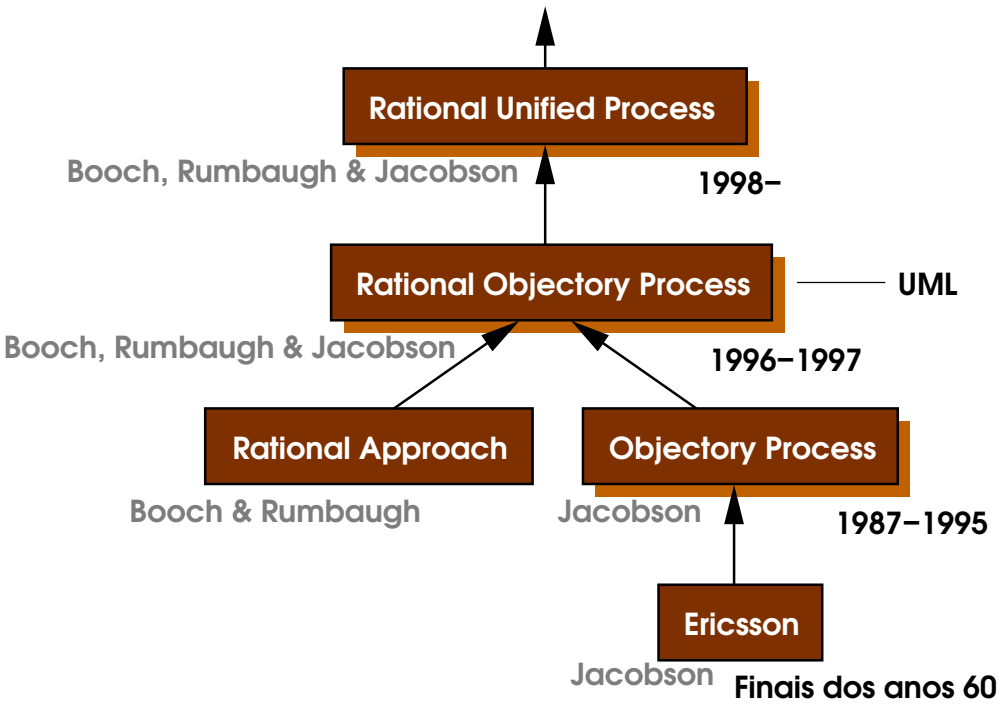
Unified Software Development Process

Sumário

- Breve história do Unified Process
- O Unified Process
- O ciclo de vida do Unified Process
- O RUP (Rational Unified Process)

Breve História do Unified Process



O Unified Process

O Unified Process é:

- Uma *framework* para o processo de desenvolvimento, genérica e adaptável.

O Unified Process fomenta:

- O desenvolvimento baseado em componentes.

O Unified Process utiliza:

- o UML como ferramenta de modelação durante todas as fases do processo de desenvolvimento.

Três ideias chave

- desenvolvimento guiado por *Use Cases* (Casos de Uso)
- desenvolvimento centrado na arquitectura;
- desenvolvimento iterativo e incremental.

Desenvolvimento guiado por Use Cases

- Um *use case* representa uma interacção entre o sistema e um humano ou outro sistema;
- O modelo de *use cases* é utilizado para:
  - guiar a concepção do sistema (captura de requisitos funcionais);
  - guiar a implementação do sistema (implementação de um sistema que satisfaça os requisitos);
  - guiar o processo de testes (testar que os requisitos são satisfeitos).

