Análise e Design Orientado a Objetos utilizando UML (*Unified Modeling Language*)

Carga Horária: 72 horas

Professor *Sergio Akio Tanaka*
e-mail **sergio.tanaka@unifil.br**

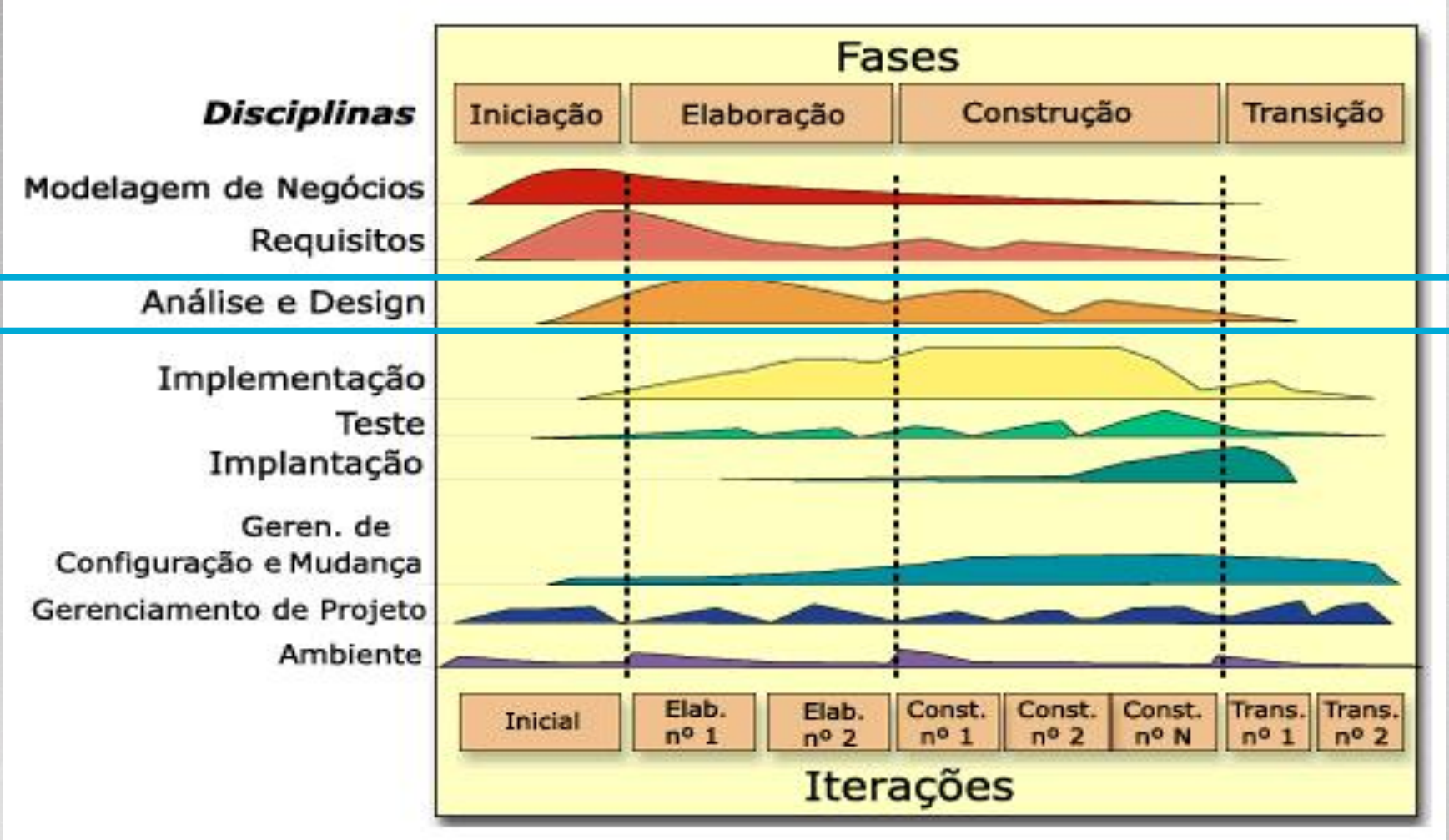
Apresentação da NOTAÇÃO UML *(Unified Modeling Language)*

Apresentação da UML

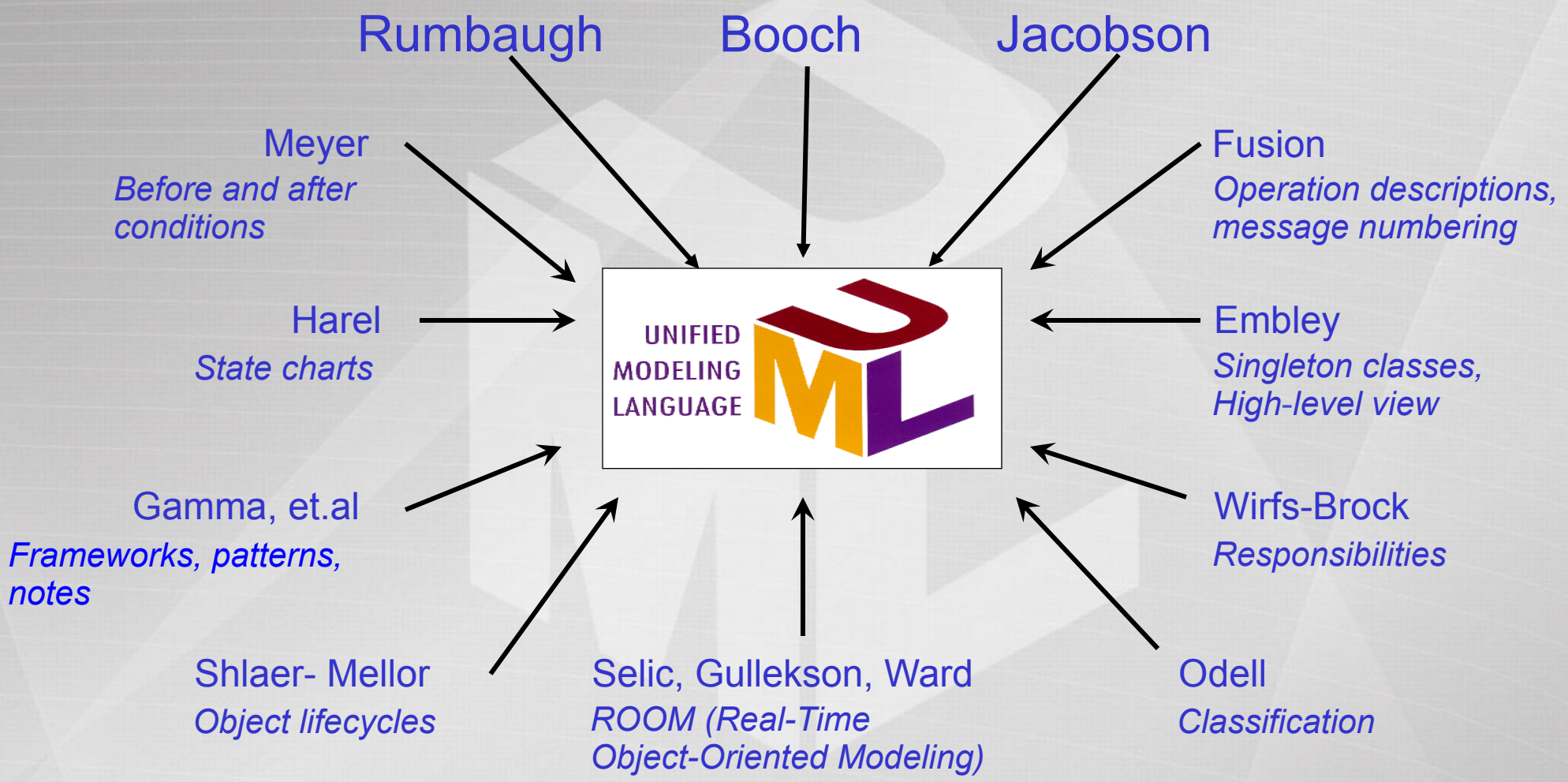
UML, Linguagem formada pela junção das melhores técnicas de modelagem (OMG 2008).

UML, Linguagem de Modelagem Unificada, é uma linguagem gráfica para visualização, especificação, construção e documentação de artefatos de sistemas complexos de software (BOOCH 2006).

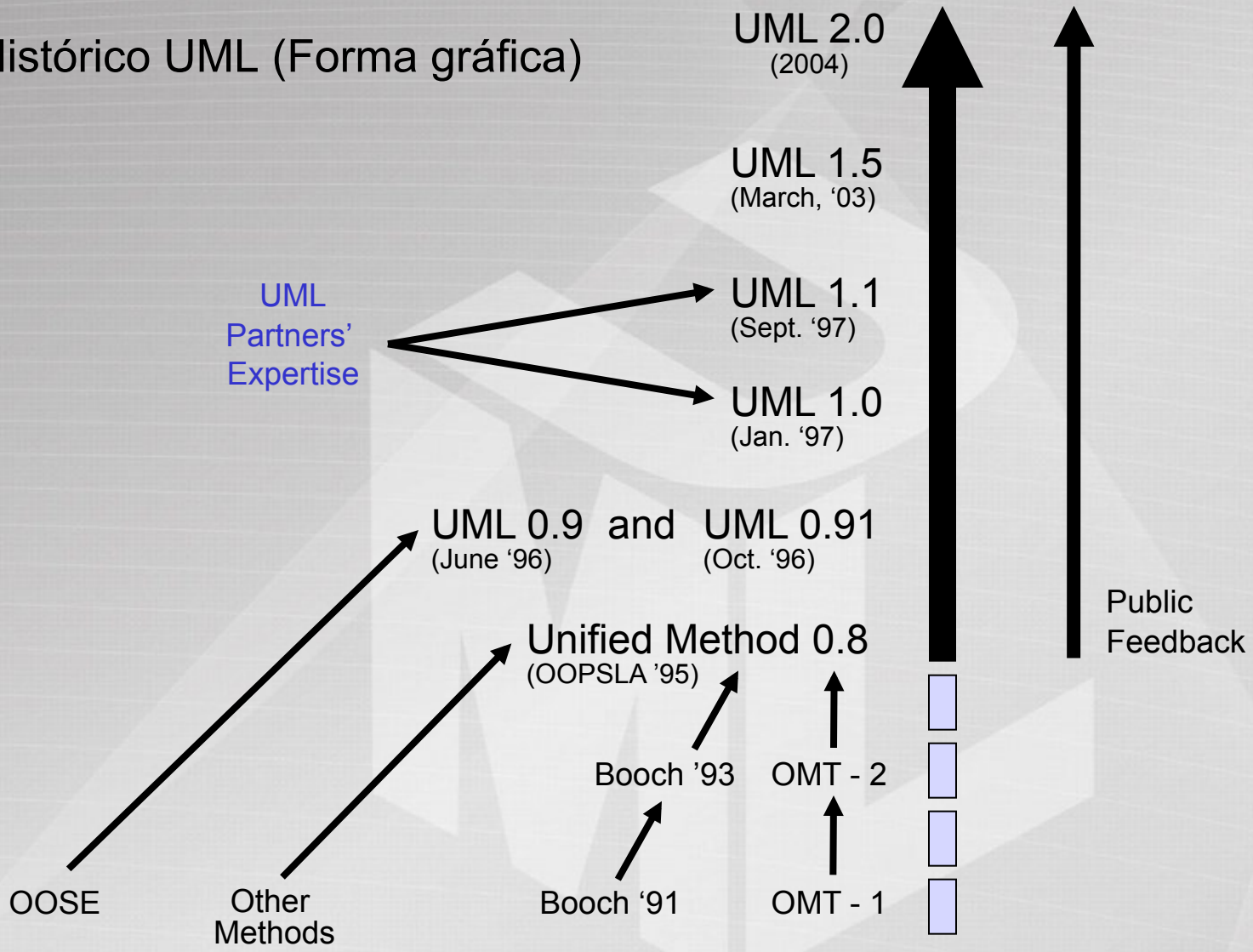
Ciclo de Vida do Desenvolvimento do Software



