Rational Unified Process

Introdução

Fernando Pedrosa – fpedrosa@gmail.com

Bibliografia

- Phillip Kruchten – Rational Unified Process Made Easy. Addison Wesley
- www.ibm.com (RMC)
- www.wthreex.com/rup

Des. de SW – Principais Problemas

Segundo Kruchten:

- Necessidades do usuário mal compreendidas
- Falta de habilidade para tratar mudanças de requisitos
- Descoberta tardia de problemas sérios
- Baixa qualidade de software
- Problemas com papéis e responsabilidades

Rational Unified Process (RUP)

- Criado por Booch, Jacobson e Rumbaugh, e implementado pela Rational
- Em seu livro, os amigos se referem a ele como Unified Process. RUP é o nome comercial dado pela Rational
- Em 2003 a IBM compra a Rational. RUP continua sendo, até hoje, o principal *framework* de processos no qual as metodologias se baseiam

O que é ?

- É uma plataforma de processos
  - Adaptável
  - Deve ser configurada para selecionar os elementos apropriados às necessidades da organização
- Fornece atividades, artefatos e guias ligados
  - Às ferramentas IBM/Rational
  - À linguagem UML

Características

- Iterativo e Incremental
  - O ciclo de vida do produto é dividido em iterações, cada uma entregando incrementos (partes acabadas) do software
- Guiado por casos de uso
  - Os casos de uso conectam todas as fases e visões, sendo utilizados por todos os stakeholders
- Centrado na arquitetura
  - Envolve aspectos estáticos e dinâmicos
  - Evolui a partir das necessidades do produto

Características

- Orientado a Objetos
  - Componentes são construídos através de Objetos e estes colaboram entre si para realizar os casos de uso
- Planejado por riscos
  - Os riscos são analisados continuamente e os de maior criticidade são tratados prioritariamente

Exercícios [1]

(CAIXA – CESPE 2010)

[45] Não se utilizam diagramas de caso de uso em projetos desenvolvidos de acordo com o RUP (rational unified process).

(CEHAP – CESPE 2009)

[27A] O RUP foi projetado em conjunto com a UML e os processos de negócios são modelados usando casos de uso que, posteriormente, serão desenvolvidos para modelar os requisitos de sistema.

(TCU – CESPE 2010)

[109] O processo unificado de software é centrado na arquitetura e orientado por casos de uso, o que sugere um fluxo de processo iterativo e incremental.

Exercícios [1]

(PETROBRAS – CESGRANRIO 2008)

[48] Um princípio fundamental do Processo Unificado é

(A) ser centrado em arquitetura.

(B) empregar times auto-dirigidos e auto-organizados.

(C) o desenvolvimento em cascata.

(D) a programação em pares.

(E) a propriedade coletiva do código fonte.

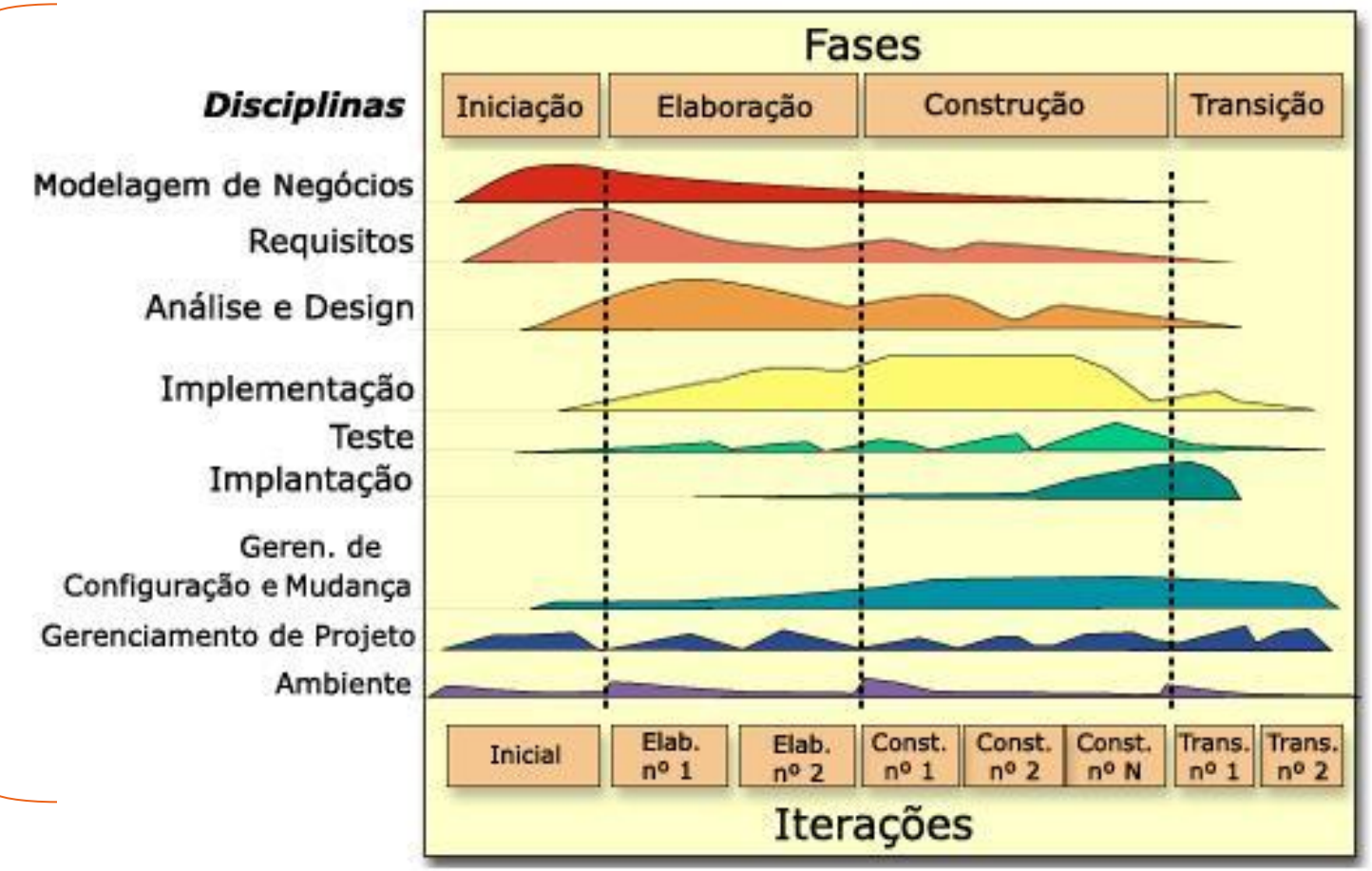
(BASA – CESPE 2010)

[80] A metodologia RUP, que consiste no desenvolvimento interativo com foco na redução dos riscos do projeto, agrega um valor real à organização que necessita manter padrões relativos às comunicações externas e à comunicação com a equipe de desenvolvimento.

Gráfico das Baleias

Eixo dinâmico

Eixo estático



Dimensões

O RUP tem duas dimensões

- A primeira dimensão representa o aspecto dinâmico do processo
  - Eixo horizontal
  - Expresso em termos de fases, marcos e iterações
- A segunda dimensão representa o aspecto estático do processo
  - Eixo vertical
  - Expresso em termos de componentes, disciplinas, atividades, artefatos, papéis…

Exercícios [2]

(SECONT/ES – CESPE 2010)

[76] O processo unificado é estruturado em duas dimensões. A dimensão horizontal representa o aspecto dinâmico do processo, onde estão representadas suas fases, às quais estão associados marcos que determinam sua finalização. Na outra dimensão estão representadas as disciplinas, que agrupam logicamente as atividades. É possível haver disciplina que não esteja presente em todas as fases.

(EMBASA – CESPE 2009)

[70] A primeira dimensão do RUP representa o aspecto dinâmico do processo quando ele é aprovado e é expressa em termos de fases, iterações e marcos.

Metodologia baseada no RUP

O que compõe a Metodologia?

- Processo de Desenvolvimento
- Conjunto de métodos e práticas bem definidas
  - Com responsáveis
  - Entradas/Saídas
  - Ordem de precedência
- Inclui:
  - Ferramentas, Tecnologias, Pessoas, Padrões e guias

Quem? O quê? Como? Quando?

Componentes da Metodologia

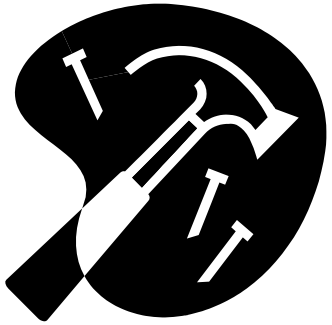
Modelos, Padrões
e Guias

Equipes Treinadas

Metodologia

Linguagem Padrão

Ferramentas



Benefícios da Metodologia

- Qualidade de software
- Maior produtividade
- Maior previsibilidade
- Maior controle sobre custos e prazos

Conceitos Chave do RUP

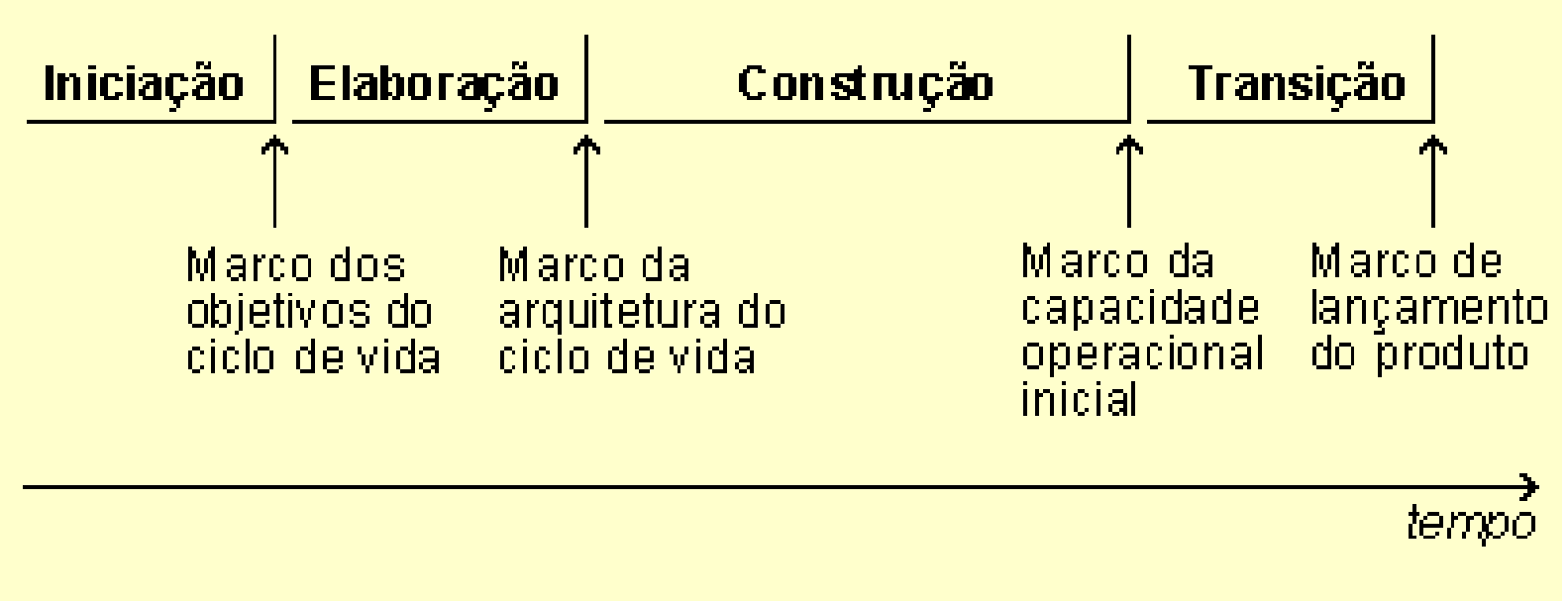
- Fases e Iterações
- Disciplinas/Fluxo de Atividades
- Atividades/Tarefas
- Artefatos/Produtos de Trabalho
- Papéis

Fases

Concepção	Elaboração	Construção	Transição
Estabelecer o escopo, e estimar custos e riscos	Assegurar que os principais riscos foram diminuídos e definir uma arquitetura executável	Desenvolver de modo iterativo e incremental um produto completo para a Transição	Disponibilizar o Software para seus usuários finais

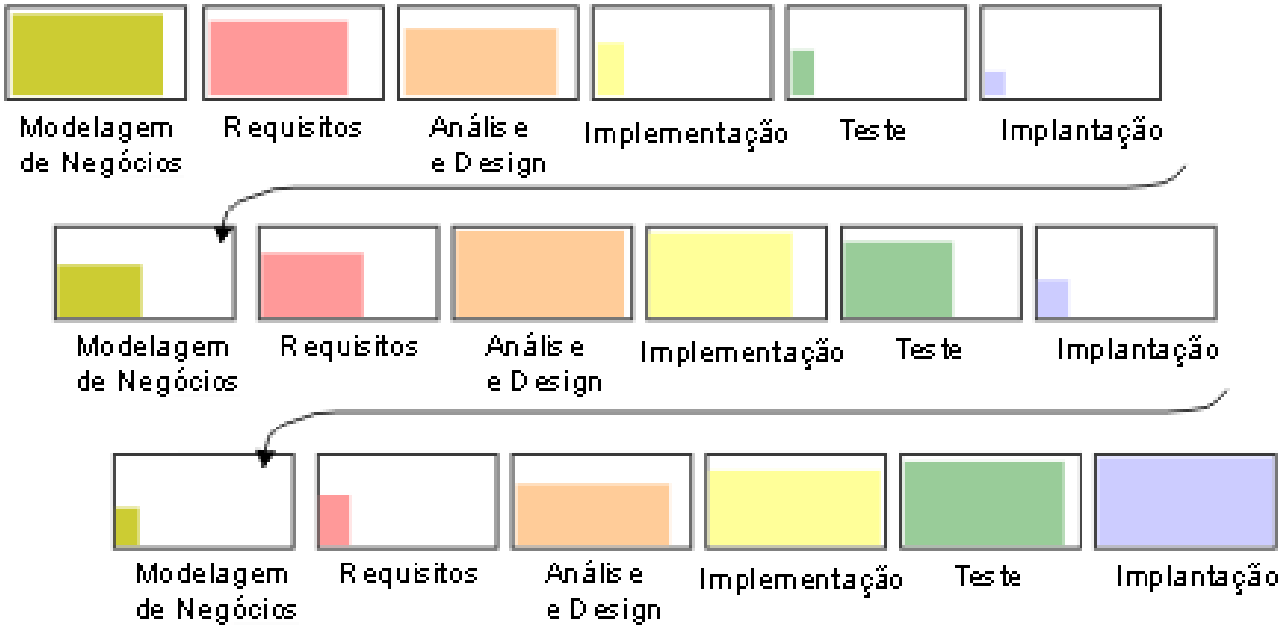
Fases

Cada fase termina com um Marco



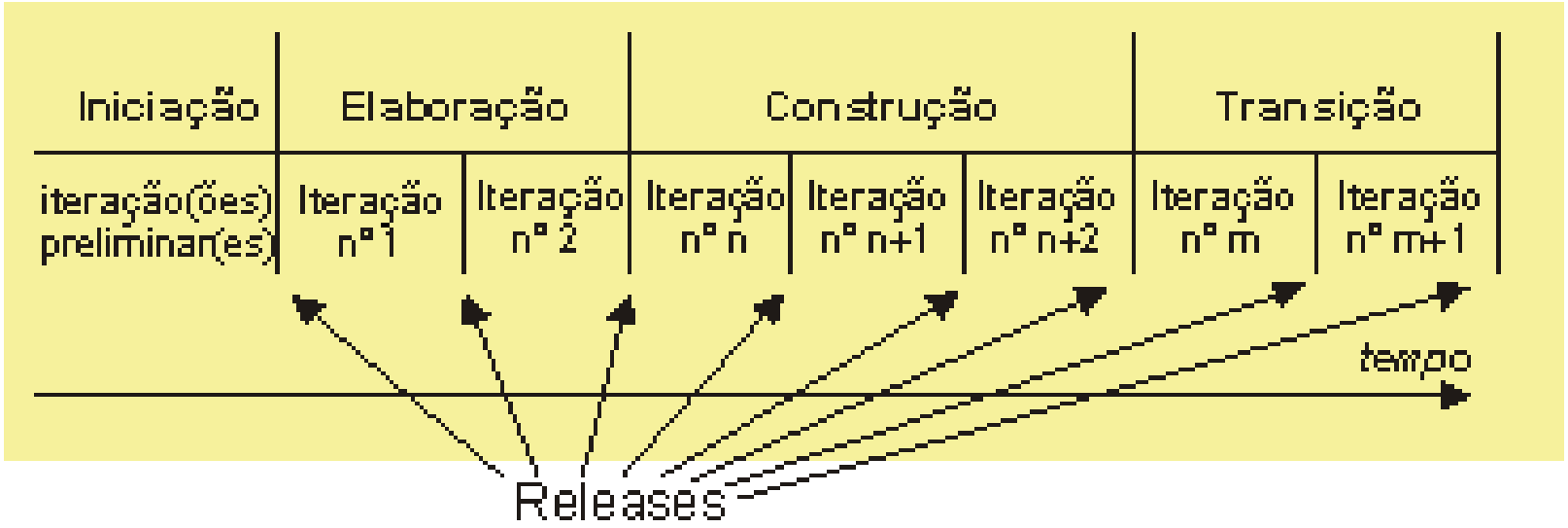
Iterações

Cada passagem pela sequência de disciplinas do projeto se chama **iteração**



Fases e Iterações

Cada fase pode ser dividida em iterações



Exercícios [3]

(SERPRO – CESPE 2010)

[72] O framework de processo RUP (rational unified process) organiza o ciclo de vida de um produto de software desde o início de sua concepção até a sua aposentadoria, na seguinte sequência de etapas: concepção, elaboração, construção e transição

(CEHAP – CESPE 2009)

[27 C] Ao contrário do modelo em cascata, no qual as fases coincidem com as atividades do processo, o RUP compreende as fases de concepção, elaboração, construção e transição.

Exercícios [3]

(MPE/RR – CESPE 2008)

[80] No Processo Unificado, a vida de um sistema é dividida em ciclos; cada ciclo, por sua vez, é dividido em fases e, entre as fases, tem-se a fase Construção, na qual as atividades visam capturar requisitos ainda não capturados na fase anterior e produzir uma arquitetura executável, a ser usada na fase Elaboração.

[81] O Processo Unificado é iterativo e incremental. Ao final de cada iteração, a qual é um miniprojeto, os modelos que representam o sistema encontram-se em um determinado estado, denominado baseline. As atividades de cada fase de um ciclo de vida podem ser distribuídas entre várias iterações.

Exercícios [3]

(ANATEL – CESPE 2006)

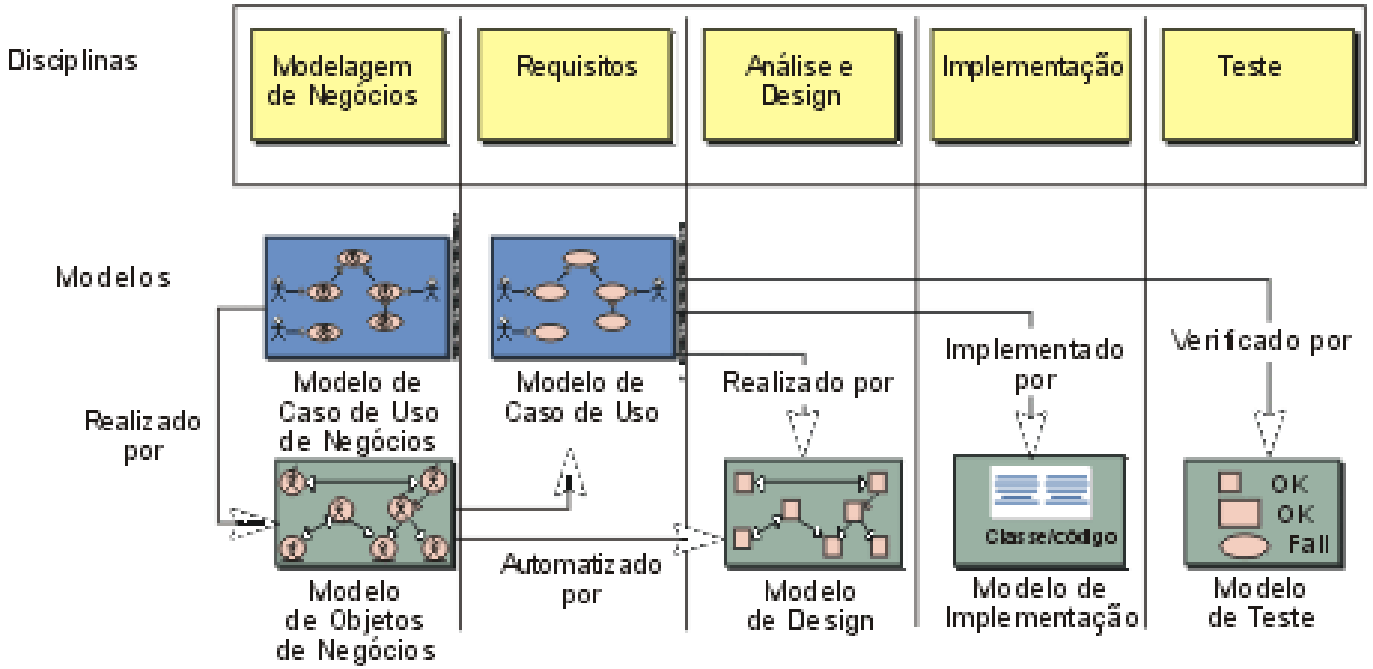
[86] O ciclo apresentado na figura, que compreende uma execução seqüenciada das atividades de modelagem de negócios, requisitos, análise e desenho, implementação, testes, avaliação etc., forma o denominado ciclo de vida de software no modelo RUP.

Disciplinas

- São um conjunto de atividades (fluxo de trabalho) relacionadas a uma “área de interesse” do projeto
- Ajudam a compreender o projeto a partir de uma perspectiva em cascata

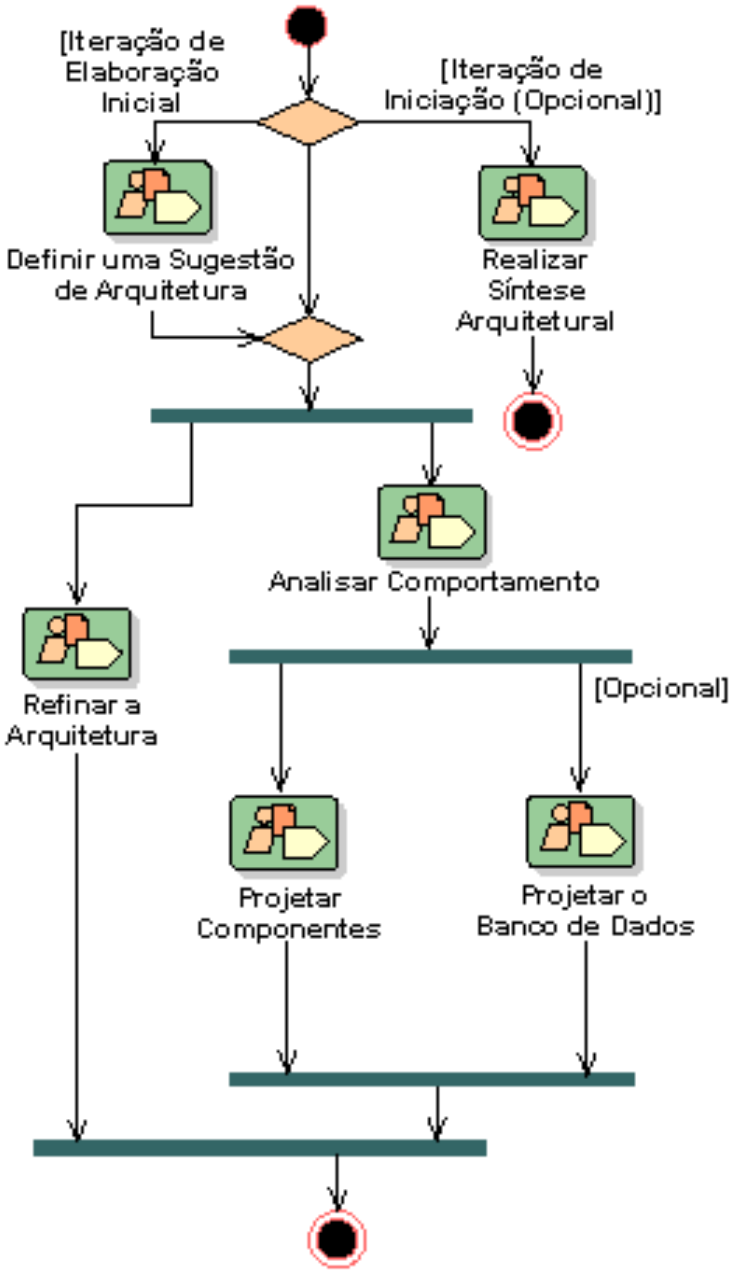
Disciplinas

Algumas disciplinas estão associadas a conjuntos específicos de modelos



Disciplinas

Cada disciplina possui um fluxo de trabalho (ex: Análise e Design)



Disciplinas

- Disciplinas básicas
  - **M**odelagem de Negócios
  - **R**equisitos
  - **A**nálise e projeto
  - **I**mplementação
  - **T**estes
  - **I**mplantação

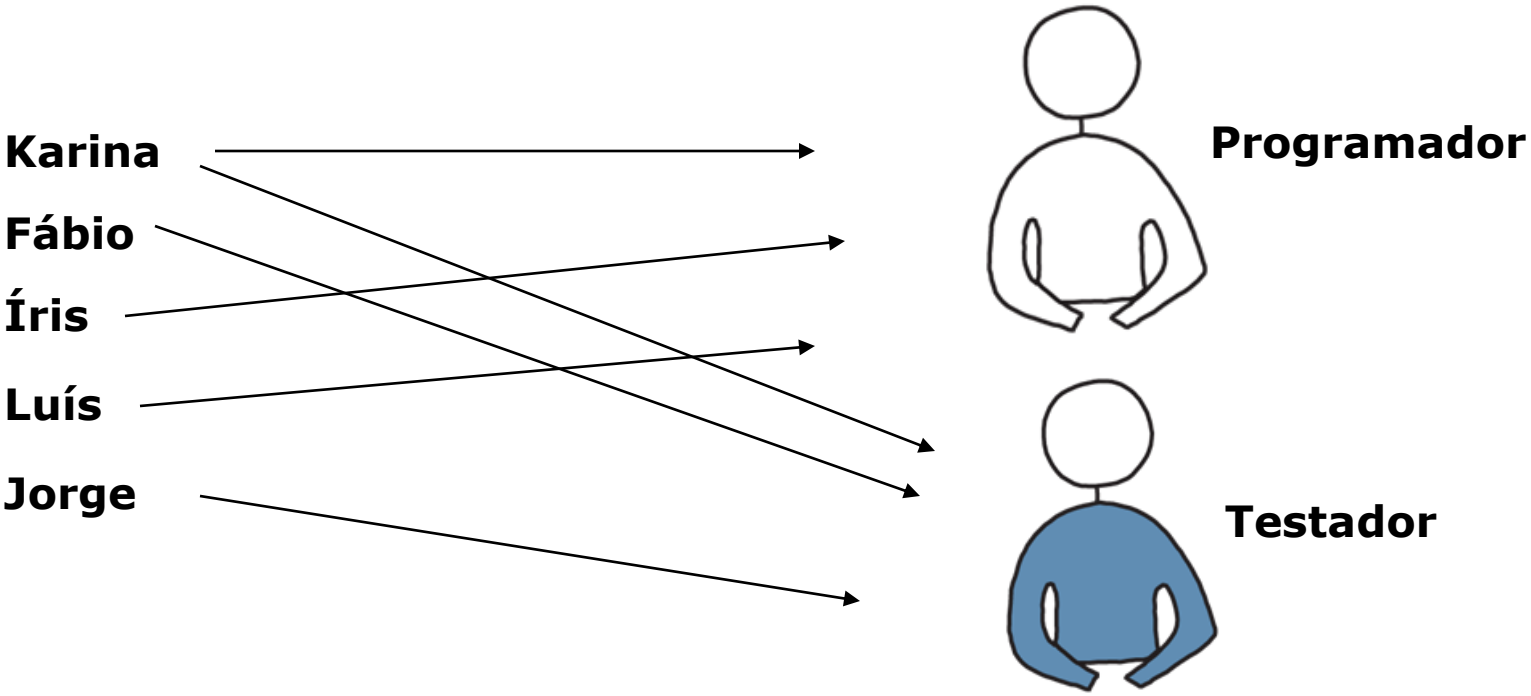
Disciplinas de suporte

- **G**erenciamento de Projeto
- **G**erenc. de configuração e mudanças
- **A**mbiente

MRAITIGGA

Papeis

Definem o comportamento e as responsabilidades no processo



Não representam pessoas!

