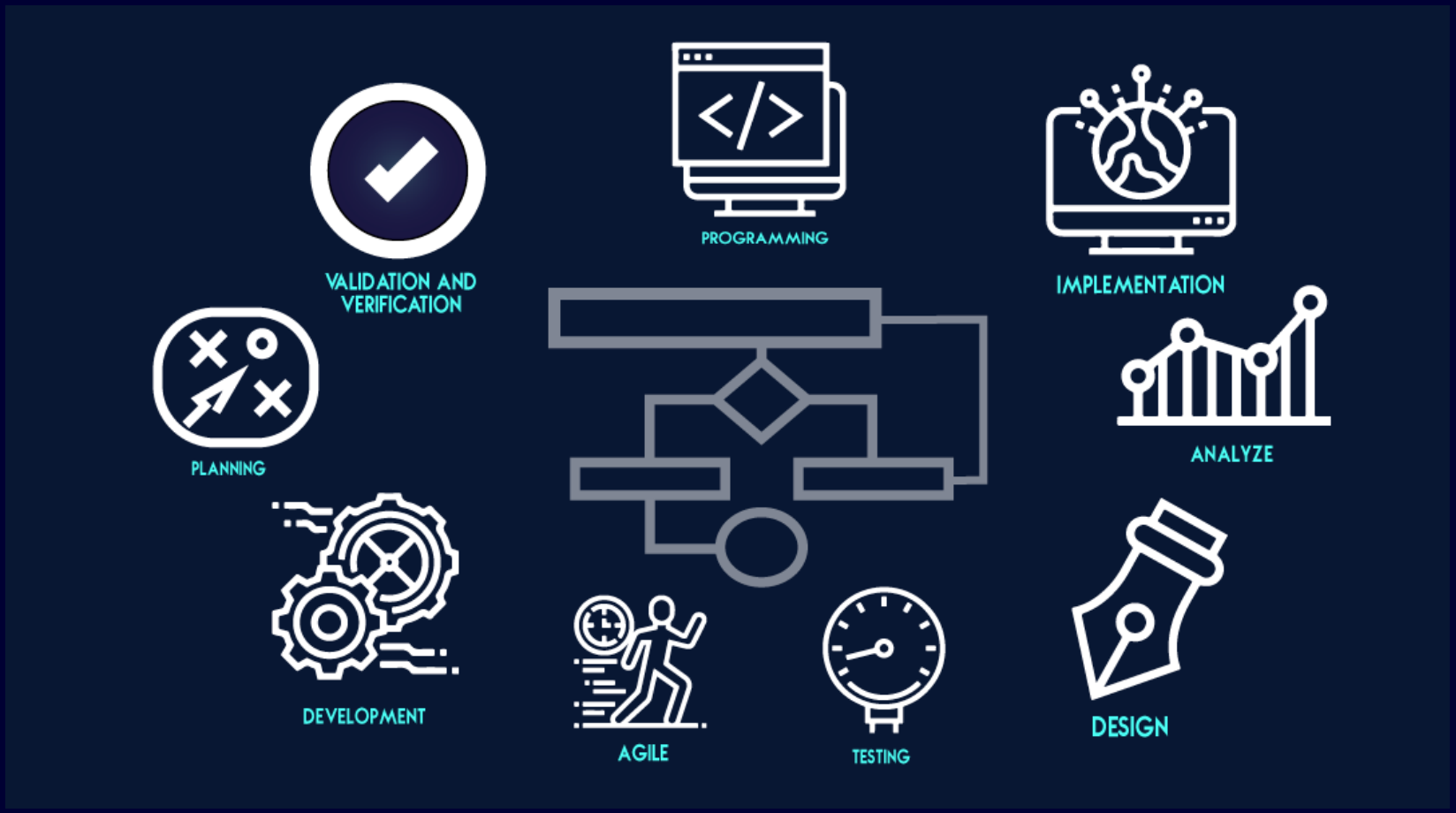
Unichristus
Centro Universitário Christus



Engenharia de Software: Ciclos de vida

Lattes - linkedin
euristenhojr@gmail.com
http://www.unichristus.edu.br/

Euristenho Queiroz de Oliveira **Júnior**
Especialista em Engenharia de Software
MSc em Engenharia de Software

AGENDA

1. Apresentação
2. Livros
3. Acordo de Convivência
4. Modelos x Processos
5. - Modelos de Desenvolvimento
6. - RUP
7. - Ágil x Tradicional
8. - Exercícios

Unichristus
Centro Universitário Christus

Apresentação

FORMAÇÃO ACADÊMICA

- Graduado em Telemática/Telecomunicações - IFCE (2002 - 2008)
- Especialista em Engenharia de Software - FA7 (2011 - 2013)
- MSc em Engenharia de Software - UFPE (2011 - 2015)

CURRÍCULO PROFISSIONAL

- Atuei 4 anos na empresa privada
- 10 anos no ambiente Público
- Atualmente Líder Técnico de 45 Projetos de Tecnologia na SEPOG/PMF

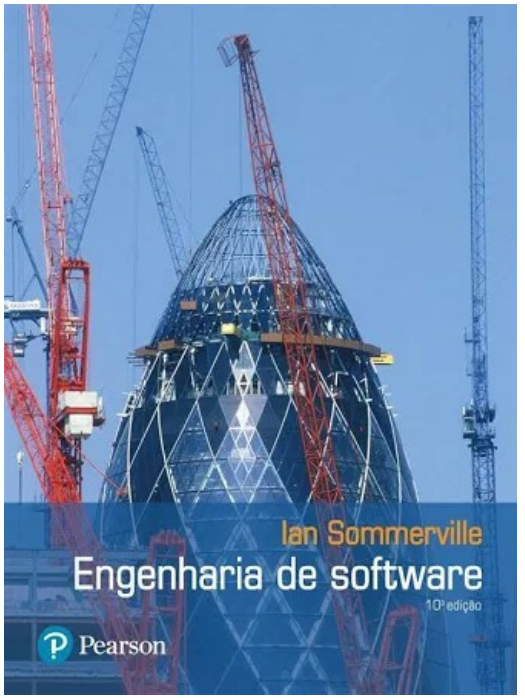
Apresentação

DOCÊNCIA

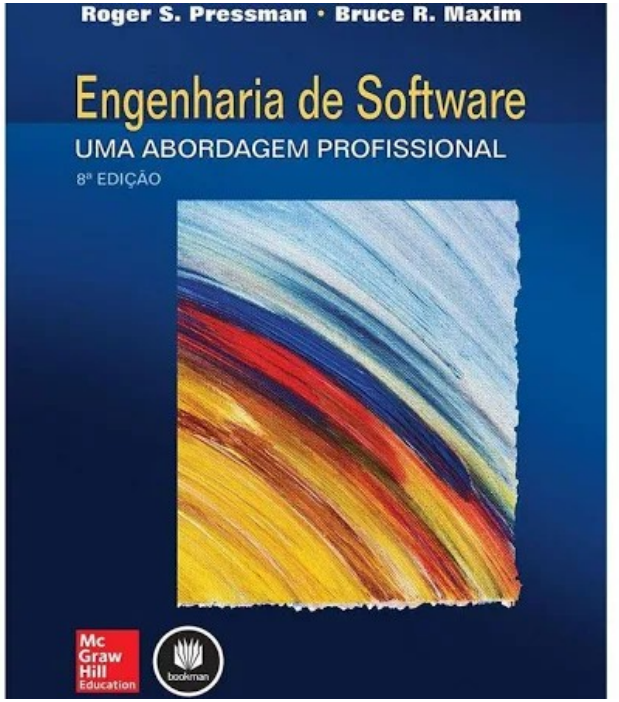
- Professor Substituto das Disciplinas de Sistemas de Informação – FA7 (2011 - 2012)
- Professor da Especialização em Sistemas WEB – FJN (2011 - 2012)
- Professor de Bancas de graduação em Sistemas de Informações – FA7 (2012)
- Professor dos Cursos de Tecnologia da Informação da Unifanor (2015 – 2018)
- Professor do Curso de Sistemas de Informação Unichristus (2018 - Atual)

Livros

- **Engenharia de Software** - 10ª Ed – Lan Sommerville - Pearson



- **Engenharia de Software – Uma abordagem profissional**- 8ª Ed. AMGH



Dicas de Convivência

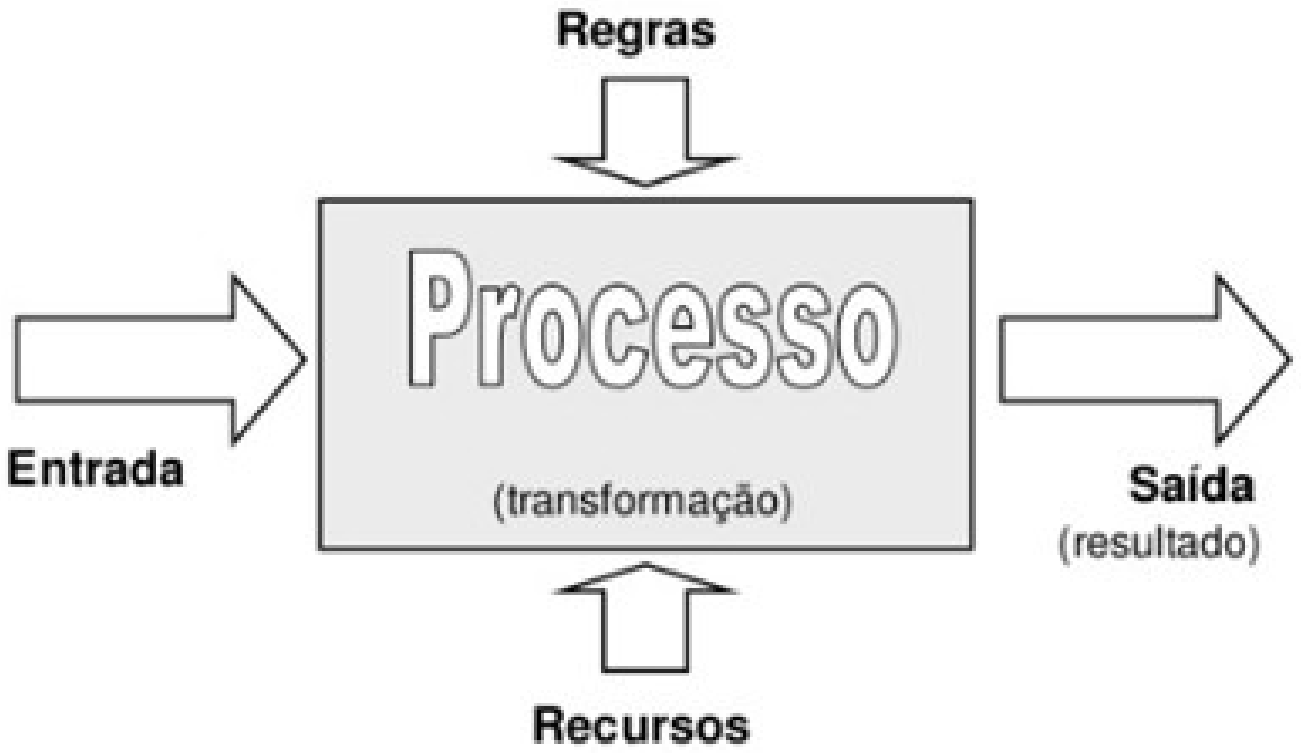
- Horários
- Conversas
- Dúvidas
- Celular
- Avaliações

Questionamentos

Objetivo desta Aula

O que é processo de desenvolvimento de software. A importância do processo de desenvolvimento de software. Elementos de processo de desenvolvimento de software: fases; atividades; artefatos; e papéis.

PROCESSO DE DESENVOLVIMENTO



PROCESSO DE DESENVOLVIMENTO

O processo de desenvolvimento de software é um conjunto de atividades que visam a criação do software e sua qualidade. Existem várias etapas ou processso para se criar um software, o modelo de desenvolvimento pode ser dividido em Requisitos, Modelagem, Código e Outros.

PROCESSO DE DESENVOLVIMENTO

O Guia PMBOK® define processo como sendo um conjunto de atividades inter-relacionadas realizadas para obter um conjunto específico de produtos, resultados ou serviços (PMBOK, 2008). Segundo o IEEE, um processo é uma sequencia de passos executada com um determinado objetivo (IEEE, 2003).

Para o CMMI, um processo é definido quando tem uma descrição que é mantida, ou seja, tem documentação que detalha o que é feito (produto), quando (etapas), por quem (papéis), os itens utilizados (insumos) e os itens produzidos (resultados)(CMMI, 2006). Os processos podem ser definidos com mais ou menos detalhes e suas etapas podem ter ordenação parcial, o que pode permitir paralelismo entre algumas delas (PAULA FILHO, 2009).

PROCESSO DE DESENVOLVIMENTO

Focando no desenvolvimento de software, Ian Sommerville define um processo de software como um conjunto de atividades que leva à produção de um produto de software (SOMMERVILLE, 2007). Roger S. Pressman define processo de software como um arcabouço para as tarefas que são necessárias para construir software de alta qualidade (PRESSMAN, 2006). Wilson de Paula Filho faz uma analogia interessante, para ele processo é uma receita a ser seguida (PAULA FILHO, 2009).

Processos de softwares são complexos e como todos os processos intelectuais e criativos dependem de julgamento humano. A existência de um processo de software não garante que o software será entregue no prazo, de que ele irá satisfazer as necessidades do cliente, ou exibirá os atributos arquiteturais que manterão as características de qualidade em longo prazo. Um processo deve ser acoplado a uma sólida prática de engenharia de software e deve ser avaliado para garantir que satisfaça a um conjunto de critérios básicos de processo que demonstram ser essenciais para uma engenharia de software bem sucedida (PRESSMAN, 2006).

PROCESSO DE DESENVOLVIMENTO

Não existe um processo ideal. As organizações devem criar, verificar, validar e aperfeiçoar seus próprios métodos (CMMI, 2006). Várias destas desenvolvem abordagens inteiramente diferentes, adequadas à sua realidade, para o desenvolvimento de software. No caso de alguns sistemas, como os sistemas críticos[5], é necessário um processo de desenvolvimento muito bem estruturado. Nos sistemas de negócios, com requisitos que mudam rapidamente, um processo flexível e ágil é provavelmente mais eficaz (SOMMERVILE, 2007).

PROCESSO DE DESENVOLVIMENTO

Existem vários processos de desenvolvimento de software, porém algumas atividades fundamentais são comuns a todos eles (SOMMERVILE, 2007):

- Especificação: define a funcionalidade do software e as restrições sobre sua operação.
- Projeto e implementação: o software que atenda a especificação deve ser produzido.
- Validação de software: o software deve ser validado para garantir que ela faça o que o cliente deseja.
- Evolução: o software deve evoluir para atender aos novos requisitos que naturalmente surgirão.

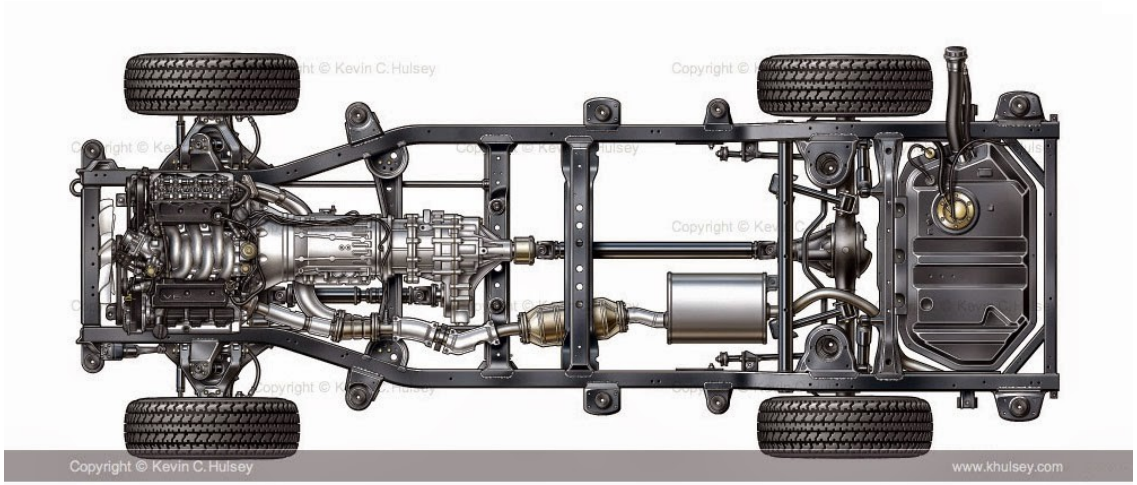
PROCESSO DE DESENVOLVIMENTO

Processos de software têm como base modelos de processo genéricos. Esses modelos genéricos não são descrições definitivas de processos de software. Ao contrário, são abstrações do processo que podem ser usadas para explicar diferentes abordagens para o desenvolvimento de software. Eles podem ser considerados como frameworks de processo que podem ser ampliados e adaptados para criar processos mais específicos de engenharia de software. Os modelos genéricos de processos de software amplamente utilizados são o modelo em cascata, o modelo de desenvolvimento evolucionário e o modelo de desenvolvimento baseado em componentes. Estes, não são mutuamente exclusivos e comumente são utilizados em conjunto, especialmente para desenvolvimento de sistemas de grande porte (SOMMERVILLE, 2007).

PROCESSO DE DESENVOLVIMENTO

PROCESSO DE DESENVOLVIMENTO

MODELOS



PROCESSO DE DESENVOLVIMENTO

PROCESSOS

MODELOS

PROCESSO DE DESENVOLVIMENTO

- Procuram descrever formalmente e de maneira organizada todas as atividades que devem ser seguidas para a obtenção segura de um produto de software

- A escolha do modelo de processo de software depende:
  - da natureza do projeto e da aplicação
  - dos métodos e ferramentas a serem usados
  - dos controles e produtos que precisam ser entregues
- Existem vários *modelos de processo de software* (ou *paradigmas de engenharia de software ou modelos de ciclo de vida*)
- Cada um representa uma tentativa de colocar ordem em uma atividade inerentemente caótica

PROCESSO DE DESENVOLVIMENTO

Só o processo não resolve!