Inputs to the UML



Histórico UML (Forma gráfica)

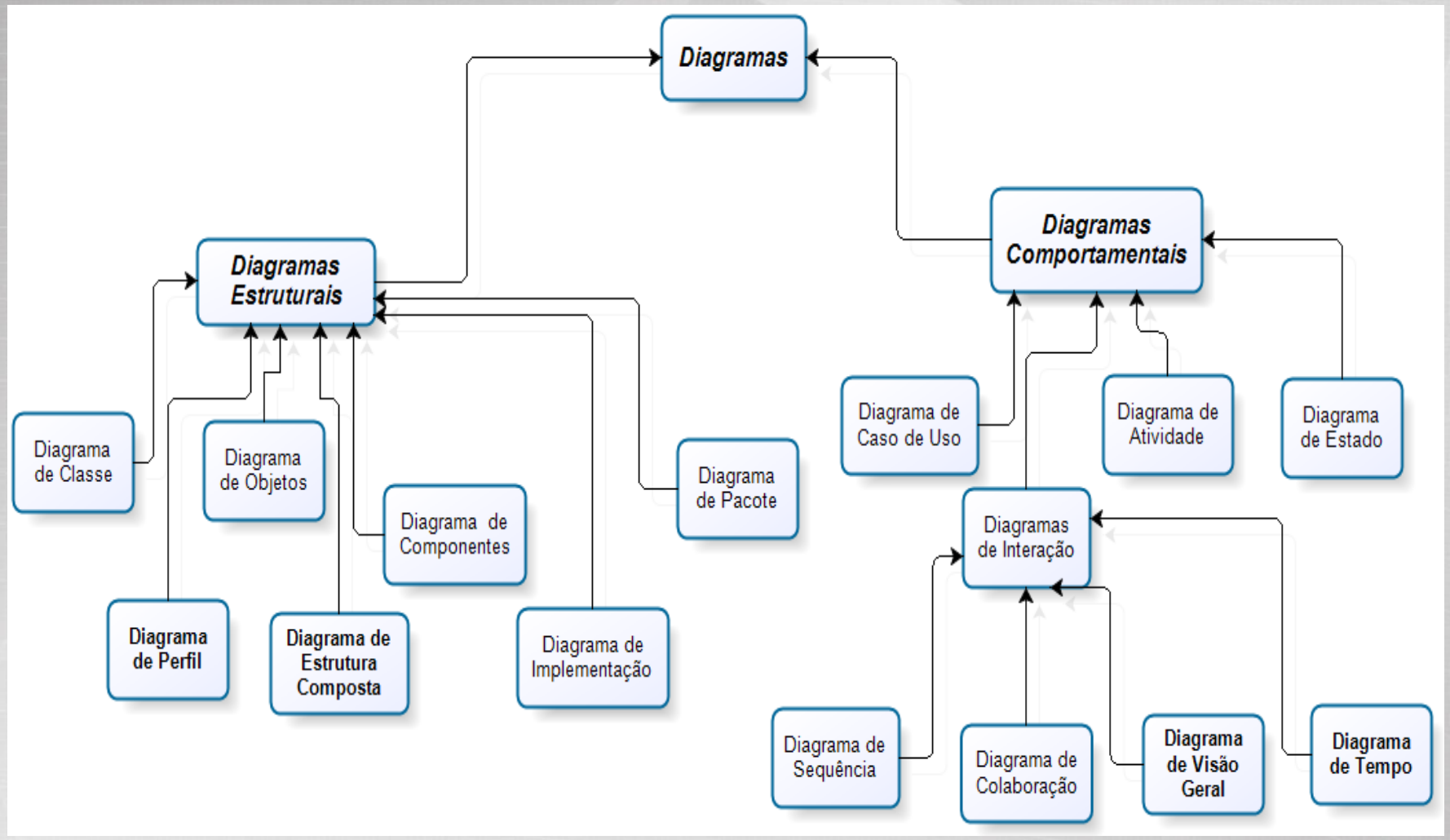


Tipos de Modelagem - V. 1.x

- Modelagem Estrutural
  - 1-Diagrama de Classes
  - *Diagramas de Pacotes;* => oficializado como diagrama na versão 2.0
  - *Diagramas de Objetos.* => oficializado como diagrama na versão 2.0
- Modelagem Comportamental
  - 2-Diagramas de Casos de Uso;
  - 3-Diagramas de Seqüência;
  - 4-Diagramas de Colaboração; => diagrama de comunicação na versão 2.0
  - 5-Diagramas de Atividades;
  - 6-Diagramas de Estados;
  - *Processos e Threads;*
  - *Eventos e Sinais.*
- Modelagem Arquitetural
  - 7-Diagramas de Componentes;
  - 8-Diagramas de Implantação;
  - *Patterns e Frameworks.*

Tipos de Modelagem - V. 2.4

Na versão 2.4, a UML tem quatorze diagramas divididos em:



Melhores Práticas

- Aplicando as 6 melhores práticas:
  - Desenvolvimento Iterativo
  - Gerência de Requisitos
  - Uso da Arquitetura de Componentes
  - Modelo Visual (UML)
  - Verificação Contínua da Qualidade
  - Gerência de Mudanças

PROCESSO CONFIGURÁVEL

Princípios Chave para o Desenvolvimento Orientado a Negócios

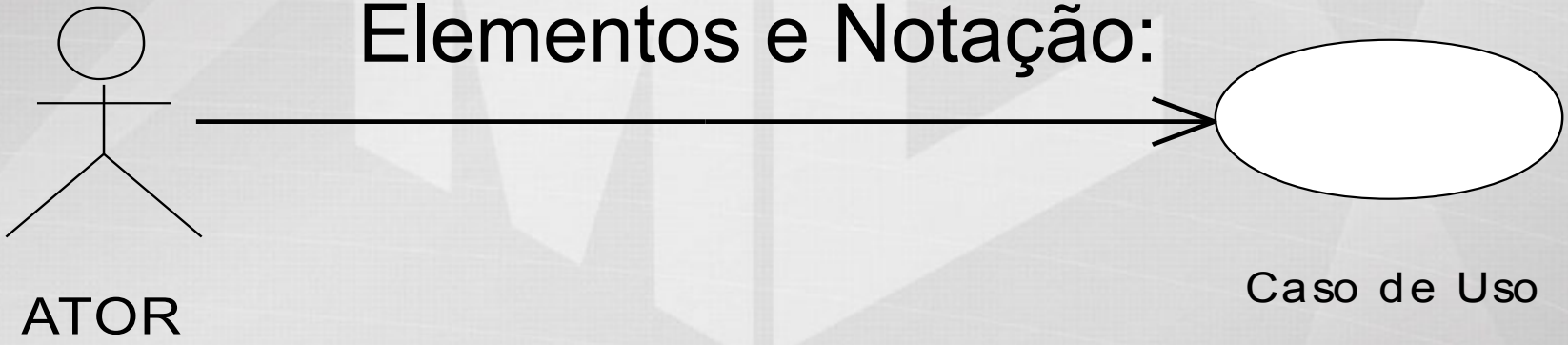
- Adaptar o Processo
- Equilíbrio que compete as Prioridades dos Stakeholders
- Colaboração Entre os Times
- Demonstrar o Valor Iterativamente
- Elevar o Nível de Abstração
- Focar Continuamente na Qualidade

REVISÃO
Modelagem Utilizando Casos de Uso de Sistemas

“Breve Revisão”- Diagrama de Casos de Uso

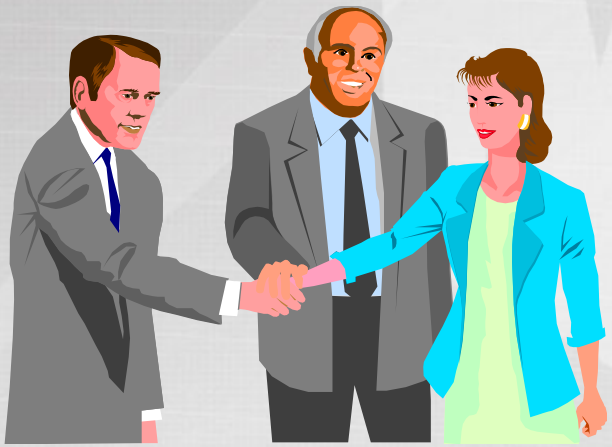
Representa o estudo e modelagem da interface externa do sistema.
Descrição dos requerimentos dos usuários.

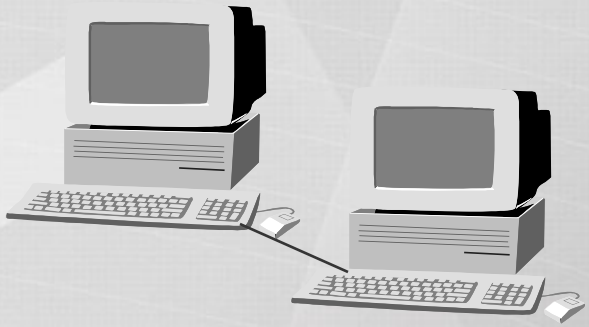
Elementos e Notação:



Ator

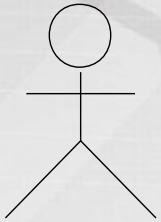
“ATOR” é qualquer pessoa, departamento, sistema computacional e dispositivos que utilizam funcionalidades do Sistema.



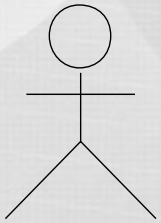


Ator

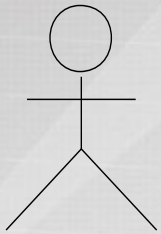
Exemplo de Atores:



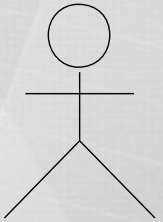
Diretoria



Departamento Pessoal



Aluno



Estoque

Caso de Uso

“Representa qualquer interação de serviços entre um ATOR e o SISTEMA. Cada serviço é representado como um Caso de Uso (*Use Case*)”.

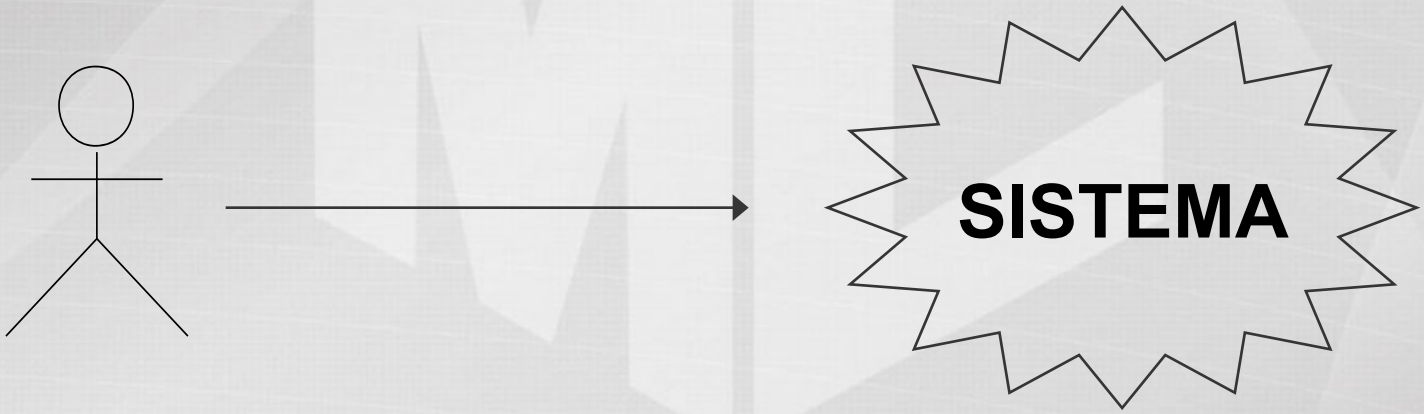
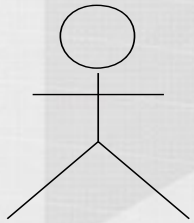


Diagrama de Casos de Uso

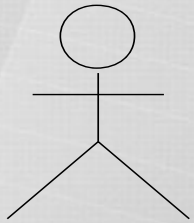
OBSERVAÇÕES:

Não representar para o mesmo ATOR mais do que uma missão.