Atividades/Tarefas

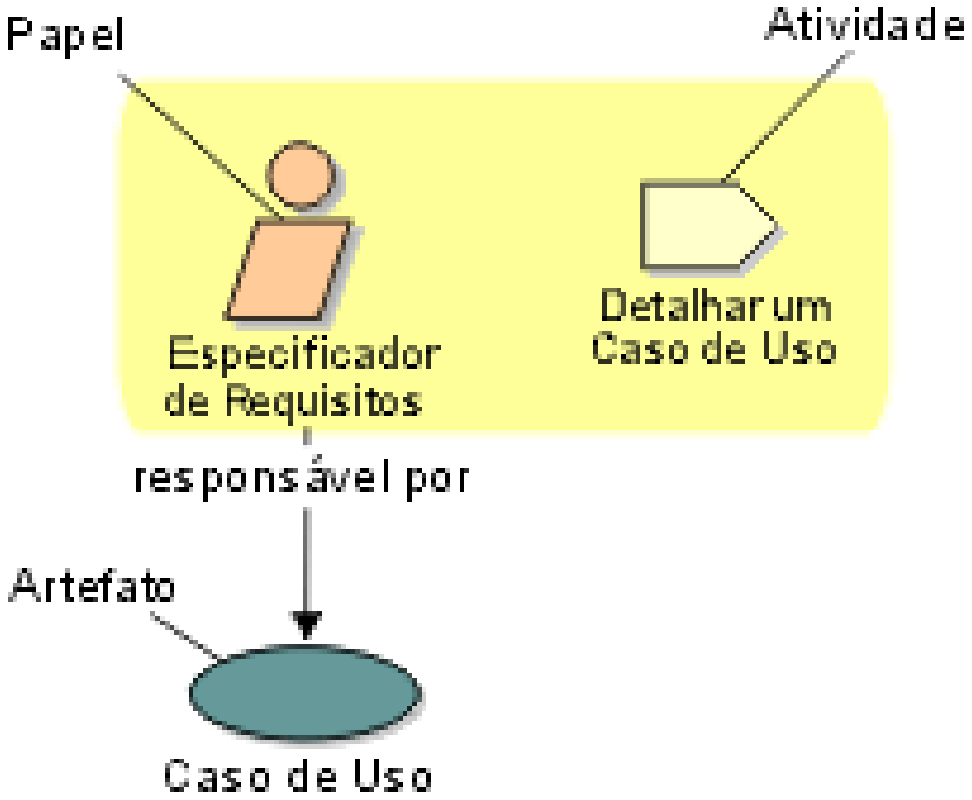
- Unidade de trabalho desempenhada por um papel
- Inseridas no contexto de uma Disciplina
- Compostas de:
  - Finalidade
  - Passos
  - Entradas e saídas
  - Papel responsável
  - Guias e padrões

Artefatos/Produtos de Trabalho

- São o resultado de um processo de trabalho
- Utilizados como entradas e/ou saídas na execução das atividades
- Podem ser:
  - Modelos
  - Documentos
  - Código fonte
  - Executáveis, etc…

Em resumo...

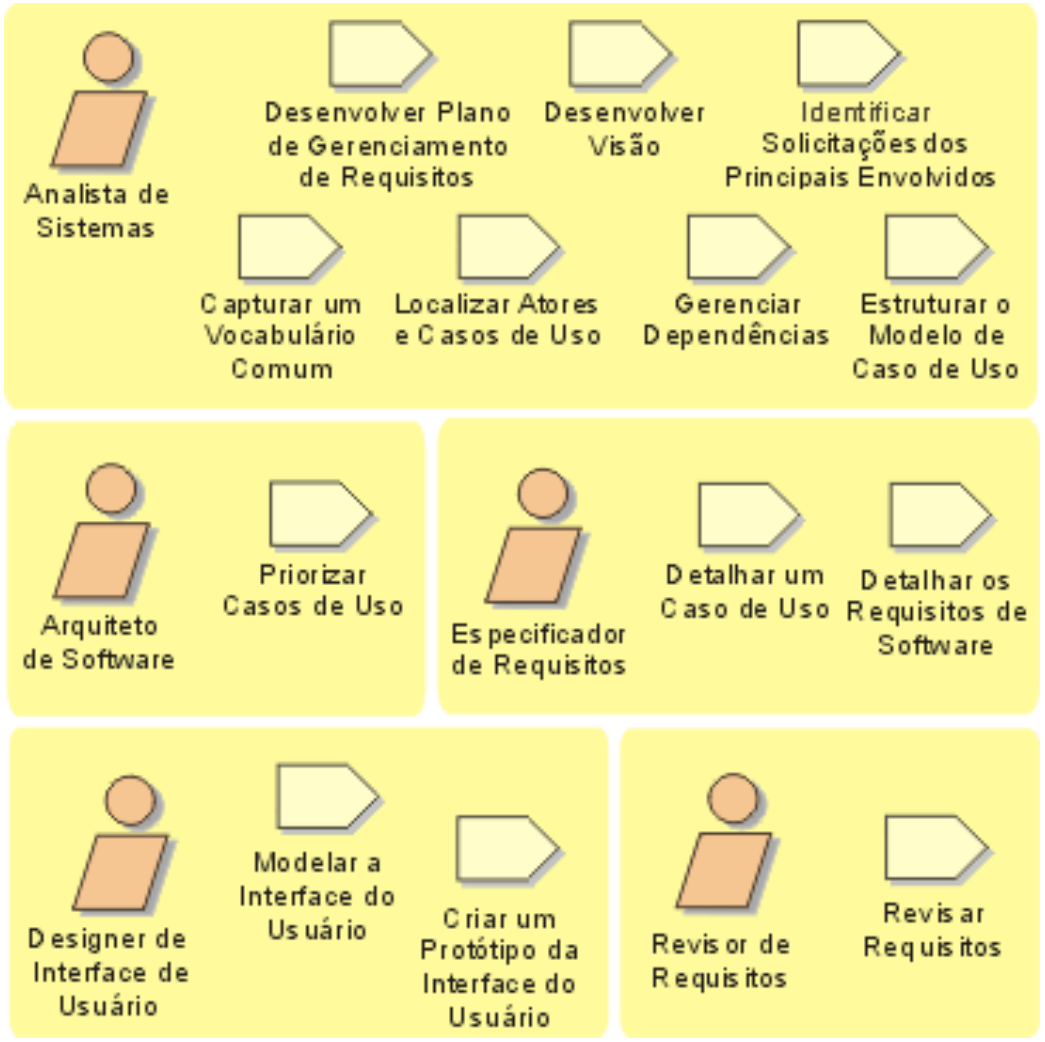
- Papéis executam Atividades que geram Artefatos



Visão Geral das Atividades

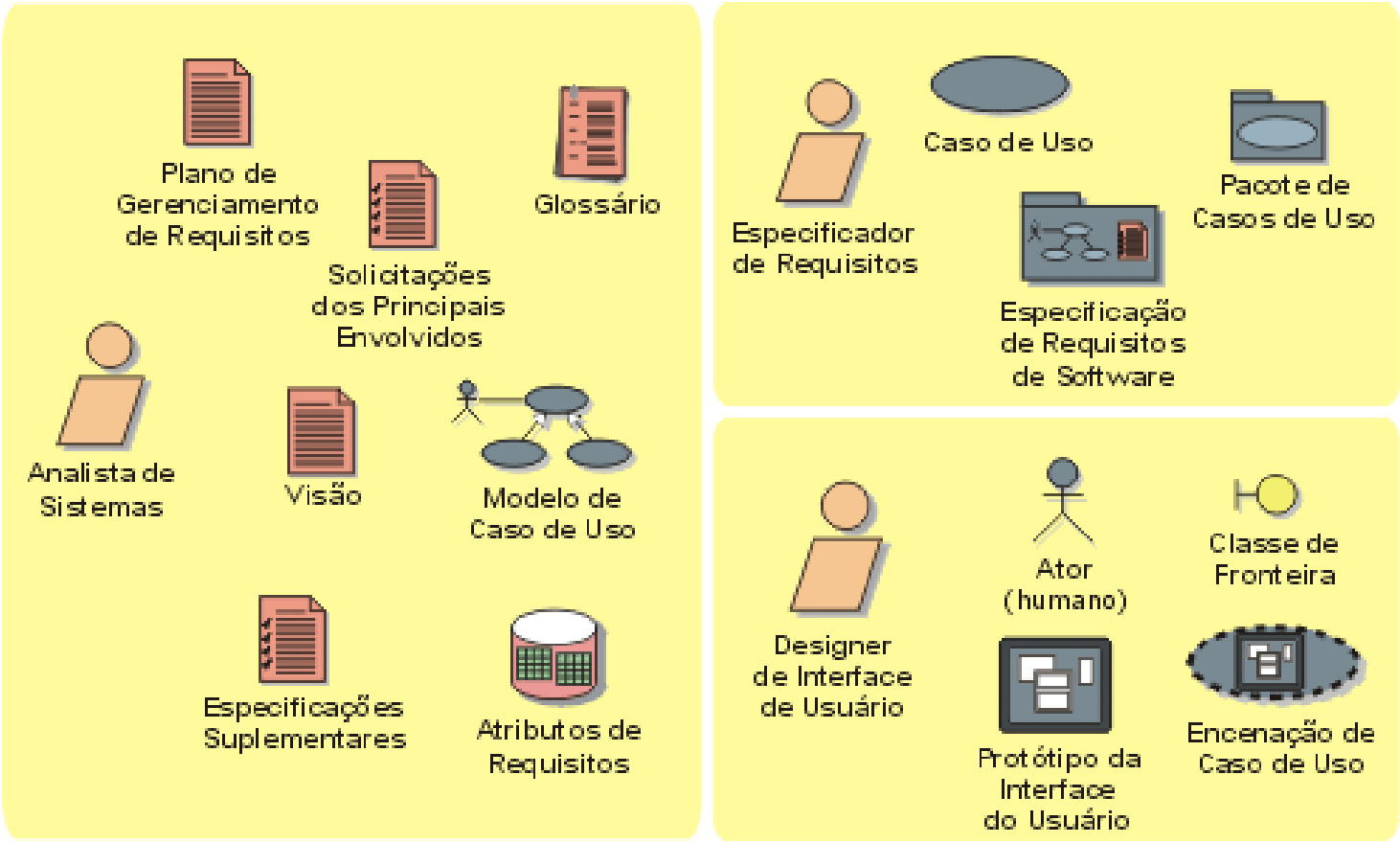
Cada disciplina tem uma visão geral de **atividades** executadas

Ex:
Disciplina de Requisitos



Visão Geral de Artefatos

Cada disciplina tem uma visão geral dos **artefatos** relacionados (ex: Requisitos)



Exercícios [4]

(MPE/RR – CESPE 2008)

[79] No Processo Unificado, atividades são organizadas em fluxos de atividades. Algumas atividades produzem artefatos, que podem ser de engenharia ou gerenciais. Entre os artefatos criados, há modelos que visam especificar o sistema a partir de certos pontos de vista e níveis de abstração.

(TRE/MT – CESPE 2010)

[32-C] As disciplinas de suporte (apoio) do RUP são: gerenciamento de classes; gerenciamento de produto; e ambiente.

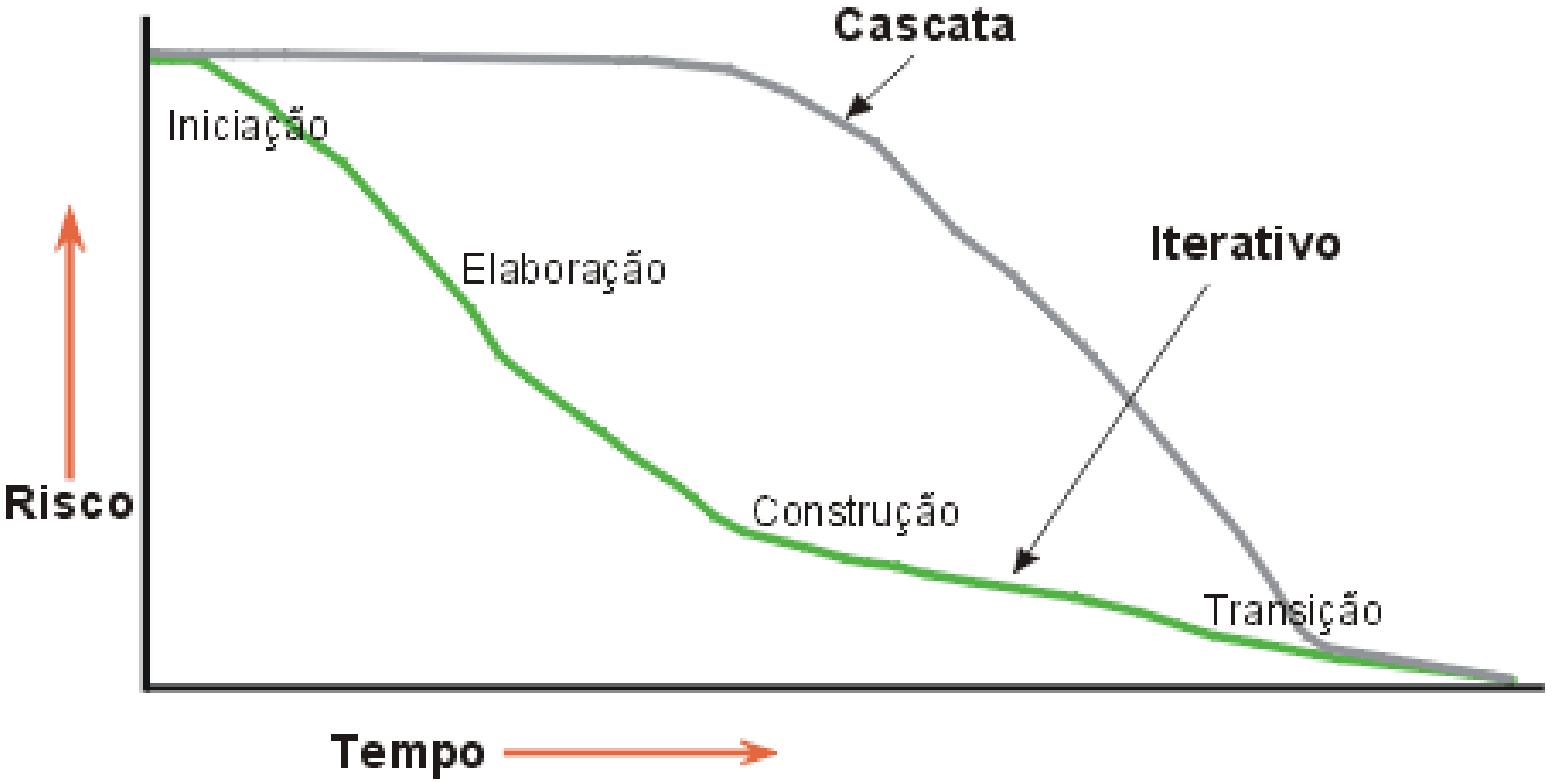
Exercícios [4]

[32-D] Um papel é uma definição abstrata de um conjunto de atividades executadas e dos respectivos artefatos. Exemplos de papéis no RUP são: analistas, desenvolvedores e testadores. Explicitamente, papéis de gerentes não fazem parte dos papéis possíveis no RUP.

[32-E] As disciplinas de engenharia do RUP são: modelagem de negócios; requisitos; análise e projeto; implementação; teste; qualidade; e implantação.

Melhores Práticas: Desenvolvimento Iterativo

Por que desenvolver iterativamente?



Características

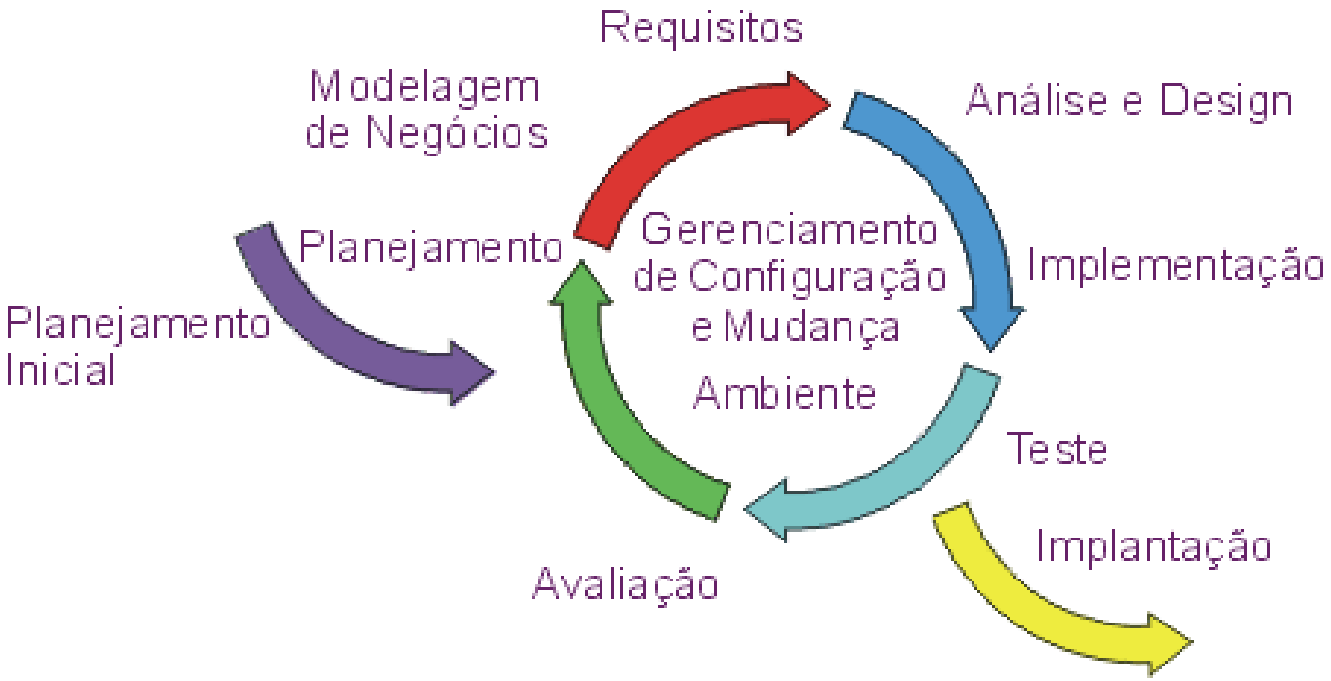
- Desenvolvimento Iterativo lida com mudanças
  - As táticas e os requisitos variáveis são acomodados
- Diminui riscos
  - Os riscos são reduzidos mais cedo, pois os elementos são integrados progressivamente
- Busca melhor qualidade
  - A melhoria e o refinamento do produto são facilitados, tornando-o mais robusto

Características

- Aprende e melhora
  - As organizações podem aprender a partir dessa abordagem e melhorar seus processos
- Aumenta o reuso
  - Identificar partes comuns quando estão parcialmente projetadas ou implementadas é mais fácil que identificar todas as semelhanças no início

Iterações

- Incorporam um conjunto **quase** seqüencial de atividades:

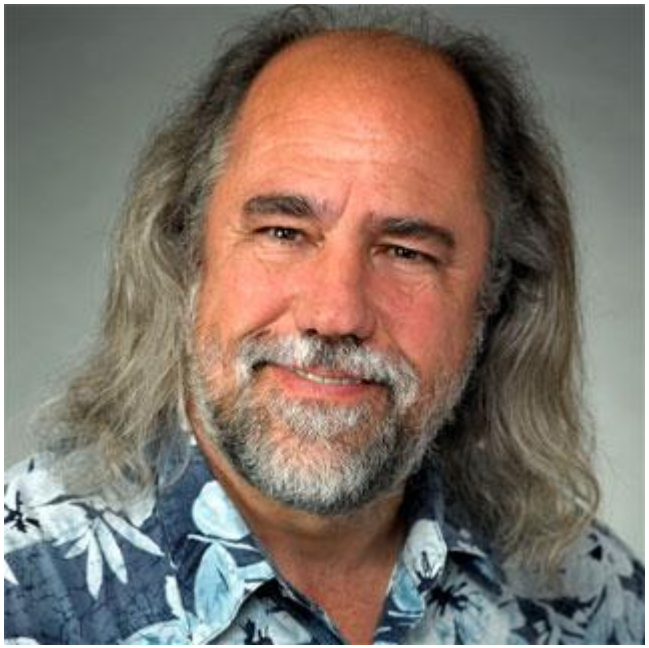


Desenvolvimento Iterativo

"Com a abordagem em cascata, tudo parece bem até quase no final do projeto; às vezes, até a metade da integração. Aí, tudo desmorona. Com a abordagem iterativa, é muito difícil esconder a verdade durante muito tempo" – cliente da Rational

Em resumo, esta é a ideia:

"Jogue fora um pouco do trabalho durante o caminho, para não ter que jogar todo o trabalho fora ao final." – Grady Booch



Exercícios [5]

(TRE/MT – CESPE 2010)

[44-A] Uma das principais características do RUP é o uso da iteração, que, por meio de refinamentos sucessivos, melhora o entendimento do problema.

[44-C] Pelo fato de o RUP ser muito complexo, seu foco evita a redução dos riscos do projeto. Essa fase é tratada diretamente na UML.

Melhores Práticas: Gerência de Requisitos

Gerenciamento de Requisitos

- Um **requisito** é uma condição ou uma restrição com a qual o sistema deverá estar em conformidade
- O gerenciamento de requisitos é uma abordagem sistemática para:
  - Localizar,
  - Documentar,
  - Organizar e Controlar os requisitos variáveis em um sistema.

Problemas com Requisitos

- Nem sempre os requisitos são óbvios e podem vir de várias fontes.
- Requisitos relacionam-se entre si, mas deve haver consistência nos relacionamentos
- Cada requisito é diferente, eles não são igualmente importantes ou fáceis de entender

Problemas com Requisitos

- Há várias partes interessadas
- O número de requisitos pode se tornar impossível de gerenciar se eles não forem controlados
- **Os requisitos são alterados**

Como gerenciar as dificuldades?

- **Analise o problema**
  - Entenda o “problema por trás do problema”
  - Estabeleça um vocabulário comum
  - Proponha soluções em alto nível

Como gerenciar as dificuldades?

- **Entenda a necessidade das partes interessadas**
  - Todos querem algo. Determine qual é a melhor “fonte”
  - Utilize técnicas de elicitação de requisitos
  - Faça acordos, balanceie prioridades

Como gerenciar as dificuldades?

- **Defina o sistema**
  - Defina o que sistema deve fazer, em termos gerais, utilizando linguagem natural e gráfica
- **Gerencie o escopo do sistema**
  - O que está dentro? O que está fora?
  - Estabeleça prioridades!

Como gerenciar as dificuldades?

- **Gerencie requisitos variáveis**
  - Garanta que os requisitos tenham uma estrutura que os tornem facilmente atualizáveis
  - Rastreie os requisitos

Casos de Uso guiam o desenvolvimento

- O RUP recomenda a utilização de Casos de Uso como método para a organização dos requisitos funcionais
- Em vez de fazer uma lista de requisitos, organize-os na visão de como o usuário poderá utilizar o sistema
  - Utilize **Casos de Uso**!

Casos de Uso

- Um caso de uso define um conjunto de cenários
- Cada cenário descreve o comportamento do sistema em termos de sequências de ações
- Um caso de uso deve produzir um resultado de valor observável para um **ator**
  - Atores são as entidades que interagem com o sistema

Quem utiliza Casos de Uso?

- Clientes
  - Para entenderem o comportamento do sistema e aprovar o fluxo de eventos
- Arquitetos de Software
  - Para identificar características da arquitetura
- Analistas, Projetistas, Desenvolvedores
  - Para entender o comportamento do sistema e refiná-lo

Quem utiliza Casos de Uso?

- Testadores
  - Utilizam os casos de uso como base para gerar casos de teste
- Gerentes
  - Para planejar e acompanhar o progresso do projeto
- E outras partes interessadas...
- Por isso se diz que o RUP é **guiado por Casos de Uso**

Exercícios [6]

(EMBASA – CESPE 2010)

[72] No RUP, os manuais dos sistemas e as rotinas de teste são definidos a partir dos casos de uso. Entretanto, os elementos da arquitetura e a estratégia de implantação do sistema, por se relacionarem com a infraestrutura e não com os requisitos funcionais, não são definidos com base nos casos de uso.

Melhores Práticas: Arquitetura de Componentes

Arquitetura

Segundo a IEEE:

- “O conceito de mais alto nível de um sistema em seu ambiente”
- A arquitetura de um sistema é a sua organização ou estrutura de componentes significativos que interagem através de interfaces

Arquitetura

Arquitetura de Software inclui:

- As decisões significativas sobre a organização de um sistema
- A seleção de elementos estruturadores e suas interfaces
- A especificação do comportamento dos elementos do sistema e como eles colaboram entre si

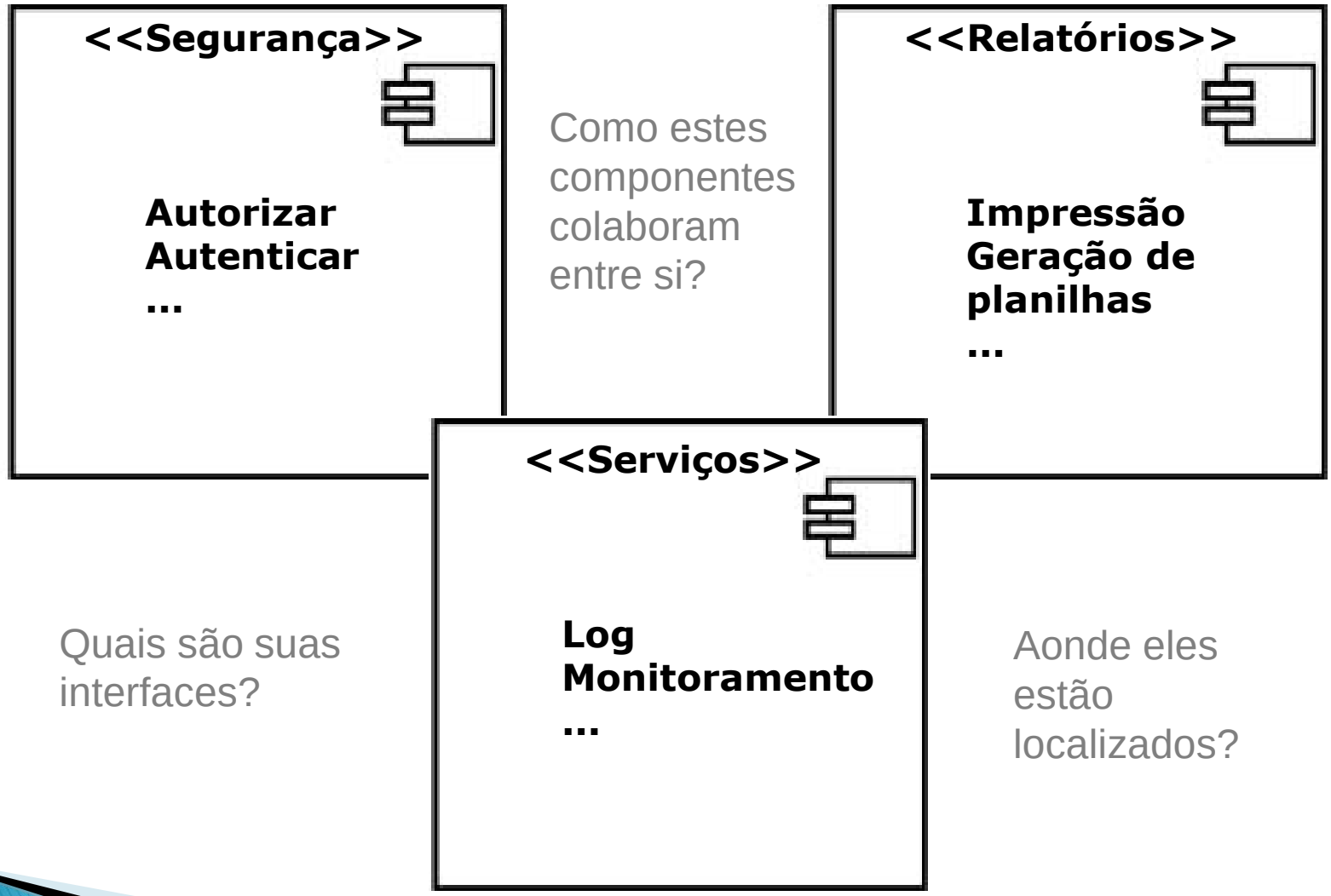
Arquitetura

- Não se preocupa apenas com estrutura e comportamento, mas também com:
  - Funcionalidade
  - Desempenho
  - Segurança
  - Reuso
  - Manutenibilidade
  - Decisões tecnológicas e econômicas, ...

