Processos de Software

- Modelo de Processo Genérico
- Processos prescritivos

Objetivos

- Apresentar e discutir a evolução histórica dos modelos de processos de software
- Discutir os problemas relacionados ao uso de processos de software para propor soluções

Referências Bibliográficas

- Básica:

  PRESSMAN, Roger S; MAXIM, Bruce R. **Engenharia de software**: uma abordagem profissional. 9. Porto Alegre AMGH 2021 1 recurso online.

- Complementar:

  SOMMERVILLE, Ian. Engenharia de software. 10 ed. São Paulo: Pearson, 2019.

O que é um processo?

- “Um processo é uma sequência de passos realizados com um propósito. Ou seja, processo é o que você faz. O processo integra pessoas, ferramentas e procedimentos. Processo é o que pessoas fazem, utilizando procedimentos, métodos, ferramentas, e equipamentos, para transformar matéria prima (inputs) em um produto (outputs) de valor para os seus clientes.”

  (PAULK, Mark C. The capability maturity model: guidelines for improving the software process. Boston: Addison Wesley, 2003.)

O que é um Processo de Desenvolvimento de Software?

- “Um processo de desenvolvimento de software pode ser definido como um conjunto de atividades, métodos, práticas e transformações que pessoas utilizam para desenvolver ou dar manutenção em softwares ou em seus produtos associados (projetos, manuais, código, etc.)”

  (PAULK, Mark C. The capability maturity model: guidelines for improving the software process. Boston: Addison Wesley, 2003.)

- “… processo de software como uma metodologia para as atividades, ações e tarefas necessárias para desenvolver um software de alta qualidade.” (PRESSMAN, 2021. p. 20)

Processo é sinônimo de Engenharia de Software?

- “Um processo de software define a abordagem adotada conforme um software é elaborado pela engenharia. Mas, a engenharia de software também engloba tecnologias que fazem parte do processo – métodos, técnicas e ferramentas automatizadas.”

  (PRESSMAN, 2021. p. 20)

Ou seja, é resposta é “sim e não”!

Modelo de Processo Genérico (1/2)

- Atividades metodológicas
  - Comunicação
  - Planejamento
  - Modelagem
  - Construção
  - Entrega

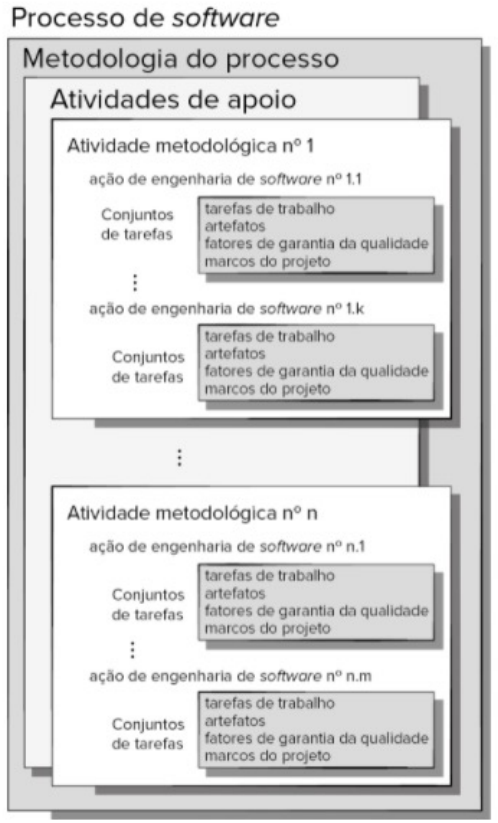


Figura 2.1
Uma metodologia do processo de *software*.

(PRESSMAN, 2021. p. 21)

Modelo de Processo Genérico (2/2)

- Atividades de apoio
  - Controle e acompanhamento do projeto
  - Administração de riscos
  - Garantia de qualidade
  - Revisões técnicas
  - Medição
  - Gerenciamento de configuração de software
  - Gerenciamento do reuso
  - Preparo e produção de artefatos

Fluxo de Processo

- Linear
- Iterativo
- Evolucionário
- Paralelo

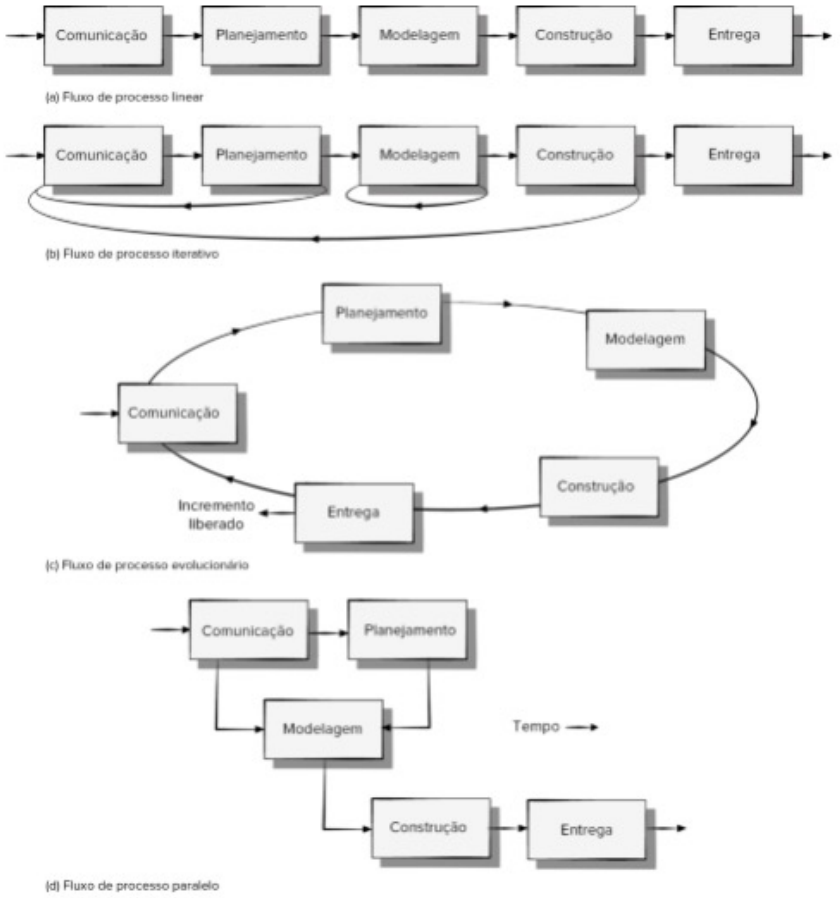


Figura 2.2
Fluxo de processo.

(PRESSMAN, 2021. p. 21)

Modelo de Processo Genérico

- As atividades do processo devem ser definidas, basicamente, de acordo com a complexidade e com os evolvidos no projeto.
- Ou seja, projetos diferentes demandam conjuntos de tarefas diferentes. A equipe de software escolhe o conjunto de tarefas fundamentada no problema e nas características do projeto.

