Departamento de Informática, Universidade do Minho

http://www.di.uminho.pt

Desenvolvimento de Sistemas de Informação

http://sim.di.uminho.pt/ensino.php3

LESI - 4º Ano / 2º Semestre (5308O7)

LMCC - 4º Ano / 2º Semestre (7008N8 - Opção II)

2002/2003

José Creissac Campos

jose.campos@di.uminho.pt

Desenvolvimento de Sistemas de Informação

Escolaridade

2T + 2TP (3,5 creditos)

Equipa Docente

- José Creissac Campos – T e TP
  Contacto: jose.campos@di.uminho.pt / x4447
  Atendimento: 3ª 09h-11h / 4ª 11h-13h / 5ª 11h-13h
- António Ramires Fernandes – TP
  Contacto: af@di.uminho.pt
  Atendimento: 4ª 15h-17h

Programa

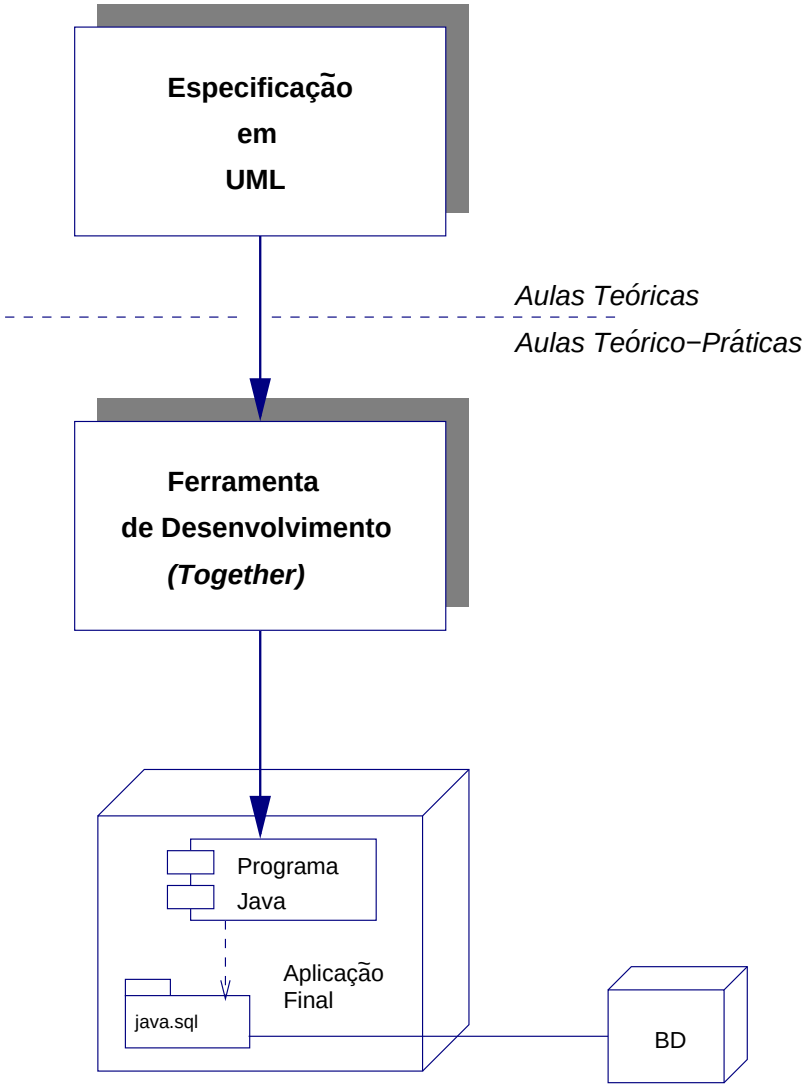
Aulas Teóricas

- Introdução aos Sistemas de Informação
- Modelação de Sistemas de Informação em UML

Aulas Teórico-Práticas

- JDBC: Acesso a Bases de Dados a partir de Java
- Estudo de Ferramentas de suporte ao desenvolvimento de SI utilizando UML.
  Together: www.togethersoft.com
  Levantar software e licença junto dos técnicos.

Esquema da Disciplina



Motivação

Estudos mostram que para grandes projectos (+50,000 linhas de código):

- 1/3 dos projectos é abandonado antes de estar terminado e apenas 20-25% dos projectos terminados é considerado bem sucedido;
- produtividade média está abaixo das 10 linha de código por dia e, em média, encontram-se 60 erros por cada 10,000 linhas de código;
- custo de manter o software ultrapassa o dobro do custo de desenvolvimento e, em média, a duração dos projectos ultrapassa em 50% os prazos estimados.

- Exemplos: aeroporto de Denver, foguetão Arianne.

Desenvolvimento de software não pode ser encarado como *arte*, mas como Engenharia.
Necessitamos de métodos e ferramentas apropriados.
Em DSI apresenta-se uma proposta, existem outras!