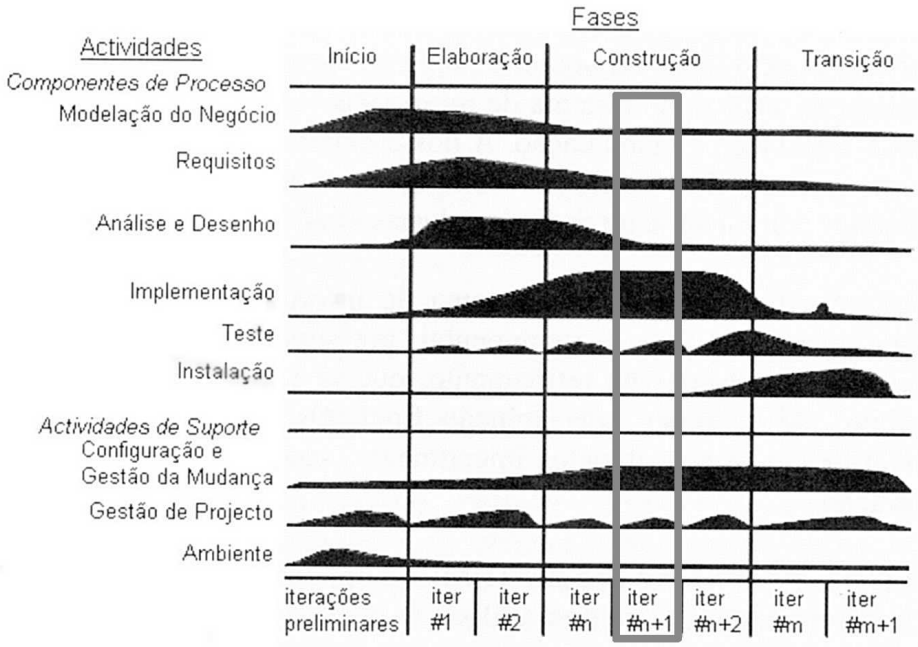
Desenvolvimento centrado na arquitectura

- O modelo de use cases descreve a função do sistema, o modelo da arquitectura descreve a forma;
- O modelo arquitectural permite tomar decisões sobre:
  - O estilo de arquitectura a adoptar;
  - Quais os componentes do sistema e quais as suas interfaces;
  - A composição de elementos estruturais e comportamentais.

Desenvolvimento iterativo e incremental

- Permite dividir o desenvolvimento em "pedaços geríveis";
- Em cada iteração:
  - identificar e especificar *use cases* relevantes;
  - criar uma arquitectura que os suporte;
  - implementar a arquitectura utilizando componentes;
  - verificar que os *use cases* são satisfeitos.
- Selecção de uma iteração:
  - grupo de *use cases* que extendam a funcionalidade;
  - aspectos de maior risco.

O ciclo de vida do Unified Process



- O Unified Process divide o desenvolvimento de software em 4 fases:
  - Início (*Inception*);
  - Elaboração (*Elaboration*);
  - Construção (*Construction*);
  - Transição (*Transition*).
- Cada fase é constituida por um conjunto de iterações.
- Em cada iteração são realizadas um conjunto de actividades:
  - Análise de requisitos;
  - Análise e desenho;
  - Implementação;
  - Testes;
  - Instalação.
- Em função da fase em que se está a realizar a iteração, algumas actividades têm mais peso que outras.
- O objectivo é que, em cada iteração, seja produzida uma versão do produto final.

Fases do Unified Process

Início

- Identificar problema
- Decidir âmbito e natureza do projecto
- Fazer estudo de viabilidade

Resultado da fase: decisão de avançar com o projecto.

Elaboração (Análise/Concepção Lógica)

- O que vamos construir (quais os requisitos?)
- Como vamos fazê-lo? (qual a arquitectura?)
- Que tecnologias vamos utilizar?

Resultado da fase: uma arquitectura geral (conceptual) do sistema.

Construção (Concepção Física/Implementação)

- Processo iterativo e incremental
- Em cada iteração: análise/especificação/codificação/teste/integração de parte do SI

Resultado da fase: um sistema de informação!

Transição

- Acertos finais na instalação do SI
- Optimização, formação.

Resultado da fase: um SI instalado e 100% funcional.

O RUP (Rational Unified Process)

- O RUP é uma *concretização* do Unified Process.
- O RUP fornece:
  - ferramentas de gestão do processo de desenvolvimento segundo a *framework* definida pelo Unified Process (ver página 64);
  - ferramentas para a modelação e desenvolvimento baseadas no UML;
  - uma base de conhecimento (*knowledge base*).
- Na verdade as coisas passaram-se um pouco ao contrário...

Apresentação do UML

Sumário

- UML
- Breve história do UML
- Diagramas do UML
- Meta modelo do UML
- Mecanismos de extensão
- Decorações

UML — Unified Modelling Language

UML: *Unified Modelling Language* (Booch, Jacobson & Rumbaugh)

- UML foi pensado para o desenvolvimento de sistemas orientados aos objectos
  → permite explorar o paradigma OO
- UML possibilita o trabalho a diferentes níveis de abstracção
  → facilita a comunicação
- UML não é uma linguagem, mas um conjunto de linguagens
  → inclui modelos para as diferentes fases do desenvolvimento.
- UML não é um processo de desenvolvimento de software, pode ser usado com diferentes processos.

