

Modelagem Orientada a Objetos

e

UML

Conteúdo

- ⇒ introdução à modelagem: o que é um modelo, porque usamos, e tipos
- ⇒ introdução à modelagem orientada a objeto
- ⇒ o que é a UML , suas visões , e diagramas
- ⇒ softwares utilizados para modelagem orientada a objeto

O que é um modelo?

- ⇒ Um modelo é uma representação da realidade; é uma ligação, uma ponte entre conceitos teóricos e observações Aris (1978).
- ⇒ O desenvolvimento de um modelo envolve o uso do conceito de abstração, a visão do todo e de suas partes.
- ⇒ Adaptando Bender (1978), podemos dizer que no desenvolvimento de um modelo devemos: i) formular as idéias precisamente [com clareza]; ii) ser concisos no uso da linguagem; e iii) identificar bibliotecas e sistemas disponíveis.
- ⇒ Alguns modelos são focados em aspectos estruturais, como um projeto das fundações e da estrutura de um edifício. Outros modelos são focados na dinâmica e no comportamento do sistema. O uso de diferentes modelos nos permite ver os sistemas de diferentes formas; cada modelo, fornece-nos visão/detalhamento de uma parte do sistema, Sonerviile (1993).

Tipos e Exemplos de Modelos

⇒ Modelos estruturais

- Maquetes (maquete de uma casa, de um automóvel).
- Mapas (mapa das ruas de uma cidade).
- Projetos (projeto de uma casa).
- Modelos matemáticos (equações diferenciais, funções, exemplo: $y = f(x) = a + b \cdot x$).
- Modelos multiescala (miniaturas, modelos reduzidos).

⇒ Modelos dinâmicos

- Modelos físicos/químicos/biológicos (exemplo: a equação do movimento retilíneo uniforme).
- Modelos numéricos (exemplo: método de integração por Simpson).
- Algoritmos/fluxogramas (são diagramas utilizados para representar a dinâmica de determinado sistema).

Por que usar Modelos?

Linguagem Universal!

A engenharia civil utiliza com frequência [modelos simplificados da realidade](#), como, por exemplo, plantas de edifícios e casas. As plantas são utilizadas por permitirem o entendimento de como a obra deve ficar depois de pronta; é o modelo utilizado pelo engenheiro (ou arquiteto) para se [comunicar](#) com o proprietário, bem como com o mestre de obra. Assim, a planta viabiliza a troca de informações entre todos os envolvidos no projeto.

Com relação ao desenvolvimento de software, o uso de uma linguagem universal, como a UML e como a C++, facilita a troca de informações e códigos entre os desenvolvedores.

Por que usar Modelos?

O problema do Custo!

Usamos modelos para **reduzir custos**. Antigamente, um carro novo era desenvolvido utilizando-se protótipos em tamanho real, a um custo muito elevado e com poucas variações e testes. Na prática, **boa parte dos testes era feita pelos próprios proprietários**. Atualmente, um carro é feito valendo-se de modelos de computador, em um programa CAD, reduzindo consideravelmente o custo do desenvolvimento de novos modelos. Os modelos do carro são testados em simuladores.

Por que usar Modelos?

O problema da Escala!

A maquete de uma casa é um [modelo reduzido](#) da casa, em uma outra escala. A vantagem do uso de modelos em escalas reduzidas é seu baixo custo. Por exemplo: o [teste](#) da performance aerodinâmica de um carro, como um fórmula 1, é normalmente feito em um tunel de vento com modelos reduzidos.

Com relação ao desenvolvimento de um software, a questão da escala está associada ao crescimento do software. A cada versão novos recursos são adicionados, e o uso de modelos permite a previsão antecipada do crescimento do software.

Nota: quando um modelo não muda ao modificarmos a escala do problema, dizemos que o mesmo é invariável com relação à escala, Aris (1978).

Por que usar Modelos?

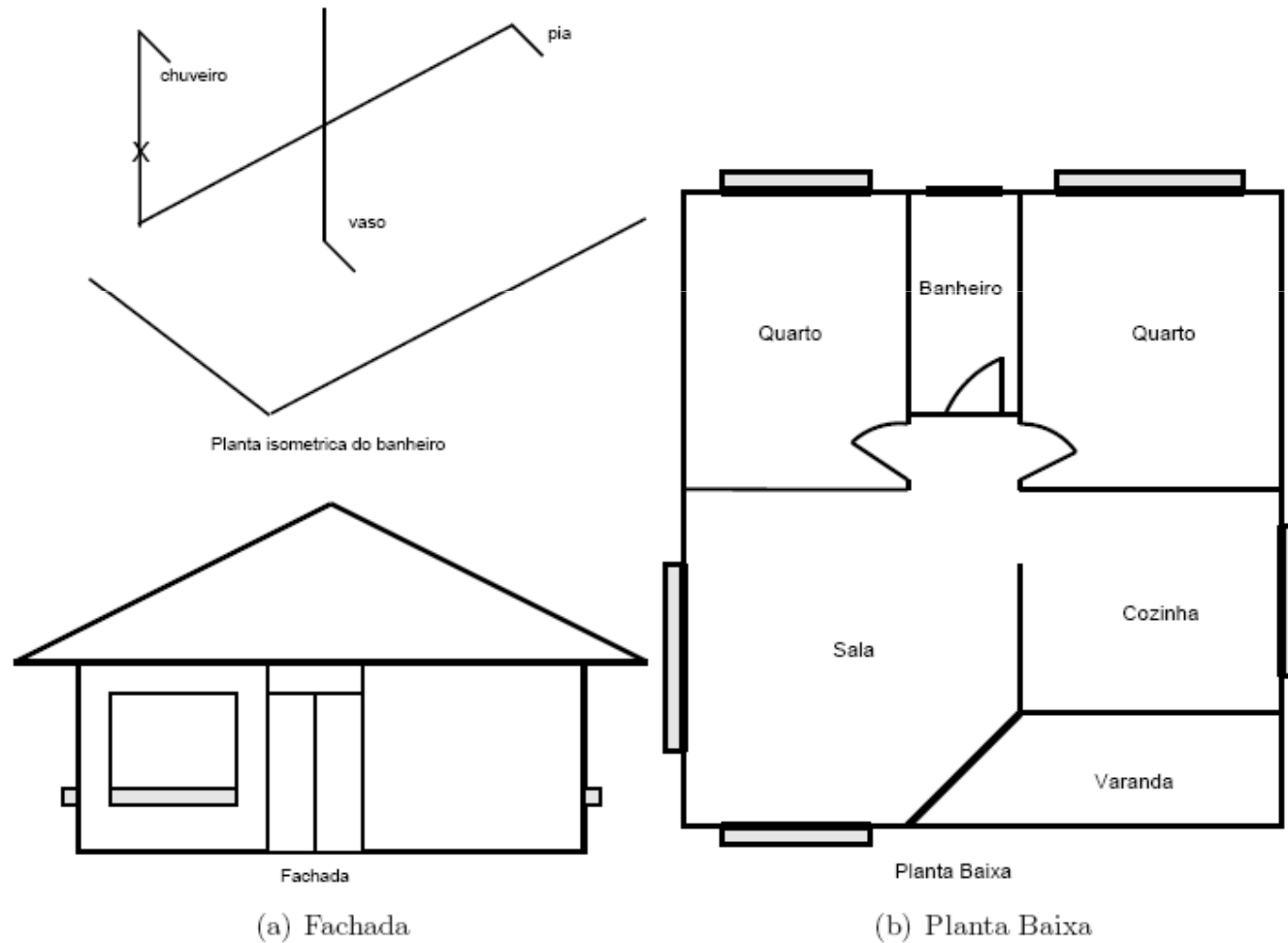
Outras Vantagens!