UML

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

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

Alguns Exemplos de Diagramas

Diagrama de Use Case

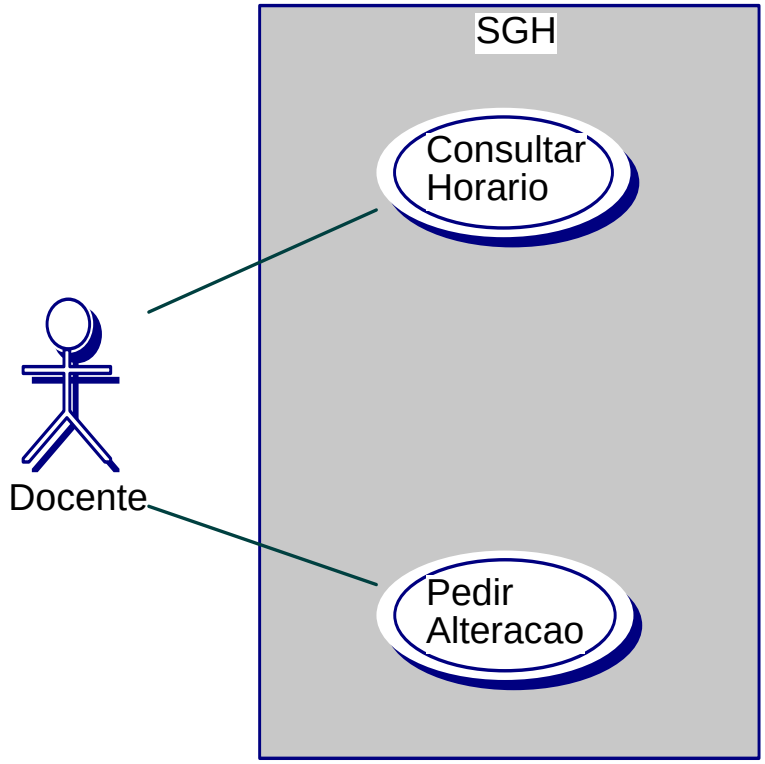


Diagrama de Classe

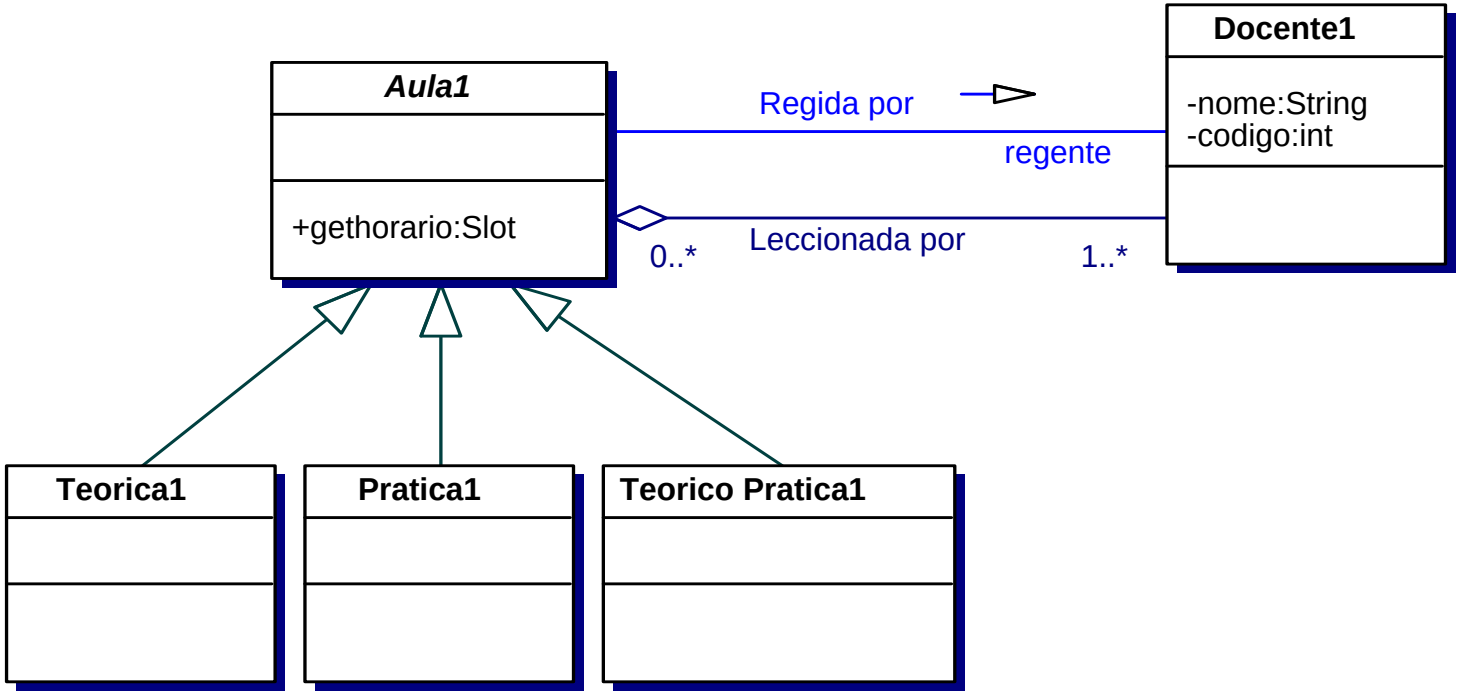
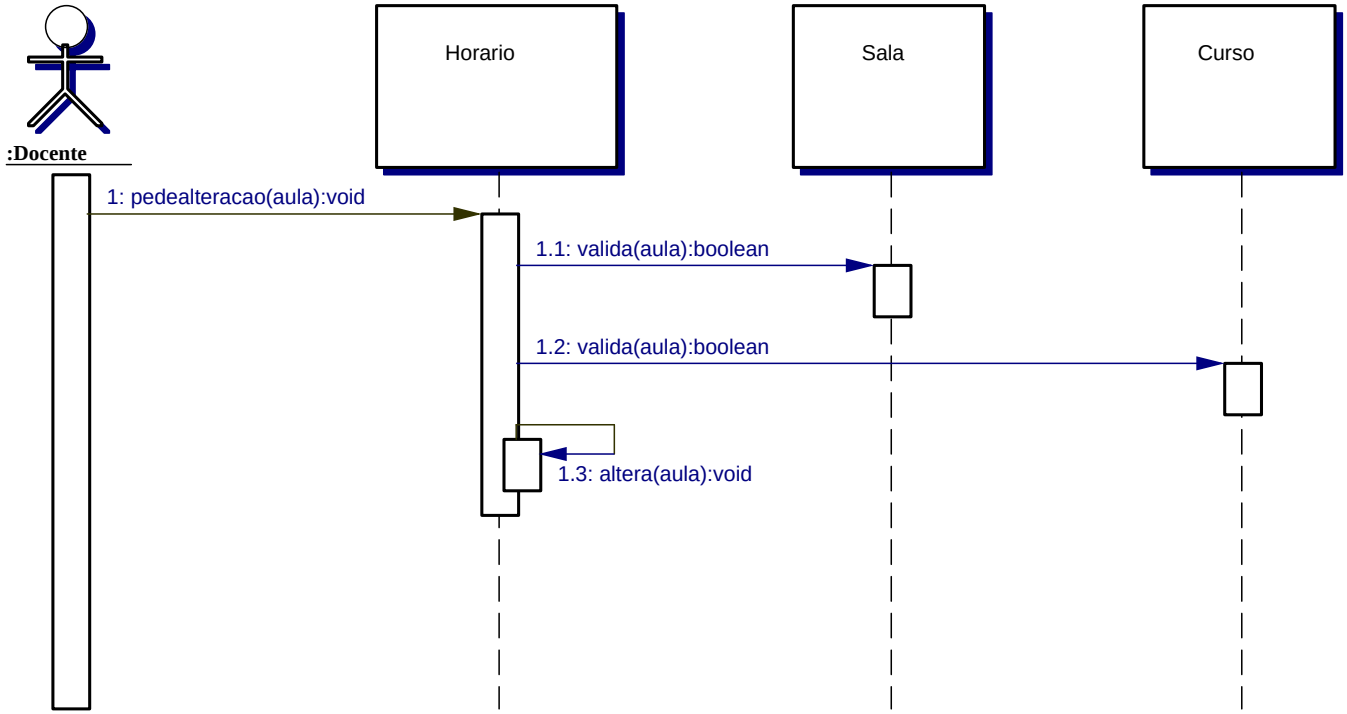


Diagrama de Sequência



Aulas Teóricas

- Introdução aos Sistemas de Informação (SI)
  - Conceitos base — O que é um SI?
  - O Processo de Desenvolvimento de Software — diferentes abordagens
- Modelação de SI em UML
  - História do UML
  - Visão Geral — os diferentes níveis de modelação
  - Modelação Comportamental
    - Diagramas de *Use Case*
    - Diagramas de Interacção (sequência/colaboração)
    - Diagramas de Estado
    - Diagramas de Actividade
  - Modelação Estrutural
    - Diagramas de Classe (revisão de conceitos OO)
    - Diagramas de *Package*
  - Modelação Arquitectural
    - Diagramas de *Deployment*

Bibliografia

- G. Booch, J. Rumbaugh, I. Jacobson. The Unified Modeling Language User Guide. Reading, MA, Addison-Wesley, 1998.
- J. Rumbaugh, I. Jacobson, G. Booch. The Unified Modeling Language Reference Manual. Reading, MA, Addison-Wesley, 1999.
- M. Fowler. UML Distilled. Addison-Wesley, 1998.
  (bom livro! mas notação desactualizada...)
- P. Stevens, R. Pooley. Using UML. Addison-Wesley, 2000.
- TogetherSoft Corporation. Together User's Guide, version 6.0.1. 2002.
  (vem com o Together)

Em português:

- F. Mário Martins. Notas Teóricas de DSI. 1997.
- M. Nunes, H. O'Neill. Fundamental do UML. FCA. 2001.

Avaliação

- Exame (≥ 8 — 60%)
- Trabalho Prático (40%)
- Nota: $.6\text{Exame} + .4\text{Trabalho} \geq 10$

Trabalho Prático

- Grupos de 3 elementos
- A realizar por fases durante o semestre.
  - Cada fase vale uma parte da nota final.
  - Entregas fora de prazo:
    - até à data da fase seguinte — penalização de 50%
    - depois da data da fase seguinte — 0 (zero) valores
    - a última fase não terá tolerância
- Será conhecido a 11 de Março.

A fazer...

Esta semana

- Inscrição nos turnos TP.
- Inscrição dos grupos de trabalho.

Procurar folhas na recepção do DI, a partir de Quarta-Feira.

A ver...

Ao longo do semestre

- Página da cadeira (a partir de `http://sim.di.uminho.pt/ensino.php3`)
- Fórum DSI do CIUM (recebem?)

Introdução aos Sistemas de Informação

Sumário

- Definição de Sistema de Informação.
- Metodologias de Desenvolvimento.
- Processos de desenvolvimento.
- Introdução ao UML.

O que é um Sistema de Informação?

sistema, *s. m.* Reunião de parte ligadas entre si, formando uma estrutura complexa. Conjunto de meios, processos destinados a produzir um resultado = MÉTODO.

informação, *s. f.* Conjunto de conhecimentos reunidos sobre um determinado assunto; documentação.

Dicionário da Língua Portuguesa Contemporânea, Academia de Ciências de Lisboa

Sistema de Informação:

Fornece um meio para reunir informação de uma dada organização, fornecendo procedimentos para registo e disponibilização dessa informação com o objectivo de auxiliar a organização nas suas actividades.

- não necessariamente software;
- vários tipos...

Podem ser definidos segundo duas perspectivas:

- visão funcional — meio tecnológico para registar, guardar e disseminar informação;
- visão estrutural — colecção de pessoas, processos, dados, modelos, tecnologia e linguagens;

Alguns tipos de SI

- SI Transaccionais
- SI de Tempo Real
- SI baseados em Bases de Dados
- SI de Apoio à Decisão
- SI Periciais (Baseados em Conhecimento)
- SI de Escritório

Atenção: taxonomias são sempre redutoras!

Sejam de que tipo forem, os SI:

- existem para auxiliar a organização
  $\longrightarrow$ mais um meio a juntar a tantos outros;
- devem ser concebidos em função das necessidades da organização
  $\longrightarrow$ vão ser usados pela organização, não por quem os concebeu.

É necessário perceber a organização para conceber um bom SI.

- Quais as actividades da organização;
- Qual a informação que flui na organização;
- Quais as tarefas das pessoas da organização.

Entender o problema antes de desenvolver a solução!