Departamento e sistema pessoal



Sistema pessoal



Departamento pessoal

Descrição de Caso de Uso

Descrita no Artefato: Especificação de Casos de Uso

Casos de Uso – Nível de Especificação

- Definir Funcionalidades Genéricas;
- Regras de Negócios;
- Visão do Usuário;

O Escopo de um Caso de Uso

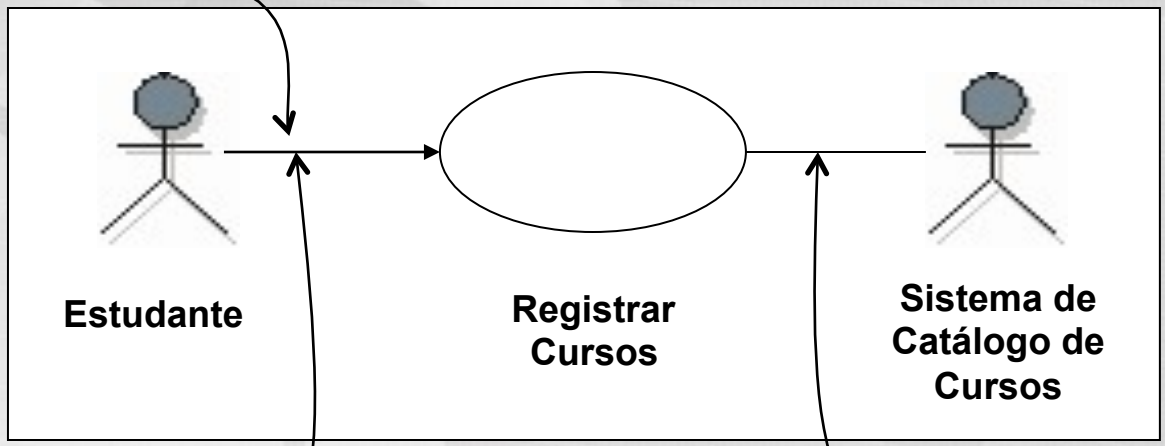
Geralmente, é difícil decidir se um conjunto de interações do usuário do sistema (ATOR COM O SISTEMA), ou uma caixa de diálogo, é um ou são vários casos de uso. Considere o uso de uma máquina de reciclagem. O cliente insere itens de depósito como, por exemplo, latas, garrafas e caixotes na máquina de reciclagem. Ao inserir todos os itens de depósito, basta pressionar um botão e um recibo é impresso. Esse recibo pode ser trocado por dinheiro.

Cada Comunicação-Associação é um conjunto de diálogo

Estudante faz o login no sistema →

Sistema aprova o login ←

Estudante solicitação informação do Curso →



Sistema mostra a lista de cursos ←

Estudante seleciona cursos →

Sistema confirma curso disponível ←

Sistema mostra aprovação da programação ←

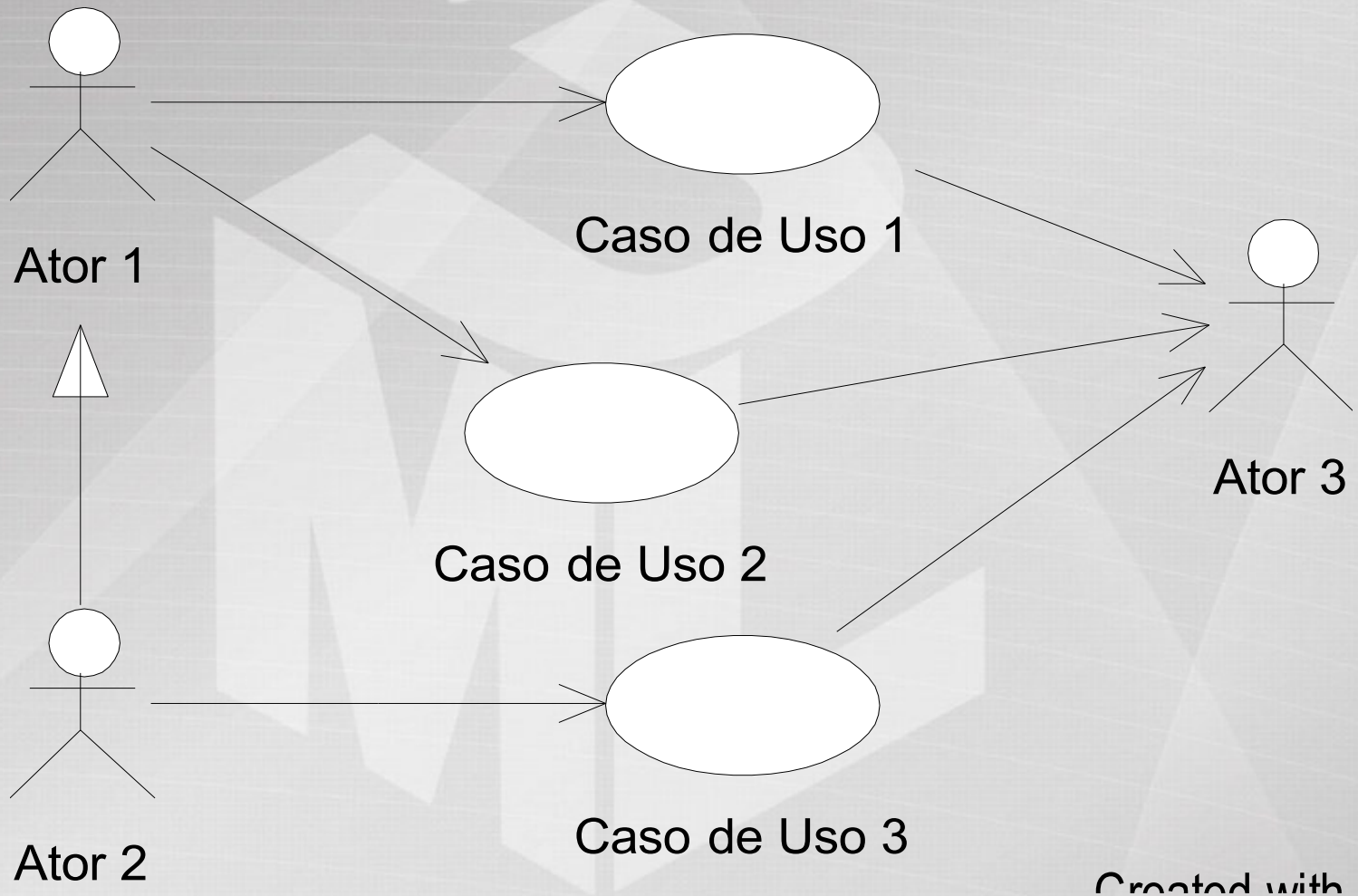
Sistema transmite solicitação →

Catálogo de Cursos retorna informação do curso ←

Herança entre atores

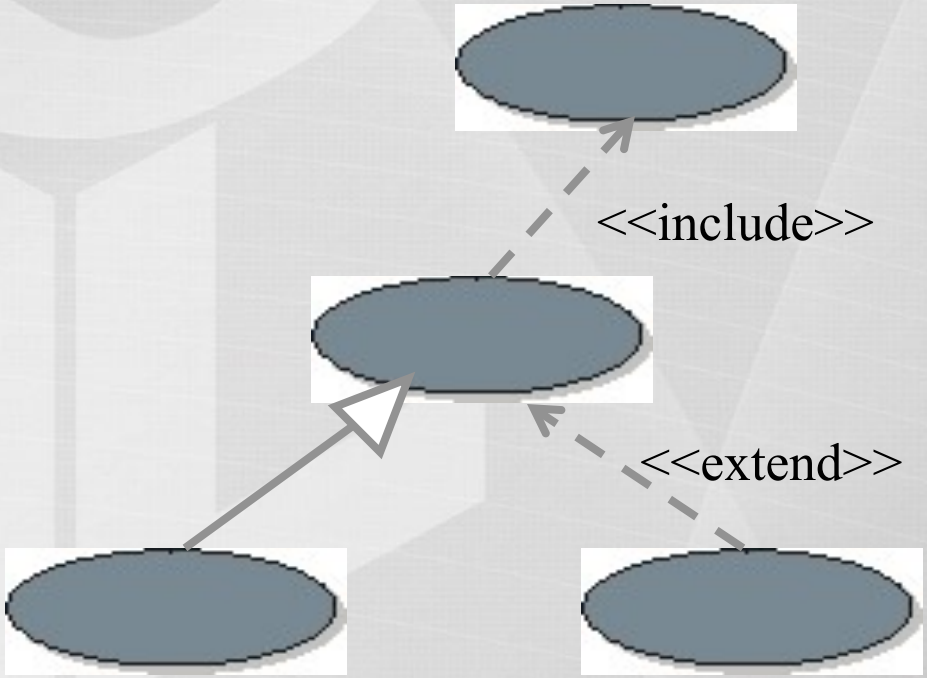
- Pode-se utilizar herança entre atores;
- Herança entre atores significa que um ator preenche os mesmos papéis de outro ator, podendo também preencher papéis adicionais;
- Na UML herança é indicado pelo relacionamento de generalização.

Herança entre atores



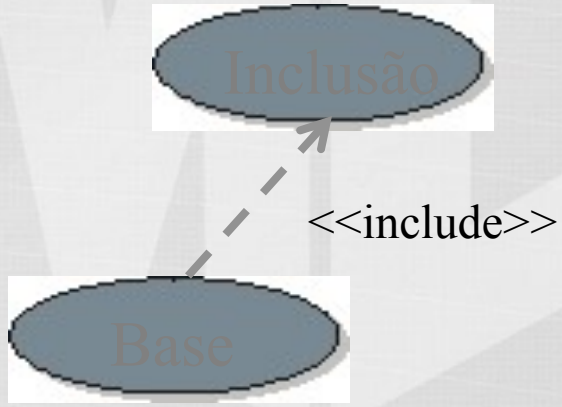
Relacionamentos Entre Casos de Uso

- *Include*
- *Extend*
- Generalização

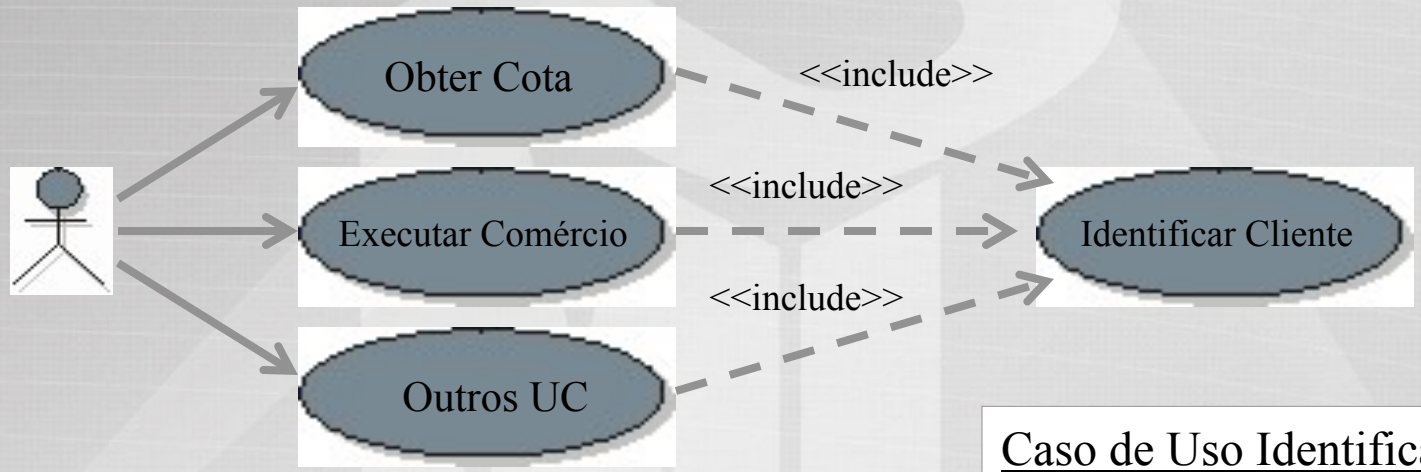


O que é um Relacionamento *Include*?