- ⇒ Maior facilidade para testar uma entidade física antes de lhe dar uma forma final.
- ⇒ Maior facilidade na comunicação entre as diversas pessoas envolvidas (pela utilização de notação uniforme).
- ⇒ Redução da complexidade dos sistemas.
- ⇒ Possibilidade de testar o sistema em escalas reduzidas.

Várias Visões?

Vários Modelos!

Figura 1: Modelos da realidade – a planta de uma casa.



O que é Modelagem Orientada a Objetos?

Relembre-se da figura da TV; a visão do usuário é diferente da visão do engenheiro. Para que nossa linguagem de comunicação seja uniformizada, precisamos utilizar a mesma base de informações, os mesmos conceitos.

Paradigmas de programação tradicionais, como o paradigma de programação estruturada, ensinam-nos uma nova linguagem (centenas de conceitos novos e seus relacionamentos). Todos os programadores devem entender e utilizar esta nova linguagem.

Contudo, a modelagem orientada a objeto elimina a necessidade de se aprender uma nova linguagem, pois a mesma é baseada em conceitos que já conhecemos – os objetos e seus relacionamentos. Isto ocorre porque, na modelagem orientada a objeto, os modelos da realidade são desenvolvidos tendo como fonte de inspiração os objetos.

Histórico da Modelagem Orientada a Objetos?

Segundo Fowler and Scott (2000), os métodos de análise e projeto orientado a objetos surgiram entre 1988 e 1992, destacando-se:

- i) o enfoque em projetos recursivos de Sally Shlaer e Steve Mellor;
- ii) os trabalhos de Coad e Yourdan, Coad and Yourdon (1993);
- iii) os cartões CRC de Beck e Cunningham (1989);
- iv) as técnicas de Booch, Booch (1986);
- v) o método TMO - Técnica de Modelagem de Objetos, de Rumbaugh et al. (1994);
- vi) os casos de uso de Ivar Jacobson.

Em 1997, o grupo de gerenciamento de objetos, conhecido como OMG – *Object Management Group*, adotou a UML como linguagem-padrão, o que provocou um impulso na aceitação da UML. Hoje em dia a UML é utilizada universalmente para modelagem de software orientado a objeto.

Quais as **Vantagens** da Modelagem OO?

Segundo Pressman (2002), a modelagem orientada a objeto apresenta algumas vantagens inerentes:

- ⇒ Reuso de componentes do programa.
- ⇒ Desenvolvimento mais rápido e com melhor qualidade.
- ⇒ Facilidade de manutenção, adaptação e ampliação.

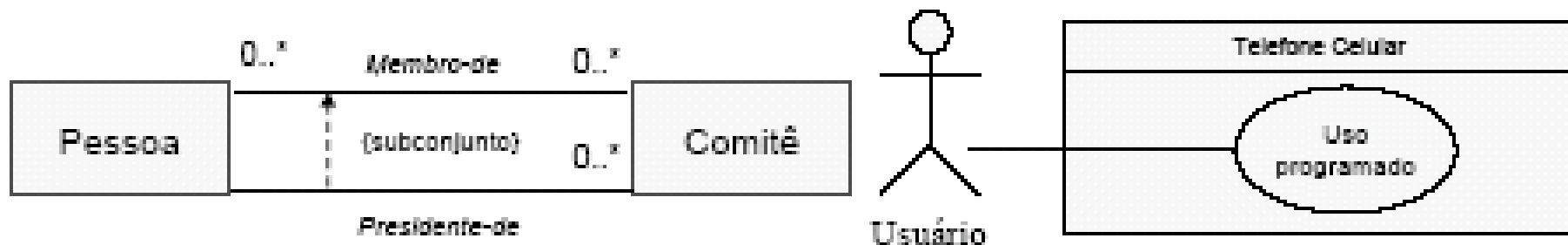
O que é a UML?

A UML é o padrão utilizado para o desenvolvimento de modelos de software orientados a objeto. Segundo Fowler and Scott (2000), o código é o meio preciso e detalhado, e a linguagem natural é muito imprecisa; UML seria então o meio termo.

- ⇒ A UML é uma linguagem visual que usa a idéia "uma imagem vale mais do que mil palavras".
- ⇒ A UML é uma linguagem de modelagem independente de processo.
- ⇒ A UML é dinâmica; inovações e ajustes são acrescentados a cada nova versão.
- ⇒ A UML é o padrão da indústria para modelagem de software orientado a objeto.
- ⇒ A UML é extensível, podendo ser adaptada às suas necessidades.

O que é a UML?

- Linguagem visual para especificação (modelagem) de sistemas orientados a objetos
 - Fornece representação gráfica para os elementos essenciais do paradigma de objetos
 - Classes, atributos, objetos, troca de mensagens, ...



O que é a UML?

- De propósito geral
 - Não está presa a uma etapa do desenvolvimento de software
 - Análise
 - Projeto
 - Implementação
 - Testes
 - Não está presa a um processo
 - Ciclo de vida em cascata
 - Incremental
 - Processo Unificado
 - ...
 - Não está presa a uma linguagem de programação

O que é a UML?