O que é um bom Sistema (de Informação)

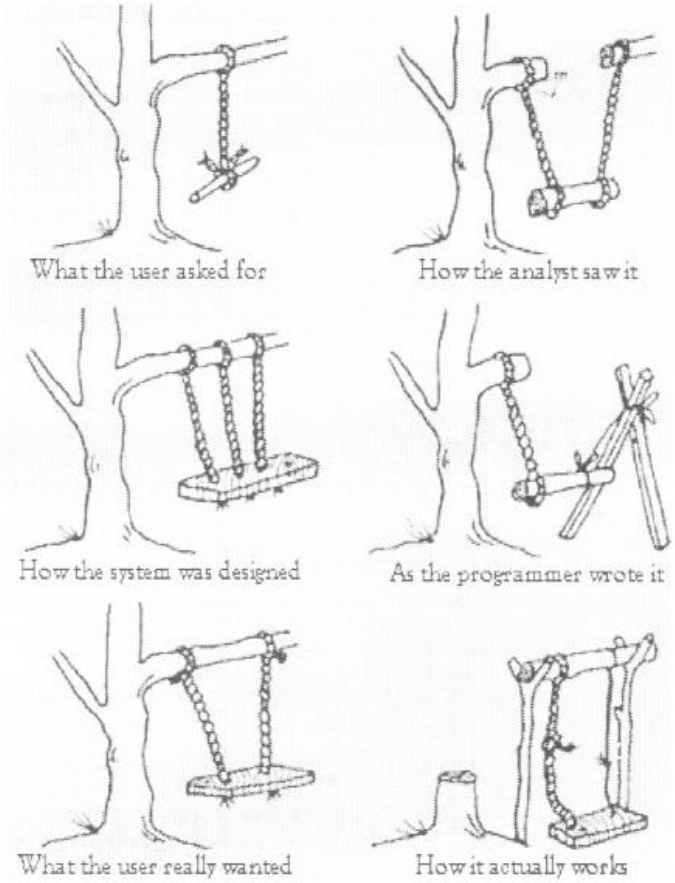
Aquele que satisfaz as necessidades dos seus utilizadores:

- **útil e utilizável** — bom software facilita a vida dos utilizadores → deve responder às necessidades reais dos utilizadores;
- **confiável** — sem *bugs*!
- **flexível** — as necessidades dos utilizadores mudam, os *bugs* têm que ser corrigidos → *bug* do ano 2k veio mostrar a falta de flexibilidade de muitos sistemas!;
- **acessível** (financeiramente) — quer na compra como na manutenção → fácil e rápido de desenvolver;
- **disponível** — se não está disponível nada mais interessa! → está disponível a tempo e horas? está disponível na plataforma tecnológica pretendida?

Como são bons Sistemas (de Informação) desenvolvidos?

O que a organização pretende é um SI (i.e. código) eficiente.

Cabe-nos desenvolve-lo, mas existem riscos:



- Riscos associados aos requisitos.
- Riscos tecnológicos.
- Riscos de competência.
- Riscos políticos.

Riscos associados aos requisitos

- O que é que os utilizadores querem?
- Que linguagem falam? / Quais os conceitos?

É necessário comunicar com os peritos da organização para:

- compreender que tarefas o sistema deve suportar;
- compreender como o sistema *encaixa* nas actividades da organização.

Um dos maiores desafios é construir o sistema *certo*.

Riscos tecnológicos

- Qual a tecnologia mais apropriada?
- Como controlar a complexidade?

É necessário validar as soluções tecnológicas o mais cedo possível.

Riscos de competência

- É necessário saber-se o que se está a fazer (obviamente?).
- Exemplo da OO: fácil de aprender/difícil de dominar.

Riscos políticos

- Por muito bom que seja o SI só terá sucesso tiver o apoio das pessoas *certas*.

É necessária uma metodologia de desenvolvimento

Método

Processo de desenvolvimento + Linguagem de modelação

Processo de desenvolvimento

- Um conjunto de regras que definem como o desenvolvimento de um sistema deve ser efectuado.
- Normalmente inclui uma descrição dos documentos que devem ser produzidos e em que ordem.
- Pode incluir indicação da linguagem de modelação a ser utilizada.

Linguagem de modelação

- Permite-nos escrever modelos da solução a desenvolver.
- Um modelo é uma representação abstracta de uma especificação ou sistema, efectuada de um ponto de vista específico.
- Usualmente diagramática, mas pode ser textual.
- Uma linguagem de modelação tem:
  - sintaxe – regras que definem quais os diagramas que são legais;
  - semântica – regras que definem o significado de diagramas legais.

Modelos UML são **diagramáticos**.

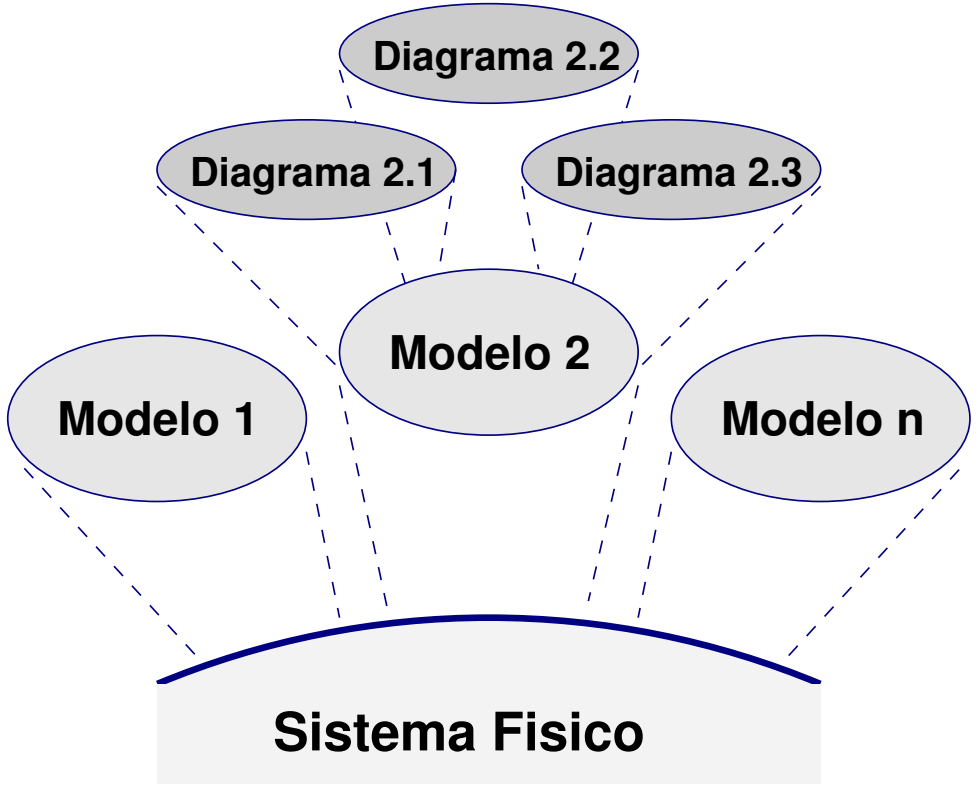
Tipos de modelos

- *Preditivos*
  Utilizados para prever o comportamento do sistema.
- *Normativos*
  Utilizados para definir comportamentos adequados do sistema.
- *Descritivos*
  Utilizados para descrever a estrutura e comportamento do sistema.

De um modo geral os modelos do UML são **descritivos**.

- Vários modelos para representar a mesma realidade.
- Vários diagramas para representar o mesmo modelo.
  (os diagramas não são os modelos!)

Os diagramas não são os modelos!



Vantagens da utilização de modelos

- Ajudam a **compreender** a realidade.

  Sendo abstracções da realidade, os modelos permitem descrever o que é considerado essencial num dado contexto, escondendo detalhes desnecessários/irrelevantes nesse contexto.

- Ajudam a **comunicar** ideias de forma simplificada.

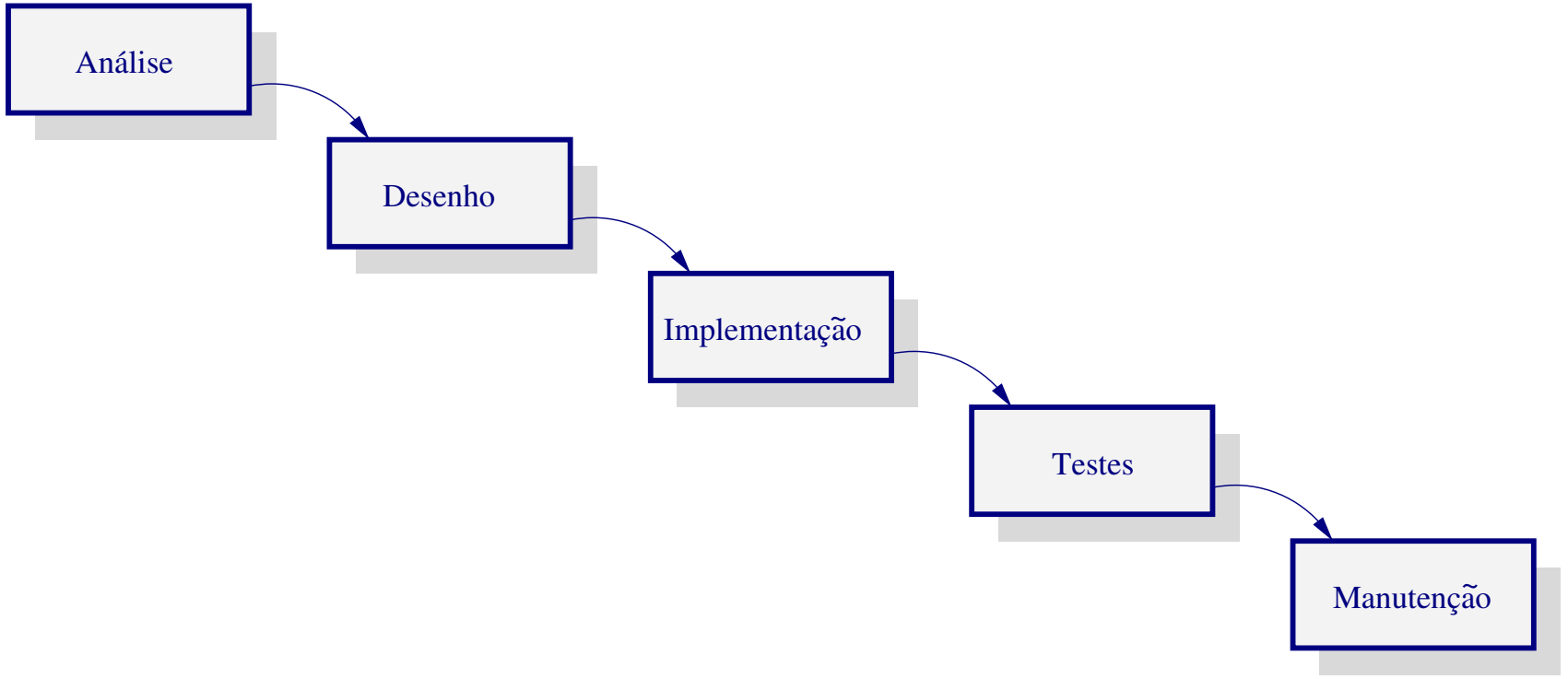
  Sendo simplificações da realidade, permitem comunicar apenas os aspectos pretendidos.

