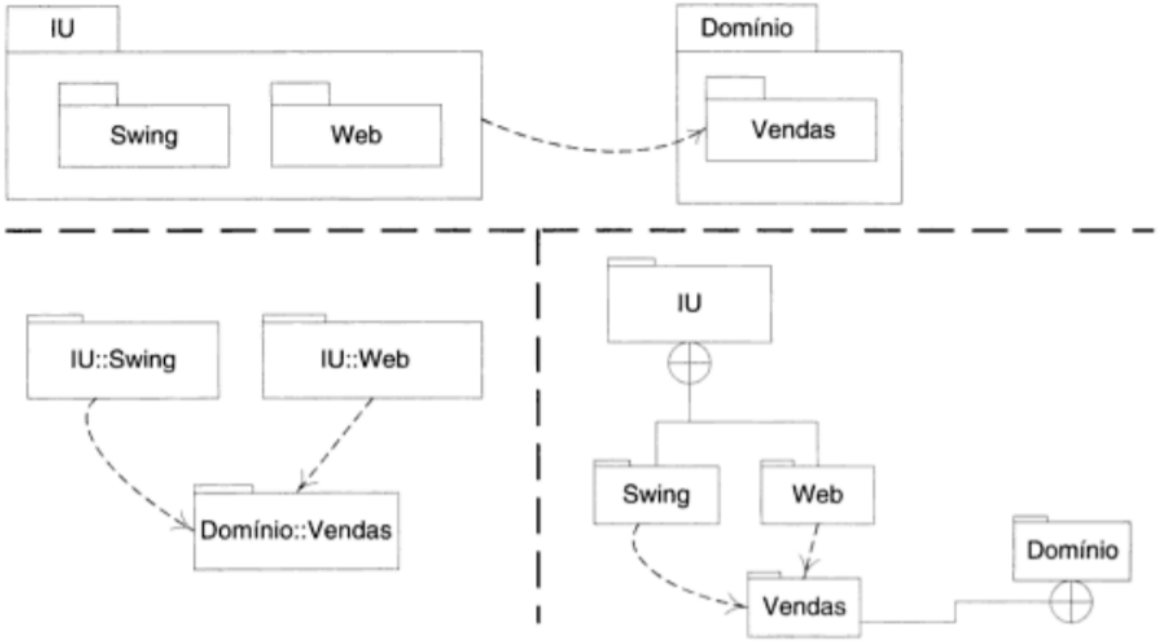


Figura 13.3 Abordagens alternativas UML para mostrar aninhamento de pacotes usando pacotes embutidos, nomes UML plenamente qualificados e o símbolo círculo-cruz.

A linha de dependência UML pode ser utilizada para mostrar as dependências entre pacotes ou entre elementos dos pacotes.

Fonte: Larman, 2007.

Diagrama de Pacotes - Dependências

Newton
Quem se prepara, não para.

Acesso: indica que um pacote requer assistência das funções de outro pacote.
Exemplo:

Fonte: https://www.lucidchart.com/pages/pt/diagrama-de-pacotes-uml

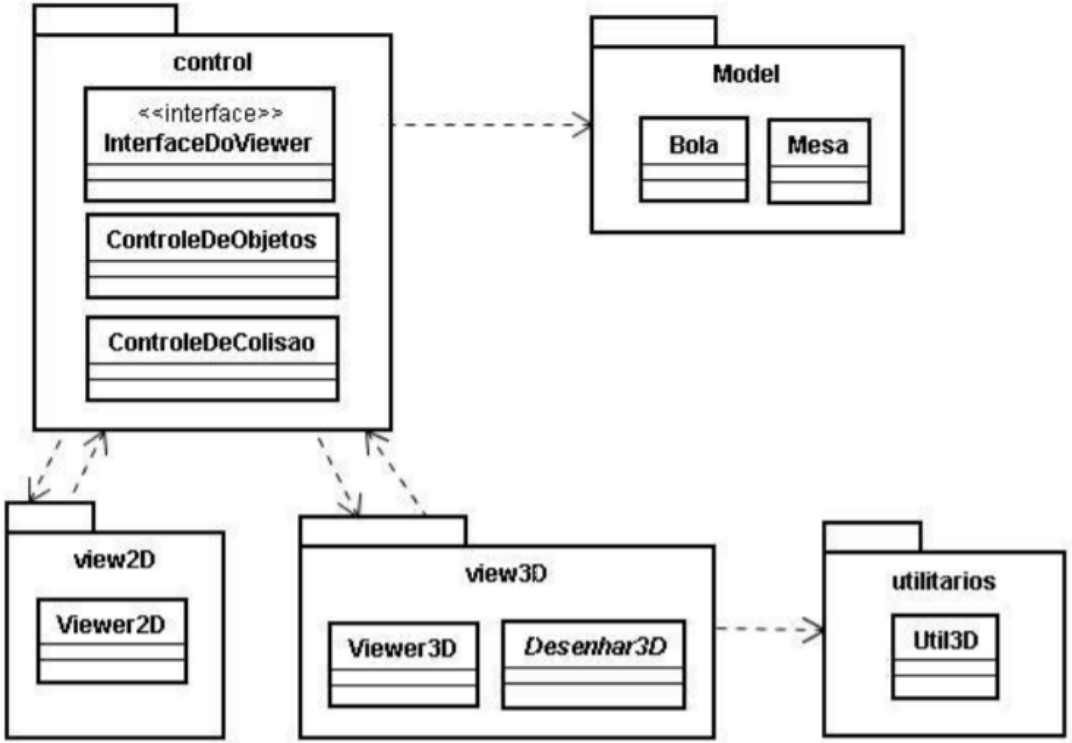
Diagrama de Pacotes - Dependências

Newton
Quem se prepara, não para.

Importação: indica que a funcionalidade foi importada de um pacote para outro.
Exemplo:

Fonte: https://www.lucidchart.com/pages/pt/diagrama-de-pacotes-uml

Diagrama de Pacotes - Exemplos



Fonte: Campos e Costa, 2017.

Diagrama de Pacotes - Exemplos

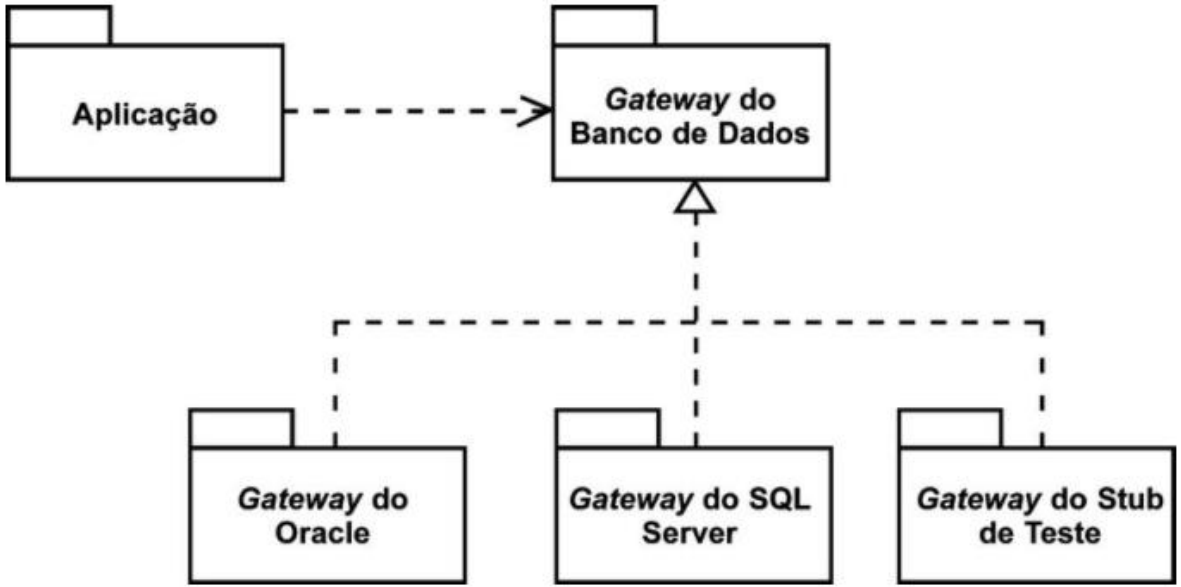


FIGURA 7.4 Um pacote implementado por outros pacotes

Fonte: Martin, 2011.