Padrões de Processo

- Descreve um problema relacionado ao processo que pode ser encontrado em um trabalho de engenharia de software
- Identifica o ambiente onde problema é normalmente encontrado
- Sugere soluções para o problema

Tipos de Padrões de Processo (1/3)

- Padrões de estágio
  - Define um problema associado a uma atividade metodológica
  - *Exemplo*: Estabelecendo Comunicação

Tipos de Padrões de Processo (2/3)

- Padrões de tarefas
  - Define um problema associado a uma ação de engenharia de software ou tarefa de trabalho relevante para a sua prática
  - *Exemplo*: Levantamento de Necessidades

Tipos de Padrões de Processo (3/3)

- Padrões de fases
  - Define a sequência das atividades metodológicas que ocorrem dentro do processo
  - *Exemplo*: Modelo Espiral ou Prototipação

Veja um exemplo na página 24 do livro do PRESSMAN, 2021

Avaliação e Aperfeiçoamento de Processos

- SCAMPI (*Standard CMMI Assesment Method for Process Improvement*)
- CBA IPI (*CMM – Based appraisal for Internal Processo Impovement*)
- SPICE (ISO/IEC15504)
- ISO 9001:2000 para Software

Modelo de Processo Prescritivo

- Esses modelos defendem uma abordagem ordenada para a engenharia de software
- Tentam trazer ordem ao “caos” (*code and fix*)
- Prescrevem um conjunto de elementos do processo
- Também são conhecidos como ***modelos tradicionais***

Modelos de Processo Prescritivo

- Modelo em cascata
- Modelos de processo incremental
- Modelos de processo evolucionário
  - Prototipação
  - Modelo espiral

Modelo em Cascata (*Waterfall Model*)

- Também conhecido como *Ciclo de Vida Clássico*

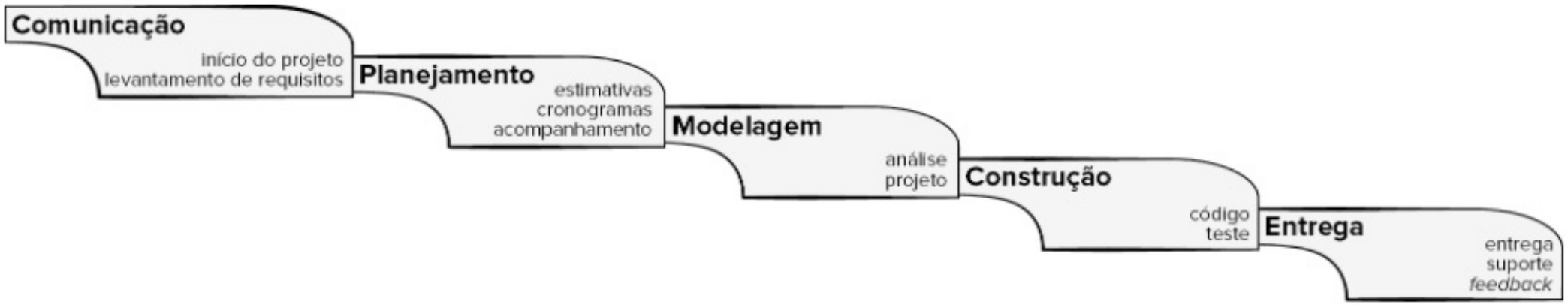


Figura 2.3
O modelo cascata.

(PRESSMAN, 2021. p. 25)

Modelo em Cascata: alguns problemas

- Projetos reais raramente seguem o fluxo sequencial proposto pelo modelo.
- Com frequência, é difícil para o cliente estabelecer explicitamente todas as necessidades no início da maioria dos projetos.
- O cliente deve ter paciência. Uma versão operacional do(s) programa(s) não estará disponível antes de estarmos próximos ao final do projeto.
- Erros graves podem não ser detectados até o programa operacional ser revisto.

Modelos de processo incremental

- Combina elementos dos fluxos de processos lineares e paralelos
- Cada sequência linear produz um incremento executável do software
- Cada incremento é um núcleo, uma base para o próximo
  → *produto essencial*

Modelos de processo incremental

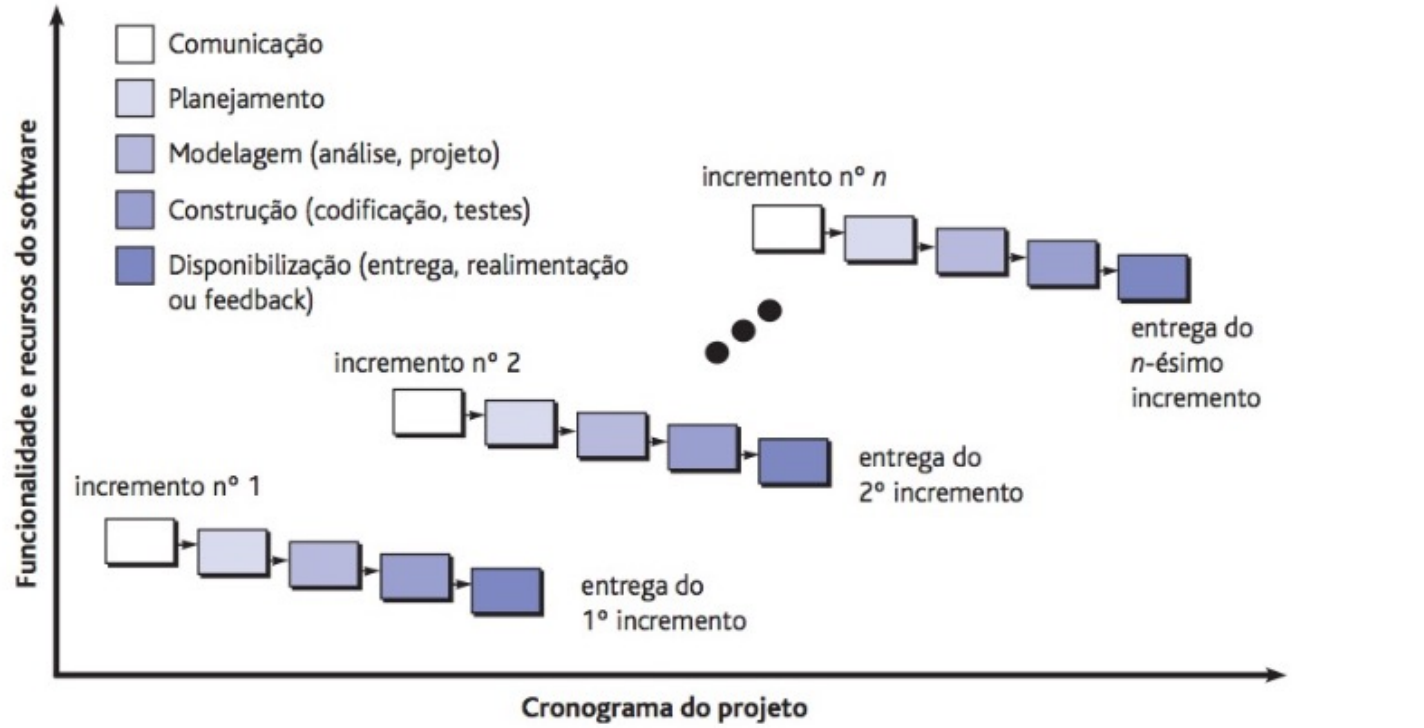


FIGURA 4.3 O modelo incremental.