- Ajudam a **documentar** as decisões tomadas durante o desenvolvimento.

  Os modelos desenvolvidos constituem uma base documental para a descrição do processo de desenvolvimento.

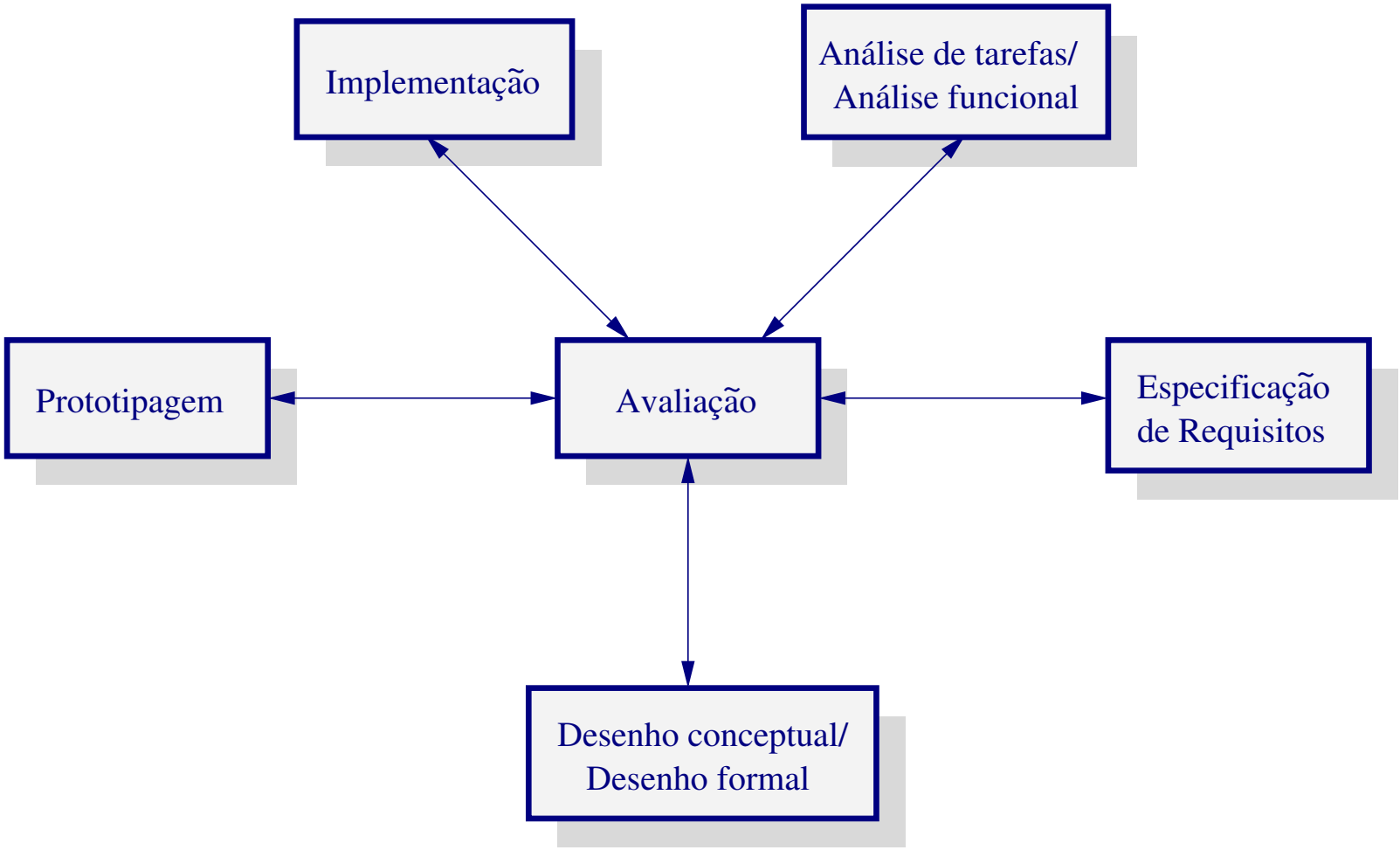
Métodos de desenvolvimento

Modelo Waterfall



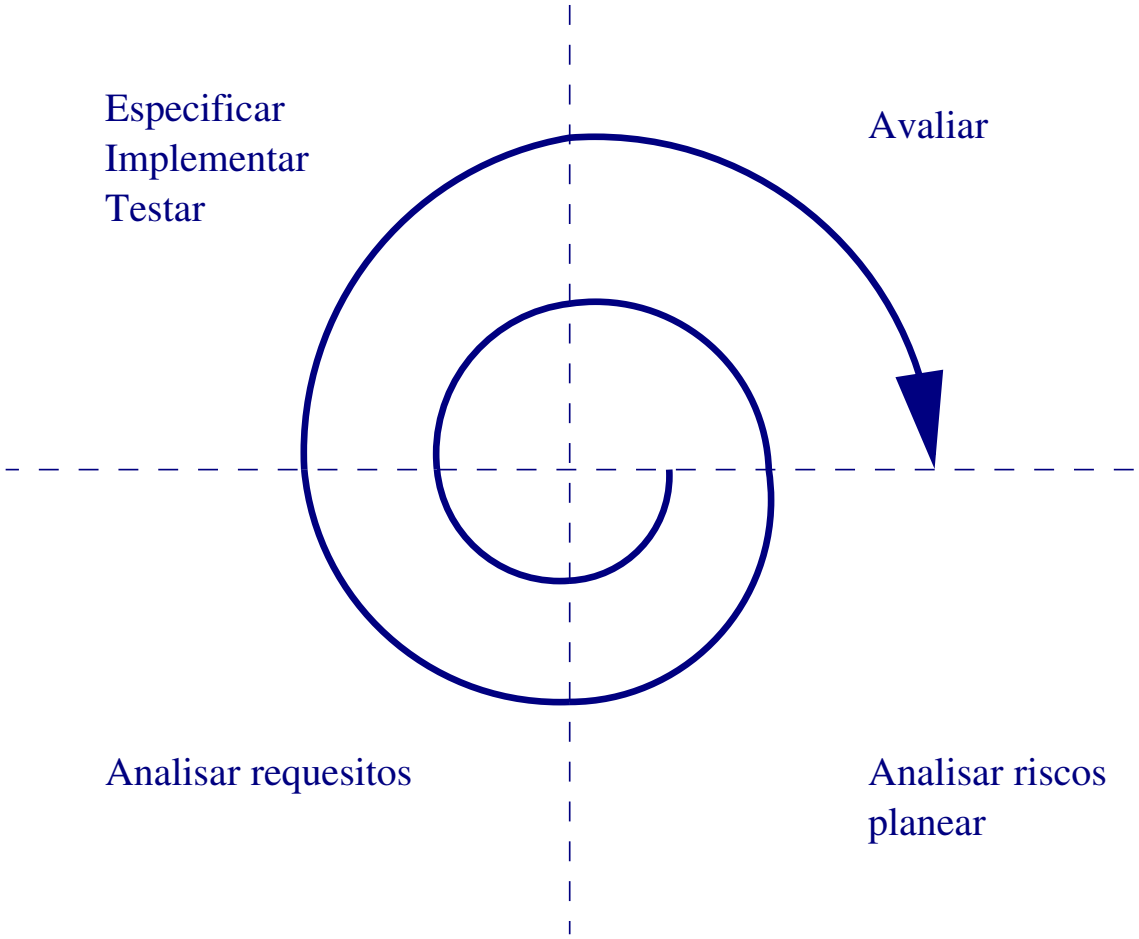
- Define o método como uma série de etapas executadas sequencialmente.
- Assume que é sempre possível tomar as decisões mais correctas → irrealista!