- Um relacionamento de um caso de uso base para um caso de uso de inclusão
- O comportamento definido no caso de uso de inclusão é **inserida explicitamente** para o caso de uso base



Relacionamento *Include*: Exemplo RU e-st



Caso de Uso Obter Cota

1. Incluir "Identificar Cliente" para verificar identidade do cliente
2. Mostrar opções. Cliente seleciona "Obter Cota"
3. ...

Caso de Uso Identificar Cliente

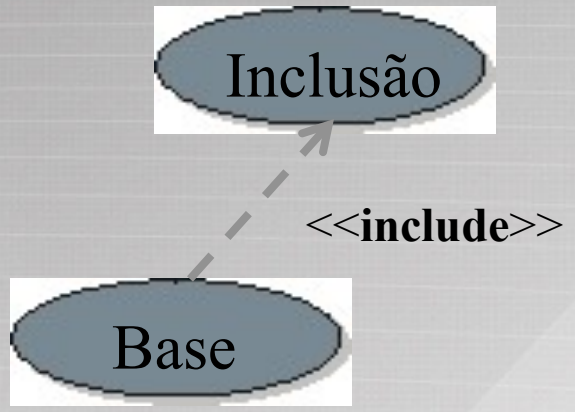
1. Logon
2. Validar logon
3. Entrar com a senha
4. Verificar a senha

A1: Logon inválido

A2: Senha errada

A3: .....

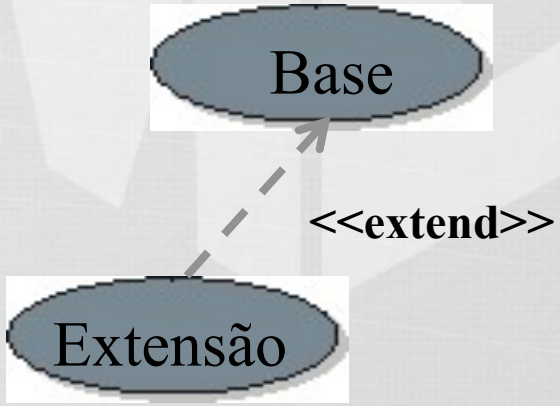
Por Que Usar um Relacionamento de *Include*?



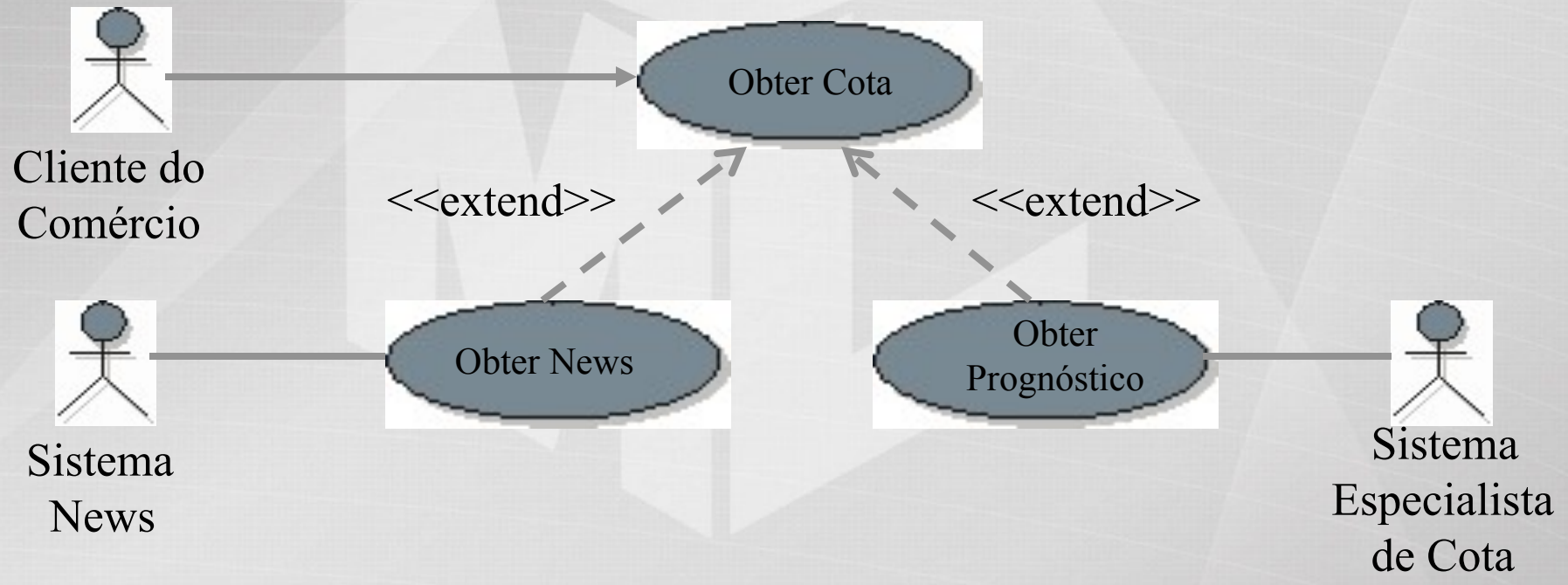
- Fatorar comportamento comum para **dois ou mais** casos de uso
  - Evitar descrever qualquer comportamento múltiplas vezes (reutilização).
  - Assegurar comportamento comum consistente com o resto.
- Fatorar e encapsular comportamento de um caso de uso base
  - Simplificar fluxo de eventos complexos (ver exemplo RU-st)
  - Fatorar comportamento que não faz parte de um propósito primário.

O que é um Relacionamento *Extend*?

- Conexão de um caso de uso estendido para um caso de uso base
  - Inserir comportamento estendido de um caso de uso num caso de uso base
  - Inserir somente se a condição de extensão for verdadeira
  - Inserir num caso de uso base ao ponto de extensão nomeado



Relacionamento *Extend*: Exemplo RU e-st



Relacionamento *Extend*: Exemplo RU e-st (cont.)

Caso de Uso Obter Cota

Fluxo Básico

1. Include "Identificar Cliente" para verificar identidade do cliente.
2. Mostrar opções
3. Cliente seleciona "Obter Cota."
4. Cliente obtém cotas
5. Cliente obtém outras cotas
6. Cliente faz logoff

A1. Sistema Cota indisponível

...

Pontos de Extensão:

O ponto de extensão "Serviço Opcional" ocorre no Passo 3 do Fluxo Básico.

Caso de Uso Obter News

Este caso de uso extend o Caso de Uso Obter Cota, ao ponto de extensão "Serviços Opcionais".

Fluxo Básico:

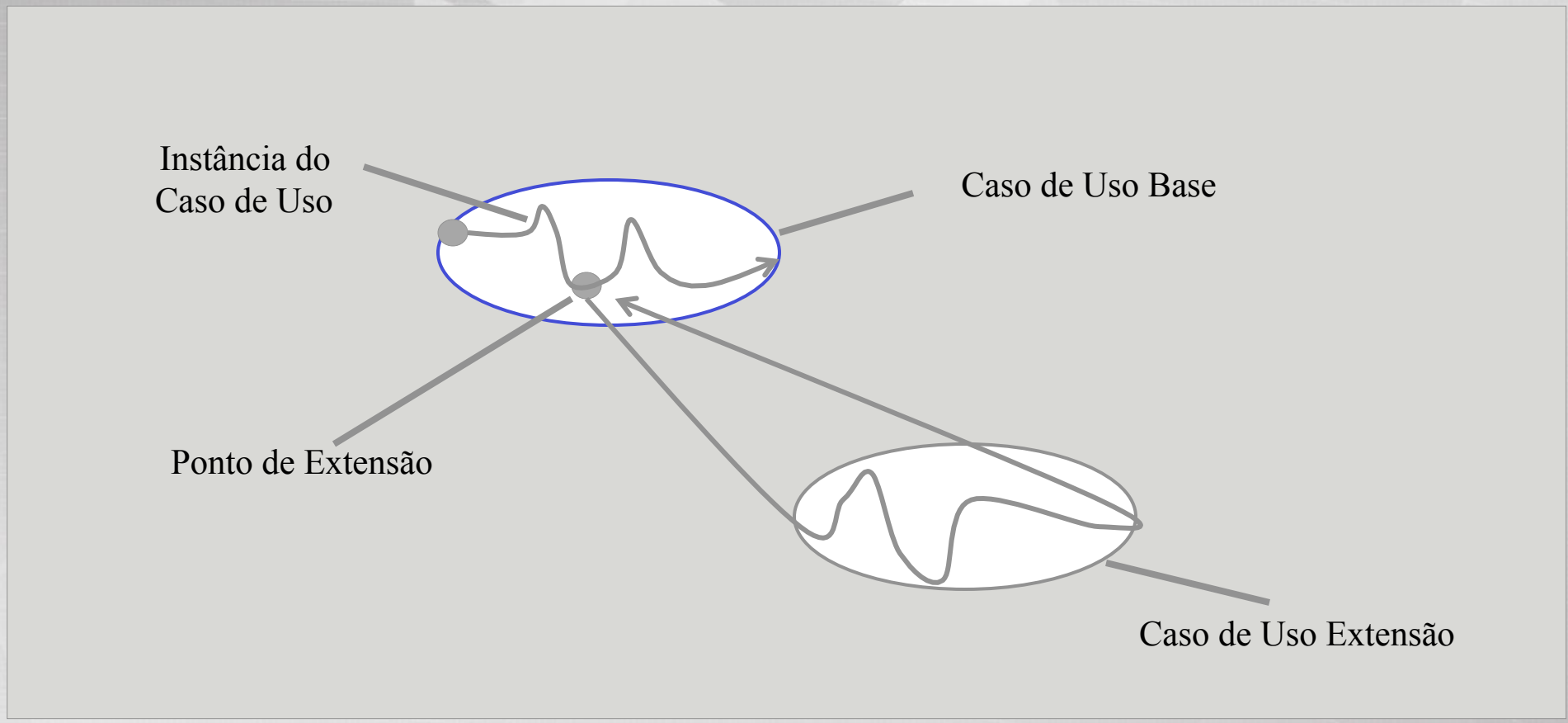
1. Se o Cliente selecionar "Obter News", o sistema pergunta ao cliente o período de tempo e o número de itens news.
2. Cliente entra com o período de tempo e o número de itens. O sistema envia o símbolo da ação de comércio para o Sistema News, recebe resposta, e mostra os news para o cliente.
3. O Caso de Uso Obter Cota continua.

