Engenharia de Software

Conceitos e Metodologias para Desenvolvimento de Software

Cascata, Prototipação, Espiral e RUP

Prof. MSc. Edilberto Silva

Adaptações Prof. Clenio Emidio

Alguns conceitos

- **u.su.á.rio**
- adj. 1. Que, por direito proveniente de uso, frui as utilidades da coisa. 2. Que serve para nosso uso. S. m. Aquele que, por direito de uso, frui as utilidades da coisa.
- **cli.en.te**
- s. m. 1. Pessoa protegida. 2. Pessoa que recorre a um médico, a um advogado, a um dentista etc., habitualmente. 3. Freguês.

O que é análise?

- Decomposição em elementos constituintes;
- Decomposição de um todo em suas partes constituintes;
- Exame de cada parte do todo.

O que é Sistemas?

- Um grupo de itens que interagem entre si ou que sejam interdependentes formando um todo unificado.

O que é Análise de sistemas?

- Estudo de métodos (“ maneiras de fazer”) utilizáveis nas atividades organizacionais relacionadas com a adoção de **tecnologias de informação** para suporte da organização (e em especial do seu **sistema de informação**)

Habilidades do Analista/Engenheiro

Concentração
Criatividade
Iniciativa
Persuasão
Comunicativo
Autoconfiança
Simplicidade
Ação Conciliadora
Clareza de Raciocínio
Espírito de Grupo
Percepção
Sensibilidade
Persistência
Flexibilidade
Determinação

O que é Engenharia de software?

- Engenharia de Software é o campo da engenharia que se concentra no desenvolvimento, manutenção e gerenciamento de sistemas de software. Envolve a aplicação de princípios científicos e tecnológicos para projetar, construir, testar e melhorar software, garantindo sua eficiência, segurança e qualidade.
- Os engenheiros de software trabalham em equipes para criar aplicativos, sistemas operacionais e plataformas que atendem a necessidades específicas, usando práticas como programação, modelagem e gerenciamento de projetos para otimizar processos e entregar soluções inovadoras.

- https://www.napratica.org.br/profissao-engenheiro-de-software/

Atribuições do Engenheiro

- Entre as principais atribuições do engenheiro de software, estão:

1. **Desenvolver softwares e apps**
2. **Gerenciar projetos ligados aos softwares**
3. **Arquitetar o design estrutural dos programas**
4. **Realizar testes nos sistemas**

- Desde maio de 2018, a profissão de Engenheiro de software é regulamentada. Atualmente, para ser registrado junto ao CREA (Conselho Regional de Engenharia e Agronomia) é obrigatório ter formação em Engenharia de Software.

Considerações

- O engenheiro de software é uma das profissões do futuro que mais crescem no Brasil e no mundo. Ele é responsável por desenvolver, gerenciar e testar novos softwares, aplicativos, programas e até jogos.
- Em um cenário em que as empresas buscam cada vez mais a tecnologia para facilitar e gerenciar processos, a mão de obra destes profissionais se torna ainda mais necessária. Os softwares são responsáveis por direcionar e programar ações em celulares, computadores, tablets e outros dispositivos inteligentes.

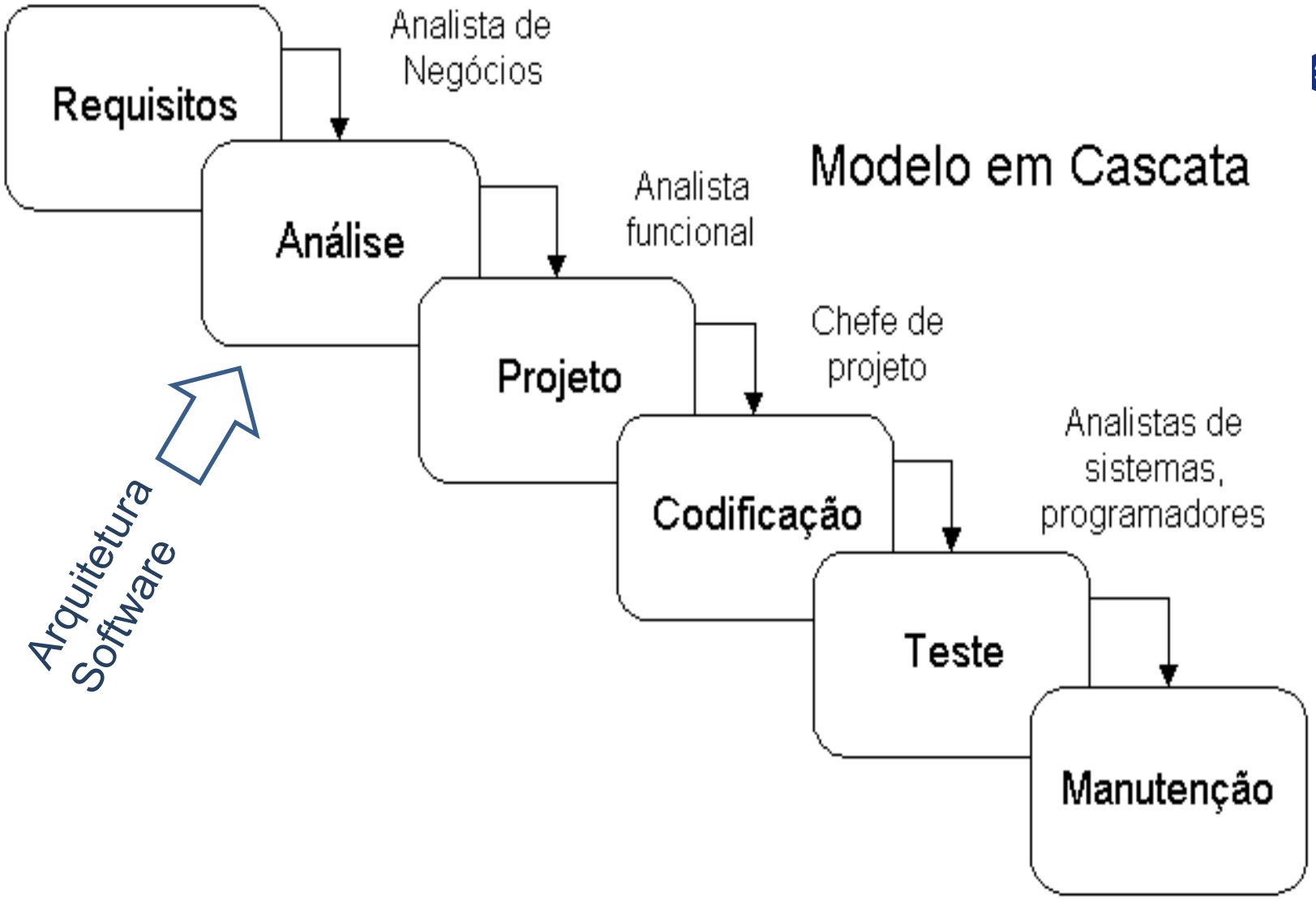
- A **engenharia de software** é uma área profissional em ascensão tanto no País quanto internacionalmente. Com investimentos de cerca de 2,3% do PIB em 2019, o Brasil respondeu por 1,8% do mercado global e lidera na América Latina, representando 40,7% do setor.

- https://www.serasaexperian.com.br/carreiras/blog-carreiras/engenheiro-de-software/

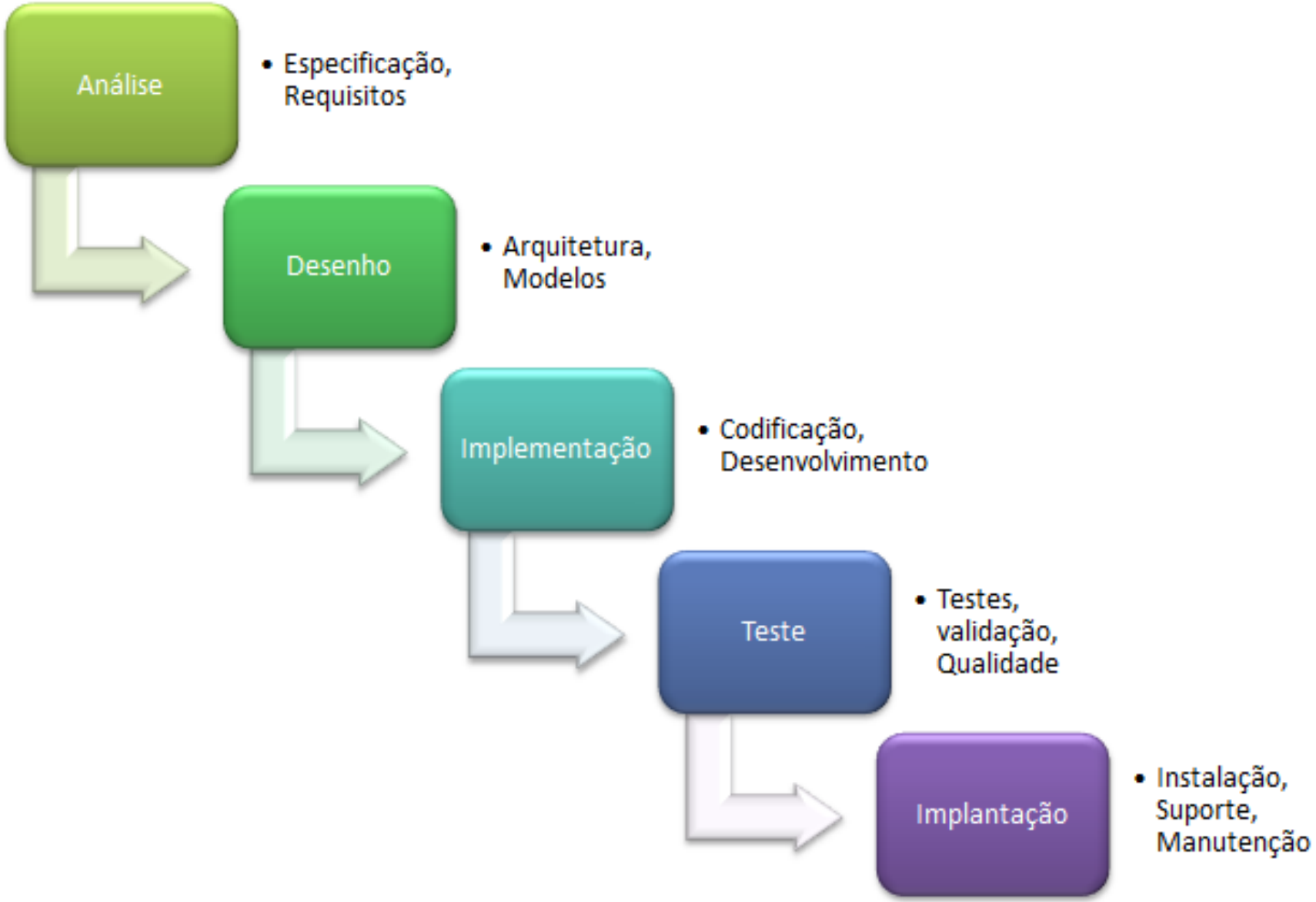
Paradigmas de Engenharia de Software

- Principais paradigmas:
  - Cascata.
  - Prototipação.
  - Espiral
  - Fases Genéricas [Pressman]
  - RUP - Rational Unified Process
- Ágeis
  - XP eXtreme Programing
  - OpenUP
  - SCRUM

Modelo Cascata



Cascata


Análise
• Especificação, Requisitos
Desenho
• Arquitetura, Modelos
Implementação
• Codificação, Desenvolvimento
Teste
• Testes, validação, Qualidade
Implantação
• Instalação, Suporte, Manutenção

Etapas do Modelo Cascata

- Requisitos
  - Coleta os requisitos do software.
  - Gera o documento de especificação do sistema que serve de base para o orçamento, cronograma, esforço, ferramentas a serem utilizadas, etc.
- Análise
  - Compreensão clara e precisa do domínio do problema e das funcionalidades do software.
  - Levantamento e revisão em conjunto com representantes do cliente, usuários chaves e outros especialistas da área de aplicação

Etapas do Modelo Cascata

- Projeto
  - Concentra-se na definição das estruturas de dados, arquitetura do software, detalhes procedimentais e caracterização da interface.
- Codificação
  - Tradução do projeto para uma linguagem legível para a máquina.
  - Se o projeto for bem detalhado essa tarefa pode ser automatizada.

Etapas do Modelo Cascata

- Teste
  - Inicia-se logo após a geração do código.
  - Visa garantir que uma entrada do programa produz o resultado esperado.
- Manutenção
  - Ocorre em função de:
    - Ocorrência de erros
    - Adaptações para acomodar mudanças externas
    - Acréscimos funcionais
    - Problemas de desempenho

Modelo Cascata

Vantagens

- Padroniza os métodos para análise, projeto, codificação, testes e manutenção.
- Etapas semelhantes às etapas genéricas aplicáveis a todos os paradigmas.

Desvantagens

- Projetos reais raramente seguem o fluxo seqüencial que esse modelo propõe. Sempre ocorre alguma interação e/ou superposição.
- Dificilmente os clientes são capazes de relacionar todos os requisitos de uma só vez no início do projeto
- Maioria dos programas só estará disponível quando o cronograma já está bastante adiantado.
- Dificuldades para se introduzir alterações quando o processo está avançado.

Prototipação

- Criação de um modelo do software que será implementado.
- Facilita a definição dos objetivos do software.
- Retrata a interação entre o usuário e o software.

Etapas

Coleta e refinamento dos requisitos.