Modelo em Estrela



- Este modelo coloca em destaque a necessidade de avaliar o sistema em todas as fases de desenvolvimento.

Modelo em espiral

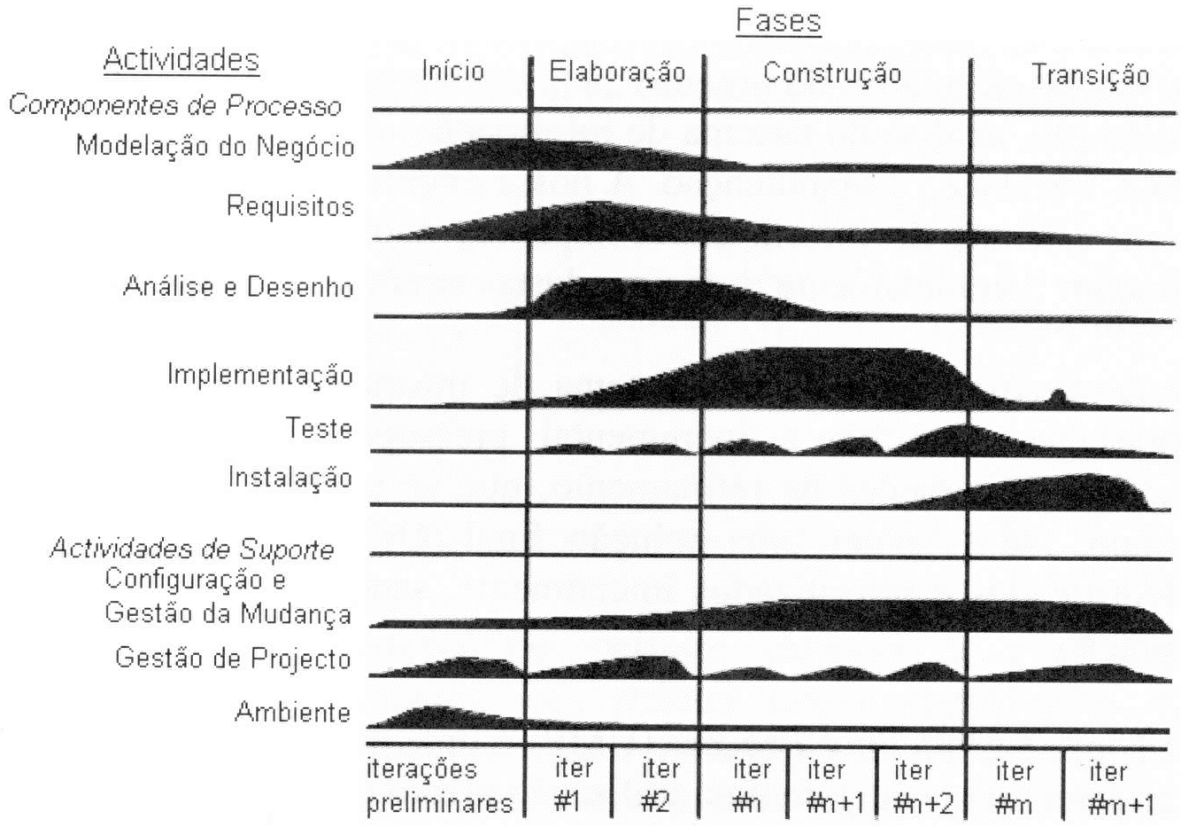


- Neste modelo reconhece-se a necessidade de iterar para controlar riscos.

Uma proposta de método

Um método iterativo e incremental.

- Início (Organização)
- Elaboração
- Construção (1..n)
- Transição



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.

A nossa linguagem de modelação vai ser o UML

- UML foi pensado para o desenvolvimento de sistemas orientados aos objectos
  → permite explorar o paradigma OO (cf. *riscos tecnológicos*)
- UML possibilita o trabalho a diferentes níveis de abstracção
  → facilita a comunicação (cf. *riscos de requisitos*)
- UML não é uma linguagem, mas um conjunto de linguagens
  → inclui modelos para as diferentes fases do desenvolvimento.

Entender o problema, pensar (modelar) a solução!

Modelação de Sistemas de Informação em UML

Breve história do UML

- 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
- 2003 — versão actual é o UML 1.3, está em preparação o UML 2.0