– Padrão OMG

- Em <http://www.omg.org> estão disponíveis documentos eletrônicos que contém
 - Sumário da UML
 - Semântica
 - Guia da Notação
 - Extensões da Linguagem

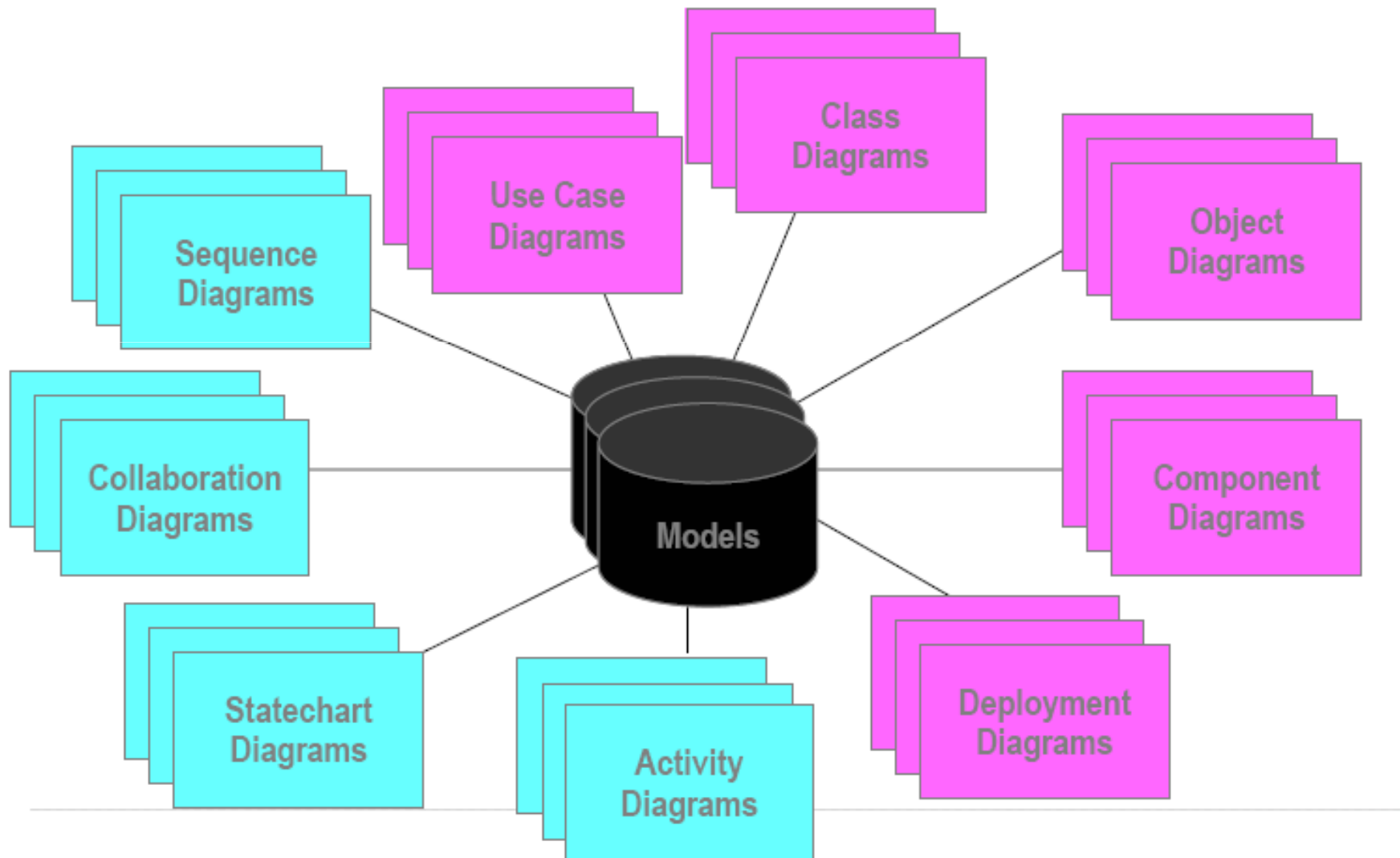
O que podemos **Modelar** com **UML**?

- Requisitos
- Comportamento e sua estrutura lógica
- A dinâmica de seus processos
- Necessidades físicas em relação ao equipamento no qual o sistema será implantado.

Modelos *versus* Diagramas

- Um modelo contém todos os elementos de informação sobre o sistema. É independente de como os elementos são visualmente representados. Descrição Completa.
- Um diagrama é uma visualização particular de certos tipos de elementos de um modelo. Geralmente expõe um subconjunto das informações detalhadas de um elemento.
- Um elemento do modelo pode existir em diversos diagramas, mas existe apenas uma definição desse elemento no modelo.

Diagramas da UML



Diferentes Visões?

Vários Diagramas!

A seguir são apresentadas as diferentes visões da UML e os respectivos diagramas:

- ⇒ **Visão do modelo do usuário** – mostra a visão do usuário do sistema, sendo descrita principalmente pelos casos de uso.
 - Diagrama de caso de uso.

- ⇒ **Visão do modelo estrutural** – mostra a estrutura do sistema (equivale ao modelo de objetos do método TMO).
 - Diagrama de pacotes.
 - Diagrama de classes, atributos, métodos, associações, agregações, heranças, e dependências.
 - Diagrama de objetos.
 - Diagrama de estrutura composta.

Diferentes Visões?

Vários Diagramas!

⇒ **Visão do modelo dinâmico** – mostra a dinâmica e o comportamento do sistema, sua interação com o usuário e com sistemas externos (equivale ao modelo dinâmico e funcional do método TMO).

- Diagrama de seqüência (eventos e mensagens).
- Diagrama de comunicação/colaboração.
- Diagrama de máquina de estado (estados do objeto).
- Diagrama de atividades (detalha as funções/métodos).
- Diagrama de tempo.

Diferentes Visões?

Vários Diagramas!

⇒ **Visão de implementação** – mostra aspectos estruturais do sistema relacionados às necessidades de implementação (equivale em parte ao projeto do sistema do método TMO).