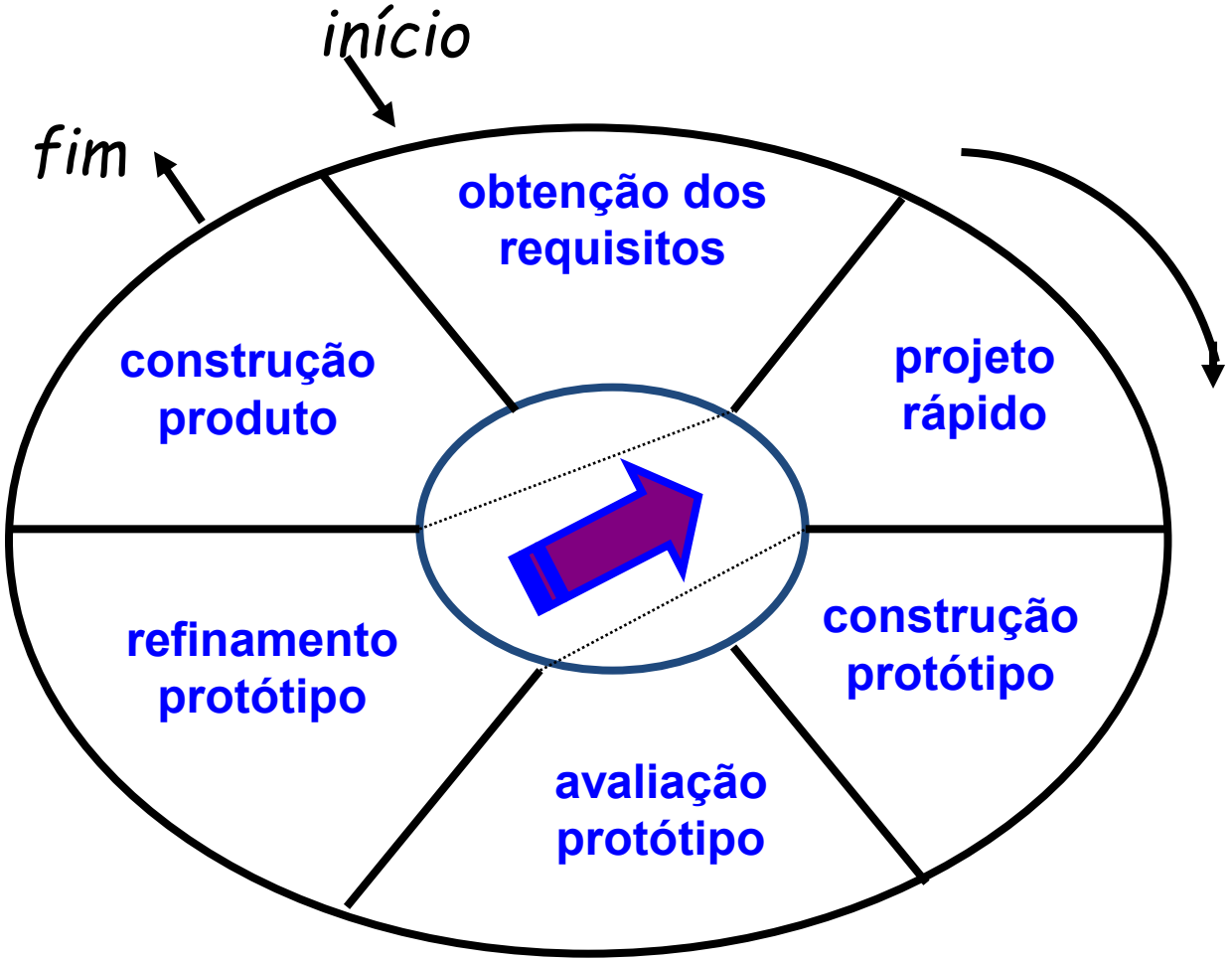
- Definição dos objetivos globais.

Projeto rápido.

- Identificação interações entre usuário e software.
- Concentra-se nas entradas e saídas do software.

Construção do protótipo.

Etapas da Prototipação



Modelo Prototipação

Vantagens

- Melhora a qualidade da especificação do software a ser desenvolvido, contribuindo para uma queda nos custos de desenvolvimento e manutenção.
- Antecipa o treinamento dos usuários.
- Partes do protótipo podem ser aproveitadas no desenvolvimento do sistema.

Desvantagens

- O custo na maioria dos casos é considerado muito alto.
- O cliente tende a confundir o protótipo com uma versão do sistema.

Ferramentas para Prototipagem

- **Ferramentas de Desenvolvimento Low-Code/No-Code**
  1. **Bubble**: Ferramenta no-code que permite criar aplicativos web interativos e funcionais sem precisar escrever código.
  2. **OutSystems**: Plataforma low-code que facilita a criação de aplicativos complexos com menos esforço de desenvolvimento.
  3. **Webflow**: Para prototipagem e desenvolvimento de sites, combina design visual com a capacidade de exportar código limpo.
  4. **Adalo**: Voltado para o desenvolvimento de aplicativos móveis, permite a criação de protótipos com funcionalidades completas.

Prototipagem

- **Ferramentas de Prototipagem Interativa**
  1. **InVision**: Permite transformar designs em protótipos interativos, simular navegação, e colaborar com feedback direto.
  2. **Marvel**: Ferramenta simples para criar protótipos interativos rapidamente a partir de esboços ou designs.
  3. **Proto.io**: Permite criar protótipos que imitam completamente a funcionalidade e aparência de um app finalizado, sem necessidade de programação.
  4. **Framer**: Oferece prototipagem interativa avançada com animações e transições detalhadas. Possui suporte a código para personalização mais profunda.

Prototipagem

- **Ferramentas de Prototipagem de Software Funcional**
  1. **Visual Studio Code**: Para quem deseja criar protótipos mais funcionais, é uma excelente escolha para desenvolvimento rápido com suporte a múltiplas linguagens.
  2. **Xcode**: Ferramenta oficial da Apple para prototipagem e desenvolvimento de apps iOS e macOS.
  3. **Android Studio**: Ferramenta oficial do Google para desenvolvimento e prototipagem de aplicativos Android

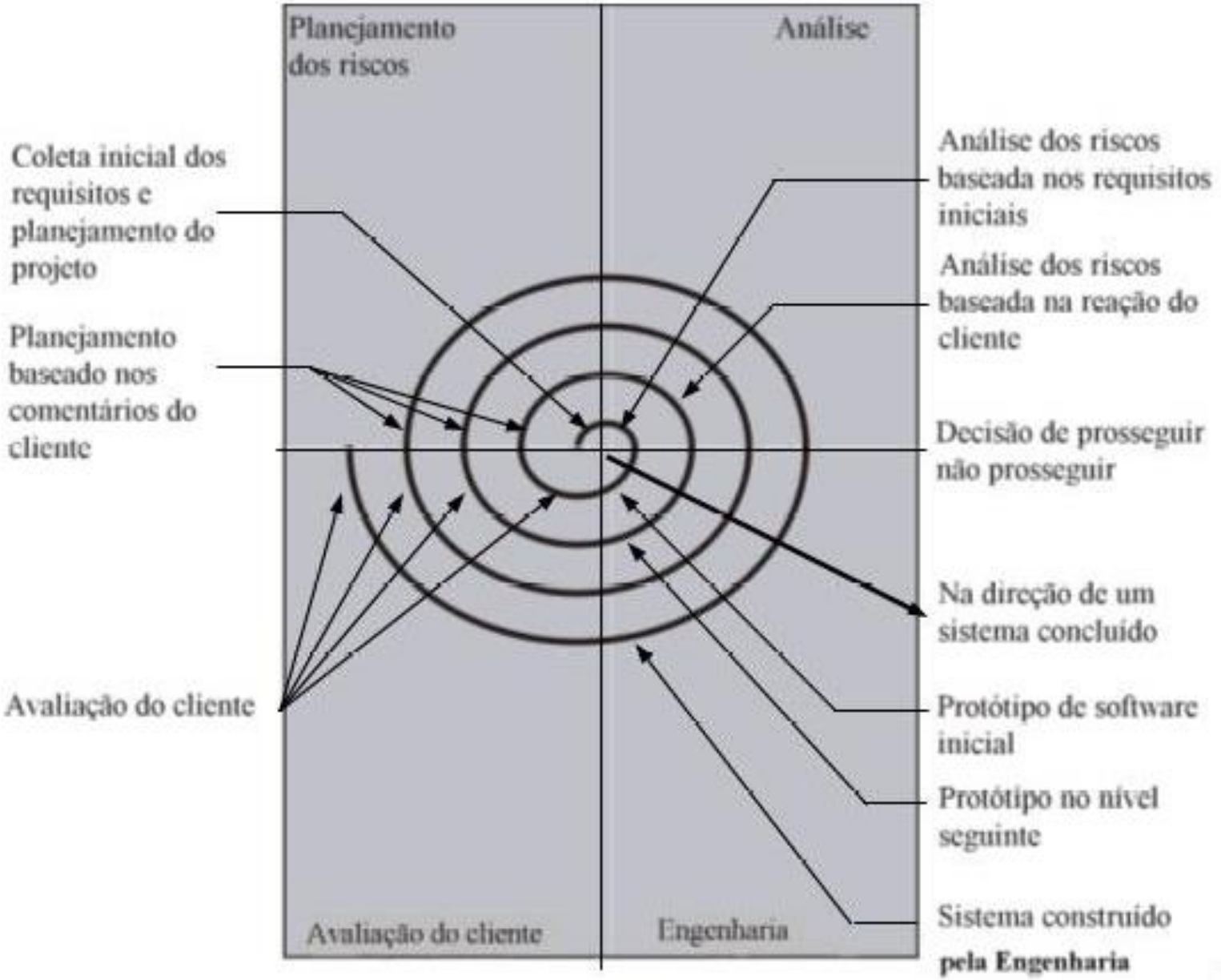
Outras Ferramentas

- Sketch;
- InVision;
- Adobe XD;
- Axure RP;
- JustinMind;
- UXPin;
- Framer;
- Origami;

1. Miro*
2. Figma*

Modelo Espiral

- Aproveita as melhores características do modelo cascata e da prototipação.
- Acrescentando um novo elemento: *a análise de riscos.*