`www.uml.org`

Alguns Diagramas UML

Diagrama de Use Case

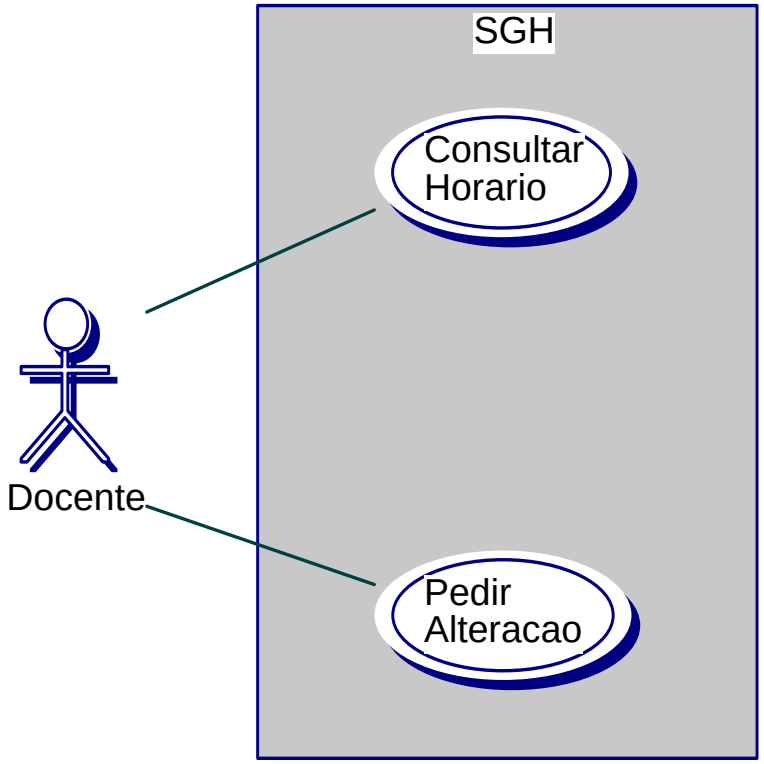


Diagrama de Classe

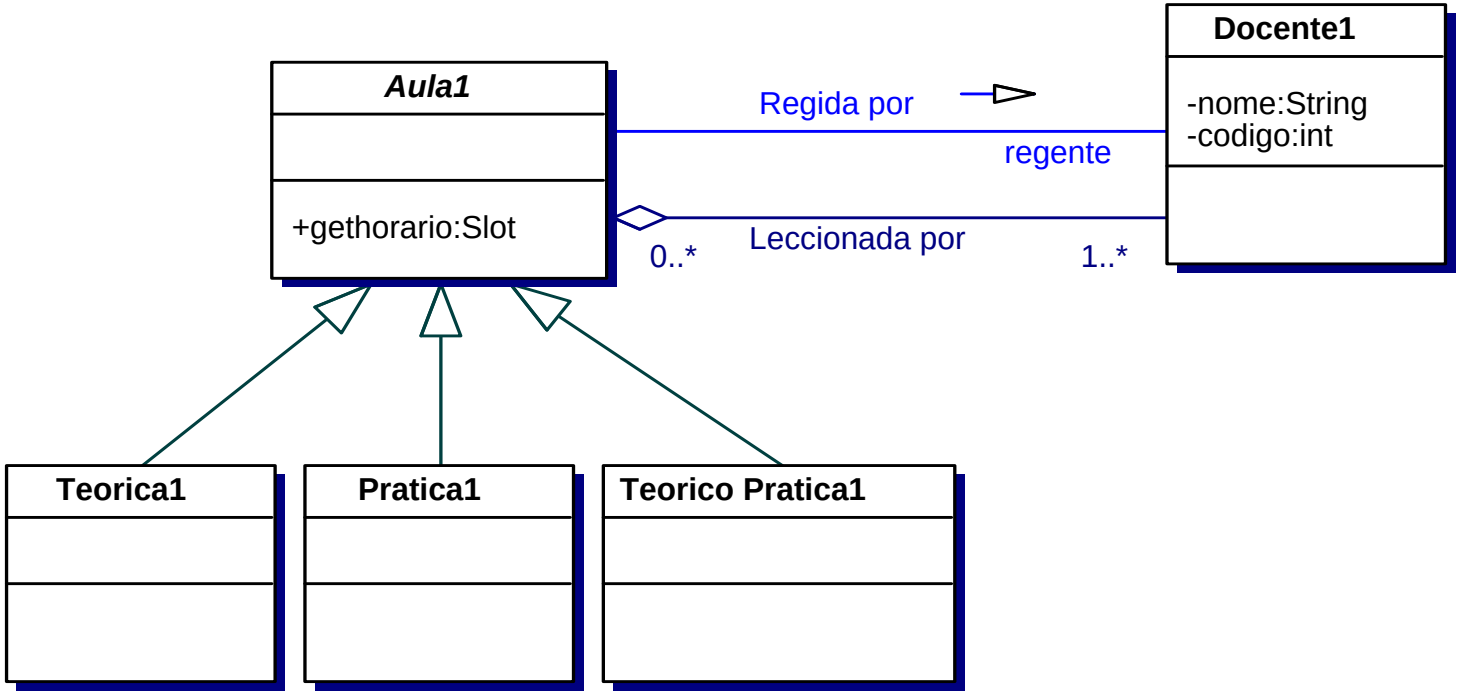
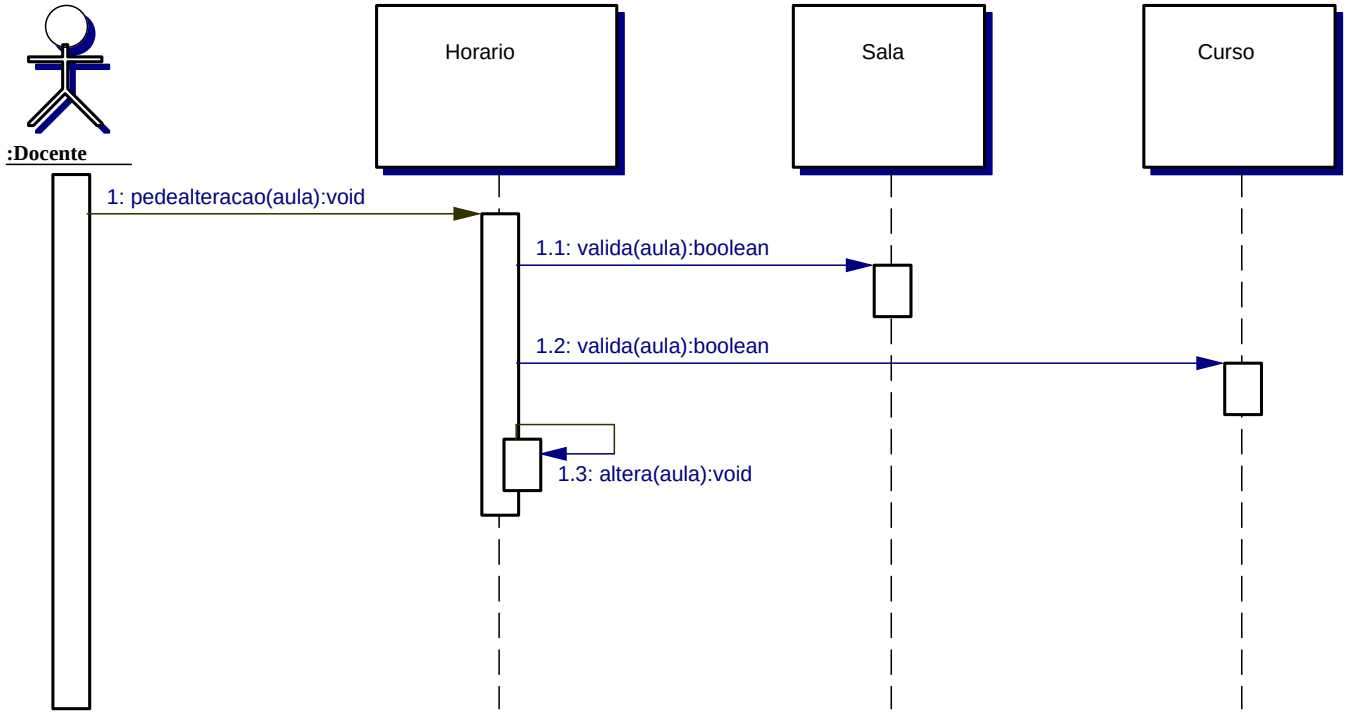


Diagrama de Sequência



Diagramas de Use Case

Definição dos requisitos do sistema. Duas abordagens possíveis:

- Visão estrutural — interna
- Visão orientada às tarefas — externa

Visão Estrutural (OO)

- Definir classes;
- Definir métodos das classes;
- Definir interface com o utilizador (comportamento do sistema face ao utilizador).

Problemas: O que interessa ao utilizador é o comportamento do sistema, no entanto a interface com o utilizador só é definida no final do processo.

- Perigo de o sistema não fornecer toda a funcionalidade pretendida;
- Perigo de o sistema fornecer funcionalidade não pretendida (= desperdício de trabalho).

Visão orientada às tarefas

- Identificar *Actores* — quem vai interagir com o sistema?
- Identificar *Tarefas* — o que se pretende do sistema?
- Identificar classes de suporte à realização das tarefas.

Vantagens:

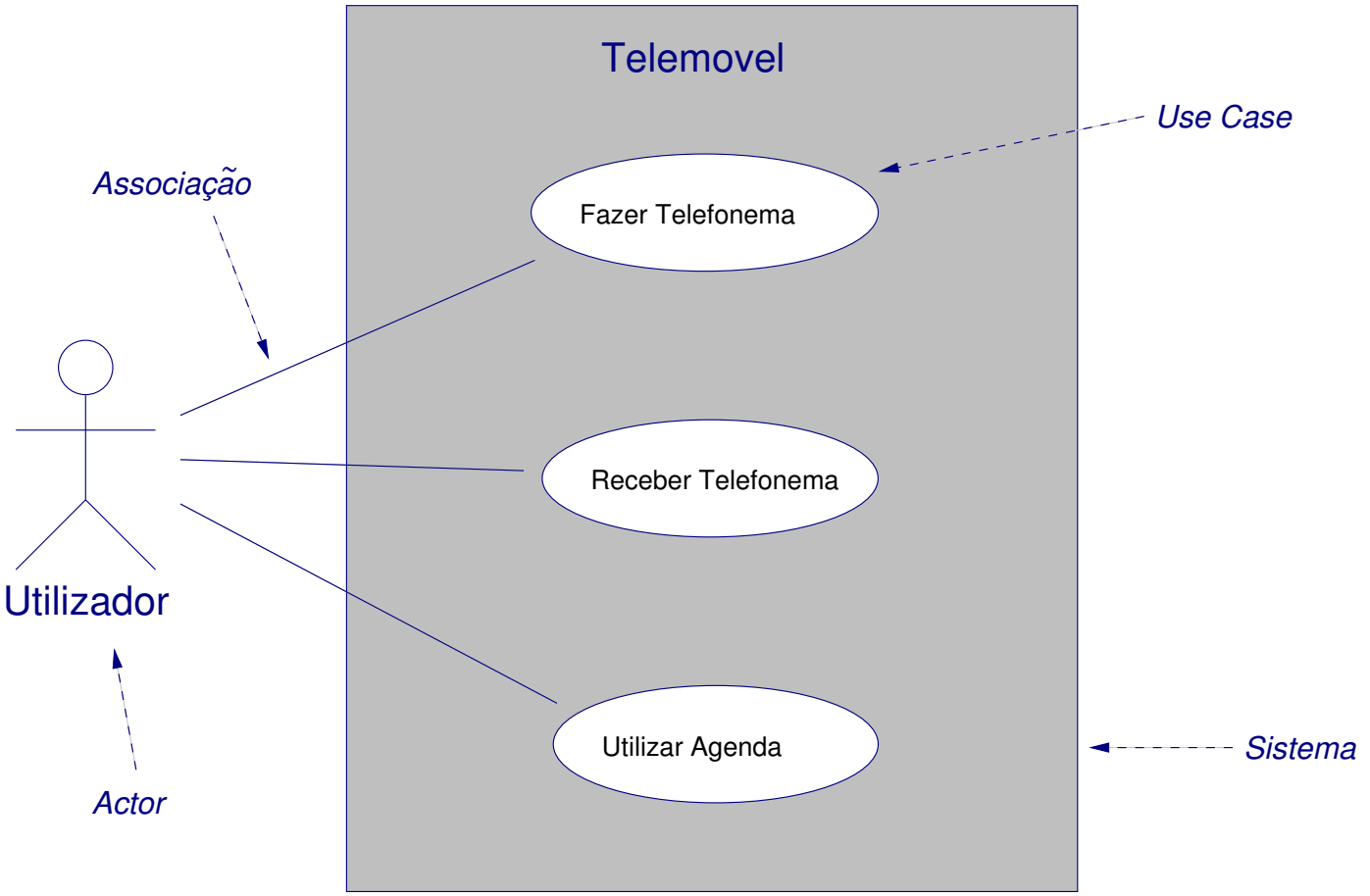
- Não há trabalho desnecessário;
- S.I. suporta as tarefas do cliente.

Para expressar as tarefas utilizamos Diagramas de *Use case*.

Diagramas de Use Case

- Modelam o contexto geral do **sistema**. Quais os **actores** que com ele se relacionam e que **tarefas** deve suportar.