A1: Sistema News Indisponível

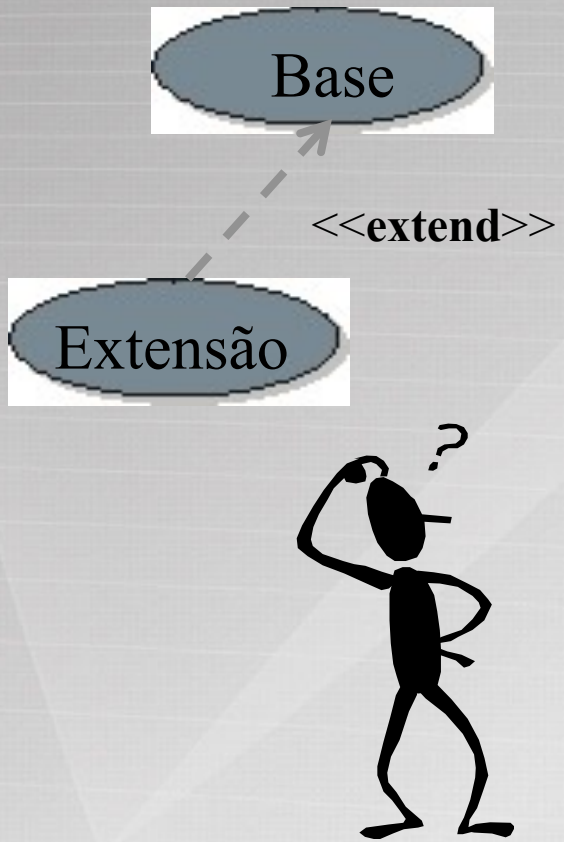
A2: Não Existe News Sobre Esta Ação

Execução de um *Extend*

- Executado quando o ponto de extensão é alcançado e a condição de extensão (decisão) for verdadeira



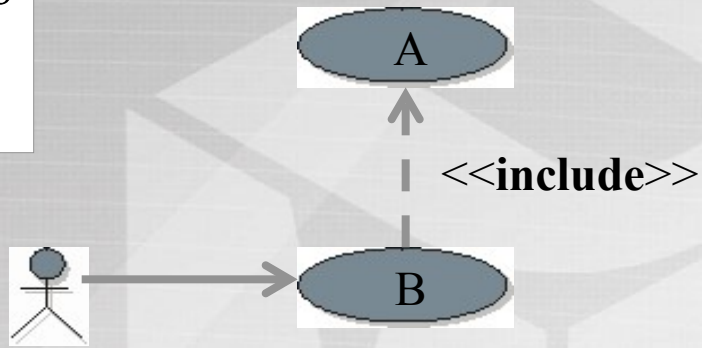
Por Que Usar um Relacionamento *Extend*?



- Fatorar um comportamento opcional ou excepcional
  - Executado somente numa certa condição
  - Fatorando a simplificação do fluxo de eventos do caso de uso base
  - Exemplo: ligando um alarme
- Adicionar comportamento estendido
  - Desenvolver comportamento separadamente, possivelmente numa versão posterior
  - Exemplo: Caso de Uso Obter News

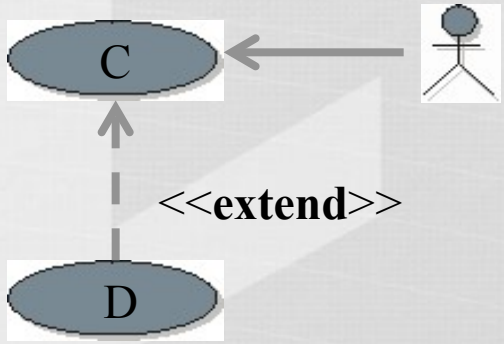
Casos deUso (Concreto versus Abstrato)

Um caso de uso é abstrato ou concreto



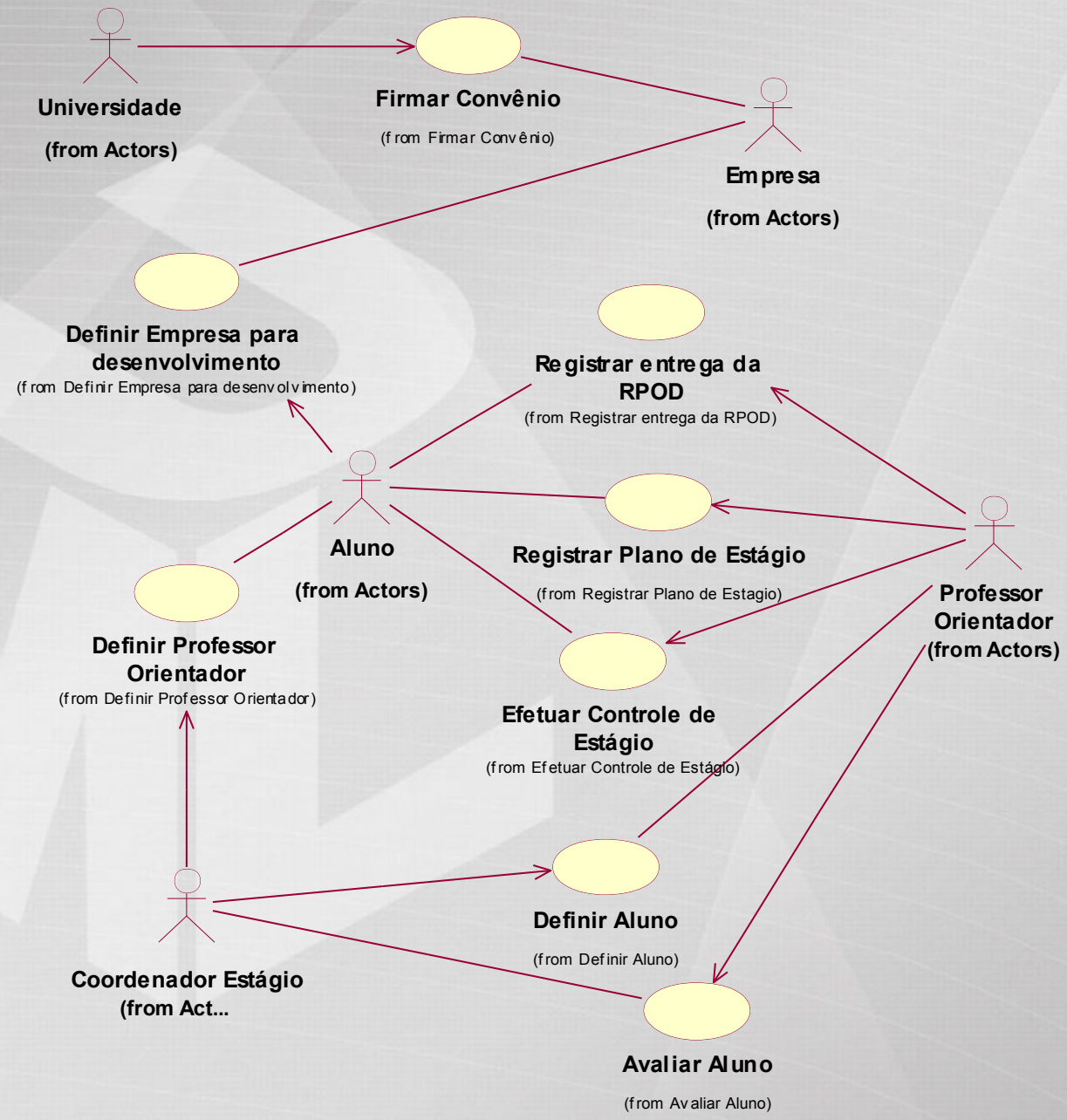
Casos de uso abstrato (A & D):
- Não tem que ser completo.
- Existe somente por outros casos de uso.
- Nunca são instanciados para si mesmo (não é inicializado por um ator).

Casos de uso concreto (B & C):
- Deve ser completo e significativo
- Pode ser instanciado para si mesmo

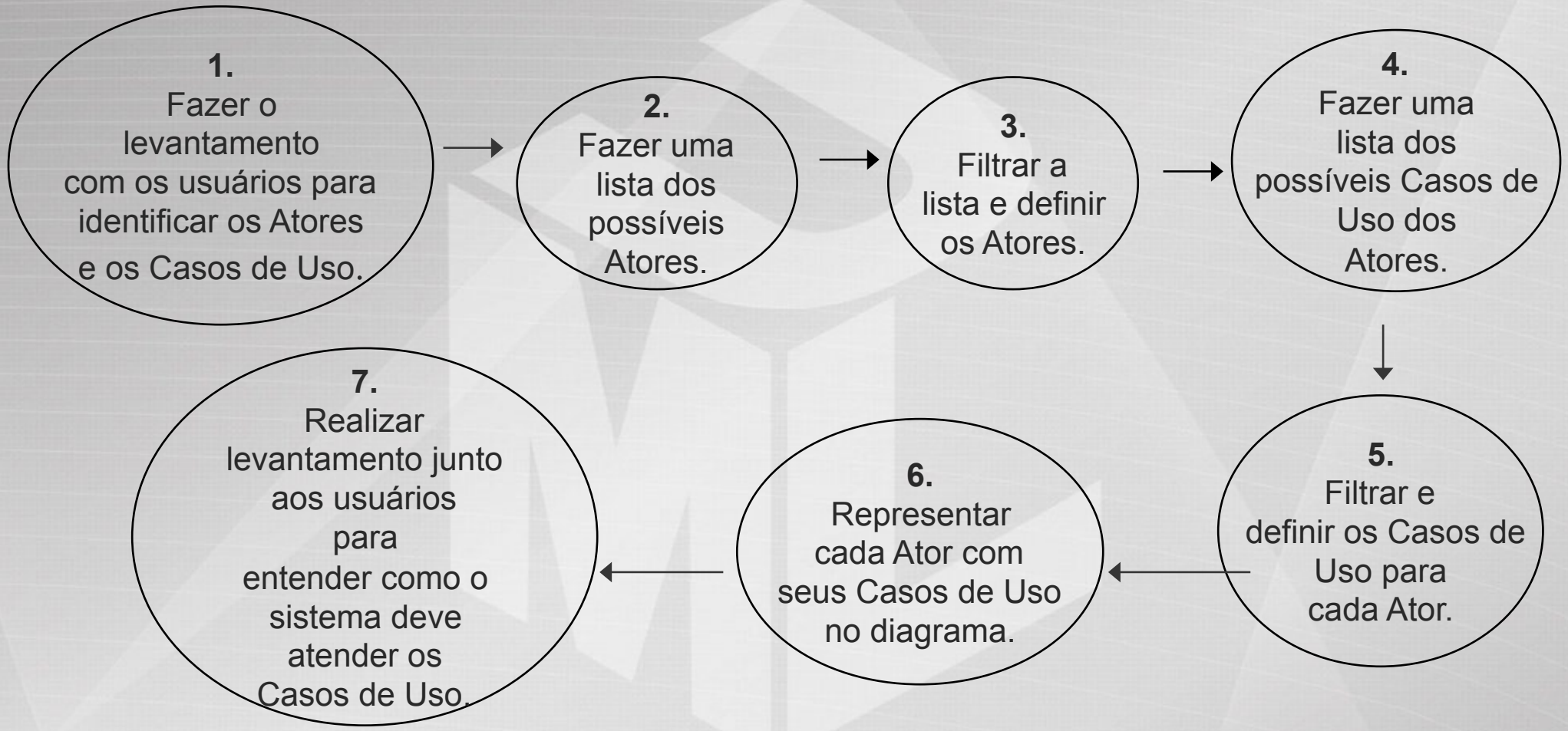


É comum o caso de uso de extensão ser abstrato, mas não é necessário que seja.

Exemplo de um diagrama de Caso de Uso para um Sistema de Controle de Estágio



Diagramas de Casos de Uso



Representação da Arquitetura de Software através de Pacotes

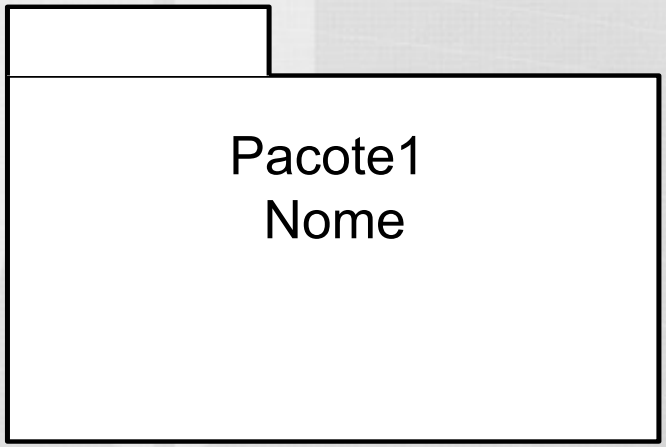
Pacote

- Um pacote é um mecanismo de propósito geral para a organização de elementos em grupos. Um pacote é representado graficamente como uma pasta com uma guia.

