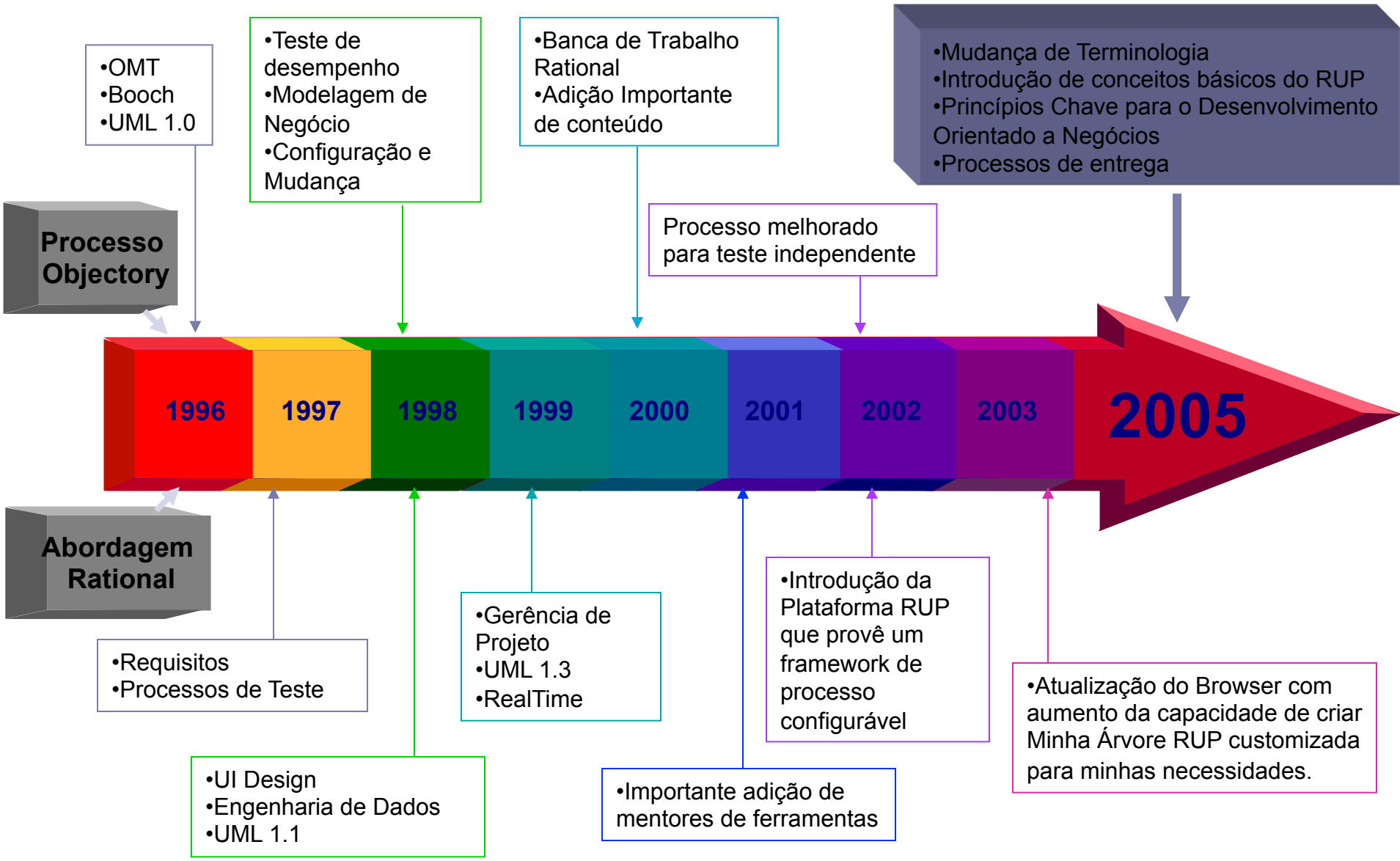
Engenharia de Software I

RUP 7 e Gerência de Requisitos

- **Prof.:** ***Sergio Akio Tanaka***
- **E-mail: sergio.tanaka@unifil.br**

História do RUP



Módulo 1 – Melhores práticas de Engenharia de Software

Módulo 1 – Melhores Práticas de Engenharia de Software

"Problemas do desenvolvimento de software"

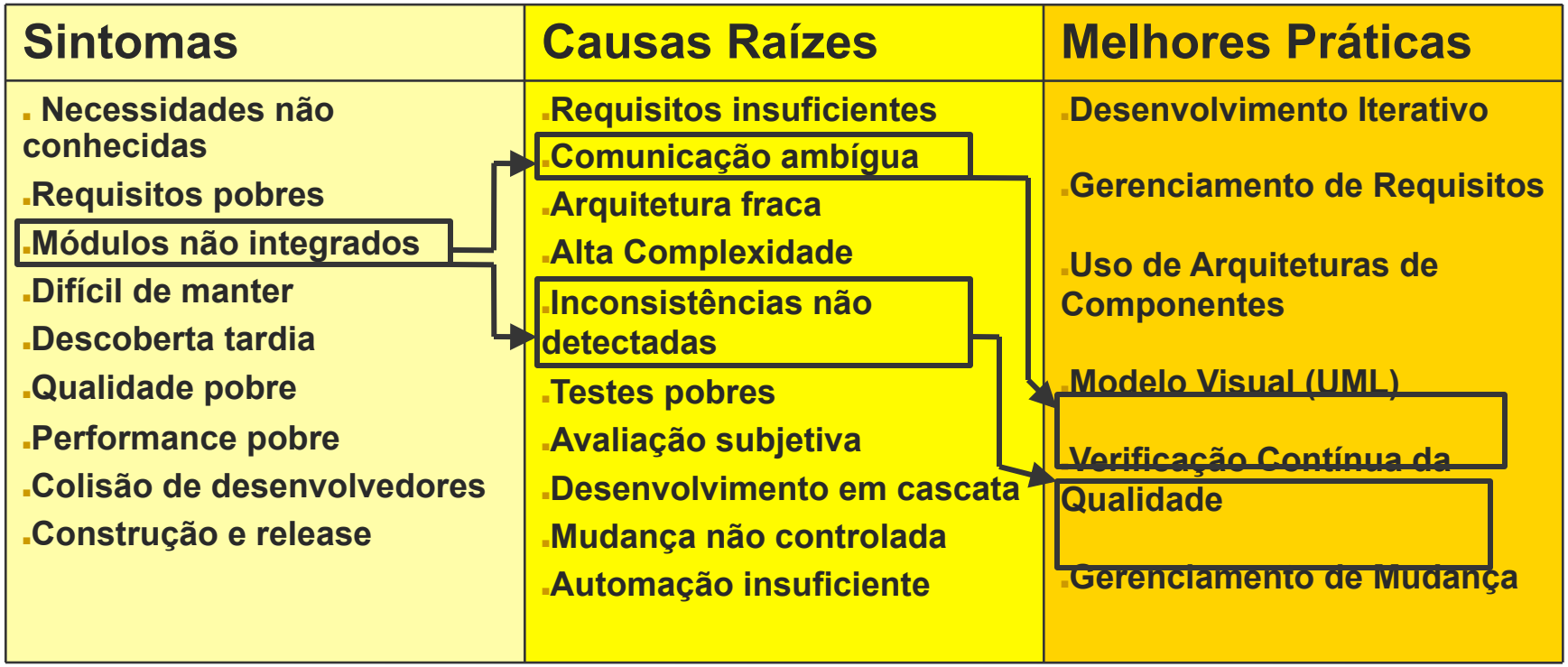
- Prática 1: Desenvolvimento Iterativo
- Prática 2: Gerenciamento de Requisitos
- Prática 3: Uso de Arquiteturas de Componentes
- Prática 4: Modelagem Visual (UML)
- Prática 5: Verificação Contínua da Qualidade
- Prática 6: Gerenciamento de Mudança

O RUP dentro do contexto das seis melhores práticas

Sintomas dos problemas de desenvolvimento de software

- Necessidades do usuário ou negócios não conhecidas
- Requisitos pobres
- Módulos não integrados
- Manutenção difícil
- Descoberta tardia de defeitos
- Qualidade e experiência do usuário final pobres
- Pobre performance de carga
- Esforços da equipe não coordenados
- Questões de construção e lançamento de release

Sintomas traçados para as causas raízes



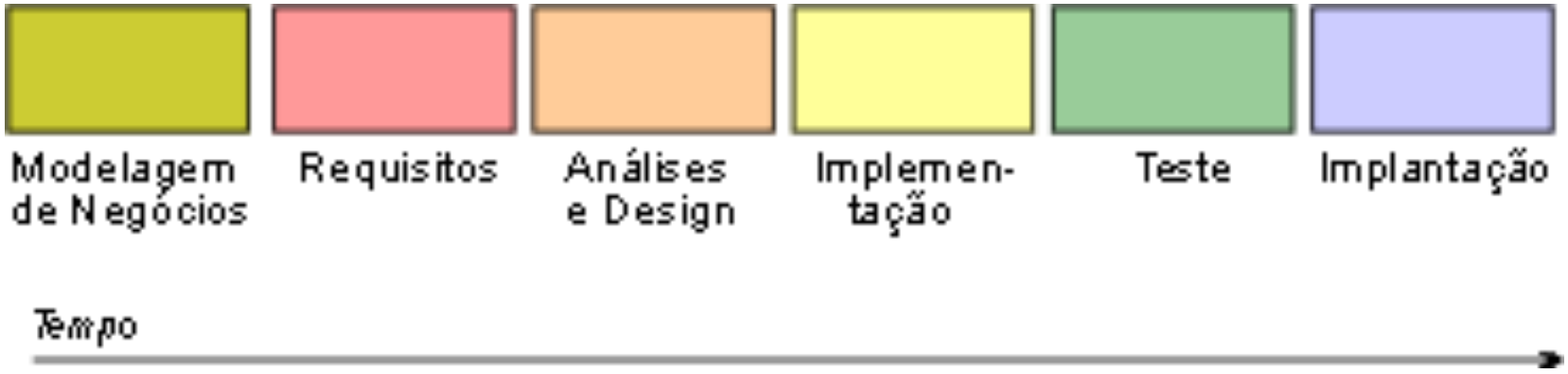
Prática 1: Desenvolvimento Iterativo

Melhores Práticas

- **Desenvolvimento Iterativo**
- Gerenciamento de Requisitos
- Uso de Arquiteturas de Componentes
- Modelagem Visual (UML)
- Verificação Contínua da Qualidade
- Gerenciamento de Mudanças

Característica do Desenvolvimento em Cascata (*Waterfall*)

- Atrasos na confirmação de resolução de riscos críticos
- Mensurar o progresso pela avaliação dos produtos de trabalho que foram mal estimados
- Atrasos, integração e testes agregados
- Impedimentos próximos à entrega
- Freqüentemente resulta em iterações não planejadas