Exemplo


Telemovel
Use Case
Associação
Fazer Telefonema
Receber Telefonema
Utilizador
Utilizar Agenda
Sistema
Actor

Sistema

- define as fronteiras do sistema

Use Case

- Uma unidade coerente de funcionalidade — um serviço (neste caso, uma tarefa)
- define um comportamento do sistema sem revelar a estrutura interna — apenas mostra a comunicação entre sistema e actores
- deve incluir o comportamento normal, bem como variações (erros, etc.)
- o conjunto de todos os *use case* define a funcionalidade do sistema
- para já vamos definir o comportamento com texto *estruturado*:

Use case: Fazer Telefonema

Pré-condição:

Telefone ligado e em descanço

Comportamento Normal:

1. Utilizador marca numero e pressiona OK
3. Telefone transmite sinal de chamada
4. Utilizador aguarda
5. Telefone estabelece ligação
6. Utilizador fala
7. Utilizador pressiona tecla C
8. Telefone desliga chamada

Comportamento alternativo:

3. Telefone transmite sinal de ocupado
4. Utilizador presssiona C
5. Telefone desliga chamada

Comportamento alternativo:

3. Telefone rejeita chamada

Pós-condição:

Telefone ligado e em descanço

Actor

- uma abstracção para uma entidade fora do sistema
- um actor modela um propósito — pode não mapear 1 para 1 com entidades no mundo real
- um actor não é necessariamente um humano — pode ser um computador, outro sistema, etc.
- cada actor define um conjunto de papeis que utilizadores do sistema podem assumir
- o conjunto de todos os actores definem todas as formas de interacção com o sistema

Associação

- representa comunicação entre o actor e o sistema — através de *use cases*

Exemplo

> Definir um sistema de gestão de horários. O sistema deve permitir fazer a gestão dos horários de um dado departamento. Para esse efeito o sistema terá que possuir informação sobre os docentes, cursos e salas do departamento. Em cada departamento existe um gestor de horários responsável pela definição dos horários. O sistema deve permitir que os docentes formulem pedidos de alteração e consultas aos horários por parte de docentes e alunos.

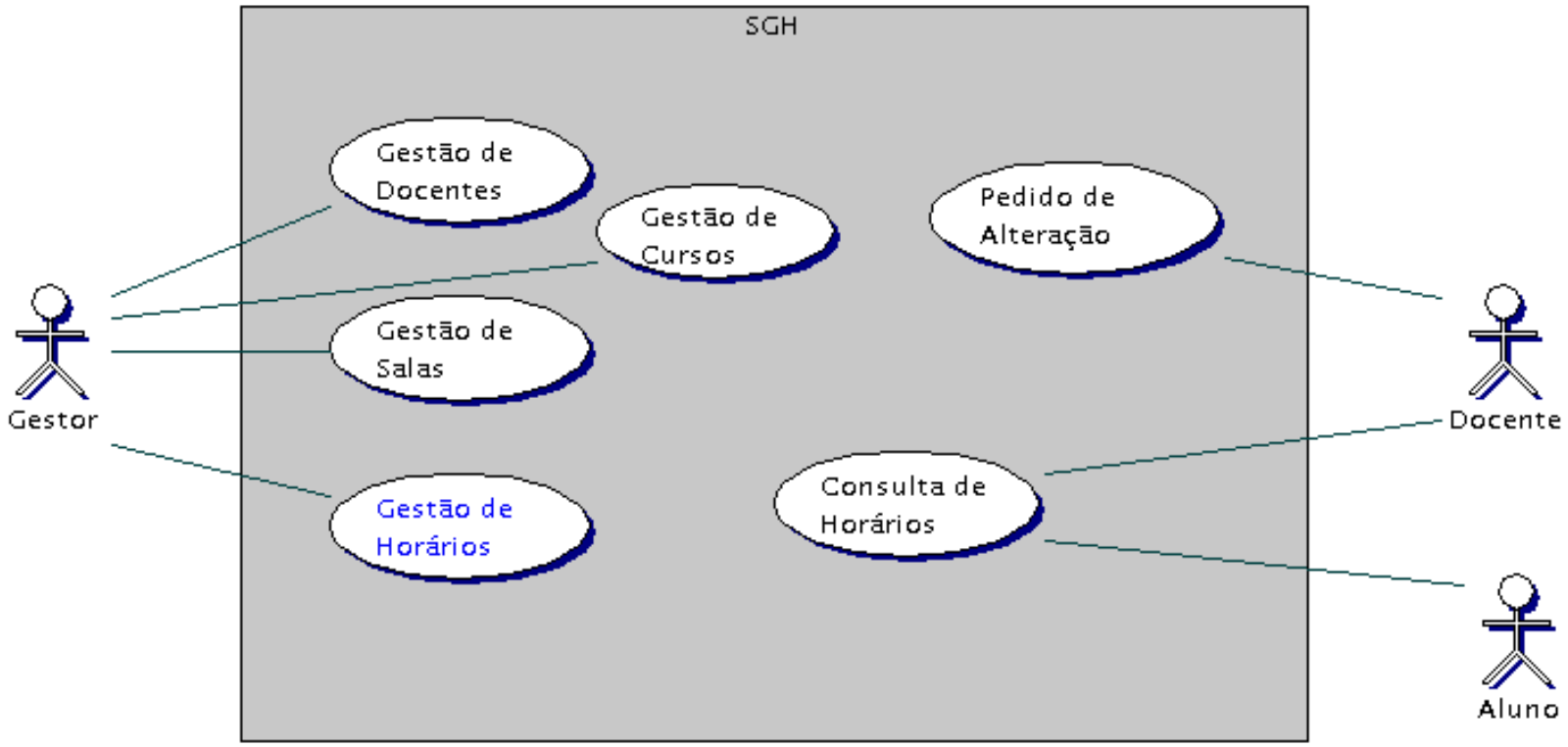
Etapas a cumprir:

1. Identificar actores
2. Identificar *use cases*

Actores:

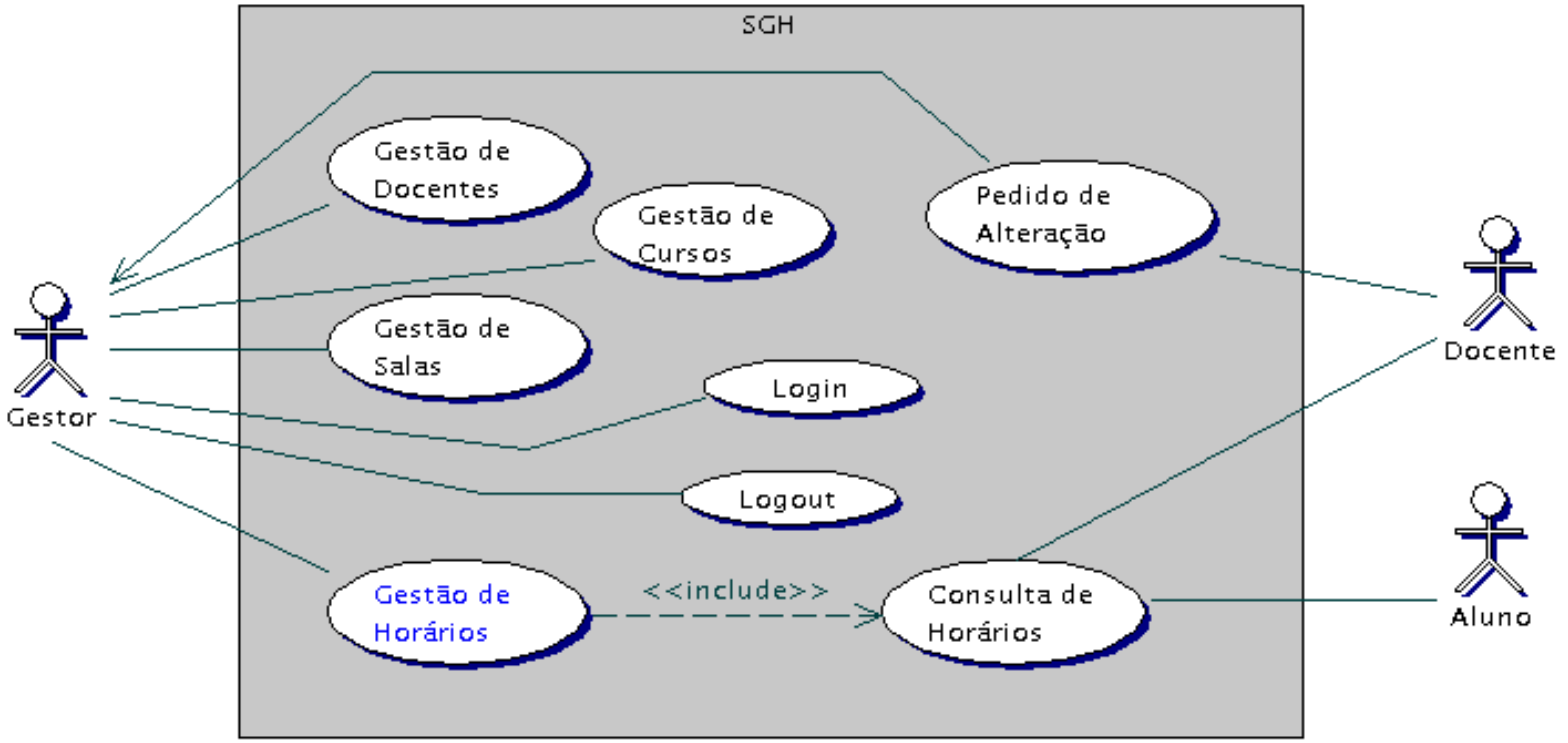
- Gestor
- Docente
- Aluno

Visão geral – versão 1



- falta mecanismo de autenticação do gestor
- falta alguma forma de comunicar ao gestor os pedidos de alteração

Visão geral – versão 2



- são adicionadas pré-condições nos *use case* Gestão de Docentes, Cursos, Salas e Horários a exigir que tenha sido feito login.