Linguagem padrão

+

Ferramentas de apoio

Processo de desenvolvimento

+

+

Modelos, padrões e guias

+

Equipes treinadas

PROCESSO DE DESENVOLVIMENTO

Ingredientes de um processo

- Modelo de ciclo de vida
- Conjunto de atividades
  - Bem definidas
  - Com responsáveis
  - Com artefatos de entrada e saída
  - Com dependências entre as mesmas e ordem de execução

PROCESSO DE DESENVOLVIMENTO

Benefícios da utilização de processos

- Qualidade de software
- Produtividade no desenvolvimento, operação e manutenção de software
- Permitir ao profissional controle sobre o desenvolvimento dentro de custos, prazos e níveis de qualidade desejados
- Permitir ao profissional estimar custos e prazos com maior precisão
- Promove uma visão e cultura comum
- Captura e institucionaliza boas práticas

PROCESSO DE DESENVOLVIMENTO

Mas não é tão fácil...

- Os benefícios não virão de imediato!
- É necessário
  - Treinamento adequado
  - Adaptação da metodologia ao contexto no qual ela será utilizada
  - Apoio especializado para as equipes de desenvolvimento
  - Tempo para absorção da metodologia

PROCESSO DE DESENVOLVIMENTO

Maturidade de Processos

Processo IMATURO

- A maioria das organizações de software nessa situação atua como “bombeiros”
  - O fogo está sob controle?
  - Constantemente reativas – sem tempo para as melhorias
  - Os bombeiros se queimam
  - Seu único controle é: prevenção do incêndio

PROCESSO DE DESENVOLVIMENTO

Maturidade de Processos

Processo MADURO:

- É bem conhecido por todos os envolvidos
- Permite auditoria da fidelidade ao processo
- Propicia adoção disciplinada de tecnologias
- Os papéis e responsabilidades são claramente definidos
- Permite acompanhamento da qualidade do produto
- Permite acompanhamento da satisfação do cliente
- O cronograma, custo e qualidade são alcançados
- Há melhoria contínua do processo

PROCESSO DE DESENVOLVIMENTO

Maturidade de Processos

Processo MADURO:

- A organização possui uma infra-estrutura que efetiva e consistentemente aplica o processo
- Gerência deve “alimentar” a cultura de gestão – “se ninguém se importa, todo mundo esquece”
- Um processo institucionalizado resiste mesmo sem as pessoas que o definiram originalmente

PROCESSO DE DESENVOLVIMENTO

Maturidade de Processos

IMATURO

- Processo improvisado pelas pessoas
- Processo não é seguido ou cumprido
- Grande dependência dos atuais desenvolvedores
- Baixa visibilidade do processo para seu progresso e qualidade
- Funcionalidade e qualidade do produto comprometidas para atender prazo
- Custos excessivos de manutenção
- Tecnologia → Processo

MADURO

- Processo é definido, documentado e melhorado continuamente
- Processo é entendido, utilizado e "vivo"
- Processo suportado pela gerência
- Processo verificado e cumprido
- Grande visibilidade do processo alinhado ao negócio da organização
- Papéis e responsabilidades claramente definidas
- Processo → Tecnologia

PROCESSO DE DESENVOLVIMENTO

Processos e Qualidade de Software

- Morte à abordagem tradicional com teste final!!!
- Falta de foco no cliente = falta de qualidade
- Foco no processo
  - Não basta esperar o produto final
  - Bons processos → Bons produtos
  - Qualidade no trabalho → Qualidade no produto
  - Qualidade é parte do processo e responsabilidade de todos!
- Disciplina x criatividade
  - Melhorar/otimizar processos repetitivos que compõem a criação
  - Liberar a capacidade criadora

PROCESSO DE DESENVOLVIMENTO

Processos e Qualidade de Software

- Qualidade como parte do processo de desenvolvimento
  - Arquitetura testada
  - Testes unitários
  - Testes automatizados
  - Testes de aceitação
  - Revisão por pares
  - Análise estática de código
  - Integração contínua
  - Dividir para conquistar: pequenos releases
  - Feedback: releases constantes, validação com cliente
  - Etc etc etc...

MODELOS

Modelos de ciclo de vida de software

- Conjunto de fases, atividades, marcos e artefatos que guiam o desenvolvimento, operação e manutenção de um sistema
- Ferramenta para planejamento e gerenciamento!
- Não existem modelos certos ou errados, apenas adequados ou não a uma determinada situação

MODELOS

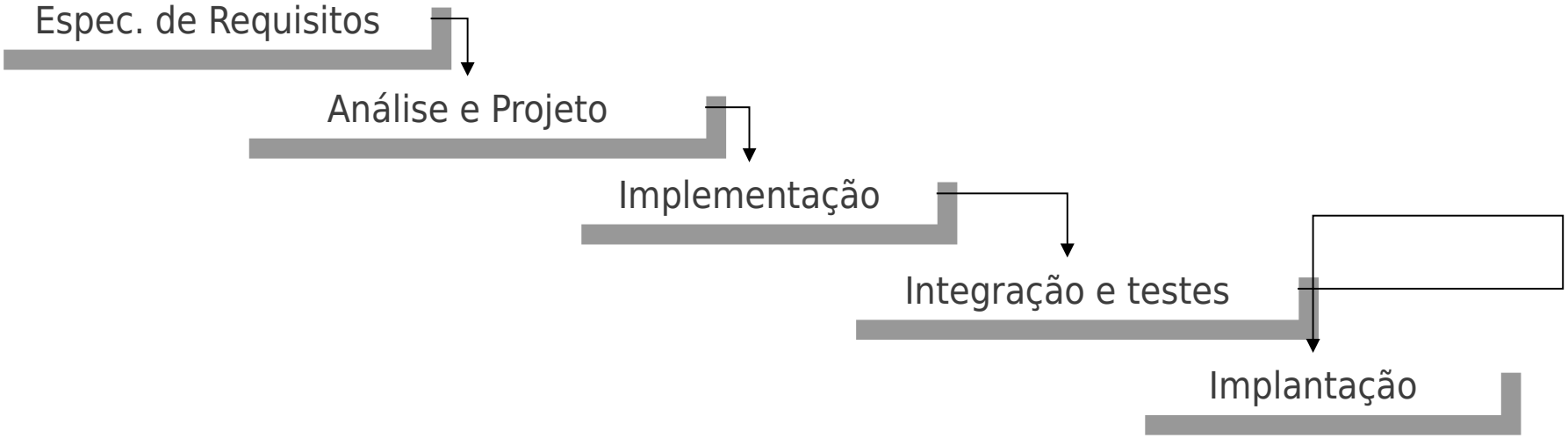
Modelos de ciclo de vida de software

- Força bruta, *code and fix*, *nike-way*
- Cascata
- Espiral
- Prototipagem
- Programação exploratória
- Iterativo
- …

MODELOS

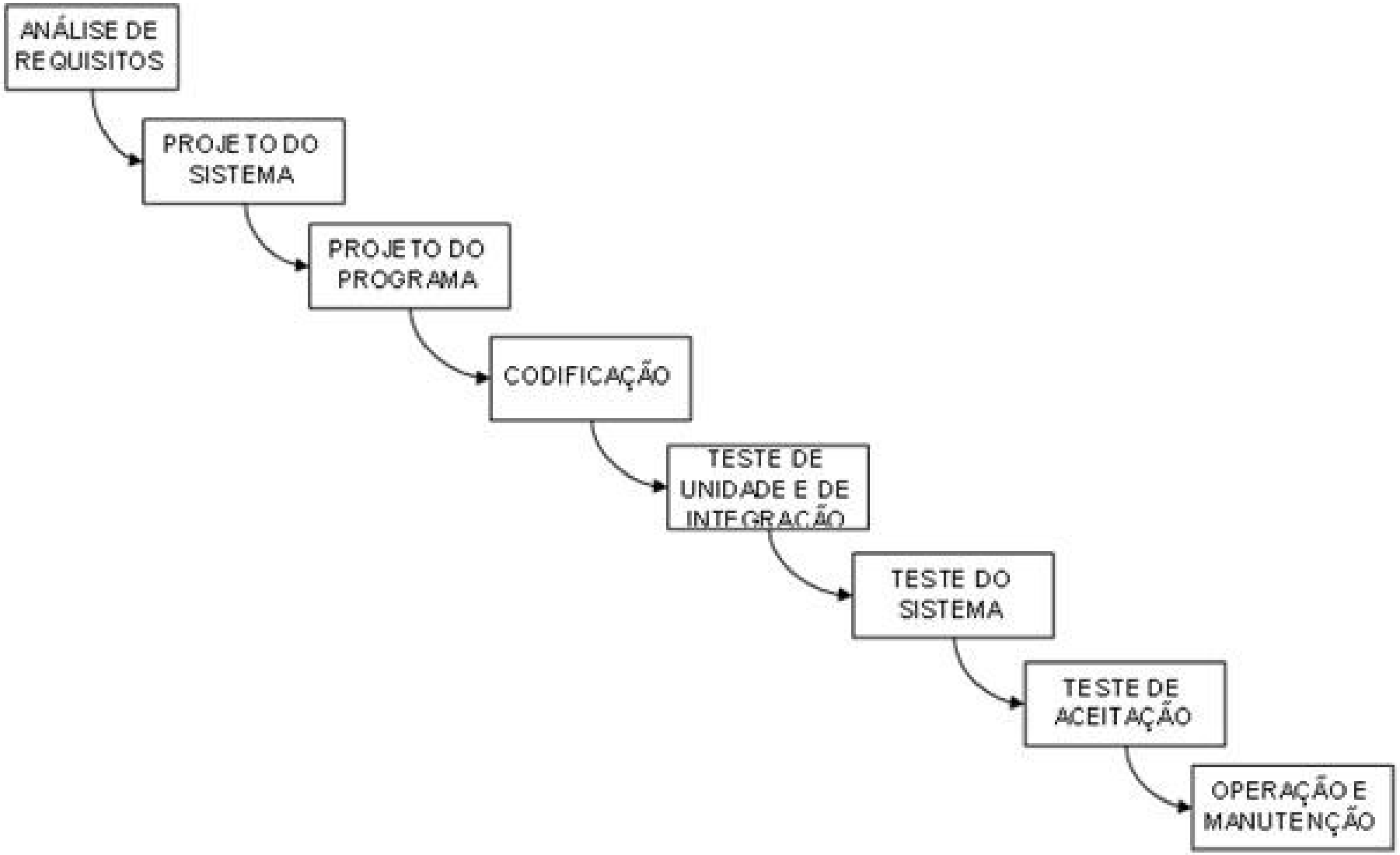
Modelo Cascata

- Um dos mais antigos, e ainda um dos mais usados!
- Várias atividades executadas de forma sistemática e seqüencial



Modelo Cascata

MODELOS



MODELOS

Modelo Cascata

- Fixa pontos específicos para a entrega de artefatos
- É simples e fácil de aplicar, facilitando o planejamento
- Na prática, existe uma interação entre as atividades e cada atividade pode levar a modificações nas anteriores
  - na maioria dos casos existe interação e superposição!
- Pressupõe que os requisitos ficarão estáveis
- Atrasa a redução de riscos

MODELOS

Modelo Cascata

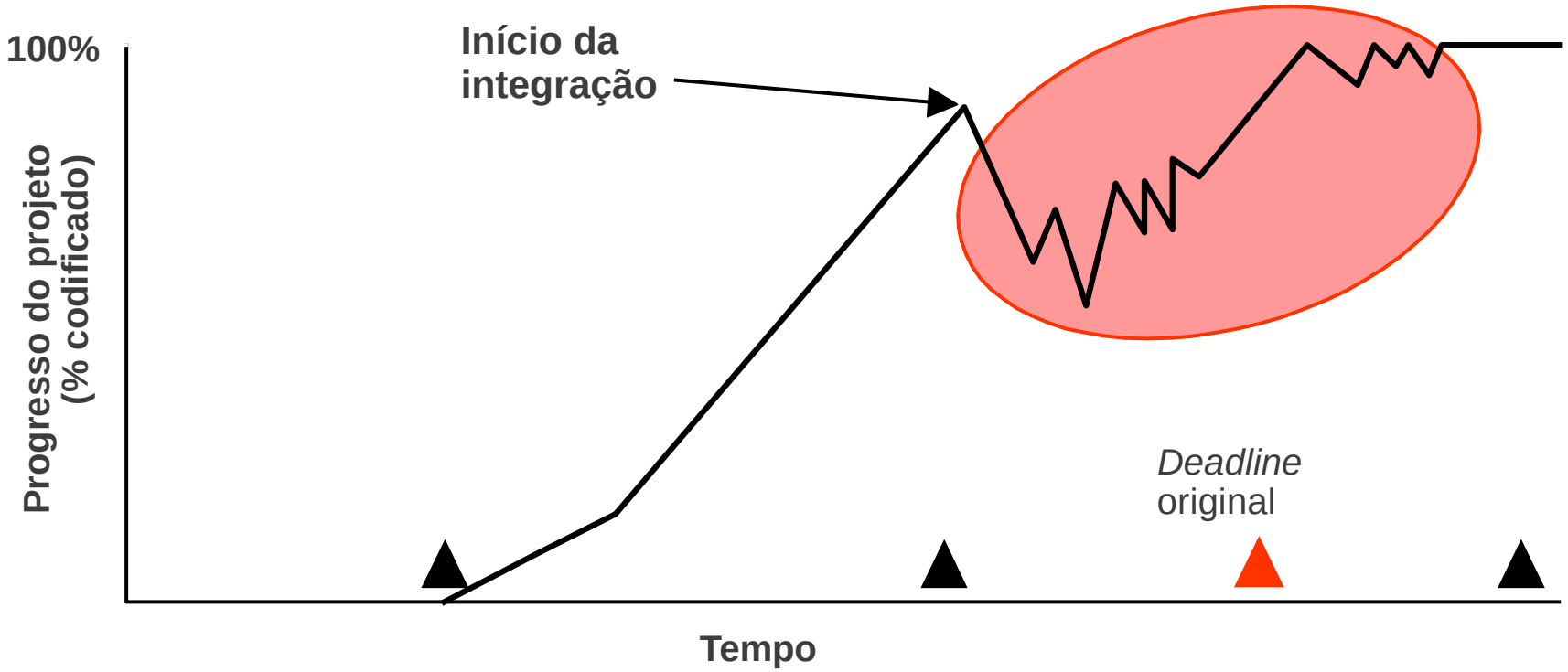
- *Fases*
  - Definição e análise de requisitos
  - Projeto do sistema e do software
  - Implementação e testes de unidade
  - Integração e testes do sistema
  - Operação e manutenção

- *Desvantagens*
  - Dificuldade de acomodar as mudanças após o processo ter sido iniciado
  - Particionamento inflexível do projeto em fases distintas
  - Dificuldade de responder a requisitos do usuário que mudam

  - **Portanto, esse modelo mais apropriado quando os requisitos são bem compreendidos**