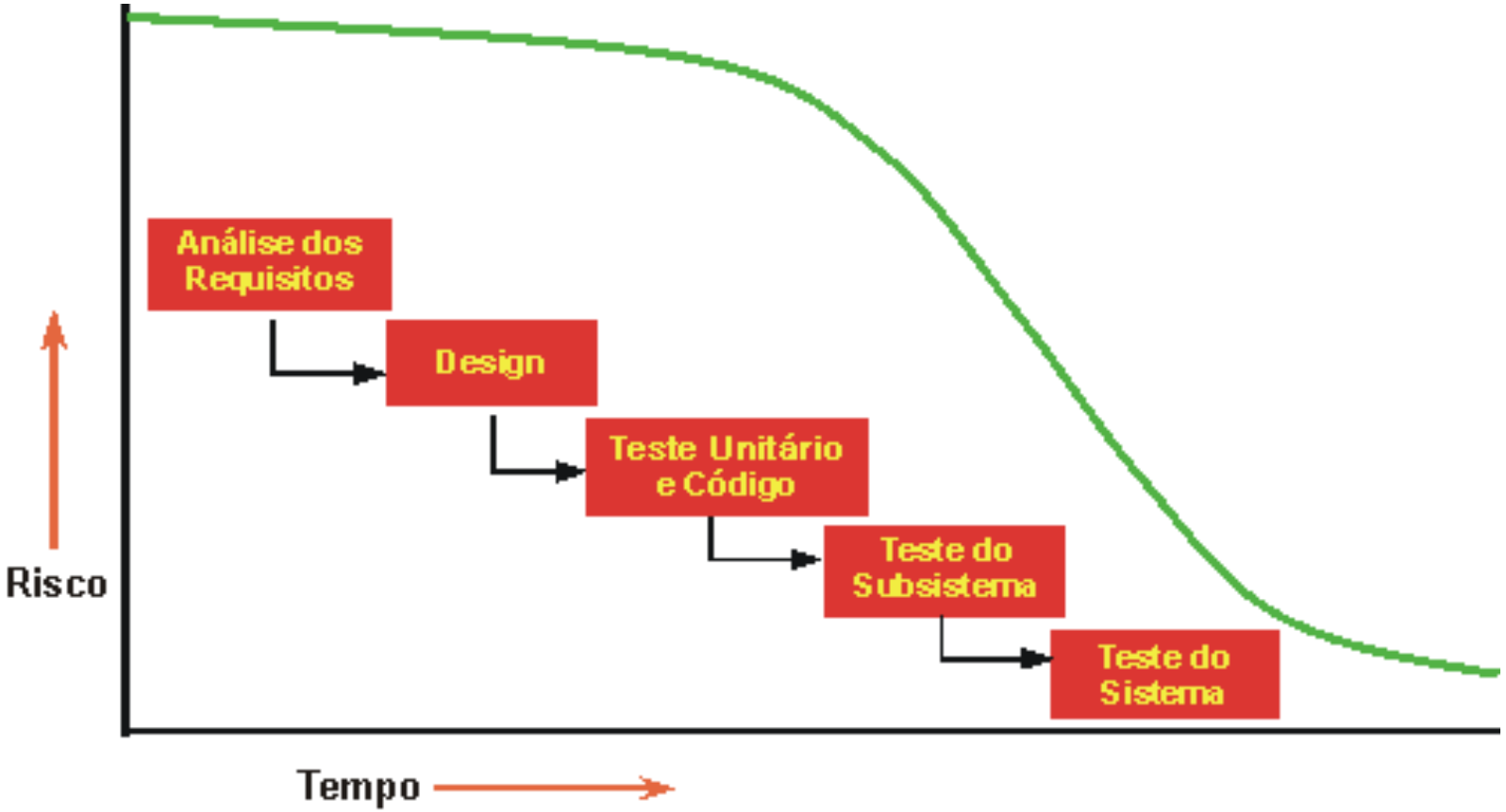
Desenvolvimento Iterativo Produz um Executável

Processo Iterativo

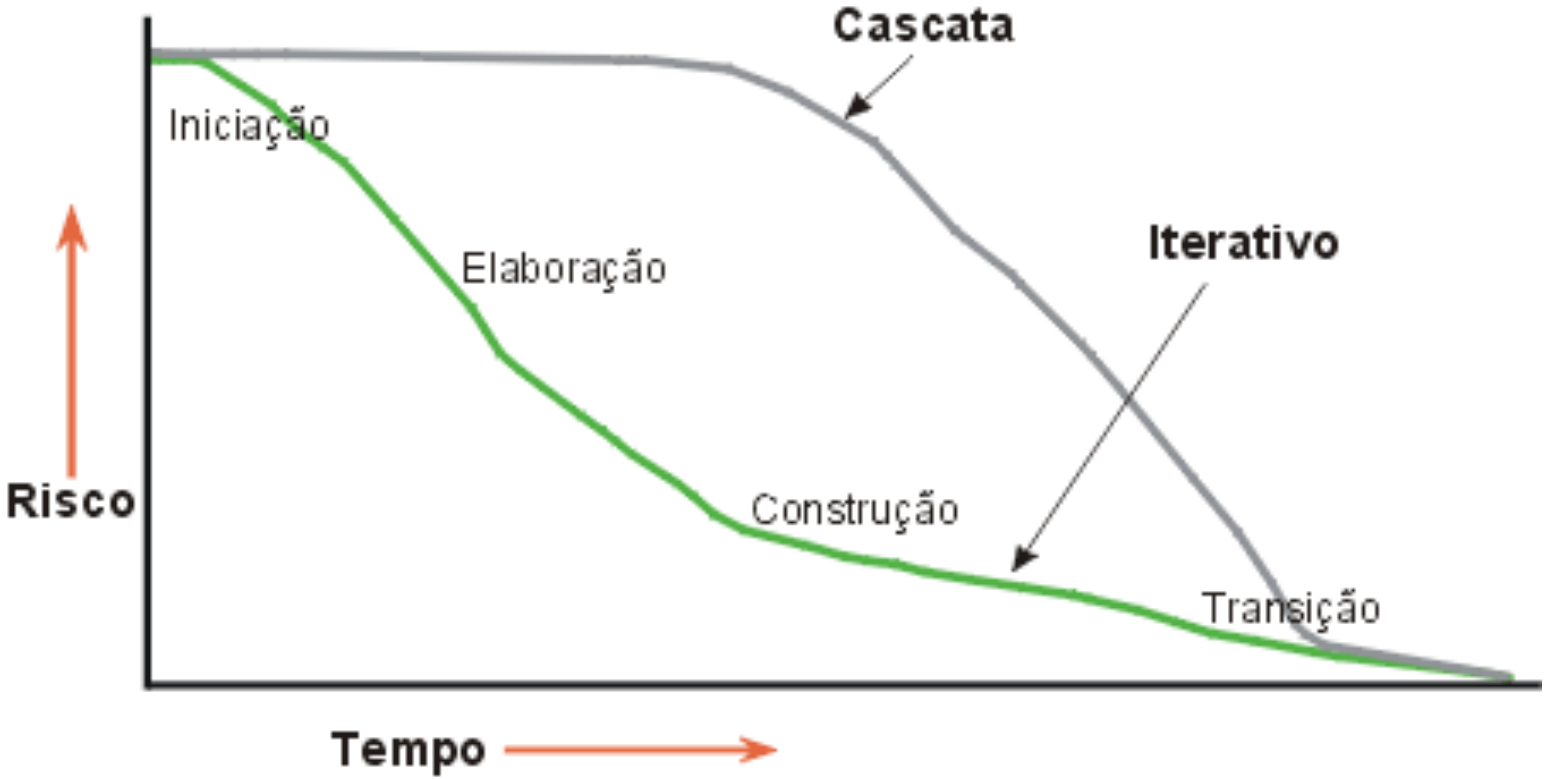
Iterações auxiliam a:

- Resolver principais riscos antes de fazer grandes investimentos.
- Possibilitar feedback antecipado do usuário.
- Fazer teste e integração continuamente.
- Focar o projeto com objetivos nos Milestones.
- Tornar possível a implantação de implementações parciais.

Perfil dos Riscos

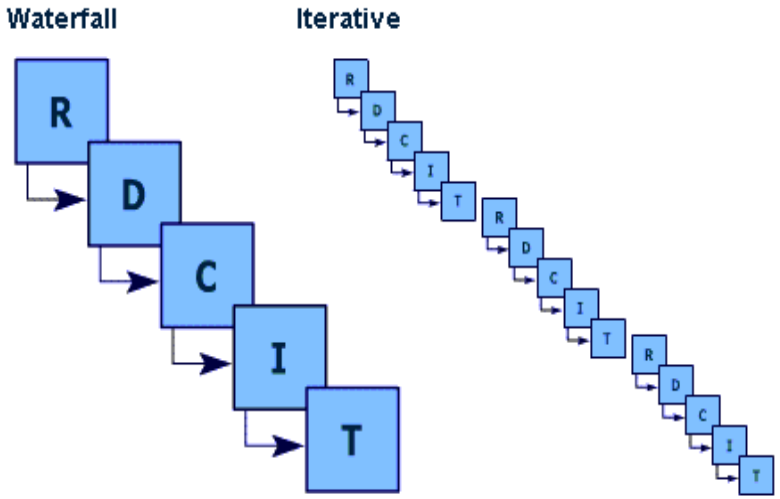


Perfil dos Riscos



Iteração e Waterfall

- Cada iteração é um processo completo de um waterfall em miniatura, iniciando com a reavaliação dos requisitos e terminando com um software executável e integrado.
- Uma iteração constrói uma parte do sistema e possibilita que as iterações subsequentes sejam refocadas ou ajustadas baseadas nas lições aprendidas.



Prática 2: Gerenciamento de Requisitos

Melhores Práticas

- Desenvolvimento Iterativo
- **Gerenciamento de Requisitos**
- Uso de Arquiteturas de Componentes
- Modelo Visual (UML)
- Verificação Contínua da Qualidade
- Gerenciamento de Mudanças

Gerenciamento de Requisitos

Tenha certeza de que você:

- **Resolve o problema certo**
- **Constrói o sistema certo**

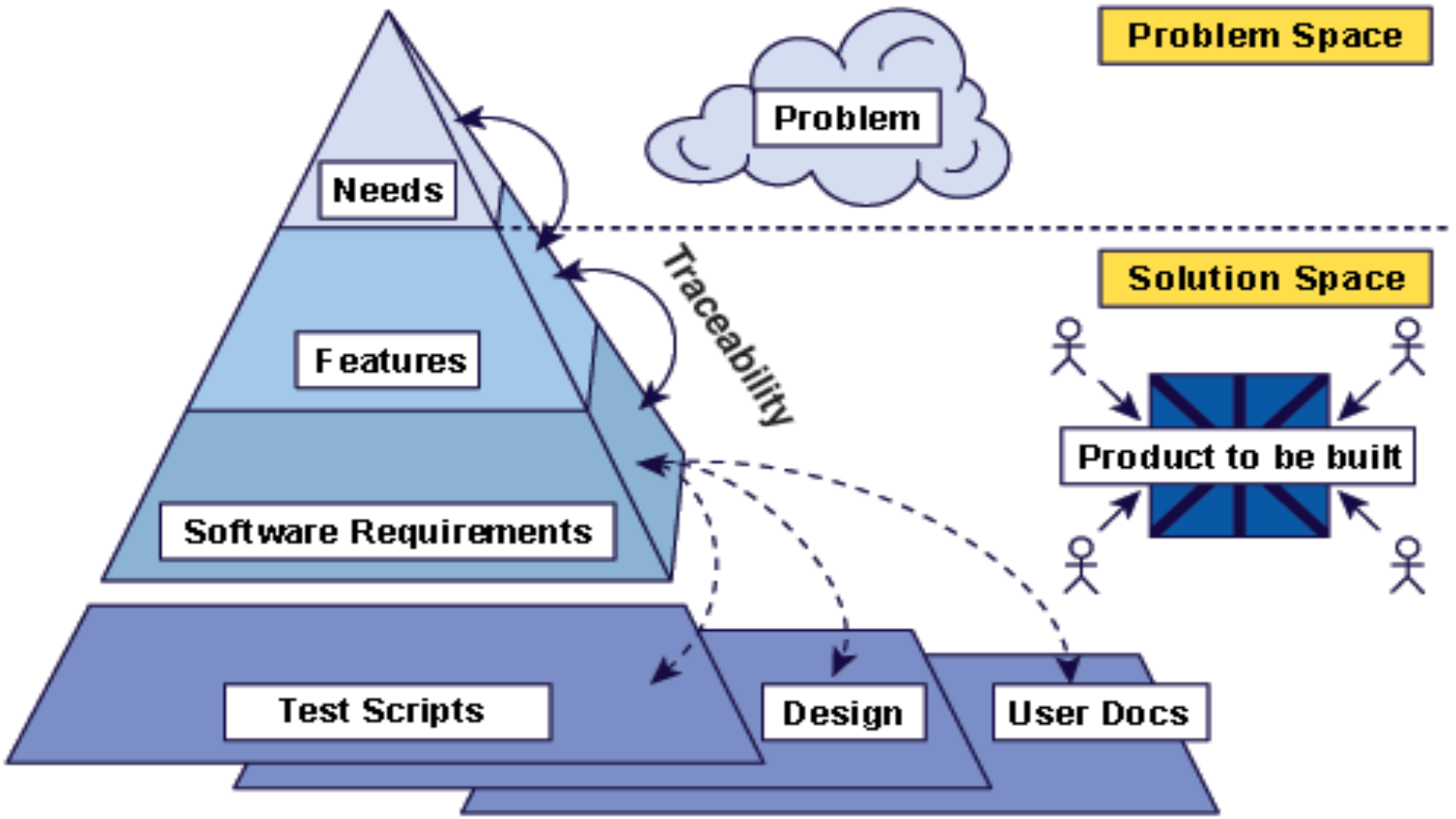
Tendo uma abordagem sistemática para:

- **Sistematização**
- **Organização**
- **Documentação**
- **Gerenciamento**

Aspectos do Gerenciamento de Requisitos

- Analisar o problema
- Entender as necessidades do usuário
- Definir o sistema
- Gerenciar o escopo
- Refinar a definição do sistema
- Gerenciar as mudanças dos requisitos

Mapa de território



Prática 3: Uso de Arquiteturas de Componentes

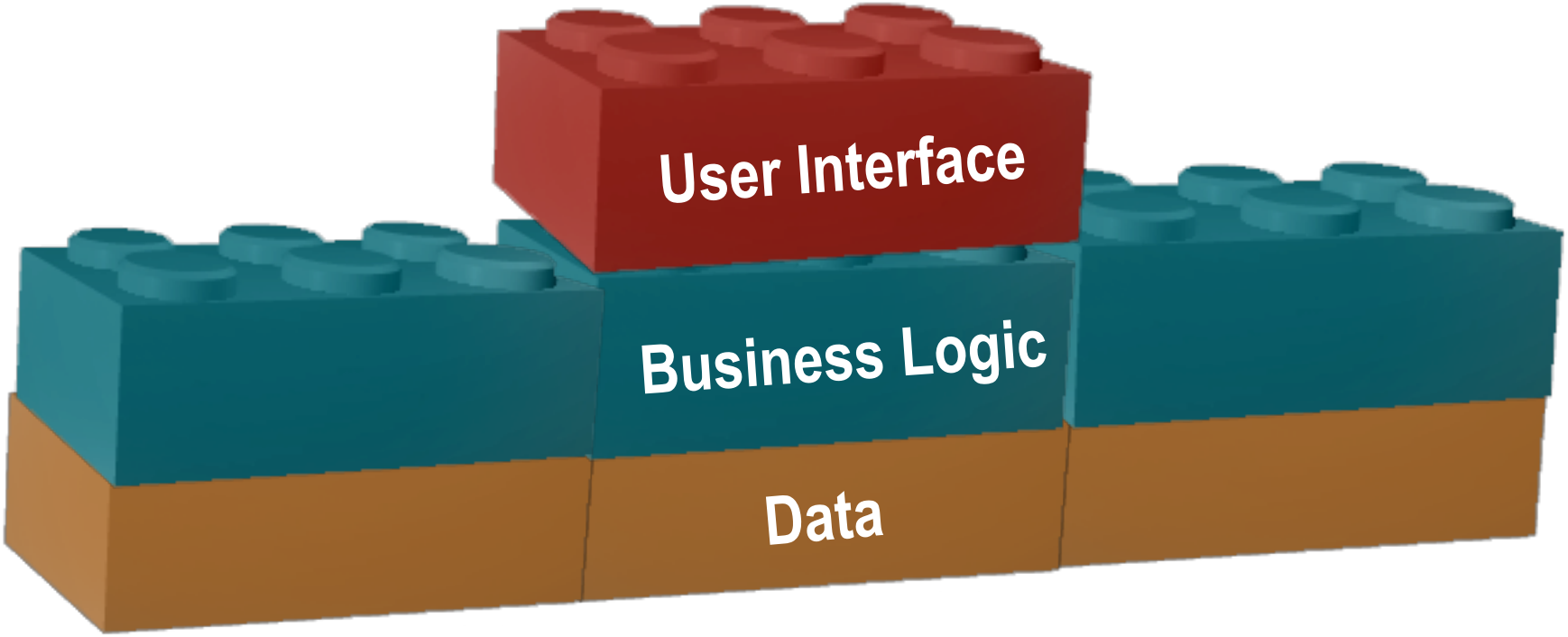
Melhores Práticas

- Desenvolvimento Iterativo
- Gerenciamento de Requisitos
- **Uso de Arquiteturas de Componentes**
- Modelagem Visual (UML)
- Verificação Contínua da Qualidade
- Gerenciamento de Mudanças

Arquitetura

Componentes

- Suporta mudanças de linguagem e plataforma
- Adapta-se as futuras necessidades e tecnologias

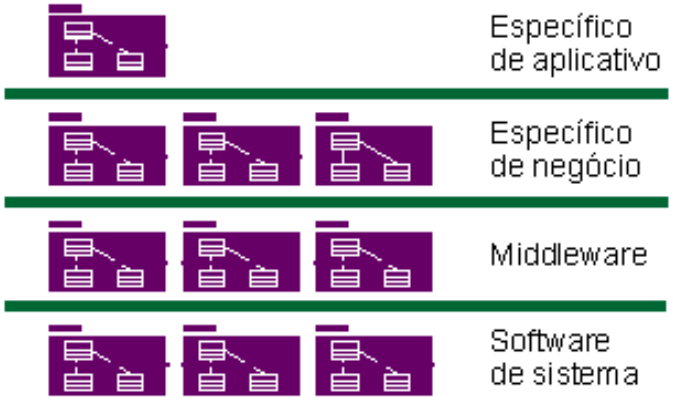


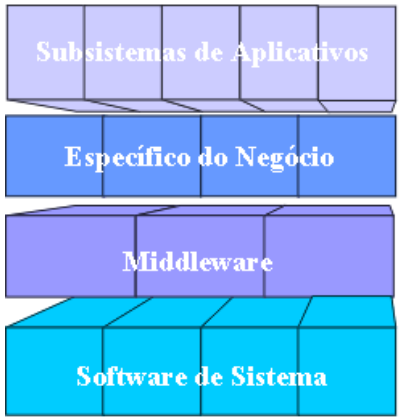
Flexibilidade das Arquiteturas Baseadas em Componentes

- **Flexibilidade**
  - Conhece requisitos atuais e futuros
  - Melhora a extensibilidade
  - Possibilita reuso
  - Encapsula as dependências do sistema
- **Baseado em Componente**
  - Reuso ou customização de componentes
  - Seleção de componentes disponíveis comercialmente
  - Evolui incrementalmente os software existentes

Propósito de uma arquitetura baseada em componente

- Base para reuso
  - Reuso de componente
  - Reuso de arquitetura
- Base para gerenciamento de projeto
  - Planejamento
  - Modularização
  - Alocação de pessoal
  - Entrega
- Controle intelectual
  - Gerenciamento de complexidade
  - Gerenciamento de integridade





Subsistemas de Aplicativos distintos que compõem um aplicativo – contêm o software que agrega valor desenvolvido pela organização.

Específico do Negócio – contém um número de sistemas reutilizáveis específicos do tipo de negócio.

Middleware - oferece subsistemas para plataforma e classes de utilitários – serviços independentes para computação de objetos distintos em ambientes heterogêneos e outros.

Software do Sistema - contém o software para a infra-estrutura real como, por exemplo, sistemas operacionais, interfaces para hardware específico, drivers de dispositivos etc.

Prática 4: Modelo Visual (UML)

Melhores Práticas

- Desenvolvimento Iterativo
- Gerenciamento de Requisitos
- Uso de Arquiteturas de Componentes
- **Modelagem Visual (UML)**
- Verificação Contínua da Qualidade
- Gerenciamento de Mudanças

Por que Modelagem Visual?

Para:

- **Capturar a estrutura e o comportamento**
- **Mostrar como os elementos ficam juntos**
- **Manter o design e implementação consistentes**
- **Esconder ou expor detalhes conforme apropriado**
- **Promover comunicação não ambígua: A UML fornece uma linguagem para todos os participantes**

A Modelagem é importante porque ela ajuda a equipe de desenvolvimento a visualizar, especificar, construir e documentar a estrutura e comportamento de uma arquitetura do sistema. Ajuda a gerenciar a complexidade do software.

Modelagem Visual com Unified Modeling Language

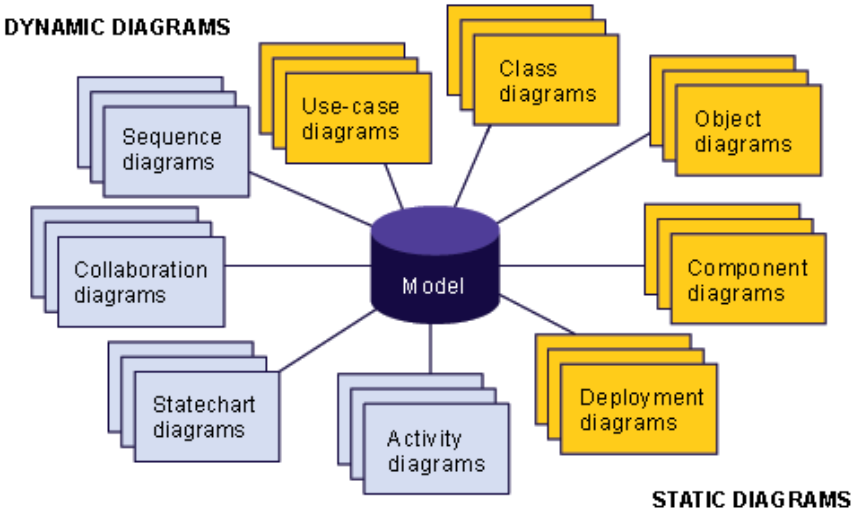
- Múltiplas visões
- Sintaxe e semântica precisas

Diagramas Dinâmicos

- Diagramas de Seqüência
- Diagramas de Colaboração
- Diagramas de Estado
- Diagramas de Atividade