Use case: Pedido de Alteração

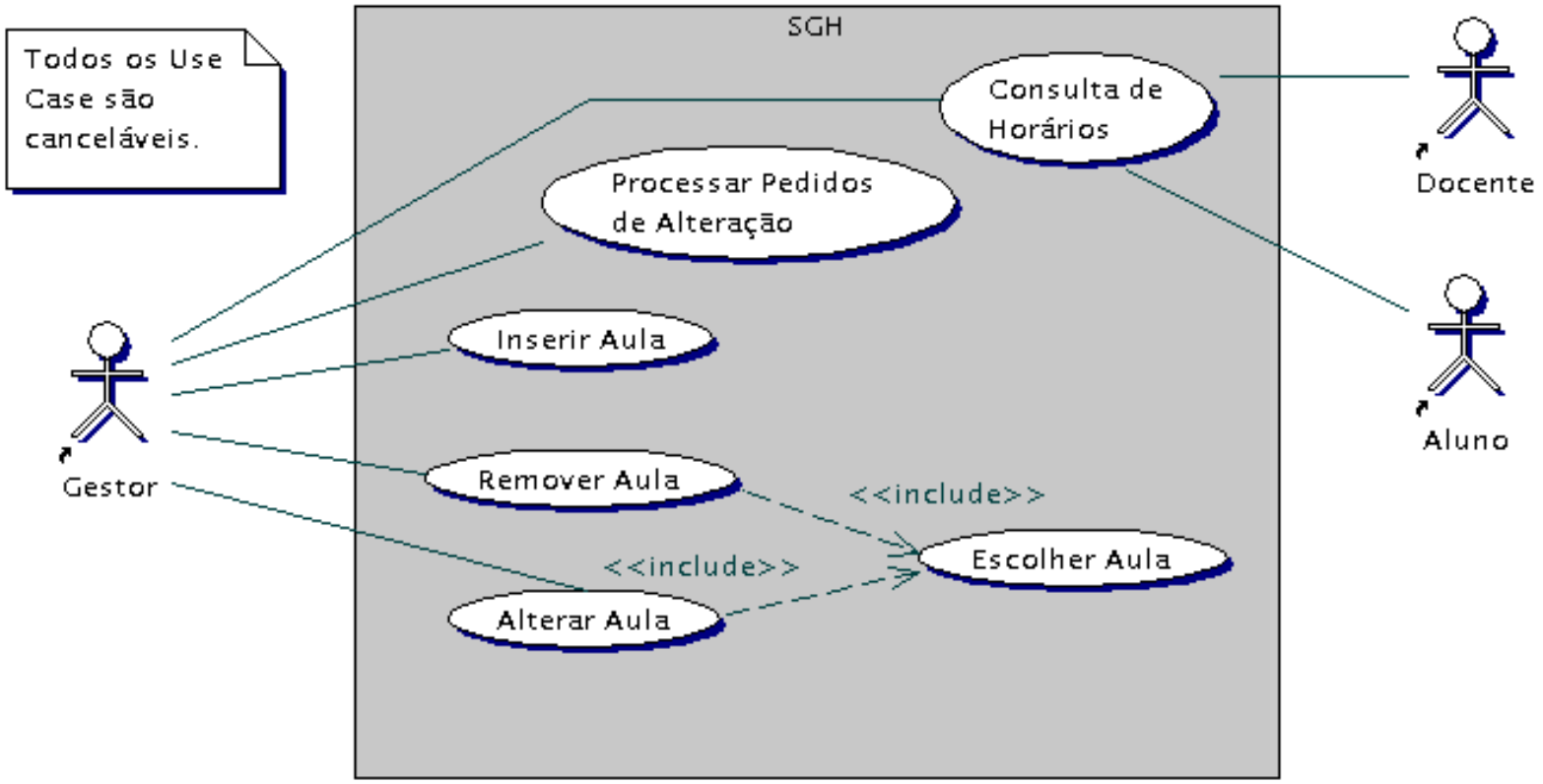
Comportamento Normal:

1. Docente escolhe aula
2. Sistema apresenta informação da aula
3. Docente faz alterações
4. Docente confirma pedido de alteração
5. Sistema envia aviso ao Gestor

Comportamento alternativo:

Diálogo pode ser cancelado até ao passo 5

Gestão de Horários



Use case: Escolher Aula

Comportamento Normal:

1. Sistema apresenta listas de cursos, anos e disciplinas
2. Actor selecciona curso, ano e disciplina enquanto o Sistema actualiza as listas em função dos valores já seleccionados
3. Sistema apresenta lista de aulas
4. Actor escolhe uma aula

Comportamento alternativo:

Diálogo pode ser cancelado

Use case: Inserir Aula

Pré-condição:

Gestor fez login

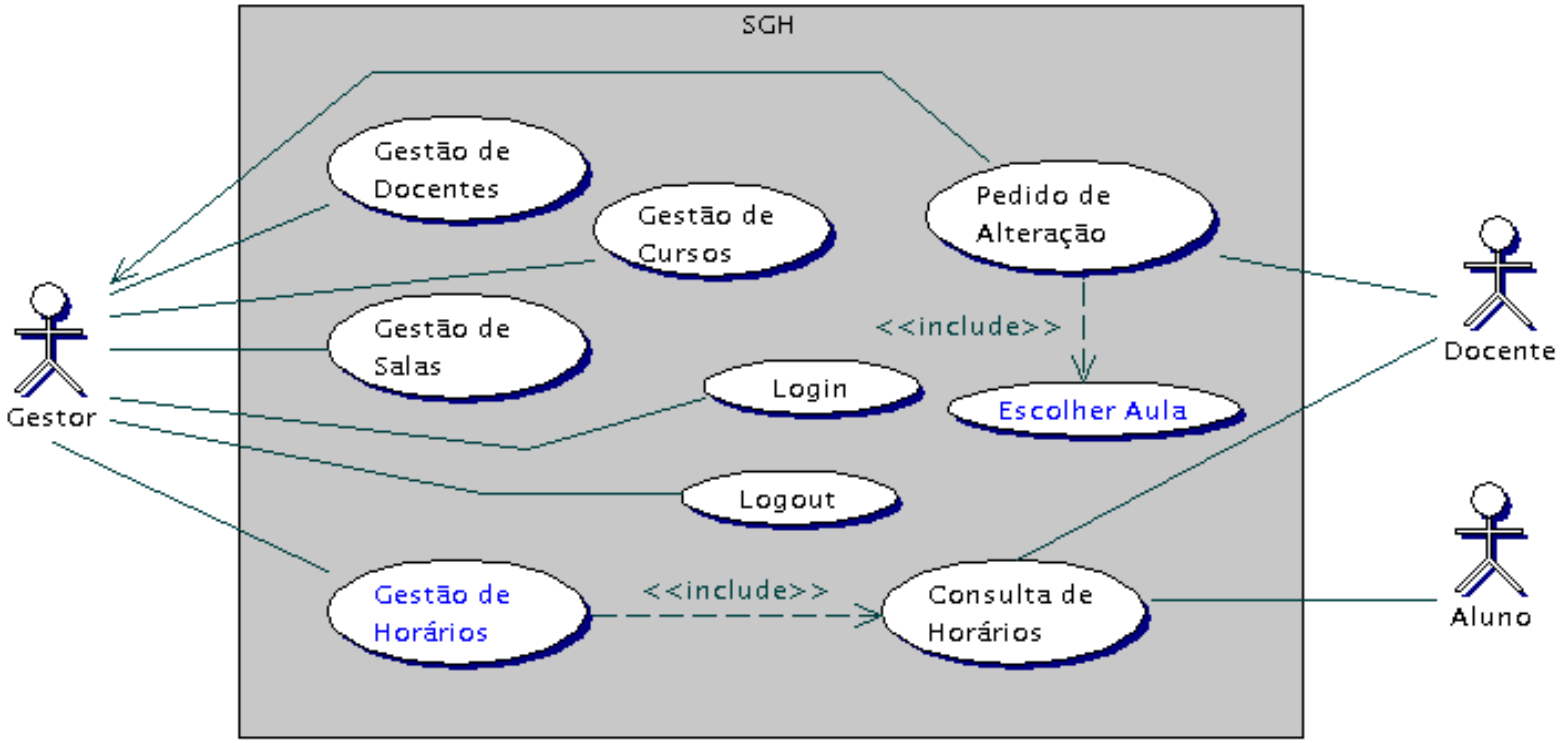
Comportamento Normal:

1. Sistema apresenta listas de docentes, disciplinas, aulas e salas
2. Gestor selecciona docente, disciplina, aula e sala enquanto o Sistema vai apresentando as horas em que é possível adicionar aulas
4. Docente selecciona uma hora e confirma inserção

Comportamento alternativo:

Diálogo pode ser cancelado

Visão geral – versão 3



Conteúdo