Componentes

- **O que são?**
  - Grupos coesos de código fonte ou executável com interfaces e comportamentos bem definidos
  - Fornecem forte encapsulamento de conteúdo
  - Substituíveis
  - Exemplos: módulos, pacotes, subsistemas, componentes OTS

Componentes



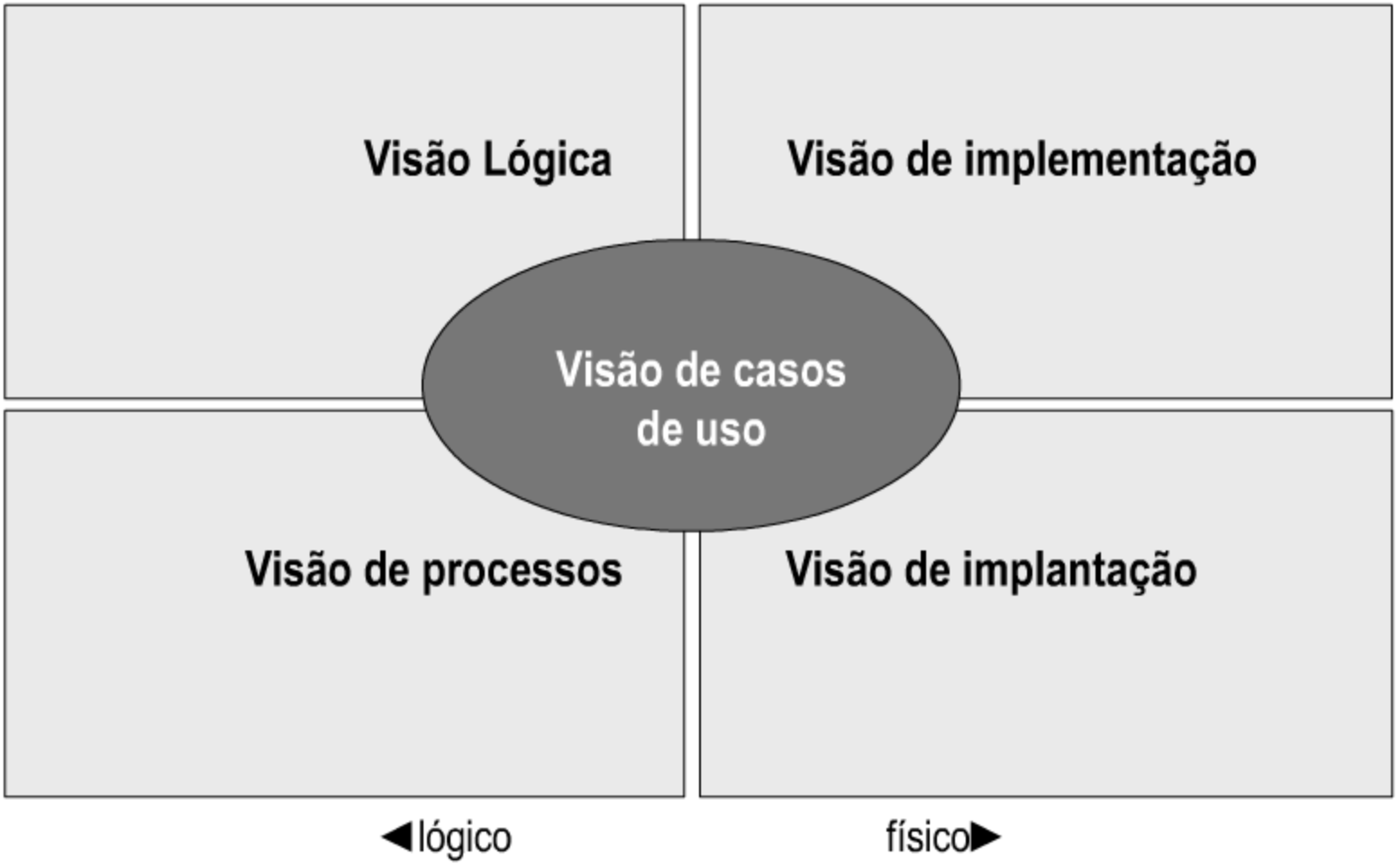
Vantagens de uma Arquitetura baseada em Componentes

- Permite reuso em grande escala
- Permite gerenciar a complexidade do projeto e manter a integridade do sistema
- Identifica, isola, projeta, desenvolve e testa “pedaços” bem formados do sistema
- Possibilita usar componentes de prateleira (*off the shelf)*

Visões Arquiteturais

- No RUP a arquitetura é representada por uma série de visões de arquitetura diferentes
- Em sua essência, as Visões são fragmentos que ilustram os elementos "significativos em termos de arquitetura"
- É conhecido como o **modelo de visão 4+1**

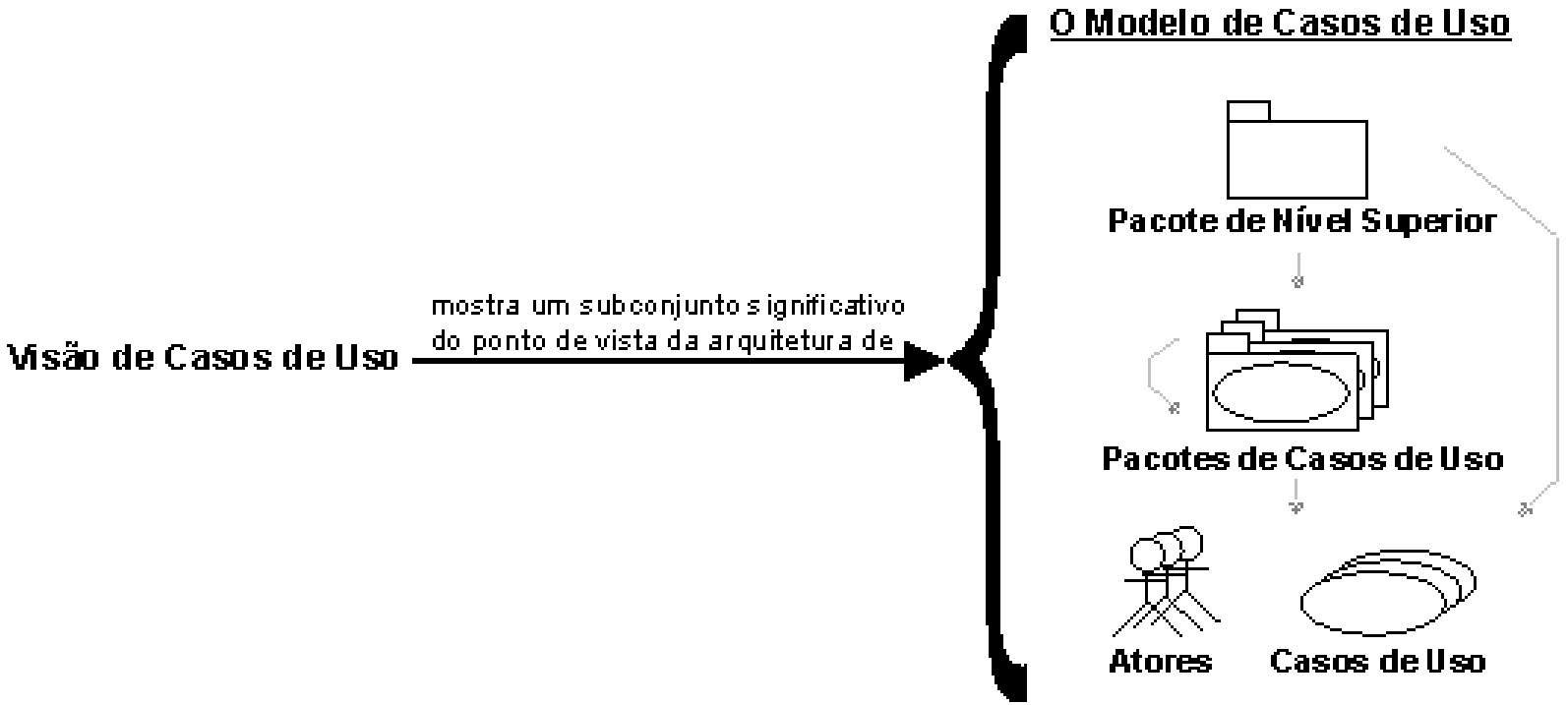
4+1 Visões


Visão Lógica
Visão de implementação
Visão de casos de uso
Visão de processos
Visão de implantação
◀lógico
físico▶

Visão de Casos de Uso

- Contém Casos de Uso e cenários que abrangem comportamentos significativos em termos de arquitetura, classes ou riscos técnicos
- É uma visão **obrigatória** do documento de arquitetura de software

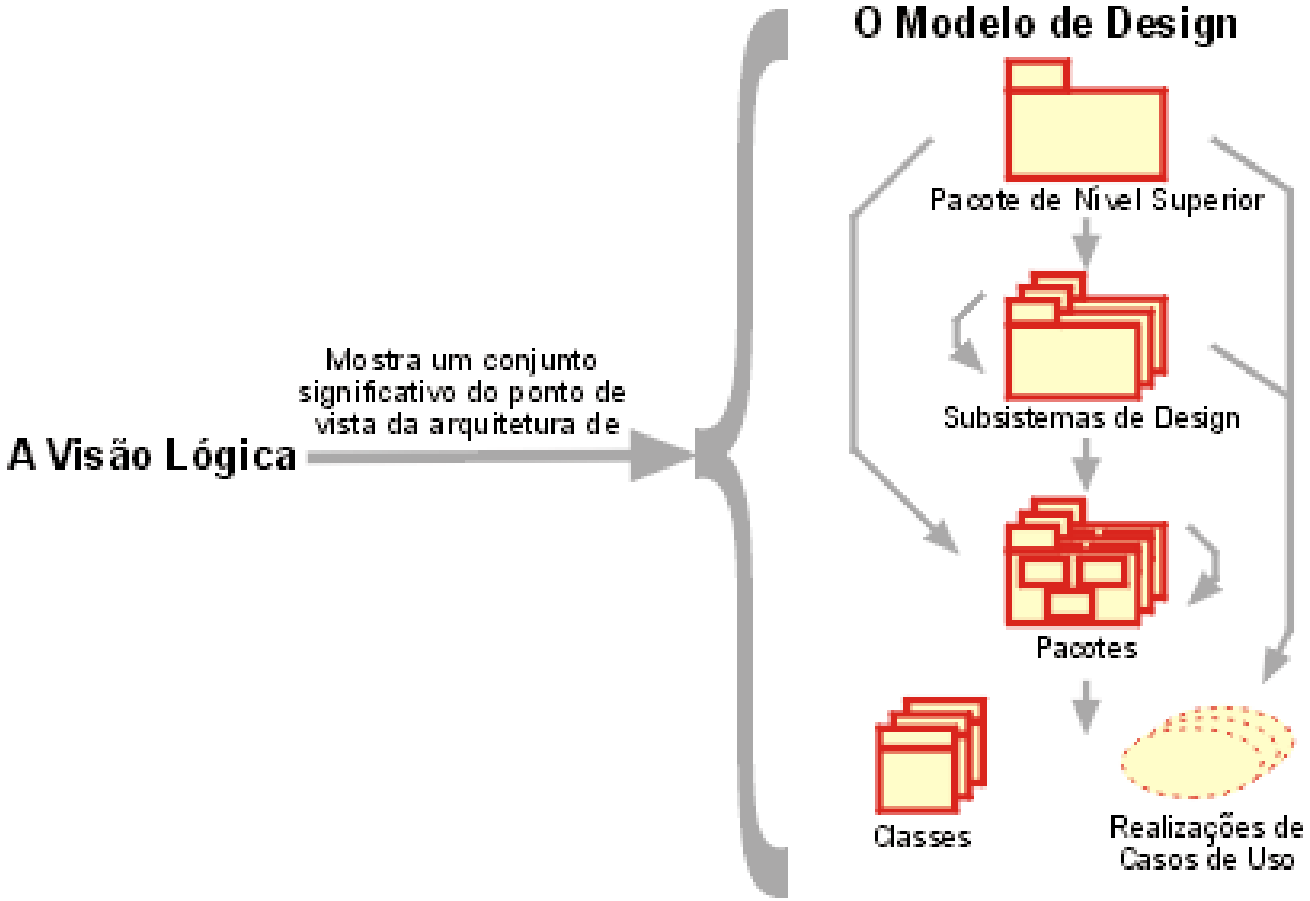
Visão de Casos de Uso



Visão Lógica

- Contém as classes de projeto mais importantes e sua organização em pacotes e subsistemas, e a organização desses pacotes em camadas
- É uma visão **obrigatória** do documento de arquitetura de software

Visão Lógica



Visão de Implementação

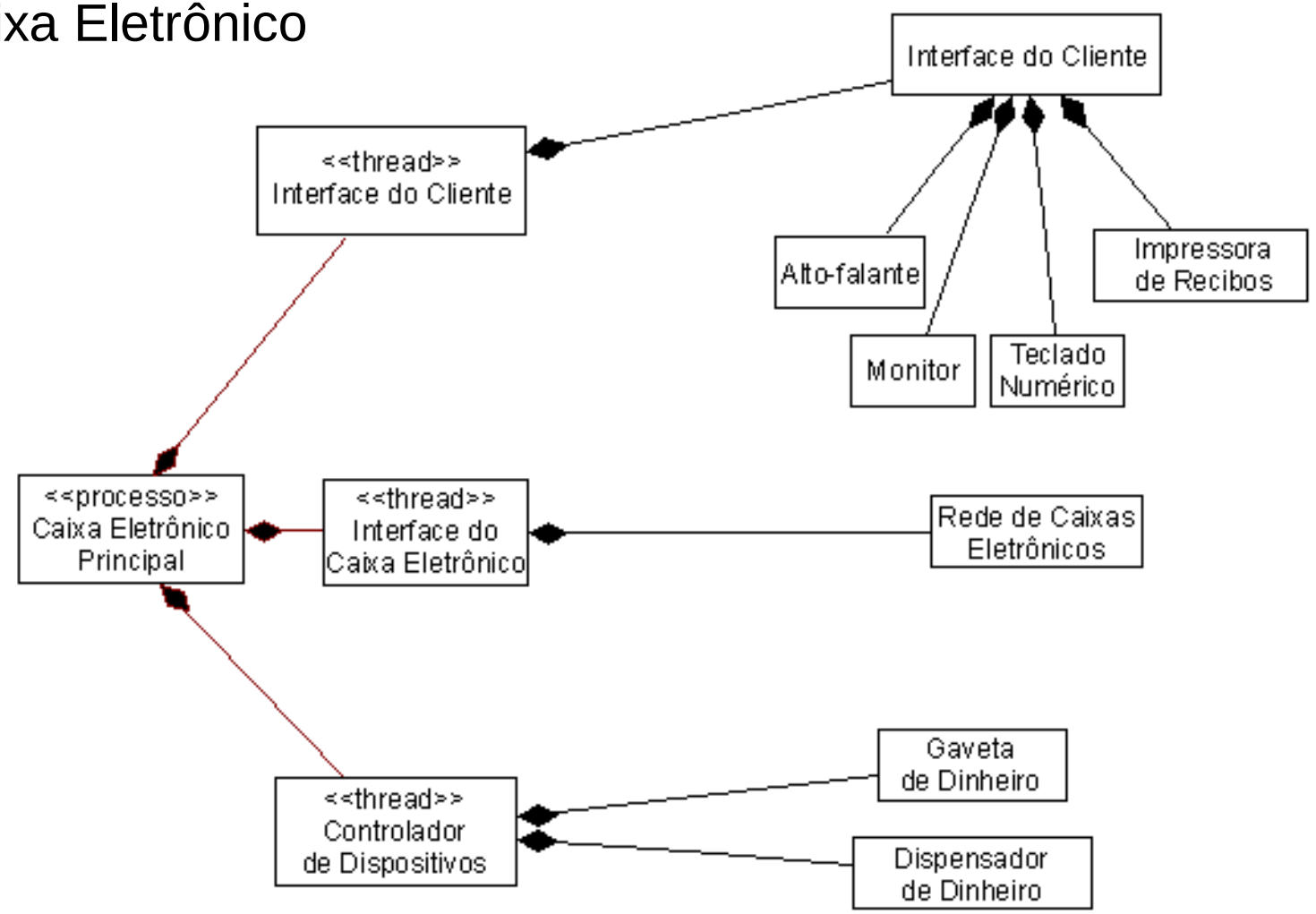
- Contém uma visão geral do Modelo de Implementação e sua organização em termos de módulos em pacotes e camadas
- Detalha os pacotes e módulos da Visão Lógica (detalhes “físicos”)
- É uma visão **opcional** do documento de arquitetura de software
  - Deve ser usada apenas se a implementação não for derivada diretamente do modelo de projeto (isto é, há detalhes adicionais)

Visão de Processos

- Contém a descrição das tarefas (processos e threads) envolvidas, suas interações e configurações e a alocação dos objetos e classes de projeto em tarefas
- É uma visão **opcional** do documento de arquitetura de software
  - Só precisa ser usada se o sistema tiver alto grau de paralelismo

Visão de Processos

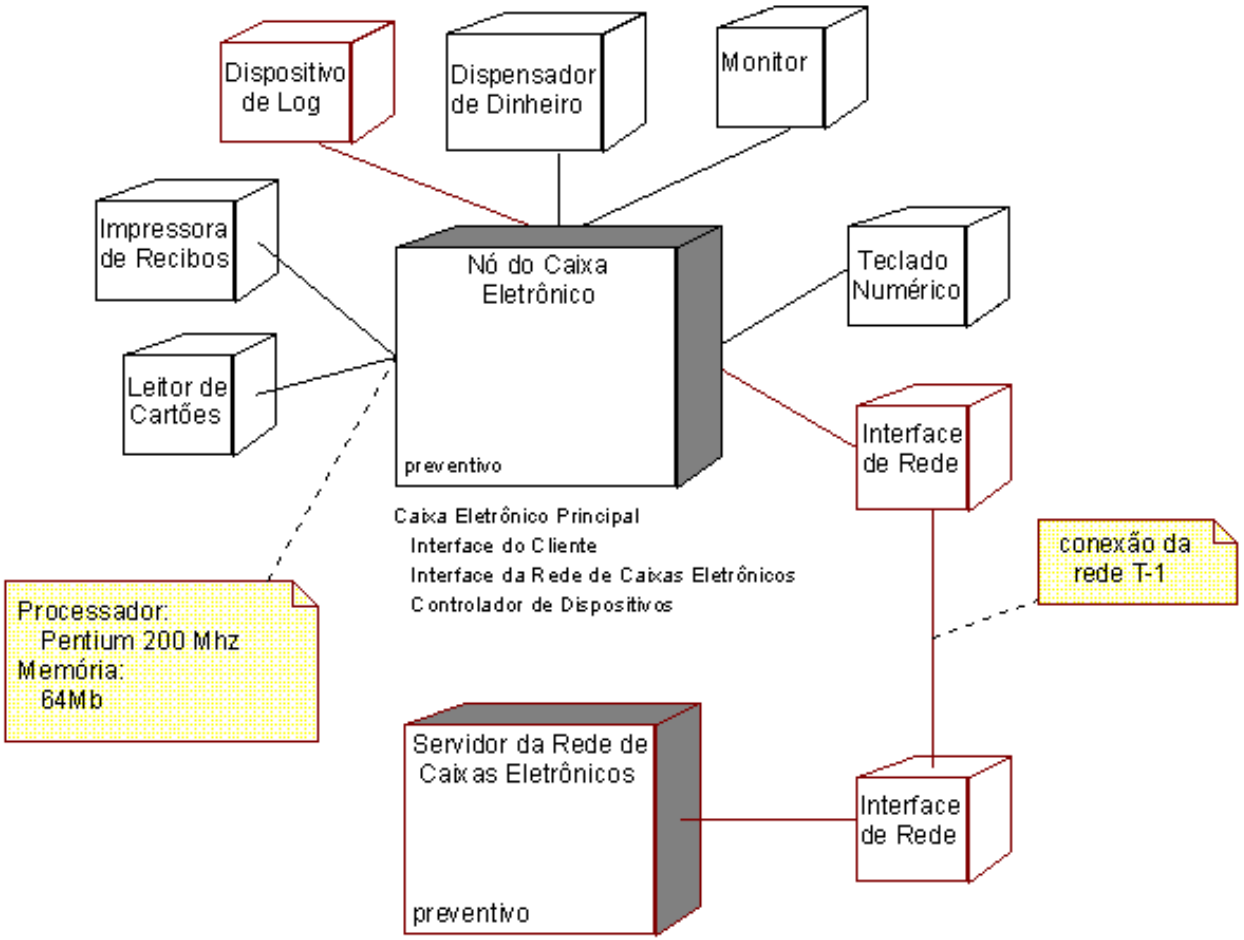
Caixa Eletrônico



Visão de Implantação

- Contém a descrição dos vários nós físicos do sistema e a alocação de tarefas atribuídas a eles
- É uma visão **opcional** do documento de arquitetura de software
  - Só precisará ser usada se o sistema estiver distribuído em vários nós físicos

Visão de Implantação



Exercícios [7]

(TRE/BA – CESPE 2010)

[62] Na engenharia de software baseada em componentes, na qual se supõe que partes do sistema já existam, o processo de desenvolvimento concentra-se mais na integração dessas partes que no seu desenvolvimento a partir do início. Essa abordagem é baseada em reúso para o desenvolvimento de sistemas de software.

(SAD/PE – CESPE 2010)

[35-D] É desejável que o valor da coesão e o do acoplamento, duas importantes propriedades da arquitetura de um software, sejam maximizados durante a engenharia de software.

Melhores Práticas: Modelagem Visual (UML)

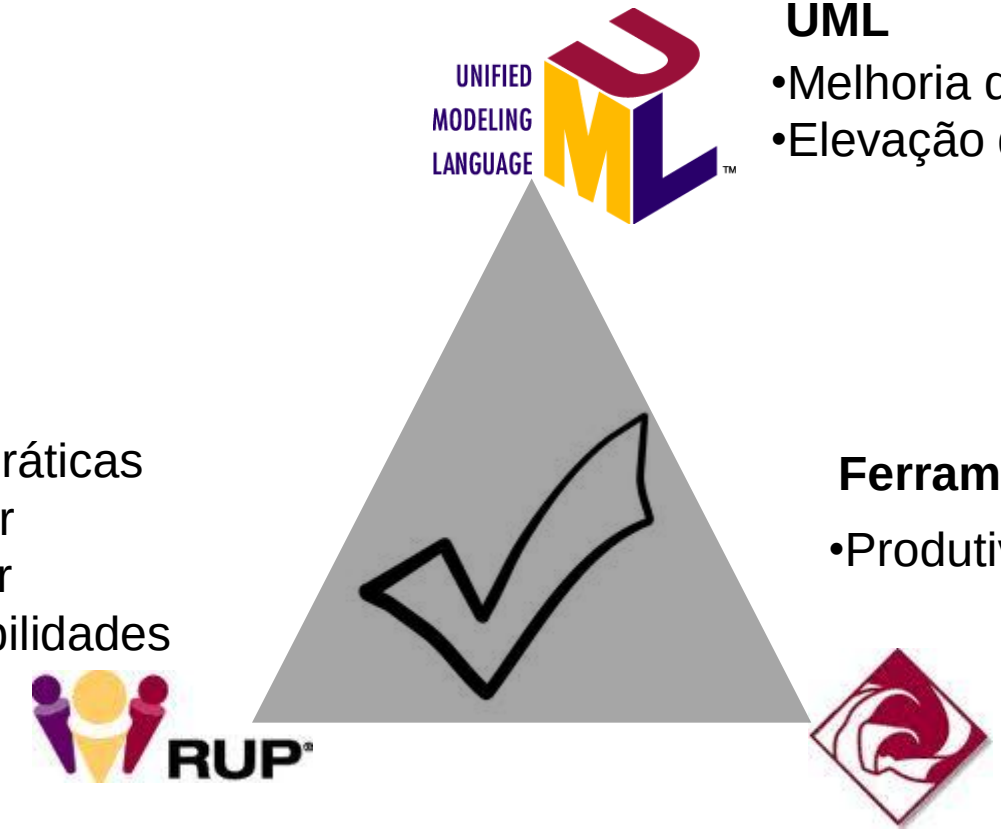
O que é modelar visualmente?

- Fazer uso de notação de design gráfica e visual para capturar o projeto do software
- Permitir que o nível de abstração seja aumentado
- Capturar requisitos com precisão
- Melhorar a comunicação da equipe (acabando com a ambiguidade)

RUP preconiza o uso de UML

- **UML – Unified Modeling Language**
  - Notação padrão para modelagem de Software
  - Oferece múltiplas perspectivas de um problema
  - Mantém projeto e implementação consistentes

UML x Processo x Ferramenta



UML

- Melhoria da comunicação
- Elevação da abstração

Processo

- Melhores práticas
- O que fazer
- Como fazer
- Responsabilidades

Ferramenta

- Produtividade

Exercícios [8]

(TCU – CESPE 2010)

[108] UML (unified modeling language) é uma tecnologia concorrente com o processo unificado, no que diz respeito ao apoio à prática de engenharia de software orientada a objetos.

(TJ/CE – CESPE 2008)

[90] O modelo de ciclo de vida prescrito pela metodologia RUP é iterativo, incremental, direcionado por riscos, adota as áreas de processos de gerência de processos prescritas pelo modelo CMMI e é baseado em modelagem visual com UML e ferramentas CASE.

Melhores Práticas: Verificação da qualidade

Qualidade

- Para as finalidades do RUP, 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*"

Controle x Garantia da Qualidade

- Controle da Qualidade
  - Tem foco no **produto**, e em encontrar defeitos específicos
  - Preza pelos **resultados** do seu trabalho
- Garantia da Qualidade
  - Tem foco nos **processos** e como eles estão sendo executados
  - Garante que você está fazendo as coisas de maneira correta

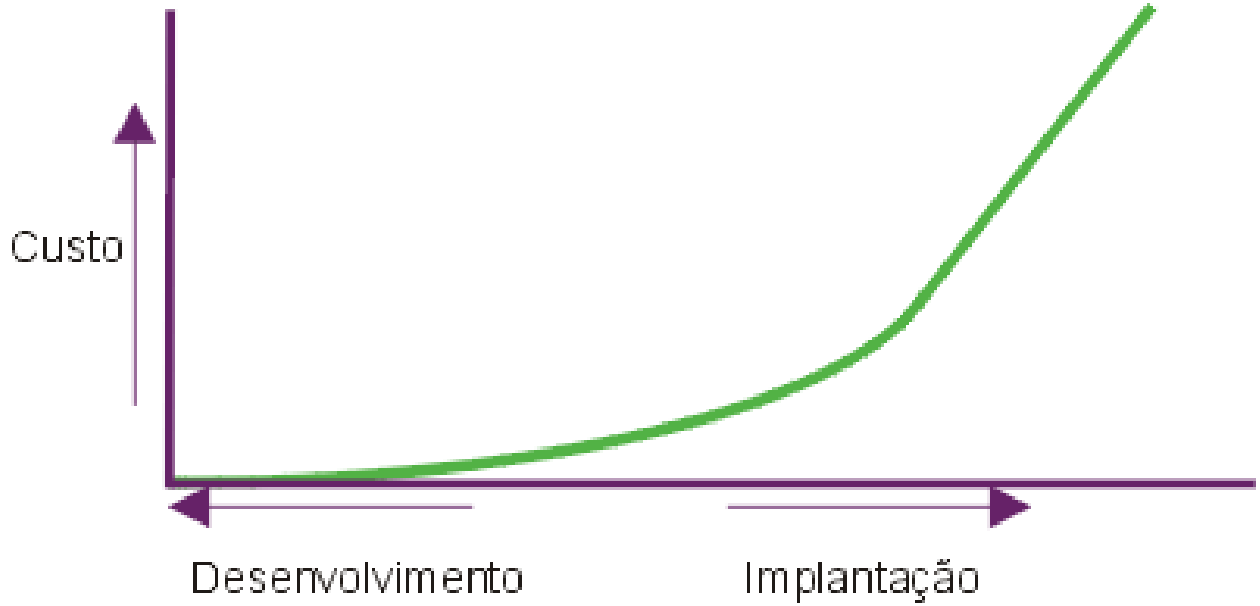
Dimensões da Qualidade

Qualidade é multidimensional

- Andamento: progresso do projeto
- Variação: diferença entre planejado e executado
- Confiabilidade: robustez
- Funcionalidade: casos de uso implementados
- Desempenho: tempo de execução em diversas situações reais

Mas afinal de contas: por que gerenciar qualidade?

- O custo sobe exponencialmente...
- Verificar a qualidade durante o ciclo de vida é **essencial**!



Exercícios [9]

(TRE/BA – CESPE 2010)

[53] Uma falha comum em projetos de sistemas computacionais é não assegurar a qualidade do software. Normalmente, essa questão é discutida após o término dos projetos, ou a qualidade fica sob a responsabilidade de equipe diferente da equipe de desenvolvimento. O RUP, proposto pela IBM, é um processo que provê uma solução disciplinada sobre como assinalar tarefas e responsabilidades dentro de uma organização de desenvolvimento de software, porém, não auxilia no controle do planejamento e verificação da qualidade.

Melhores Práticas: Gerenciamento de Mudanças

O ambiente do projeto...

- Vários desenvolvedores
- Diferentes equipes
- Diferentes locais
- Várias iterações, releases, produtos, plataformas

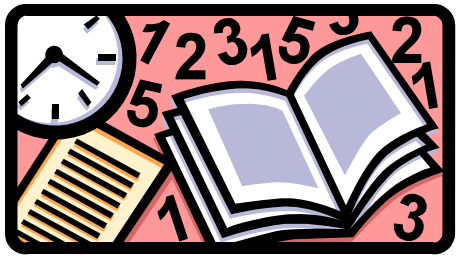
Na ausência de controle disciplinado, o processo de desenvolvimento rapidamente se transforma no caos!