(PRESSMAN, 2016. p. 44)

Modelos de processo incremental

- *Exemplo 1*: um editor de texto que é implementado só com as funções básicas em sua primeira versão
- *Exemplo 2*: um Sistema de Gestão Empresarial para substituição de um Sistema Legado

Modelos de processo evolucionário

- Modelos evolucionários são iterativos ou incrementais
- Apresentam características que possibilitam desenvolver versões cada vez mais completas do software
- Modelos comuns: Prototipação e Modelo Espiral

Prototipação

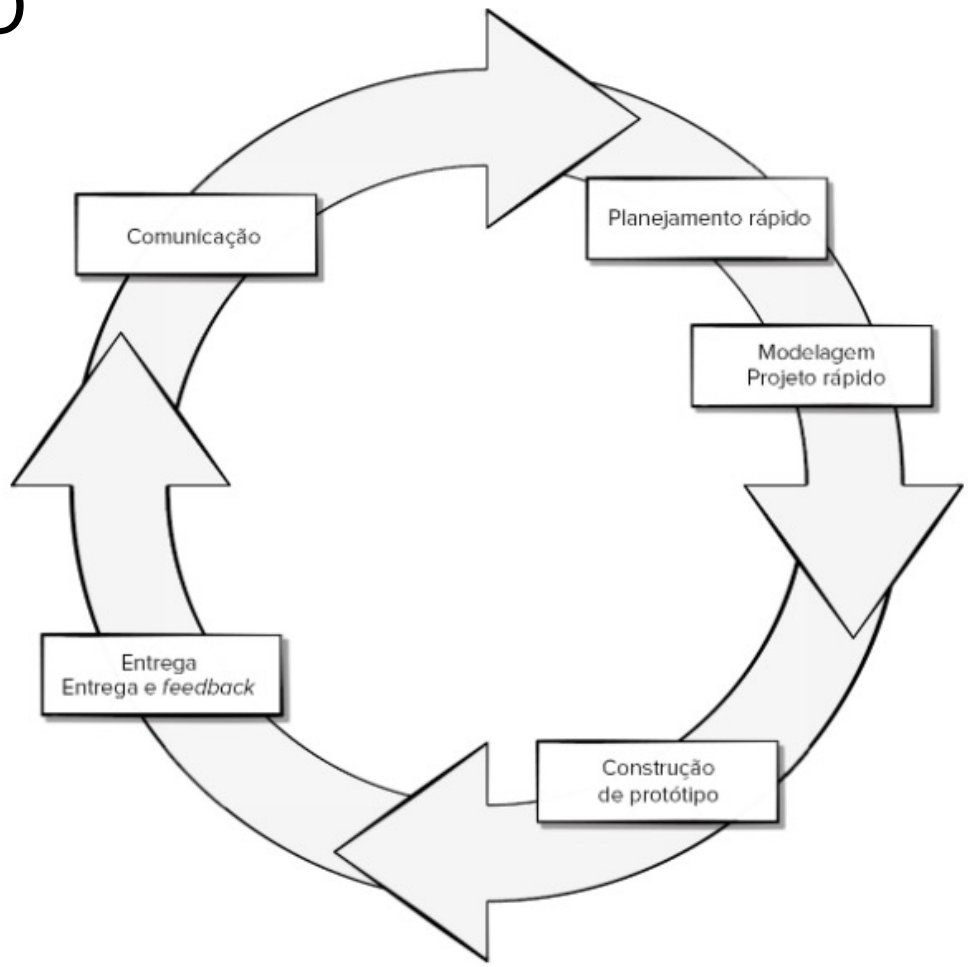


Figura 2.4
O paradigma da prototipação.

(PRESSMAN, 2021. p. 27)