Breve história do UML

- anos 60
  - Simula 67 é a primeira linguagem orientada aos objectos;
- anos 70
  - Aparece o Smalltalk;
- anos 80
  - as linguagens orientadas aos objectos (OO) tornam-se *utilizáveis* Smalltalk estabiliza; surgem o Objective C, C++, Eiffel, CLOS, etc.
  - finais dos anos 80 — surgem as primeiras metodologias de modelação OO
- anos 90
  - existem dezenas de metodologias de modelação OO Shlaer/Mellor, Coad/Yourdon, Booch, OMT (Rumbaugh), OOSE (Jacobson), etc. etc.
  - meados dos anos 90 — começam as tentativas de unificação dos métodos
  - 1994 — Rumbaugh junta-se a Booch na Rational Software Corporation
  - 1995 — Booch e Rumbaugh apresentam a versão 0.8 do *Unified Method* (viria depois

a mudar de nome para *Unified Modelling Language*); Jacobson junta-se a Booch e Rumbaugh
  - 1996 — o OMG (*Object Management Group*) pede propostas para um standard de modelação OO
  - Setembro, 1997 — a Rational, em conjunto com outras empresas (HP, IBM, Oracle, SAP, Unisys, ...), submete o UML 1.0 ao OMG como proposta de standard (existiram outras propostas)
  - Novembro, 1997 — o UML é aprovado como standard OO pelo OMG; o OMG assume a responsabilidade do desenvolvimento do UML
- 00(?!)
  - 2004 — está a ser finalizado o UML 2.0

`www.uml.org`

Diagramas do UML

O UML define os seguintes oito diagramas base:

- ❑ Diagramas de Use Case
- ❑ Diagramas de Classe
- Diagramas comportamentais
  - Diagramas de Interacção
    - ❑ Diagramas de Sequência
    - ❑ Diagramas de Colaboração
  - ❑ Diagramas *statechart* (de estados)
  - ❑ Diagramas de Actividade
- Diagramas de implementação
  - ❑ Diagramas de Componentes
  - ❑ Diagramas de Instalação (*Deployment*)

❑ Diagramas UML

- A escolha dos modelos a utilizar tem uma grande impacto na forma como um dado

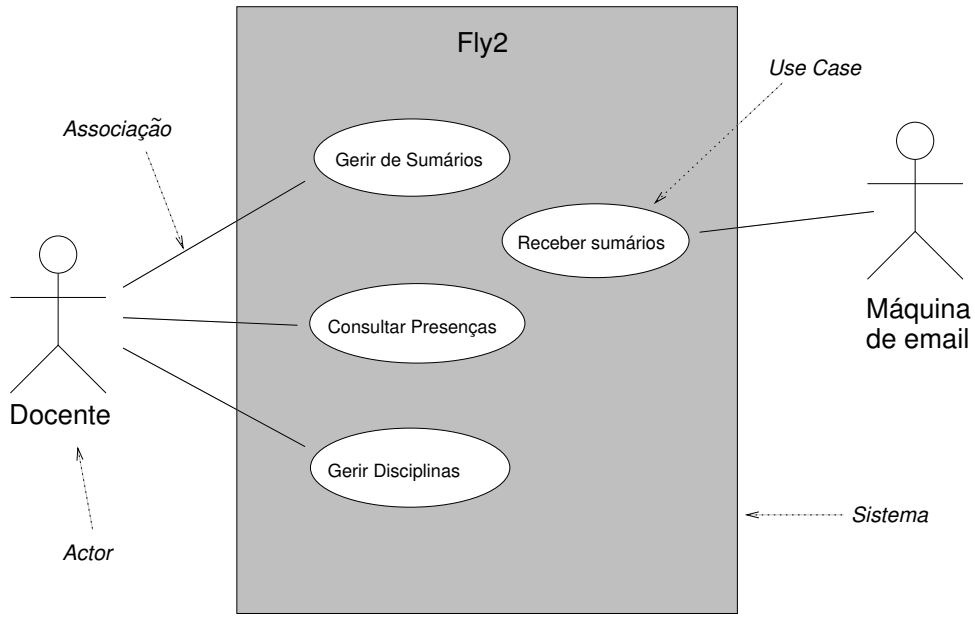
problema é abordado e, consequentemente, na solução que se irá atingir.