MODELOS

Desenvolvimento cascata atrasa a redução de riscos

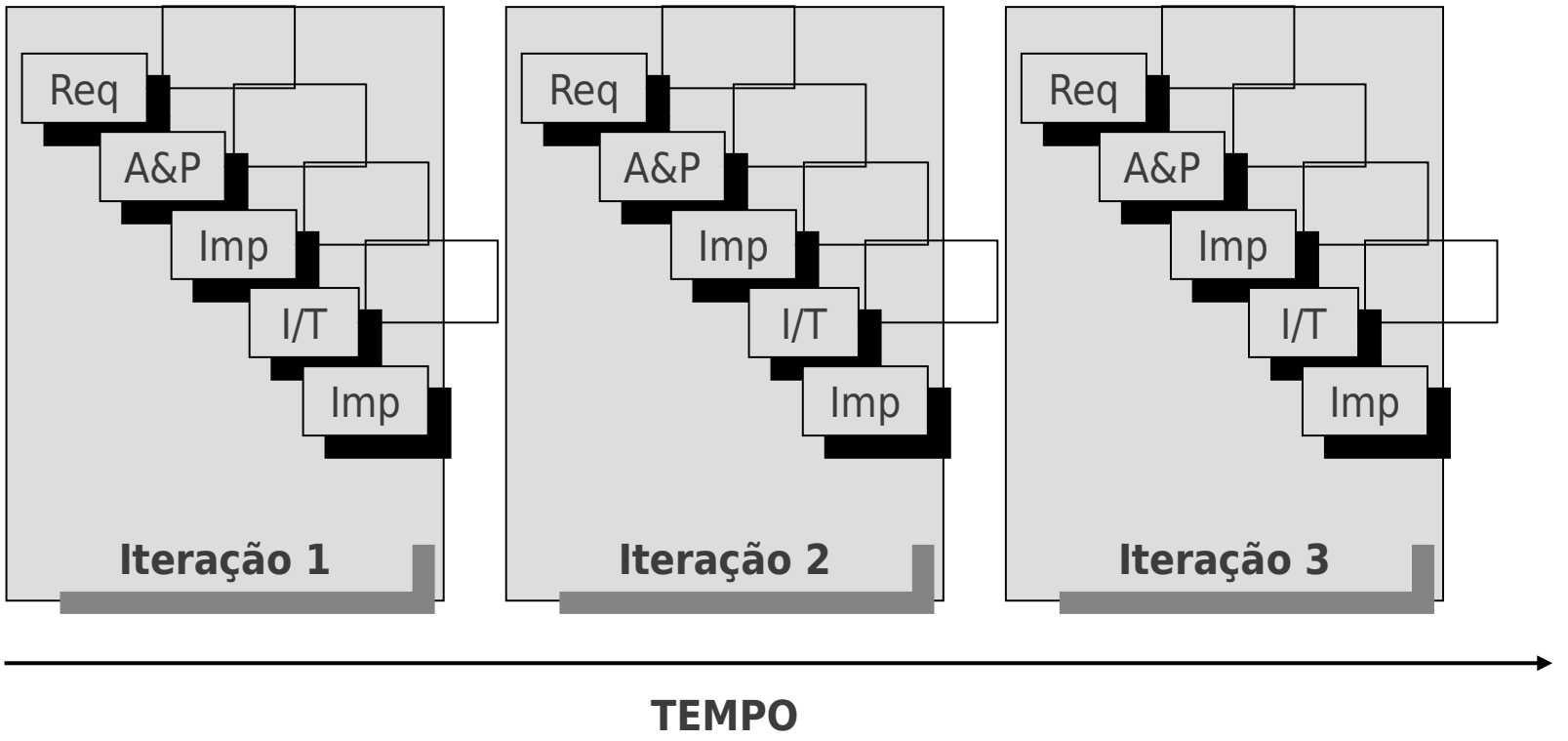


Fonte: Software Project Management, Walker Royce

MODELOS

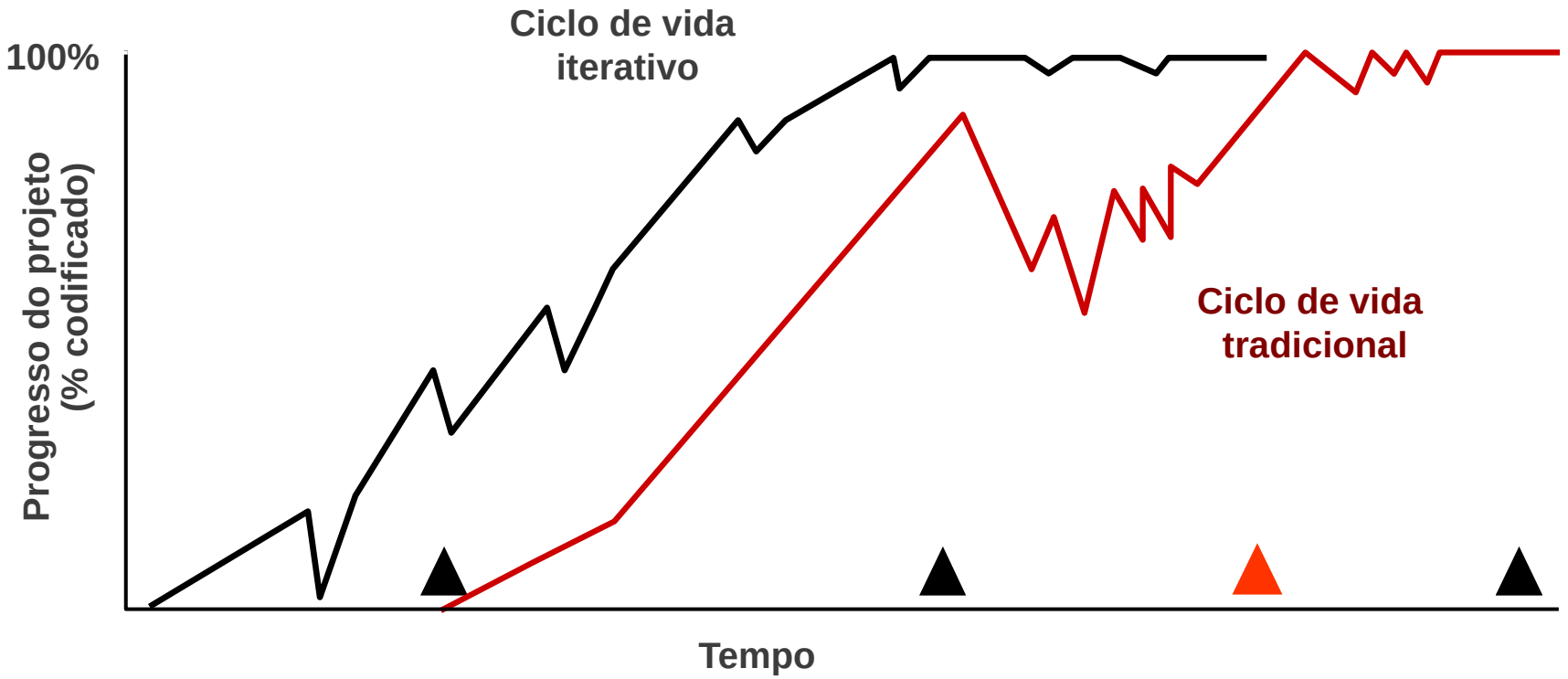
Modelo Iterativo

- Aplicação do modelo cascata iterativamente
- As iterações iniciais atacam os maiores riscos



MODELOS

Desenvolvimento iterativo antecipa a redução de riscos



Fonte: Software Project Management, Walker Royce

MODELOS

Modelo Iterativo

- Testes e integração são realizados desde o início, de forma contínua
- Riscos críticos são resolvidos antes que grandes investimentos sejam realizados
- Permite feedback dos usuários desde cedo
- Pequenos objetivos, foco em curto-prazo
- Progresso é medido de forma mais concreta
- Implementações parciais podem ser implantadas
- Utiliza as vantagens do modelo cascata, sem atrasar a resolução de riscos!

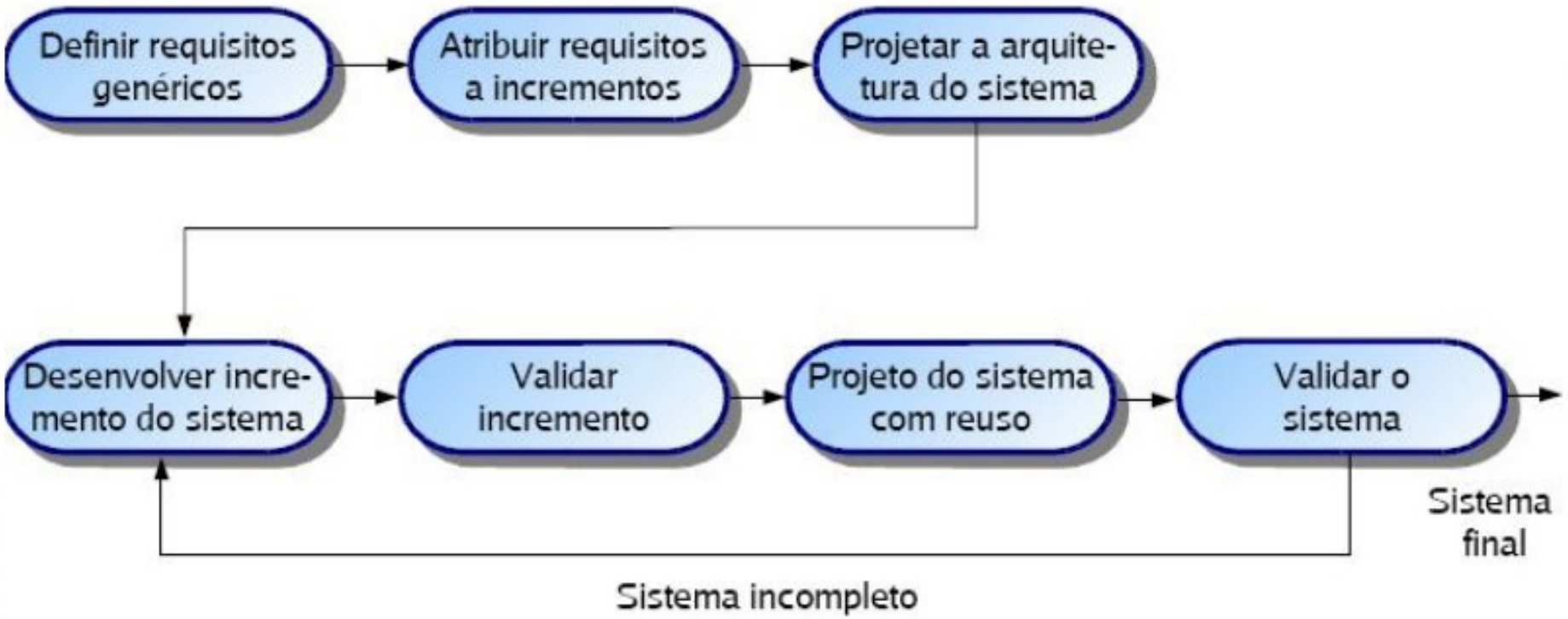
MODELOS

Desenvolvimento Incremental

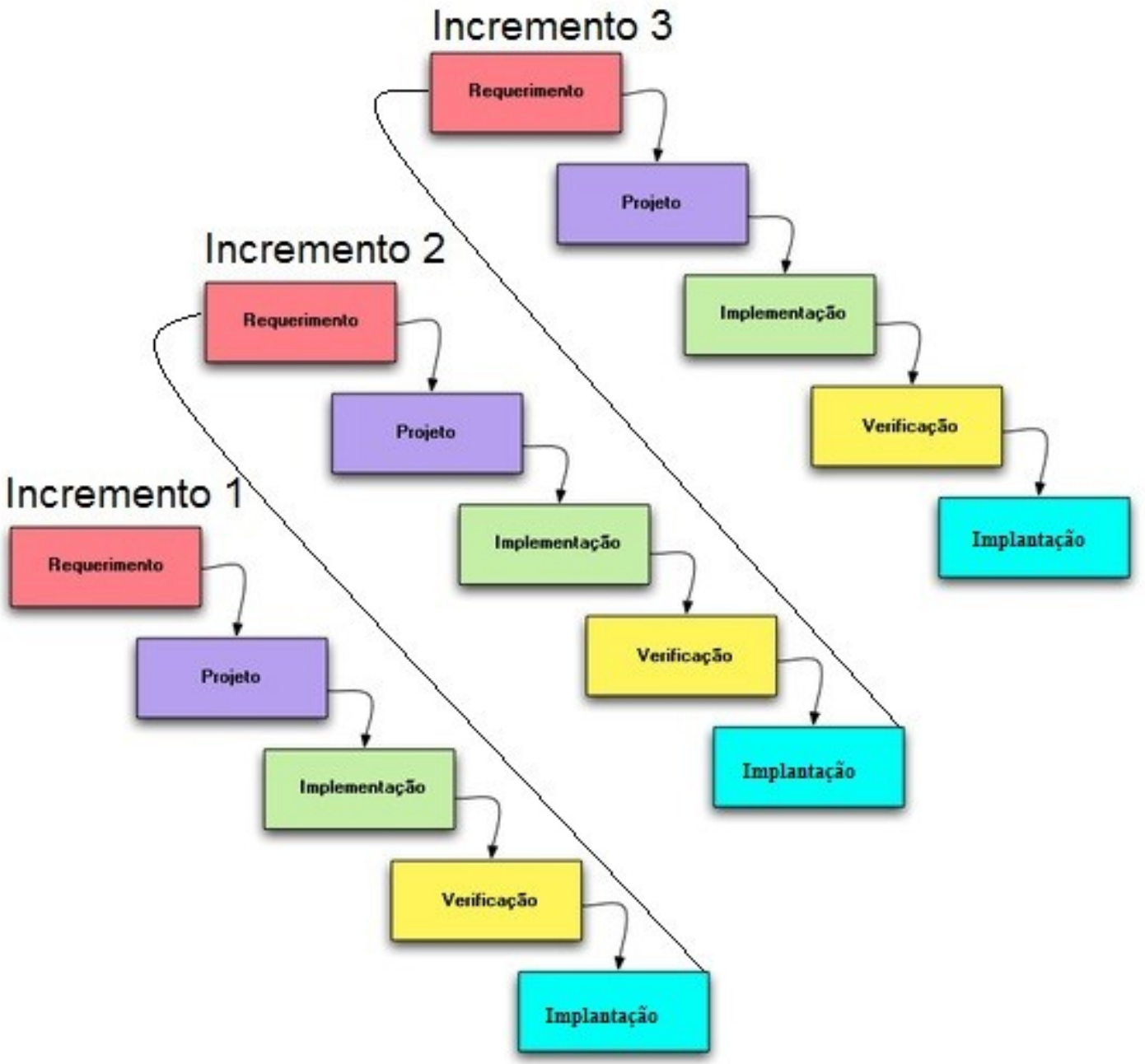
- *Também chamado desenvolvimento iterativo.*
- *Ao invés de entregar o sistema como uma única entrega, particiona-se o desenvolvimento e a entrega em incrementos, com cada incremento contendo parte da funcionalidade requerida*
- *Os requisitos do usuário são priorizados e os requisitos de prioridade mais alta são incluídos nos incrementos iniciais*
- *Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são congelados, ainda que os requisitos para incrementos posteriores continuem a evoluir*

MODELOS

Desenvolvimento Incremental



MODELOS

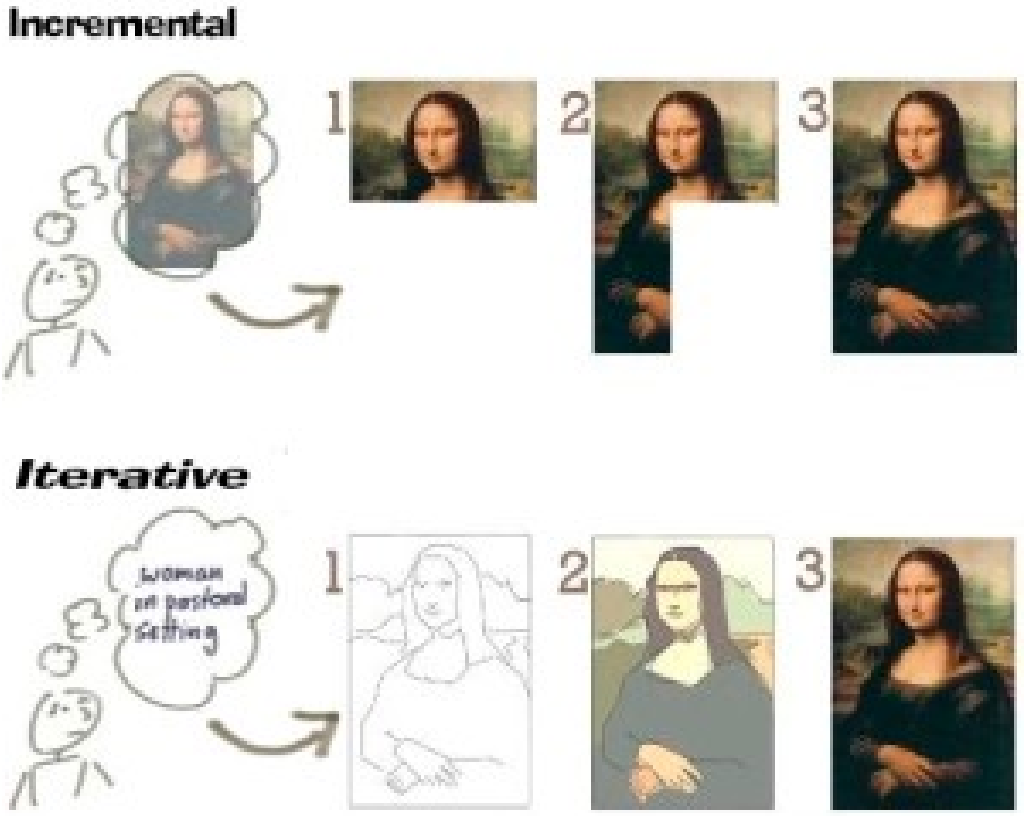

Incremento 3
Requerimento
Projeto
Implementação
Verificação
Implantação
Incremento 2
Requerimento
Projeto
Implementação
Verificação
Implantação
Incremento 1
Requerimento
Projeto
Implementação
Verificação
Implantação

MODELOS

Desenvolvimento Incremental

- *Vantagens*
  - *Cada incremento pode agregar valor para o cliente, portanto a funcionalidade do sistema está disponível mais cedo*
  - *Incrementos iniciais atuam como um protótipo para ajudar a descobrir requisitos para os incrementos posteriores*
  - *Menor risco de falha do projeto como um todo*
  - *Os serviços de mais alta prioridade do sistema tendem a receber a maior parte dos testes*

MODELOS



MODELOS

Desenvolvimento Evolucionário

- *Base:*
  - Desenvolver uma implementação inicial
  - Expor resultado ao comentário do usuário
  - Aprimoramento por meio de muitas versões

- *Existem 2 tipos de desenvolvimento evolucionário:*
  - Desenvolvimento exploratório
    - O objetivo é trabalhar com os clientes e evoluir um sistema final a partir de uma especificação genérica inicial. O desenvolvimento se inicia com as partes do sistema que estão compreendidas.
  - Fazer protótipos descartáveis
    - O objetivo é compreender os requisitos do sistema. O protótipo se concentra em fazer experimentos com partes dos requisitos que estejam mal compreendidas.

MODELOS

Programação Exploratória

- Idéia geral:
  - Desenvolvimento da primeira versão do sistema o mais rápido possível;
  - Modificações sucessivas até que o sistema seja considerado adequado;
  - Após o desenvolvimento de cada uma das versões do sistema ele é mostrado aos usuários para comentários.
- Adequado para o desenvolvimento de sistemas onde é difícil ou impossível de se fazer uma especificação detalhada do sistema;
- Principal diferença para os outros modelos é a ausência da noção de programa correto.

MODELOS

Programação Exploratória

- Tem sido mais usada no desenvolvimento de sistemas de inteligência artificial -- geralmente sistemas que tentam emular capacidades humanas;
- Em geral este tipo de desenvolvimento exige ferramentas de alto nível e máquinas poderosas e dedicadas;
- A maioria dos sistemas desenvolvidos com sucesso usando a programação exploratória foi implementada usando pequenos grupos de profissionais altamente qualificados e motivados.

MODELOS

Programação Exploratória

- É raramente usada no desenvolvimento de sistemas de grande porte e de vida longa. Principais razões:
  - Mudanças contínuas tendem a produzir sistemas cuja estrutura é desorganizada. Como conseqüência, a manutenção tende a ser mais difícil e cara.
  - O gerenciamento de projetos de software normalmente se baseia em modelos nos quais deve-se produzir relatórios regulares, usados para avaliar o progresso do projeto.
  - Como na programação exploratória o sistema é modificado com freqüência, não é razoável produzir muita documentação.
  - Ainda não se sabe como utilizar grandes grupos de pessoas eficientemente neste tipo de desenvolvimento.

MODELOS

Prototipagem descartável

- O objetivo é entender os objetivos do sistema. Começa com requisitos vagamente entendidos.
- Como na programação exploratória, a primeira fase prevê o desenvolvimento de um *programa* para o usuário experimentar.
  - No entanto, o objetivo aqui é estabelecer os requisitos do sistema.
  - O software *deve* ser reimplementado na fase seguinte.