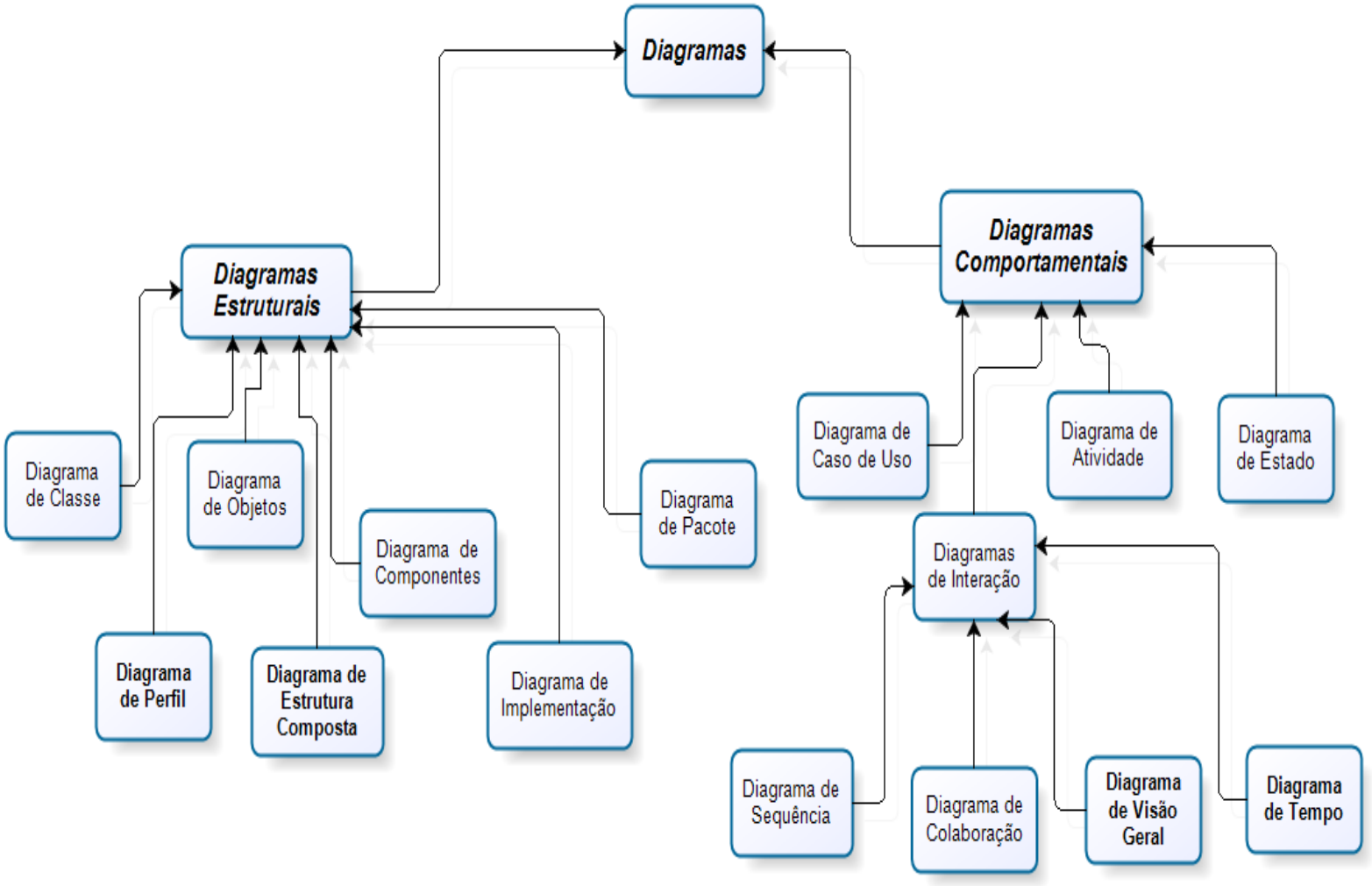
Diagramas Estáticos

- Diagramas de Caso de Uso
- Diagramas de Classe
- Diagramas de Objeto
- Diagramas de Componentes
- Diagramas de Implantação



Tipos de Modelagem - V. 2.4

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



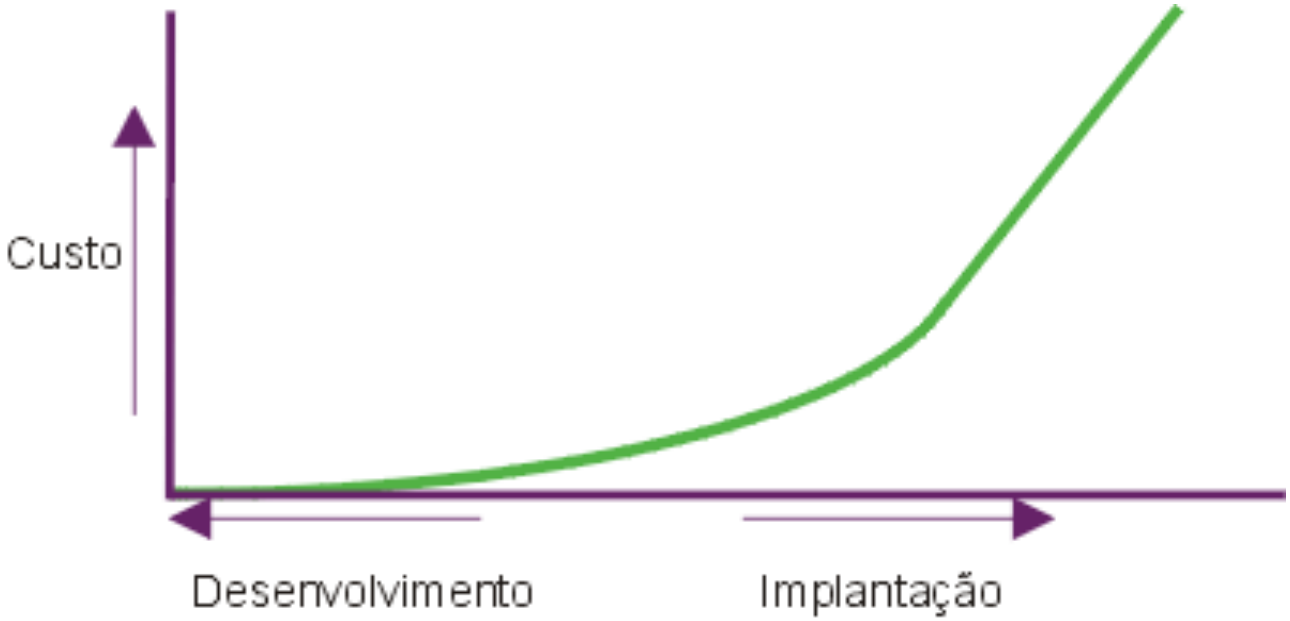
Prática 5: Verificação Contínua da Qualidade

Melhores Práticas

- Desenvolvimento Iterativo
- Gerenciamento de Requisitos
- Uso de Arquiteturas de Componentes
- Modelo Visual (UML)
- **Verificação Contínua da Qualidade**
- Gerenciamento de Mudanças

Prática 5: Verificação Contínua da Qualidade

Os problemas de software são 100 a 1000 vezes mais custosos para encontrar e reparar após a implantação. A verificação e o gerenciamento da qualidade durante o ciclo de vida do projeto é essencial para atingir os objetivos corretos no momento certo.



Prática 5: Verificação Contínua da Qualidade

- Para as finalidades do Rational Unified Process (RUP), a qualidade é definida como:
- "...a característica de ter demonstrado a realização da criação de um produto que atende ou excede os requisitos acordados, conforme avaliado por medidas e critérios acordados, e que é criado em um processo acordado."

Testando as dimensões da qualidade

- Funcionalidade – testa cada cenário usado
- Usabilidade – testa a aplicação da perspectiva do usuário final
- Confiabilidade – testa consistência e previsibilidade da aplicação
- Performance – testa a resposta online sobre pico e carga
- Suportabilidade – testa a habilidade para manutenção e suporte dos produtos

Teste a cada iteração

- Em cada iteração testes automatizados são criados. Como novos requisitos são adicionados em iterações subseqüentes, novos testes são gerados e executados.

Teste dentro do ciclo de vida do desenvolvimento do produto

- O ciclo de vida do teste é uma parte do ciclo de vida do software.
- O processo de design de desenvolvimento de testes pode ser tão complexo e árduo quanto o processo de desenvolvimento de software.

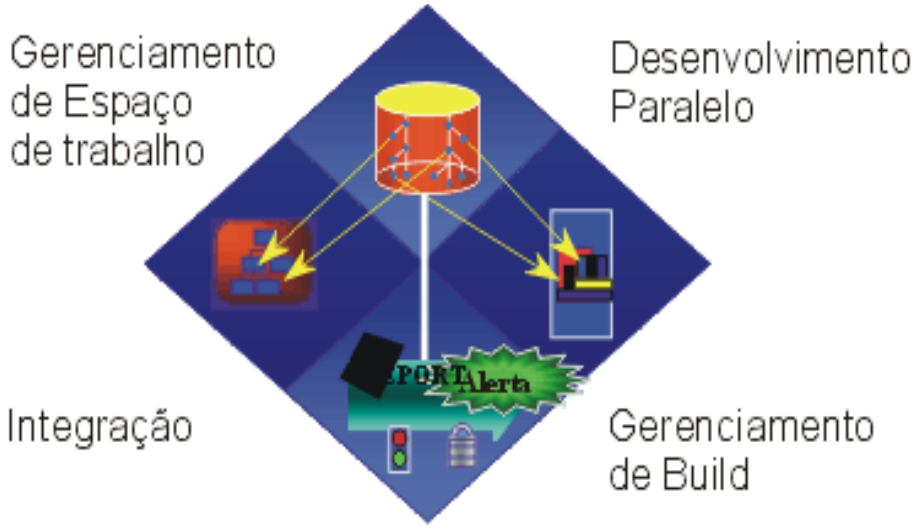
Prática 6: Gerenciamento de mudanças

Melhores Práticas

- Desenvolvimento Iterativo
- Gerenciamento de Requisitos
- Uso de Arquiteturas de Componentes
- Modelagem Visual (UML)
- Verificação Contínua da Qualidade
- **Gerenciamento de Mudança**

O que fazer para controlar?