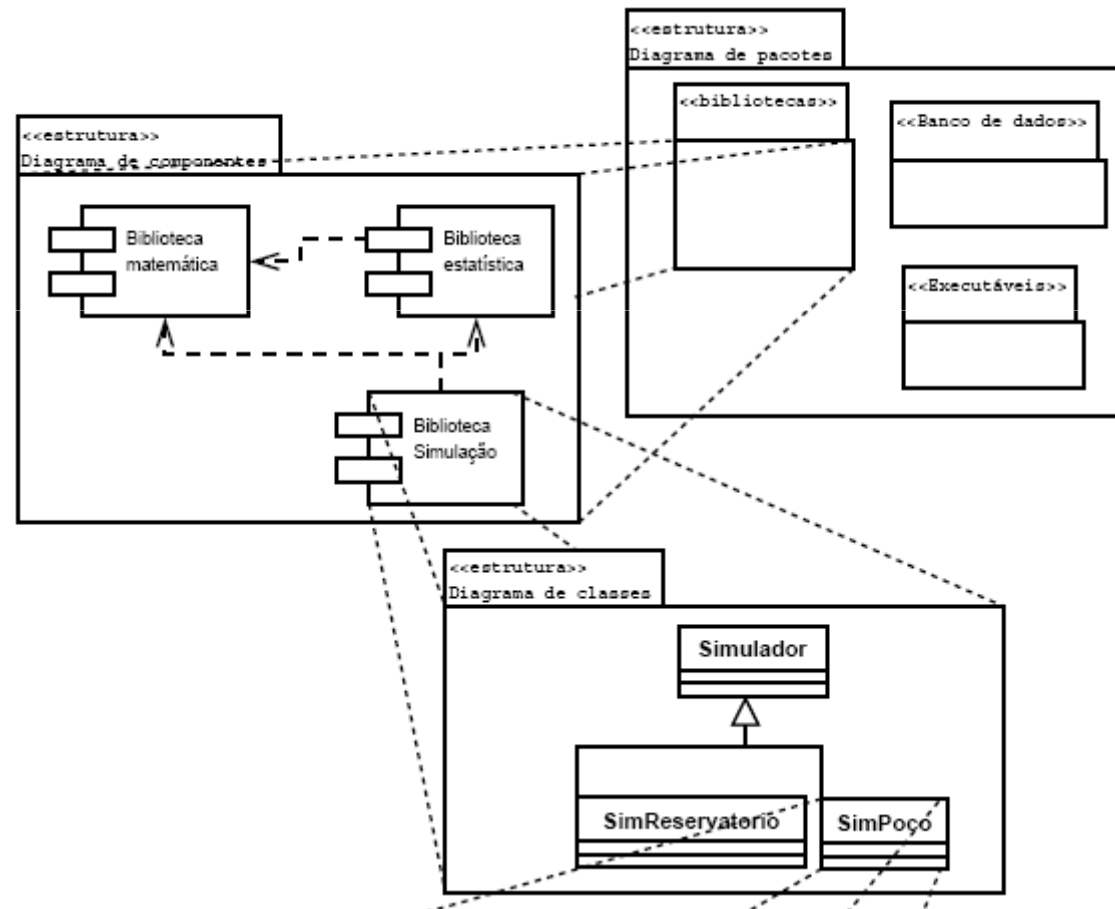
- Diagrama de componentes.

⇒ **Visão do modelo de ambiente** – mostra o ambiente-alvo e as necessidades para se colocar o sistema em funcionamento.

- Diagrama de implantação.

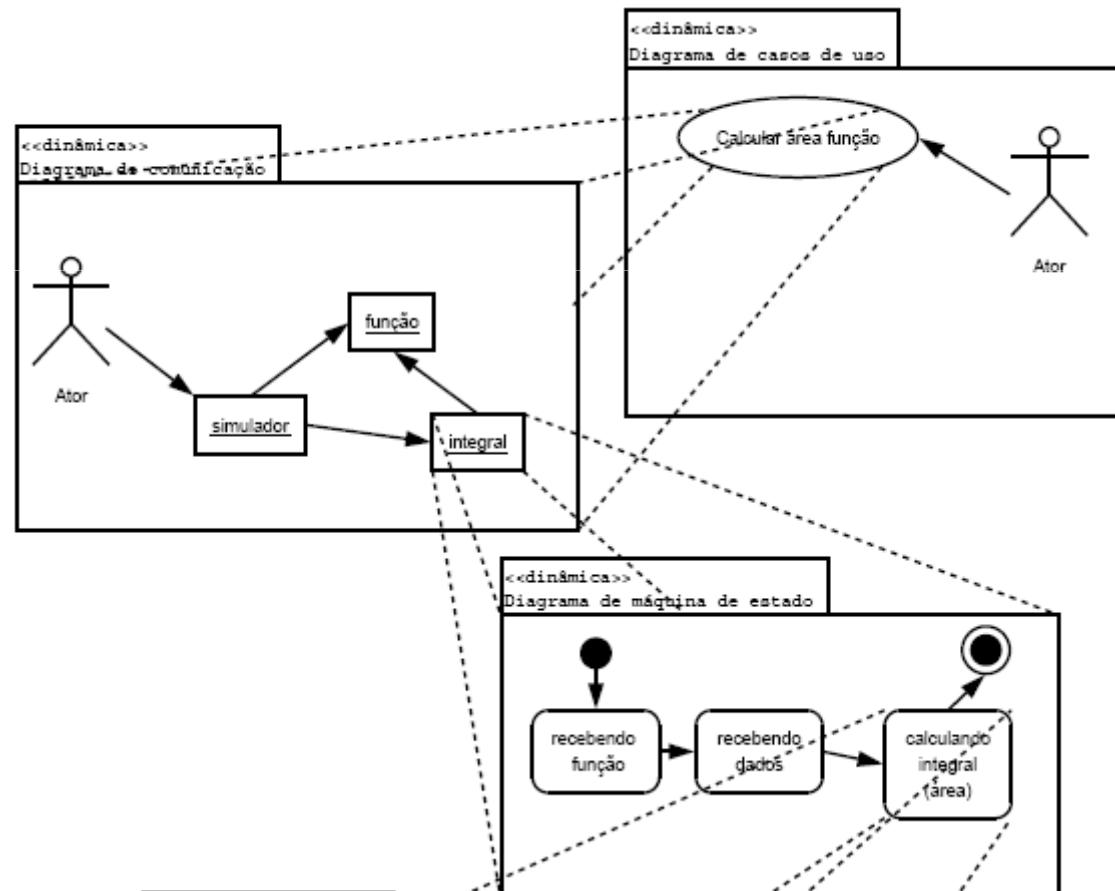
Diferentes Visões?

Vários Diagramas!