Diagrama de Pacotes - Exemplos

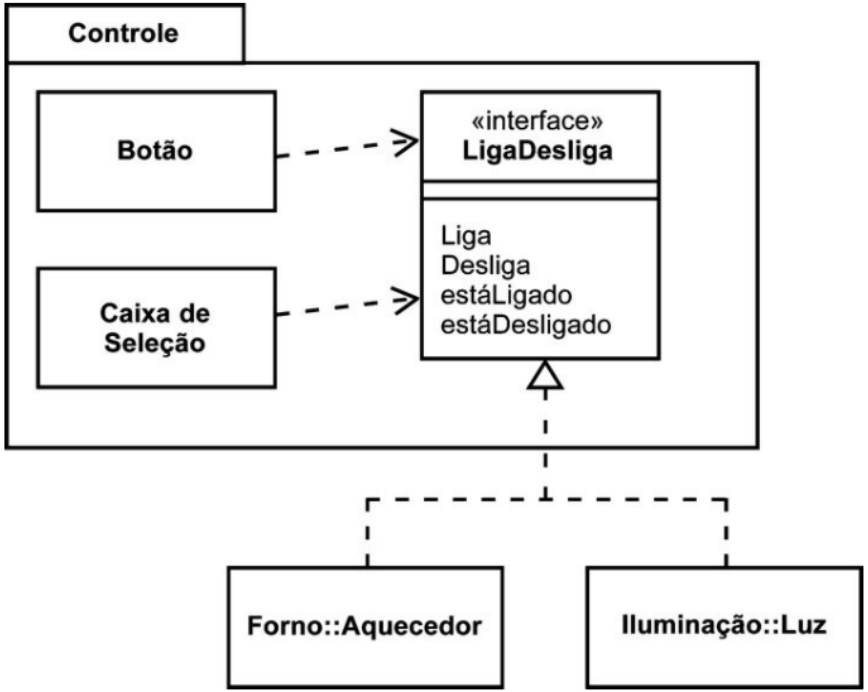


FIGURA 7.5 Definindo uma interface requerida em um pacote de cliente

Fonte: Martin, 2011.

Diagrama de Componentes

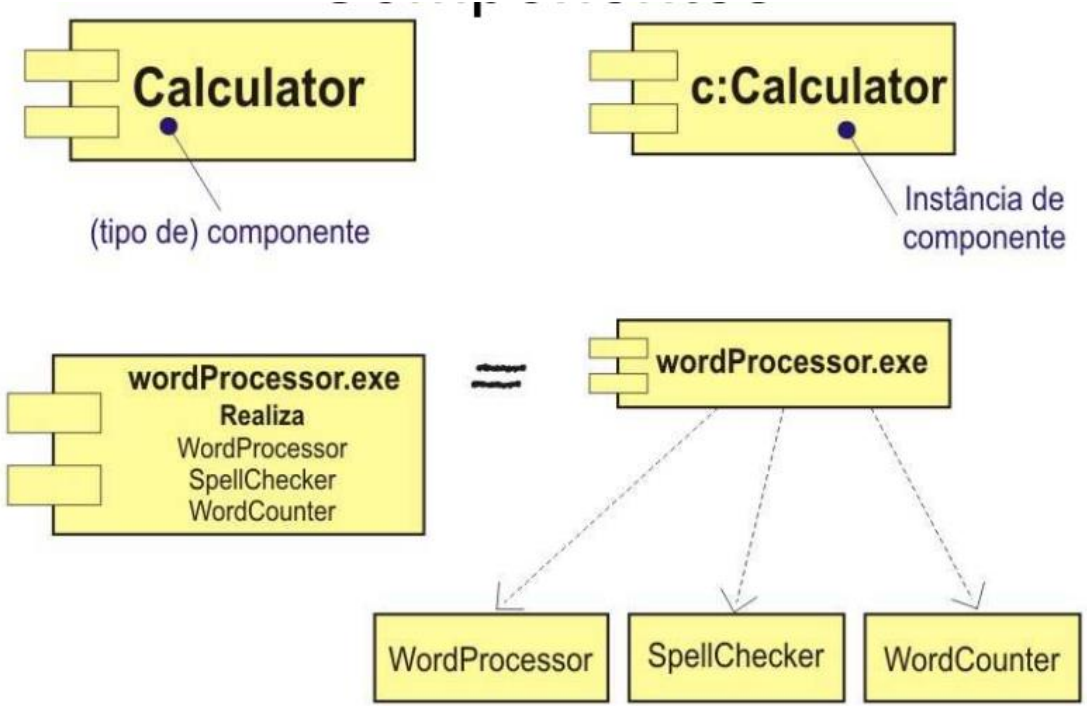
Newton
Quem se prepara, não para.