Modelo Espiral Típico

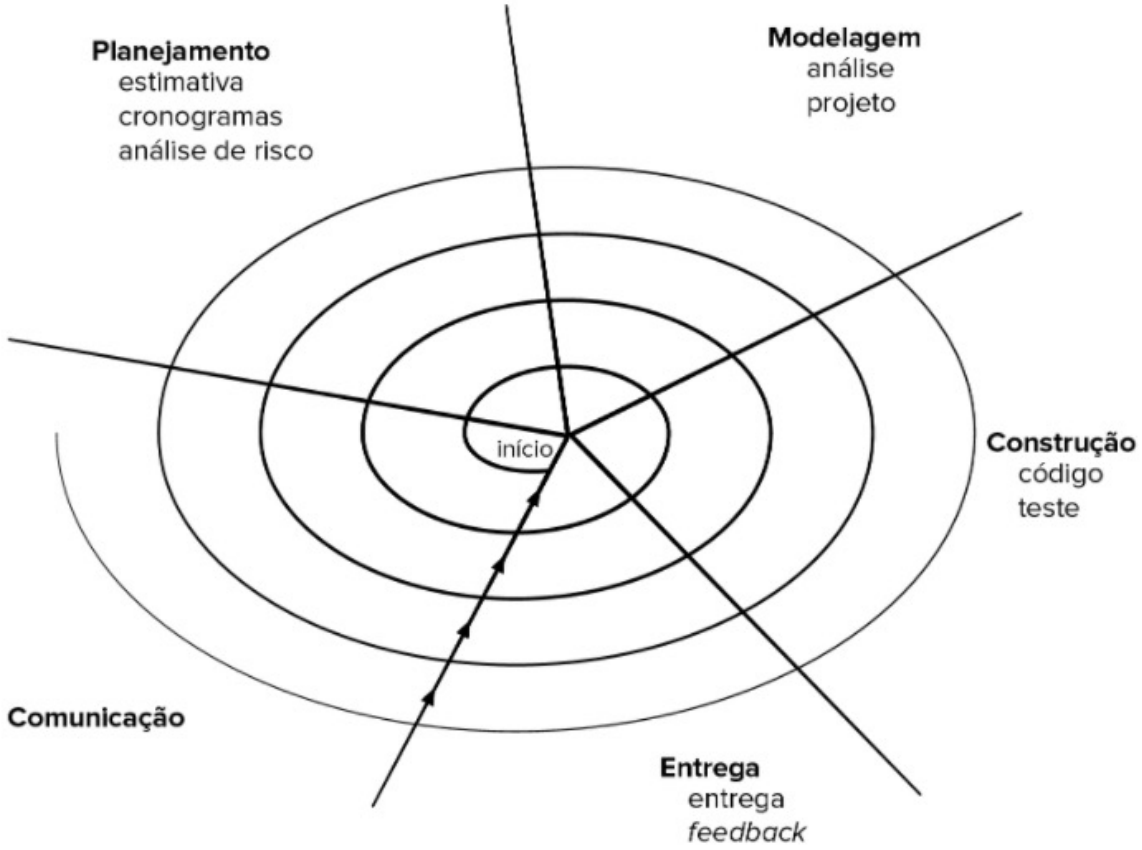
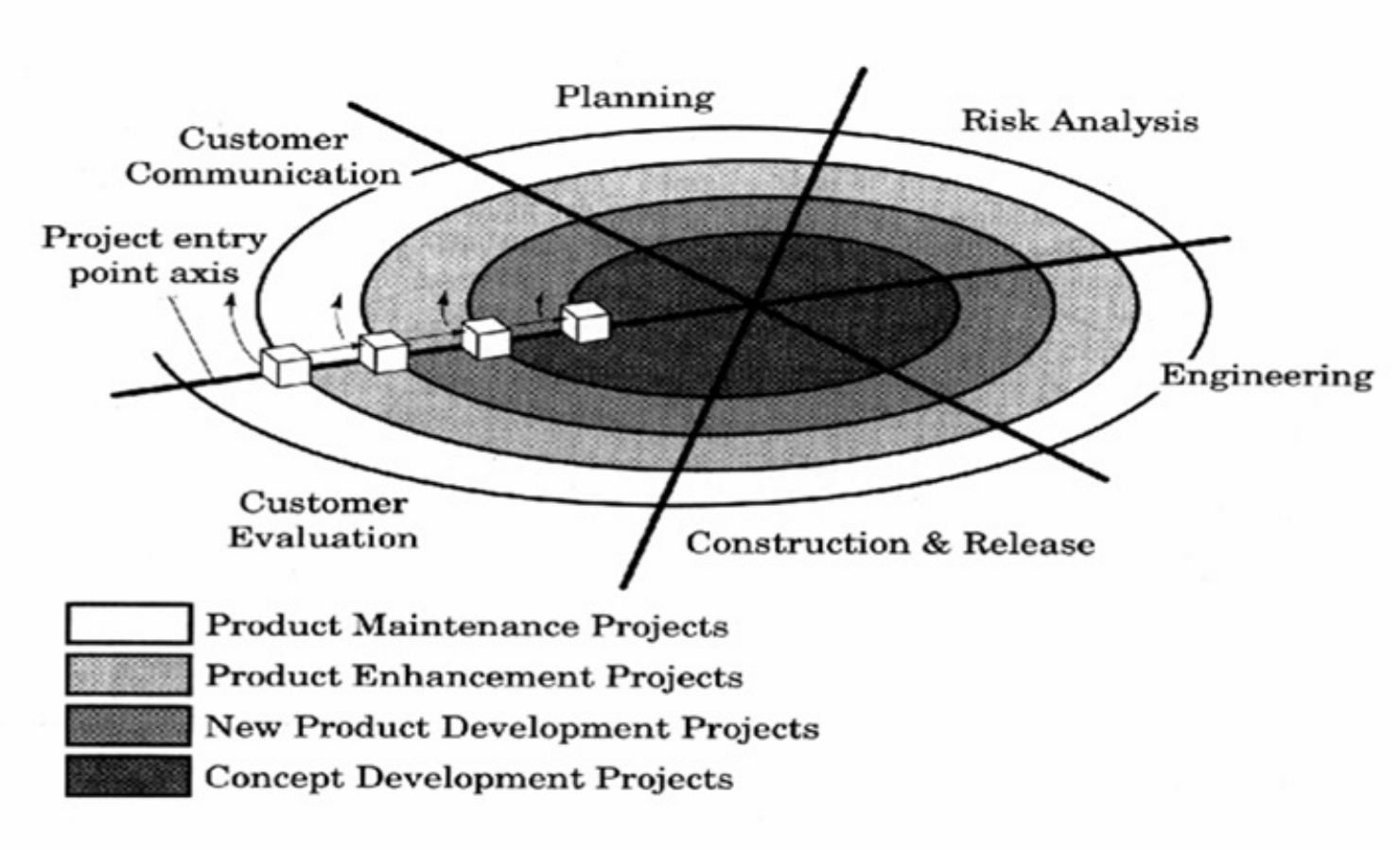


Figura 2.5
Modelo espiral típico.

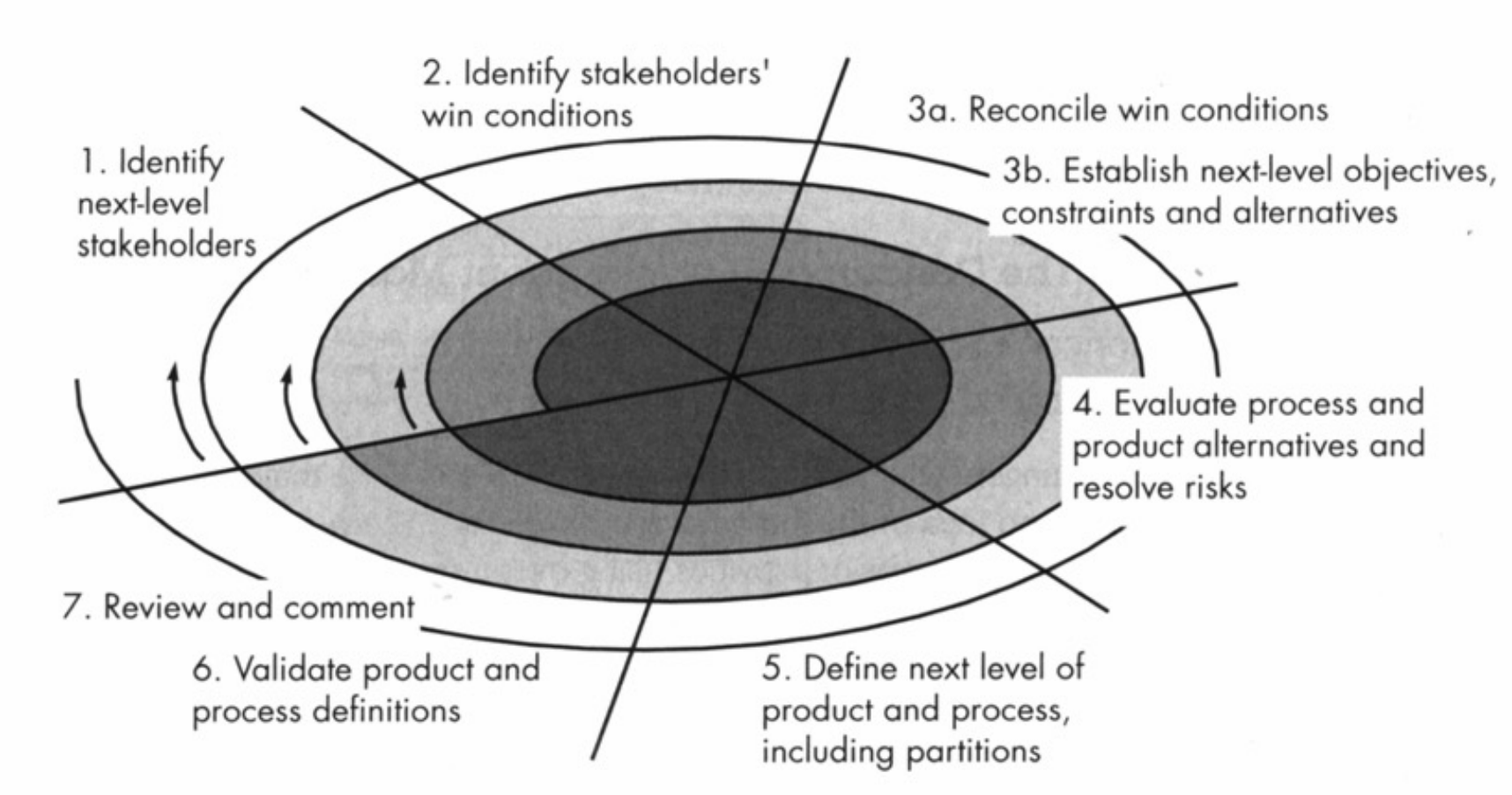
(PRESSMAN, 2021. p. 29)