MODELOS

Prototipagem descartável

- A construção de protótipos com os quais os usuários possam *brincar* é uma idéia bastante atrativa:
  - Para sistemas grandes e complicados.
  - Quando não existe um sistema anterior ou um sistema manual que ajude a especificar os requisitos.
- Os objetivos do protótipo devem estar bem claros *antes* do início da codificação. Possíveis objetivos:
  - Entender os requisitos dos usuários.
  - Definir a interface com os usuários.
  - Demonstrar a viabilidade do sistemas para os gerentes.

MODELOS

Prototipagem descartável

- Uma decisão importante a ser tomada é escolher o que será e o que não será parte do protótipo.
  - Não é economicamente viável implementar todo o sistema!
  - Os objetivos do protótipo são o ponto de partida.

MODELOS

Prototipagem: o que incluir no protótipo?

- Algumas possibilidades:
  - Implementar todas as funções do sistema mas com um número reduzido de detalhes.
  - Implementar um subconjunto das funções, possivelmente com um número maior de detalhes.
  - Desconsiderar requisitos associados a velocidade, espaço, confiabilidade, etc.
  - A menos que o objetivo do protótipo seja definir a interface com o usuário, desconsiderar a parte de manipulação de erros.

MODELOS

Prototipagem: possíveis vantagens

- Protótipos contribuem para melhorar a qualidade da especificação dos futuros programas, o que leva à diminuição dos gastos com manutenção.
- O treinamento dos usuários pode ser feito *antes* do *produto* ficar pronto.
- Partes do protótipo podem ser usadas no desenvolvimento do sistema final.

MODELOS

Prototipagem: possíveis desvantagens

- Em geral o grande argumento contra a construção de protótipos é o custo.
  - A construção do protótipo atrasa o início da implementação do sistema final.
    - Atrasos são um dos maiores problemas dos projetos de software.
    - Construir um protótipo pode não ser tão mais rápido assim do que construir o sistema final.
    - Se os ambientes utilizados forem diferentes este custo será um custo extra.

MODELOS

Prototipagem: possíveis desvantagens

- O cliente vê algo que parece ser uma versão do software desejado e não entende porque o produto precisa ser reconstruído.
  - A tendência é o cliente *exigir* que *pequenos acertos* sejam feitos para que o protótipo se transforme no sistema final.
  - Freqüentemente a gerência cede ...
- Muitas das *concessões* feitas na implementação do protótipo visando a construção rápida podem vir a fazer parte do sistema final.
  - Utilização de linguagens, ferramentas, algoritmos, etc. que sejam inadequados e/ou ineficientes.

MODELOS

Desenvolvimento Evolucionário

- Desvantagens
  - *Falta de visibilidade do processo*
  - *Os sistemas frequentemente possuem pouca estrutura*
  - *Podem ser exigidas habilidades especiais (por exemplo, em linguagens para desenvolvimento rápido)*

- Aplicabilidade
  - *Para sistemas interativos pequenos ou de médio porte*
  - *Para partes de sistemas grandes (por exemplo, a interface com o usuário)*
  - *Para sistemas de vida curta*

MODELOS

O Modelo Espiral

- Foi criado visando abranger as melhores características do modelo clássico e da prototipagem.
- Acrescenta aspectos gerenciais ao processo de desenvolvimento de software.
  - análise de riscos em intervalos regulares do processo de desenvolvimento de software;
  - planejamento;
  - controle;
  - tomada de decisão.

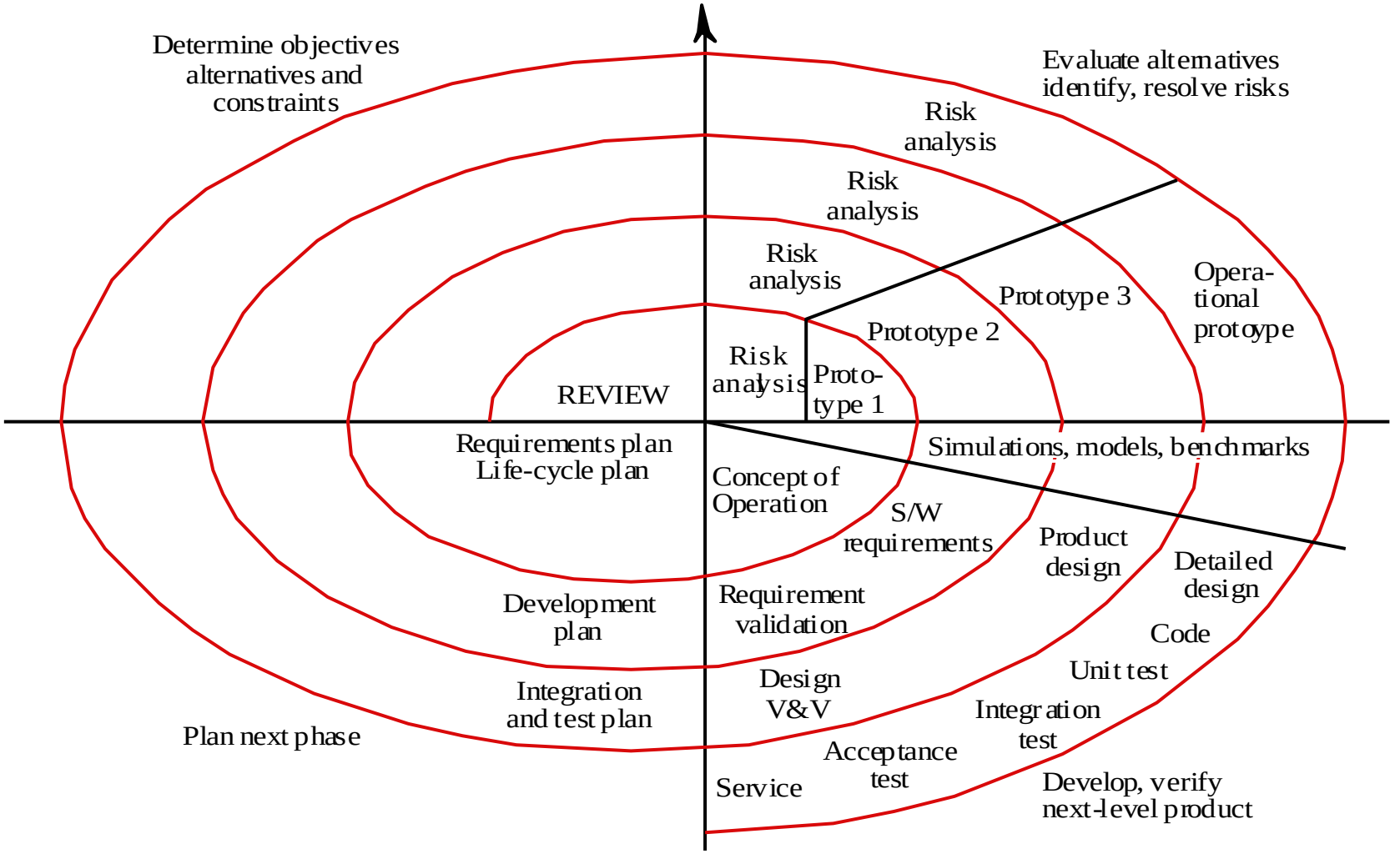
MODELOS

Fases do modelo espiral

- Definição dos objetivos
  - Objetivos específicos para a fase do projeto são identificadas
- Avaliação e redução do risco
  - Os riscos principais são identificados, analisados e busca-se informações para reduzir estes riscos
- Desenvolvimento e validação
  - Escolha de um modelo apropriado para a fase do desenvolvimento
- Planejamento
  - O projeto é revisto e planos são feitos para o próximo passo da espiral

MODELOS

O Modelo Espiral



MODELOS

O Modelo Espiral

- Observações:
  - De um certo modo este modelo é semelhante à programação exploratória.
  - A cada ciclo da espiral, versões progressivamente mais completas do software são construídas;
  - Antes de cada ciclo, uma análise de riscos é feita;
  - Ao fim de cada ciclo é feita uma avaliação se deve-se prosseguir para o próximo ciclo.

MODELOS

Modelo Espiral – 1º Quadrante

- *Um ciclo se inicia com a tarefa "Determinação de objetivos, alternativas e restrições"*
- *Objetivos principais*
  - comprometimento dos envolvidos
  - estabelecimento de uma estratégia para alcançar os objetivos da fase que se inicia

MODELOS

Modelo Espiral – 2º Quadrante

- *Na segunda tarefa, "Avaliação de alternativas, identificação e solução de riscos", executa-se uma análise de risco.*
- *Protótipos são uma forma de avaliar riscos.*
- *Objetivos principais*
  - detectar riscos
  - avaliar soluções que ofereçam menor risco de implementação
  - adotar atividades para reduzir os ricos principais
- *Se o risco for considerado inaceitável, o projeto pode ser encerrado.*

MODELOS

Modelo Espiral – 3º Quadrante

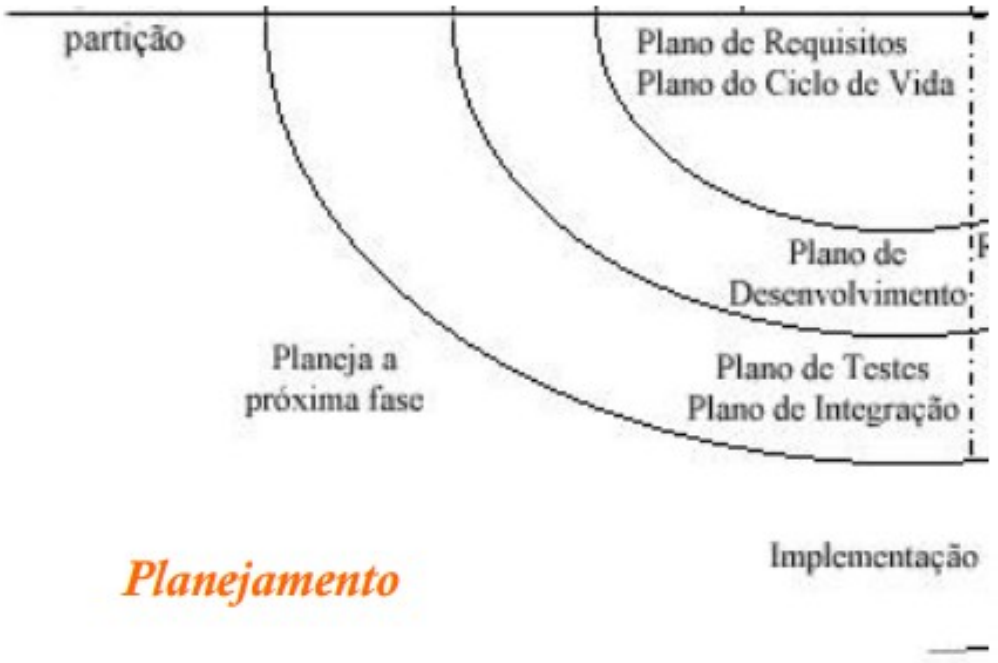
- *Na terceira tarefa ocorre o desenvolvimento do produto.*
- *Deve ser escolhido um modelo de desenvolvimento de software específico*

- *Objetivos principais*
  - definir e validar os requisitos
  - projetar o software
  - projetar a validação e verificação
  - codificar
  - realizar testes
    - *integração*
    - *unidade*
    - *aceitação*

MODELOS

Modelo Espiral – 4º Quadrante

- *Na quarta tarefa o produto é avaliado e se prepara para iniciar um novo ciclo*
- *O projeto é revisado e a próxima fase da espiral é planejada*
- *Objetivos principais*
  - planejar requisitos
  - planejar ciclo de vida
  - planejar desenvolvimento
  - planejar integração e testes



MODELOS

Vantagem do modelo espiral

- Foca atenção nas opções de reuso
- Foca atenção em eliminação precoce de erros
- Qualidade desde o início
- Integra desenvolvimento e manutenção

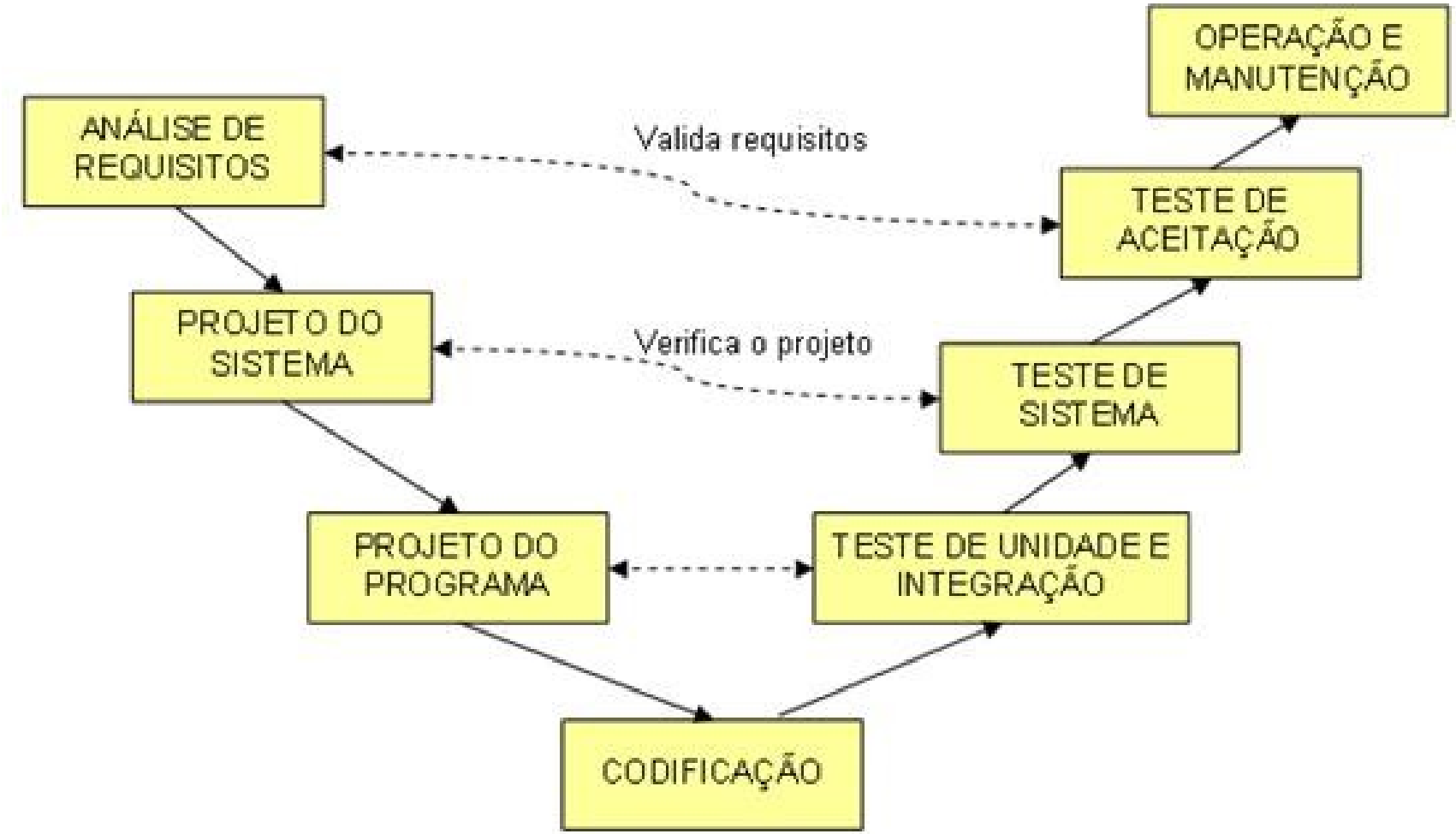
MODELOS

Problemas do modelo espiral

- Contrato de desenvolvimento geralmente já especifica o modelo de processo e produtos (deliverables)
- É difícil convencer gerentes de que todo este processo é controlável
- Requer experiência em avaliação de risco
- Precisa de refinamento para uso geral

MODELOS

Modelo V



MODELOS

Modelos de Processos Prescritivos

- Visam assegurar a aderência às normas e padrões estabelecidos
- Processos com forte ênfase em controles e documentações
- Voltados para equipes grandes e/ou dispersas
- Em geral, especificam várias atividades além do desenvolvimento de software
- Buscam satisfazer o maior número possível de contextos, para serem usados como o único processo
- Atendem a muitas exigências do CMMI e ISO
- Geralmente são "personalizados" para cada projeto e/ou empresa

MODELOS

Alguns Modelos de Processos Prescritivos

- Unified Process (UP/RUP) e suas instanciações (EUP – *Enterprise Unified Process*)
- MSF (*Microsoft Solutions Framework*)
- ICONIX
- CATALISYS
- OPEN (*Object-oriented Process, Environment and Notation*)

MODELOS

Modelos de Processos Ágeis

- ***Agilidade*** é a habilidade tanto para criar quanto para responder à mudança, de forma a obter lucro em um ambiente turbulento de negócios
- Visam diminuir a quantidade de documentação
- Têm o foco nas pessoas e suas interações

Alguns Modelos de Processos Ágeis

- XP (*Extreme Programming*), SCRUM
- AUP, Família CRYSTAL
- FDD (*Feature-Driven Development*)
- DSDM (*Dynamic Systems Development Method*)
- ASD (*Adaptive Software Development*)

PROCESSO DE DESENVOLVIMENTO

Processo de *software*

- Conjunto estruturado de actividades necessárias para desenvolver sistema de *software*
  - Especificação
  - Desenho
  - Validação
  - Evolução
- Modelo de processo de *software* é representação abstracta de processo, descrevendo-o sob um ponto de vista particular

PROCESSO DE DESENVOLVIMENTO: RUP

RUP – Rational Unified Process

PROCESSO DE DESENVOLVIMENTO: RUP

Rational Unified Process - RUP

- Desenvolvido pela Rational (atualmente parte da IBM)
- Originário da Objectory
- É vendido como um produto
- Página: http://www-306.ibm.com/software/br/rational/rup.shtml
- Atualmente faz parte do IBM Rational Method Composer
- Trial: http://www-128.ibm.com/developerworks/downloads/r/rup/
- Outro link: www.wthreex.com/rup/