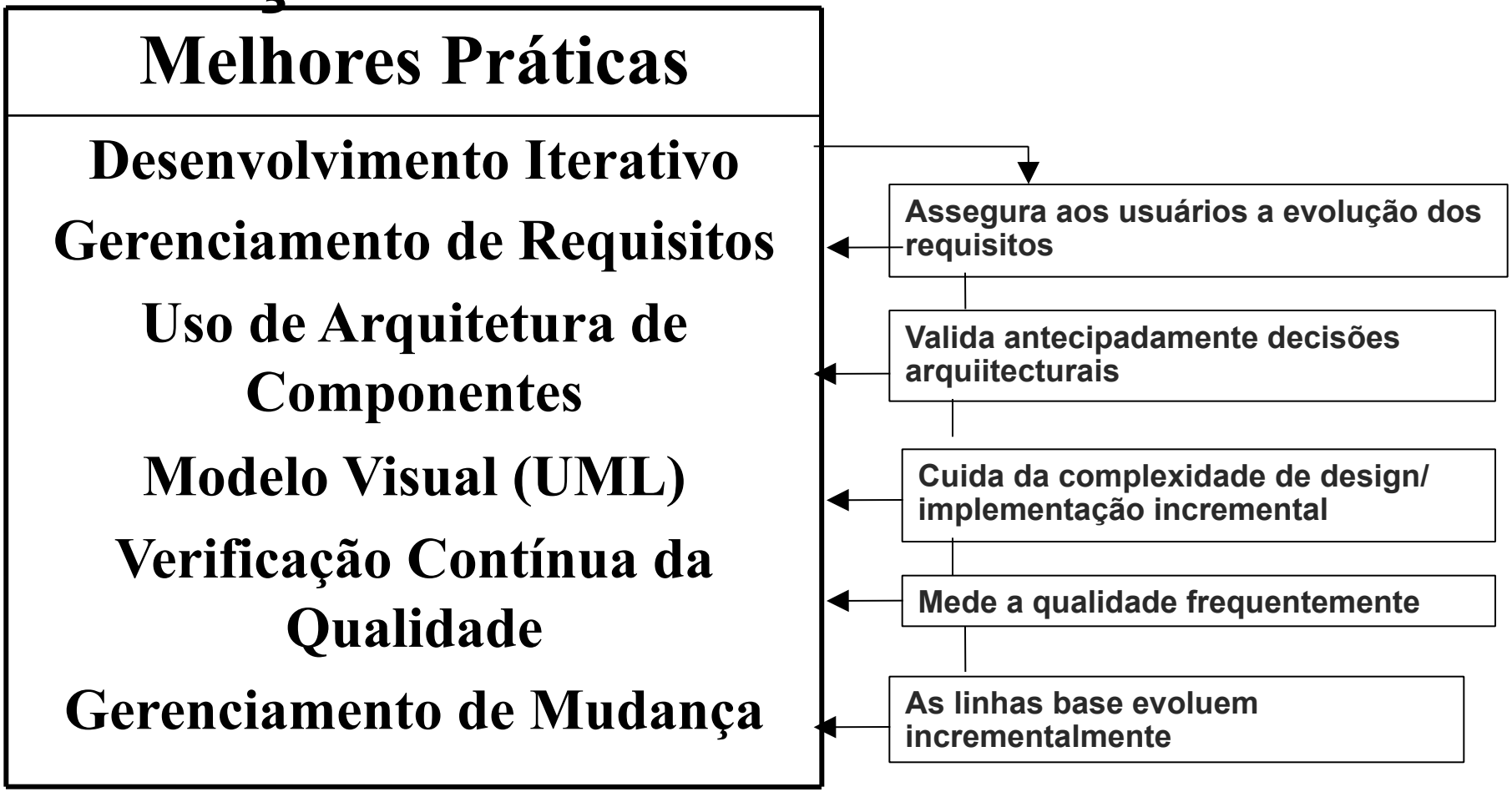
- Espaço de trabalho seguro para cada desenvolvedor
- Gerenciamento de construção/ integração automatizado
- Desenvolvimento paralelo



Aspectos de um sistema de gerenciamento de mudança

- Gerenciamento de requisição de mudança (CRM – Change Request Management)
- Relatando o status da configuração
- Gerenciamento de Configuração (CM)
- Rastreamento de Mudança
- Seleção de Versão

As Melhores Práticas se Reforçam



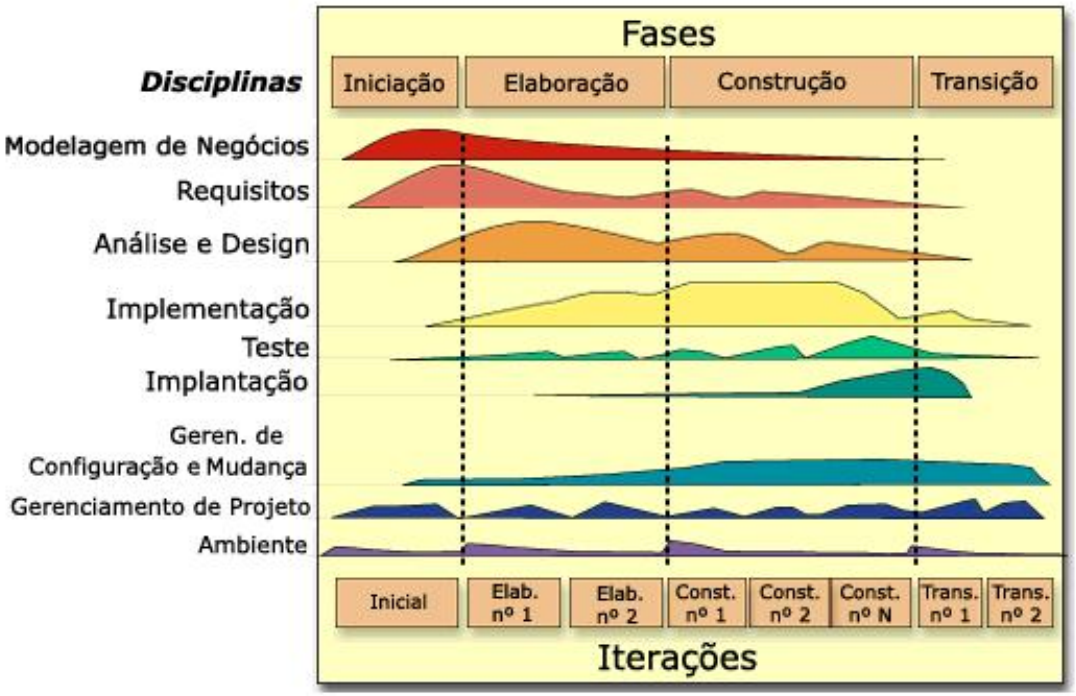
Implementando as Melhores Práticas

Por que ter um processo?

- Fornece diretrizes para o desenvolvimento eficiente de software de qualidade
- Reduz o risco e aumenta previsibilidade
- Promove uma visão e cultura comuns
- Captura e institucionaliza as melhores práticas

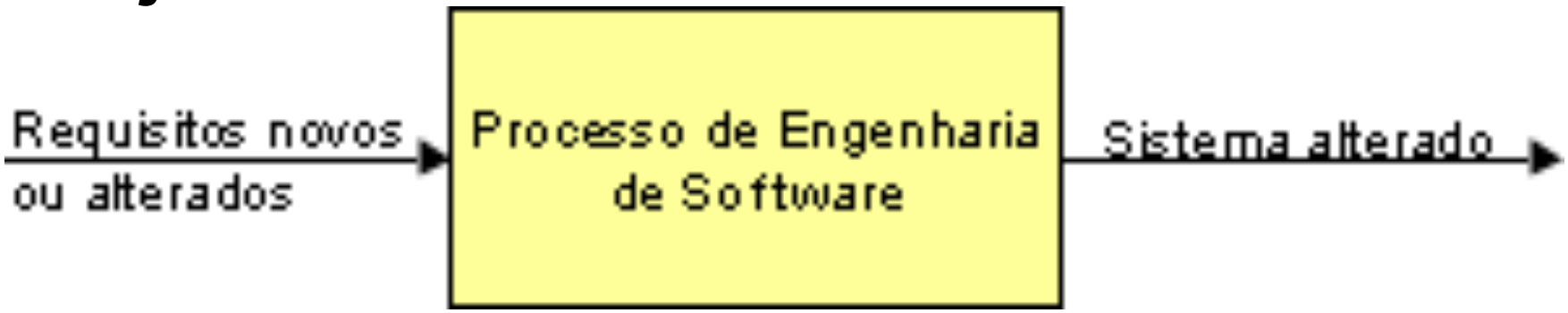
Alcançando as Melhores Práticas

- Abordagem iterativa
- Guia para atividades e artefatos
- Processo focado na arquitetura
- Casos de uso que dirige o design e implementação
- Modelos que abstraem o sistema



Uma definição baseado em equipe do processo

Um processo define QUEM está fazendo O QUE, QUANDO, e COMO, em ordem para alcançar um certo objetivo.



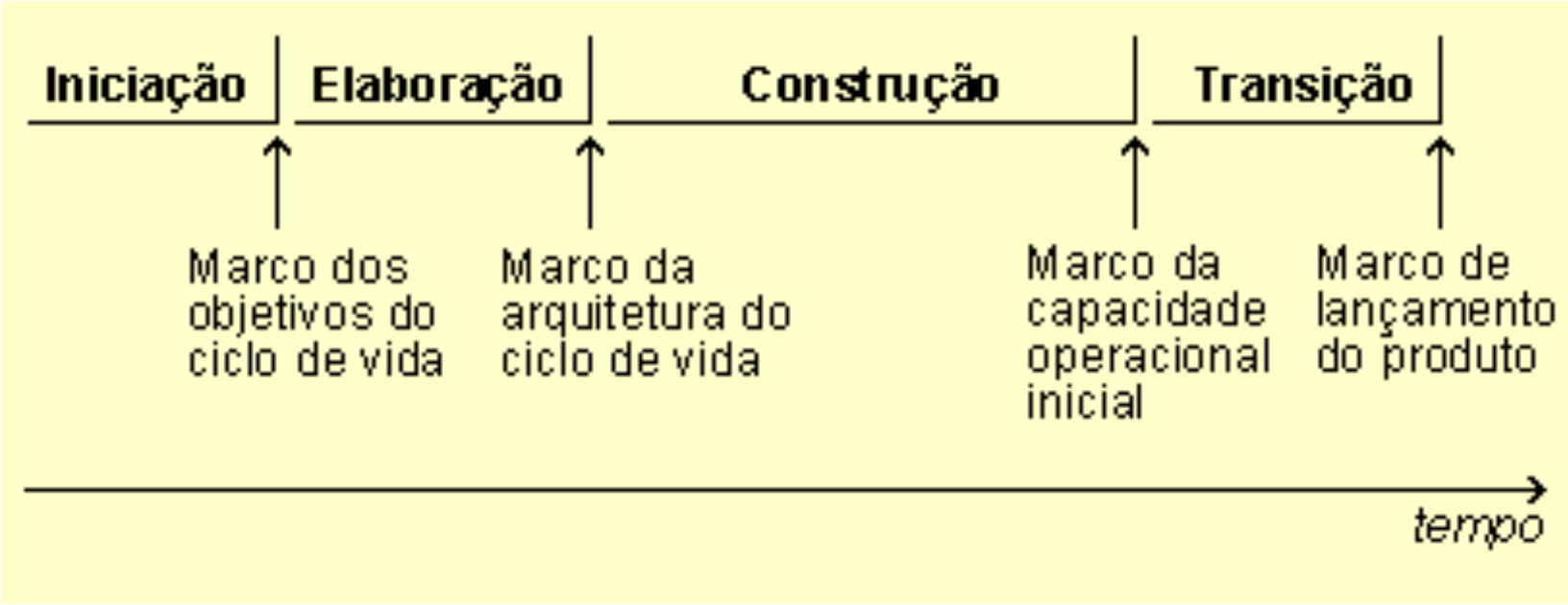
O processo de engenharia de software é o processo de desenvolvimento de um sistema a partir dos requisitos, sejam eles novos (ciclo de desenvolvimento inicial) ou alterados (ciclo de evolução).