- A *Abstracção* (prestar atenção aos detalhes importantes e ignorar ou irrelevantes) é um factor fundamental.
- Assim:
  - Todo o sistema complexo deve ser abordado através de um pequeno conjunto de vistas/modelos tão independentes quanto possível;
  - Cada modelo pode ser expresso a diferentes níveis de detalhe;
  - Os melhores modelos são aqueles que têm relação directa com a realidade.
- Os oito tipos de diagramas identificados anteriormente procuram cobrir todas as necessidades de modelação que ocorram durante o desenvolvimento de software.

Um exemplo

Desenvolver uma aplicação para gestão de sumários e presenças nas aulas para universidades.

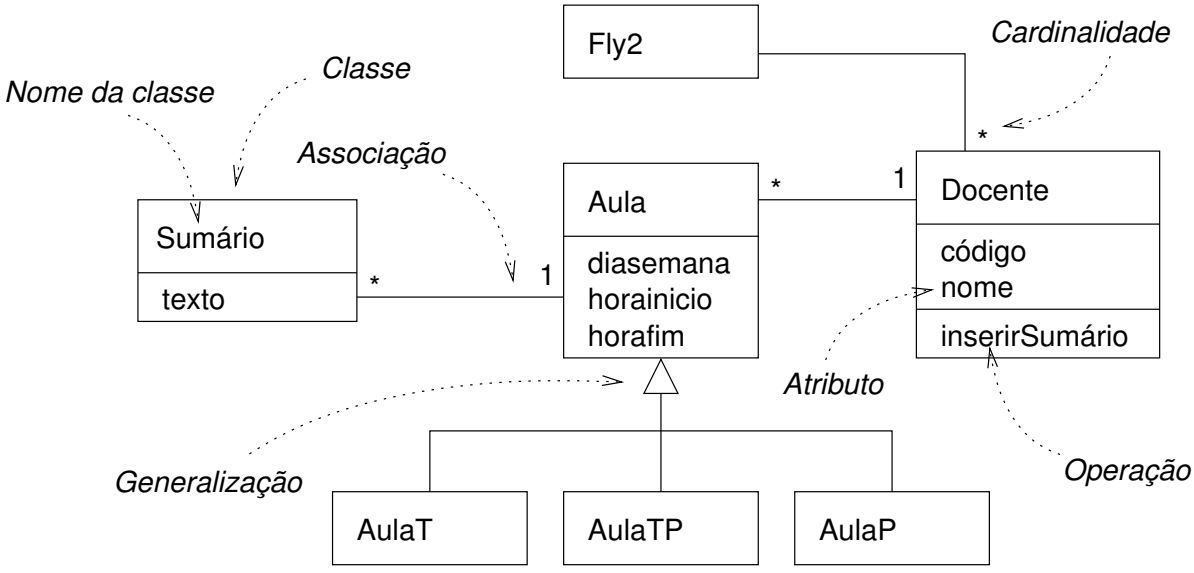
Diagramas de Use Case



- Identificam os utilizadores do sistema / capturam os requisitos funcionais.