PROCESSO DE DESENVOLVIMENTO: RUP

Rational Unified Process - RUP

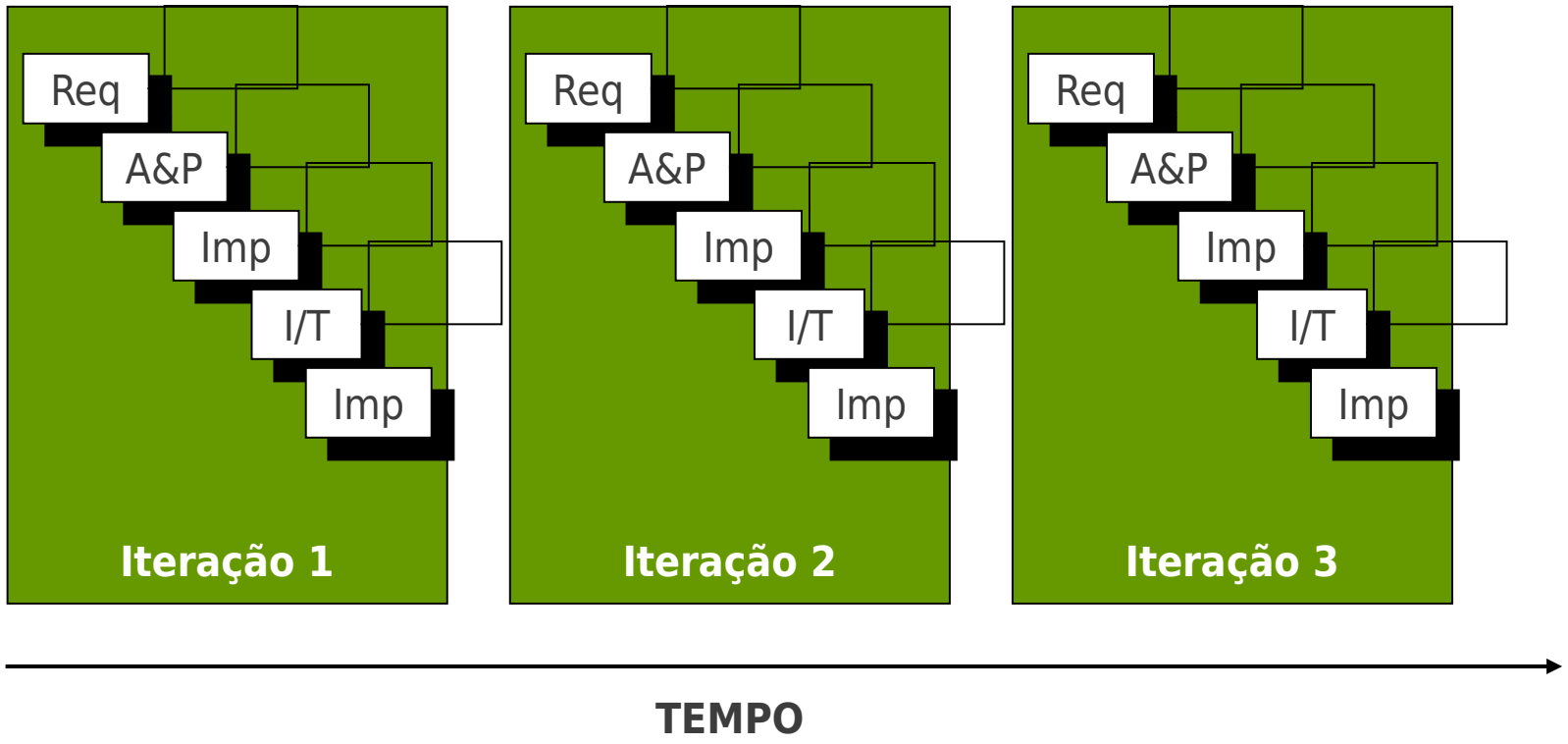
- Desenvolvido pela Rational (atualmente parte da IBM)
- Originário da Objectory
- É vendido como um produto
- Página: http://www-306.ibm.com/software/br/rational/rup.shtml
- Atualmente faz parte do IBM Rational Method Composer
- Trial: http://www-128.ibm.com/developerworks/downloads/r/rup/
- Outro link: www.wthreex.com/rup/

Características 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
  - Orientado a objetos

PROCESSO DE DESENVOLVIMENTO: RUP

Iterativo e incremental



PROCESSO DE DESENVOLVIMENTO: RUP

Iterativo e incremental

- Em cada iteração:
  - são identificados e especificados os casos de uso mais relevantes
  - é feita a análise e projeto dos casos de uso, usando-se a arquitetura como guia
  - são implementados componentes que realizam o que foi projetado
  - verifica-se se os componentes satisfazem os casos de uso escolhidos

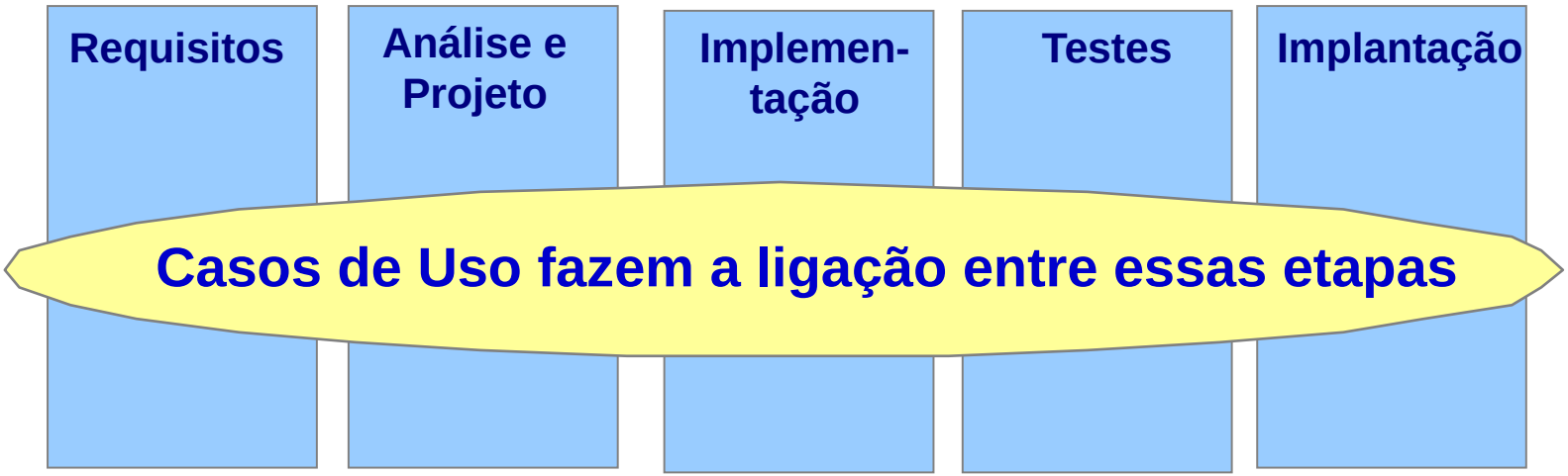
PROCESSO DE DESENVOLVIMENTO: RUP

Guiado por casos de uso

- Casos de uso são usados para especificar requisitos
- Durante a análise, projeto e implementação os casos de uso são “realizados”
- Durante os testes, verifica-se se o sistema realiza o que está descrito no Modelo de Casos de Uso
- Casos de uso são usados no planejamento e acompanhamento das iterações

PROCESSO DE DESENVOLVIMENTO: RUP

Casos de uso são usados durante todo o processo



PROCESSO DE DESENVOLVIMENTO: RUP

Baseado na arquitetura do sistema

- A arquitetura é prototipada e definida logo nas primeiras iterações
- A arquitetura guia o projeto e implementação das diversas partes do sistema
- A arquitetura serve para organizar o desenvolvimento, estruturar a solução e identificar oportunidades de reuso
- Os casos de uso dizem o que deve ser feito e a arquitetura descreve como

PROCESSO DE DESENVOLVIMENTO: RUP

Orientado a objetos

- Análise e Projeto em UML
  - UML é uma linguagem usada para especificar, modelar e documentar os artefatos de um sistema
  - É um padrão da OMG e têm se tornado o padrão empresarial para modelagem OO

PROCESSO DE DESENVOLVIMENTO: RUP

Principais elementos do RUP

- Conceitos
  - Fases e Iterações
  - Disciplinas
  - Atividades
  - Artefatos
  - Responsáveis

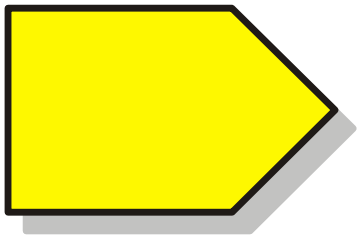
Principais elementos do RUP

- **Quem** está fazendo **o quê**, **quando** e **como** para alcançar um objetivo em particular
  - Quem : Papéis
  - O Quê : Artefatos
  - Como : Atividades
  - Quando : Fases e Iterações

PROCESSO DE DESENVOLVIMENTO: RUP

Atividades

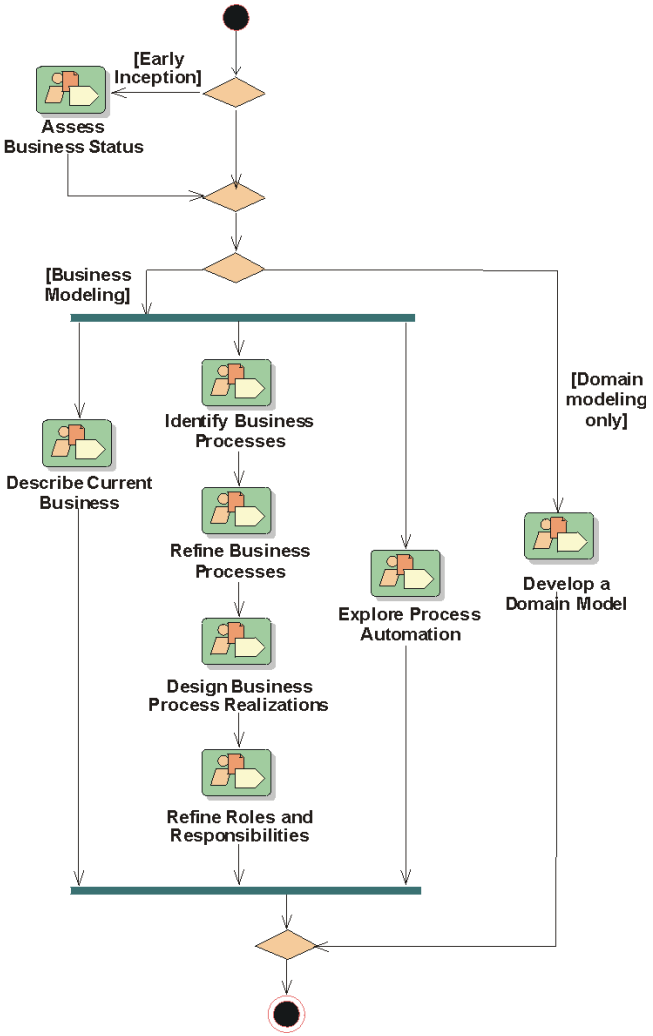
- Unidade de trabalho
- Composta de:
  - Objetivos
  - Passos
  - Entradas e saídas
  - Responsável
  - Guias e padrões



PROCESSO DE DESENVOLVIMENTO: RUP

Disciplinas

- Agrupam atividades correlacionadas
- Também chamadas “fluxos de atividades”
- Cada disciplina possui um workflow composto de detalhes que são definidos por ativi



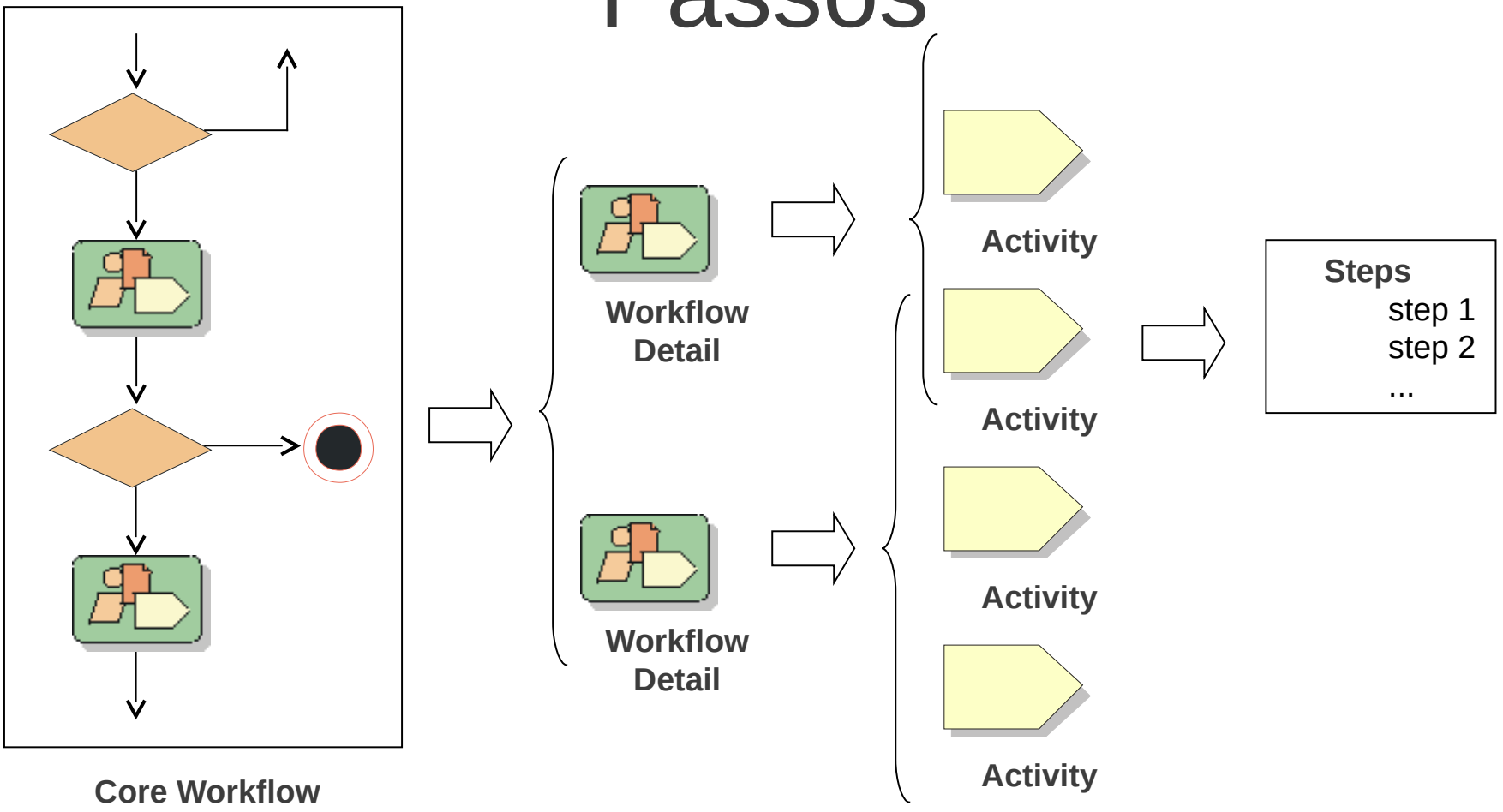
PROCESSO DE DESENVOLVIMENTO: RUP

Disciplinas

- Principais
  - Modelagem de Negócio
  - Análise de Requisitos
  - Análise e Projeto
  - Implementação
  - Testes
  - Implantação
- Auxiliares
  - Gerência de Alterações
  - Gerência de Projeto
  - Ambiente

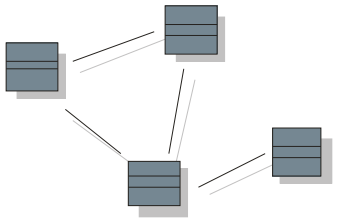
PROCESSO DE DESENVOLVIMENTO: RUP

Workflows, Atividades e Passos

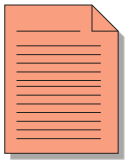


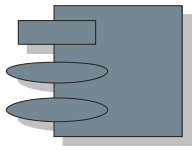
PROCESSO DE DESENVOLVIMENTO: RUP

Artefatos



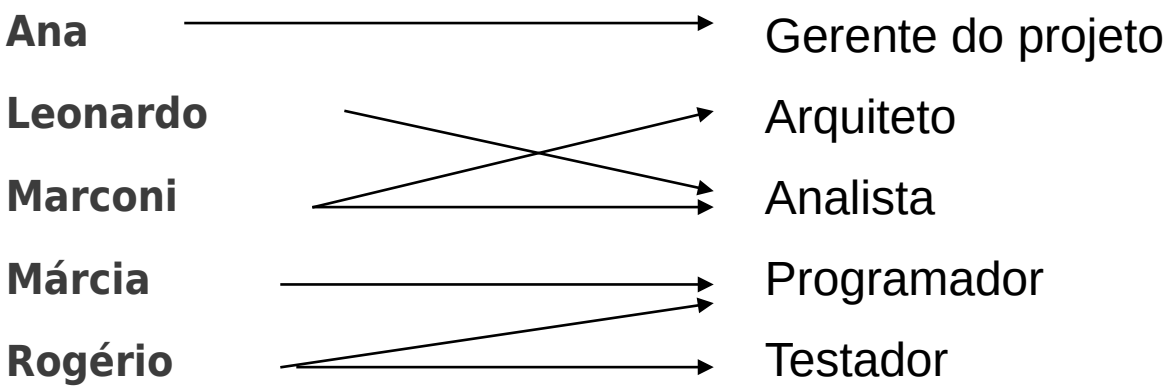
- Entradas e saídas das atividades
- Uma atividade precisa de artefatos (entrada) para produzir outros artefatos (saída)
  - Ex: a atividade “Implementar Componente” produz componentes de software (artefatos de saída) a partir dos projetos de componentes (artefatos de entrada)
- Possuem modelos para
  - indicar como devem ser feitos
  - padronizar os formatos dos documentos