Pacote

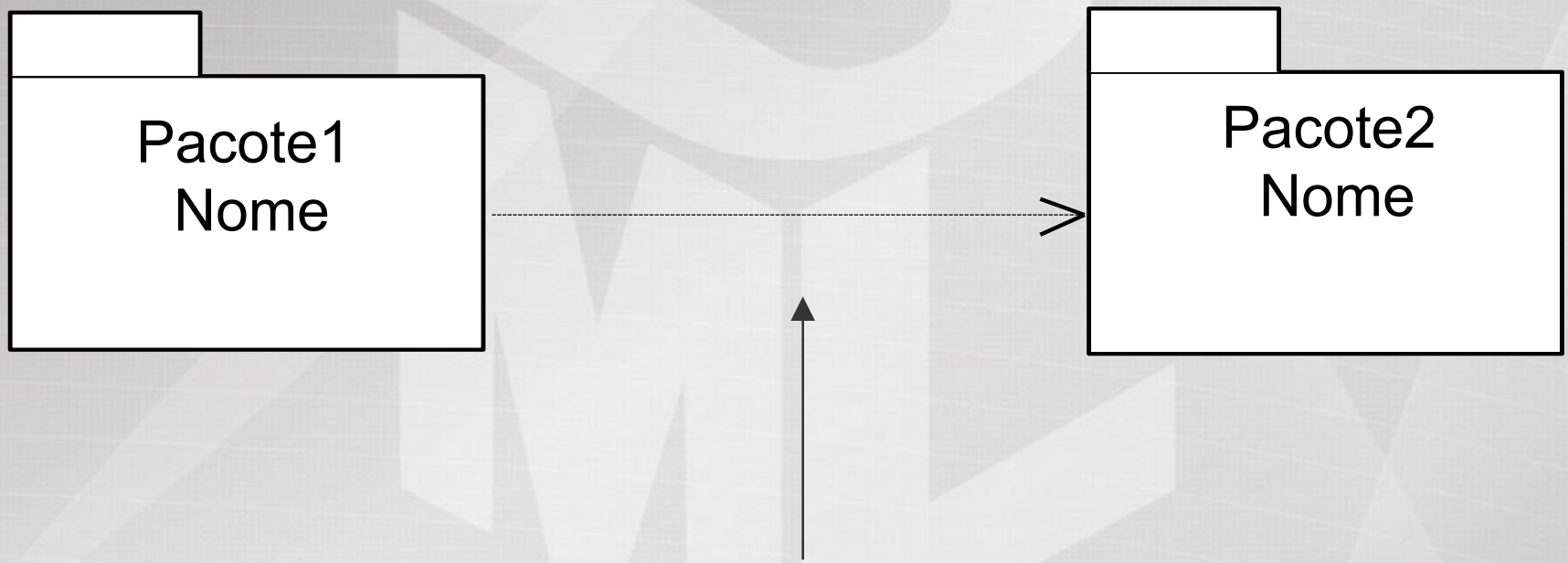
- Cada pacote deve ser nomeado;
- O nome dos pacotes são escritos sobre os pacotes;
- Ligações entre os pacotes são representadas para indicar a dependência entre os pacotes;