Modelo Espiral (outra versão)



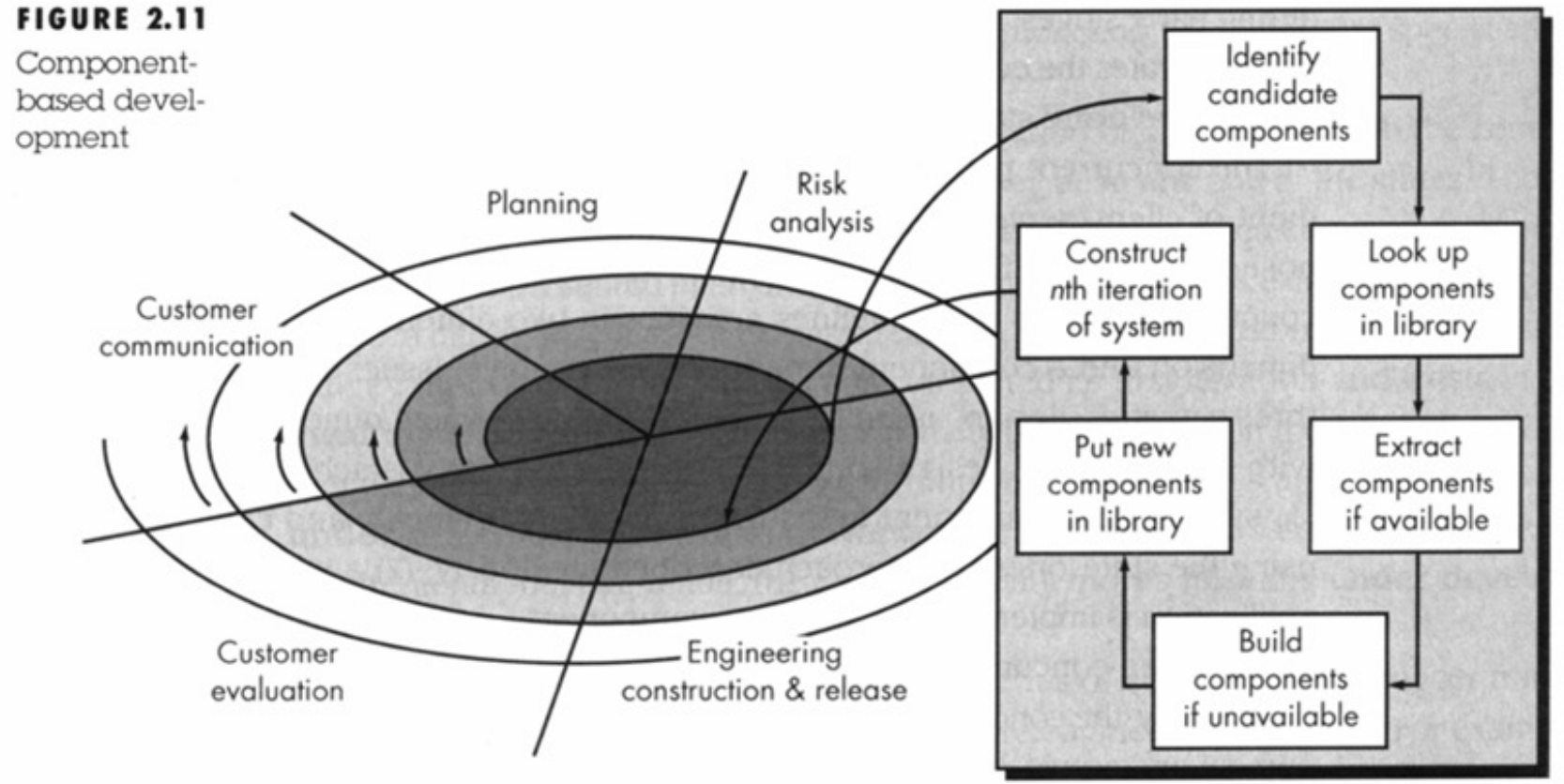
(PRESSMAN, 2001. p. 37)

Modelo Espiral “WinWin”



(PRESSMAN, 2001. p. 39)

Modelo Espiral Baseado em Componentes



(PRESSMAN, 2001. p. 42)