Definem-se pelo menos três tipos distintos de componentes:

- **Componentes de instalação:** constituem a base dos sistemas executáveis (e.g., DLL, executáveis, controles Active-X, classes Java).
- **Componentes de trabalho:** a partir dos quais são criados os componentes de instalação (e.g., arquivos com código fonte, arquivos de dados, documentos).
- **Componentes de execução:** criados como resultado da execução de um sistema (e.g., processos, threads, agentes de software).

Fonte http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ES_II_2013_1/UML_Aula3.pdf

SOA – Diagrama de Componentes



Fonte http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ES_II_2013_1/UML_Aula3.pdf

Diagrama de Componentes

Newton
Quem se prepara, não para.

Um **componente de software é uma parte física de um sistema**: existe de fato num determinado computador e não apenas na mente do analista, como acontece com o conceito de classe.

Adicionalmente, **um componente implementa uma ou mais classes**, as quais são representadas dentro do ícone **de componente ou com relações explícitas de dependência de implementação**.

A UML identifica os seguintes estereótipos para componentes:
«document»: denota um documento. –
«executable»: denota um programa que possa ser executado num nó.
«file»: denota um documento contendo código fonte ou dados.
«library»: denota uma biblioteca dinâmica ou estática.
«table»: denota uma tabela de uma base de dados.

Fonte http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ES_II_2013_1/UML_Aula3.pdf

Diagrama de Componente

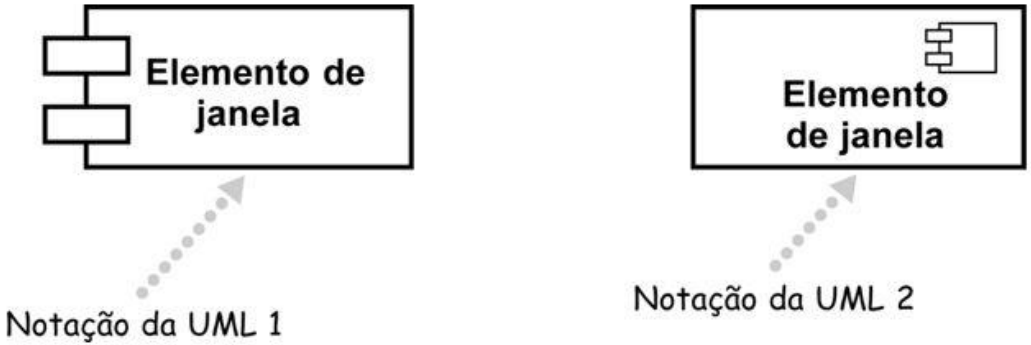


FIGURA 14.1 Notação para componentes.