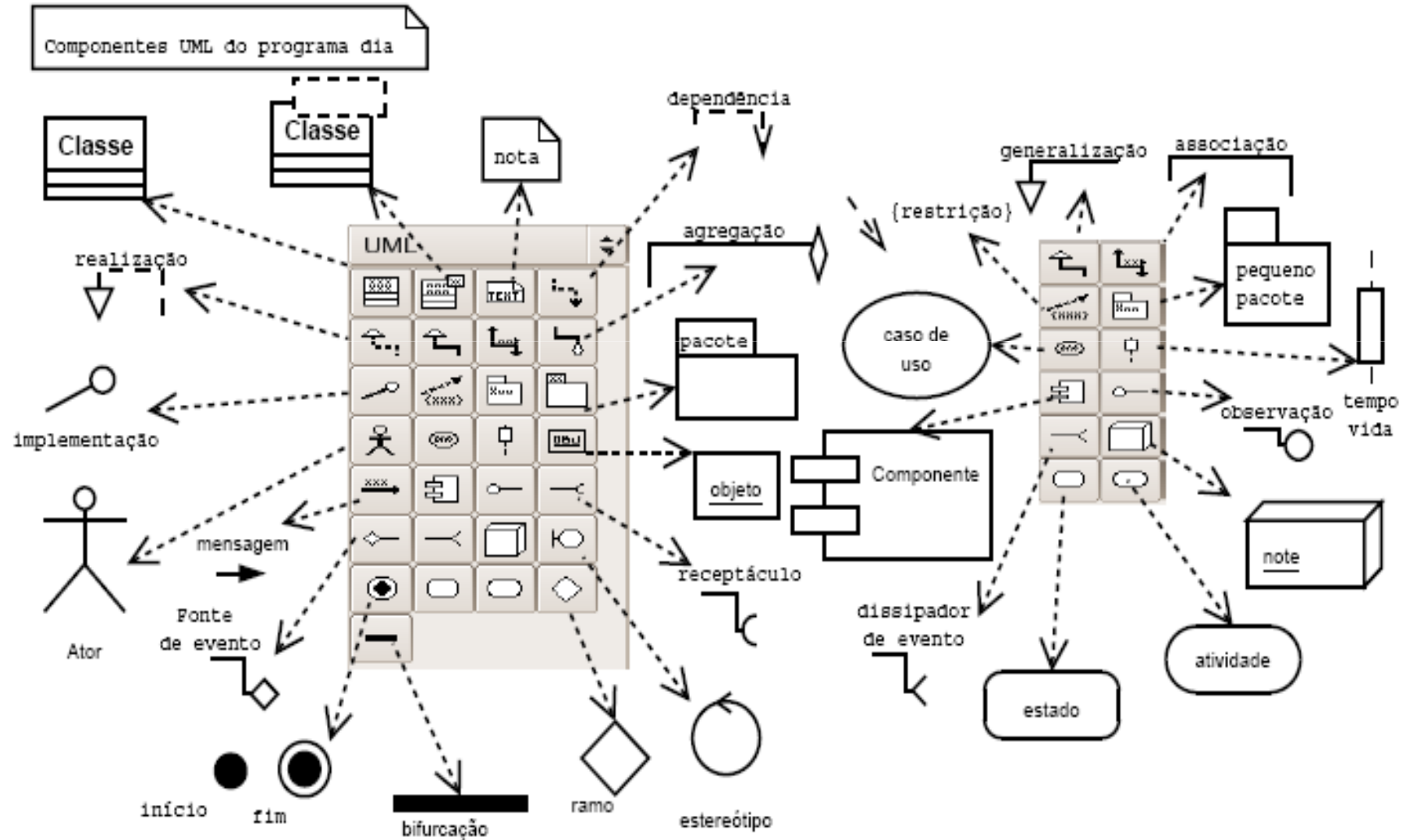


Diferentes Visões?

Vários Diagramas!



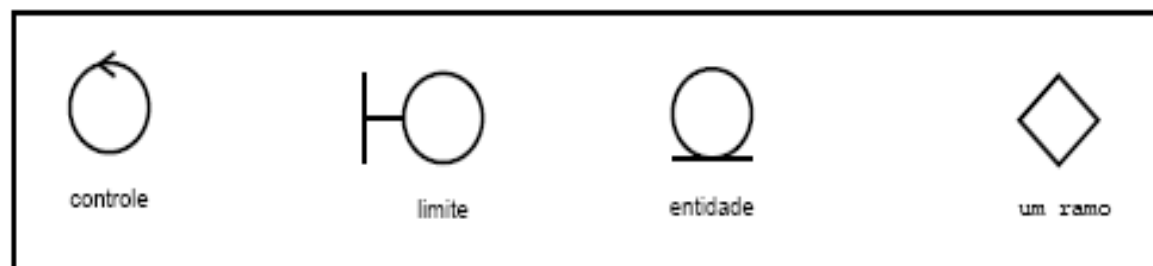
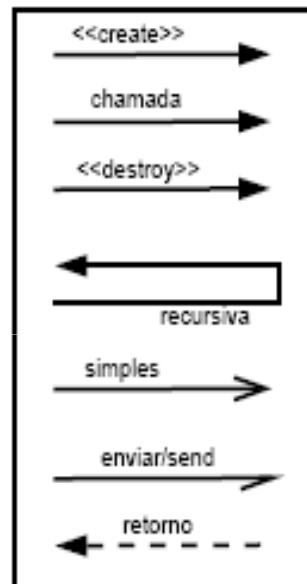
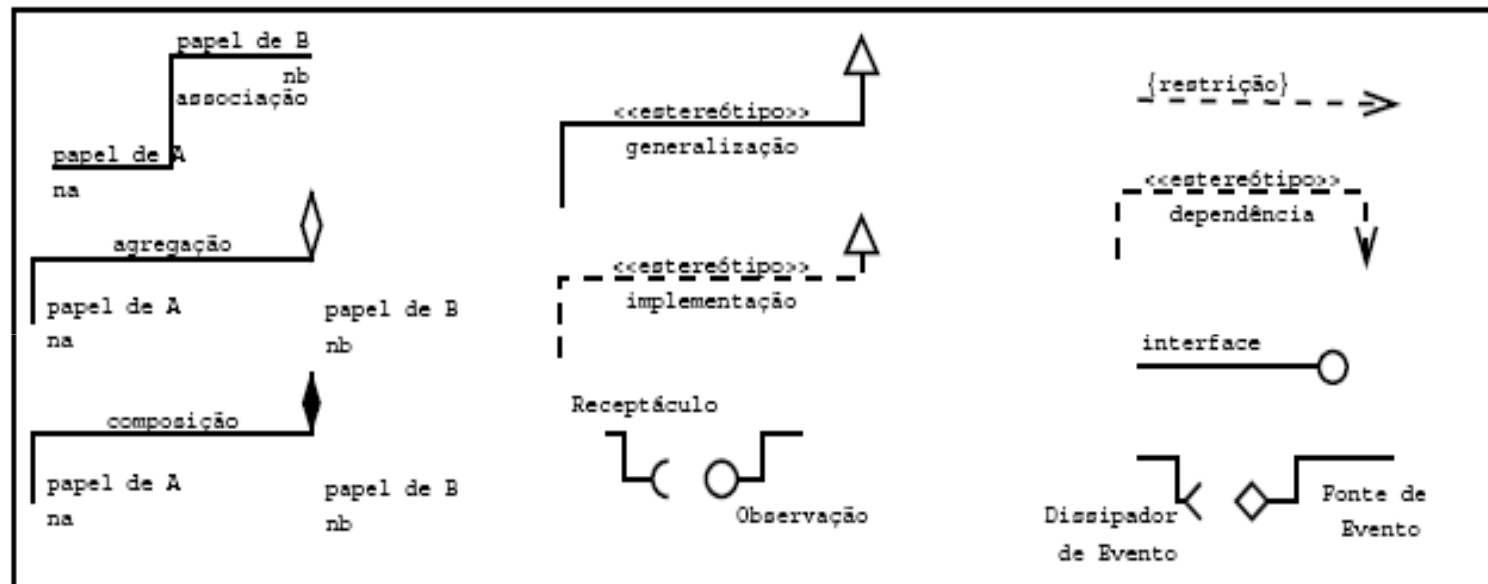
Os Elementos Básicos da UML