Estrutura do processo – fases do ciclo de vida

O RUP tem quatro fases:

- Início – define o escopo de projeto
- Elaboração – plano do projeto, especifica funcionalidades, arquitetura básica
- Construção – construção do produto
- Transição – transição do produto dentro da comunidade do usuário final

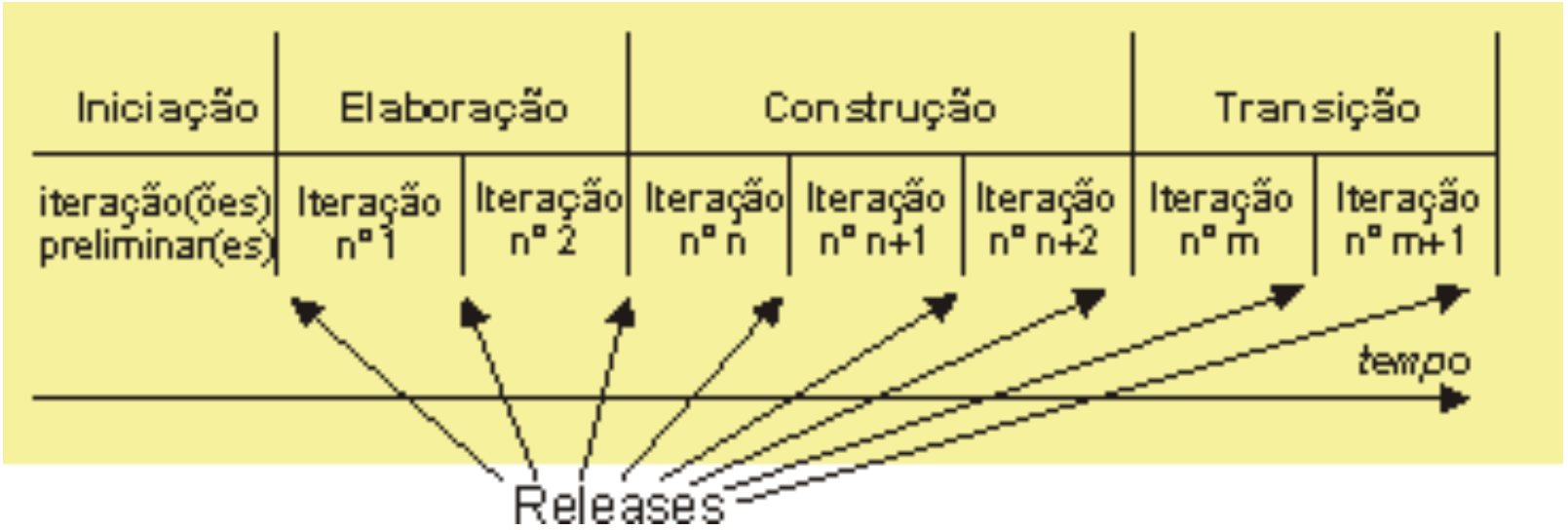
Fronteiras das fases e principais marcos



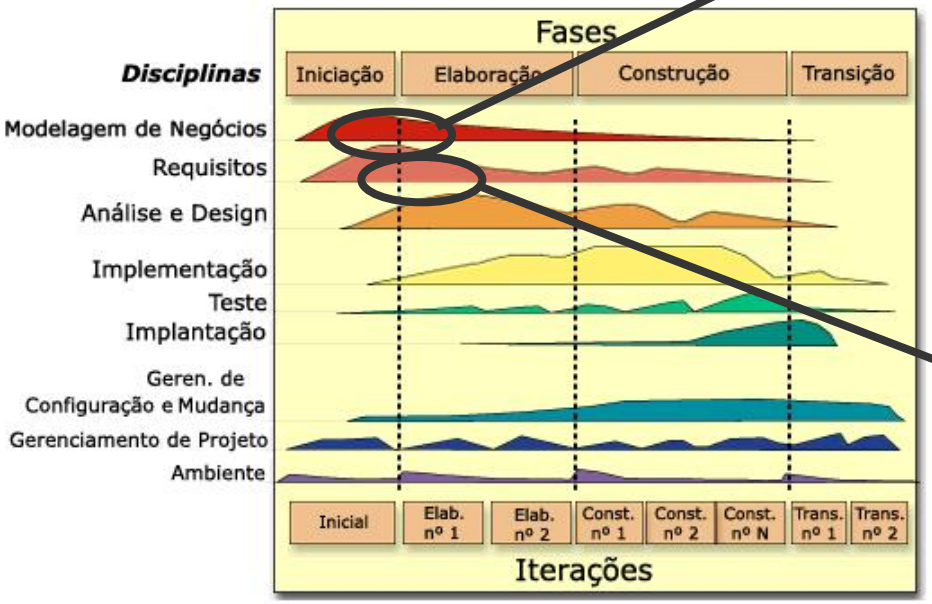
A cada marco principal, é revisado o projeto e decidido se prosseguimos com o projeto como planejado, se abortamos o projeto, ou se o revisamos. O critério usado faz esta decisão variar por fase.

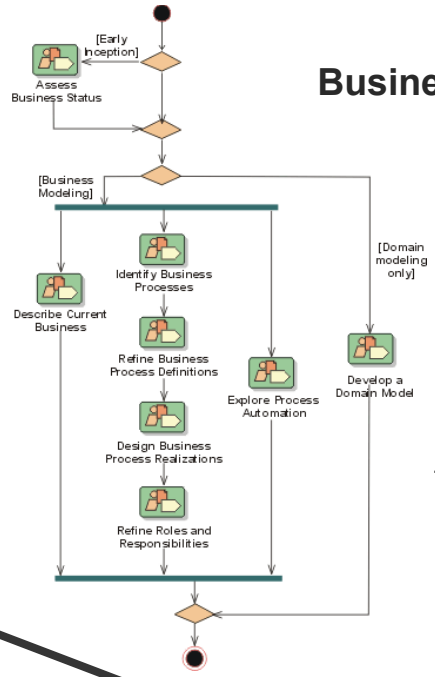
Iterações e Fases

- Uma iteração é uma seqüência distinta de atividades baseadas em um plano estabelecido e um critério de avaliação, resultando em um release executável (interno ou externo).

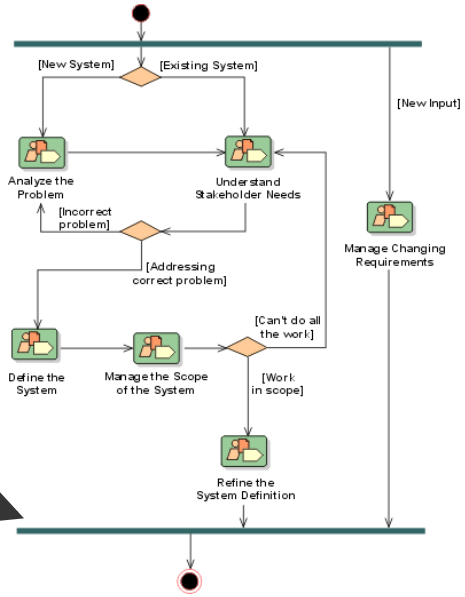


As disciplinas guiam o desenvolvimento iterativo





Business Modeling: workflow



Requirements: workflow

Princípios Chave para o Desenvolvimento Orientado a Negócios

- As melhores práticas testadas e aprovadas do RUP tem sido a base para a evolução de ferramentas e processos por mais de uma década.
- Hoje, como o desenvolvimento de software está se tornando uma capacidade de negócios chave, nossas melhores práticas estão amadurecendo dentro de um contexto de desenvolvimento orientado a negócios.
- Os seis princípios, abaixo, re-articulam nossas melhores práticas para um ciclo de vida amplo de sistemas continuamente evoluindo, nos quais o elemento principal de evolução é o software:

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

Processos Ágeis

- XP
- Scrum
- OpenUP
- AUP
- DSDM
- Lean Software Development
- Adaptive Software Development
- Rational Unified Process (RUP) – se adapta e atende aos Processos Ágeis
- MSF
- FDD
- Crystal Clear
- EssUP e

Revisão

- As melhores práticas guiam a engenharia de software pelas causas raízes.
- As melhores práticas se reforçam.
- O processo guia a equipe em quem faz o que, quando, e como.
- O RUP é uma maneira de alcançar as melhores práticas.

Gerência de Requisitos

Requisitos

É definido como uma condição ou capacidade com a qual o sistema deve estar de acordo.

Gerência de Requisitos

- **Funcionais**
  - Especifica ações que um sistema deve ser capaz de executar.

- **Não Funcionais**
  - Descreve os atributos do sistema ou atributos do ambiente de sistema.

Usabilidade, Confiabilidade, Performance, Suportabilidade

Gerência de Requisitos

Requisitos Funcionais

O que devem fazer ?

Um requisito funcional especifica ações que um sistema deve ser capaz de executar, sem levar em consideração os aspectos físicos, assim, especificando a entrada e o comportamento de produção de um sistema

20%

Gerência de Requisitos

Requisitos Não-Funcionais

O que devem possuir ?

Um requisito não funcional descreve os atributos do sistema ou atributos do ambiente de sistema. Representada pela sigla URPS: (Usabilidade, Confiabilidade, Performance, Suportabilidade).

80%

"URPS" do "FURPS"

Functionality **(funcionalidade)**	Capacidade do conjunto das características	Generalidades Segurança
Usability **(usabilidade)**	Fatores humanos Estética	Consistência Documentação
Reliability **(confiabilidade)**	Freqüência / Severidade de falhas recuperáveis	Previsibilidade Precisão MTBF
Performance	Rapidez Eficiência Uso de Recursos	Processamento Tempo de Resposta
Supportability **(suportabilidade)**	Testável Extensível Adaptável Manutenível Compatibilidade	Configurável Resistente Instalável Localizável Robusto

Especificando Requisitos de Usabilidade

- Definição de “Usabilidade"
  - É a facilidade que o software tem para entendimento e operação das intenções dos usuários.
- Requisitos de Usabilidade
  - Tempo de treinamento em requisitos, tarefa de medição de tempos
  - Habilidades do usuário (iniciante/avançado)
  - Comparação com outros sistema que os usuários conhecem e gostam
  - Sistema de ajuda *on-line*, tipos de ferramentas, necessidades de documentação
  - Conformidade com padrões
    - Exemplos: Windows, guias de estilo, padrões de GUI

Especificando Requisitos de Confiabilidade

- Definição de “Confiabilidade”
  - É a habilidade que o software deve comportar-se consistentemente de maneira aceitável pelo usuário
- Requisitos de Confiabilidade
  - Disponibilidade (xx.xx%)
  - Precisão
  - MTBF - média de tempo entre falhas (xx h)
  - Máximo de erros por/KLOC (0-x) (*Line Of Code*)
  - Erros por classe - crítico, importante, insignificante

Especificando Requisitos de Performance

- Definição de "Performance"
  - É uma medida de velocidade ou de executar eficientemente o sistema.
- Requisitos de Performance
  - Capacidade
  - Processamento
  - Tempo de resposta
  - Memória
  - Modo de degradação
  - Uso eficiente de recursos escassos
  - Processador, memória, disco, banda da rede