Ver página 64

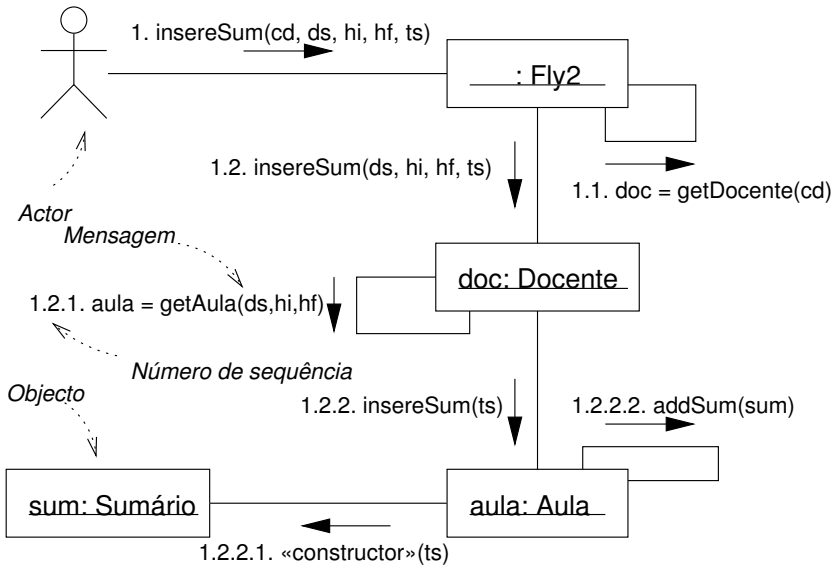
Diagramas de Classe



- Modelam a arquitectura(?) do sistema.

Ver página 64

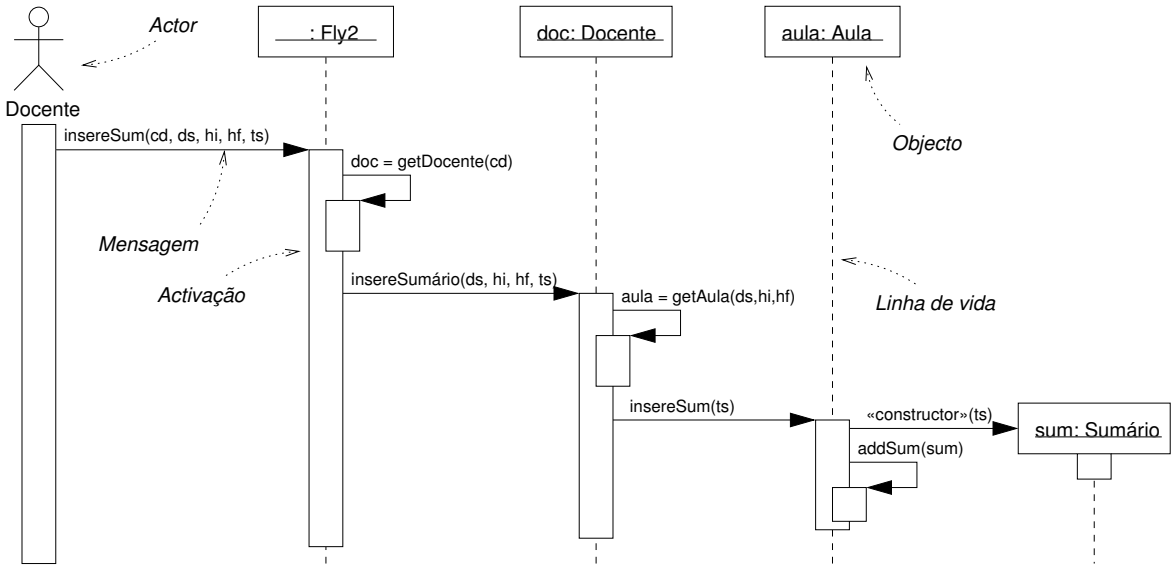
Diagramas de Colaboração



- Modelam o comportamento do sistema (ênfase no aspecto estrutural).