Outros Elementos da UML

Notação UML para associações, agregação, composição, generalização, implementação, observação e receptáculo, fonte e dissipador de eventos, restrição, dependência, interface.

Tipos de mensagens



Ícone de classe.

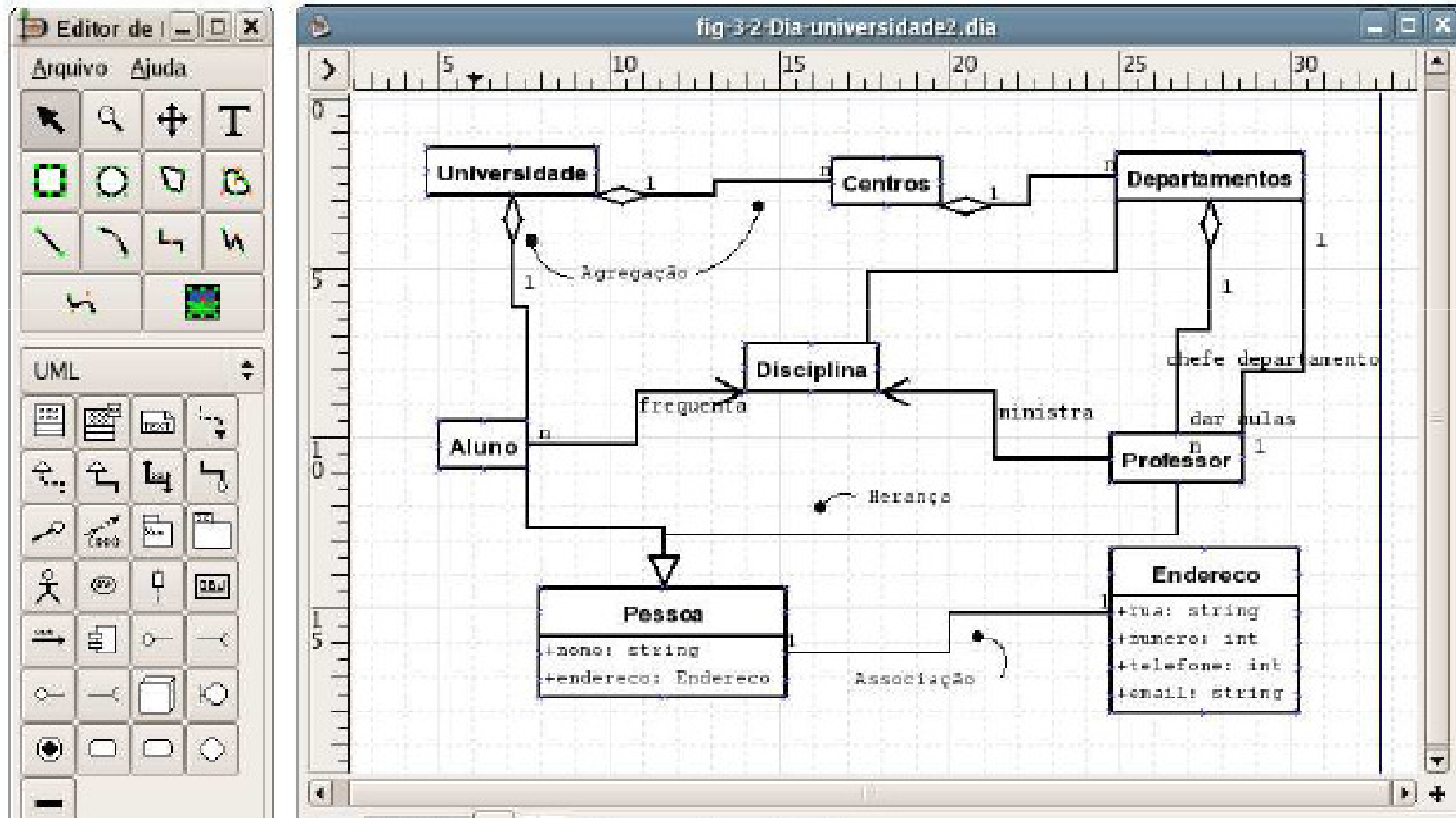
Ferramentas para Modelagem UML

- ⇒ **Rational Rose** – é um pacote profissional que, além da montagem dos diagramas, permite simultaneamente a implementação dos códigos. É um pacote pago, disponível para diversas plataformas no endereço <http://www.rational.com>.
- ⇒ **With Class** – é um pacote profissional, disponível para diversas plataformas no endereço <http://www.microgold.com/index.html>.
- ⇒ **Dia** – o modelador *dia* é muito simples de usar, é software livre. Pode ser obtido em <http://www.gnome.org/gnome-office/dia.shtml>.
- ⇒ **Umbrello** – o modelador *umbrello* também é muito simples de usar, é software livre. Pode ser obtido em <http://uml.sourceforge.net/index.php>.
- ⇒ **Visual Paradigm** – pacote profissional, com versão livre.