Especificando Requisitos de Suportabilidade

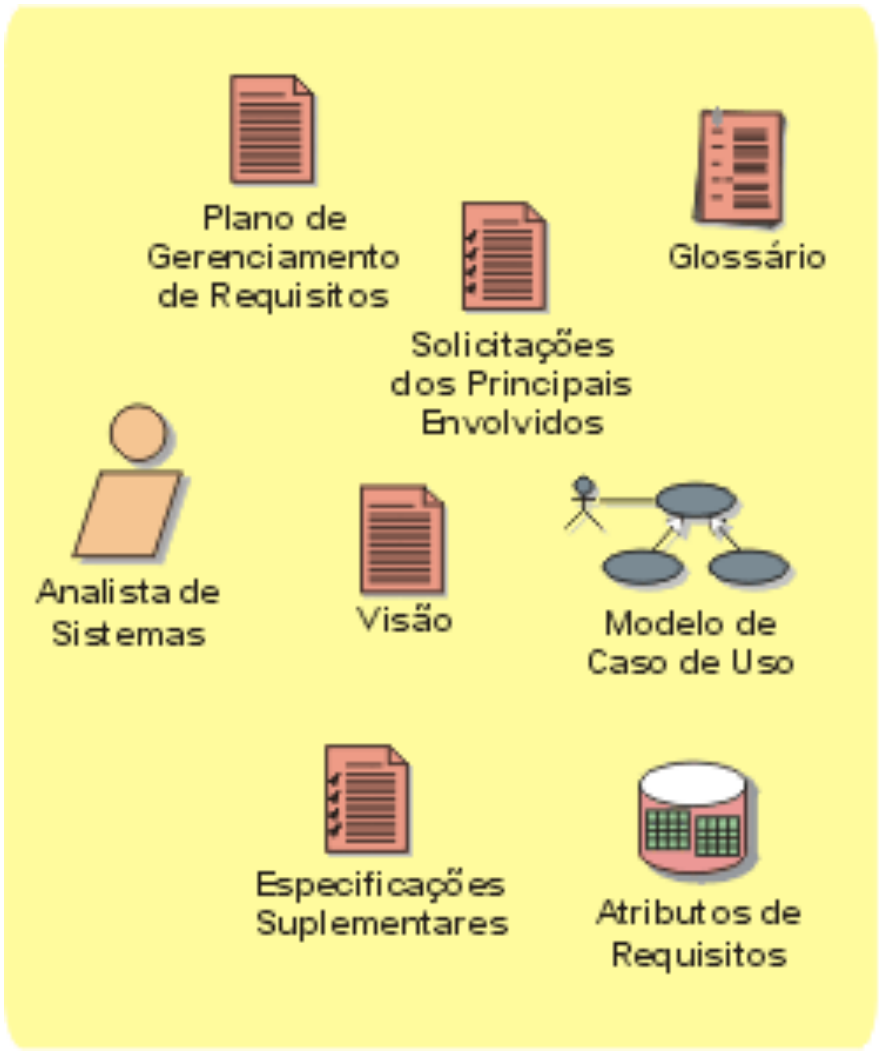
- Definição de "Suportabilidade"
  - É a habilidade que o software deve ter para acomodar facilmente melhorias e correções
- Requisitos de Suportabilidade
  - Linguagens, DBMS, ferramentas, etc.
  - Padrões de programação
  - Tratamento de erros e relatórios padrões
- Muita dificuldade para especificar
  - Se não for mensurável ou observável, ele não é um requisito
  - Ele é uma restrição do projeto?
  - Ele é uma intenção ou meta?

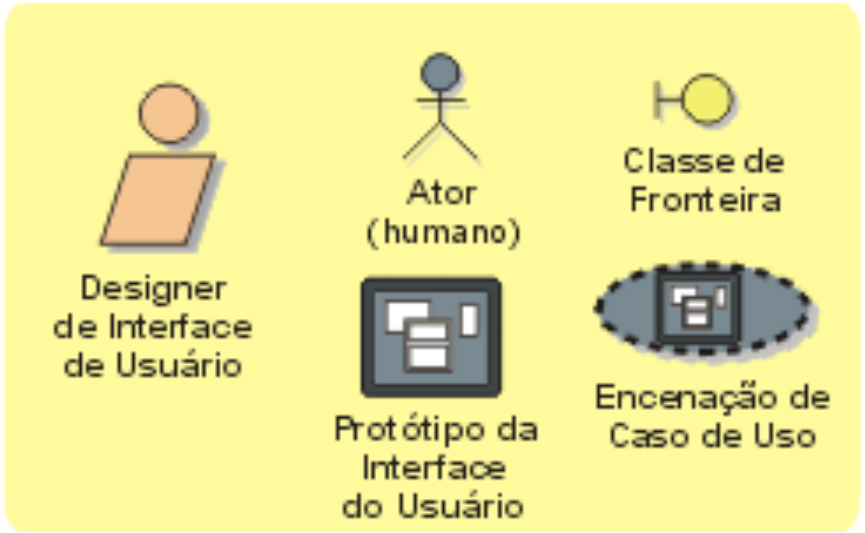
Artefatos

- **Artefatos** são produtos de trabalho finais ou intermediários produzidos e usados durante os projetos.

- Os artefatos são usados para capturar e transmitir informações do projeto.

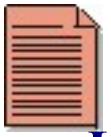
Papéis e artefatos



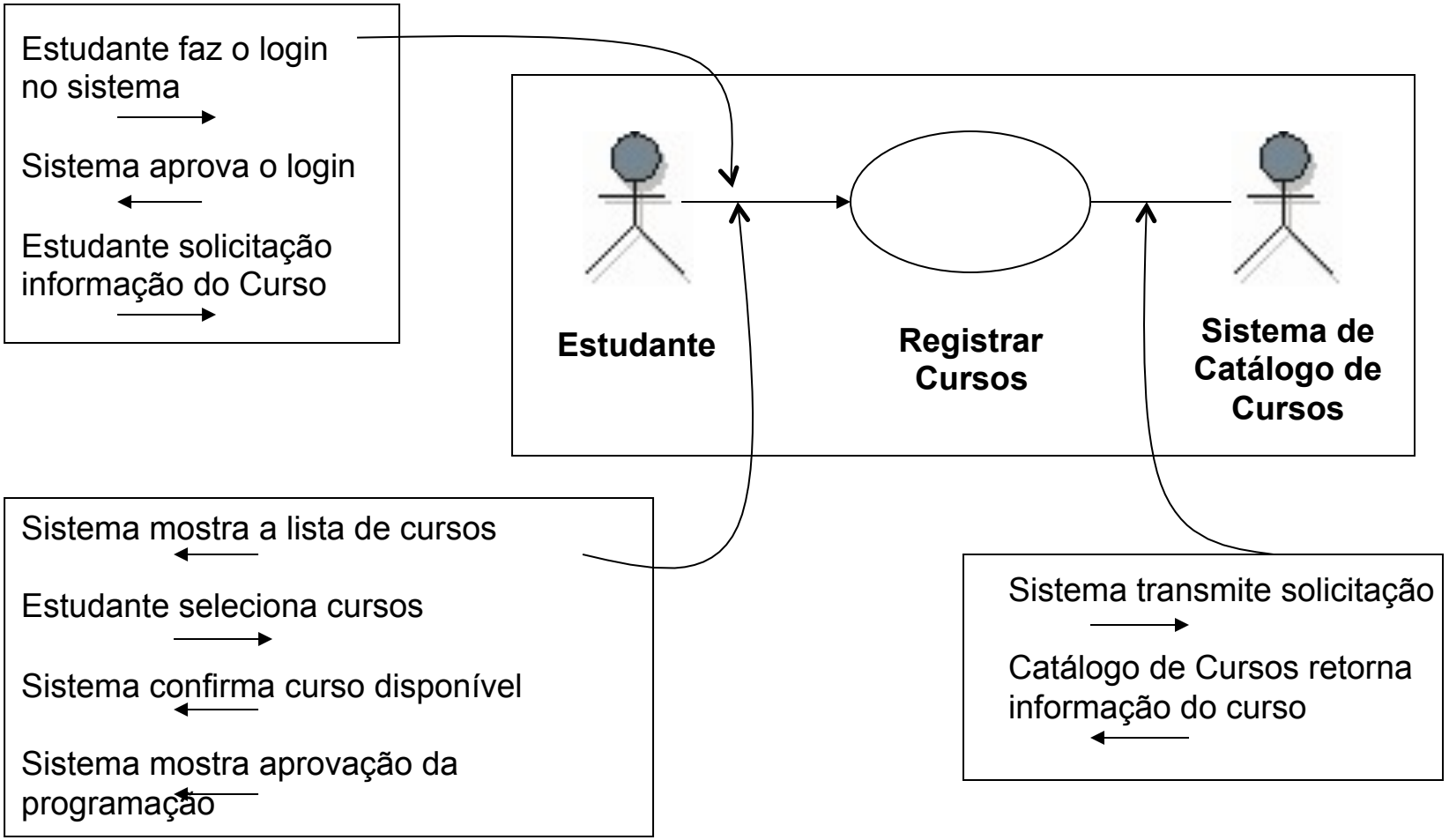


Artefatos

Quais são os artefatos usados para gerenciar requisitos?

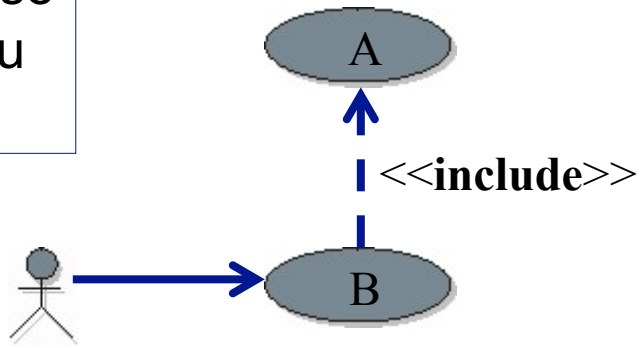
Onde documentamos nossos processos de requisitos?  **RM Plan**	**Onde listaremos os *stakeholders* e usuários?**  **Visão**
Onde definimos o problema?  **Visão**	**Onde identificamos os ambientes e plataformas?**  **Visão**
Onde manteremos os casos de uso (UC)?  **Espec. UC**	**Onde manteremos terminologias comum?**  **Glossário**

Cada Comunicação-Associação é um conjunto de diálogo



Casos deUso (Concreto versus Abst

Um caso de uso é abstrato ou concreto

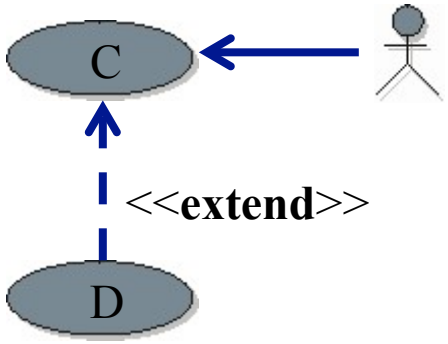


Casos de uso abstrato (A & D):

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

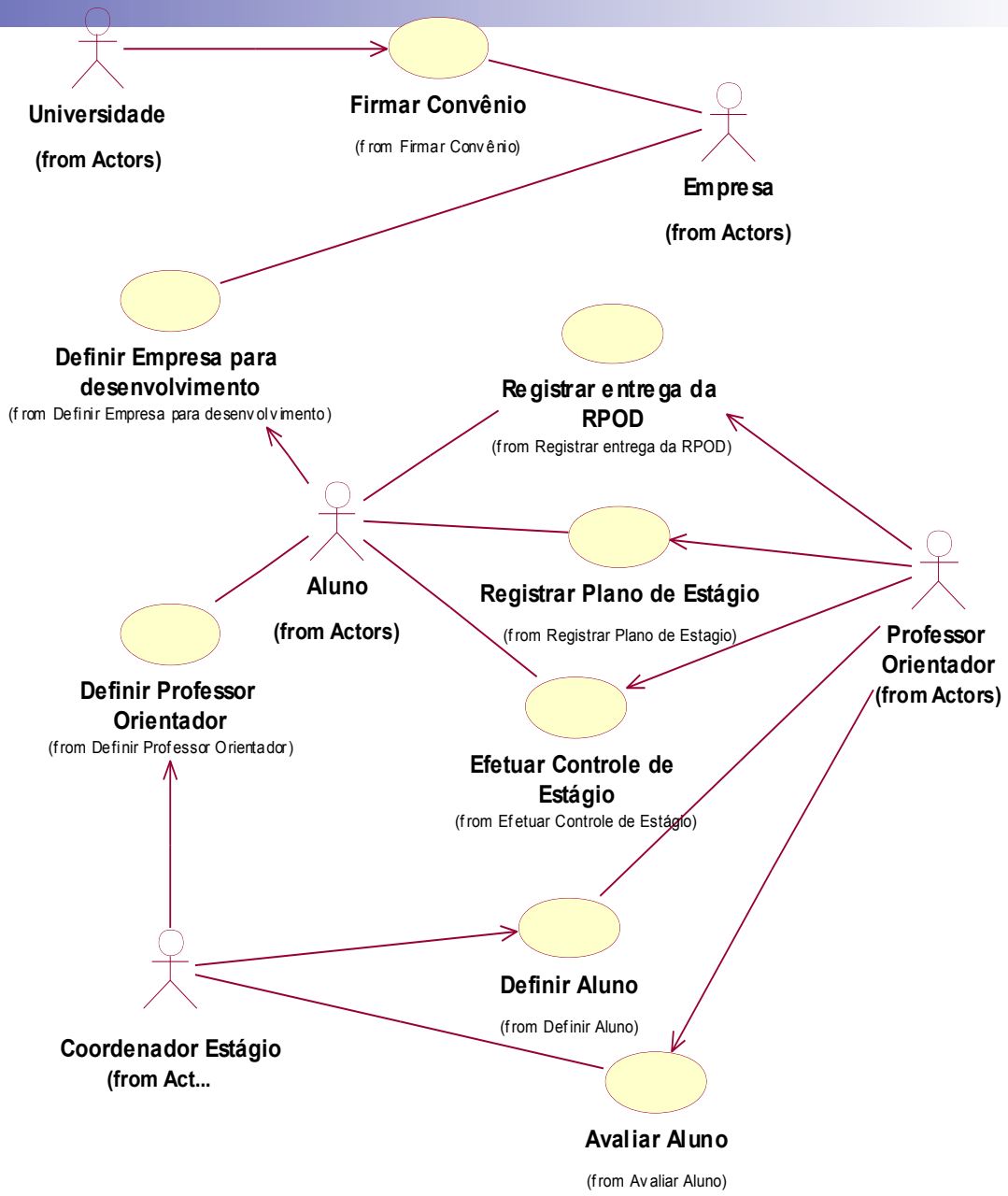
Casos de uso concreto (B & C):