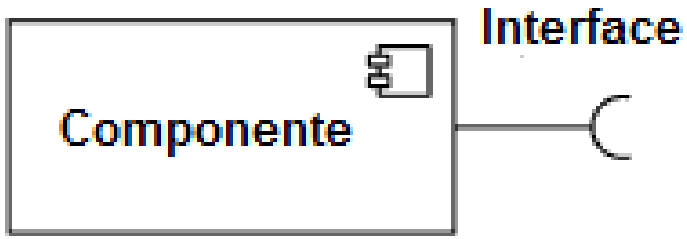
Fonte: Martin, 2011.

Diagrama de Componente

Newton
Quem se prepara, não para.

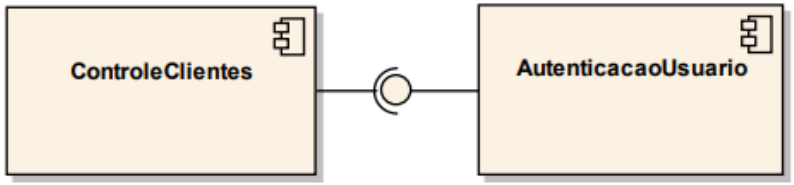
Interface requerida

A interface necessária para que o componente se comunique com outros componentes. Esta interface será conectada, então, em uma interface fornecida de outro sistema.

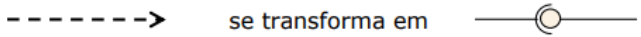


Fonte: http://micreiros.com/diagramas-de-componentes/

Diagrama de Componente



Notação gráfica de componentes, ilustrando as interfaces fornecida (**pelo componente AutenticacaoUsuario**) e exigida (**pelo componente ControleClientes**)



Transformação de relacionamentos de dependência em notação de interfaces fornecidas e exigidas.

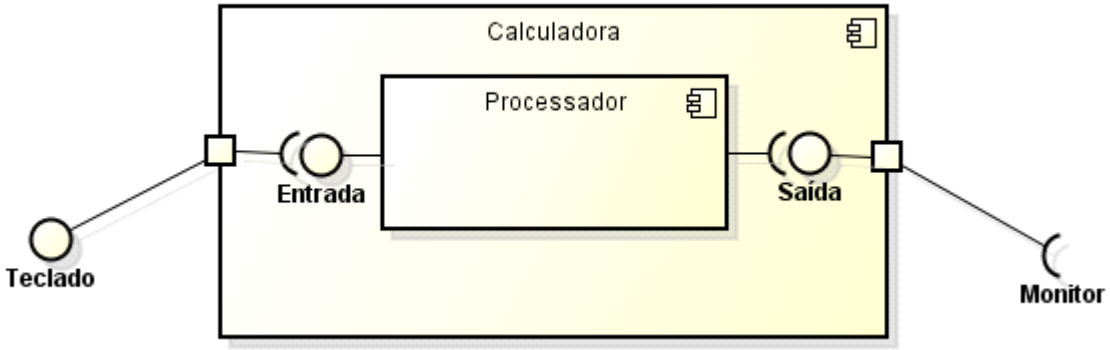
Fonte: http://luizantoniopereira.com.br/downloads/publicacoes/AnaliseEModelagemComUML.pdf

Diagrama de Componente

Newton
Quem se prepara, não para.

Portas

São elementos que permitem que elementos internos possam se comunicar com elementos externos do componente.



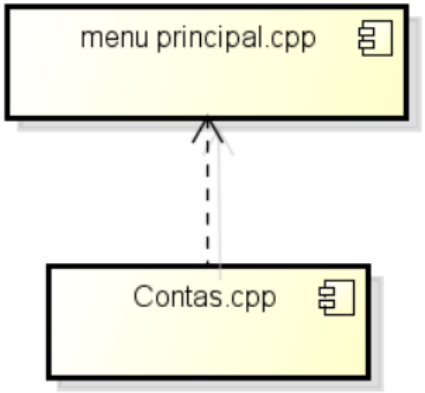
Fonte: http://micreiros.com/diagramas-de-componentes/

Diagrama de Componente

Newton
Quem se prepara, não para.