Modelo Espiral

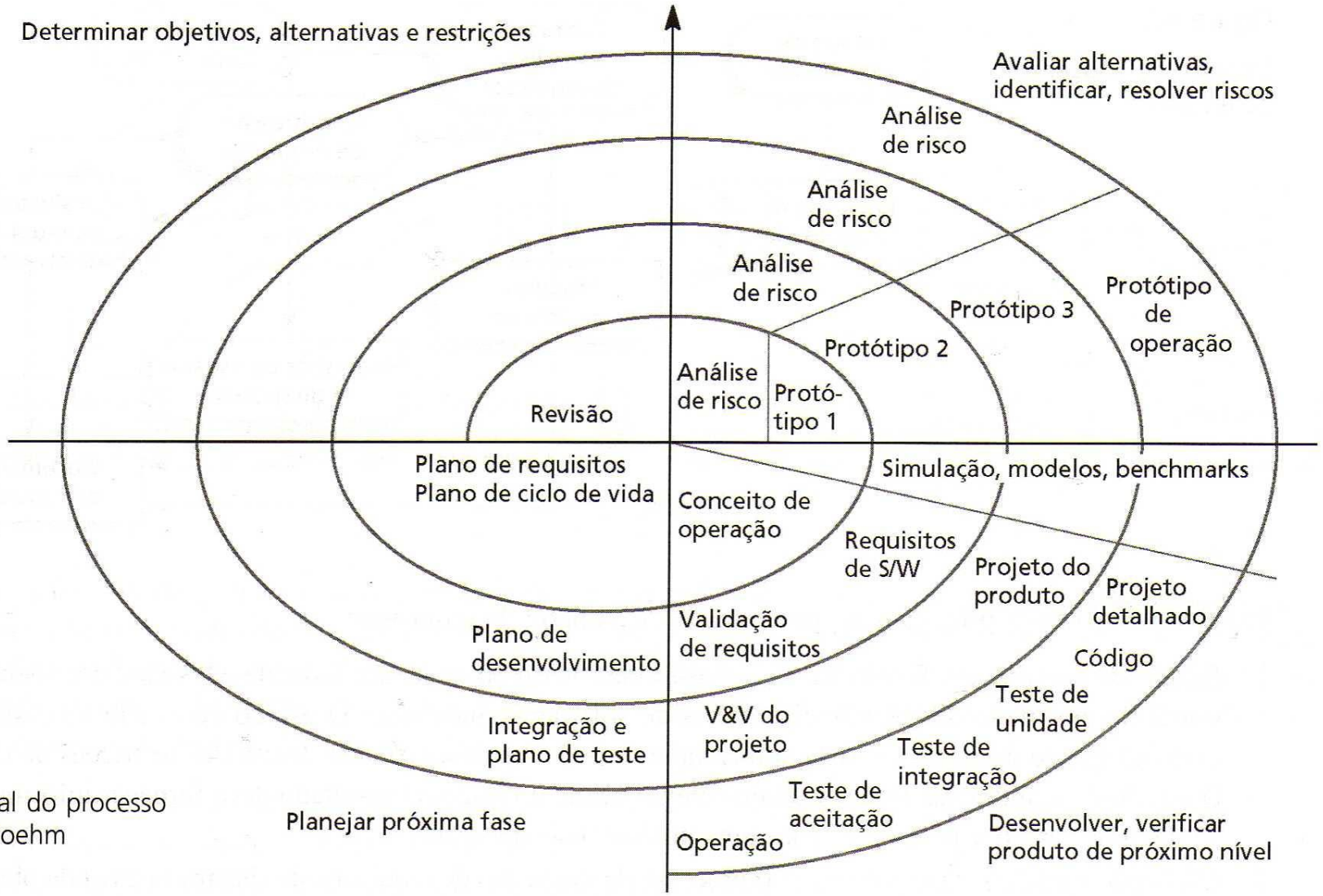
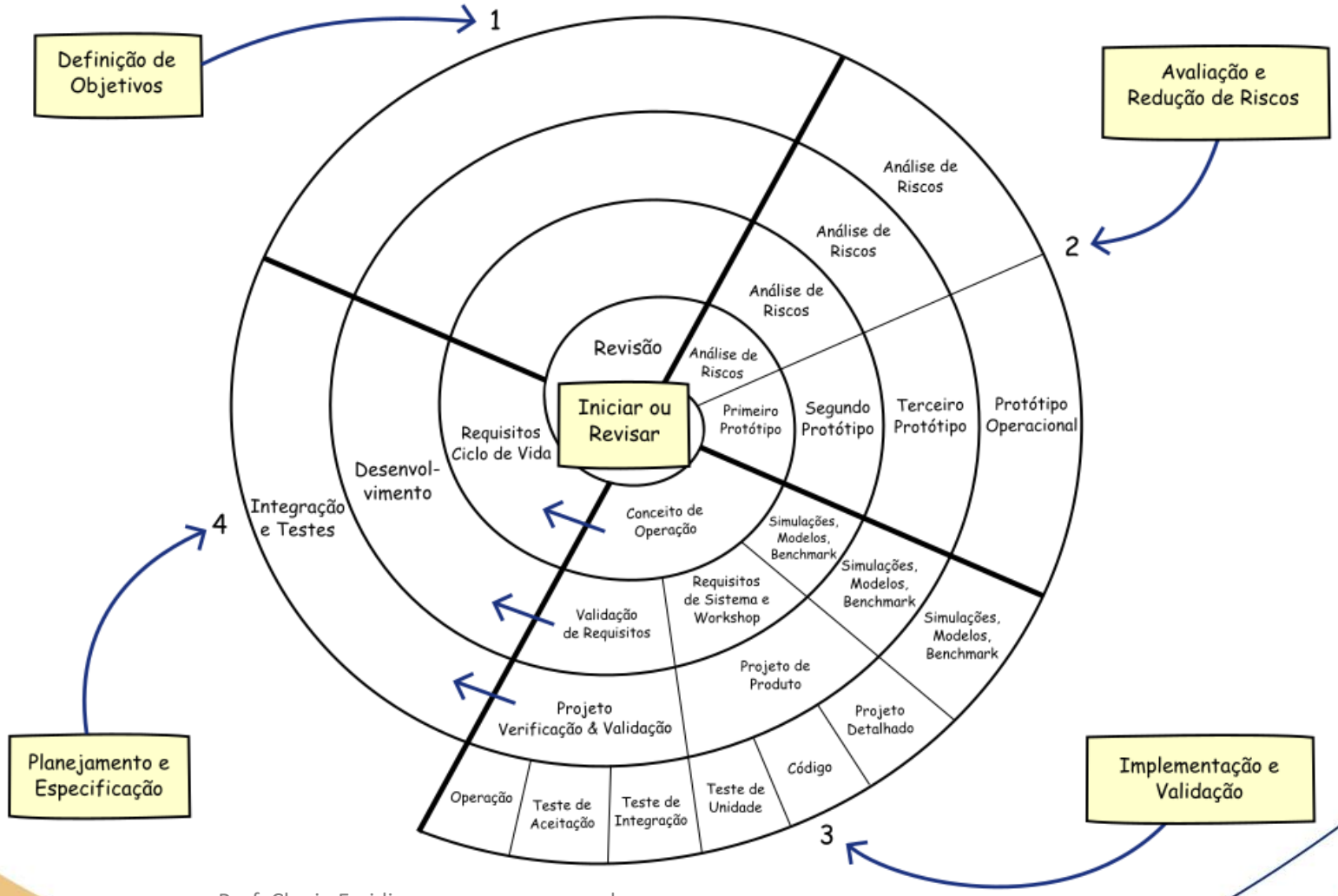


Figura 4.5

Modelo em espiral do processo de software de Boehm (©*IEEE*, 1988).


Definição de Objetivos
1
Avaliação e Redução de Riscos
2
Implementação e Validação
3
Planejamento e Especificação
4
Análise de Riscos
Análise de Riscos
Análise de Riscos
Análise de Riscos
Revisão
Iniciar ou Revisar
Primeiro Protótipo
Segundo Protótipo
Terceiro Protótipo
Protótipo Operacional
Requisitos Ciclo de Vida
Desenvol-vimento
Integração e Testes
Conceito de Operação
Simulações, Modelos, Benchmark
Simulações, Modelos, Benchmark
Simulações, Modelos, Benchmark
Requisitos de Sistema e Workshop
Validação de Requisitos
Projeto de Produto
Projeto Verificação & Validação
Projeto Detalhado
Código
Teste de Unidade
Teste de Integração
Teste de Aceitação
Operação

Etapas do Modelo Espiral

- Planejamento.
  - Definição dos objetivos, alternativas e restrições.
- Análise dos Riscos.
  - Análise de alternativas e identificação dos riscos sob o ponto de vista técnico e de gerência.
- Engenharia.
  - Desenvolvimento do produto.
- Avaliação do Cliente.
  - Avaliação dos resultados da engenharia.

Modelo Espiral

Vantagens

- Modelo evolutivo possibilita uma maior integração entre as fases e facilita a depuração e a manutenção do sistema.
- Permite que o projetista e cliente possa entender e reagir aos riscos em cada etapa evolutiva. Controle dos riscos.
- Fácil de ser alterado durante o desenvolvimento de acordo com as necessidades do cliente.

Desvantagens

- Processo bastante demorado por conta das iterações
- Avaliação dos riscos exige muita experiência.
- O modelo é relativamente novo e não tem sido amplamente utilizado.
- Não é utilizado para projetos pequenos por conta de sua complexidade.

Fases Genéricas: Definição

- Concentra-se no ***"quê"*.**
  - Que informações devem ser processadas?
  - Que funções devem ser desempenhadas?
  - Quais são as exigências fundamentais do software?
  - Quais são as restrições do projeto?

Etapas

- Análise do Sistema:
  - Define o papel que o software deverá desempenhar.
- Planejamento do Projeto de Software:
  - Analisa os riscos, aloca os recursos, estima os custos e define as tarefas que serão realizadas.
- Análise de Requisitos
  - Define detalhadamente o domínio da informação e a função do software.

Fases Genéricas: Desenvolvimento

- Concentra-se no ***"como"*.**
  - Como projetar as estruturas de dados e a arquitetura do software?
  - Como implementar os detalhes procedimentais?
  - Como traduzir o projeto para uma linguagem de programação?
  - Como realizar os testes?

Etapas

- Projeto de Software:
  - Traduz os requisitos do software em um conjunto de representações
- Codificação:
  - Converte as representações do projeto em uma linguagem artificial executada pelo computador.
- Realização de Testes de Software:
  - Procura por defeitos de função, lógica e implementação do software.

Fases Genéricas: Manutenção

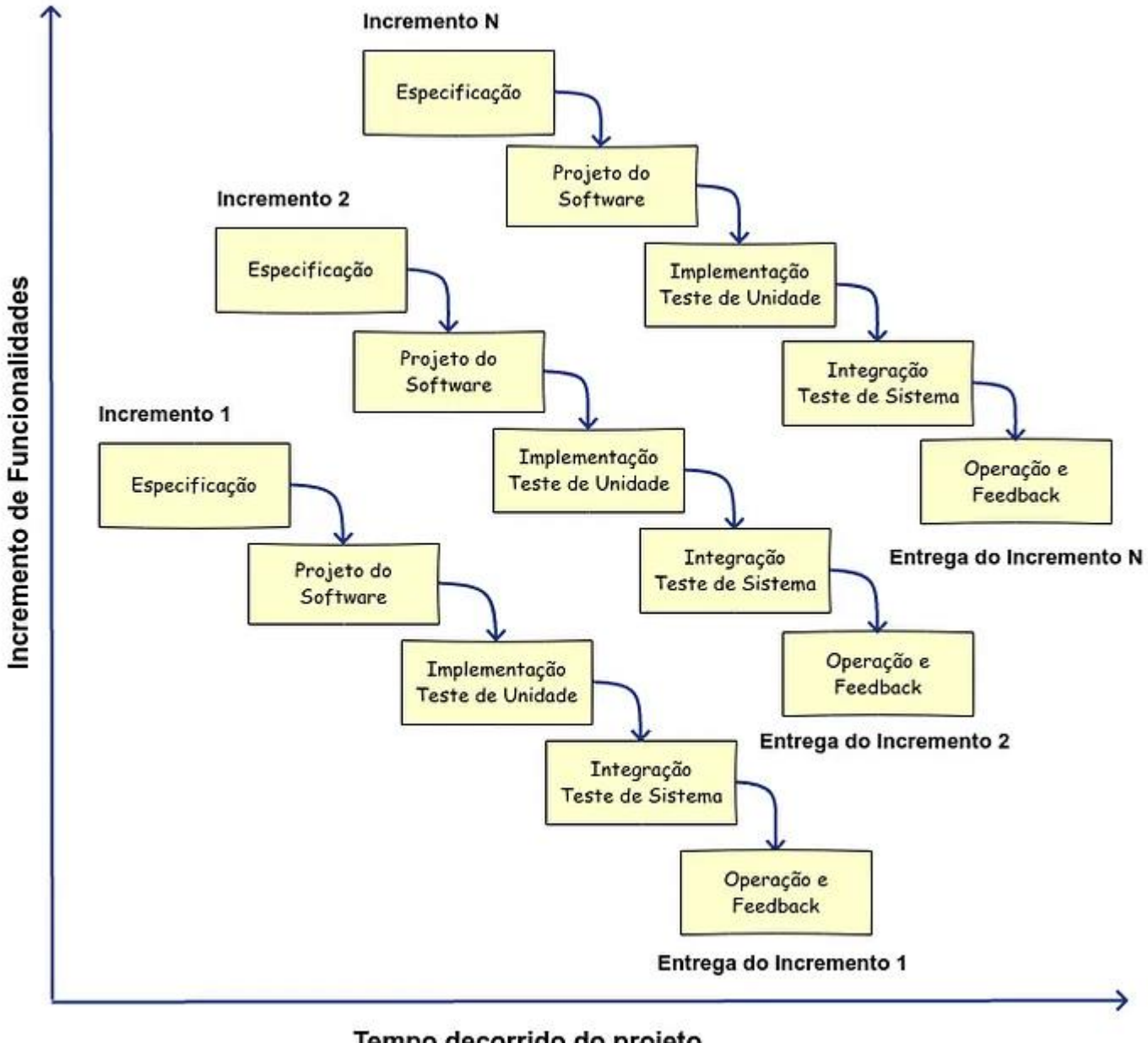
- Concentra-se nas **mudanças** sofridas pelo software.

Tipos de Mudanças:

- Correção
  - Corrige defeitos encontrados pelo cliente (manutenção corretiva).
- Adaptação
  - Acomoda mudanças ocorridas no ambiente do software (manutenção adaptativa).
- Melhoramento Funcional
  - Estende o software para além de suas exigências funcionais originais (manutenção perfectiva).

Desenvolvimento Incremental

- Em vez de entregar o sistema como um todo, o desenvolvimento e a entrega são divididos em incrementos, com cada incremento entregando parte da funcionalidade requerida
- Requisitos dos usuários são priorizados e os requisitos de mais alta prioridade são incluídos nas iterações iniciais
- Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são "congelados". Embora os requisitos possam continuar a evoluir para incrementos posteriores.


Incremento N
Especificação
Projeto do Software
Implementação Teste de Unidade
Integração Teste de Sistema
Operação e Feedback
Entrega do Incremento N
Incremento 2
Especificação
Projeto do Software
Implementação Teste de Unidade
Integração Teste de Sistema
Operação e Feedback
Entrega do Incremento 2
Incremento 1
Especificação
Projeto do Software
Implementação Teste de Unidade
Integração Teste de Sistema
Operação e Feedback
Entrega do Incremento 1
Incremento de Funcionalidades
Tempo decorrido do projeto

Vantagens

- É particularmente útil quando não há mão-de-obra disponível para uma implementação completa, dentro do prazo comercial de entrega estabelecido pelo projeto (PRESSMAN, 2006);
- O cliente não precisa receber todo o sistema para poder usá-lo. Como a implementação é feita por pedaços ordenados por prioridades, o cliente pode ter acesso às partes mais importantes antes, podendo utilizá-las imediatamente;
- *"O custo de acomodar as mudanças nos requisitos do cliente é reduzido".* (SOMMERVILLE, 2011) A quantidade de análise e documentação a ser refeita é muito menor do que o necessário no modelo em cascata.
- É mais fácil obter feedback dos clientes sobre o desenvolvimento que foi feito. Os clientes podem fazer comentários sobre as demonstrações do software e ver o quanto foi implementado. Os clientes têm dificuldade em avaliar a evolução por meio de documentos de projeto de software.
- Para PRESSMAN (2006) e SOMMERVILLE (2011), é possível obter entrega e implementação rápida de um software útil ao cliente, mesmo se todas as funcionalidades não forem incluídas. Os clientes podem usar e obter ganhos a partir do software inicial antes do que seria possível com um processo em cascata.