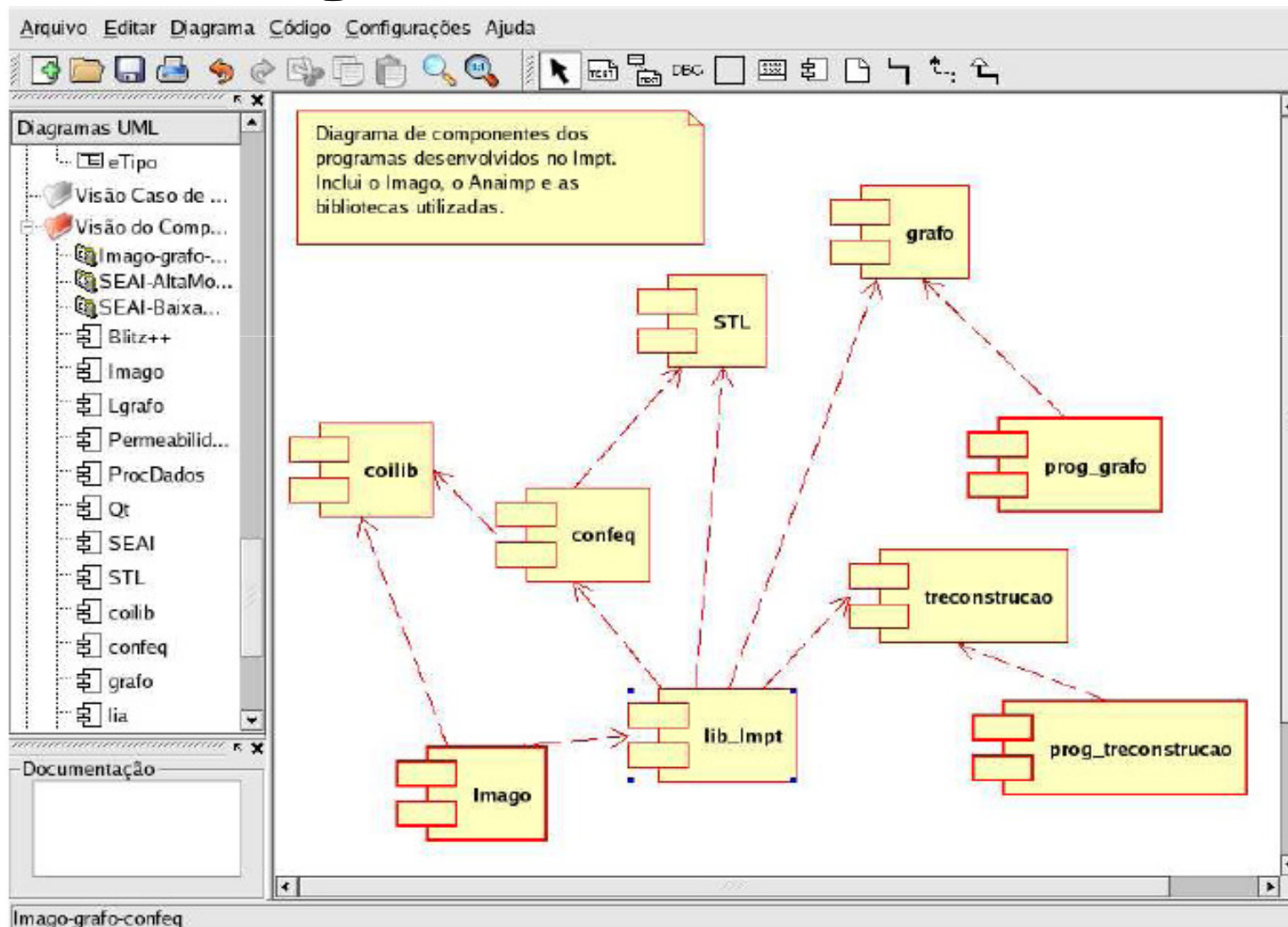
Outras Ferramentas para Modelagem UML

Ferramenta	Tipo de Licença	Suporte UML	Ponto forte
Rational Rose	Rational Software	UML 2.0	É parte de um ambiente integrado
Jude	livre	Parcialmente, UML 2.0	usabilidade
Poseidon	livre	UML 1.5	usabilidade
Enterprise Architect	Sparx Systems	UML 2.0	Mais recursos compatíveis com a UML 2.0
VP-UML	Pode ser liberada	UML 2.0	Oferece cópia para instituições de