- **Notação:**



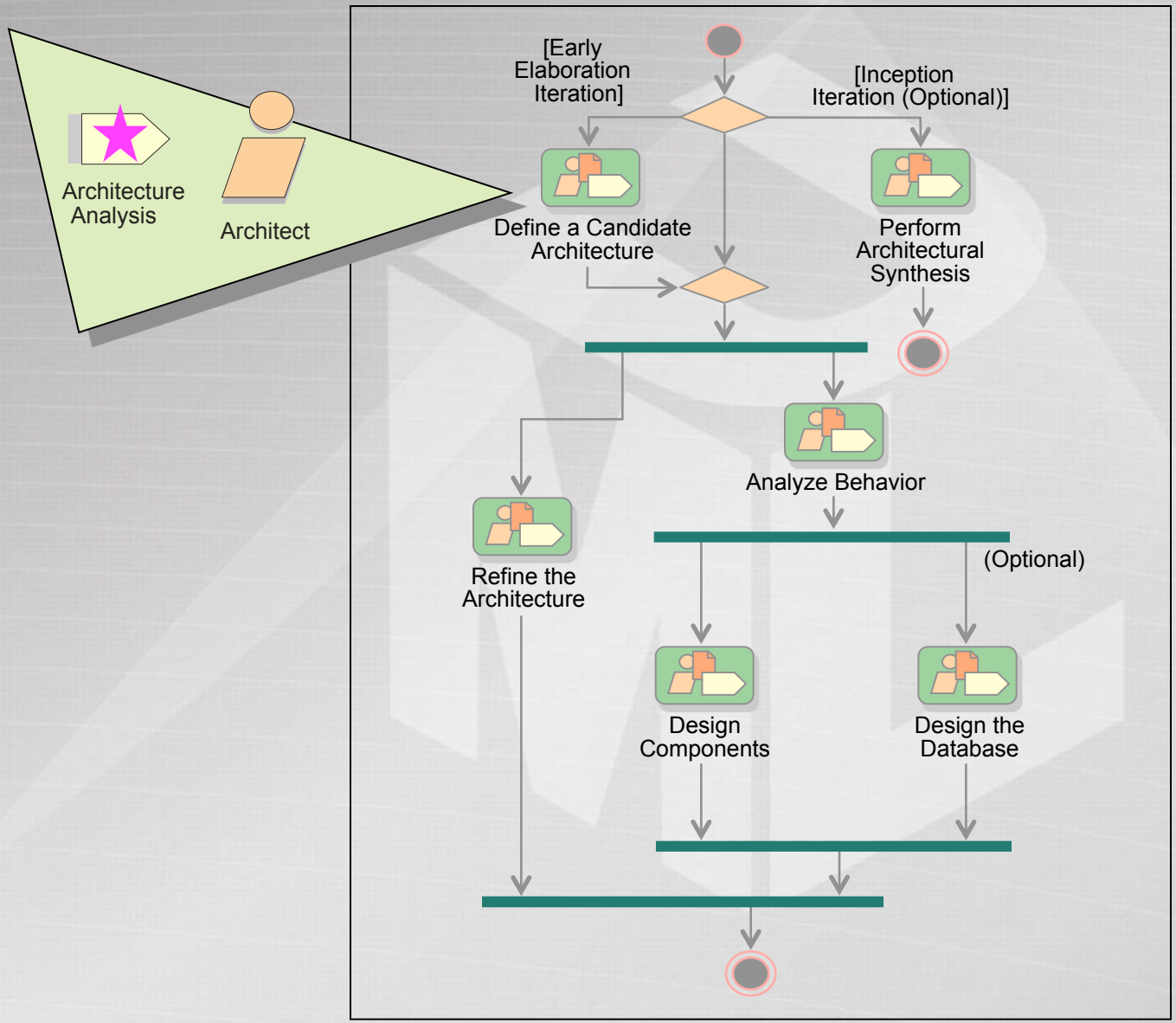
Dependência de Pacotes

- **Notação de Dependência:**

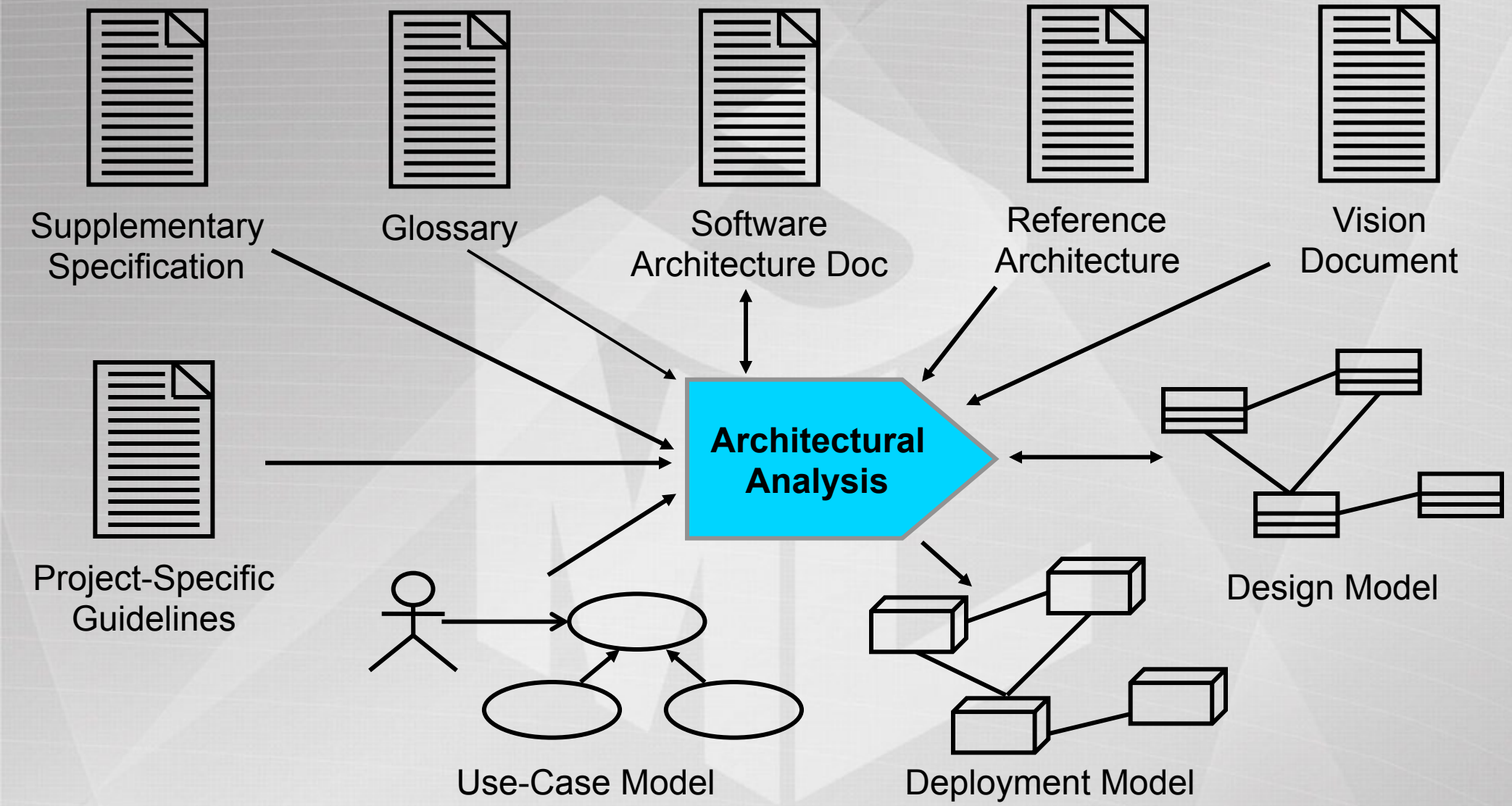


Relacionamento de dependência

Contexto em Análise Arquitetural

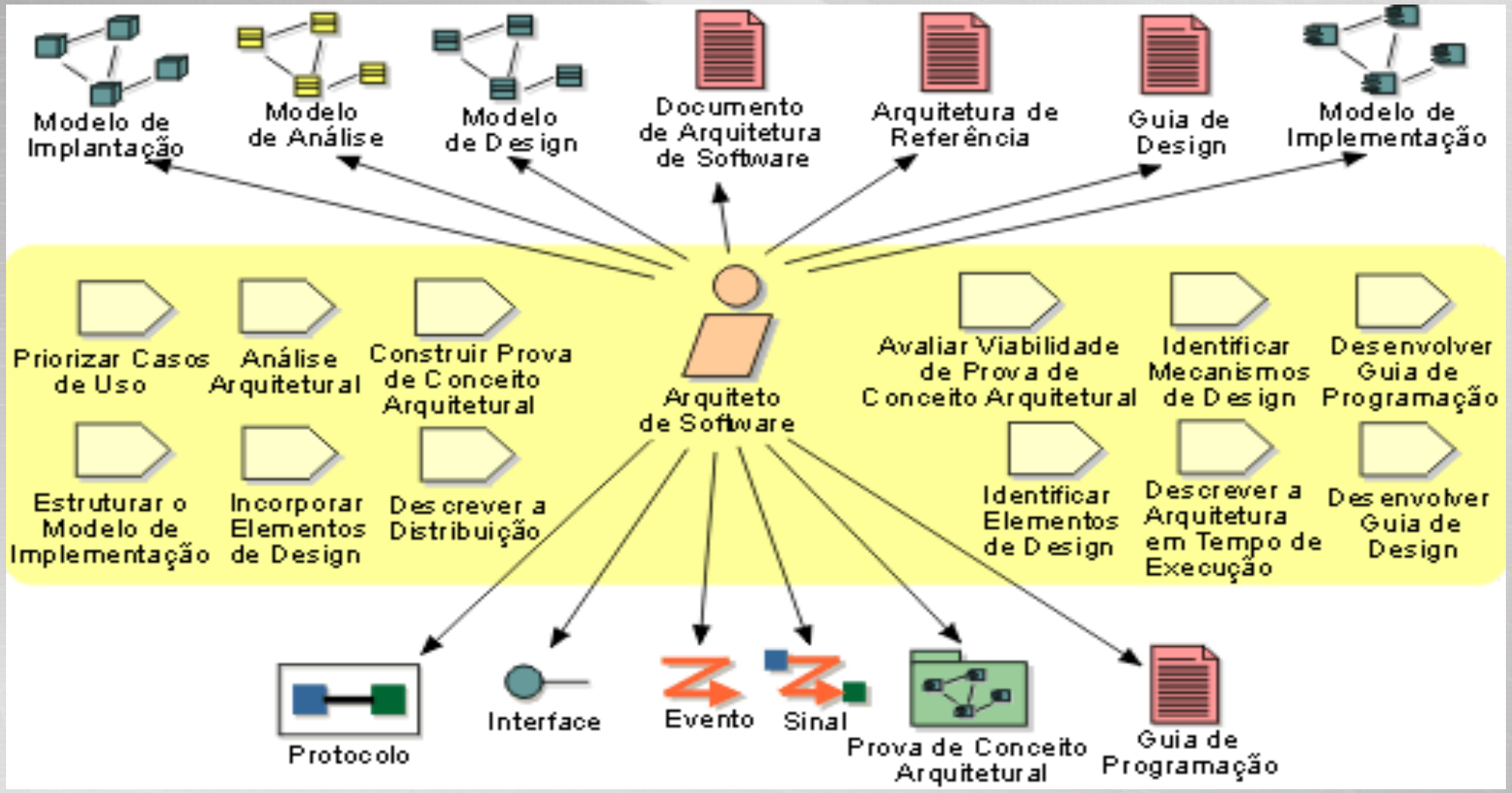


Visão Geral da Análise Arquitetural



Papel: Arquiteto de Software

- O papel arquiteto de software lidera e coordena as atividades e os artefatos técnicos no decorrer do projeto. O arquiteto de software estabelece a estrutura geral de cada visão de arquitetura: a decomposição da visão, o agrupamento dos elementos e as interfaces entre esses principais agrupamentos. Portanto, comparado aos outros papéis, a visão do arquiteto de software é ampla, e não detalhada.


Modelo de Implantação
Modelo de Análise
Modelo de Design
Documento de Arquitetura de Software
Arquitetura de Referência
Guia de Design
Modelo de Implementação
Priorizar Casos de Uso
Análise Arquitetural
Construir Prova de Conceito Arquitetural
Estruturar o Modelo de Implementação
Incorporar Elementos de Design
Descrever a Distribuição
Arquiteto de Software
Avaliar Viabilidade de Prova de Conceito Arquitetural
Identificar Mecanismos de Design
Desenvolver Guia de Programação
Identificar Elementos de Design
Descrever a Arquitetura em Tempo de Execução
Desenvolver Guia de Design
Protocolo
Interface
Evento
Sinal
Prova de Conceito Arquitetural
Guia de Programação

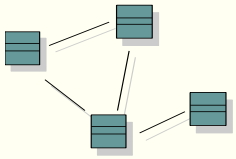
Qual o conhecimento de um Arquiteto de Software?

- "O arquiteto ideal deve ser uma pessoa erudita, um matemático, familiarizado com estudos históricos, um estudioso aplicado de filosofia, conhecedor de música, que não desconheça medicina, detentor de saber jurídico e familiarizado com astronomia e cálculos astronômicos." - Vitruvius, há aproximadamente 25 anos a.C.

Arquitetura de Software

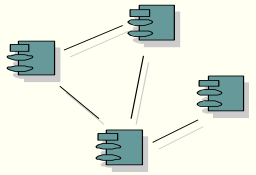
Segundo Buschmann 1996, a arquitetura de software consiste da descrição dos subsistemas, e dos componentes de um sistema de software e dos relacionamentos entre eles.

Arquitetura 4 + 1



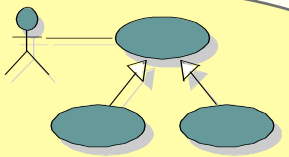
Logical View

Analista/Projetista
Estrutura



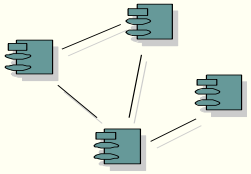
Implementation View

Programador
Gerência de Software



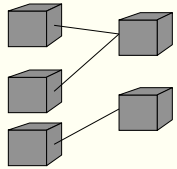
Usuário-final
Funcionalidade

Use-Case View



Process View

System integrators
Performance
Escalabilidade
Rendimento



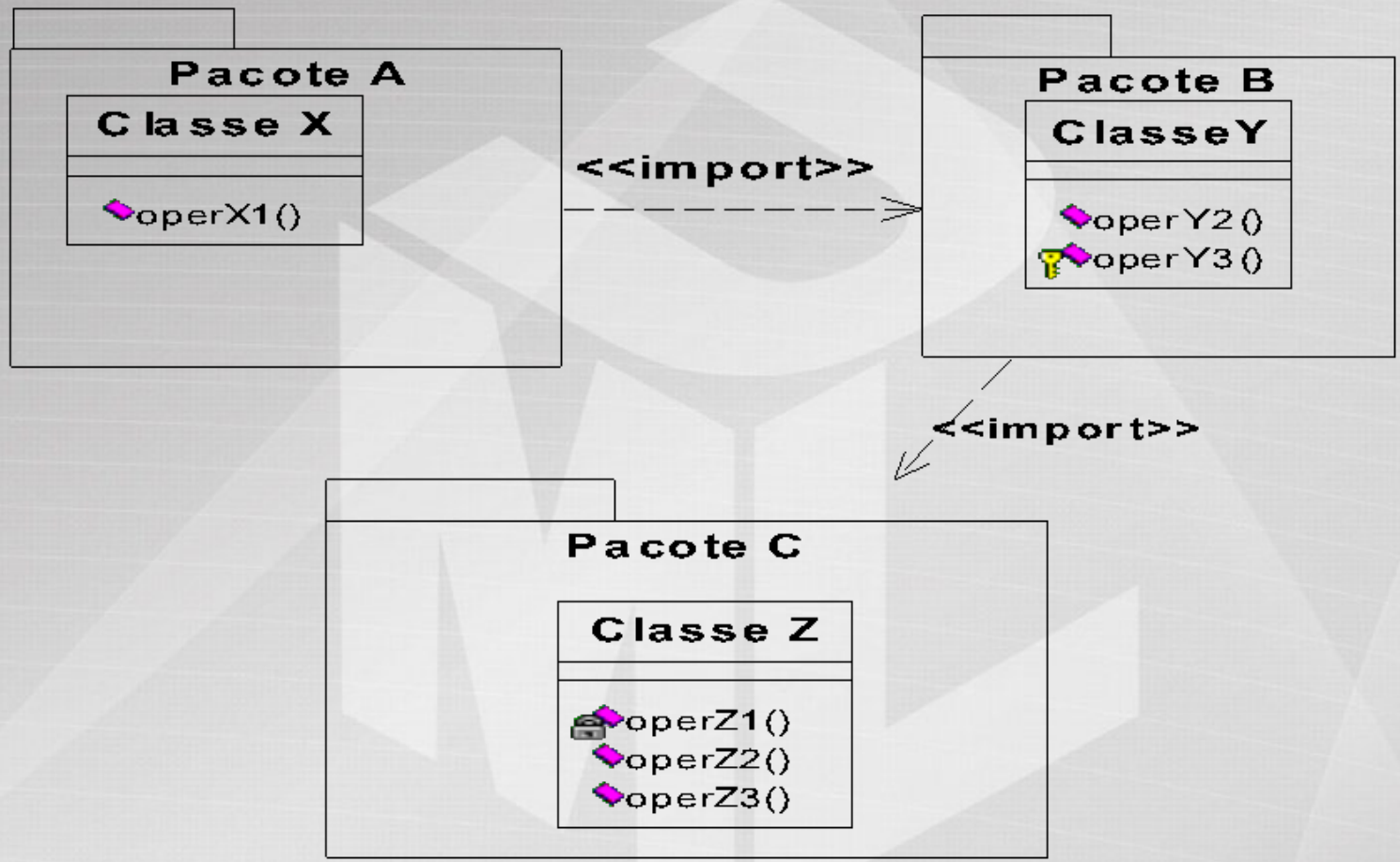
Deployment View

System engineering
Topologia do Sistema
Entrega, instalação
comunicação

Pacotes (Importação e Exportação)

- Na UML, a modelagem de um relacionamento de importação é feita como uma dependência assinalada pelo estereótipo <<*import*>> como adorno;
- As partes públicas de um pacote são chamadas suas exportações;
- As partes exportadas por um pacote são visíveis somente para o conteúdo dos pacotes que importam aquele pacote explicitamente.
- As dependências de importação e de acesso não são transitivas.

Pacotes (Importação e Exportação)



Pacotes

- Dois ou três níveis de aninhamento constituem o limite do que será conveniente utilizar. Mais do que o aninhamento, utilize a importação para a organização de seus pacotes.