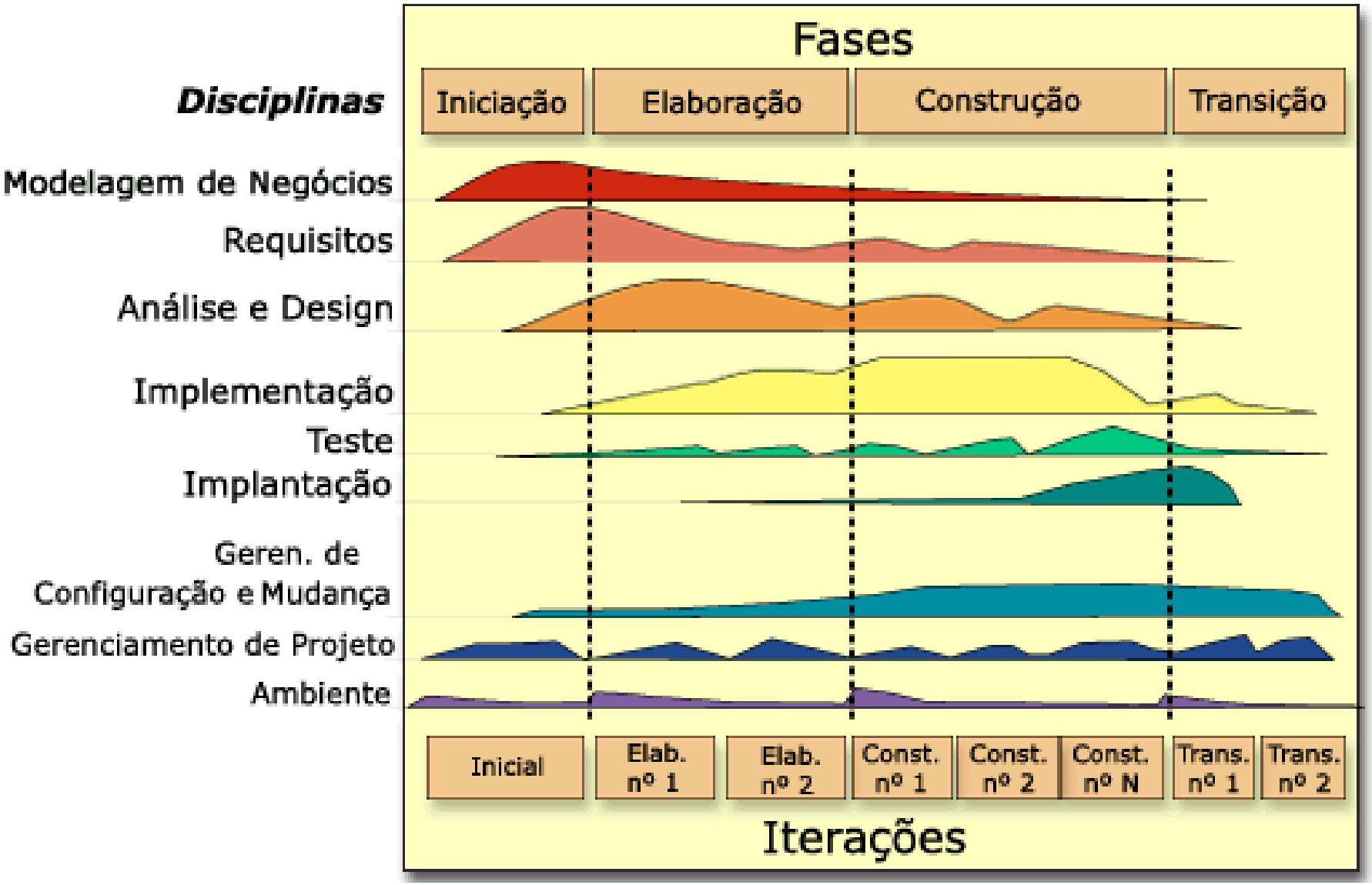
Desvantagens

- Os problemas são particularmente críticos para os sistemas grandes, complexos e com vida-longa, onde várias equipes desenvolvem diferentes partes do sistema;
- Sistemas de grande porte necessitam de um framework ou arquitetura estável, e as responsabilidades das diferentes equipes de trabalho do sistema precisam ser claramente definidas, respeitando essa arquitetura.
- O **progresso não é visível** e os gerentes precisam de entregas regulares para mensurar o progresso. Se os sistemas forem desenvolvidos com rapidez, não será economicamente viável produzir documentos que reflitam cada uma das versões do sistema;
- **A estrutura do sistema tende a se degradar com a adição dos novos incrementos**. Tempo e dinheiro devem ser direcionados para refatoração e melhorias do software, pois as constantes mudanças sem previsão do futuro tendem a corromper sua estrutura. A incorporação de mudanças do software torna-se cada vez mais difícil e custosa

Visão Geral do RUP

- Objetivo
  - Depois desta aula você terá uma visão geral do RUP (uma metodologia para desenvolvimento de software), incluindo suas características e seus componentes principais.
  - É um processo proprietário de Engenharia de software criado pela Rational Software Corporation, adquirida pela IBM e ganhou o nome de IRUP IBM Rational Unified Software(desde 2003)
  - Utiliza uma abordagem de orientação a objetos em sua concepção e é projetado e documentado utilizando a notação UML para ilustrar os processos em ação.
  - Assegurar uma produção de alta qualidade de software, que realiza a necessidade do usuário seguindo prazos e orçamentos.

Desenvolvimento Iterativo e Incremental: RUP – Rational Unified Process



O que é o RUP?

- O nome é uma abreviação de Rational Unified Process
  - mas na verdade é
    - Processo + Métodos + Linguagem (UML)
  - e os autores argumentam que é
    - *Framework para gerar processos*

O que é o RUP?

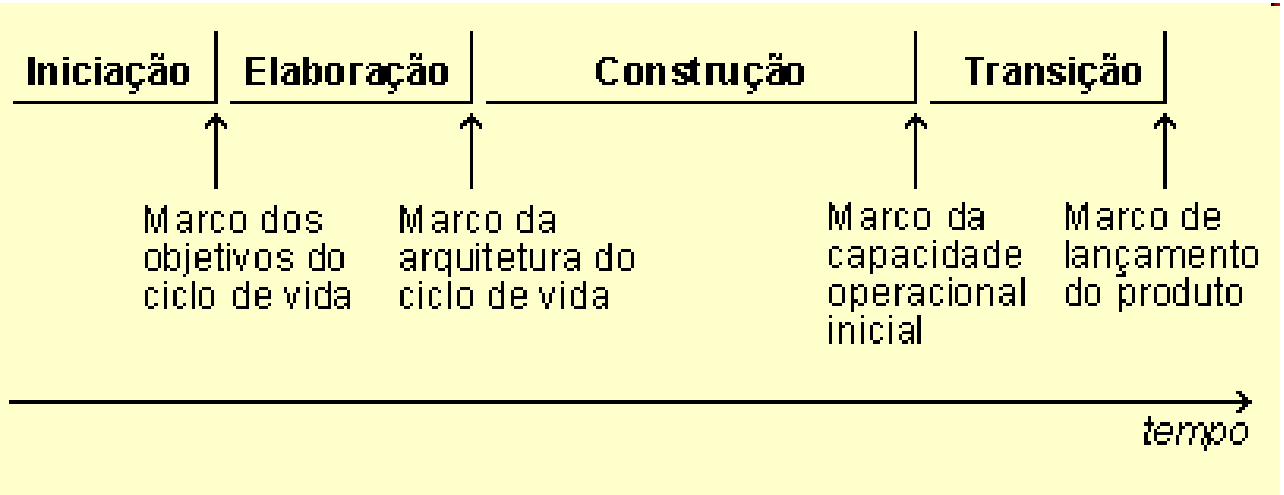
- Conjunto de atividades
  - bem definidas
  - com responsáveis
  - com artefatos de entrada e saída
  - com dependências entre as mesmas e ordem de execução
  - com modelo de ciclo de vida
  - descrição sistemática de como devem ser realizadas
  - guias (de ferramentas ou não), *templates*
  - utilizando diagramas de UML

Características Principais do RUP

- O desenvolvimento de sistemas seguindo o RUP é
  - Iterativo e incremental
  - Guiado por casos de uso (use cases)
  - Baseado na arquitetura do sistema

Iterativo: Aquele que faz progresso através de tentativas repetidas de refinamento.
Incremental: o software é criado e entregue por pedaços, cada incremento representa um subconjunto de funcionalidades completas

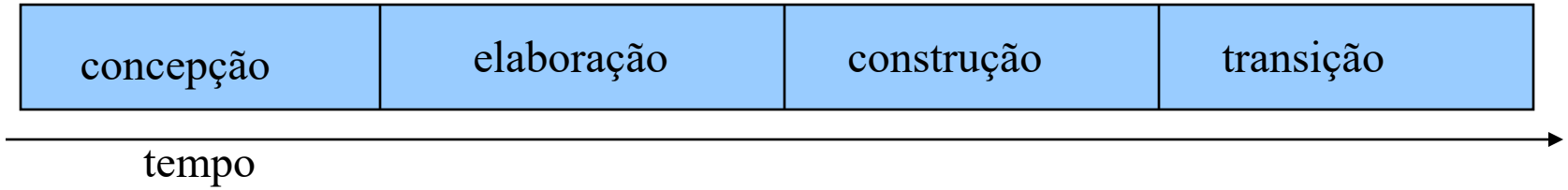
RUP – Rational Unified Process



- Concepção (define o escopo do projeto)
- Elaboração (define os requisitos e a arquitetura)
- Construção (desenvolve o sistema)
- Transição (implanta o sistema)

O RUP é iterativo e incremental

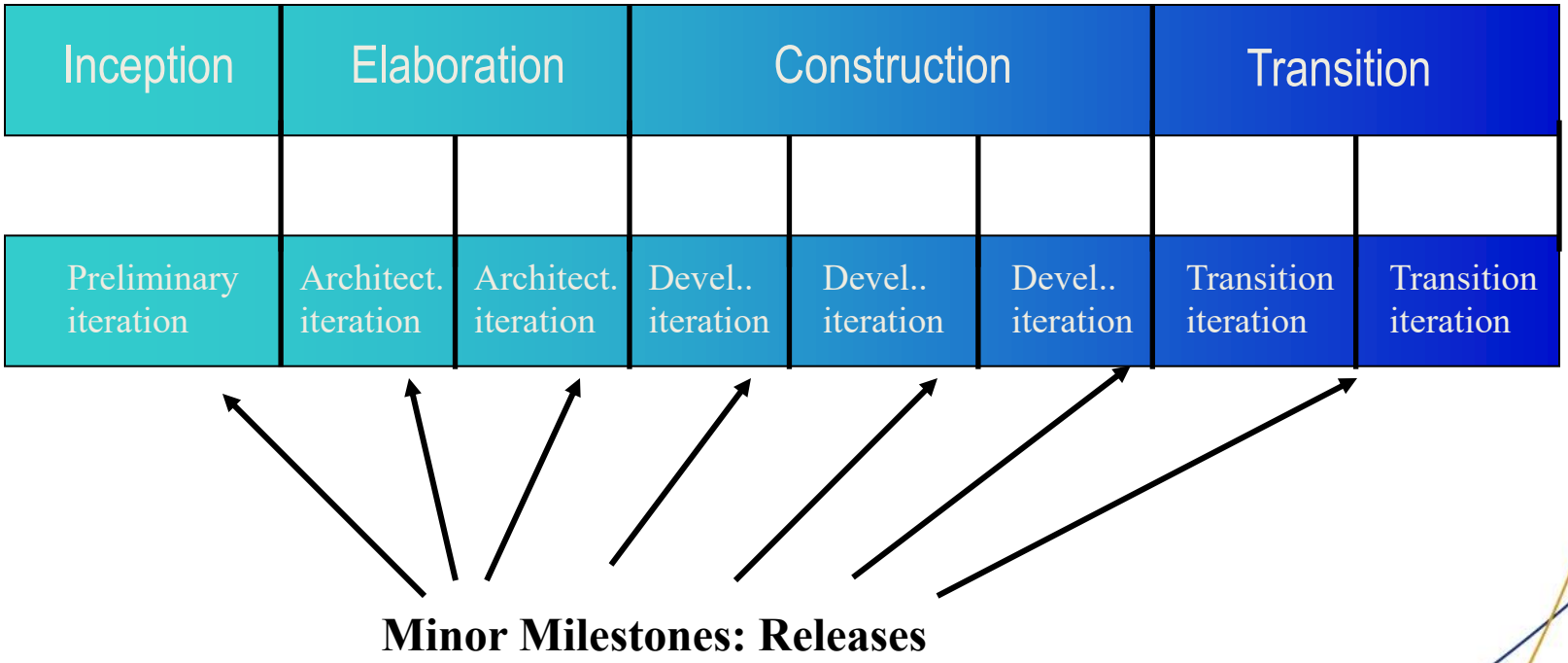
- O ciclo de vida de um sistema consiste de quatro fases:



- Concepção (define o escopo do projeto)
- Elaboração (detalha os requisitos e a arquitetura)
- Construção (desenvolve o sistema)
- Transição (implanta o sistema)

O RUP é iterativo e incremental

- Cada fase é dividida em iterações:



O RUP é iterativo e incremental

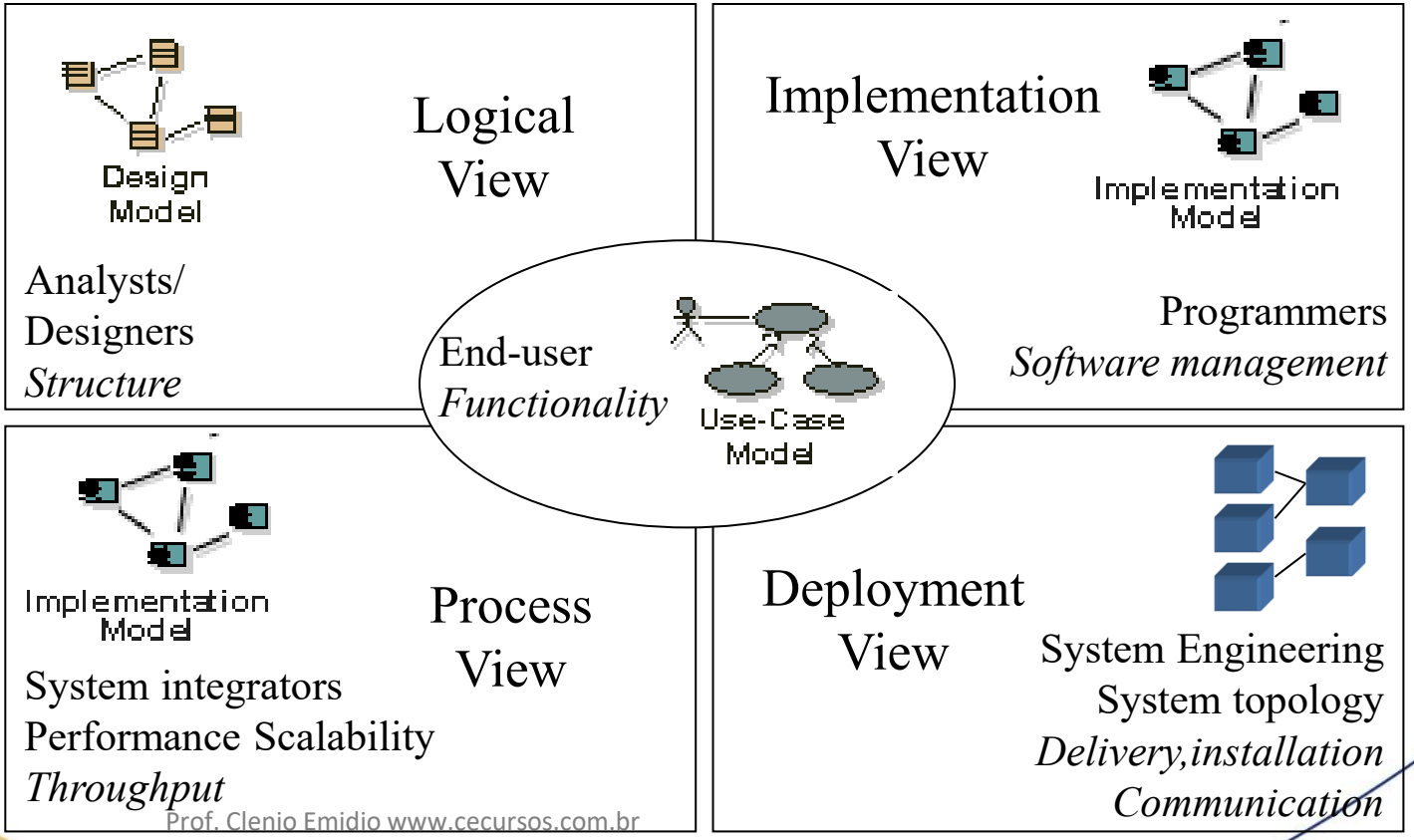
- Cada iteração
  - é planejada
  - realiza uma seqüência de atividades (de elicitação de requisitos, análise e projeto, implementação, etc.) distintas
  - geralmente resulta em uma versão executável do sistema
  - é avaliada segundo critérios de sucesso previamente definidos

O RUP é baseado na arquitetura do sistema

- Arquitetura
  - visão geral do sistema em termos dos seus subsistemas e como estes se relacionam
- A arquitetura é prototipada e definida logo nas primeiras iterações
- O desenvolvimento consiste em complementar a arquitetura
- A arquitetura serve para definir a organização da equipe de desenvolvimento e identificar oportunidades de reuso

O RUP é baseado na arquitetura do sistema

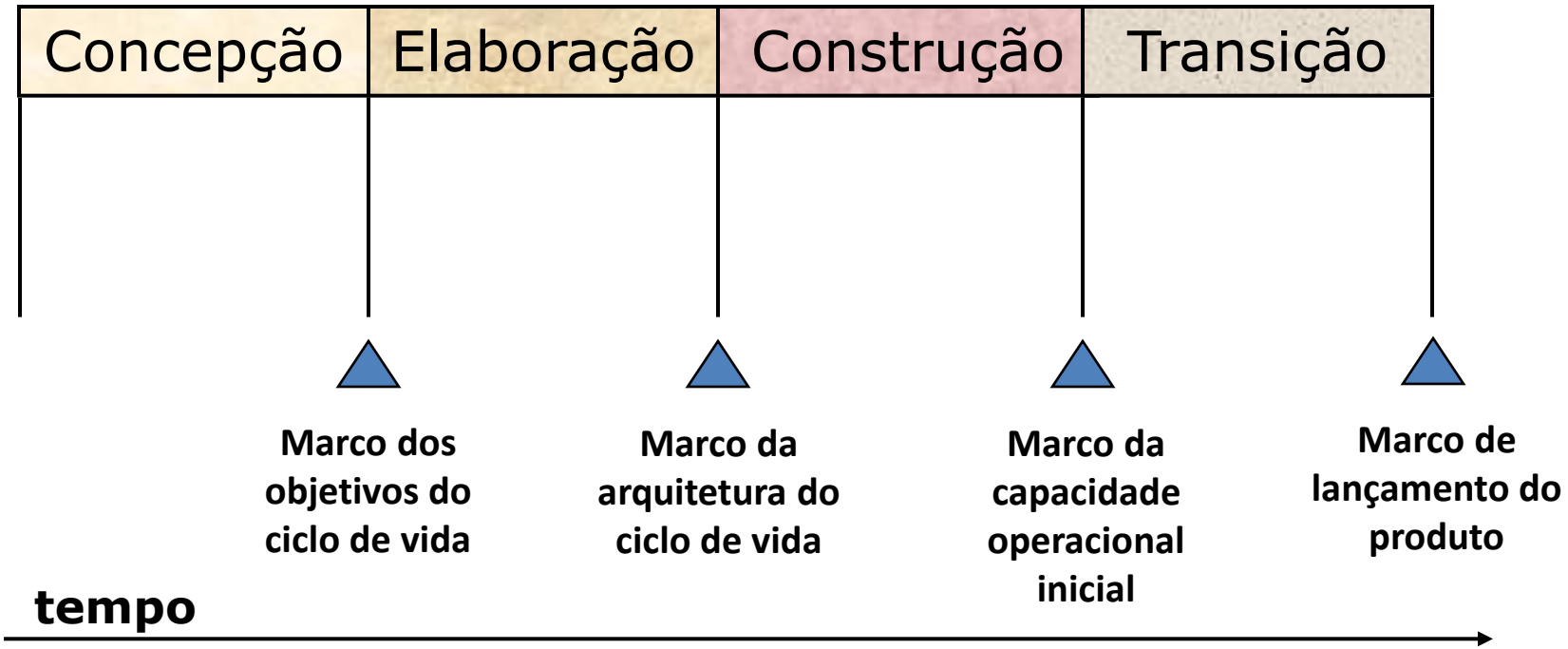
- Idealmente, tem-se 5 visões da arquitetura



Organização do RUP

- Fluxos de atividades
- Atividades
  - passos
  - entradas e saídas
  - guias (de ferramentas ou não), *templates*
- Responsáveis (papel e perfil, não pessoa)
- Artefatos

Marcos



O projeto poderá ser anulado ou completamente repensado caso o marco não seja atingido.

Marco dos Objetivos do Ciclo de Vida

- Análise dos objetivos do ciclo de vida do projeto e tomada de decisão de prosseguir com o projeto ou cancelá-lo

Critérios de Avaliação

  - Consentimento dos envolvidos sobre a definição do escopo e as estimativas de custo/programação.
  - Consenso de que o conjunto correto de requisitos foi capturado e de que existe uma compreensão compartilhada desses requisitos.
  - Consenso de que as estimativas de custo/programação, as prioridades, os riscos e o processo de desenvolvimento são adequados.
  - Todos os riscos foram identificados e existe uma estratégia atenuante para cada um.

Marco da Arquitetura do Ciclo de Vida

- Exame nos objetivos e o escopo do sistema, a opção de arquitetura e a resolução dos principais riscos.

Critérios de Avaliação

- A Visão e os requisitos do produto são estáveis.
- A arquitetura é estável.
- As abordagens principais a serem usadas no teste e na avaliação foram comprovadas.
- O teste e a avaliação de protótipos executáveis demonstraram que os principais elementos de risco foram tratados e resolvidos com credibilidade.
  - Os planos de iteração para a fase de construção têm detalhes e fidelidade suficientes para permitir o avanço do trabalho.
  - Os planos de iteração para a fase de construção são garantidos por estimativas confiáveis.
  - Todos os envolvidos concordam que a visão atual poderá ser atendida se o plano atual for executado para desenvolver o sistema completo, no contexto da arquitetura atual.
  - A despesa real em oposição à despesa planejada com recursos é aceitável.

Marco da capacidade operacional inicial

- Produto pronto para ser passado para a Equipe de Transição.
- Toda a funcionalidade desenvolvida e os testes **alfa** (se houver algum) foram concluídos.
- Manual do usuário desenvolvido e uma descrição do release atual.

Critérios de Avaliação

- Envolvem respostas para as questões:
  - Este release do produto é estável e desenvolvido o suficiente para ser implantado na comunidade de usuários?
  - Todos os envolvidos estão prontos para a transição para a comunidade de usuários?
  - As despesas reais com recursos ainda são aceitáveis se comparadas com as planejadas?

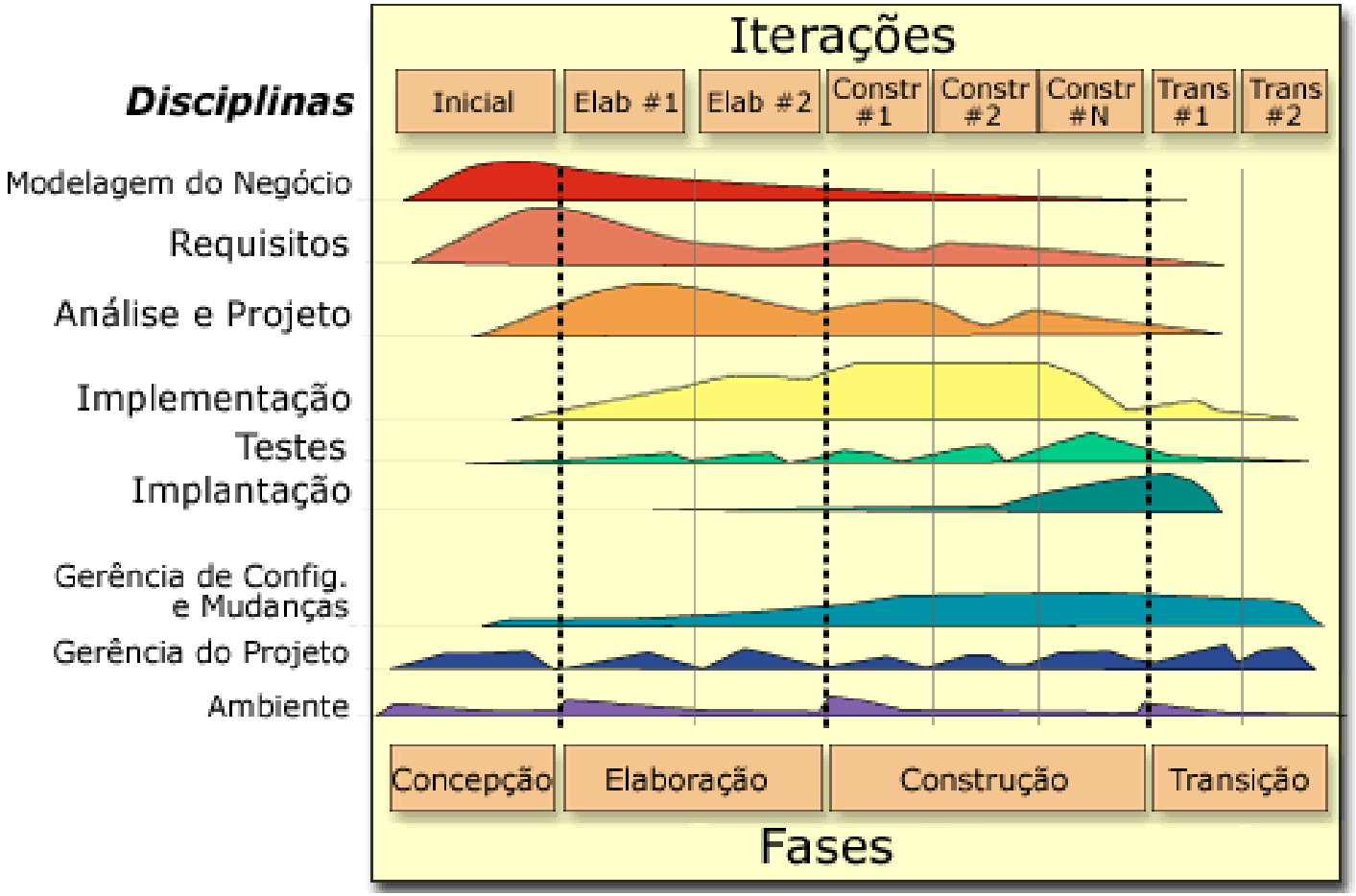
Marco do Release do Produto

- Decisão se os objetivos foram atendidos e se outro ciclo de desenvolvimento deve ser iniciado.
- Pode coincidir com o fim da fase de iniciação do próximo ciclo.
- O Marco do Release do Produto é o resultado da conclusão com êxito dos Artefatos