Item de Configuração

- Uma entidade que satisfaz algum propósito para o usuário final e que pode ser unicamente identificada
- Podem ser
  - Arquivos-fonte, Executáveis, DLLs, etc.
  - Planos, especificações, modelos, etc.
  - Casos de teste, manuais, documentação de apoio, etc.
- Estão sob a Gestão da Configuração

Gerenciamento de Mudanças

- Gerência de Mudanças é o processo de avaliar, coordenar e decidir sobre a realização de mudanças propostas a itens de configuração (IC's)
- Apenas mudanças aprovadas são implementadas nos IC's e nos dados e documentos relacionados



Gerenciamento de Mudanças

No RUP, abrange as atividades de:

- Coordenação de atividades e artefatos
  - Procedimentos repetíveis para gerenciar mudanças sobre os artefatos
- Coordenação de iterações e releases
  - Mantém o controle sobre os releases ao final de cada iteração (baselines)
- Controle de mudanças no software
  - Mantém uma estrutura bem definida para gerenciar mudanças no software

Exercícios [10]

(TRE/MT – CESPE 2010)

[32-B] O RUP promove o uso de seis melhores práticas: desenvolva iterativamente; gerencie requisitos; use arquiteturas de componentes; modele visualmente (UML); verifique qualidade de software continuamente; e gerencie mudanças.

Exercícios [10]

(BNDES – CESGRANRIO 2009)

[51] A gerência de desenvolvimento de sistemas de uma empresa está reformulando seu processo de software. Para isso, deseja criar uma metodologia de desenvolvimento baseada no Processo Unificado. A respeito desse processo, é INCORRETO afirmar que o(a)

(A) desenvolvimento é iterativo, incremental e orientado por casos de uso.

(B) caso de uso mais crítico deve ser atacado, preferencialmente, no final.

(C) fase de transição envolve treinamento de usuários e assistência no uso do produto.

Exercícios [10]

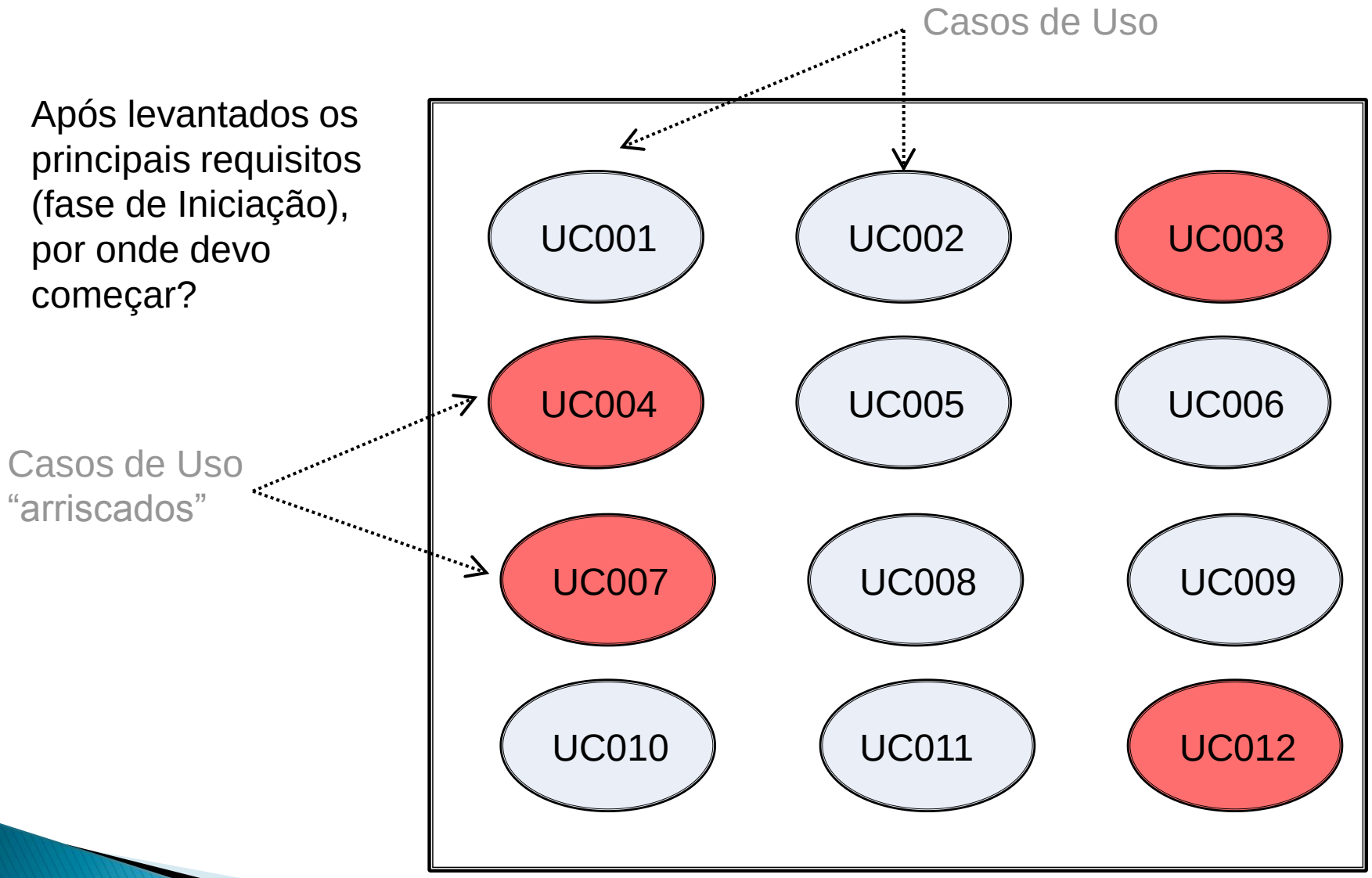
(D) arquitetura se desenvolve a partir das visões do usuário expressas em casos de uso.

(E) arquitetura, na fase de construção, é estável, ainda que possa ser evoluída.

Estudo de caso

O sistema...

Após levantados os principais requisitos (fase de Iniciação), por onde devo começar?



Priorizando...

É feita uma matriz, que estabelece uma relação entre a importância dos requisitos e o quão "arriscados" eles são

	Risco	Importância
UC001	3	7
UC002	4	7
UC003	8	9
UC004	9	8
UC005	3	5
UC006	3	8
UC007	10	10
UC008	5	5
UC009	2	7
UC010	1	
UC011	1	7
UC012	8	10

Planejando segundo o RUP

- Segundo o RUP, as funcionalidades mais "arriscadas" e importantes devem ser implementadas primeiro
  - Elas devem ser "estabilizadas" já nas iterações da fase de Elaboração
- Ao final da fase de Iniciação, já existe o planejamento (de alto nível) do projeto feito
  - O **detalhamento** das funcionalidades é feito apenas para aquelas que serão feitas na próxima iteração
- Este ciclo de re-planejamento se dá ao longo das iterações, até o produto estar completo

Gabarito dos Exercícios

- [1] – [45] E, [27A] C, [109] C, [48] A, [80] C (E)
- [2] – [76] C, [70] C
- [3] – [72] E, [27 C] C, [80] E, [81] C, [86] E
- [4] – [79] C, [32-C] E, [32-D] E, [32-E] E
- [5] – [44-A] C, [44-C] E
- [6] – [72] E
- [7] – [62] C, [35-D] E
- [8] – [108] E, [90] E
- [9] – [53] E
- [10] – [32-B] C, [51] B

FIM

Rational Unified Process

Fases, Disciplinas e Atividades

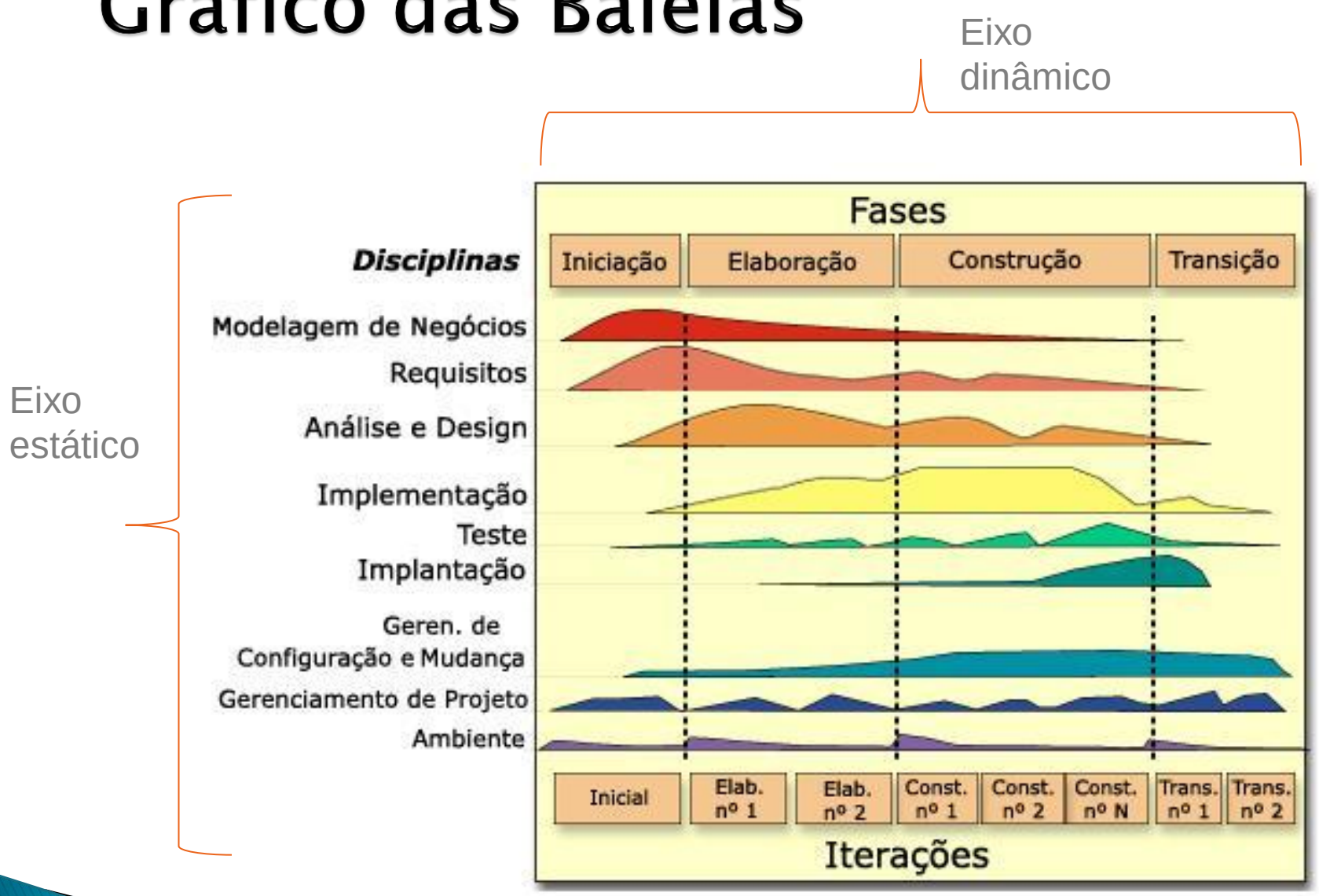
Fernando Pedrosa – fpedrosa@gmail.com

Rational Unified Process (RUP)

O RUP tem duas dimensões

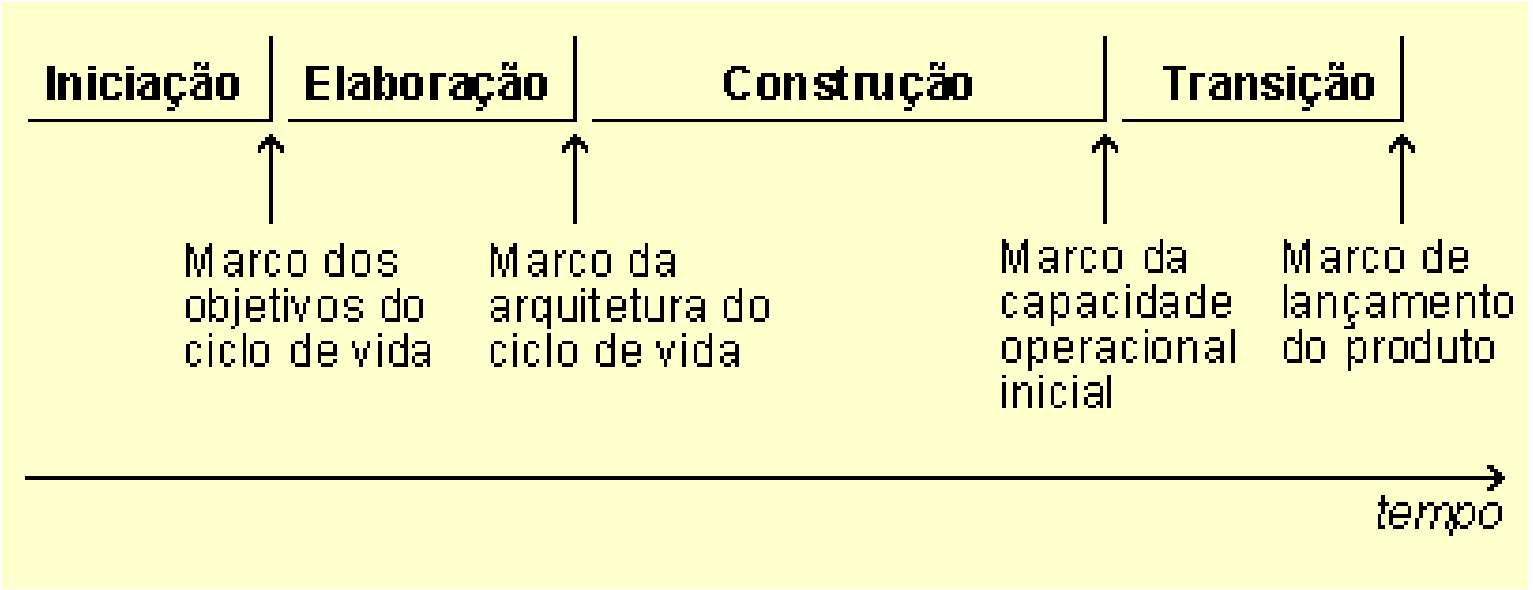
- A primeira dimensão representa o aspecto dinâmico do processo
  - Eixo horizontal
  - Expresso em termos de fases, marcos e iterações
- A segunda dimensão representa o aspecto estático do processo
  - Eixo vertical
  - Expresso em termos de componentes, disciplinas, atividades, artefatos, papéis…

Gráfico das Baleias



Fases do RUP

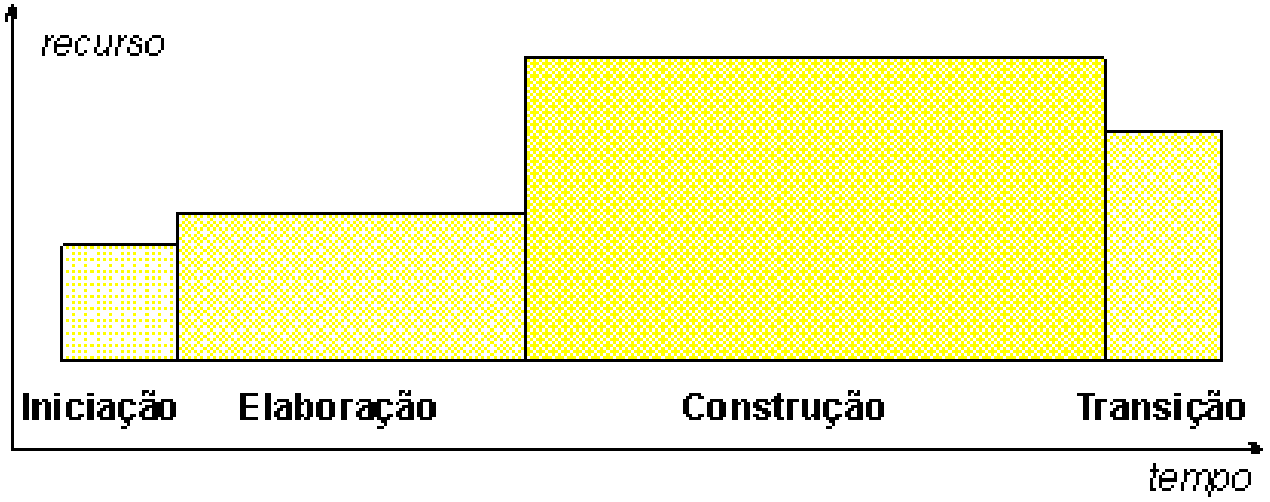
- São quatro fases sequenciais, cada uma concluída por um marco principal
- Cada fase é basicamente um intervalo de tempo entre dois marcos principais



Fases do RUP

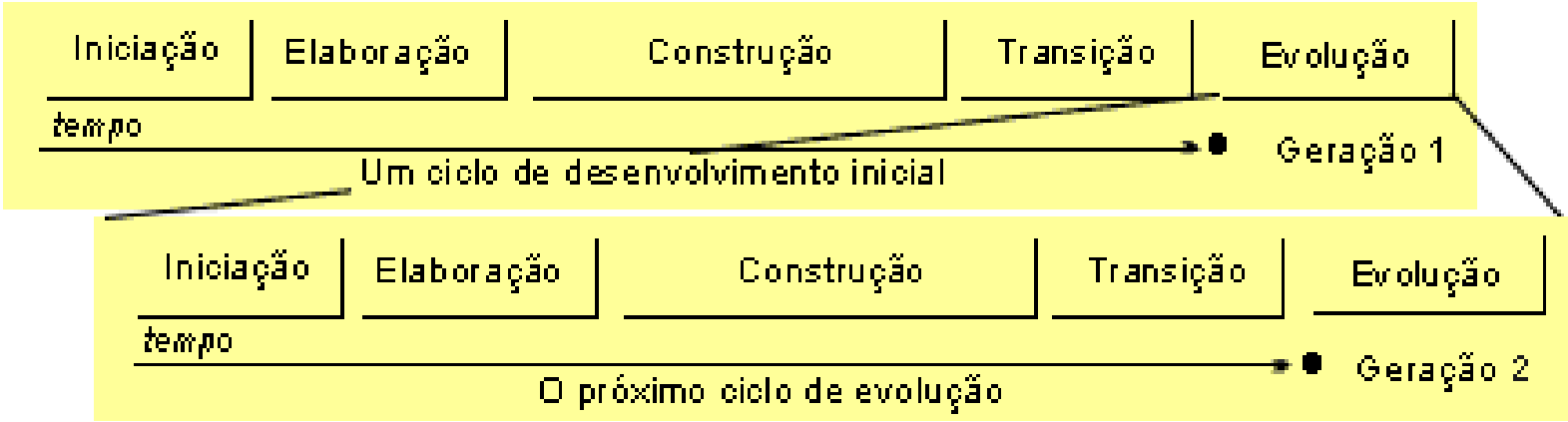
- As fases não são idênticas em termos de programação e esforço

	Iniciação	Elaboração	Construção	Transição
Esforço	~5 %	20 %	65 %	10%
Programação	10 %	30 %	50 %	10%



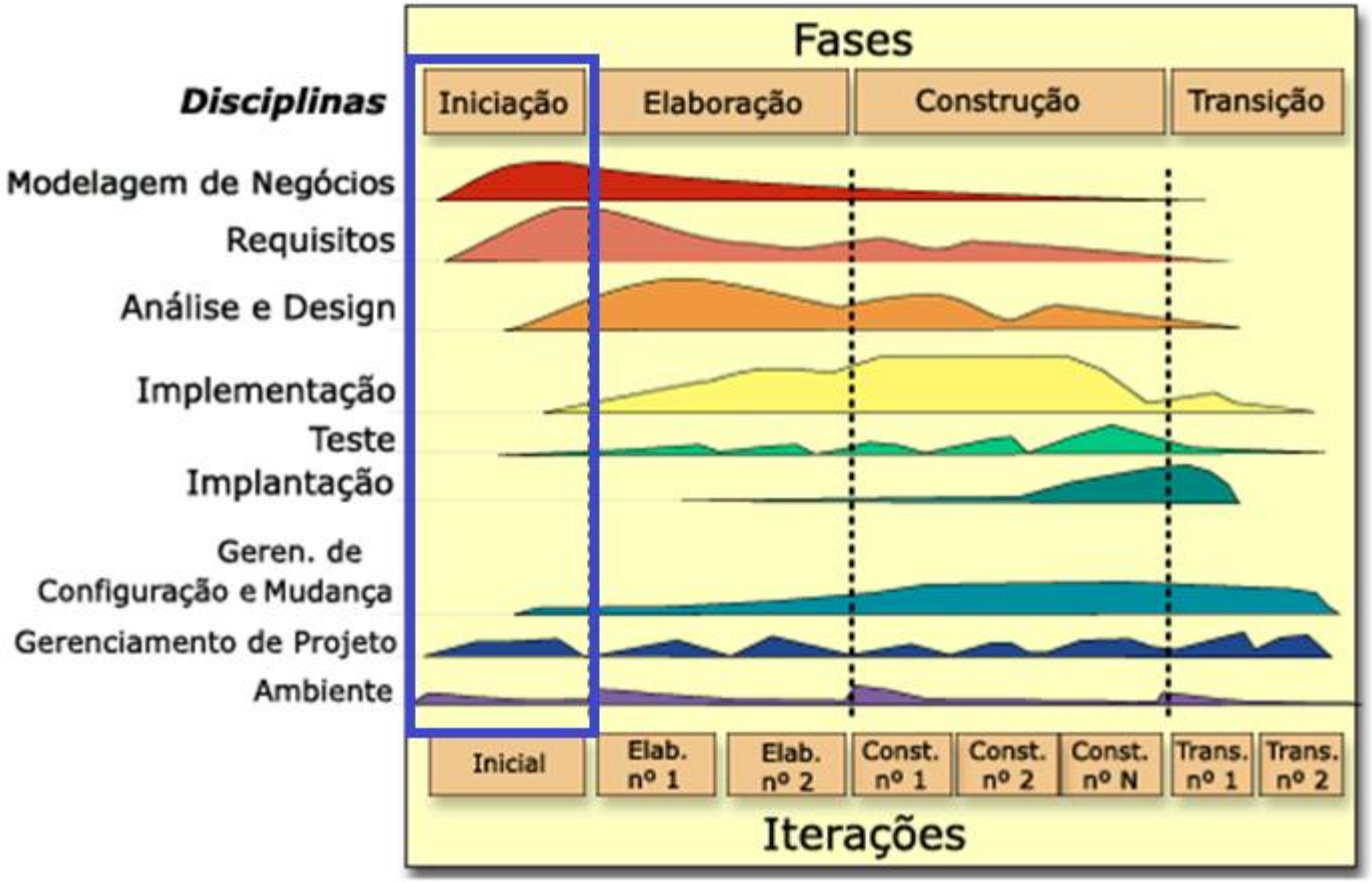
Fases do RUP

- Uma passagem pelas quatro fases é um **ciclo de desenvolvimento**
- As próximas "gerações" do produto têm ênfase em fases diferentes e são geradas por **ciclos de evolução**

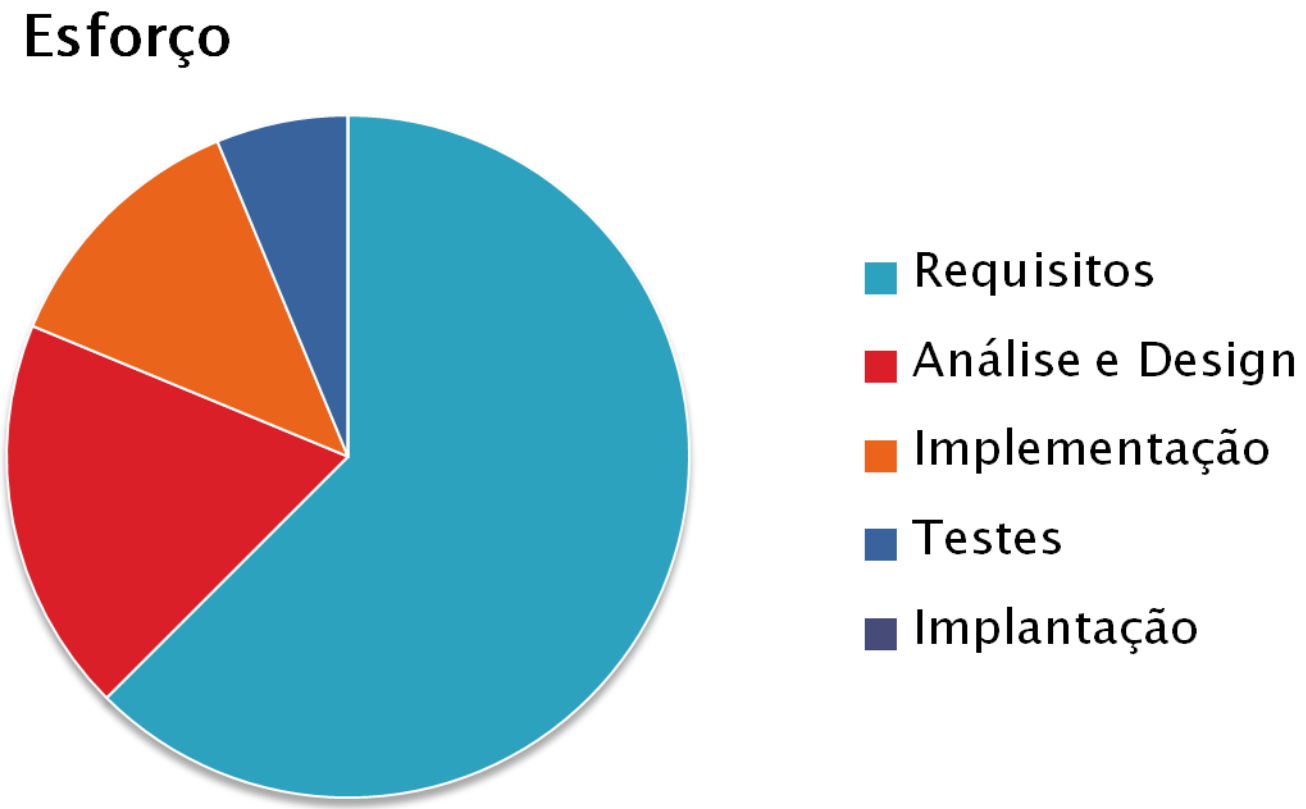


Fase de Iniciação (Concepção)

Fase de Iniciação



Fase de Iniciação



Visão Geral

- Meta principal: atingir o consenso sobre os objetivos do ciclo de vida do projeto
  - Muito importante para projetos novos
  - Para projetos evolutivos, é uma fase mais rápida
- **Compensa fazer o projeto?**
- **É possível fazer o projeto?**

Objetivos

- Estabelecer o **escopo** do Software
- Estimar **custos**
- Estimar **tempo** (cronograma)
- Estimar **riscos**
- Identificar casos de uso críticos e principais cenários operacionais
- Propor pelo menos uma opção de arquitetura para alguns cenários básicos

Principais artefatos

- Visão
  - Necessidades e características mais importantes do sistema
- Caso de Negócio
  - Informações do ponto de vista do negócio
  - Determina se vale a pena investir no projeto (ROI)
- Plano de Desenvolvimento de Software
  - Reúne todas as informações necessárias ao gerenciamento do projeto

Principais artefatos

- Modelo de Casos de Uso
  - Contém as funções pretendidas do sistema
  - Serve como um contrato estabelecido entre os clientes e os desenvolvedores
- Glossário
  - Define termos importantes usados pelo projeto

<<Iniciação>>

Ênfase na Disciplina:
Modelagem de Negócios

Modelagem de Negócios: Objetivos

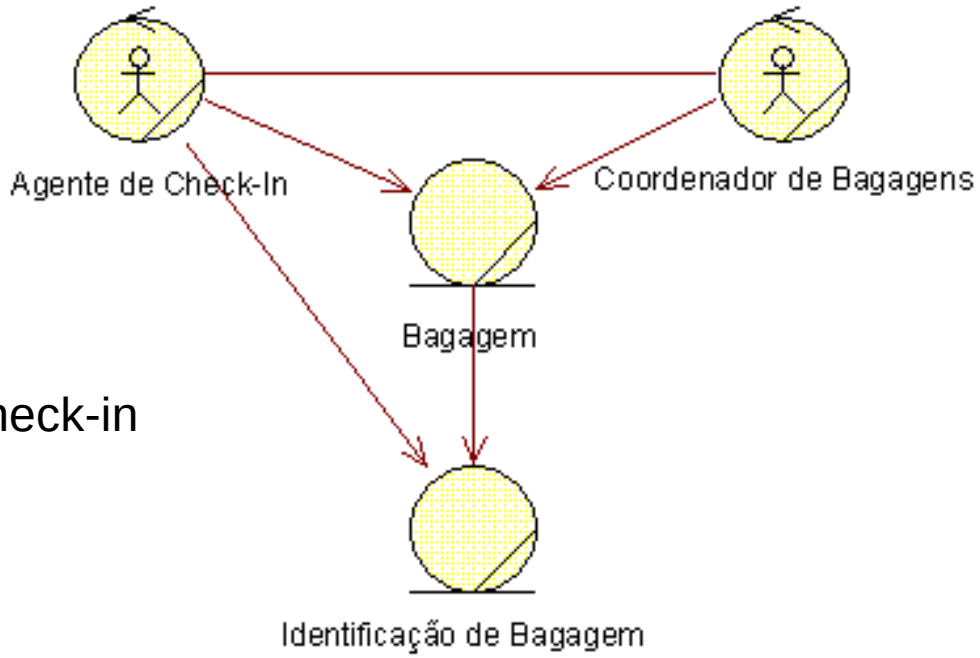
- Entender a estrutura e a dinâmica da organização-alvo, identificando oportunidades de melhoria
- Assegurar que todos os interessados tenham um entendimento comum sobre a organização
- Derivar os requisitos de sistema necessários para sustentar a organização-alvo

Modelagem de Negócios: Papeis, Atividades e Artefatos

- Principal Papel e suas Atividades
  - **Analista de Processo de Negócios**
    - Identificar os processos na organização
    - Descrever os processos
    - Definir o que pode e deve ser melhorado
    - Redesenhar os processos, se necessário
- Artefato importante para o marco
  - Modelo de Domínio (modelo de objetos de negócio)

Modelo de Domínio

Captura os tipos mais importantes de objetos no contexto de domínio



Sistema de Check-in

Relação com outras disciplinas

- Requisitos
  - Utiliza modelos de negócio como subsídio para entender os requisitos do sistema
- Análise e Design
  - Utiliza entidades de negócio para identificar classes de entidade no projeto
- Ambiente
  - Desenvolve e mantém artefatos de suporte, como o Guia de Modelagem de Negócios

<<Iniciação>>

Ênfase na Disciplina:
Requisitos

Requisitos: Objetivos

- Estabelecer o que o sistema deve fazer
- Definir as fronteiras (escopo) do sistema
- Fornecer uma base para planejar o conteúdo técnico das iterações
- Fornecer uma base para estimar o custo e o tempo de desenvolvimento do sistema

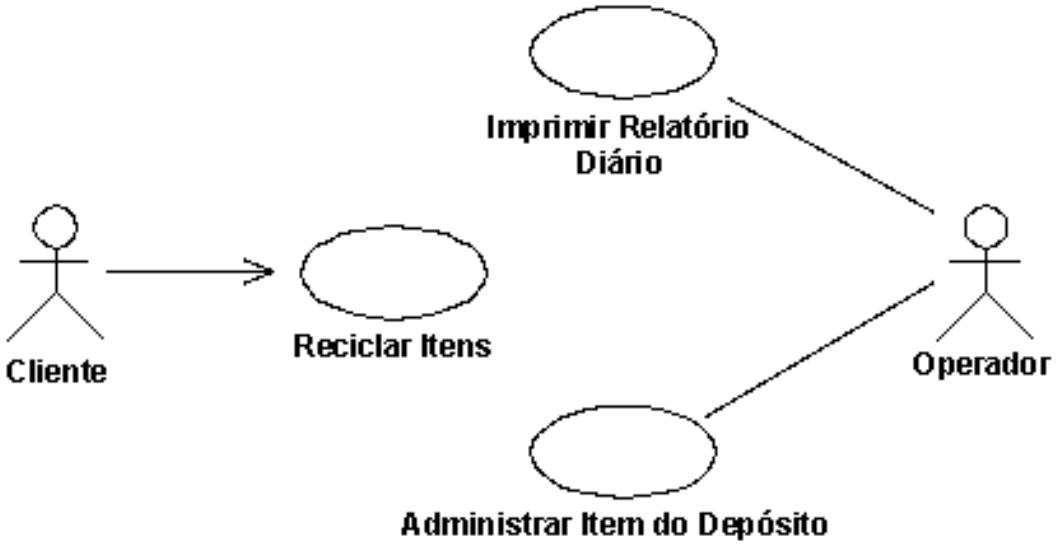
Requisitos: Papeis, Atividades e Artefatos

- Principais Papeis e Atividades
  - **Analista de Sistemas**
    - Levantar Requisitos do Sistema (Atores e CDU's)
    - Estruturar Modelo de Casos de Uso
  - **Especificador de Requisitos**
    - Detalhar Especificação de Casos de Uso
- Artefatos importantes para o marco
  - Visão
  - Glossário
  - Modelo de Casos de Uso

Modelo de Casos de Uso

Modelo das funções pretendidas do sistema. Serve como contrato entre o cliente e desenvolvedores.