Dependências

Um componente pode utilizar serviços ou depender de alguma outra forma de outros componentes do sistema

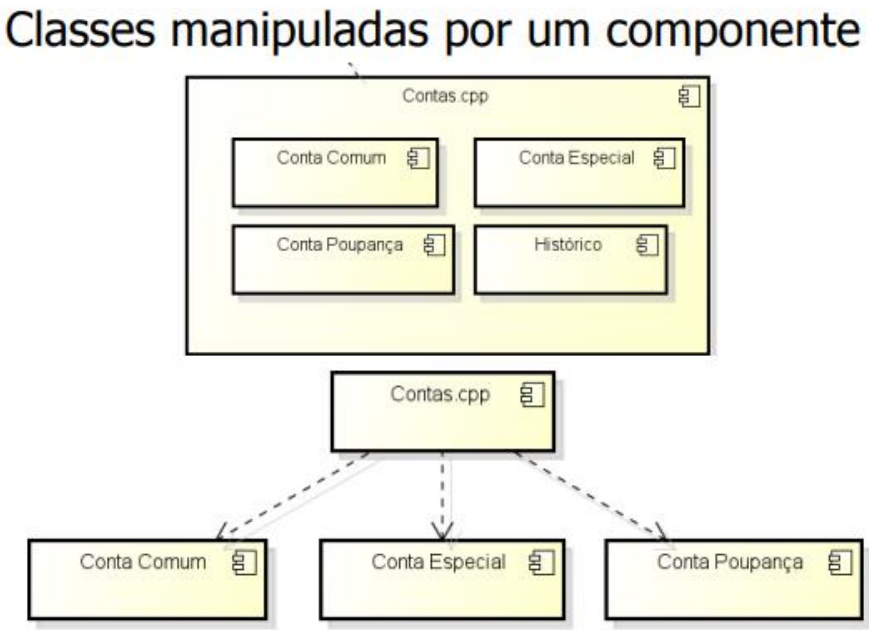


Fonte: http://www.deinf.ufma.br/~geraldo/dob/13.Componentes_Implantacao.pdf

Diagrama de Componente

Newton
Quem se prepara, não para.

Dependências



Fonte http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ES_II_2013_1/UML_Aula3.pdf

Diagrama de Componente - Exemplo

Newton
Quem se prepara, não para.

Considere a aplicação WinCOR responsável pelo gerenciamento de entrada e saída de correspondência de uma organização. A aplicação consiste num conjunto de componentes:

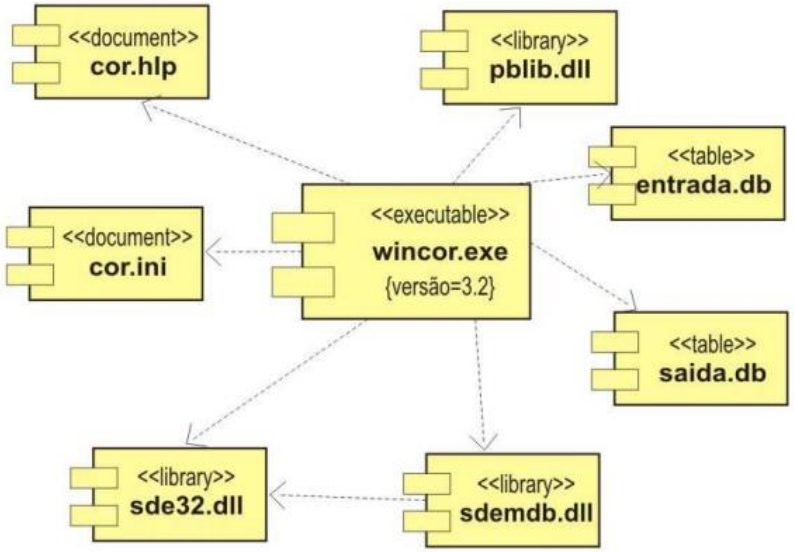
wincor.exe: arquivo que contêm o executável da aplicação;

pblib.dll, sde32.dll, sdemdb.dll: bibliotecas com código binário que providenciam funcionalidades adicionais;

wincor.hlp: arquivo de ajuda sobre a aplicação;

wincor.ini: arquivo de configuração da aplicação;

entrada.db, saida.db: arquivos/tabelas da base de dados de suporte.