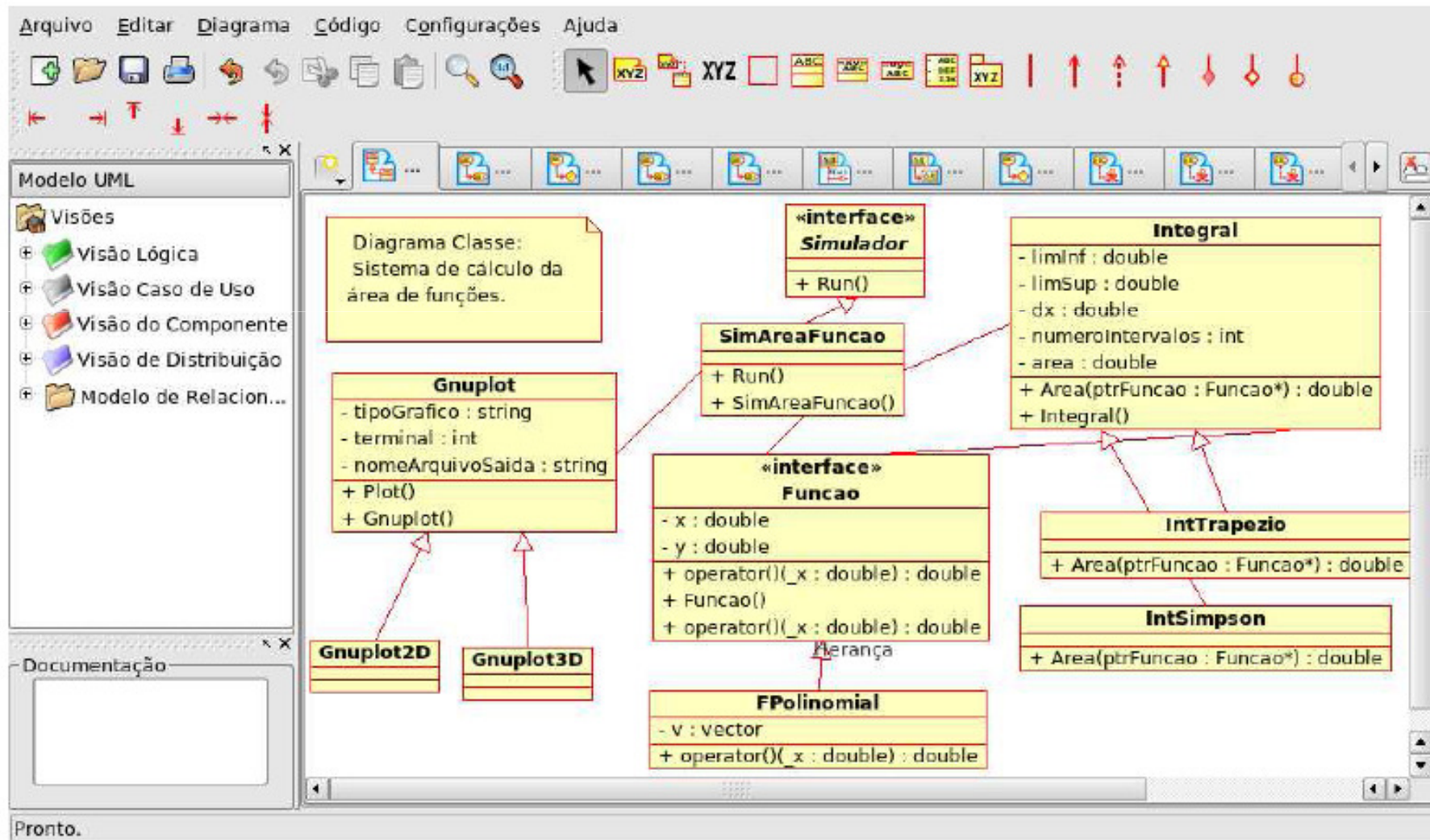
Ferramentas para Modelagem UML - DIA



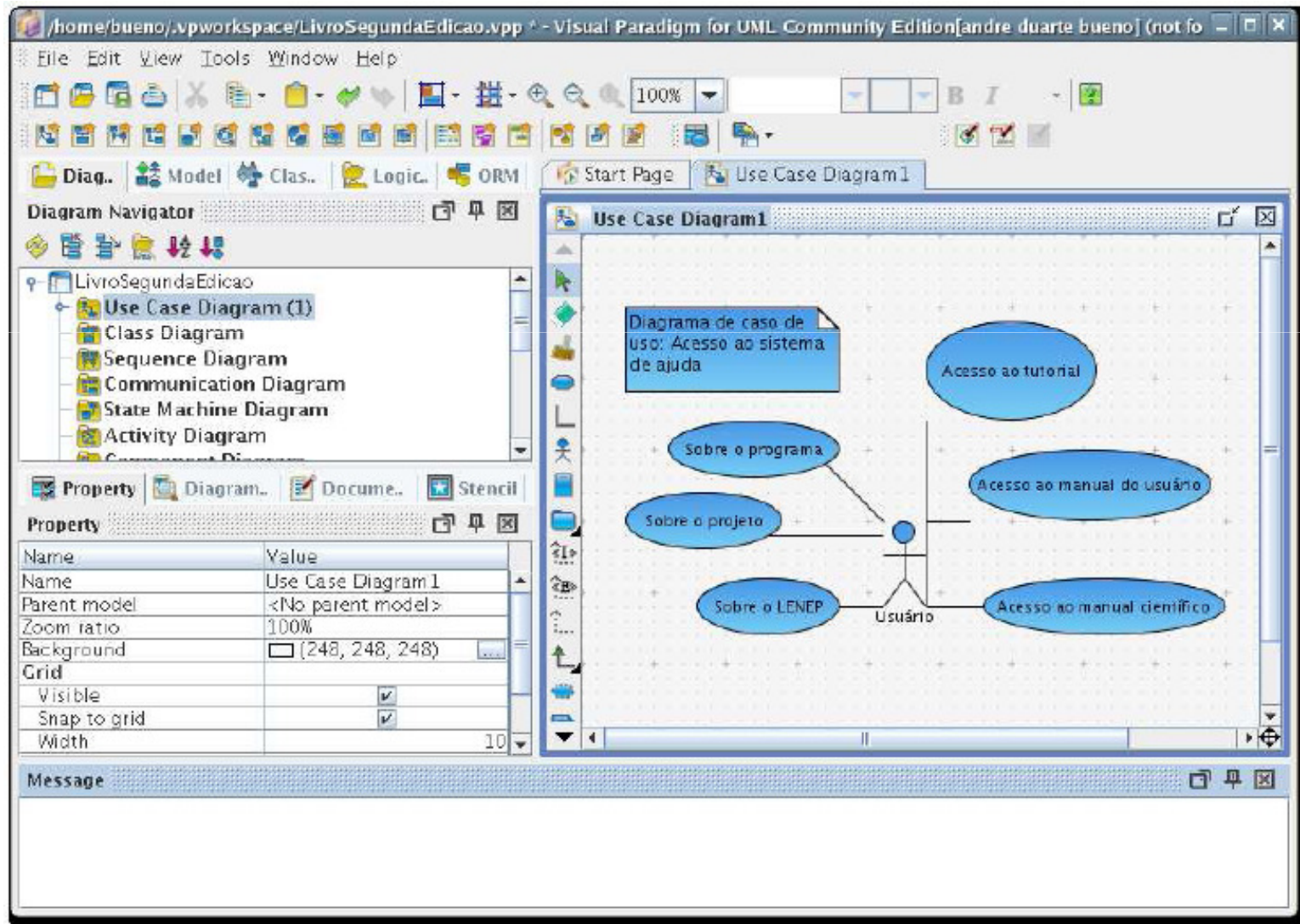
Ferramentas para Modelagem UML - Umbrello



Ferramentas para Modelagem UML - Umbrello



Ferramentas para Modelagem UML - Visual Paradigm



Resumo

Neste capítulo aprendemos que **um modelo é uma descrição simplificada e reduzida da realidade** cuja vantagem está associada ao seu **baixo custo** e ao uso de uma **linguagem universal**. Vimos que os modelos (**estruturais ou dinâmicos**) podem ser utilizados para nos ajudar a solucionar os mais variados problemas da engenharia.

Aprendemos que a idéia da **modelagem orientada a objeto é desenvolver modelos tendo como foco o conceito de objeto**. A modelagem orientada a objeto surgiu ao longo dos anos de 1980 e ficou madura nos anos de 1990 com o surgimento da UML, um sistema de modelagem que unificou as antigas notações. No final do capítulo vimos o que são e como obter programas modeladores como o *dia* e o *umbrello*.

Referências Bibliográficas

Aris, R. (1978). *Mathematical Modelling Techniques*. dover.

Bender, E. A. (1978). *An Introduction to Mathematical Modeling*. dover.

Booch, G., editor (1986). *Object-oriented development*, volume 12 of *IEEE Transactions on Software Engineering*.

Coad, P. and Yourdon, E. (1993). *Análise Orientada a Objeto*. Campus, São Paulo.

da Silva Neto, A. J. and Neto, F. D. M. (2005). *Problemas Inversos - Conceitos Fundamentais e Aplicações*. Editora UERJ.

Fowler, M. and Scott, K. (2000). *UML Essencial*. Bookman, São Paulo.