Processo WebE

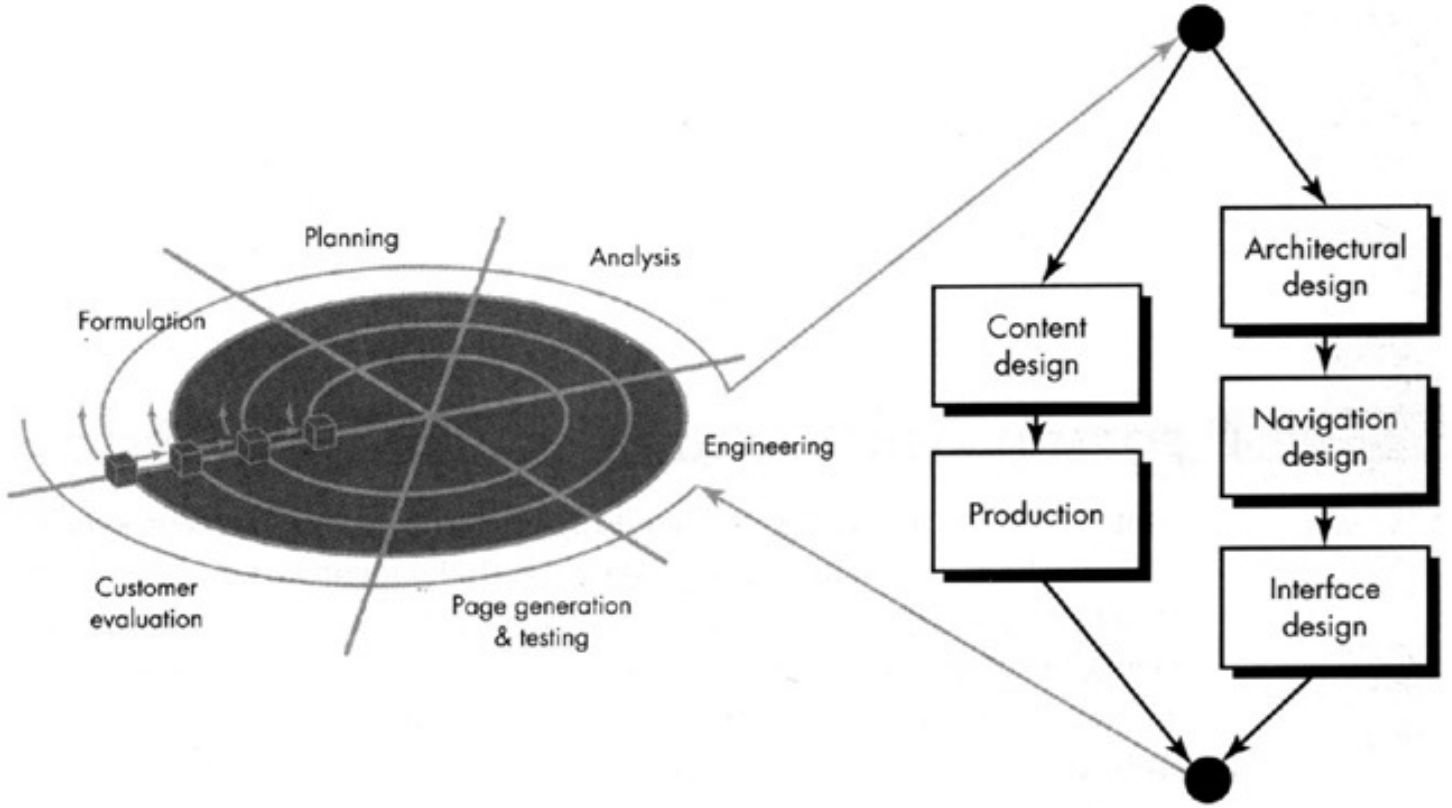
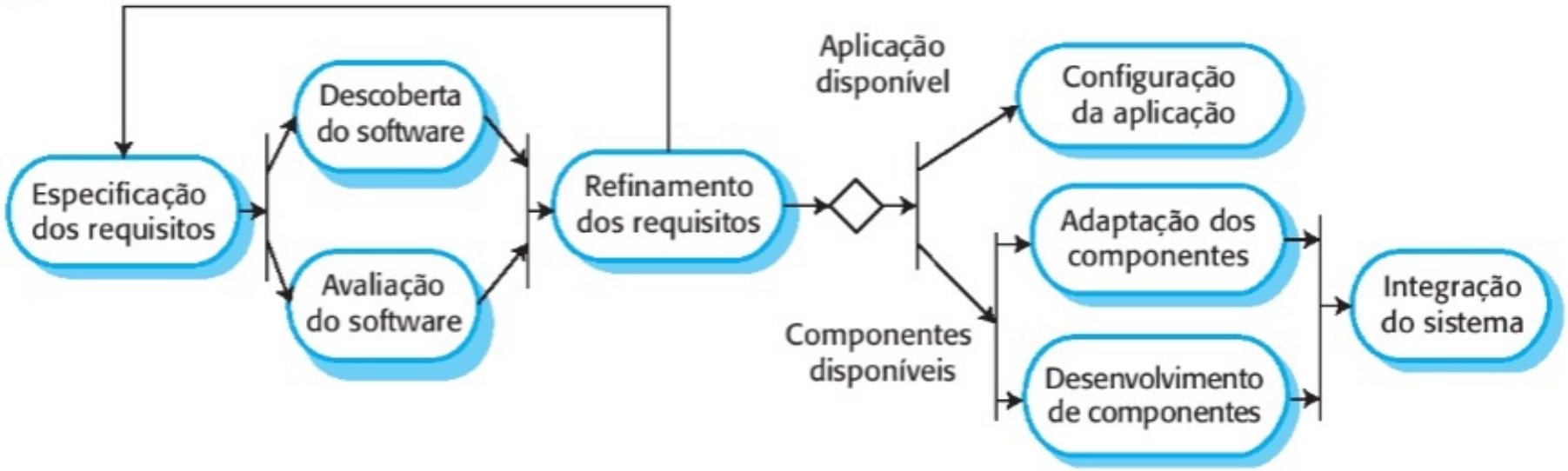


FIGURE 29.2 The WebE process model

(PRESSMAN, 2001. p. 775)

Engenharia de Software orientado ao reuso

FIGURA 2.3 Engenharia de software orientada para o reúso.



(SOMMERVILLE, 2019. p. 38)

Problemas do Desenvolvimento Incremental

Problemas do desenvolvimento incremental

Embora o desenvolvimento incremental tenha muitas vantagens, ele não está livre de problemas. A principal dificuldade é o fato de que as grandes organizações têm procedimentos burocráticos que evoluíram ao longo do tempo, o que pode levar a uma incompatibilidade entre esses procedimentos e um processo iterativo ou ágil mais informal.

Às vezes, esses procedimentos existem por um bom motivo. Por exemplo, pode haver procedimentos para garantir que o software satisfaça adequadamente as regulamentações externas (as normas contábeis da lei Sarbanes–Oxley[1] nos Estados Unidos, por exemplo). Como nem sempre é possível mudá-los, os conflitos de processo podem ser inevitáveis.

(SOMMERVILLE, 2019. p. 32)

Modelos de Processo Especializado

- Desenvolvimento baseado em componentes
  - Importância do reuso de software
- Modelo de métodos formais
  - Ênfase na especificação matemática dos requisitos de software

Processo de Software Pessoal (PSP) – 1/2

- Proposto por Watts Humphrey
- O processo de software deve se adequar às pessoas e não o contrário!
- O PSP (*Personal Software Process*) enfatiza a importância de se aprender com os erros
- É um processo desafiador e exige um nível de comprometimento bem alto

Processo de Software Pessoal (PSP) – 2/2

- O PSP (*Personal Software Process*) possui cinco atividades estruturais:
  - Planejamento
  - Projeto de alto nível
  - Revisão de projeto de alto nível
  - Desenvolvimento
  - Autópsia (análise *postmortem*)

Processo de Software em Equipe (TSP)

- Também proposto pelo Humphrey
- Baseado em lições aprendidas com o PSP
- O objetivo do TSP (*Team Software Process*) é criar equipes "autodirigidas" para produzir software de alta qualidade

Objetivo do TSP (1/2)

- Promover a criação de equipes autodirigidas que planejem e acompanhem seu próprio trabalho, estabeleçam metas e sejam proprietárias de seus processos e planos
- Mostrar aos gerentes como treinar e motivar suas equipes, mantendo alto desempenho

Objetivo do TSP (2/2)

- Acelerar o aperfeiçoamento dos processos de software
- Fornecer orientação para melhorias a organizações com elevado grau de maturidade
- Facilitar o ensino universitário de habilidades de trabalho em equipe de nível industrial

Atividades Metodológicas do TSP

- O TSP (*Team Software Process*) possui cinco atividades metodológicas:
  - Lançamento do projeto
  - Projeto de alto nível
  - Implementação
  - Integração e testes
  - Autópsia (análise *postmortem*)

Processo de Manutenção de Sistemas

- Identificação do problema
- Análise do impacto da alteração demandada pelo problema
- Planejamento da forma de abordagem do problema
- Atualização da documentação existente, levando em consideração a transição de versões
- Implementação das alterações
- Implantação das alterações
- Treinamento

Organização de Times de Software

- Estrutura centralizada
- Estrutura descentralizada
- Terceirização

Como formar uma equipe motivada para atuar em um processo de software?