Ver página 136

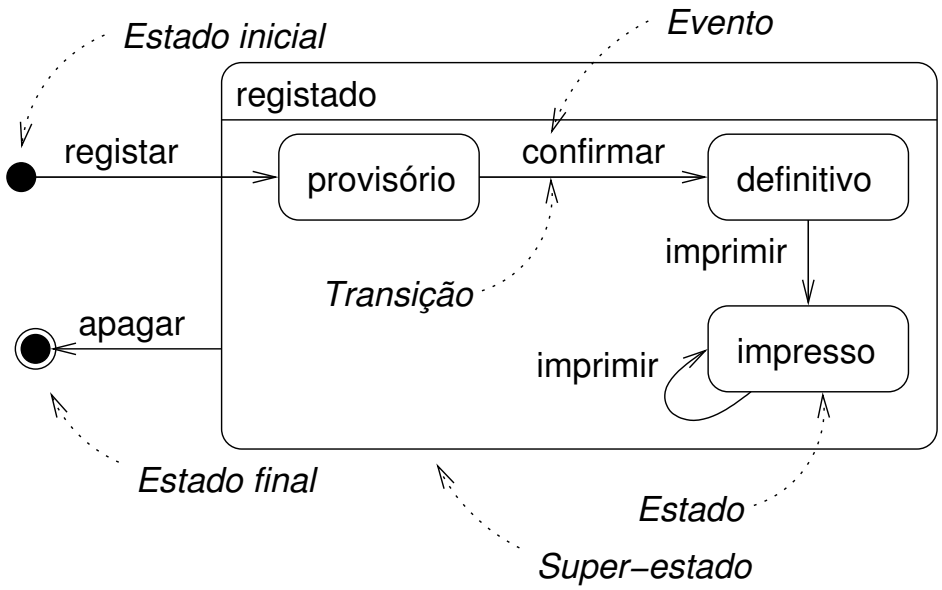
Diagramas de Sequência



- Modelam o comportamento do sistema (ênfase no aspecto temporal).

Ver página 76

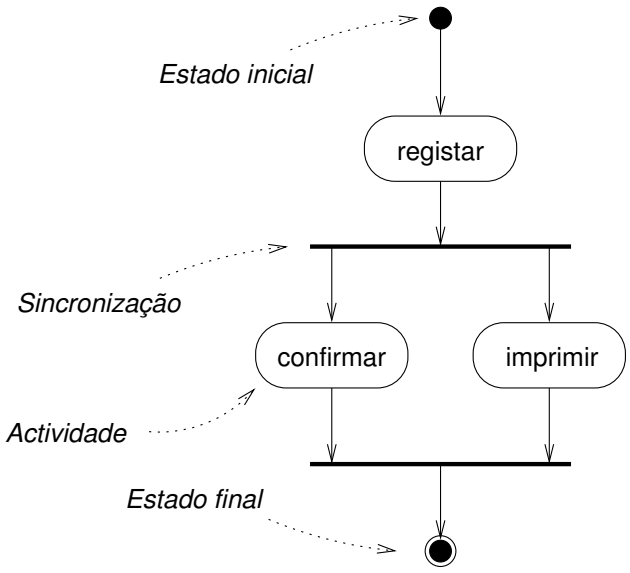
Diagramas de Statechart



- Modelam ciclo de vida de objectos no sistema.

Ver página 64

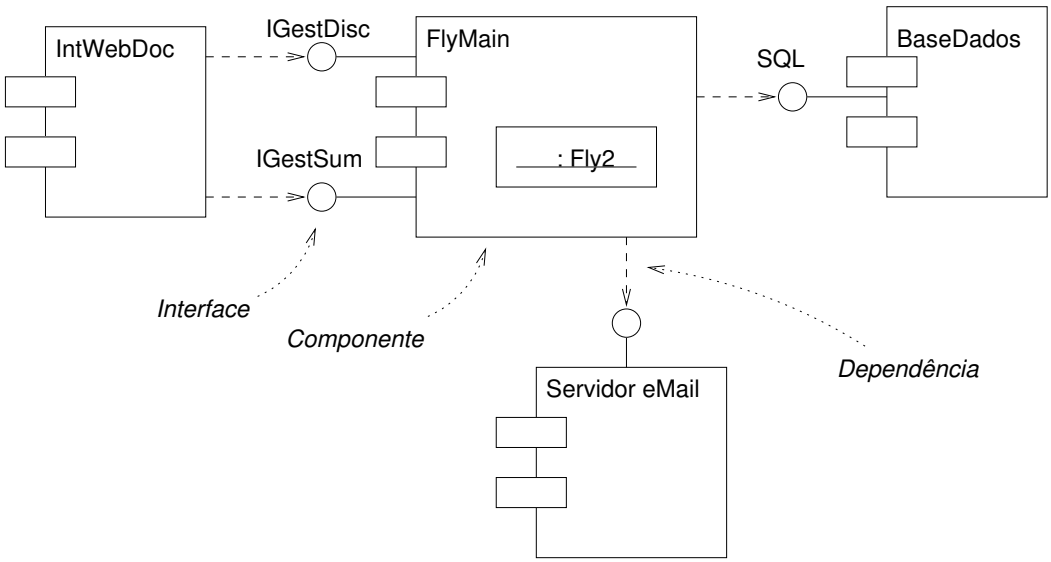
Diagramas de Actividade



- Modelam comportamento (ênfase actividades realizadas).