Sistema de
Máquina de Reciclagem



Documento de Visão

- É o documento que define a visão que os envolvidos têm do produto a ser desenvolvido
- Contém as necessidades e características mais importantes do sistema
- Fornece uma base de alto nível para que o leitor possa compreender o sistema a ser desenvolvido

Relação com outras disciplinas

- Modelagem de Negócios
  - Fornece as regras de negócio e um contexto organizacional para o sistema
- Análise e Design
  - Obtém suas informações primárias dos Requisitos. Pode encontrar falhas nos modelos de Caso de Uso
- Teste
  - Valida o sistema com base nos casos de uso

Relação com outras disciplinas

- Gerenciamento de Configuração e Mudança
  - Fornece o mecanismo de controle para as mudanças nos requisitos
- Gerenciamento de Projeto
  - Usa o modelo de casos de uso para planejar as iterações
- Ambiente
  - Desenvolve e mantém os artefatos utilizados na disciplina de Requisitos

Marco da Iniciação: Objetivos do Ciclo de Vida

- É o primeiro marco mais importante do projeto
- Critérios de avaliação
  - Os casos de uso definem claramente o escopo?
  - Caso necessário, foi possível fazer um protótipo da arquitetura?
  - Todos os riscos críticos foram encontrados? Se sim, foram mitigados?
  - **Há condições de se fazer o projeto?**

Exercícios [1]

(BASA – CESPE 2007)

[53] Na fase de concepção (inception), há atividades voltadas para a definição do escopo do sistema, identificação de atores e casos de uso, definição de vocabulário que possa ser usado nas descrições textuais do sistema, e definição de uma arquitetura candidata para o sistema que está sendo desenvolvido.

(Min. Comunicações – CESPE 2008)

[72] São objetivos da fase de concepção (inception): preparar ambiente para o projeto; elaborar plano para o projeto; definir escopo do sistema; identificar atores e casos de uso; identificar as necessidades dos stakeholders; definir níveis de prioridade dos casos de uso; propor arquitetura candidata; e definir objetivos do esforço de teste.

Exercícios [1]

(CGU – ESAF 2008)

[22] No Processo Unificado (PU), o termo Modelo de Domínio significa uma representação visual de classes conceituais ou objetos do mundo real. Assinale a opção que apresenta uma afirmativa correta quanto ao Modelo de Domínio.

A) Não trata da representação de objetos de software.

B) Significa um conjunto de diagramas que descreve classes de software.

C) Representa a camada de domínio de uma arquitetura de software.

D) Representa objetos de software com responsabilidades.

E) Aplicando a notação UML, é ilustrado como um conjunto de diagramas de classe em que são definidas as operações.

Exercícios [1]

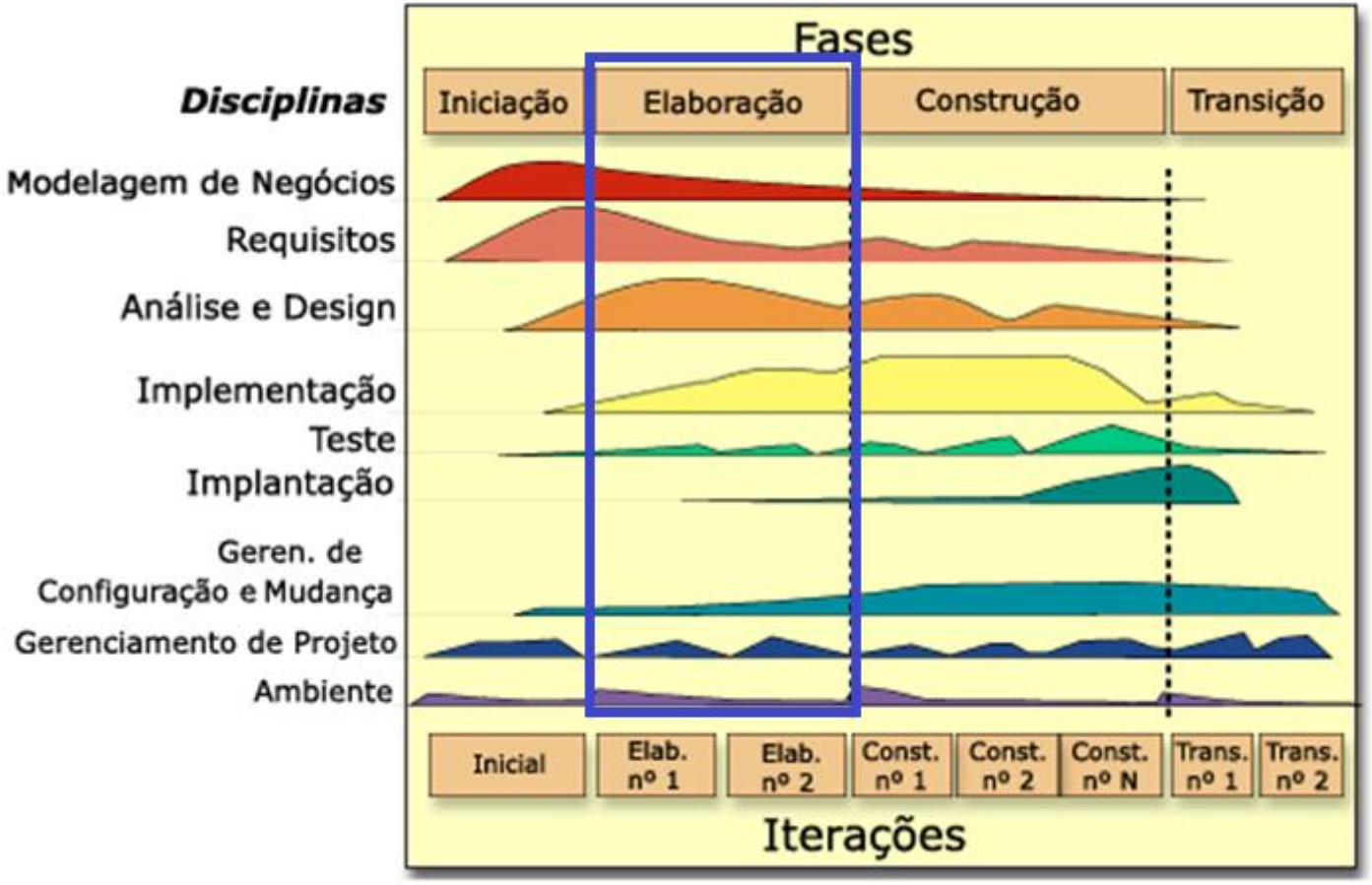
(Sergipe Gás – FCC 2010)

[31] No Processo Unificado, o Modelo de Domínio é um

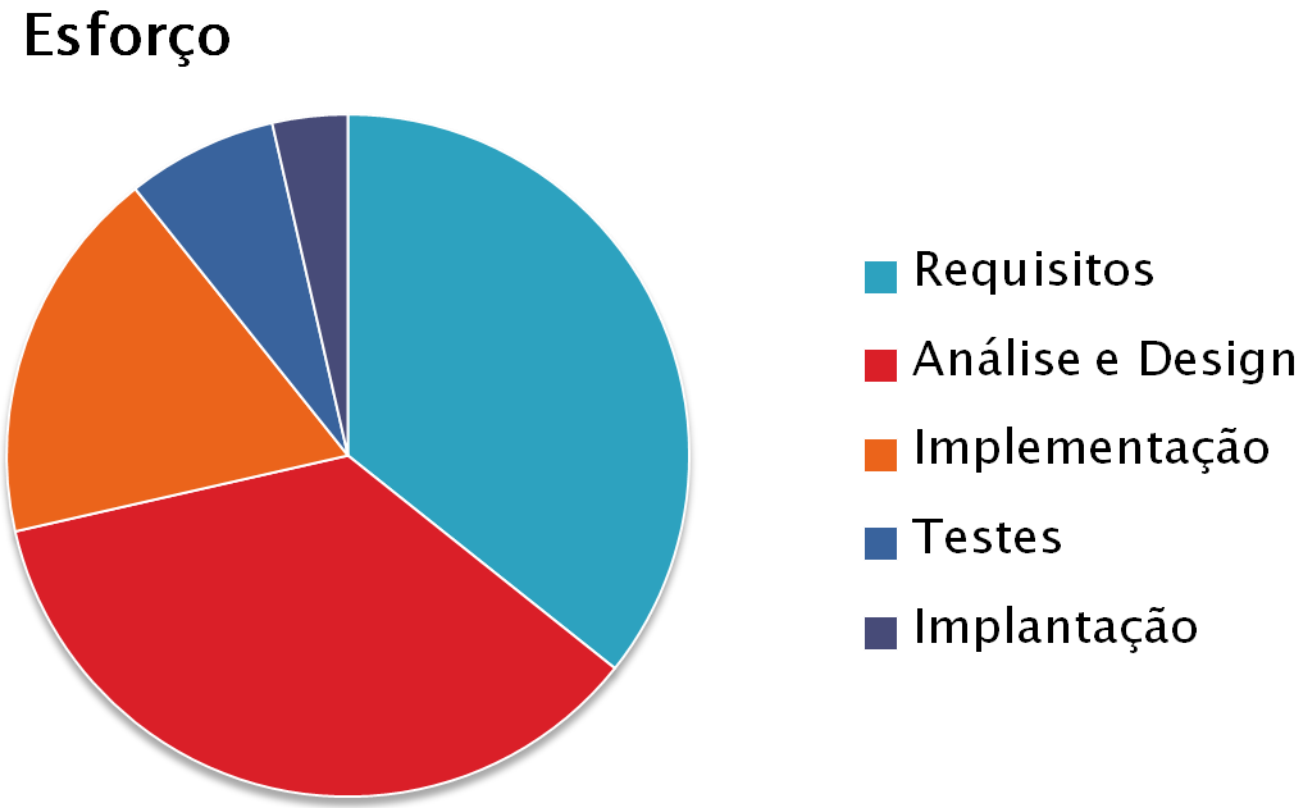
a) diagrama de classes em nível de análise.
b) diagrama de classes em nível de desenho.
c) produto da modelagem de negócios e, como tal, captura o vocabulário do sistema ou negócio sob modelagem.
d) modelo que carrega todo o detalhamento do comportamento e estrutura, que devem estar presentes em um modelo de análise.
e) modelo de domínio que carrega informações de armazenamento de informações ou normalizações, que devem estar presentes em um DER

Fase de Elaboração

Fase de Elaboração



Fase de Elaboração



Visão Geral

- A meta principal da fase de Elaboração é fornecer uma base estável para o esforço de Construção
- A arquitetura é desenvolvida a partir dos requisitos que têm maior impacto na arquitetura
- A estabilidade da arquitetura é avaliada através de um ou mais protótipos de arquitetura

Objetivos

- Assegurar que a arquitetura e os requisitos estejam estáveis para mitigar riscos
  - “Ultrapassar esta marca significa passar de uma operação rápida e de baixo risco para uma operação de alto custo e alto risco”
- Tratar todo os riscos significativos do ponto de vista da arquitetura do projeto
- Selecionar componentes e criar planos de iterações detalhados para a fase de Construção

Principais artefatos

- Protótipos
  - São usados de uma maneira direta para reduzir o risco e elicitar requisitos significativos.
- Documento de Arquitetura de Software
  - Fornece a visão geral de arquitetura abrangente do sistema, usando diversas visões de arquitetura para descrever diferentes aspectos do sistema.

Principais artefatos

- Modelo de Projeto
  - É um modelo de objeto que descreve a realização dos casos de uso e serve como uma abstração do modelo de implementação e seu código-fonte.
- Modelo de Dados
  - É um subconjunto do modelo de implementação que descreve a representação lógica e física dos dados persistentes no sistema.

<<Elaboração>>

Ênfase na Disciplina:
Análise e Design

Análise e Design: Objetivos

- Transformar os requisitos em um projeto do sistema a ser criado
- Desenvolver uma arquitetura refinada para o sistema
- Adaptar o projeto para que corresponda ao ambiente de implementação, considerando restrições de tecnologia

Análise e Design: Papeis, Atividades e Artefatos

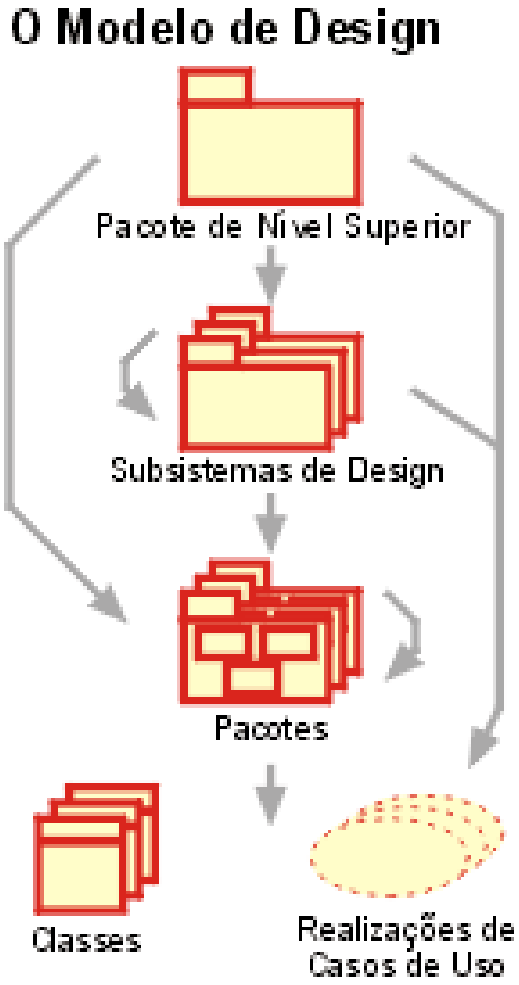
- Principais Papeis e atividades
  - **Arquiteto de Software**
    - Projetar arquitetura
  - **Designer (Projetista)**
    - Analisar casos de uso
    - Projetar casos de uso
    - Projetar subsistemas
  - **Projetista de Banco de Dados**
    - Projetar base de dados

Análise e Design: Papeis, Atividades e Artefatos

- Artefatos importantes para o marco
  - **Protótipos**
    - Quanto ao que exploram:
      - Comportamentais
      - Estruturais
    - Quanto ao seu resultado:
      - Exploratórios (ou de descarte)
      - Evolutivos
  - **Documento de Arquitetura de Software**
  - **Modelo de Design**

Modelo de Design

Modelo que descreve as realizações dos casos de uso e serve como uma abstração do modelo de implementação



Relação com outras disciplinas

- Modelagem de negócio
  - Fornece o contexto organizacional para o sistema
- Requisitos
  - Fornece a visão das funcionalidades críticas a serem implementadas
- Teste
  - Testa o sistema projetado durante a disciplina de Análise e Design
- Ambiente, Gerenciamento de Projeto

Marco da Elaboração: Arquitetura do Ciclo de Vida

- É o segundo marco mais importante do projeto. Deve-se analisar a arquitetura executável e a resolução dos principais riscos
- Critérios de Avaliação
  - A arquitetura é estável e robusta, comportando requisitos atuais e futuros?
  - Riscos críticos foram resolvidos?
  - O planejamento está bem definido em termos de cronograma, orçamento e níveis de qualidade?
  - **Devemos fechar o contrato?**

Exercícios [2]

(SERPRO - CESPE 2010)

[73] No modelo RUP, a primeira linha de base da arquitetura de um software é produzida ao final da fase de elaboração.

(ISJN - CESPE 2010)

[57] Modelo de domínio, descrição da arquitetura de software e versão preliminar do manual são resultados-alvo da fase elaboração do RUP.

[54] Na fase de elaboração, muitos componentes do sistema são implementados, testados e integrados. Essas atividades, que partem de uma arquitetura definida, validada e implementada em fases anteriores do ciclo de desenvolvimento, produzem um sistema operacional pronto para ser instalado em um ambiente em que serão feitos testes beta.

Exercícios [2]

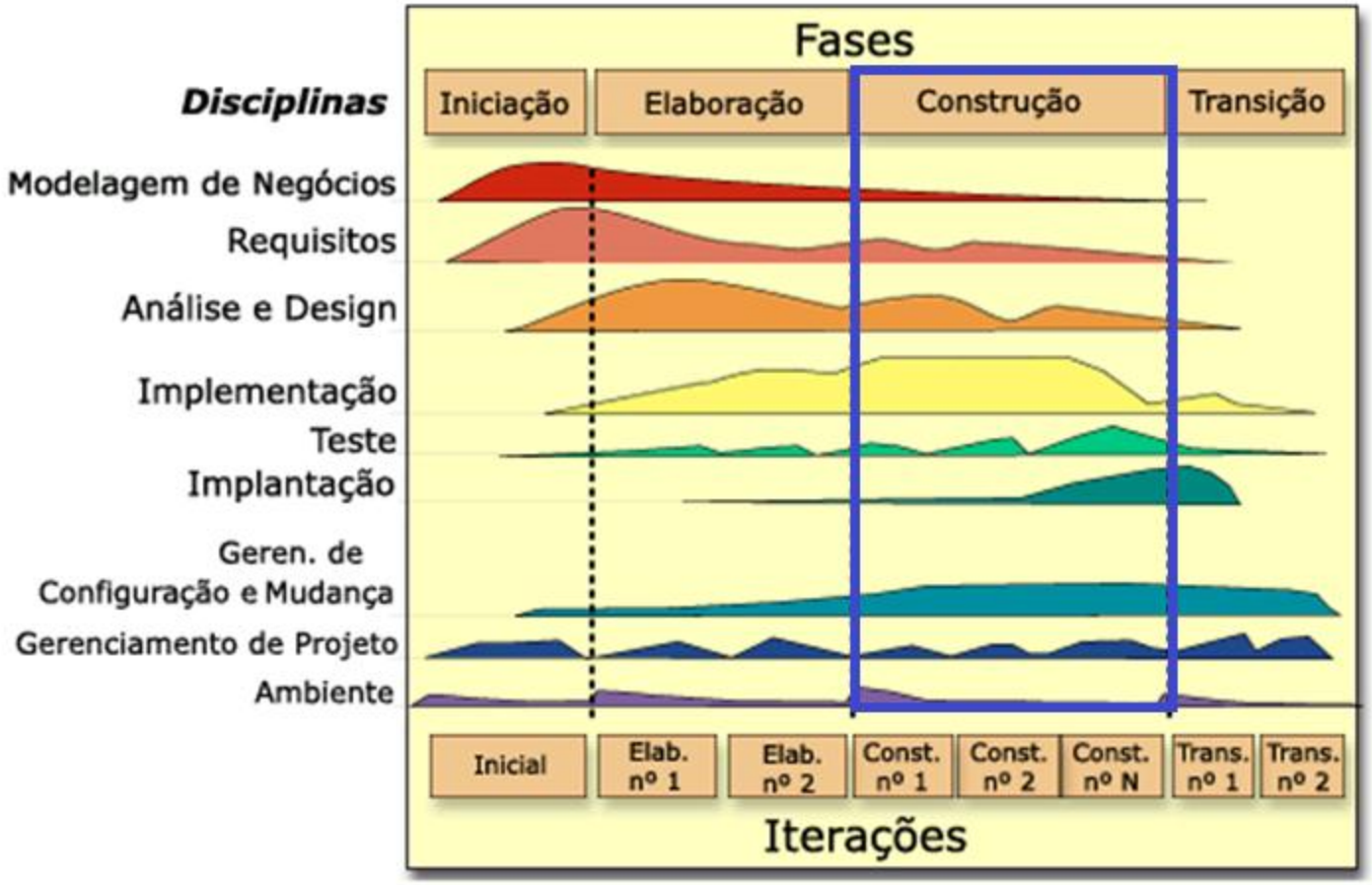
(PETROBRAS – CESGRANRIO 2010)

[39] A análise de risco no RUP é algo constante nas diversas fases do processo de desenvolvimento. Em cada uma das fases, o foco da gerência de riscos se diferencia em função do objetivo de cada fase. Assim, a manipulação dos riscos está relacionada, na fase de

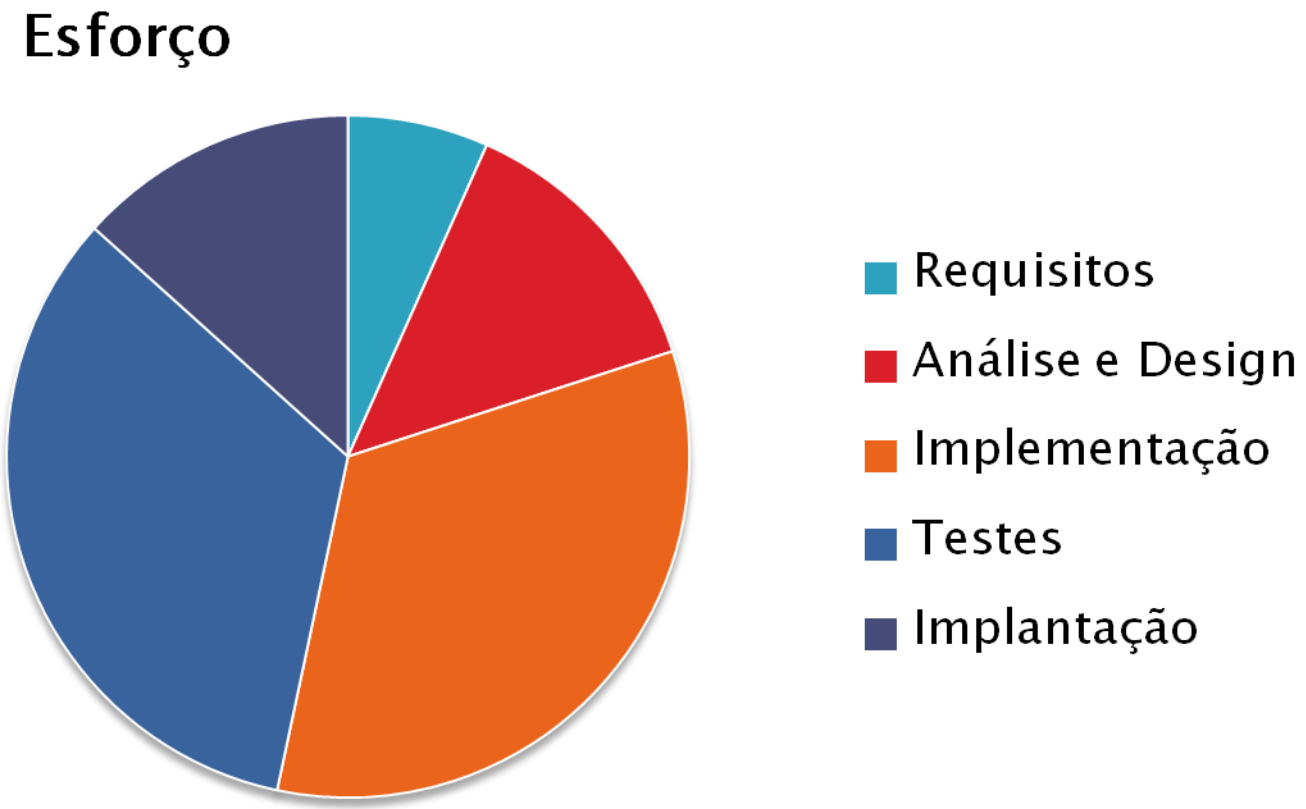
(A) análise, ao refinamento do modelo de requisitos e à sua possível alteração.
(B) construção, à instalação e distribuição do produto no ambiente do cliente.
(C) transição, à logística, uma vez que é a fase que envolve o maior número de profissionais.
(D) requisitos, à modelagem de negócio.
(E) elaboração, a questões técnicas, envolvendo a arquitetura escolhida.