Fluxos de Atividades do RUP

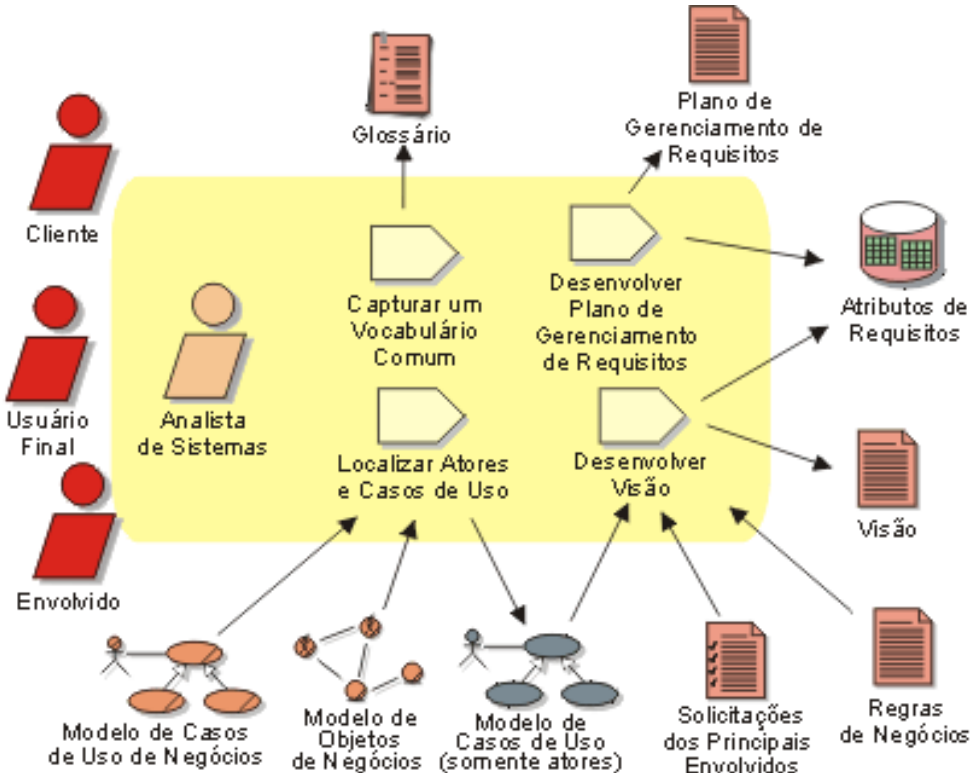
- Agrupam atividades correlacionadas
- Fluxos de atividades básicos:
  - modelagem do negócio
  - requisitos
  - análise e projeto
  - implementação
  - testes
  - distribuição
- Fluxos de atividades de suporte:
  - gerência de configuração e mudanças
  - gerência do projeto
  - configuração do ambiente

Fases, Iterações e Fluxos de Atividades



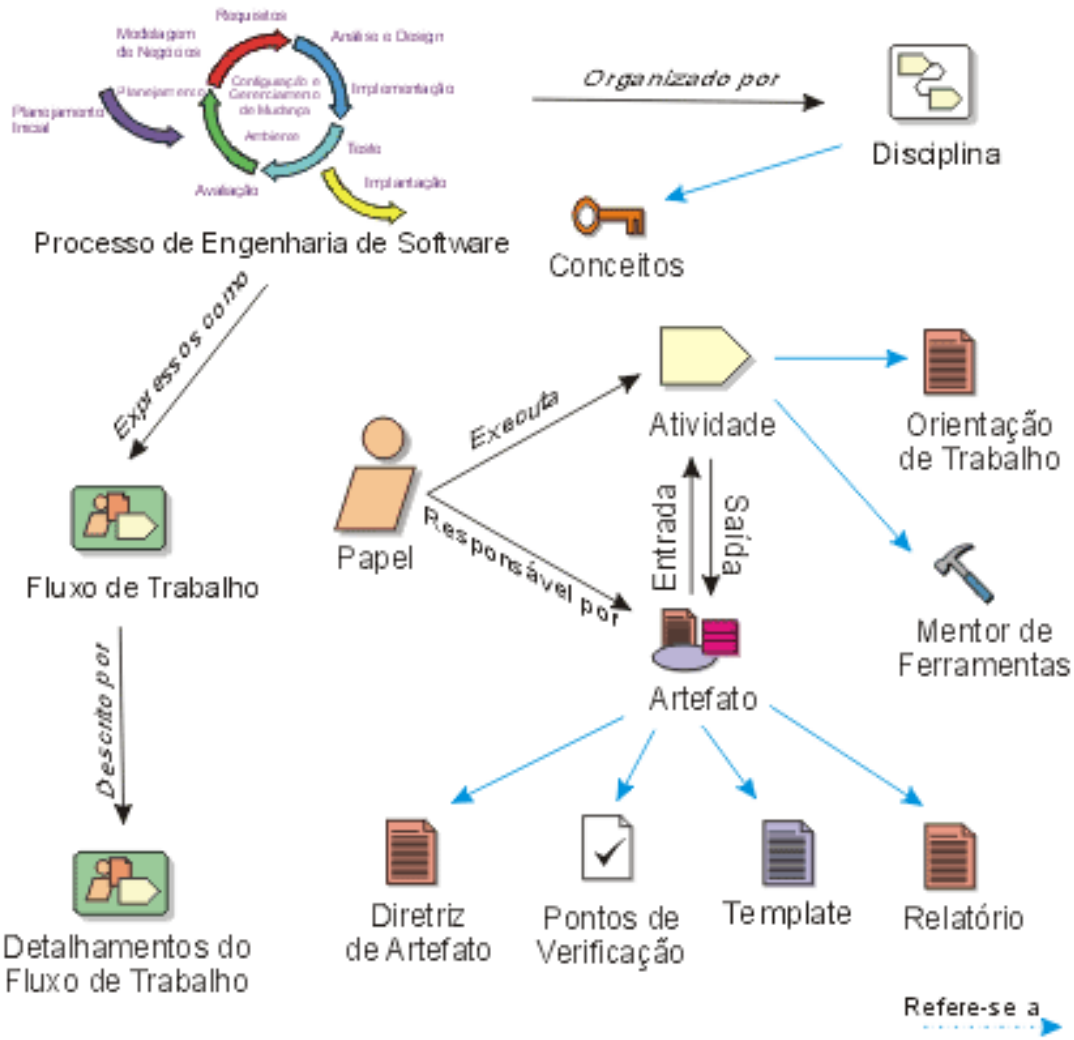
Responsáveis, Atividades e Artefatos

- Os fluxos de atividades do RUP são descritos através de responsáveis, atividades e artefatos



Fonte: Rational

Conceitos-chave

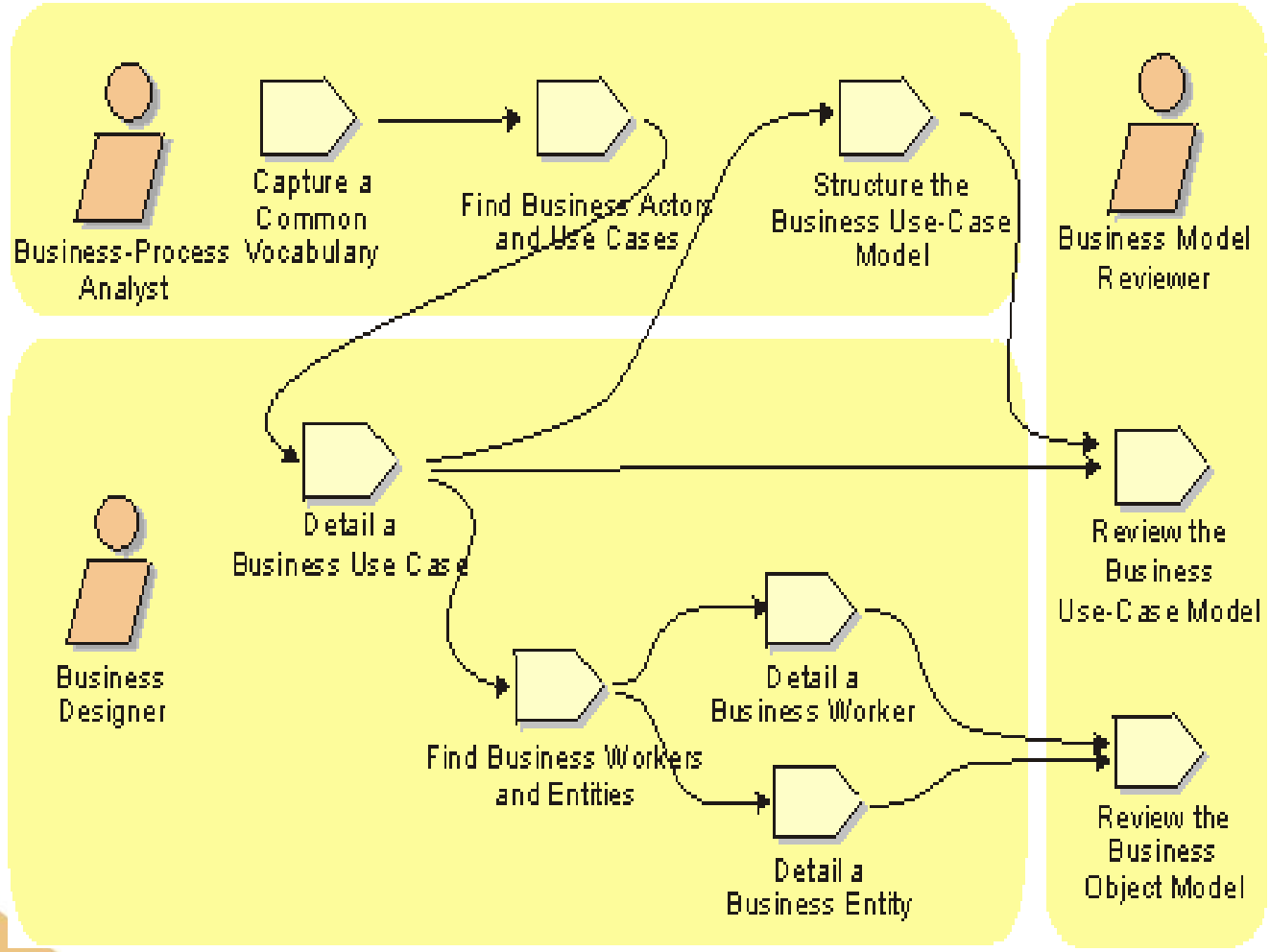


Modelagem do Negócio

- Objetivos:
  - descobrir “o problema por trás do problema”
  - entender a estrutura e dinâmica da organização
  - assegurar que os clientes, usuários e desenvolvedores têm a mesma visão do negócio
  - descobrir os requisitos do sistema necessários para suportar o negócio

Desenvolver um modelo do negócio

Modelagem do Negócio

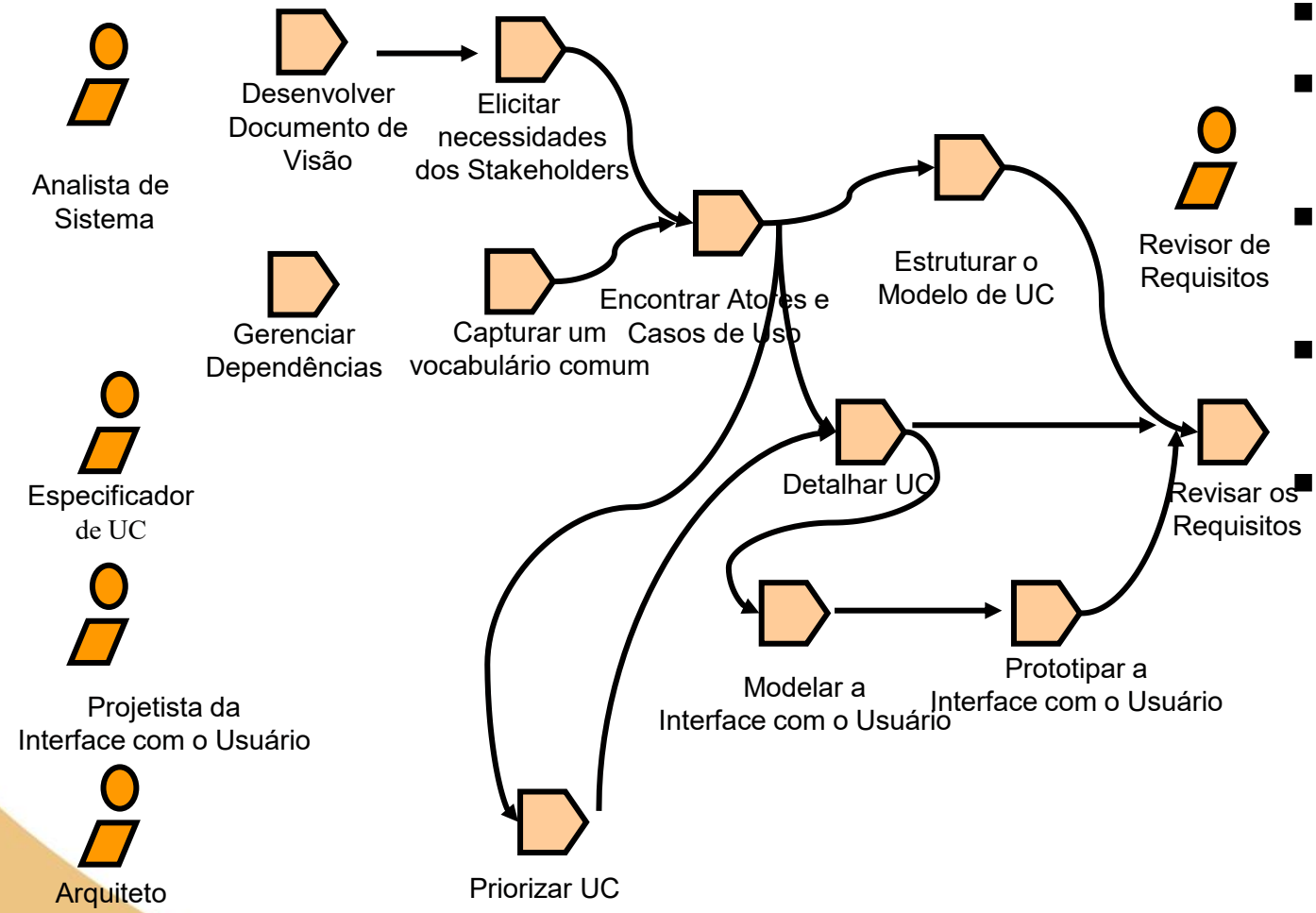


- Glossário
- Modelo de Casos de Uso do Negócio
- Modelo de Objetos do Negócio

Requisitos

- Objetivos:
  - descrever *o quê* o sistema deve fazer, em acordo com o cliente e usuários
  - definição de como gerenciar escopo e mudanças de requisitos
  - delimitar o escopo do sistema e prover uma base para o planejamento das iterações
  - definir a interface com o usuário