- Deve ser completo e significativo
- Pode ser instanciado para si mesmo

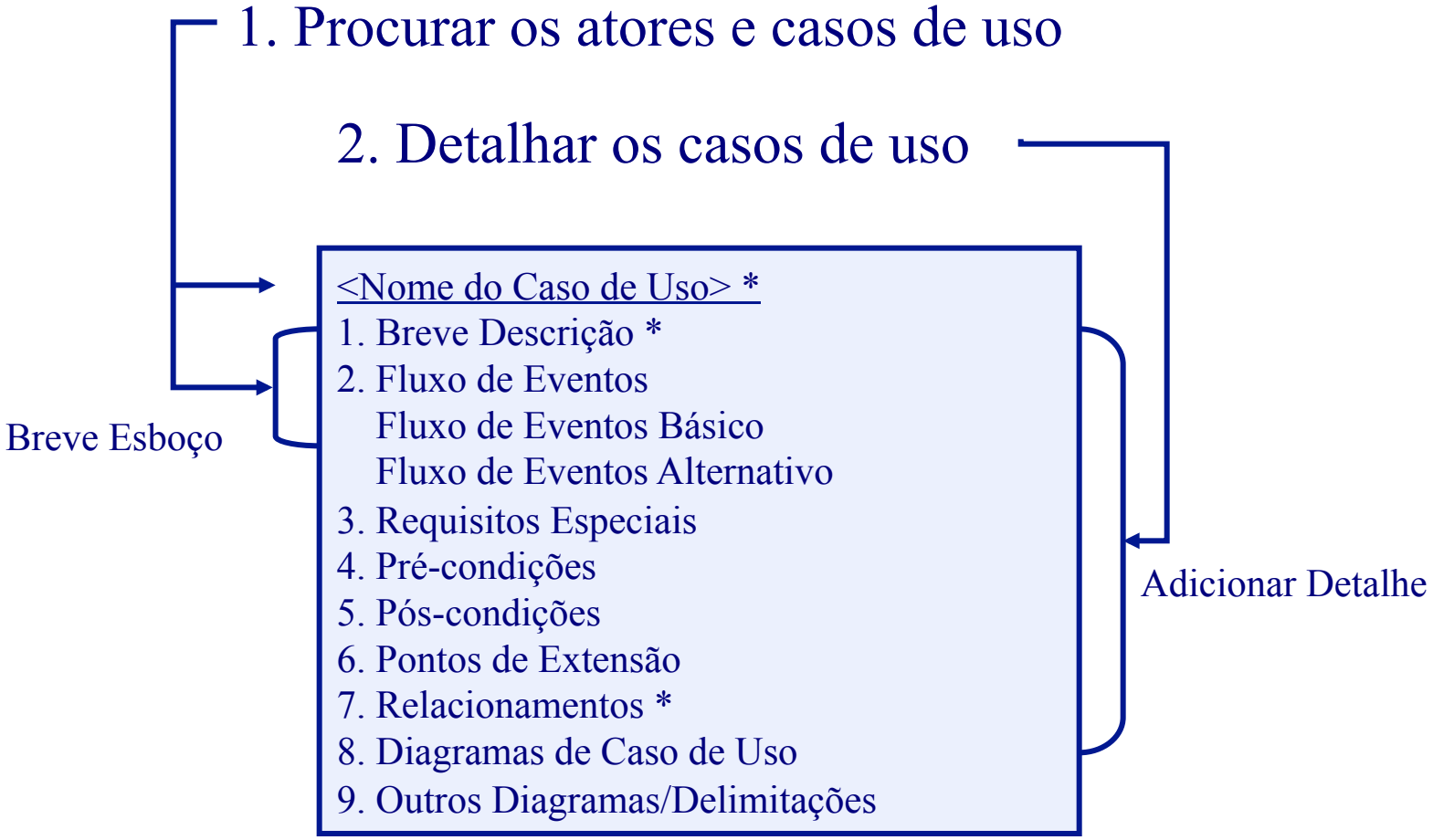


É comum o caso de uso de extensão ser abstrato, mas não é necessário que seja.

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



Detalhar um Caso de Uso



Detalhar o Fluxo de Eventos Básico

Obter Cota

1. Cliente faz logon
   O caso de uso inicia quando o Cliente do Comércio faz o logon.
   O sistema valida a identificação do cliente e senha...
2. Cliente seleciona uma das funções do "Obter Cota"
   O Cliente do Comércio escolhe o "Obter Cota". O sistema mostra a lista de garantias nas quais tem cotas.
3. Cliente Obtém Cota
   O Cliente do Comércio seleciona da lista de garantias ou entra com o símbolo do comércio para a garantia. O sistema envia o símbolo do comércio para o Sistema Cota, e recebe a Resposta do Sistema Cota.
4. Cliente Obtém Outras Cotas
   Se o Cliente do Comércio desejar adquirir cotas adicionais, o caso de uso retorna para o passo 3.
5. Cliente faz logoff
   O Cliente do Comércio faz o logoff no sistema. O caso de uso encerra.

Estruture o fluxo em passos

Número e Título em cada passo

Descrever passos (1-3 sentenças)

Fazer de cada passo uma ida e volta de eventos

Detalhar Fluxos Alternativos Gerais

Identificar Cliente

Fluxos Alternativos

A1. Cliente do Comércio indefinido

No Passo 1 do Fluxo Básico, se o sistema identifica que a identificação do cliente não é válido, o sistema mostra uma mensagem: "Desculpe, não é um cliente válido..." .
O caso de uso encerra.

A2. Senha Errada

No Passo 1 do Fluxo Básico, se o sistema identifica que a senha do cliente não é válido, o sistema mostra uma mensagem: "Desculpe, sua senha está incorreta..."

A3. Suspeita de Roubo

No Passo 1 do Fluxo Básico, se a identificação do cliente constar na de identificações roubadas, o sistema mostra uma mensagem: "Desculpe, não ...". O sistema também registra a data, hora e o endereço do computador...

A4. Sair

O Sistema RU e-st permite que o Cliente do Comércio deixe o caso de uso a qualquer hora.

A5. Sem Resposta do Usuário

A qualquer hora durante o caso de uso, se o sistema perguntar para introduzir o Cliente do Comércio...

Ocorre em qualquer lugar num outro fluxo

Detalhar Fluxos Alternativos Específico

Obter Cota

Fluxos Alternativos

A1. Cliente do Comércio indefinido

No Passo 1, Cliente faz o logon, no Fluxo Básico, se o sistema identifica que a identificação e/ou a senha do cliente não são válidos, o sistema mostra uma mensagem: "Desculpe, não é um cliente válido...". O caso de uso encerra.

A2. Sistema Cota Indisponível

No Passo 3, Cliente Obtém Cota, do Fluxo Básico, se o sistema está incapacitado para comunicação com Sistema Cota, o sistema...

A3. Sair

O Sistema RU e-st permite que o Cliente do Comércio deixe o caso de uso a qualquer hora.

A4. Suspeita de Roubo

No Passo 3, Cliente Obtém Cota, do Fluxo Básico, se o sistema não reconhecer o símbolo do comércio, o sistema notifica o Cliente do Comércio que o símbolo do comércio não foi reconhecido. O caso de uso continua no início do Passo 3...

A5. Sistema Cota não pode localizar a informação

No Passo 3, Cliente Obtém Cota, do Fluxo Básico, se o Sistema Cota que não tem a informação da solitação...

Descreve o que acontece

Início

Condição

Ações

Resumo

Pré-Condições

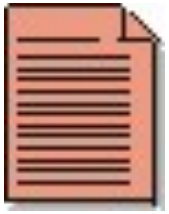
- Restrição quando pode iniciar o caso de uso
- Opcional: Usar somente se for necessário para deixar claro

Pós-Condições

- Garantir que o caso de uso encerrou
- Pode conter variações
- Opcional: Usar somente se necessário para deixar claro
  - Exemplo: Executar Comércio: Pós-condição: Ao final deste caso de uso todos os contadores devem ser atualizados para refletir as transações que ocorreram.

Visão

- Comunicação entre o gerenciamento, marketing e a equipe de projeto
- Prover um feedback inicial para o cliente
- Estimular um entendimento geral do produto
- Estabelecer escopo e prioridade para características de alto-nível

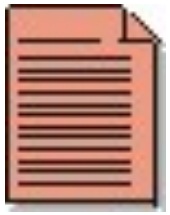


Visão

Um documento que consegue: "todas as partes trabalharem num mesmo livro"

Visão (continuação)

- Documentar o nível do sistema que descreve ”o que" e "por que" do produto ou aplicação
- Foco
  - Necessidades dos Usuários
  - Metas e Objetivos
  - Mercado alvo
  - Plataforma e ambientes do usuário
  - Características do Produto



Visão

Solicitação dos Principais Envolvidos

Este artefato contém qualquer tipo de solicitação dos principais envolvidos (cliente, usuário final, pessoal de *marketing* etc.) em relação ao sistema que será desenvolvido. Ele também pode conter referências a qualquer tipo de fonte externa com a qual o sistema deve estar de acordo.

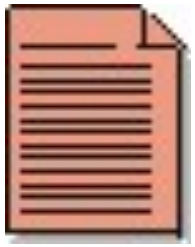
Especificações Suplementares

- Capturam os requisitos de sistema que não são capturados imediatamente nos casos de uso do modelo de casos de uso.

- Entre os requisitos estão incluídos:
  - Requisitos legais e de regulamentação e padrões de aplicativo
  - Atributos de qualidade do sistema a ser criado, incluindo requisitos de usabilidade, confiabilidade, desempenho e suportabilidade
  - Outros requisitos, como sistemas operacionais e ambientes, requisitos de compatibilidade e restrições de projeto

Glossário

- Define termos importantes usados no projeto
- Ajuda prevenir más interpretações



Glossário

- Adquirir vocabulário comum
- Iniciar o mais breve possível
- Continuar ao longo do projeto

O Glossário é desenvolvido primeiramente durante as fases de **iniciação** e de **elaboração**, pois é importante decidir uma terminologia comum logo no início do projeto.