Fase de Construção

Fase de Construção



Fase de Construção



Visão Geral

- Esclarecer os requisitos restantes
- Concluir o desenvolvimento do sistema com base na arquitetura estável
- É, de certa forma, um processo de manufatura
- A ênfase está no gerenciamento de recursos e controle de operações para alcançar maior produtividade e qualidade

Objetivos

- Minimizar custos de desenvolvimento, otimizar recursos e evitar retrabalho
- Atingir as versões úteis (alfa, beta e outros releases de teste) com rapidez
- Concluir a análise, o projeto, o desenvolvimento e o teste de todas as funcionalidades necessárias.
- Decidir se o software está pronto para ser implantado

Principais artefatos

- "O Sistema"
  - O próprio sistema executável, pronto para iniciar os testes beta
- Plano de Implantação
  - Versão inicial desenvolvida, analisada e com baseline
  - Em projetos menores pode estar embutido no Plano de Desenvolvimento do Software
- Conjunto de testes
  - Testes implementados e executados para validar a estabilidade dos releases

<<Construção>>

Ênfase na Disciplina:
Implementação

Implementação: Objetivos

- Definir o código em subsistemas e camadas
- Implementar classes e objetos em termos de componentes
- Testar os componentes desenvolvidos como unidades
- Integrar os resultados produzidos por implementadores individuais (ou equipes) ao sistema executável

Implementação: Papeis, Atividades e Artefatos

- Principais Papeis e atividades
  - **Implementador**
    - Implementar componente
    - Realizar testes unitários
  - **Integrador**
    - Integrar o sistema
- Artefatos importantes para o marco
  - “O Sistema”
  - Componentes
  - Builds

Relação com outras disciplinas

- Análise e Design
  - Representa o propósito da implementação, sendo o Modelo de Design a principal entrada desta disciplina
- Teste
  - Descreve como realizar o teste de integração de cada build
  - Descreve também como testar o sistema
- Implantação
  - Descreve como usar o modelo de implementação para produzir e liberar o código para o cliente final
- Ambiente, Gerenciamento de Projeto

<<Construção>>

Ênfase na Disciplina: **Teste**

Objetivos

- Localizar e documentar defeitos na qualidade do software
- Validar as suposições feitas nas especificações de design e requisito através de demonstração concreta
- Validar as funções do software conforme projetadas
- Verificar se os requisitos foram implementados de forma adequada

Testes: Papeis, Atividades e Artefatos

- Principais Papeis e atividades
  - **Analista de Teste**
    - Elaborar plano de testes
  - **Projetista de Teste**
    - Projetar testes
  - **Testador**
    - Implementar teste
    - Executar Testes
- Artefatos importantes para o marco
  - Conjunto de testes

Relação com outras disciplinas

- Requisitos
  - Os casos de uso fornecem a base para o que vai ser testado na disciplina de Teste
- Análise e Design
  - Fornece o projeto a ser testado pela disciplina de Teste
- Implementação
  - Produz os builds do software que serão validados pelos testes
- Ambiente, Gerenciamento de Projeto, Gerenciamento de Configuração e Mudanças

Marco da Construção: Capacidade Operacional Inicial

- Neste marco, o produto está pronto para ser passado para a Equipe de Transição.
- Toda a funcionalidade já foi desenvolvida e todos os testes **alfa** foram concluídos.
- O manual do usuário é produzido e uma existe uma descrição do release atual
- Critérios de avaliação
  - O produto está estável para ser implantado?
  - O resultado está coerente com o planejado?
  - **Os envolvidos estão prontos para a Transição?**

Exercícios [3]

(PETROBRAS – CESPE 2007)

[98] Na fase de construção, são implementados os casos de uso que tenham impacto sobre a arquitetura; na fase de transição, os casos sem impacto sobre a arquitetura, mas que descrevam funcionalidades que deverão estar presentes na versão que está sendo desenvolvida.

(PGE/PA – CESPE 2007)

[34] Na disciplina de teste, entre os artefatos que podem ser produzidos, tem-se o plano de teste. O plano de teste pode definir os objetivos dos testes no escopo de uma iteração ou do projeto, os itens a serem testados e as abordagens dos testes.

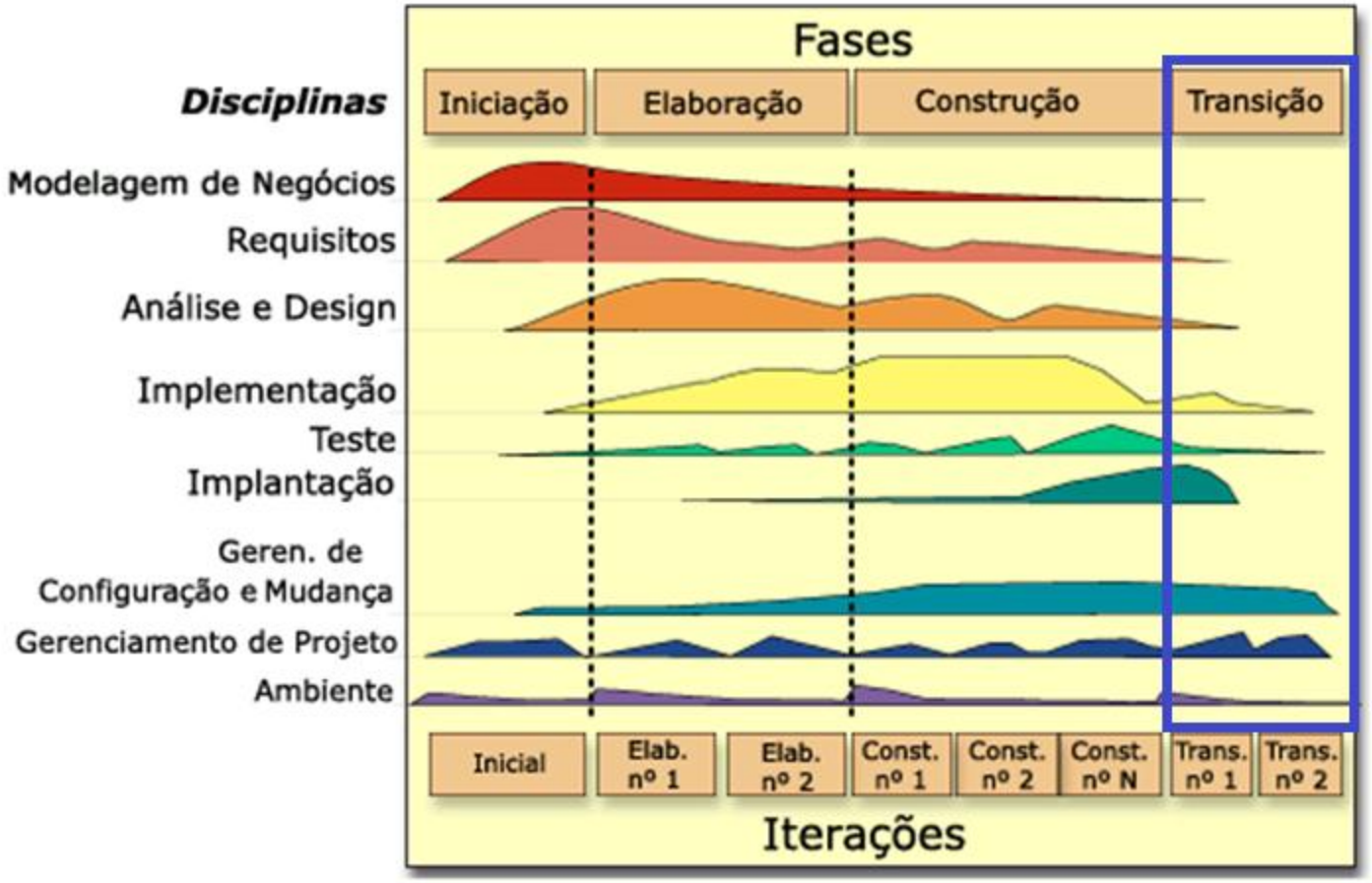
Exercícios [3]

(Min. Comunicações – CESPE 2008)

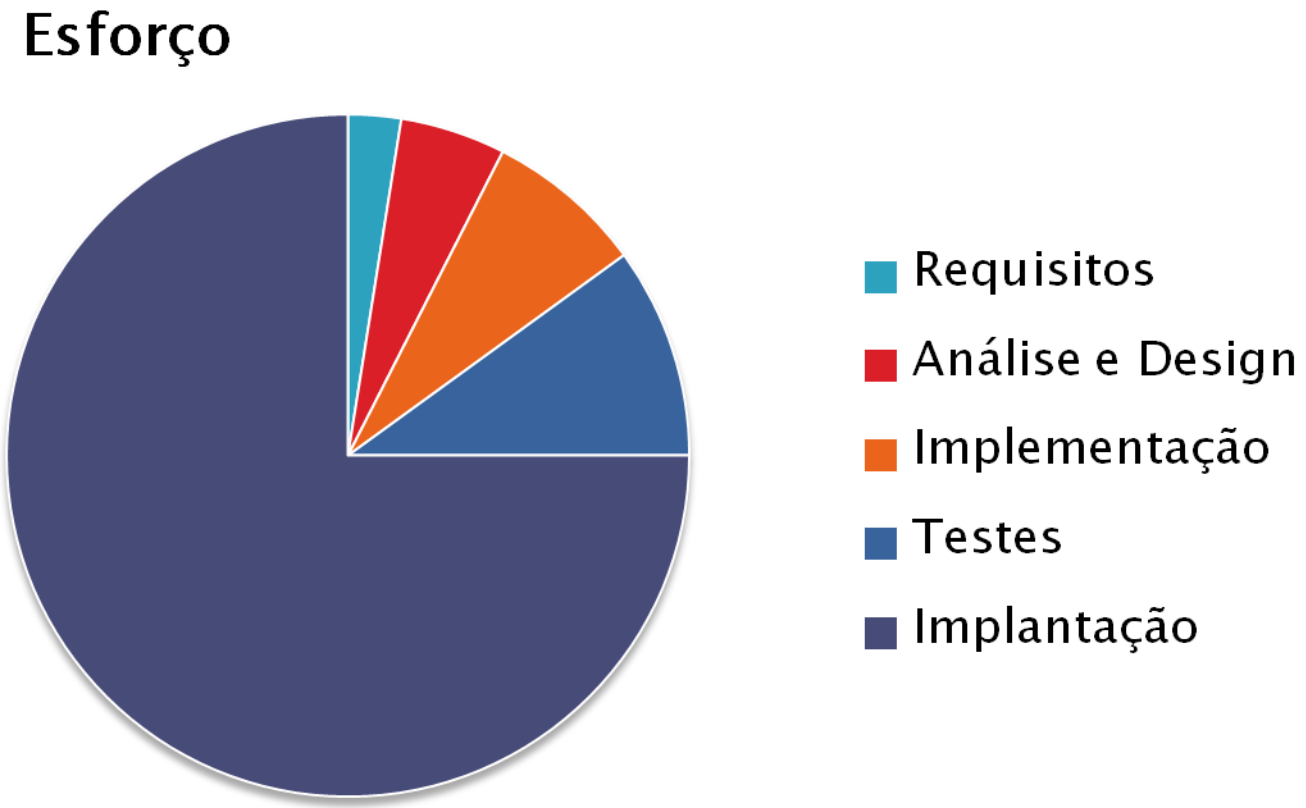
[73] A fase de construção (construction) tem os seguintes objetivos: detalhar casos de uso e requisitos do software; refinar a arquitetura proposta e demonstrar que essa arquitetura suporta os requisitos do sistema; testar e avaliar protótipos visando demonstrar que os principais riscos foram avaliados; e construir protótipos executáveis para a avaliação da arquitetura proposta.

Fase de Transição

Fase de Transição



Fase de Transição



Visão Geral

- O foco da fase de Transição é disponibilizar o software aos usuários finais
- Pode atravessar várias iterações e inclui testar os releases e ajustar pequenos defeitos com base no feedback do usuário
- O feedback prioriza apenas ajustes finos – todos os problemas estruturais mais graves devem ter sido trabalhados muito antes no ciclo de vida do projeto
- Pode ser uma fase muito fácil ou muito complexa, dependendo do tipo do produto

Objetivos

- Teste beta para validar o novo sistema
- Treinamento de usuários e equipe de manutenção
- Atividades de ajuste, como correção de erros, melhoria no desempenho e na usabilidade.
- Consentimento dos envolvidos que o software implantado atende a visão inicial

Principais artefatos

- Notas de Release
- Artefatos de instalação
- Material de treinamento

<<Transição>>

Ênfase na Disciplina: **Implantação**

Implantação: Objetivos

- Coordenar e gerenciar os testes beta e de aceitação
- Desenvolver artefatos de instalação e materiais de treinamento
- Liberar para fabricação
- Há três tipos de instalação
  - a instalação personalizada
  - o produto em uma forma "compacta"
  - acesso ao software por meio da Internet

Implantação: Papeis, Atividades e Artefatos

- Principais Papeis e atividades
  - **Gerente de Implantação**
    - Desenvolver plano de implantação
    - Escrever notas de release
  - **Desenvolver do curso**
    - Desenvolver materiais de treinamento
  - **Implementador**
    - Desenvolver artefatos de instalação

Implantação: Papeis, Atividades e Artefatos

- Artefatos importantes para o marco
  - O Build do produto
  - Notas de Release
  - Artefatos de instalação
  - Material de treinamento

Relação com outras disciplinas

- Requisitos
  - As especificações de requisitos constituem a principal fonte para a elaboração de materiais de suporte e treinamento para o usuário final
- Teste
  - Os testes são indispensáveis para a implantação do sistema
- Ambiente, Gerenciamento de Projeto, Ger. de Configuração e Mudanças

Marco da Implantação: Release do Produto

- Neste marco, você decide se os objetivos foram atendidos e se outro ciclo de desenvolvimento deve ser iniciado
- Critérios de avaliação
  - As despesas reais com recursos são aceitáveis se comparadas às planejadas?
  - **O usuário está satisfeito?**

Exercícios [4]

(CGU – ESAF 2008)

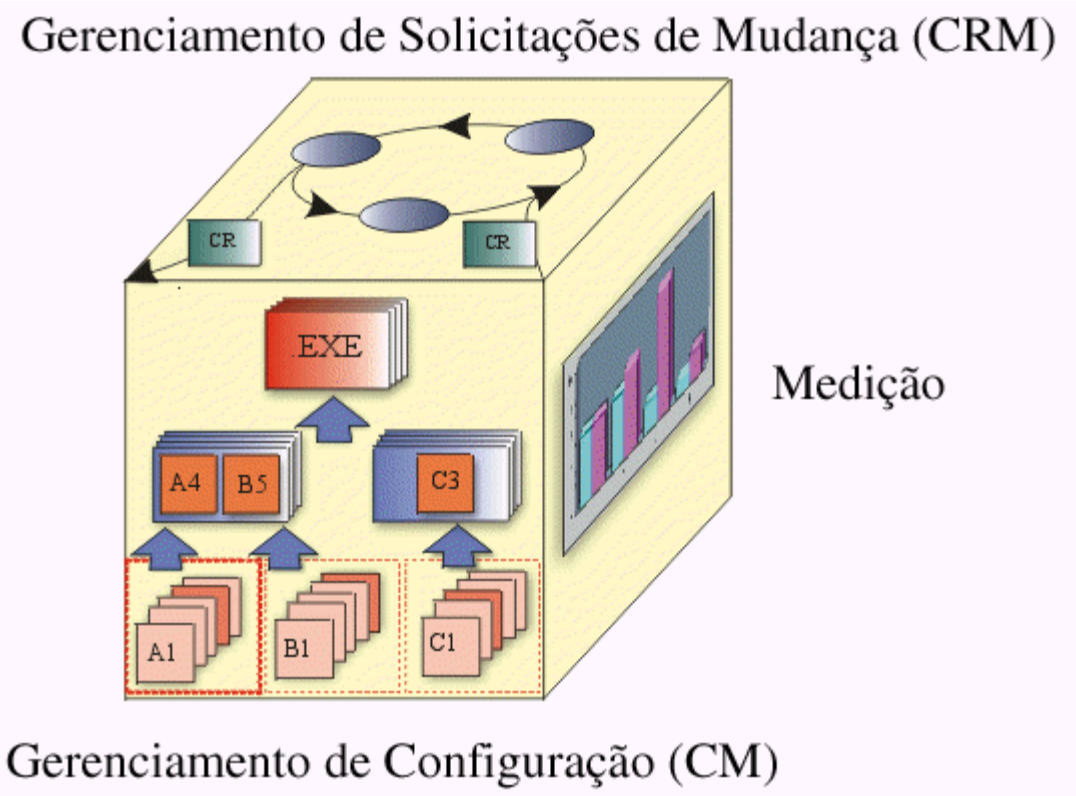
[25] No RUP (Rational Unified Process), dois dos exemplos dos artefatos de Implantação são:

A) Guia de design e Arte final do produto.
B) Material de suporte para o usuário e Guia de teste.
C) Plano de implantação e Manual de guia de estilo.
D) Notas de release e Materiais de treinamento.
E) Artefatos de Instalação e Guia de ferramentas.

Disciplinas de Suporte:
Gestão de Configuração e Mudanças

Visão Geral

- Controla mudanças feitas nos artefatos de um projeto e mantém a integridade entre eles



Objetivos

- Identificar e controlar itens de configuração
- Restringir as mudanças nesses itens
- Auditar as mudanças nesses itens
- Evitar confusões de:
  - Atualização simultânea
  - Notificação limitada
  - Várias versões

Benefícios

- Preservação da estabilidade e integridade do produto
- Suporte a métodos de desenvolvimento
- Restrição das mudanças feitas nos artefatos com base nas políticas do projeto
- Trilha de auditoria indicando por que, quando e por quem um artefato foi alterado

Gestão de Configuração e Mudanças: Papeis, Atividades e Artefatos

- Principais Papeis e atividades
  - **Gerente Configuração**
    - Configurar ambiente de CM
    - Estabelecer políticas de CM
  - **Gerente de Mudanças**
    - Estabelecer processo de controle de mudanças
    - Confirmar ou recusar CR
- Principais artefatos
  - Repositório do projeto
  - Solicitações de mudanças

Disciplinas de Suporte:
Gerenciamento de Projeto

Objetivos

- Gerenciar pessoas, equipes fases e iterações para executar e monitorar o projeto
- Planejar o cronograma do projeto
- Gerenciar a qualidade e realizar revisões
- **Gerenciar os riscos do projeto**

Atenção!

- Não cobre problemas como:
  - Gerenciamento de Pessoal: contratação, treinamento, ensino
  - Gerenciamento de Orçamento: definição, alocação, etc.
  - Gerenciamento de contratos com fornecedores e clientes

É para isso que temos o PMBOK!

Gestão de Projetos: Papeis, Atividades e Artefatos

- Principais Papeis e atividades
  - **Gerente de Projeto**
    - Planejar Fases e Iterações
    - Identificar e Avaliar Riscos
    - Monitorar o Projeto e Resolver Problemas
- Principais artefatos
  - Plano de Desenvolvimento de Software
  - Planos de Iteração
  - Lista de Riscos

Disciplinas de Suporte:
Ambiente

Objetivos

- **Configurar o processo para o projeto**
- Selecionar e adquirir ferramentas
- Desenvolver ou adaptar templates específicos para o projeto
- Desenvolver ou adaptar guias de atividades
- Oferecer à organização o ambiente de desenvolvimento de software que dará suporte à equipe de desenvolvimento.

Ambiente: Papeis, Atividades e Artefatos

- Principais Papeis e atividades
  - **Engenheiro de Processo**
    - Configurar o processo
    - Iniciar Caso de Desenvolvimento
  - **Especialista em Ferramentas**
    - Selecionar, adquirir e configurar ferramentas
- Principais artefatos
  - Caso de Desenvolvimento

Caso de Desenvolvimento

- Descreve o processo de desenvolvimento escolhido para ser seguido no projeto
- A finalidade é capturar o processo adaptado para o projeto individual
- É criado no início da fase de Iniciação e atualizado durante todo o projeto
  - À medida que o projeto progride, você adiciona mais disciplinas a cada iteração
- **É o documento que "configura" o próprio processo de desenvolvimento**

Exercícios [5]

(IJSN – CESPE 2010)

[55] Na disciplina gerência de configuração e mudanças do RUP (rational unified process), garantir a integridade dos artefatos relacionados ao projeto de software é função da gerência de solicitação de mudanças.

Exercícios [5]

(TJ/SE – FCC 2009)

[54] De acordo com o RUP, balancear objetivos, administrar riscos e superar restrições para entregar um produto que atenda às necessidades de clientes e usuários é papel do

(A) Gerente de Projetos.

(B) Analista de Sistemas.

(C) Administrador de Dados.

(D) Analista de Requisitos.

(E) Arquiteto de Software.

Exercícios [5]

(BNDES – CESGRANRIO 2008)

[1] Considerando o processo de desenvolvimento de software unificado, associe cada produto de trabalho com a fase em que deve Ser realizado. Marque a opção que ilustra a associação correta.

a) I–P, II–S, III–R, IV–P, V–Q
b) I–P, II–S, III–Q, IV–P, V–Q
c) I–P, II–R, III–Q, IV–P, V–R
d) I–Q, II–R, III–Q, IV–P, V–R
e) I–Q, II–S, III–R, IV–Q, V–S

ARTEFATOS	FASES
I - Avaliação inicial de riscos	P - Concepção
II - Relatório de execução de testes beta	Q - Elaboração
III - Modelo de projeto completo	R - Construção
IV - Modelo de negócio preliminar	S - Transição
V - Protótipo arquitetural executável	

Gabaritos dos Exercícios

- [1] 53 C, 72 C, 22 A, 31 C
- [2] 73 C, 57 E, 54 E, 39 E
- [3] 98 E, 34 C, 73 E
- [4] 25 D
- [5] 55 E, 54 A, 1 A

FIM

Requisitos

Engenharia, Levantamento, Elicitação, Gerenciamento

Fernando Pedrosa – fpedrosa@gmail.com

Bibliografia

- **Sommerville,** Ian. Software Engineering. **Editora**: Addison Wesley. (capítulos sobre Requisitos)

Engenharia de Requisitos

- Área da Engenharia de Software preocupada com:
  - Objetivos do mundo real para sistemas de software
  - Funções de sistemas de software
  - Restrições em sistemas de software
  - E o relacionamento entre esses fatores

Objetivando especificações precisas do comportamento do SW e sua evolução

Objetivos

- Estabelecer e manter concordância com os clientes e outros envolvidos sobre o que o sistema deve fazer
- Oferecer aos desenvolvedores do sistema uma compreensão melhor dos requisitos do sistema
- Definir (delimitar) as fronteiras do sistema

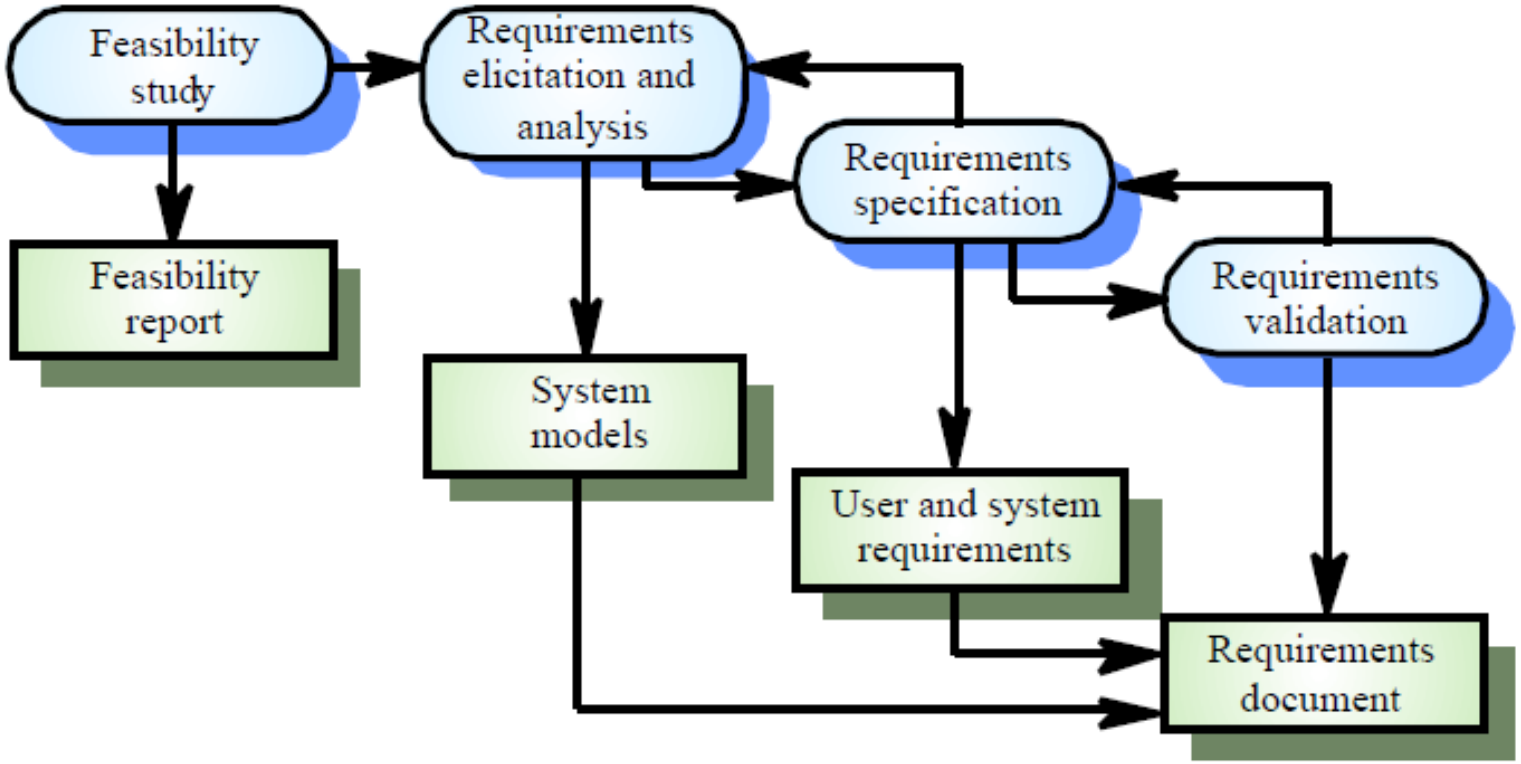
Objetivos

- Fornecer uma base para planejar o conteúdo técnico das iterações.
- Fornecer uma base para estimar o custo e o tempo de desenvolvimento do sistema.
- Definir uma interface de usuário para o sistema, focando nas necessidades e metas dos usuários

Características

- Queremos ter uma especificação de requisitos que seja:
  - Completa
  - Consistente
  - Não ambígua
  - Correta
- Que sirva, inclusive, de base para um acordo entre as partes envolvidas no processo de desenvolvimento de software

O processo de Engenharia de Requisitos



Exercícios [1]

(SAD/PE – CESPE 2010)

[35-a] A engenharia de requisitos de um software, em geral, precede a engenharia dos requisitos do sistema de informações no qual o software será usado.

(SERPRO – CESPE 2010)

[78] A área de atividade de requisitos de software apresenta maior interface com a engenharia de sistemas quando comparada à área de análise e projeto de software.

[79] Visando à maior efetividade no processo de desenvolvimento, os requisitos de software geralmente são, em geral, desenvolvidos antes dos requisitos do sistema.

Exercícios [1]

(ANAC – CESPE 2009)

[65] Requisitos descrevem um acordo ou contrato entre duas partes, especificando, entre outros aspectos, o que o sistema de software deve fazer para ser aprovado em um teste de aceitação.

O que é um requisito?

- Um requisito é definido como “uma condição ou uma capacidade com a qual o sistema deve estar de acordo”
  - Pode ser desde uma indicação abstrata, de alto nível, até uma especificação matemática detalhada

Em resumo: definem o que o sistema deve fazer e sob quais limitações ele é requisitado a operar

Exemplos de Requisitos

- “O sistema deve ser capaz de debitar e creditar uma conta corrente”
- “O sistema deve ser capaz de realizar transferências bancárias do tipo DOC e TED”
- “O sistema deve suportar pelo menos 20 transações por segundo”
- “O sistema deve estar disponível, pelo menos, durante 10 horas por dia”

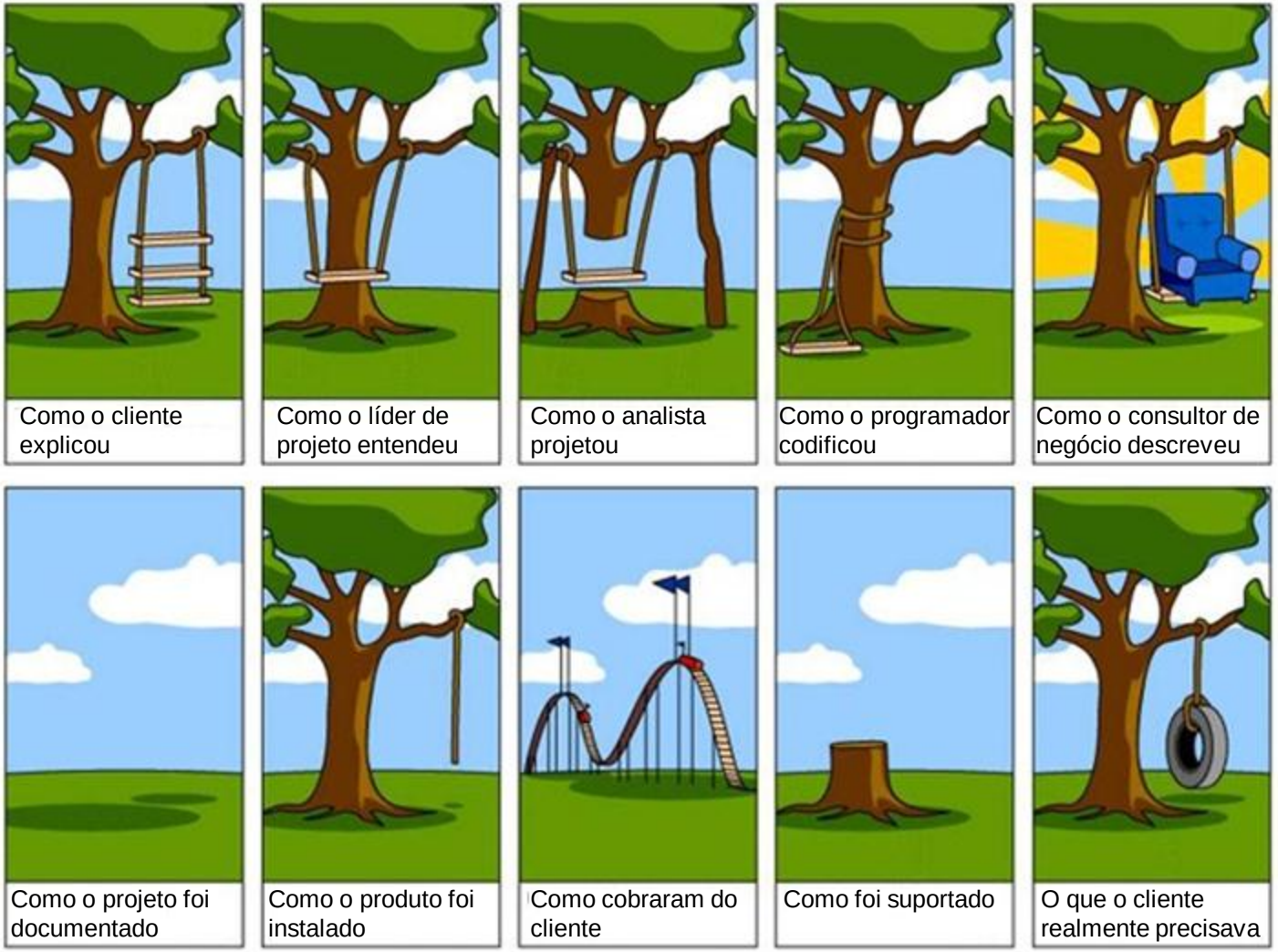
Partes interessadas (*stakeholders*)

- Engenheiros de SW responsáveis pelo desenvolvimento do sistema
- Usuários finais que irão usar o sistema depois de entregue
- Especialistas de domínio que possuem informações sobre os processos atuais
- Fiscais externos, que verificam se o sistema satisfaz os requisitos legais

Problemas com Requisitos...