Processos e Equipes no Modelo Cascata

- Modelo monolítico (*Analista-Projetista-Desenvolvedor*)
  - Planejamento
  - Não é escalável
  - Reflete modelo em cascata (waterfall)
  - Requisitos “perfeitos” são transformados em desenhos imutáveis que são implementados em códigos por programadores mecânicos
  - Antítese do modelo iterativo e incremental

Processos e Equipes no Modelo Iterativo

- Modelo Hierárquico
  - Arquiteto de Sistemas com visão macro
  - Analista de Sistemas com visão micro
  - Desenvolvedores (engenheiros de aplicação) realizam visão micro

  Centro de gravidade de projetos OO bem realizados oscila entre gerente de projeto, arquiteto de sistemas, analistas de sistemas e desenvolvedores

Desafios do Processo de Software (1/2)

- Relacionados às PESSOAS
  - “Inteligência compartilhada”
  - Integração
  - Comunicação

- Relacionados aos PROBLEMAS
  - “Dividir para Conquistar”

Desafios do Processo de Software (2/2)

- Relacionados aos PROCESSOS
  - Visão integrada
  - Ciclo de vida do processo de desenvolvimento de sistemas

- Relacionados aos PROJETO
  - Gerenciamento, controle e resultado

Gerenciamento de Riscos

- Como identificar os riscos que envolvem o processo de desenvolvimento de um software?
  - Tamanho do projeto
  - Interferência no negócio
  - Características do cliente / usuário
  - Definição do processo (metodologia)
  - Ambiente de desenvolvimento (tecnologia)
  - Experiência da equipe

Acompanhamento de Projetos de Software

- Como dividir e organizar as atividades que devem ser realizadas?
- Quais as principais ferramentas utilizadas para acompanhar o desenvolvimento de um processo de desenvolvimento de software?

Acompanhamento de Projetos de Software

- Alguns critérios que devem ser considerados para relacionar pessoas às tarefas / atividades
  - Conhecimento da tecnologia
  - Habilidade e experiência para lidar com a tecnologia
  - Grau de dificuldade da tarefa / atividade
  - Interdependência das tarefas / atividades
  - Prazo para realização

Processo Unificado

Processo Unificado (UP – *Unified Process*)

JACOBSON, Ivar, BOOCH, Grady, RUMBAUGH, James. The unified software development process. Addison Wesley, 1998.

- Baseado na construção de software a partir de componentes interconectados através de interfaces bem definidas
- Utiliza a UML (*Unified Modeling Language*)

Processo Unificado

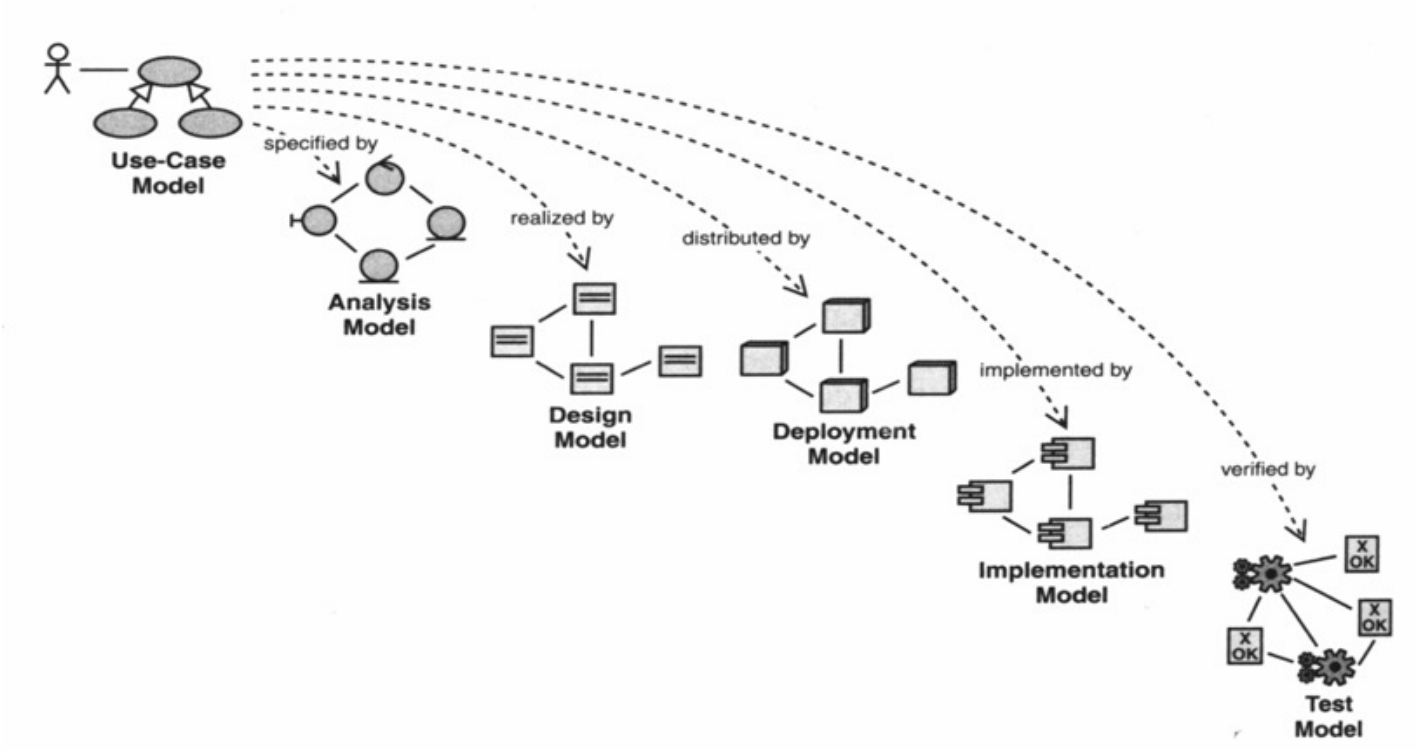
O Rational Unified Process

O *Rational Unified Process* (RUP) reúne os elementos de todos os modelos de processo genéricos discutidos aqui e apoia a prototipação e a entrega incremental do software (KRUTCHEN, 2003). O RUP é descrito normalmente a partir de três perspectivas: uma dinâmica, que mostra as fases do modelo no tempo; uma estática, que mostra as atividades do processo; e uma prática, que sugere práticas a serem utilizadas no processo. As fases do RUP são a concepção, em que se estabelece um *business case* para o sistema; a elaboração, em que são desenvolvidos os requisitos e a arquitetura; a construção, em que o software é implementado; e a transição, em que o sistema é implantado.