Ver página 79

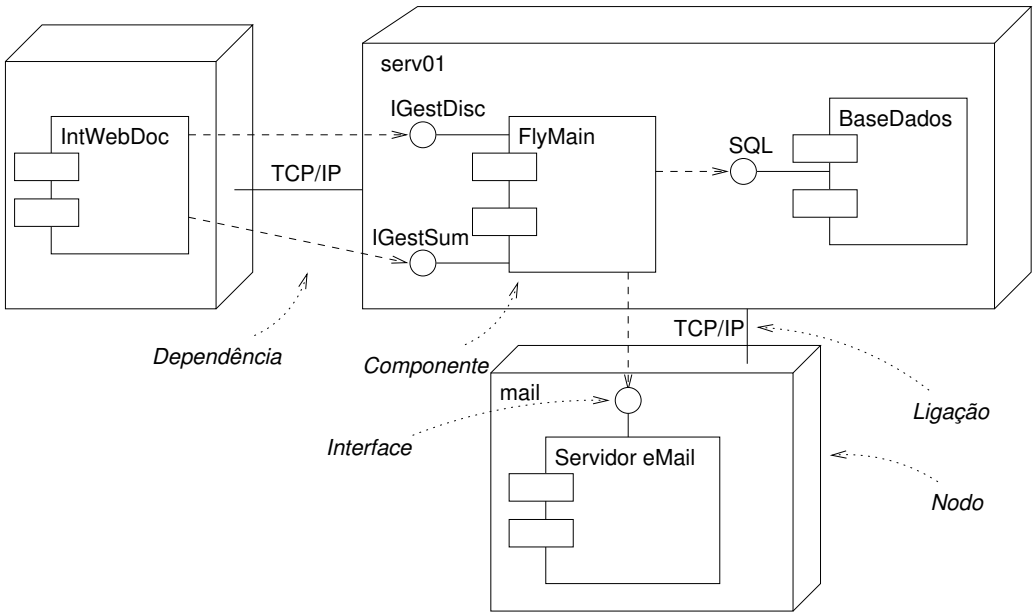
Diagramas de Componentes



- Definem como o sistema é construido a partir de componentes.

Ver página 64

Diagramas de Instalação



- Definem como o sistema deverá ser instalado.

Ver página 75

Meta-modelo do UML

Se o UML permite escrever modelos, poderemos modelar o UML?

- A definição do UML engloba quatro níveis de abstracção:
  - meta-meta modelo
    Define o elemento mais básico em que o UML se baseia: o conceito de *Coisa*.
  - meta modelo do UML
    Define o tipo de *Coisas* que podem ser utilizadas num modelo UML (por exemplo, o conceito de Classe é uma instância de *Coisa*).
  - modelos UML
    Os modelos UML que podemos escrever (por exemplo, a classe Sumário é uma instância do conceito de Classe).
  - modelos do utilizador
    Os modelos utilizados para mostrar instâncias concretas de modelos UML (cf. diagramas de objectos).

Mecanismos de extensão

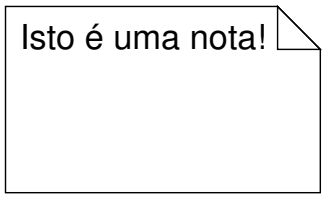
O meta-modelo define mecanismos que permitem aumentar a expressividade da linguagem.

- Restrições
  - Permitem definir restrições nos elementos de um modelo que vão para além das previstas de base.
  - Exemplo: {ordered} numa associação (para dizer, por exemplo, que a lista de sumários de uma disciplina está ordenada)
- Propriedades (*tagged values*)
  - Pares nome/valor que definem características adicionais de um elemento.
  - Exemplo: {Linguagem = Java} para definir que uma dada classe deverá ser implementada em Java.
- Estereótipos
  - Permitem efectuar a especialização semântica de elementos existentes.
  - Exemplo: <<interface>> (podem ter representação própria, neste caso: —○)

Decorações

Existem ainda dois mecanismos sem semãntica especial associada que podem ser utilizados para decorar (os elementos de) um diagrama.

- Notas
  - Utilizadas para adicionar comentários ao diagrama.
  - Podem estar associadas a um elemento específico (através de um traço interrompido).



- Compartimentos extra
  - Permitem adicionar informação directamente a um elemento de um diagrama.