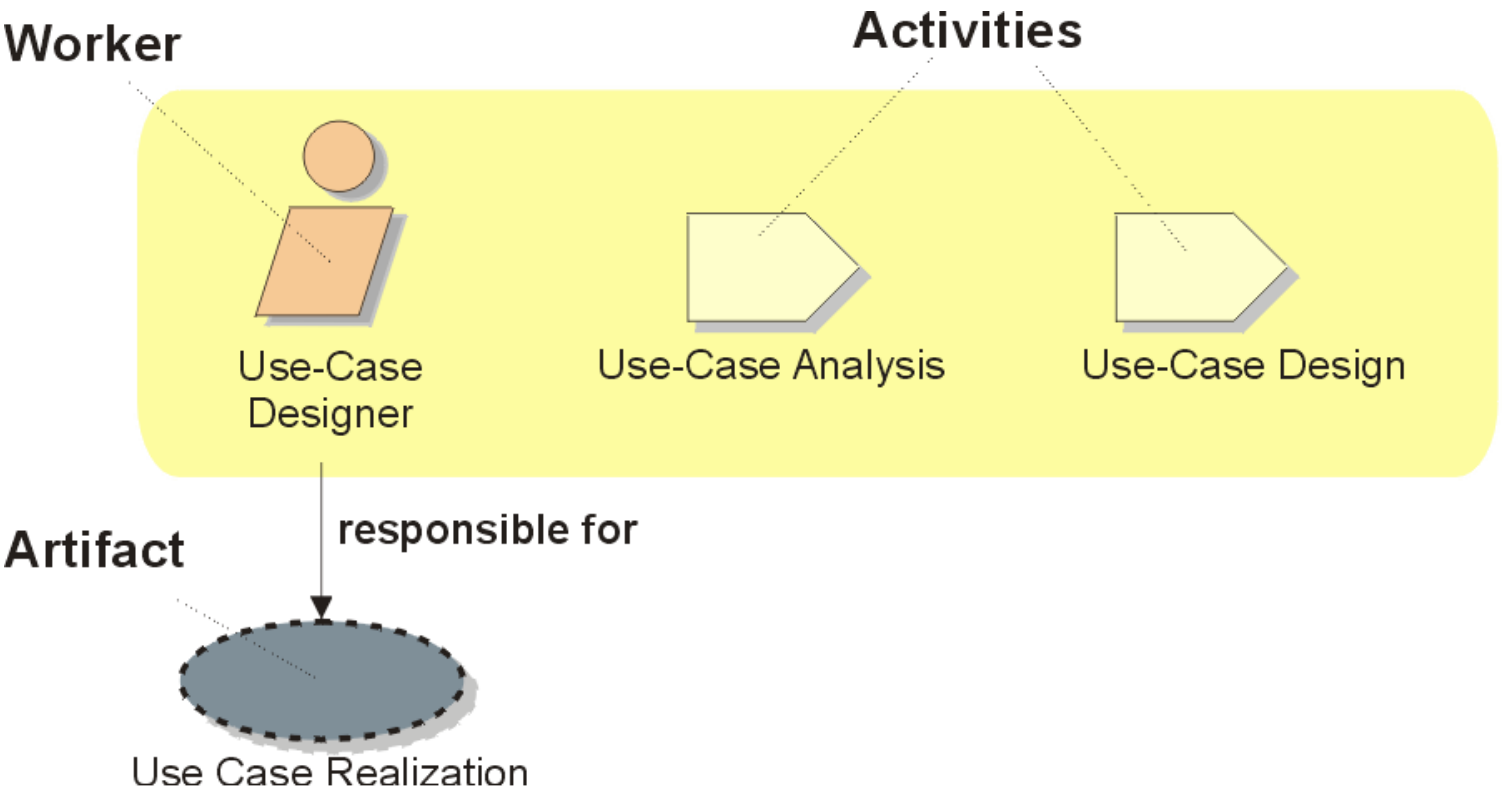
PROCESSO DE DESENVOLVIMENTO: RUP

Responsáveis



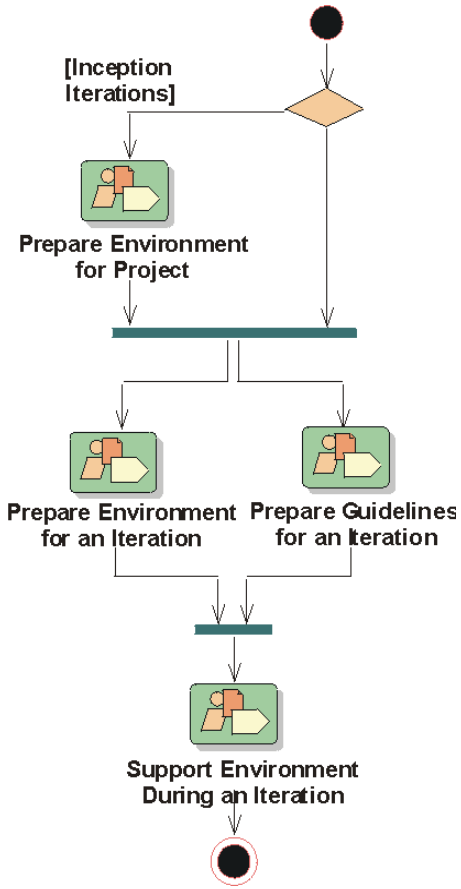
PROCESSO DE DESENVOLVIMENTO: RUP

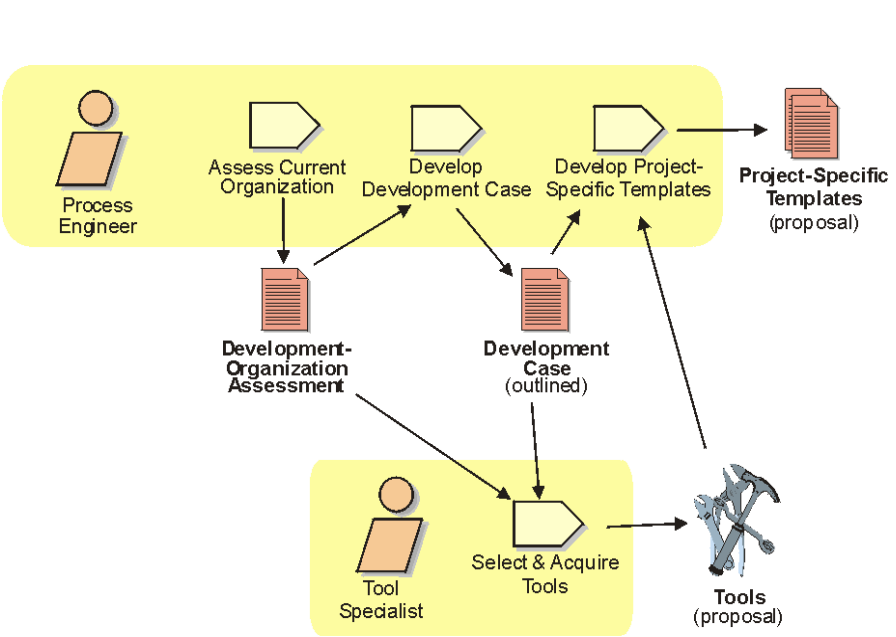
Atividades, Responsáveis e Artefatos



PROCESSO DE DESENVOLVIMENTO: RUP

Workflows, Atividades, Responsáveis e Artefatos

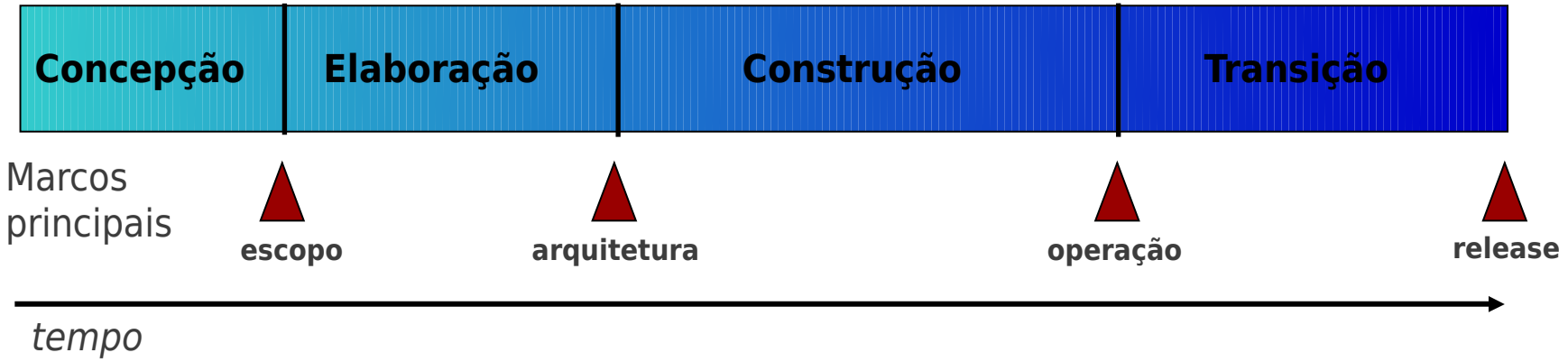




PROCESSO DE DESENVOLVIMENTO: RUP

Fases e iterações

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

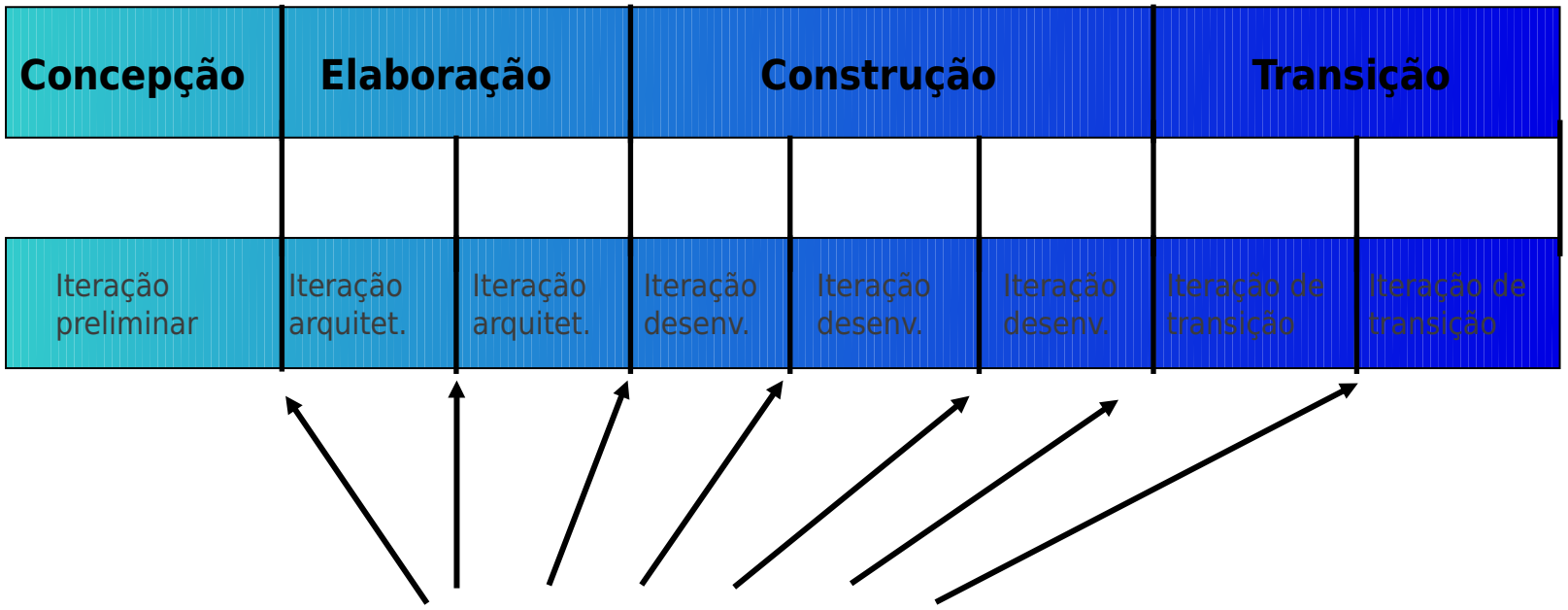


- As fases indicam a maturidade do sistema!

PROCESSO DE DESENVOLVIMENTO: RUP

Fases e iterações

- Cada fase é dividida em iterações:

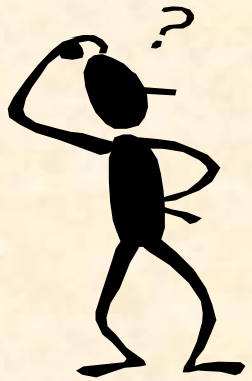


PROCESSO DE DESENVOLVIMENTO: RUP

Fases e iterações

Concepção

Estabelecer o escopo e viabilidade econômica do projeto

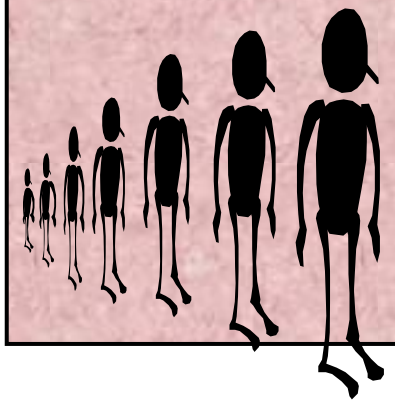


Elaboração

Eliminar principais riscos e definir arquitetura estável

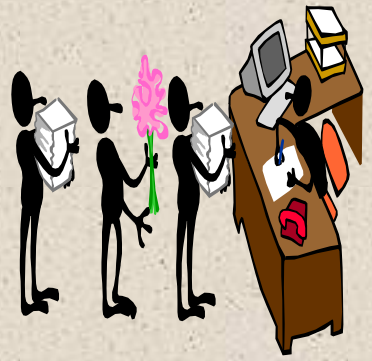
Construção

Desenvolver o produto até que ele esteja pronto para beta testes



Transição

Entrar no ambiente do usuário



PROCESSO DE DESENVOLVIMENTO: RUP

Fases e Iterações

- Fase de Concepção

"Alcançar a concordância dos gestores nos objetivos do ciclo de vida do projeto"

  - Objetivos
    - Garantir a cooperação dos gestores
    - Escopo do projeto
    - Sintetizar arquitetura candidata
    - Estimar riscos potenciais
    - Preparar o ambiente de suporte
  - Atividades
    - Formular o escopo do projeto
    - Planejar e Preparar o Caso de Negócio
    - Sintetizar uma arquitetura candidata
    - Preparar o ambiente para o Projeto
  - Marco : Objetivos do ciclo de vida

PROCESSO DE DESENVOLVIMENTO: RUP

Fases e Iterações

- Fase de Elaboração

"Fornecer uma base arquitetural estável para o esforço de projeto e implementação"

  - Objetivos
    - Assegurar a estabilidade da arquitetura e requisitos
    - Mapear todos os riscos significantes
    - Produção de protótipos evolucionários
  - Atividades
    - Definir, validar e estabilizar a arquitetura
    - Refinar a Visão do sistema
    - Refinar o Caso de Desenvolvimento
    - Refinar a arquitetura e selecionar componentes
  - Marco : Ciclo de vida da arquitetura

PROCESSO DE DESENVOLVIMENTO: RUP
Fases e Iterações

- Fase de Construção

  "Esclarecer os requisitos remanescentes e completar o desenvolvimento na arquitetura proposta"

  - Objetivos
    - Minimizar os custos de desenvolvimento
    - Alcançar a qualidade do produto
    - Alcançar versões úteis do sistema
    - Verificação se os usuários estão prontos para o sistema
  - Atividades
    - Gerenciamento de recursos, controle e processo.
    - Completar o desenvolvimento de componentes
    - Avaliar os releases segundo os critérios do Visão
  - Marco : Capacidades operacionais básicas

PROCESSO DE DESENVOLVIMENTO: RUP

Fases e Iterações

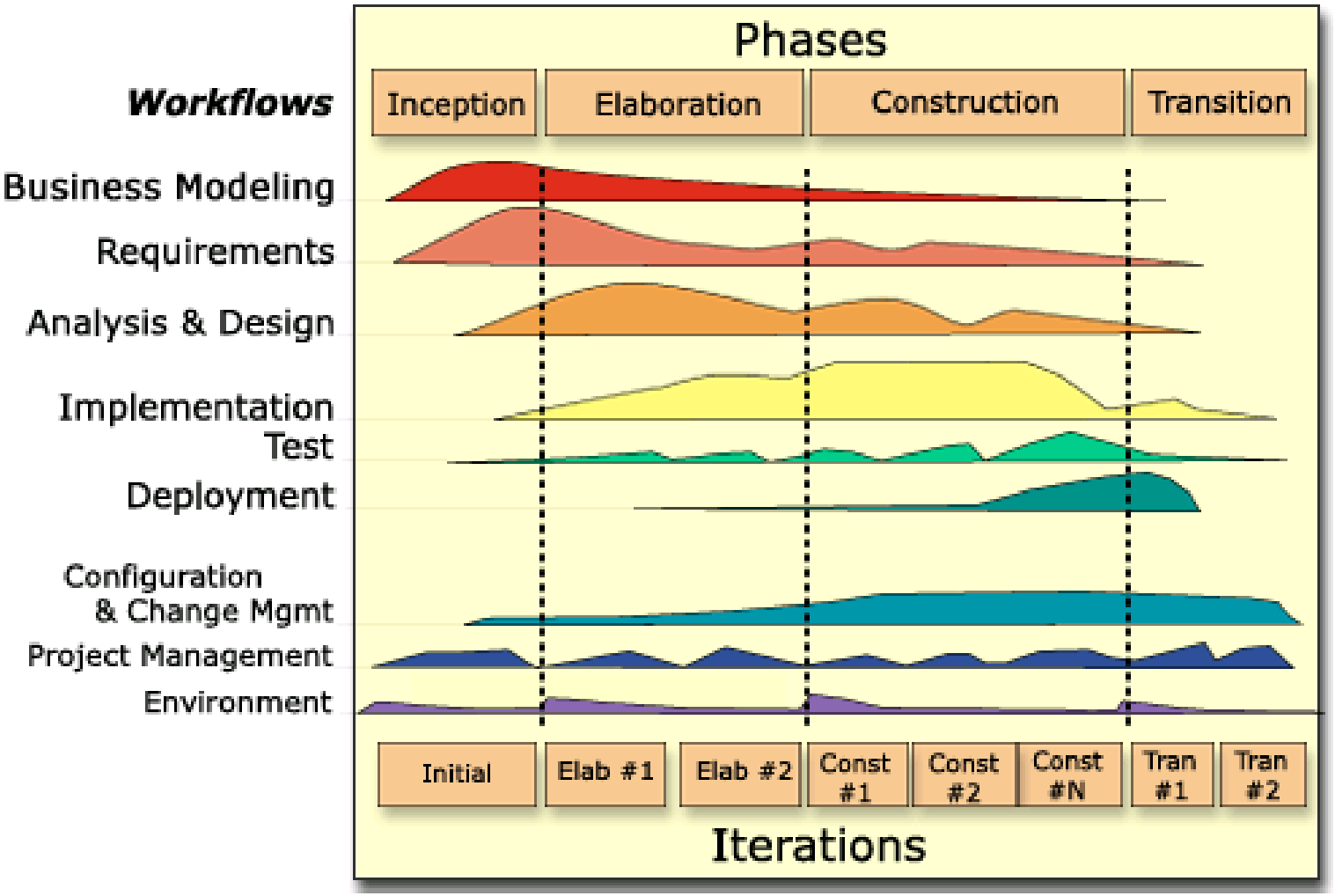
- Fase de Transição

  "Assegurar que o software está disponível para os usuários"