- Virem de várias fontes
- Não refletirem as reais necessidades dos usuários do sistema
- Serem inconsistentes e/ou incompletos
- Podem ter um alto custo para mudanças, depois de acordados
- Mal entendidos entre clientes e desenvolvedores

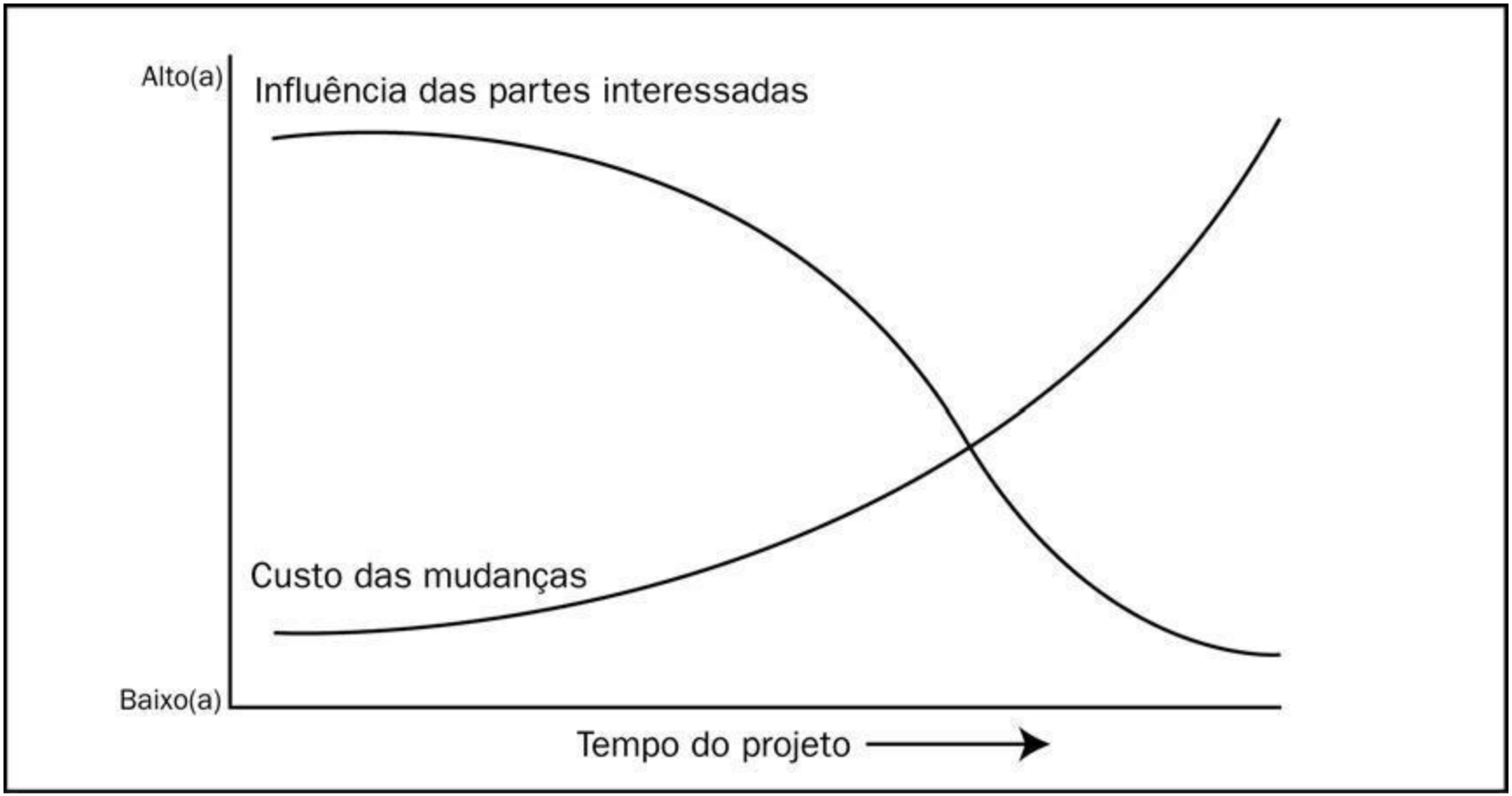
Problemas com Requisitos...



Importância da ER

- Quanto mais tarde problemas com requisitos são detectados, maior é o custo para corrigi-los
  - Pode custar até cinco vezes mais, caso o processo já esteja com ênfase na codificação ou até vinte vezes mais, caso esteja na manutenção
- **O sucesso das etapas posteriores depende da qualidade dos requisitos gerados**

Custo da mudança...



Exercícios [2]

INMETRO (CESPE 2010) 32 A engenharia de requisitos pode ser dividida em dois grupos de atividades: o desenvolvimento de requisitos e a gerência de requisitos. O desenvolvimento de requisitos inclui as seguintes etapas: elicitação de requisitos, análise e negociação de requisitos, especificação e modelagem de requisitos e validação de requisitos. A esse respeito, assinale a opção correta.

A) Nas atividades de desenvolvimento de requisitos para um sistema, deve-se tentar reduzir a participação efetiva dos usuários do sistema, visto que ela gera mais problemas que contribuições positivas.

B) Para a fase de especificação e modelagem de requisitos, a técnica mais recomendada é o JAD (joint application design), que, desenvolvido pela IBM, permite a criação de sistemas mais eficazes em menor tempo.

Exercícios [2]

C) A gerência de requisitos e o desenvolvimento de requisitos são atividades independentes uma da outra, por isso não é necessário haver interação das equipes que as realizam.

D) Atualmente, as empresas não têm tido dificuldade para implantar as atividades de desenvolvimento de requisitos e de gerência de requisitos. De fato, essas atividades estão plenamente implantadas na quase totalidade das organizações e empresas de software.

E) São atividades-chave para um gerenciamento de requisitos eficaz: analisar o problema, compreender as necessidades dos envolvidos, definir e refinar o escopo do sistema e gerenciar as mudanças de requisitos.

Tipos de Requisitos

- Funcionais
  - Definem funcionalidades do sistema
- Não Funcionais
  - Expressam restrições sob as quais o sistema deve operar ou qualidades específicas que o software deve ter
- De Domínio
  - Vêm do domínio da aplicação do sistema e refletem características do domínio

Tipos de Requisitos

- Requisitos permanentes (estáveis)
  - Derivados da atividade principal da organização. Exemplo: em um hospital sempre haverá requisitos relativos aos médicos, aos pacientes, aos tratamentos, etc. Derivados do modelo de domínio
- Requisitos Voláteis
  - Requisitos que se modificam durante o desenvolvimento ou quando o sistema está em uso. Exemplo: Requisitos resultantes de políticas governamentais

Tipos de Requisitos

Requisitos voláteis são divididos em:

- Requisitos Mutáveis
  - Se modificam por causa do ambiente do sistema
- Requisitos emergentes
  - Surgem à medida que a compreensão do cliente do sistema se desenvolve
- Requisitos conseqüentes
  - Resultam da introdução do sistema no ambiente do usuário
- Requisitos de compatibilidade
  - Dependem de outro equipamento ou processo. Conforme eles mudam, o requisito também muda

Requisitos Funcionais

- Descrevem funcionalidades ou serviços do sistema
- Dependem do tipo de software, dos usuários e do contexto onde ele será utilizado
- Podem ser escritos em alto nível, se forem voltados ao cliente, ou podem ser especificados em detalhe, para desenvolvedores

Requisitos Funcionais

Exemplos

- O sistema deve permitir cadastrar os dados pessoais dos clientes
- O sistema deve emitir relatórios gerenciais
- O sistema deve permitir a baixa automática de estoque quando da venda um produto

Requisitos não Funcionais

- Definem propriedades do sistema e restrições:
  - Usabilidade
  - Confiabilidade
  - Desempenho
  - Manutenibilidade
  - Escalabilidade
  - Portabilidade
  - ...

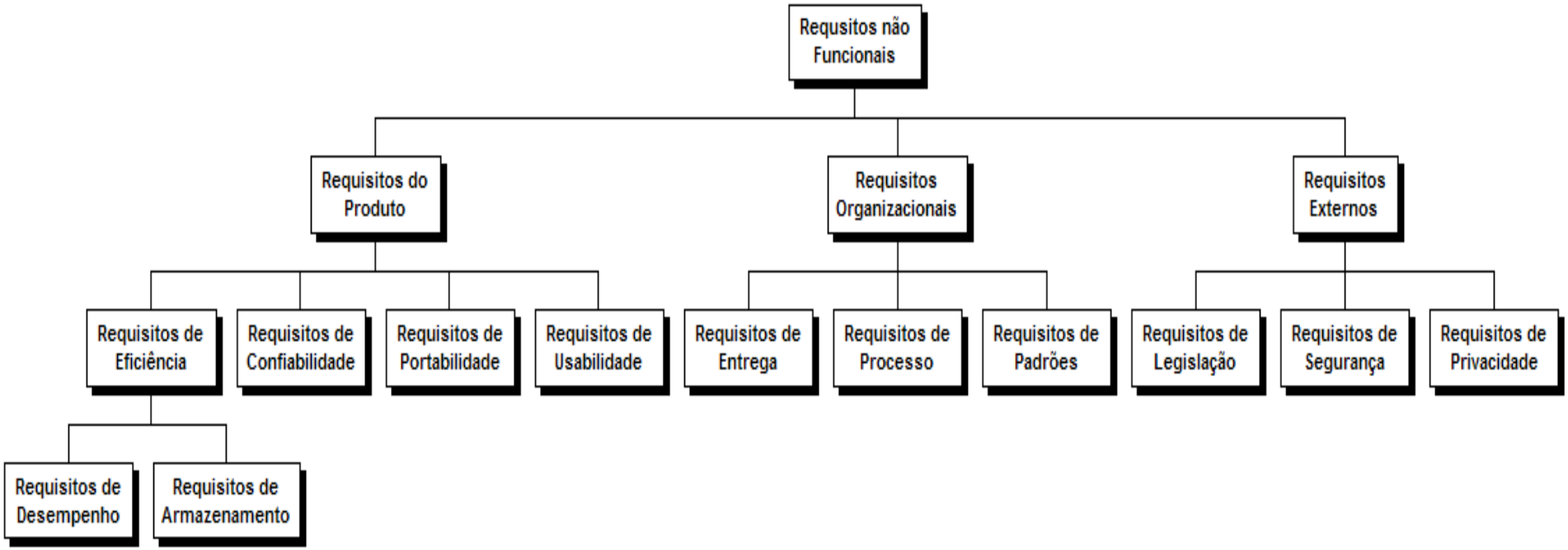
Requisitos não Funcionais

- Requisitos do **processo** também podem ser especificados obrigando o uso de uma determinada ferramenta CASE, linguagem de programação ou processo de desenvolvimento
- Requisitos não funcionais podem ser, e normalmente são, mais críticos do que os requisitos funcionais

Tipos de RNFs

- Requisitos do produto
  - Requisitos que especificam que o software entregue deve se comportar de um determinado modo, por ex.: ser confiável, robusto, rápido, etc.
- Requisitos organizacionais
  - Requisitos que são conseqüência das políticas e procedimentos organizacionais, como padrões, processos, etc.
- Requisitos externos
  - Requisitos que são externos ao sistema e seu desenvolvimento, ex: legislação, interoperabilidade, etc.

Tipos de RNFs



Problemas com RNFs

- Requisitos não funcionais podem ser extremamente difíceis de especificar precisamente
  - Como **verificamos** RNFs?
- Requisitos não funcionais devem ser verificáveis
  - Usando alguma medida que possa ser objetivamente testada
- O problema é que, muitas vezes, RNFs são conflitantes entre si!

Medindo RNFs

Propriedade	Medida
Desempenho	Transações por segundo; Tempo de resposta para eventos; etc.
Armazenamento	Megabytes; Número de chips ROM;
Usabilidade	Tempo de treinamento; Número de cliques de mouse;
Confiabilidade	Tempo médio entre falhas; Taxa de ocorrência de falhas; Disponibilidade;
Robustez	Tempo para recomeçar depois de uma falha; Probabilidade de corrupção de dados após falha;
Portabilidade	Porcentagem de declarações dependentes de plataforma; Número de plataformas-alvo

Requisitos de Domínio

- São derivados do Domínio da apli-cação e descrevem as características e funcionalidades do sistema que refletem o domínio em questão
- São transformados, posteriormente, em requisitos funcionais ou restrições (RNFs)
- Se não forem satisfeitos, também podem inviabilizar o funcionamento do sistema

Requisitos de Domínio

- Exemplo

  "A desaceleração do trem deve ser computada como: *Dtrem = Dcontrole + Dgradiente* onde *Dgradiente* = 9.81ms^2 vezes o gradiente compensado/alpha, onde os valores de 9.81ms^2/alpha variam de acordo com o tipo do trem"

- Que requisitos funcionais e não funcionais podemos derivar daí?

Problemas com Req. de Domínio

- São expressados na linguagem do domínio da aplicação
  - Alta chance do engenheiro de software não entender
- Conhecimento tácito
  - Os especialistas de domínio entendem tão bem da sua área que, muitas vezes, não pensam em tornar os requisitos de domínio explícitos

Exercícios [3]

(TRE/MT – CESPE 2010)

[31-a] O levantamento de requisitos tem como objetivo compreender o problema a ser resolvido e identificar necessidades. Os requisitos podem ser funcionais, que definem as funcionalidades do sistema, ou não funcionais, que não estão relacionados às funcionalidades.

(TRE/MT – CESPE 2010)

[33-a] Requisitos funcionais descrevem as propriedades emergentes do sistema, como segurança e tempo de resposta.

[33-b] Requisitos não funcionais são descritos de forma qualitativa e não quantitativa.

[33-c] Requisitos são provenientes de pessoas relevantes para o sistema, e não de outros sistemas que interagem com o sistema que está sendo especificado.

Exercícios [3]

(ANA – ESAF 2009)

12- Analise as seguintes afirmações sobre requisitos de sistemas de software:

I. Requisitos funcionais declaram as funções que o sistema deve fornecer, seu comportamento, e ainda, o que o sistema não deve fazer.

II. Requisitos de domínio são, exclusivamente, funcionais, pois exibem as características do domínio de aplicação do sistema.

III.Requisitos não-funcionais compreendem restrições sobre serviços ou funções do sistema.

Assinale a opção correta.

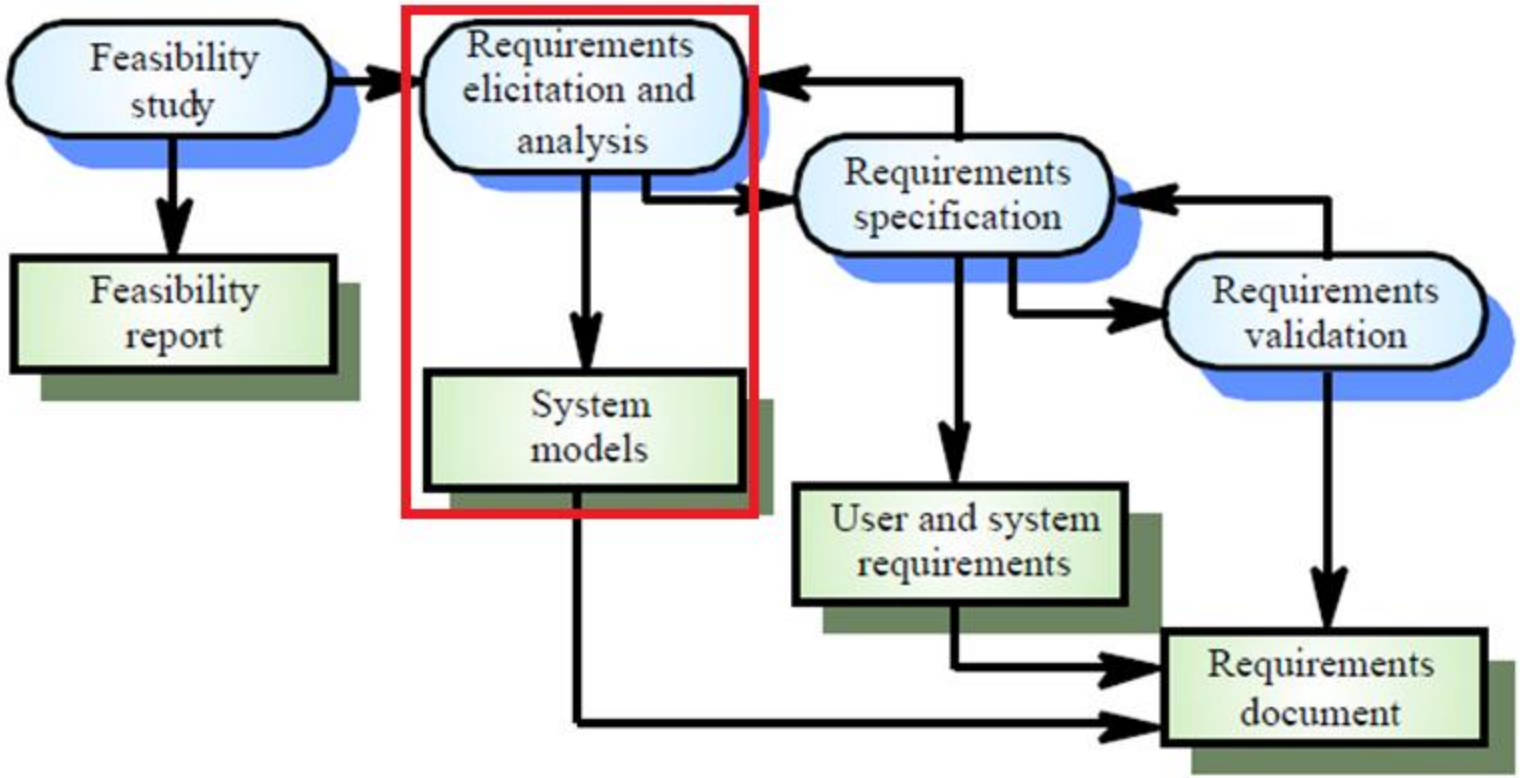
a) Apenas as afirmações I e II são verdadeiras.

b) Apenas as afirmações I e III são verdadeiras.

c) Apenas as afirmações II e III são verdadeiras.

d) As afirmações I, II e III são verdadeiras.

Elicitação e Análise de Requisitos



Elicitação: o que é?

- **Elicitar**: descobrir, tornar explícito, obter o máximo de informações para o conhecimento do objeto em questão
- A equipe técnica deve esclarecer:
  - O domínio da aplicação
  - Os serviços que a aplicação deve oferecer
  - As restrições sob as quais a aplicação deve operar
- Envolve vários *stakeholders*

Problemas

- Os interessados não sabem o que querem
- Os interessados descrevem os problemas em sua própria linguagem
- Os requisitos de cada parte interessada podem ser conflitantes
- Fatores políticos e organizacionais podem influenciar os requisitos do sistema

Atividades

- Entendimento do Domínio da Aplicação
  - Entender os problemas atuais na organização e como o software a ser implementado se ajustará a ela
- Descoberta (levantamento) dos Requisitos
  - Interagir com as partes interessadas para descobrir seus requisitos

Técnicas de Elicitação

- Entrevistas
- Questionários
- Leituras de documentos
- Observações e análises sociais (etnografia)
- Workshops de requisitos
- Cenários (Casos de Uso)
- Prototipagem
- ...

Técnicas de Elicitação: Etnografia

Etnografia

- Técnica de observação utilizada para compreender os requisitos sociais e organizacionais
- Um cientista social se insere no ambiente de trabalho onde o sistema será implantado e analisa como as pessoas trabalham
- As pessoas não precisam explicar o seu trabalho
- Fatores sociais e organizacionais importantes podem ser observados

Escopo da Etnografia

- Requisitos que são derivados da forma como as pessoas trabalham, e não de como os desenvolvedores **pensam** que os processos funcionam
- Requisitos que são derivados da cooperação e compreensão das atividades das outras pessoas

Técnicas de Elicitação: Workshop de requisitos

Workshop de requisitos

- Põe todos os *stakeholders* juntos por um período intensivo (focado)
- O **facilitador** de um workshop é o responsável pelas atividades logísticas e de organização, que inclui:
  - Dar a todos a oportunidade de falar
  - Manter a sessão sobre controle
  - Reunir informações para atributos de requisitos aplicáveis
  - Registrar as descobertas
  - Resumir a sessão e elaborar conclusões

Workshop de requisitos

- Durante o Workshop, outras técnicas de identificação podem ser utilizadas
  - Brainstorming
  - Interpretação de papéis
  - Revisão de requisitos existentes, etc.
- Ao final do workshop, o facilitador resume as descobertas em um formato apresentável

Exercícios [4]

(SERPRO – CESPE 2010)

[66] A entrevista é uma técnica de elicitação de requisitos simples, eficiente e direta, e por esses motivos, pode ser usada como fonte exclusiva de informação acerca dos requisitos do sistema.

(TCE/RN – CESPE 2009)

[51] A etnografia é uma técnica utilizada para a descoberta de requisitos de sistemas de software na qual, por meio de observações, procura-se compreender os requisitos sociais e organizacionais do ambiente onde o sistema será usado.

Exercícios [4]

(Governo do ES – CESPE 2009)

[71] Durante a elicitação de requisitos de um projeto pode ser empregada uma técnica denominada workshop, na qual os principais stakeholders de um projeto são reunidos por um curto período de tempo. Essa técnica prevê a existência de um facilitador, que deve ser um dos stakeholders e não deve interferir nas decisões do grupo ou emitir opiniões.

Técnicas de Elicitação: Casos de Uso

Casos de Uso

- Desenvolvidos por I. Jacobson, são parte integrante da UML
- São descrições textuais das funcionalidades do sistema a partir da perspectiva do usuário
- Usados para mostrar quais funcionalidades o sistema oferece e que usuários se comunicam com ele

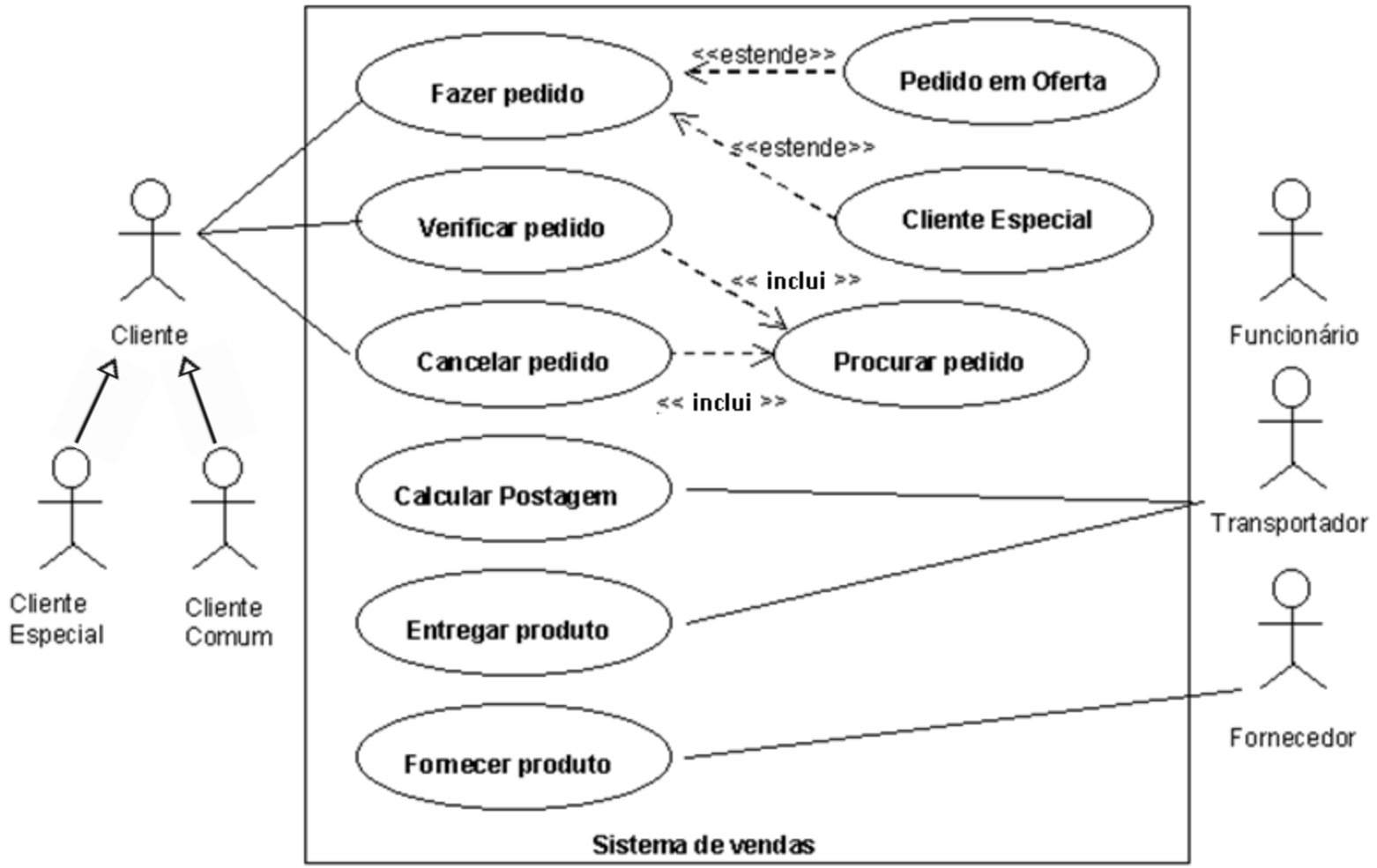
Quem usa os Casos de Uso?

- Clientes e Usuários
- Arquitetos
- Analistas, Projetistas e Implementadores
- Testadores
- Gerentes
- Escritores da documentação
- ...

Diagramas de Caso de Uso

- Servem para facilitar o entendimento de um sistema mostrando a sua “visão externa”
- São usados para modelar o contexto de um sistema ou um subsistema
- São uma das maneiras mais comuns de documentar requisitos do sistema
  - Delimitam o seu escopo
  - Definem suas funcionalidades

Exemplo: Sistema de Vendas



Especificação do Caso de Uso

Nome: Fazer Pedido
Descrição: Caso de uso que especifica o fluxo de ações para o cliente fazer um pedido no Sistema
Atores: Cliente
Pré Condição: O cliente deve estar logado no sistema

Especificação do Caso de Uso

Fluxo Principal de Eventos:

P1. O caso de uso começa quando o cliente seleciona a opção “Fazer Pedido”

P2. O cliente fornece seu nome e endereço e fornece o código do produto **[EXT1]**

P3. O sistema fornece a descrição e o preço do produto **[INC1]**

P4. O cliente fornece as informações de pagamento e escolhe a opção “confirmar” **[A1]**

P5. O sistema verifica as informações fornecidas e envia os dados para o sistema de pagamento **[E1]**

P6. O caso de uso é encerrado

Especificação do Caso de Uso

Pontos de Extensão

EXT1. O sistema estende o caso de uso “Pedido em Oferta”

Pontos de Inclusão

INC1. O sistema inclui o caso de uso “Dar informação do produto”

Fluxo Alternativo de Eventos

A1. No passo P4 cliente seleciona a opção “cancelar”

A1.1 O sistema não grava o pedido e o fluxo retorna para o passo **P6**

Especificação do Caso de Uso

Fluxo Excepcional de Eventos

E1. No passo **P5** o sistema verifica que as informações fornecidas estão incorretas

E1.1 O sistema pede ao cliente para corrigir as informações e o fluxo retorna ao passo **P4**

Pós Condições

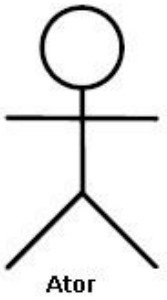
O pedido deve ter sido gravado no sistema e marcado como confirmado

Especificação do Caso de Uso

- O Caso de Uso pode conter outros dados, como:
  - Requisitos não Funcionais relacionados
  - Diagrama de atividades relacionado
  - Protótipo de interface
  - Outros diagramas
  - ...
- O importante é que as necessidades sejam entendidas e acordadas por todas as partes interessadas!