Requisitos

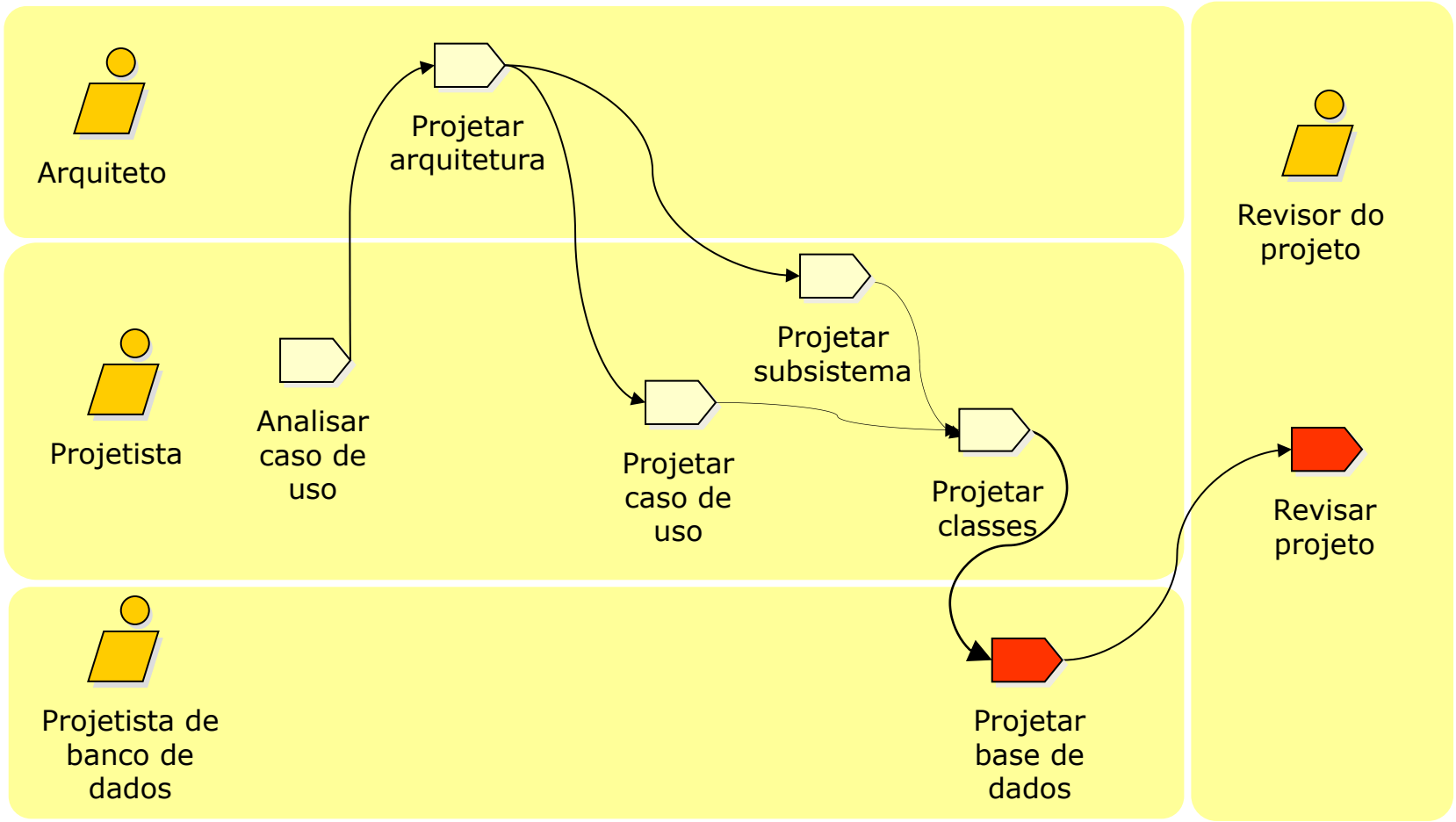


- Glossário
- Documento de Visão
- Especificações Suplementares
- Modelo de Casos de Uso
- Protótipo da Interface com o Usuário

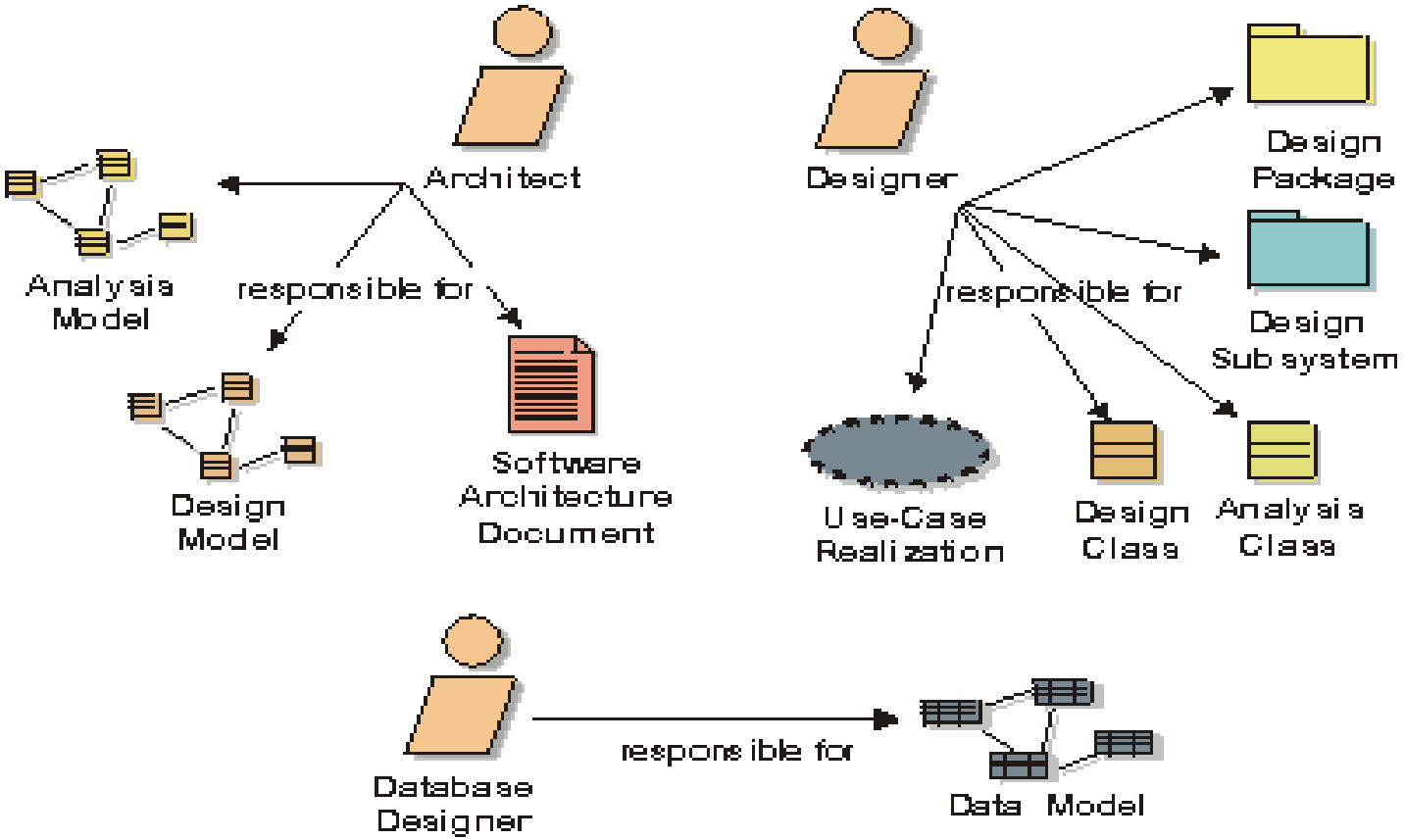
Análise e Projeto

- Objetivos:
  - transformar os requisitos em um modelo para implementação do sistema
  - encontrar uma arquitetura robusta para o sistema
- Análise:
  - assegura que os requisitos funcionais são tratados
- Projeto:
  - adapta os resultados da análise aos requisitos não funcionais e ambiente de implementação

Análise e Projeto



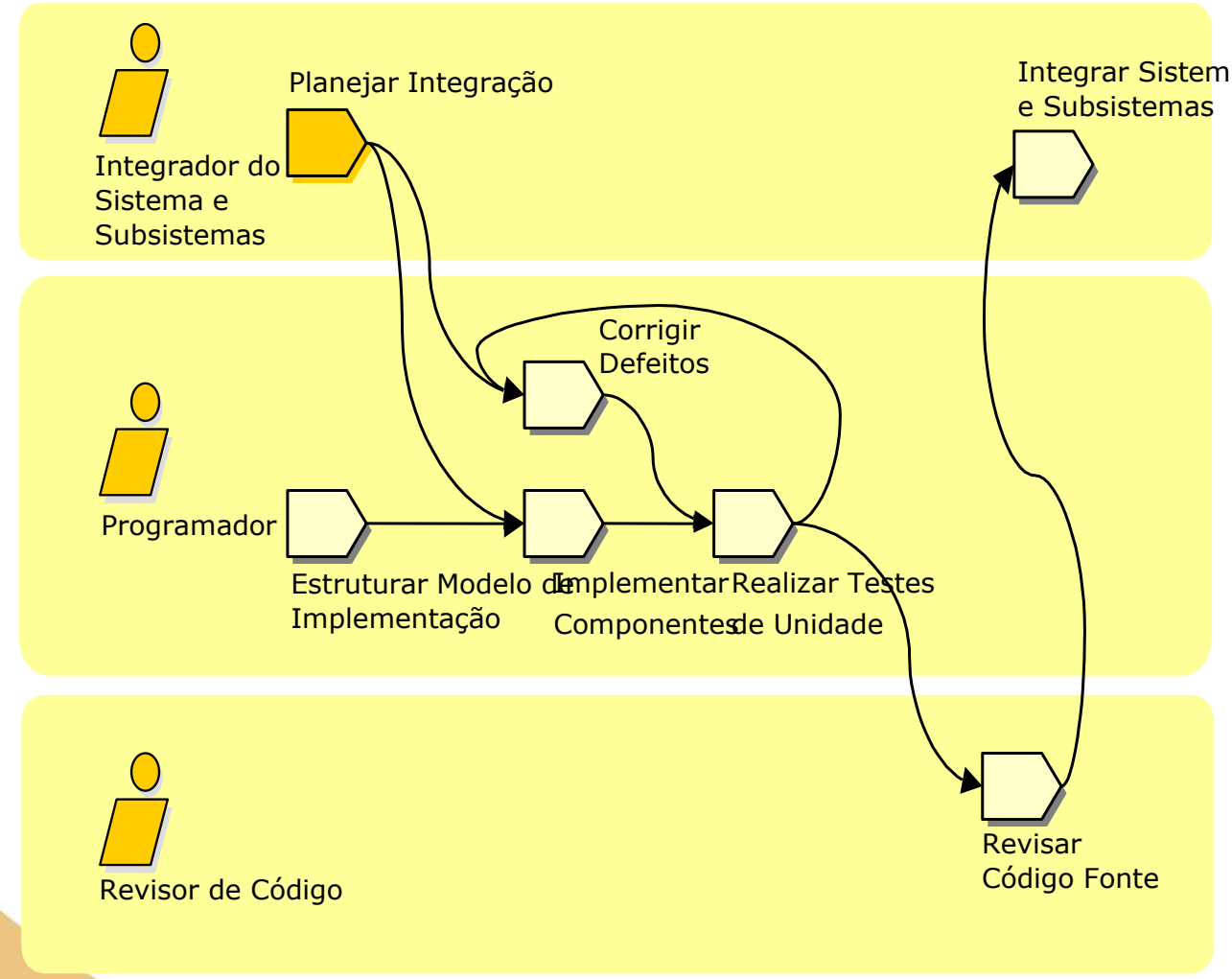
Análise e Projeto



Implementação

- Objetivos:
  - implementar os componentes necessários
  - testar os componentes implementados como unidades
  - integrar os componentes implementados em um sistema executável