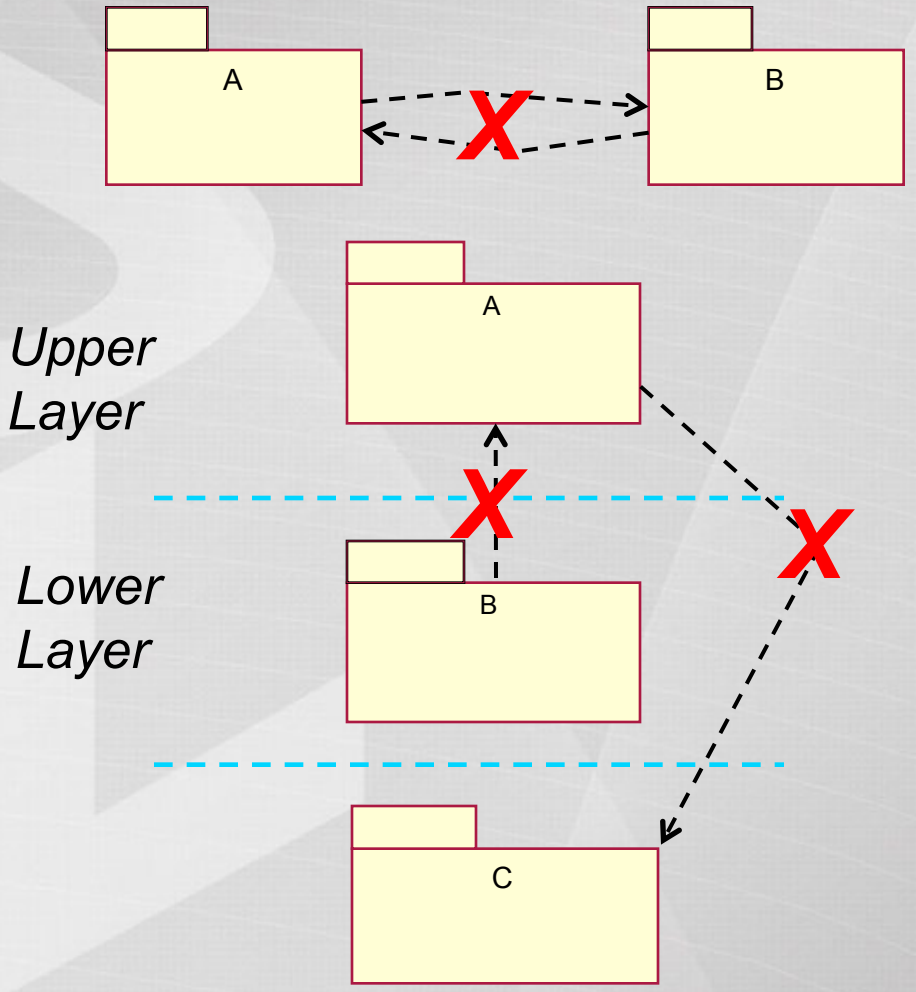
Pacotes

A UML define 5 estereótipos-padrão que se aplicam aos pacotes:

- **<<facade>>:** especifica um pacote que é apenas uma visualização de algum outro pacote;
- **<<framework>>:** especifica um pacote que é formado por padrões;
- **<<stub>>:** serve como proxy para o conteúdo público de outro pacote; ou seja, são utilizados quando possuimos ferramentas capazes de dividir um sistema em pacotes que manipulamos em diferentes momentos e potencialmente em diferentes locais.
- **<<subsystem>>:** representa uma parte independente de todo o sistema cuja modelagem está sendo feita.
- **<<system>>:** representa todo o sistema cuja modelagem está sendo feita.

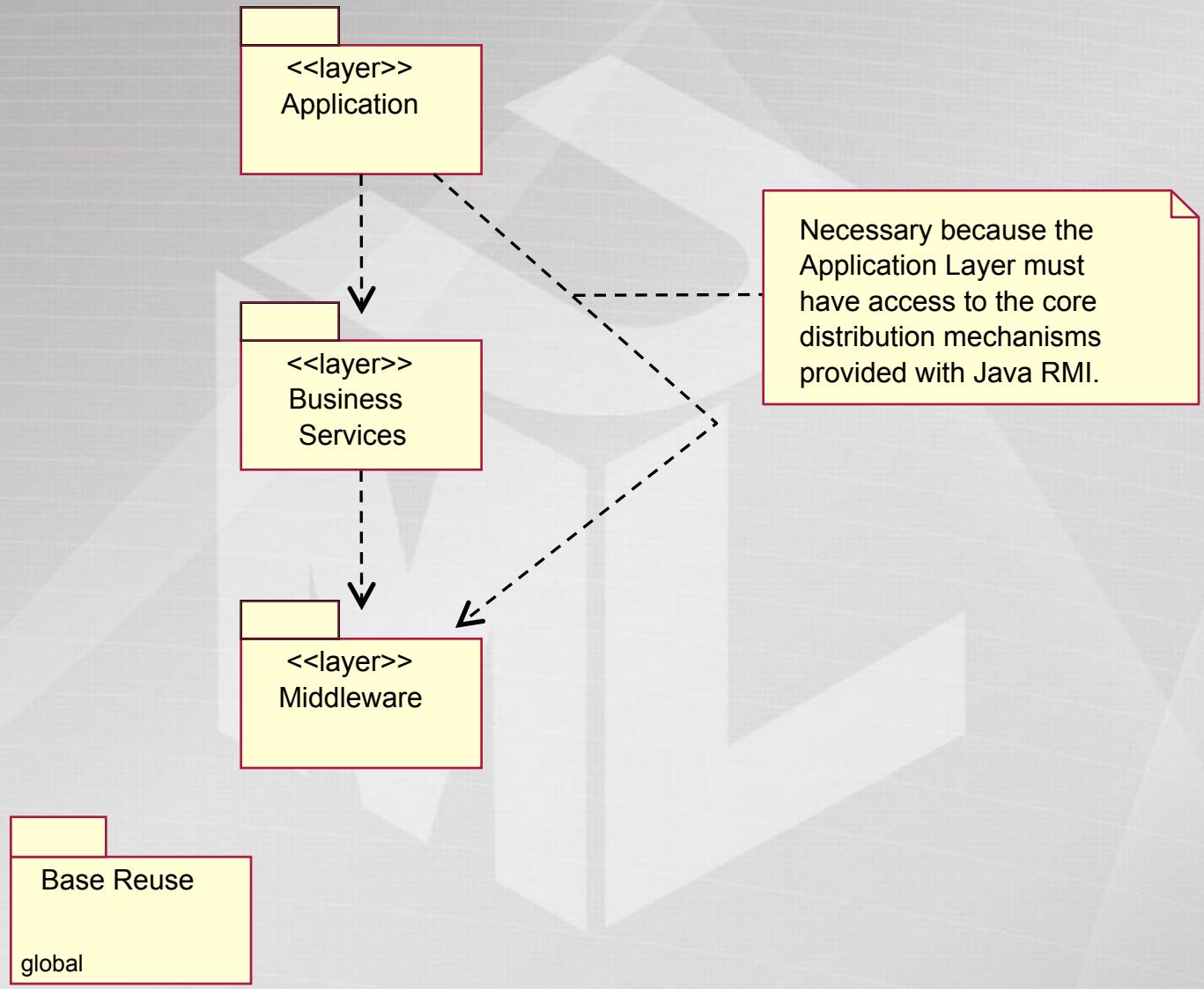
Acoplamento de Pacotes

- Acoplamento: quanto maior a dependência entre os pacotes maior o acoplamento
- Pacotes não deveriam ter co-dependências (circular)
- Pacotes da camada inferior não deveria ser dependente da camada superior. Isto é uma regra para o estilo arquitetural em camadas
- Em geral, as dependências não deveriam pular camadas

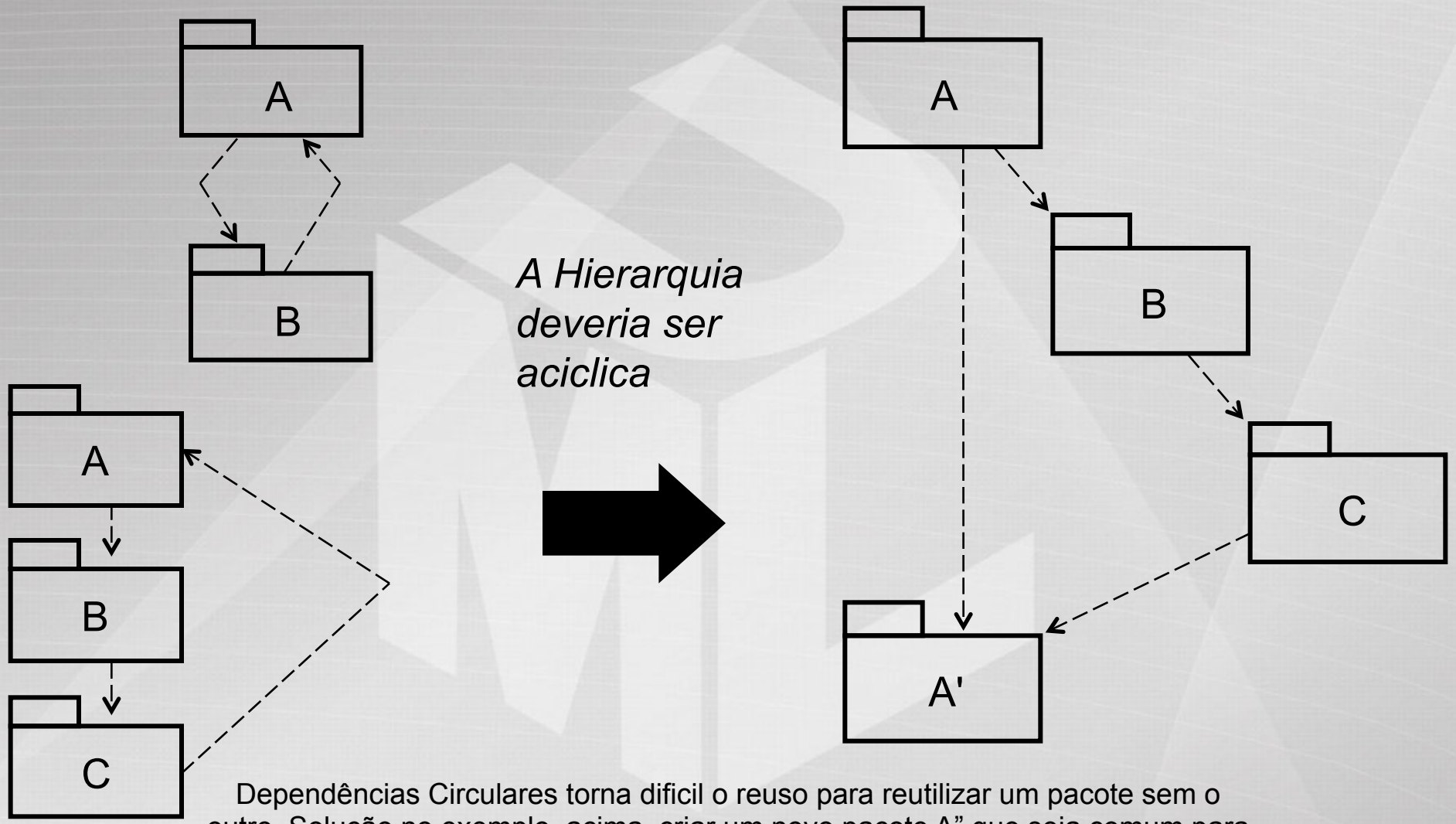


X = Violação de acoplamento

Exemplo: Arquitetura em Camadas



Evitando Dependências Circulares



Dependências Circulares torna dificil o reuso para reutilizar um pacote sem o outro. Solução no exemplo, acima, criar um novo pacote A” que seja comum para A e C.

Estilo Arquitetural

Um estilo de arquitetura pode ser definido como uma família de sistemas em termos de um modelo de estrutura de organização. Um estilo de arquitetura expressa componentes e o relacionamento entre eles, com restrições de suas aplicações, composições e regras de projeto para suas construções [BUS 96]. Mais especificamente, pode-se dizer que um estilo de arquitetura define um vocabulário de componentes e tipos de conectores, mais um conjunto de restrições sobre a combinação desses (GARLAN 1995).

Arquitetura de Software

Tipos de estilos arquiteturais:

- ✓ Pipes e Filtros
- ✓ **Camadas**
- ✓ Objetos
- ✓ Invocação Implícita
- ✓ Quadro-negro
- ✓ Repositório (Passivo)
- ✓ Arquiteturas de Aplicações Distribuídas
  - ✓ Processos Comunicantes
  - ✓ Cliente-Servidor
- ✓ Chamada de Procedimento

Exercício 1:

- Configurar na Ferramenta CASE IBM Rational Rose para a visualização da Arquitetura Multicamadas
- Elaborar o Estilo Arquitetural proposto para utilização nos estudos de casos utilizando pacotes

Arquitetura de Software

Proposta do Estilo Arquitetural em Camadas para o desenvolvimento dos estudos de casos