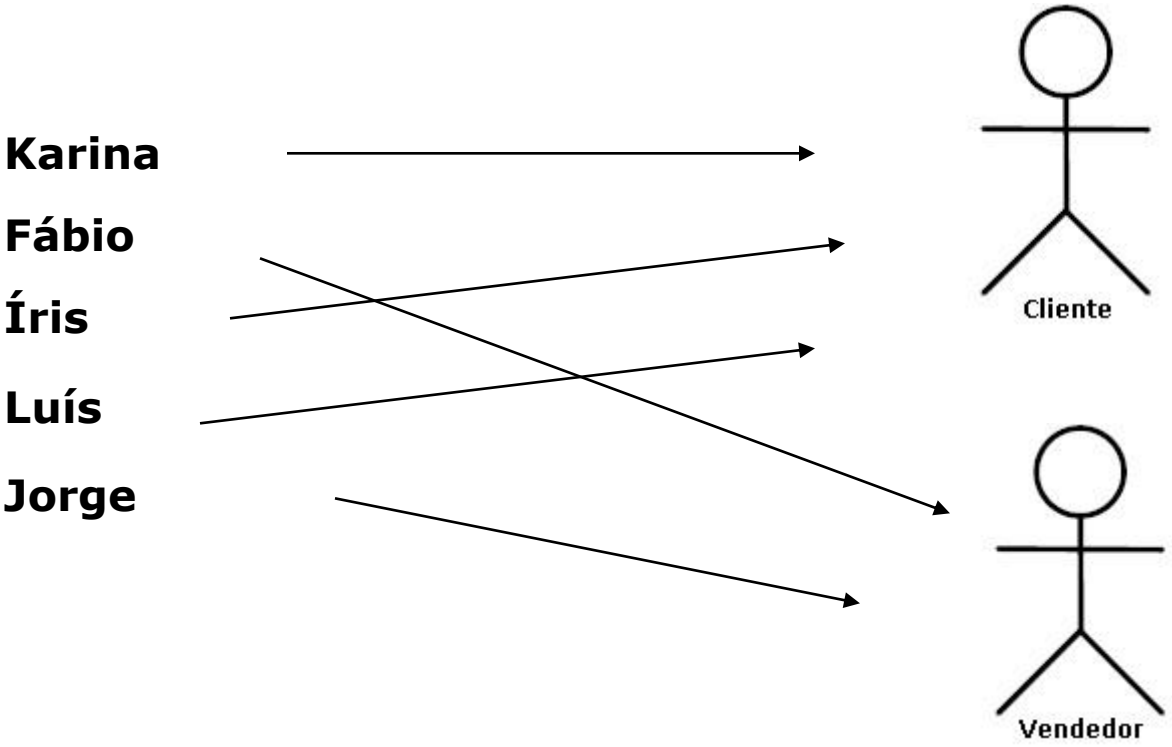
Atores

- Casos de uso são executados por **atores**
- Eles constituem as entidades externas do ambiente do sistema
- São papéis que os usuários do sistema devem desempenhar nas interações
- Uma “instância de ator” pode ser desempenhada tanto por um indivíduo quanto por um sistema ou mesmo por um dispositivo



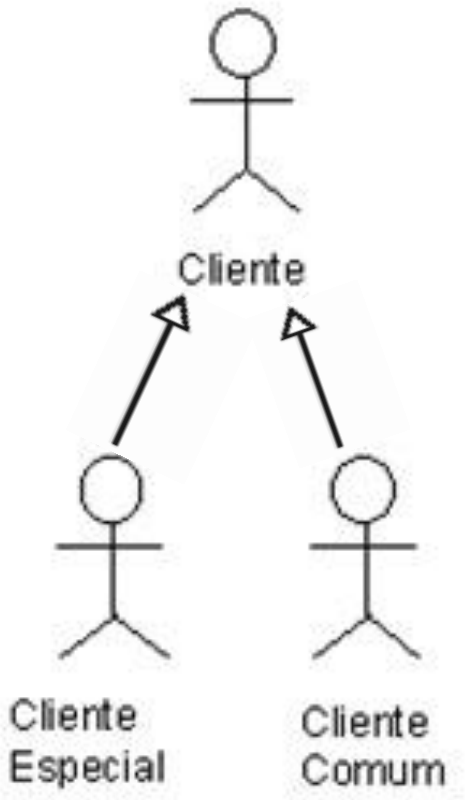
Atores

- Lembre-se, Atores representam **papéis/perfis** e não pessoas



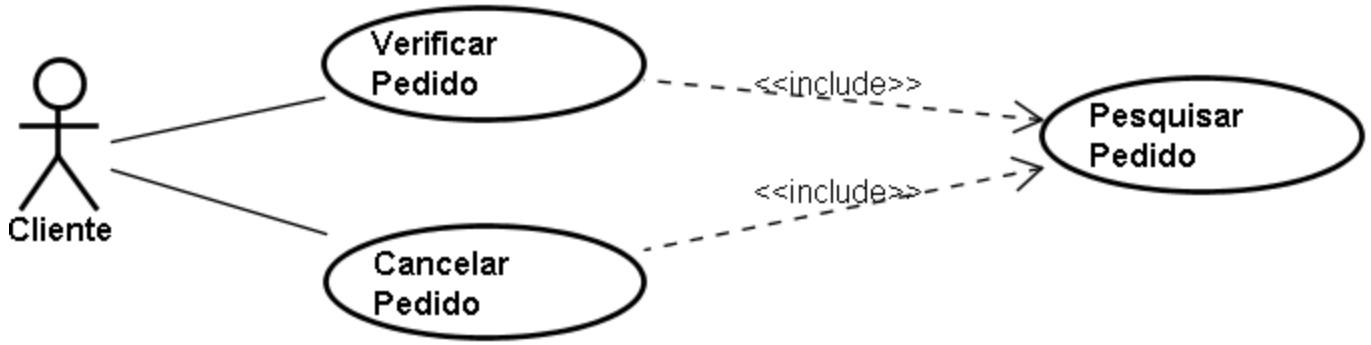
Atores: Especialização

É possível definir tipos gerais de atores e especializá-los usando o relacionamento de especialização



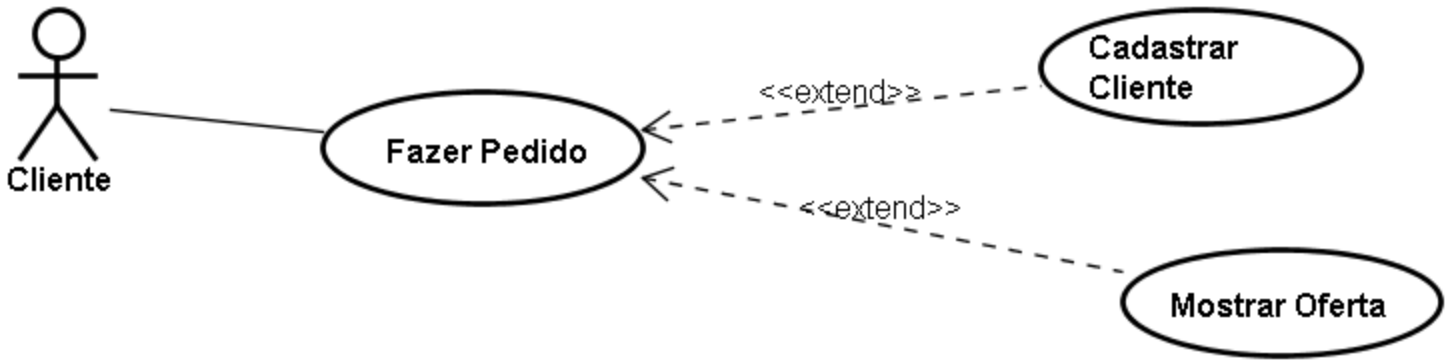
Relacionamento: Inclusão

- Um Caso de Uso base incorpora o comportamento de outro Caso de Uso
- O relacionamento é utilizado para evitar a descrição do mesmo fluxo de eventos várias vezes



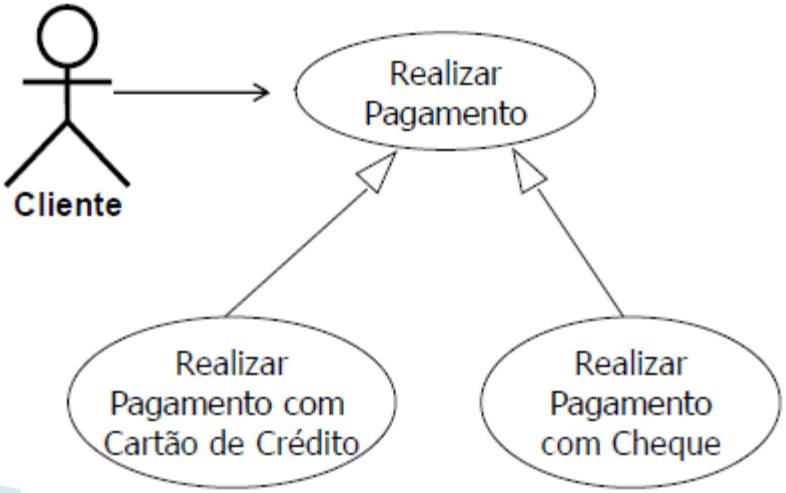
Relacionamento: Extensão

- Modela partes opcionais da execução de um Caso de Uso
- Modela fluxos que são executados somente em determinados casos, sob determinadas circunstâncias ou que dependem de escolha de um ator



Relacionamento: Generalização

- Relaciona um Caso de Uso especializado a um mais geral
- O filho herda o comportamento do pai, podendo adicionar e redefinir passos em pontos arbitrários do comportamento original



Resumo dos Relacionamentos

- Inclusão
  - Use quando o mesmo comportamento se repete em mais de um Caso de Uso e o processo de realizar X **sempre** envolve realizar Y pelo menos uma vez
- Extensão
  - Use quando você quiser modelar um comportamento **opcional** de um Caso de Uso

Resumo dos Relacionamentos

- Generalização entre Casos de Uso
  - Use quando você identificar Casos de Uso semelhantes e um deles for uma forma especial (uma especialização) do outro
- Generalização entre Atores
  - Use quando um ator (filho) é um **tipo de** outro ator mais genérico (pai)

Tipos de Casos de Uso

- Concreto
  - É iniciado por um ator e constitui um fluxo completo de eventos
- Abstrato: nunca é instanciado diretamente
  - Casos de Uso abstratos geralmente são:
    - Incluídos em outros Casos de Uso
    - Estendidos de outros Casos de Uso
    - Generalizações de outros Casos de Uso
- Atores “enxergam” apenas casos de uso concretos

Modelo de Casos de Uso

- É um modelo completo das funções do sistema em termos de Casos de Uso
- A finalidade mais importante é comunicar, de forma fácil de entender, o comportamento do sistema ao usuário final
- Contém:
  - Casos de uso, Atores, Relacionamentos
  - Pacotes de Caso de uso, Diagramas de Caso de Uso, Especificações, etc...

Benefícios

- Casos de Uso são focados no usuário do sistema, assim as *reais* necessidades são tratadas logo cedo
- São fáceis de entender
- Facilitam o acordo entre todas as partes interessadas
- Pode ser usado no levantamento, elicitação e validação dos requisitos, conectando todas as etapas

Exercícios [5]

(SERPRO – CESPE 2010)

[68] A descrição dos cenários de uso com informações acerca da utilização do sistema sob diversos pontos de vista e formas de operação deve fazer parte do levantamento dos requisitos.

(BASA – CESPE 2007)

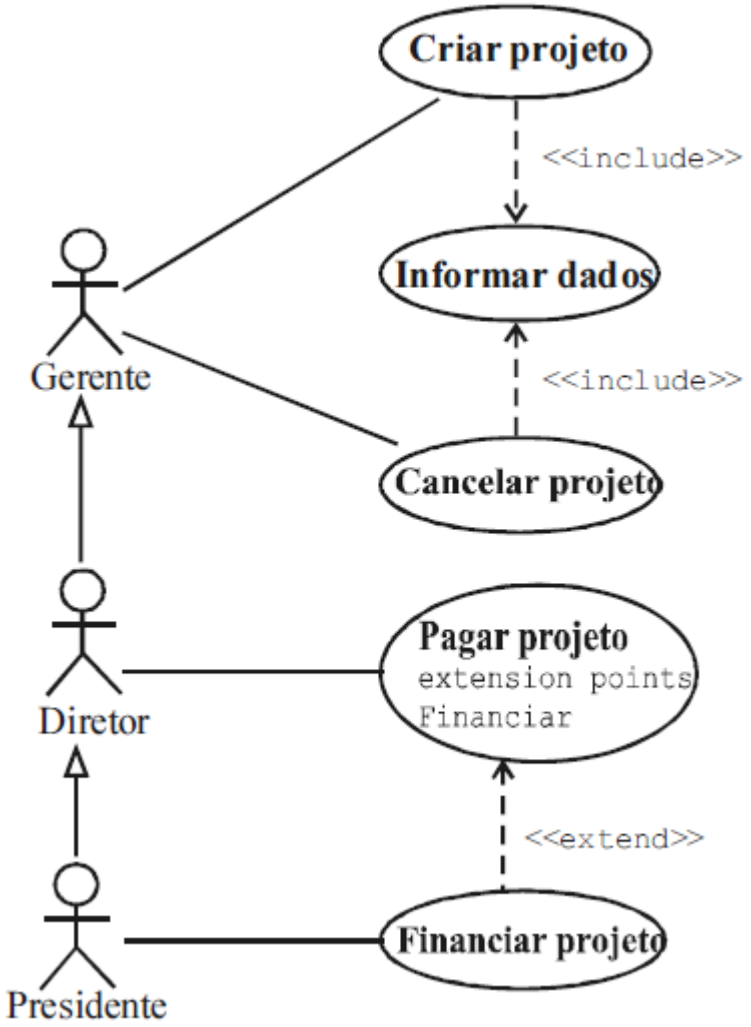
[66] A construção de um modelo de casos de uso é um meio para capturar requisitos funcionais com foco no valor dos requisitos para os usuários. Um caso de uso especifica uma seqüência de ações que o sistema pode realizar e que produzem resultados observáveis e de valor para os atores.

[67] Em um modelo de casos de uso, pode haver diferentes tipos de usuários representados por atores. Além de tipos de usuários, atores podem representar outros sistemas ou hardwares que interagem com o sistema a ser desenvolvido. Atores se comunicam com o sistema via casos de uso.

Exercícios [5]

(MPE/RR – CESPE 2008)

[87] No diagrama UML ao lado, o ator Presidente está relacionado ao caso de uso Criar projeto; o caso de uso Informar dados contém comportamento comum a dois casos de uso; o caso de uso Pagar projeto estende o comportamento Financiar projeto e Cancelar projeto é abstrato.



Análise de Requisitos

- Depois que os requisitos foram coletados, os produtos de trabalho servem como base para a análise de requisitos
- A análise de requisitos visa a descobrir alguns problemas e torná-los mais consistentes antes da especificação formal

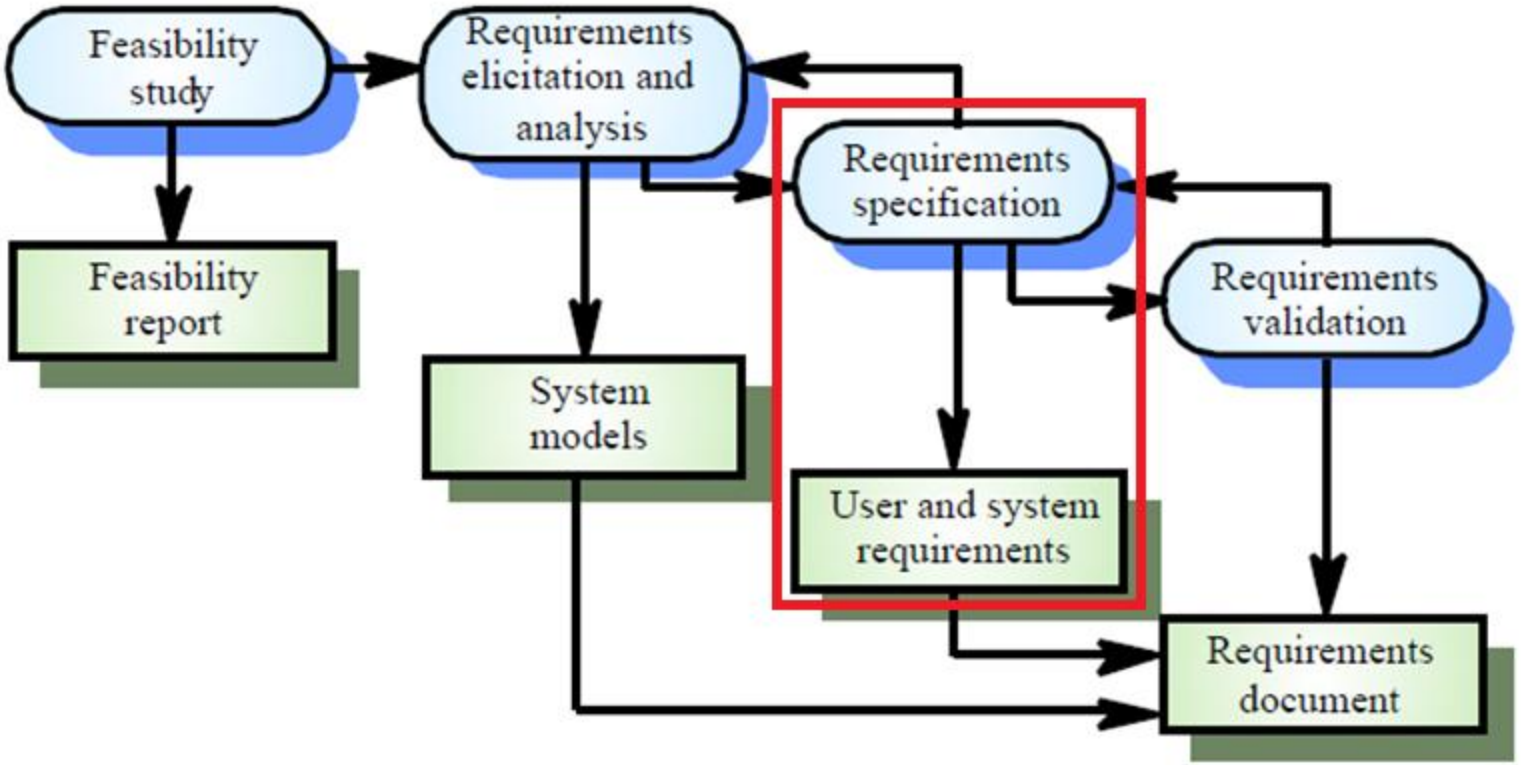
Atividades

- Classificação e organização
  - Agrupar requisitos relacionados e os organizá-los em conjuntos coerentes
- Checagens de:
  - Consistência
  - Ambiguidade
  - Omissões
  - Relacionamentos entre requisitos, etc.
- Priorização e negociação
  - Priorizar requisitos e negociar conflitos

Negociação

- A atividade de negociação é importante para que o analista possa conciliar os conflitos entre os *stakeholders*
  - Eles pedem mais do que pode ser feito
  - Ou têm "necessidades especiais"
- É papel do analista de requisitos balancear todas essas demandas
- Requer grande capacidade de interação social

Especificação de Requisitos



Especificação de Requisitos

- O termo especificação tem vários significados , podendo ser:
  - Um documento escrito
  - Um modelo gráfico
  - Um modelo matemático formal
  - Uma coleção de cenários de uso, etc.
- A abordagem utilizada depende da necessidade específica de cada projeto
  - Documentos escritos combinados com modelos gráficos para sistemas maiores
  - Cenários de uso para sistemas mais simples, etc.

Especificação do Sistema

- É o produto final produzido pelo engenheiro de requisitos
- Serve como a base para
  - Engenharia de Software
  - Engenharia de Hardware
  - Engenharia de Banco Dados, etc.
- Descreve a função de um sistema de software e as restrições impostas a ele
- Também descreve as informações que entram e saem do sistema

Exercícios [6]

(SERPRO – CESPE 2010)

[95] O documento de requisitos de software estabelece formalmente o que os desenvolvedores de sistema devem implementar e inclui a especificação resumida dos requisitos do sistema e a visão detalhada da arquitetura do sistema.

(MPOG – ESAF 2008)

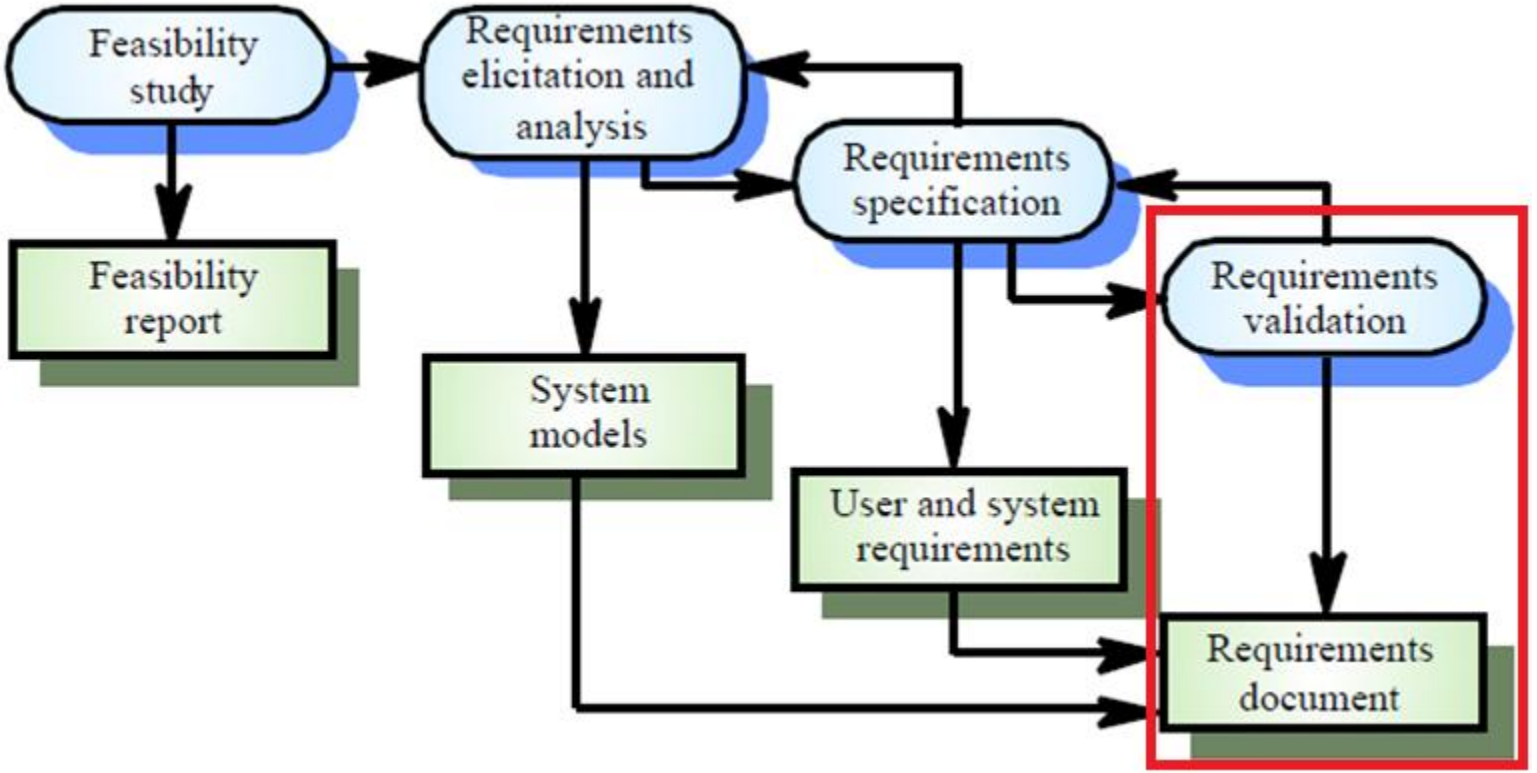
[16-e] Avaliar se os requisitos associados ao desempenho, ao comportamento e às características operacionais do sistema foram explicitamente declaradas é uma tarefa de especificação de requisitos.

Exercícios [6]

(ANATEL – CESPE 2009)

[76] A elicitação de requisitos ocorre usualmente antes da fase de análise de requisitos, e resulta na produção de uma especificação precisa das necessidades do usuário bem como dos requisitos do sistema a ser desenvolvido, o que exige maior interação social por parte do responsável pela elicitação, quando relacionada à exigência de interação durante a fase de análise.

Validação de Requisitos



Objetivos

- Demonstrar que os requisitos definem o sistema que o usuário realmente deseja
- Visa a assegurar que
  - A versão do documento de requisitos descreve as funcionalidades e características do sistema satisfatoriamente
  - Os requisitos são consistentes e de alta qualidade
  - O documento de requisitos provê uma base adequada para Projeto e Implementação

Entradas e Saídas

Entradas

- O documento de requisitos (preliminar)
- Padrões organizacionais
- Conhecimento implícito da organização

Saídas

- Lista de problemas com os requisitos
- Ações acordadas para tratar destes problemas
- Documento de Requisitos aprovado (final)

Principais técnicas

- Revisões (inspeções)
  - Um grupo de pessoas se reúne, lê e analisa os requisitos, para identificar problemas e suas possíveis soluções
- Prototipagem
  - Um protótipo executável demonstra os requisitos e ajudam os *stakeholders* a descobrir problemas
- Geração de Casos de Teste
  - Casos de teste ajudam a mostrar se os requisitos estão ambíguos ou incompletos

Exercícios [7]

(TRE/MT – CESPE 2010)

[33-e] Revisão de requisitos, prototipação e geração de casos de teste são exemplos de técnicas de validação de requisitos.

(MPE/AM – CESPE 2008)

[58] Para validar os requisitos de um sistema, é melhor realizar apenas uma revista técnica formal no final da especificação, pois assim todos os requisitos são analisados de uma única vez.

(MPE/AM – CESPE 2008)

[56] Uma das formas de resolução de ambigüidades de requisitos consiste em realizar a prototipação de partes do sistema, antes de se adotar uma solução.

Gerenciamento de Requisitos

Gerenciamento de Requisitos

- É o processo de gerenciar as mudanças nos requisitos durante o processo de Engenharia de Requisitos
- Requisitos são, inevitavelmente, incompletos e inconsistentes
  - Novos requisitos surgem à medida que as necessidades de negócios mudam e há um melhor entendimento do sistema
  - Diferentes pontos de vista normalmente têm requisitos diferentes (e conflitantes)

Requisitos sempre mudam!

- A prioridade de cada requisito muda ao longo do projeto
- Cliente não é a mesma coisa que Usuário
  - Perspectivas diferentes
- O ambiente de negócios e tecnológico do projeto muda durante o seu desenrolar
- **É necessário gerenciar tudo isso**

Rastreabilidade

- Relacionam os requisitos e avaliam seus impactos
- Rastreabilidade de **Fonte**
  - Ligação entre o requisito e o *stakeholder* que o propôs (e sua necessidade original)
- Rastreabilidade de **Requisitos**
  - Ligações entre requisitos que dependem entre si
- Rastreabilidade de **Projeto**
  - Ligação entre o requisito e o projeto (arquitetura, módulos, código) do software

Ferramentas

- É impossível rastrear requisitos sem uma ferramenta CASE adequada
- Ela deve:
  - Armazenar os requisitos em um ambiente seguro e gerenciado
  - Dar suporte ao gerenciamento de mudança dos requisitos
  - Permitir recuperar automaticamente a ligação (rastreabilidade) dos requisitos

Exercícios [8]

(IJSN – CESPE 2010)

[64] A rastreabilidade de requisitos é essencial para que o controle de mudanças possa avaliar o impacto de uma solicitação de Mudança.

(TRE/MT – CESPE 2010)

[33-d] A matriz de rastreabilidade não oferece suporte para requisitos funcionais.

(SERPRO – CESPE 2008)

[92] A gerência de requisitos tem como objetivo principal controlar a evolução dos requisitos, seja por constatação de novas necessidades, seja por constatação de deficiências nos requisitos registrados até o momento. Um exemplo de gerência de requisitos é a aplicação de revisão por pares, que constata deficiências nos requisitos especificados.

Exercícios [8]

(Governo do ES – CESPE 2009)

[72] O gerenciamento de requisitos deve compreender e controlar mudanças nos requisitos de sistema, além de avaliar os seus impactos. Para atingir esse propósito, podem ser mantidas informações de rastreabilidade a serem usadas para avaliar quais outros requisitos seriam afetados por uma mudança, bem como o impacto da mudança de requisitos no projeto e na implementação do sistema.

Gabaritos dos Exercícios

- [1] – [35-a] E, [78] C, [79] E, [65] C
- [2] – [32] E
- [3] – [31-a] E, [33-a] E, [33-b] E, [33-c] E, [12] B
- [4] – [66] E, [51] C, [71] E
- [5] – [68] C, [66] C, [67] C, [87] E
- [6] – [95] E, [16-e] E, [76] E
- [7] – [33-e] C, [58] E, [56] C
- [8] – [64] C, [33-d] E, [92] E, [72] C

FIM