Implementação

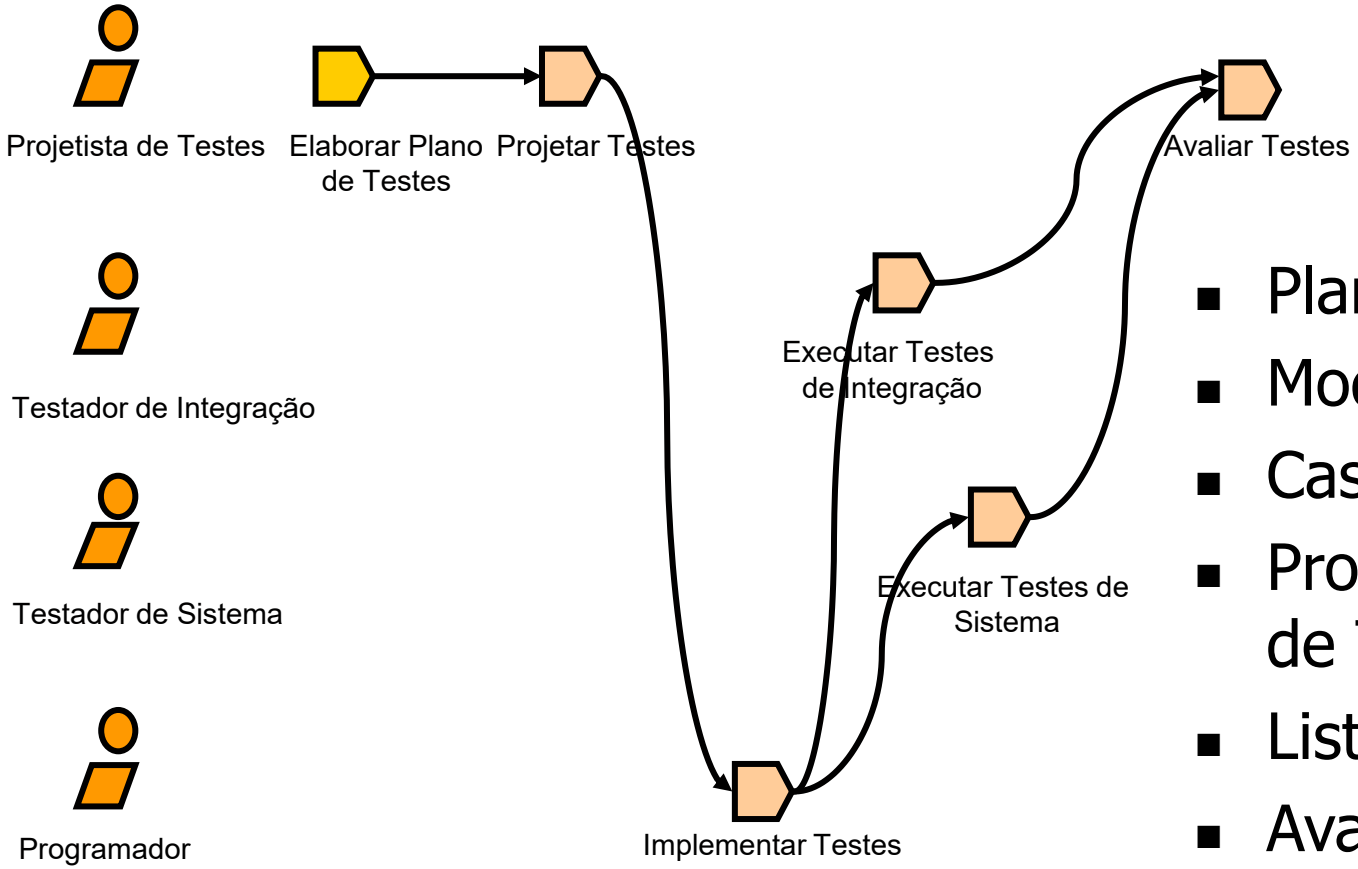


- Modelo de Implementação
- Componentes implementados
- Subsistemas implementados

Testes

- Objetivos:
  - verificar a interação e integração dos componentes
  - verificar se todos os requisitos foram corretamente implementados
  - identificar defeitos e assegurar as correções de acordo com as prioridades de entrega de cada componente

Testes



- Plano de Teste
- Modelo de Teste
- Casos de Teste
- Procedimentos de Teste
- Lista de defeitos
- Avaliação dos Testes

Implantação

- Objetivo:
  - entregar o produto aos usuários finais
- Muito dependente do contexto do negócio e do projeto => precisa ser configurado

Implantação

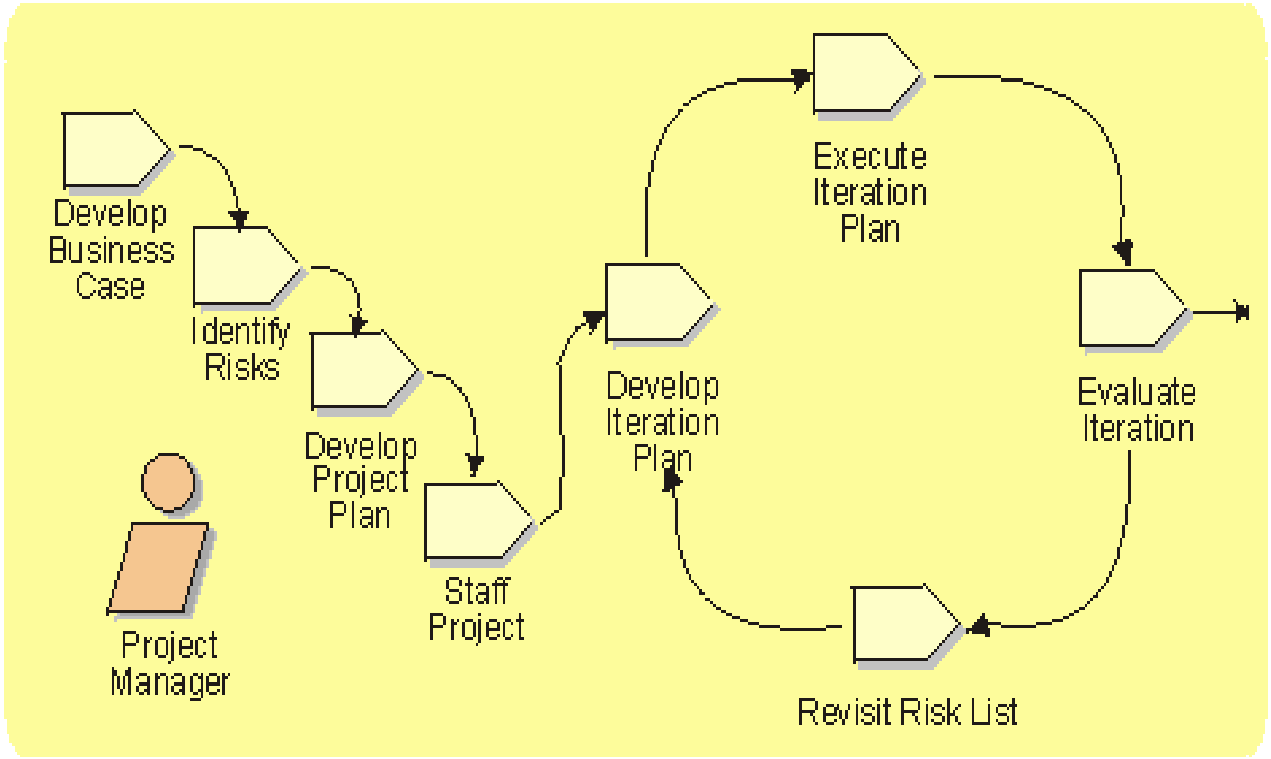
- Possíveis atividades:
  - produzir o (que falta do) software
    - scripts de instalação, documentação para o usuário, programas para conversão de dados, etc.
  - embalar e distribuir o software
  - instalar o software
  - realizar migração
    - troca de sistema antigo pelo novo, conversão de dados
  - treinamento do usuário
  - aceitação formal pelo cliente
  - planejamento e condução de beta testes

Gerência de Projeto

- Objetivos:
  - prover um framework para gerenciar projetos e riscos
  - prover orientações para o planejamento de atividades, definição da equipe, execução e monitoração de projetos

Planejamento e monitoração das iterações!

Gerência de Projeto

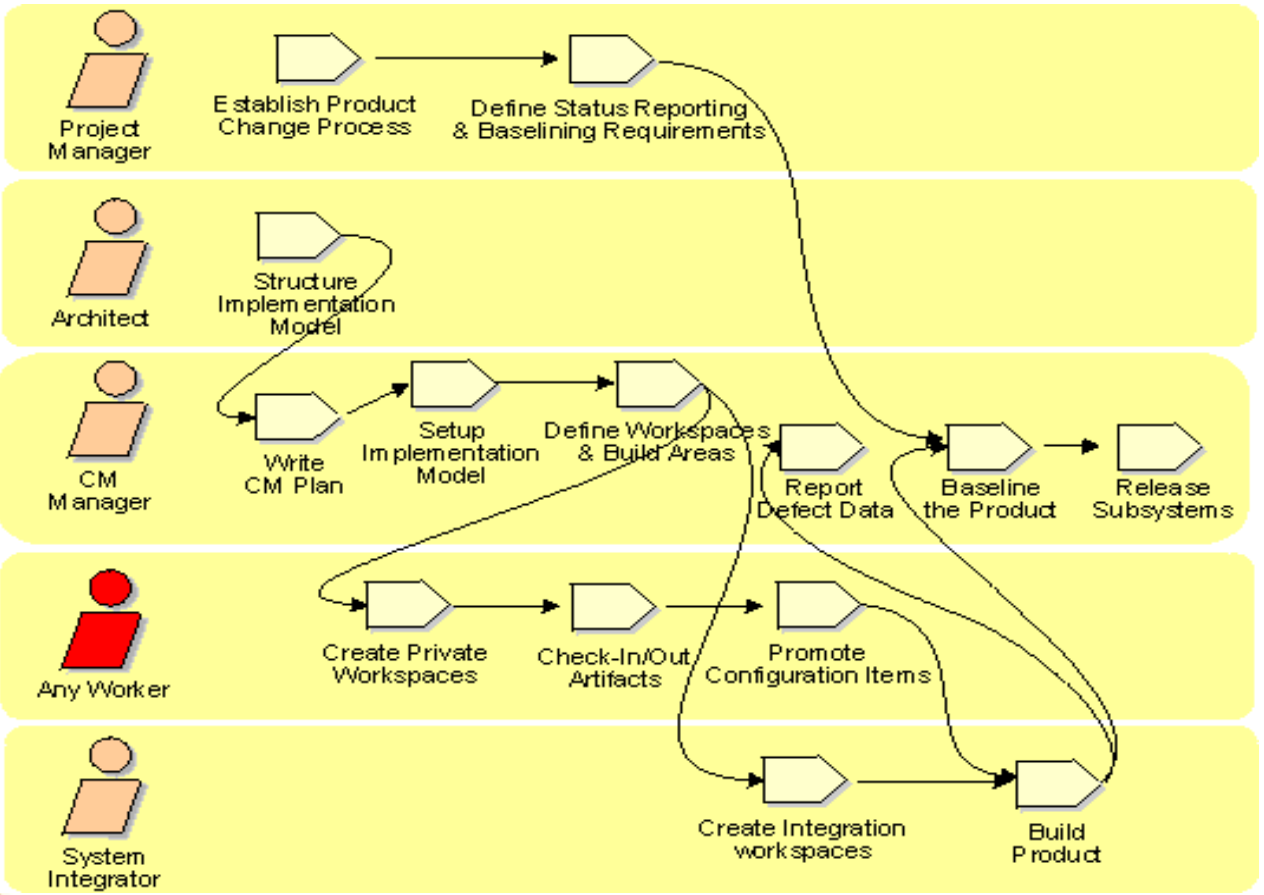


- Plano de Negócios
- Plano do Desenv. do Software:
  - plano de métricas
  - lista de riscos
  - plano do projeto
  - plano da iteração

Gerência de Configuração e Mudanças

- Objetivos:
  - identificar, definir e manipular itens de software
  - controlar modificações e versões destes itens
  - reportar e armazenar a situação dos itens e as solicitações de mudanças
  - garantir a completude, consistência e corretude dos itens
  - controlar o armazenamento, manipulação e entrega destes itens

Gerência de Configuração e Mudanças



- Plano de gerência de configuração

Fonte: Rational

Gerência de Ambiente

- Objetivo:
  - prover o processo e as ferramentas necessárias ao desenvolvimento
- Possíveis Atividades:
  - configurar o RUP
  - desenvolver guidelines
  - selecionar e adquirir ferramentas
  - adaptar ou desenvolver ferramentas
  - suportar o ambiente de desenvolvimento (backups, administração de contas, etc.)
  - treinamento
  - implantação do RUP na organização

Gerência de Ambientes

- Processo para um projeto *x* Processo para a organização
- Processo para um projeto
  - considerar tamanho, reuso, tipo do ciclo (inicial *x* de evolução)
- Processo para a organização
  - considerar cultura, pessoas, tecnologias e aplicações chave, ...

Utilizando o RUP é possível..

- Obter qualidade de software
- Produtividade no desenvolvimento, operação e manutenção
- Controle sobre o desenvolvimento dentro de custos, prazos e níveis de qualidade desejados
- Estimativa de prazos e custos com maior precisão.

Obrigado!