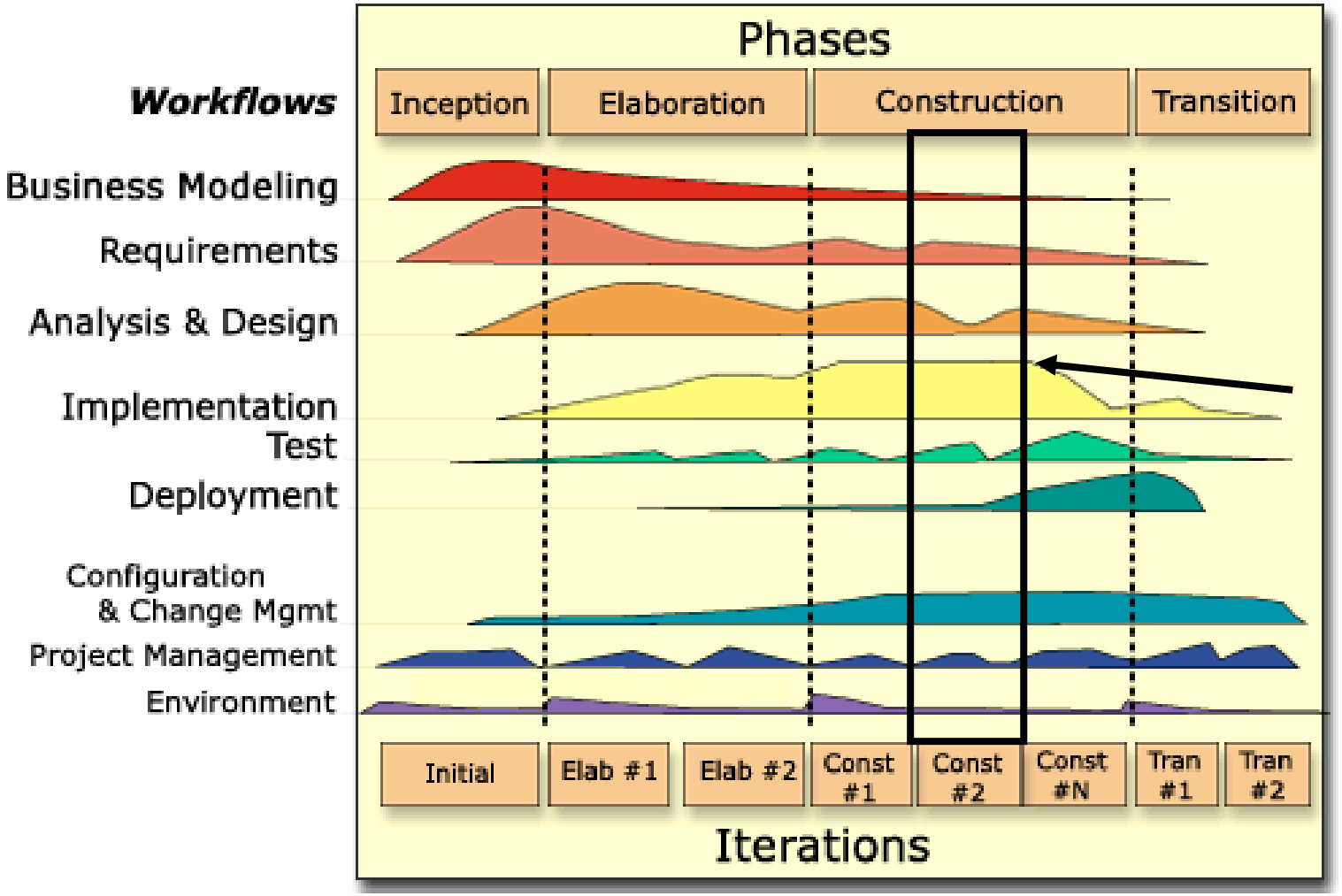
  - Objetivos
    - Testes beta para validar o sistema
    - Conversão de bases de dados (legado)
    - Treinamentos
    - Performance
  - Atividades Principais
    - Executar planos de implantação
    - Finalizar o material de suporte
    - Obter feedback do usuário
    - Ajustar o produto final
  - Marco : Release do produto

PROCESSO DE DESENVOLVIMENTO: RUP

Fases, iterações e disciplinas

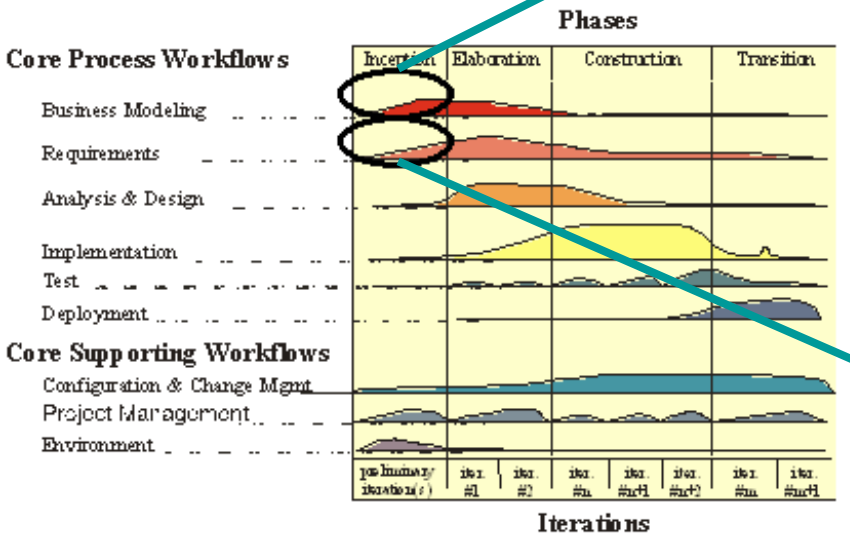


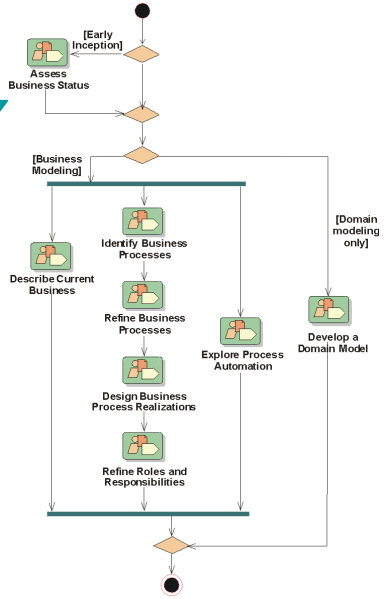
PROCESSO DE DESENVOLVIMENTO: RUP



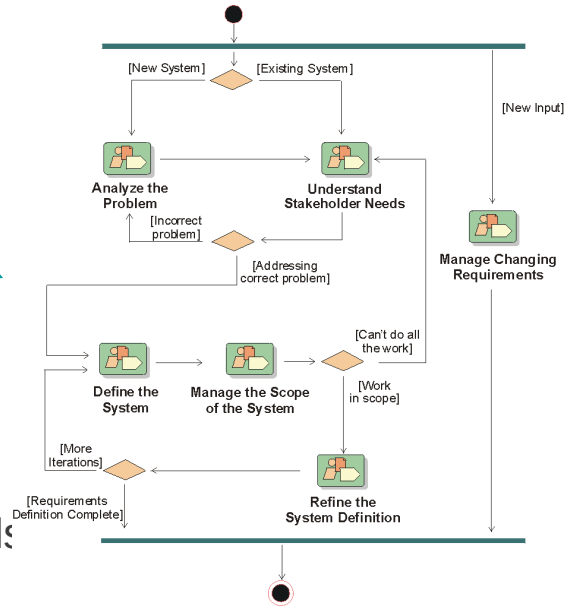
Em uma iteração, todas as disciplinas são percorridas

PROCESSO DE DESENVOLVIMENTO: RUP





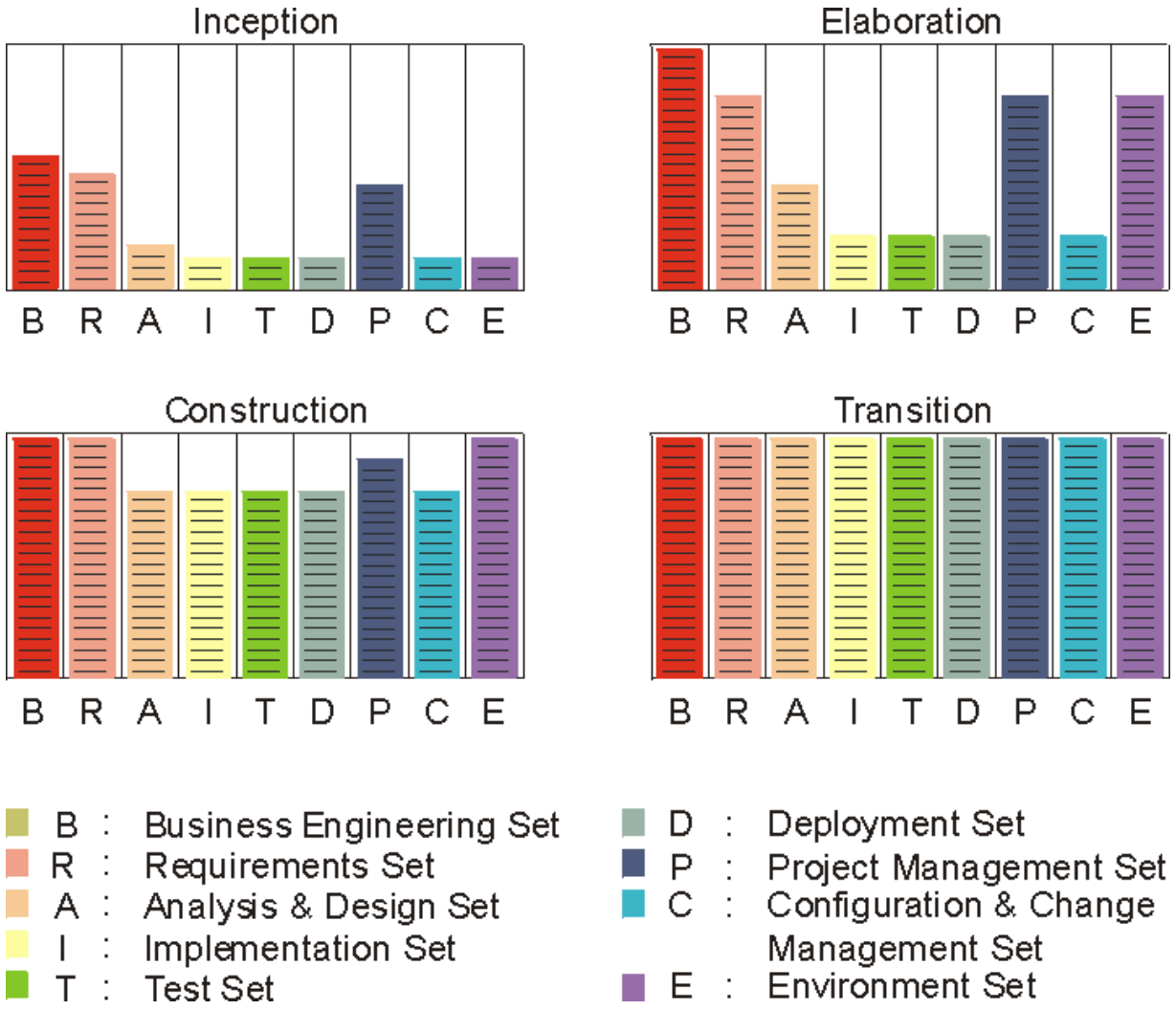
Business Modeling:
Workflow Details



Requirements:
Workflow Details

PROCESSO DE DESENVOLVIMENTO: RUP

Evolução dos artefatos



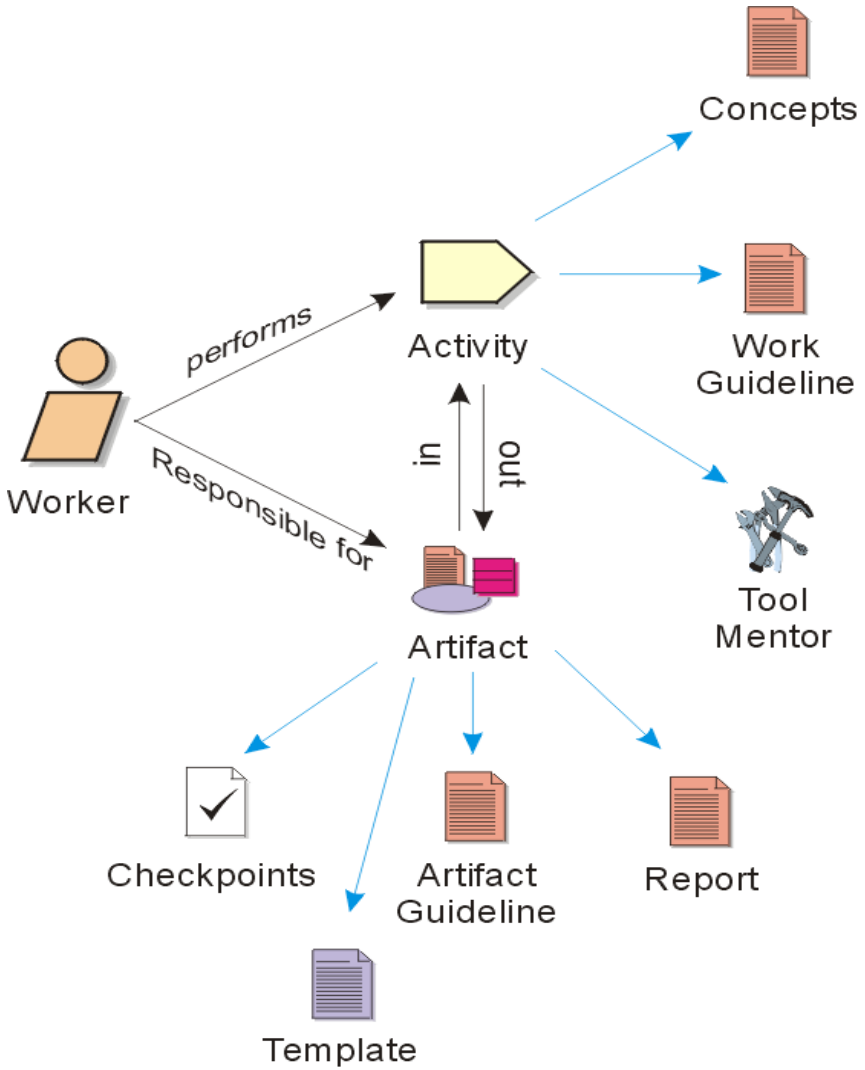
PROCESSO DE DESENVOLVIMENTO: RUP

Outros elementos

- Concepts
  - Introduzem as principais definições, idéias-chave
- Guidelines
  - Técnicas, regras, checklists, heurísticas…
- Tool mentors
  - Links para ferramentas case
- Templates
  - Modelos de artefatos
- Roadmaps
  - Formas de aplicar o processo para em contextos específicos

PROCESSO DE DESENVOLVIMENTO: RUP

Outros elementos



PROCESSO DE DESENVOLVIMENTO: RUP

Exercício

- Elabore um diagrama semelhante ao do slide anterior mostrando o relacionamento entre os seguintes elementos do RUP:
  - Disciplinas
  - Atividades
  - Artefatos
  - Responsáveis
  - Fases
  - Iterações

PROCESSO DE DESENVOLVIMENTO

Método Prescritivo

Método Ágil

PROCESSO DE DESENVOLVIMENTO

- Um **Modelo Prescritivo de Processo de Software** é um conjunto de elementos que inclui ações de engenharia de software, produtos de trabalho e mecanismos que garantam a qualidade e controle de modificações em cada projeto necessárias para o desenvolvimento de um sistema de software (PRESSMAN, 2010).

- A estrutura genérica de um processo, independente do modelo escolhido, em geral, inclui as seguintes atividades:

  - ✓ Comunicação
  - ✓ Planejamento
  - ✓ Modelagem
  - ✓ Construção
  - ✓ Implantação

PROCESSO DE DESENVOLVIMENTO

- A principal função desses modelos é colocar em ordem o caos do desenvolvimento de software.
- Não considere um modelo prescritivo de processo como estático, mas sim um processo dinâmico que adaptável ao desenvolvimento do software.
- Modelos prescritivos devem ser adaptados ao pessoal, ao problema e ao projeto.