Fonte http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ES_II_2013_1/UML_Aula3.pdf

Diagrama de Componente - Exemplo

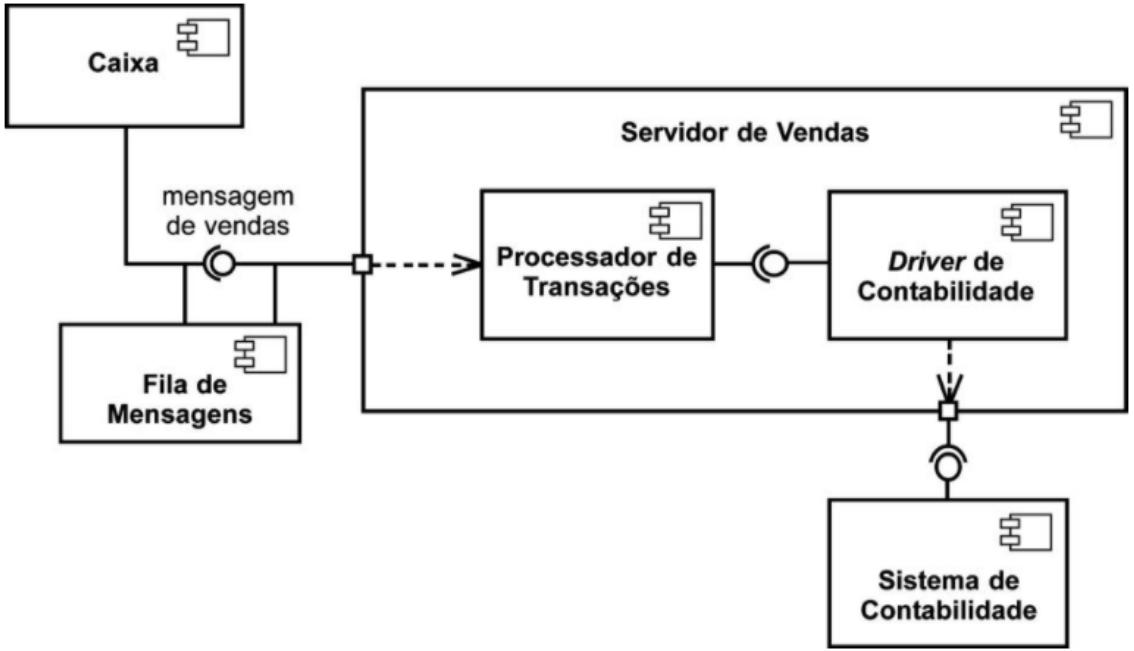


FIGURA 14.2 Um exemplo de diagrama de componentes

Fonte: Martin, 2011.

Referências

CAMPOS, Leandro de; COSTA, Gustavo K. M. Bilhar 3D. Disponível em <http://www.dca.fee.unicamp.br/courses/IA725/1s2008/proj/campos_costa/images/PackageDiagram1.jpg>. Acesso em: 25 Oct 2020.

LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objeto e ao desenvolvimento iterativo. Porto Alegre: Bookman, 2007, 3ª ed.

MARTIN, F. *UML Essencial*. Porto Alegre: Grupo A, 2011. 9788560031382. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788560031382/. Acesso em: 25 Oct 2020.

PRESSMAN, Roger S. MAXIM, Bruce R. **Engenharia de Software - Uma Abordagem Profissional**. 8.ed. Porto Alegre: Amgh Editora, 2016. 968p. ISBN 9788580555332.
SOMMERVILLE, Ian. **Engenharia de software**. 8.ed. São Paulo: A. Wesley publishing company, 2010.
.