PROCESSO DE DESENVOLVIMENTO

Abordagem Tradicional *vs.* Abordagem Ágil

Abordagem Tradicional

- Desenvolvedor hábil
- Cliente pouco envolvido
- Requisitos conhecidos, estáveis
- Retrabalho caro
- Planejamento direciona os resultados
- Conjunto de processos com metodologia definida
- Foco: grandes projetos ou com restrições de confiabilidade
- Objetivo: controlar, em busca de alcançar o planejado

Abordagem Ágil

- Desenvolvedor ágil
- Cliente comprometido
- Requisitos emergentes, mutáveis
- Retrabalho barato
- Resultados direcionam o planejamento
- Conjunto de valores com atitudes e princípios definidos
- Foco: projetos de natureza exploratória e inovadores
- Objetivo: simplificar processo de desenvolvimento

PROCESSO DE DESENVOLVIMENTO

O **pensamento ágil** propõe organizar os esforços produtivos de forma a **gerar valor** antecipadamente, a facilitar a aderência a requisitos mutantes e a manter a **visibilidade constante** e precisa durante a execução de um projeto. Como resultado deste pensamento, organizações são capazes de reduzir significativamente o risco associado ao **desenvolvimento do produto** e **maximizar o retorno sobre o investimento**.

PROCESSO DE DESENVOLVIMENTO

As figuras apresentadas nos próximos slides permitem compreender a diferença entre **processos ágeis** e **processos tradicionais** de desenvolvimento destacando características com **visibilidade, ROI, adaptabilidade e risco**. O eixo X representa o tempo decorrido em um projeto e o eixo Y representa como cada propriedade do processo se comporta no tempo.

PROCESSO DE DESENVOLVIMENTO

1. Visibilidade

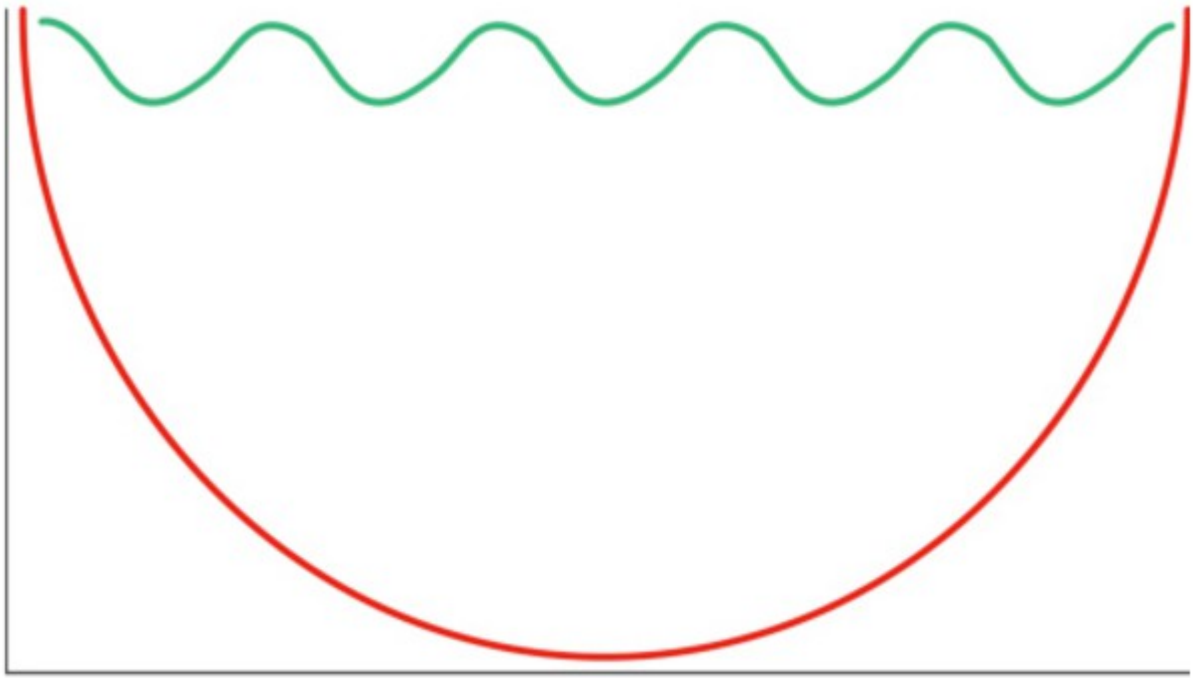


Figura 1 – Visibilidade em processos de desenvolvimento de software

PROCESSO DE DESENVOLVIMENTO

A **visibilidade** de um projeto gerenciado por um processo mais prescritivo, caso dos processos anteriores ao **Manifesto Ágil**, inicia adequada, pois a fase de concepção e o início da fase de elaboração costumam proporcionar bastante participação dos patrocinadores. Quando a fase de construção inicia-se, os patrocinadores passam a ser menos participativos, devido a grande quantidade de informação documentada, que traz uma falsa segurança sobre como se dará o projeto. Assim, só voltam a tomar conhecimento do progresso das atividades em datas específicas quando uma boa quantidade de escopo já foi construída em reuniões conhecidas como **status report**.

Já em processos mais adaptativos como o **Lean**, os ciclos curtos de entrega permitem visibilidade precisa e constante durante todo o tempo. Na **Pirâmide Lean**, os elementos que proporcionam essa visibilidade encontram-se distribuídos nas 3 camadas, mas especialmente na camada de gestão onde ocorre o planejamento e controle da produção.

PROCESSO DE DESENVOLVIMENTO

2. Retorno do Investimento

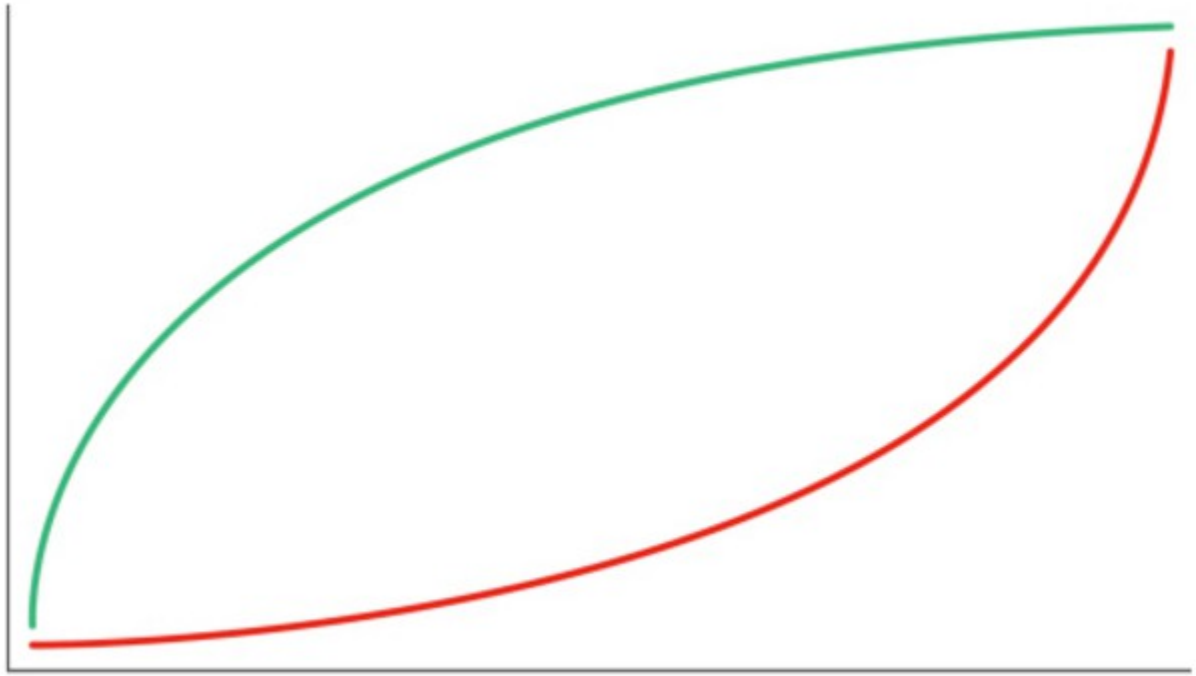


Figura 2 – Retorno do investimento em processos de desenvolvimento de software

PROCESSO DE DESENVOLVIMENTO

O **retorno do investimento** em processos mais prescritivos, representado pela série vermelha, é pequeno, inicialmente, devido às fases de planejamento intensivas e de longa duração. O retorno, quando conquistado, só pode ser percebido após meses de projeto.

A série verde demonstra retorno nos primeiros momentos do projeto por conta da estratégia de ciclos curtos de desenvolvimento, entrega e feedback dos patrocinadores. Mesmo que não seja **software pronto** para colocar em produção, após um ciclo de desenvolvimento menor que, geralmente 2 semanas, os patrocinadores recebem software funcionando.

Na Pirâmide Lean, o retorno do investimento é trabalhado nas 3 camadas.

PROCESSO DE DESENVOLVIMENTO

3. Adaptabilidade

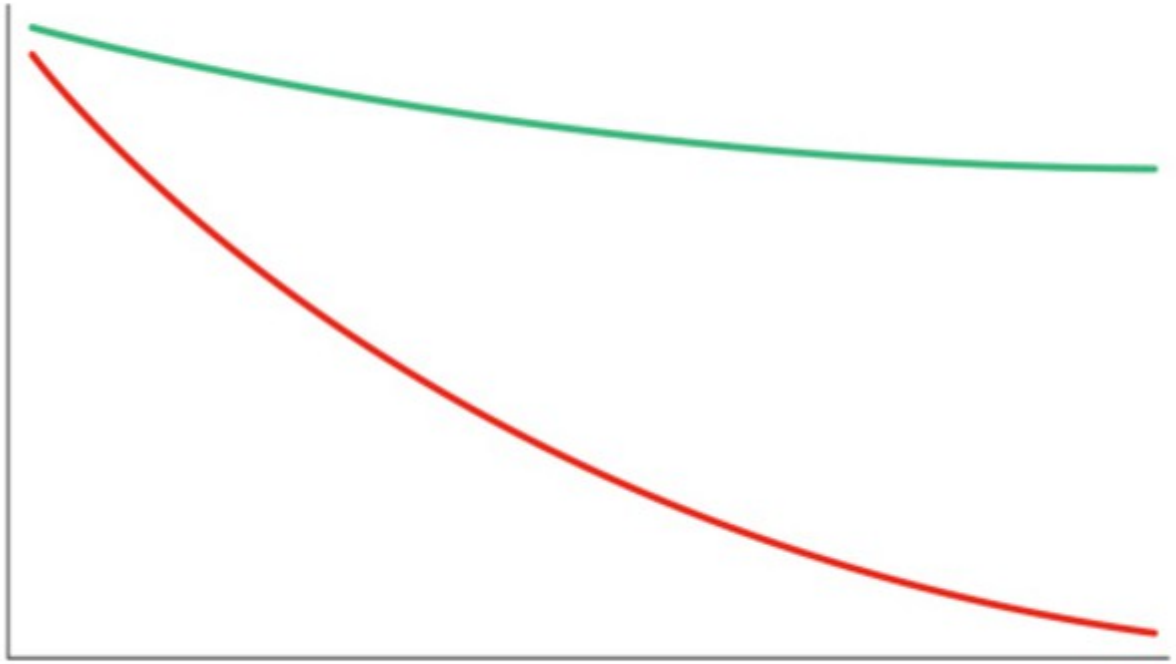


Figura 3 – Adaptabilidade em processos de desenvolvimento de software

PROCESSO DE DESENVOLVIMENTO

A **flexibilidade** para atender rapidamente as mudanças com custo reduzido é conquistada nos processos representados pela série vermelha devido às práticas de arquitetura emergente, **integração contínua e testes automatizados** constantes na base da Pirâmie Lean. Os processos mais prescritivos não conseguem agilidade para reagir às mudanças, pois as fases de qualidade e garantia de entrega não estão inseridas no desenvolvimento; elas acontecem em momento posterior. Assim, não conseguem prover segurança ao mudar sua base de código. Ademais, estão pesadamente ancoradas numa documentação ostensiva o que naturalmente os torna **resistentes a mudança**.

PROCESSO DE DESENVOLVIMENTO

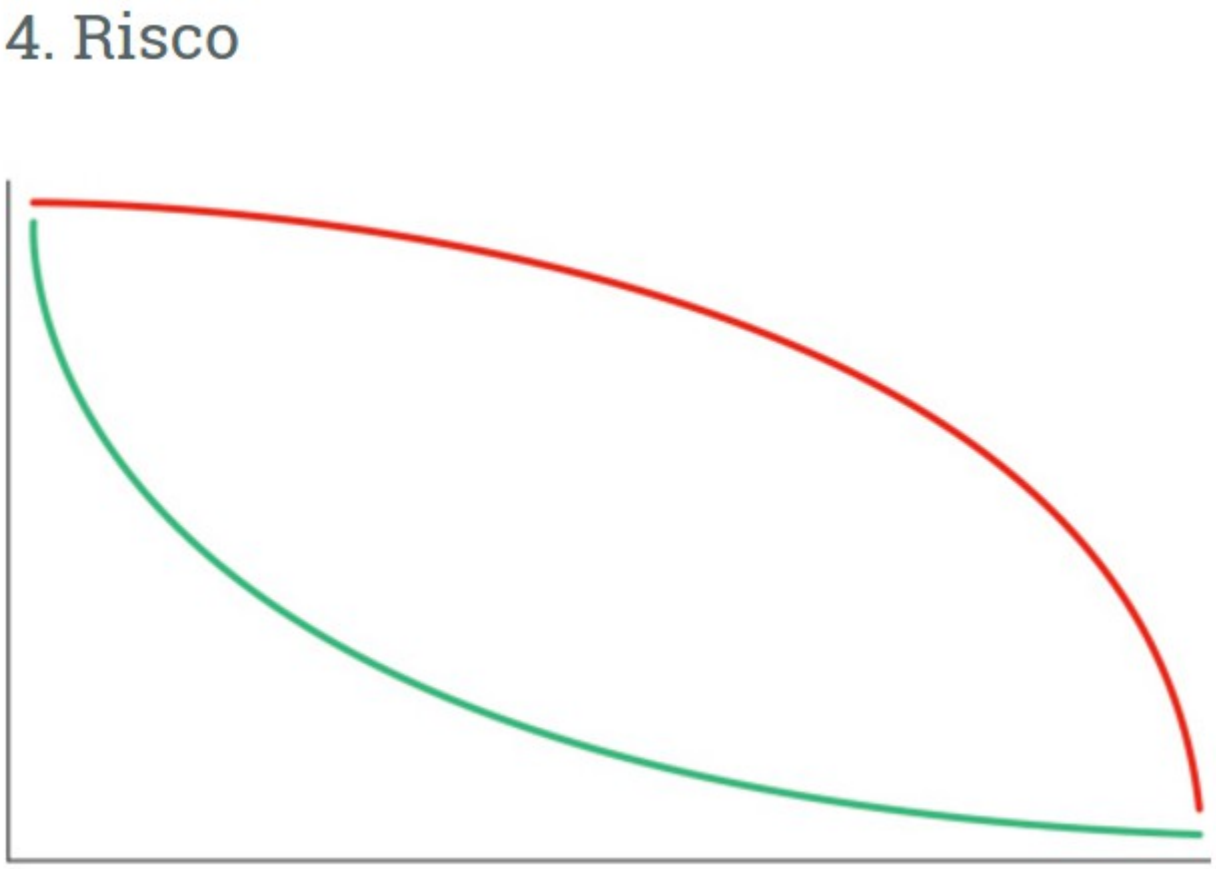


Figura 4 – Risco em processos de desenvolvimento de software

PROCESSO DE DESENVOLVIMENTO

Após as estórias que geram mais valor e que possuem pouco risco e pouco esforço, são as estórias de maior valor e alto risco que costuma iniciarem-se. Nos processos mais prescritivos os riscos são previstos e ações de mitigação são identificadas. Esse planejamento costuma ser feito para evitar surpresas. Nos processos mais adaptativos o risco é tratado como parte do processo de aprendizado. Uma vez que é evidenciado ações são naquele momento estabelecidas e a mudança de estratégia nas entregas pode acontecer sem maiores impactos. Importante esclarecer que nos 2 tipos de processo o risco é observado. Porém a proposta da cultura ágil é atacar o risco tão logo quanto possível. Na proposta mais prescritiva se planeja para que o risco seja evitado e caso aconteça para que gere o melhor impacto negativo possível.

PROCESSO DE DESENVOLVIMENTO

Quando a organização atinge um certo nível de maturidade na adoção lean e adoção ágil, ela passa a desfrutar dos benefícios abaixo:

- Redução do custo de desenvolvimento e manutenção: através da redução significativa da incidência de bugs, da identificação eficaz de riscos e de uma melhor assertividade é possível reduzir significativamente o custo de um projeto ao longo de sua vida útil;
- Antecipação de retorno de investimento: o modelo proposto é fundamentado na entrega frequente de software funcional, pronto para ser utilizado em produção. Combinado com técnicas confiáveis de priorização e planejamento é possível extrair valor do software entregue já nas primeiras iterações de desenvolvimento;
- Produtividade de 2 a 3 vezes maior comparado a abordagens tradicionais: a eliminação de desperdícios, redução da complexidade e melhora na qualidade do código possibilitam um aumento considerável na produtividade da equipe de desenvolvimento;

PROCESSO DE DESENVOLVIMENTO

- Total controle, visibilidade e gerenciabilidade do ciclo de desenvolvimento: o processo oferece informações precisas sobre o andamento dos projetos a qualquer momento, potencializando a gestão e possibilitando melhores tomadas de decisão;
- Maior assertividade e aderência: mudanças são sempre bem recebidas e seu projeto é constantemente adaptado às suas necessidades de negócio;
- Menor incerteza: ciclos curtos de planejamento e entrega, estimativas mais assertivas e uma construção com integridade em todas as etapas do processo de desenvolvimento possibilitam aumento da previsibilidade e diminuição da incerteza em desenvolvimento de software;
- Melhor qualidade no produto final: os processos de gestão de qualidade que ocorrem desde as fases iniciais de concepção do produto e por todo o desenvolvimento, garantem um produto final com qualidade superior.
- Melhoramento contínuo: além de ser um processo resiliente, que se adapta às necessidades de diferentes tipos de projeto, o processo proposto é fundamentado num ciclo contínuo de melhoramento, através de retrospectivas e métodos de inspeção e adaptação aplicados constantemente durante todo o ciclo de vida do projeto.

pesquise

1- EUP

2- RUP

3- OPENUP

4- SCRUM

5- XP

6- KANBAN

Apresentar na próxima aula...

Valendo !!!

(2ª -Insígnia da Invisibilidade)

Dúvidas