(SOMMERVILLE, 2019. p. 32)

Características chave do UP (1/2)

- Dirigido por caso de uso



(JACOBSON, 1998. p. 10)

Características chave do UP (2/2)

- Centrado na Arquitetura
- Iterativo e Incremental

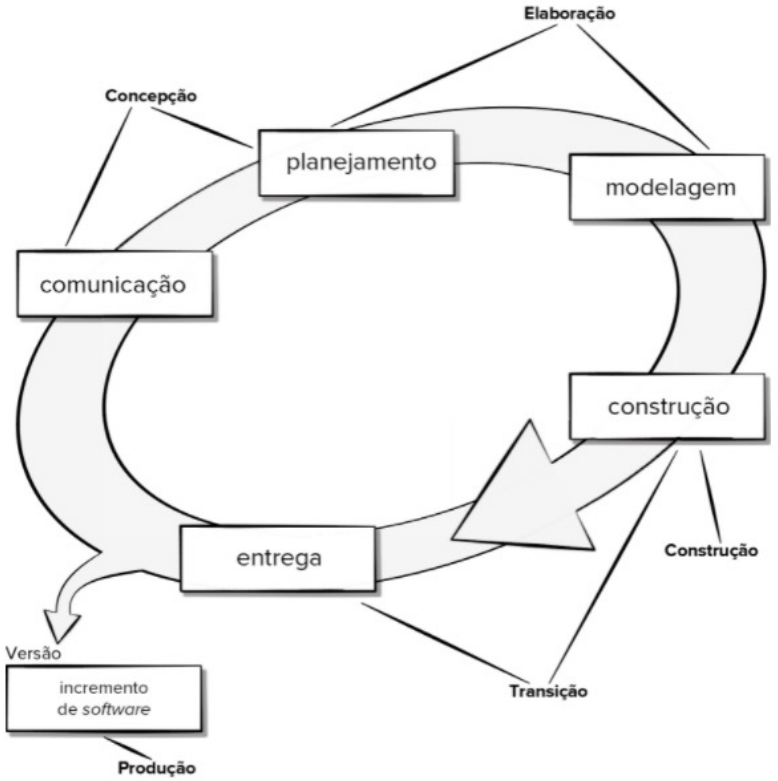
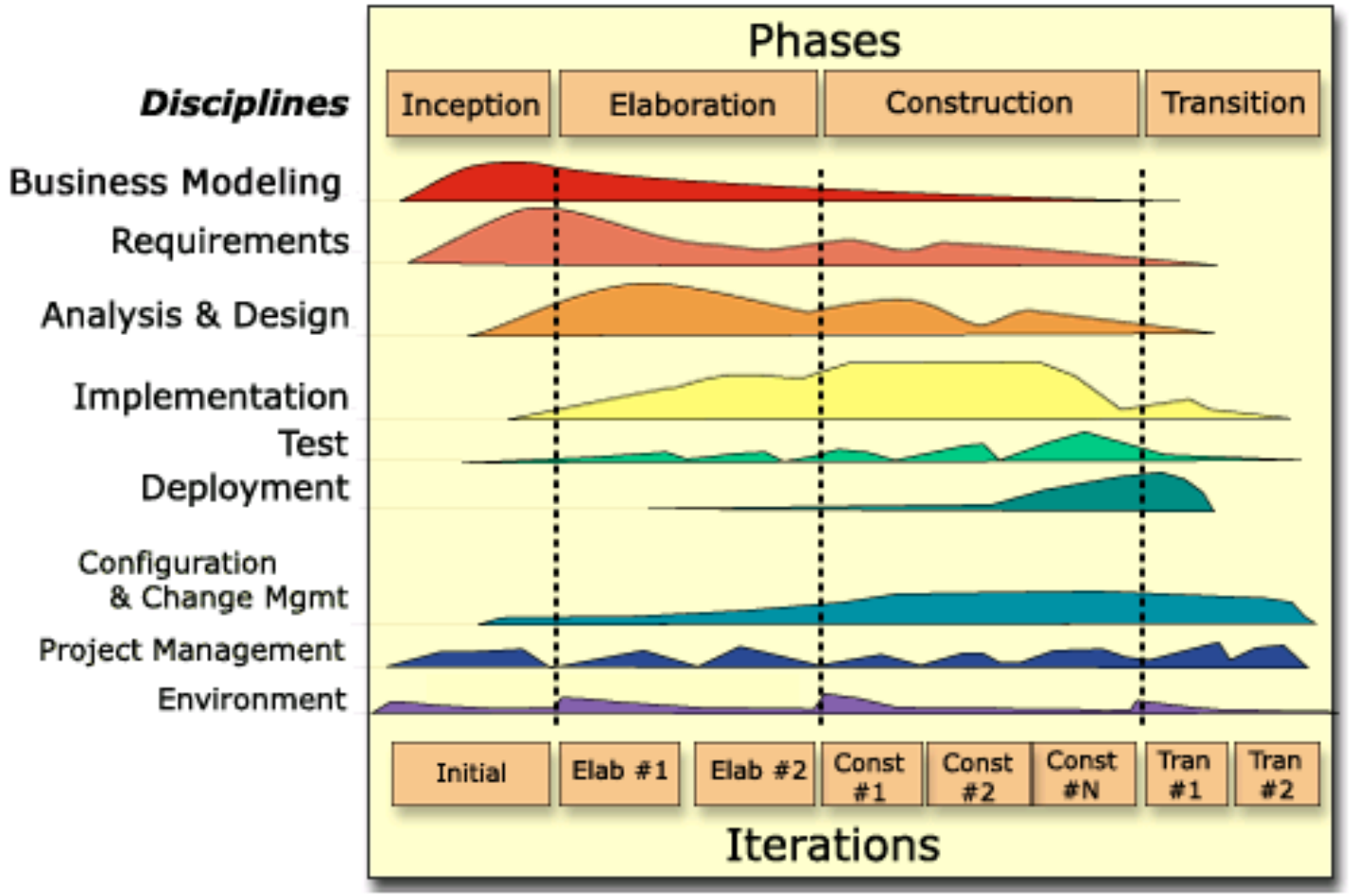


Figura 2.6
O Processo Unificado.

(PRESSMAN, 2021. p. 33)

Rational Unified Process (RUP)



Fase do RUP (1/4)

- Concepção
  - Definição dos casos de uso mais críticos, os quais representam as funções chave do sistema
  - Delimita-se o escopo do produto a ser desenvolvido, identifica-se e reduz-se principalmente os riscos críticos da tarefa / atividade

Fase do RUP (2/4)

- Elaboração
  - Descrição arquitetural do software
  - Procura-se também definir a maioria dos casos de uso, capturando a maioria dos requisitos do software
  - No final desta fase deve-se estar apto a planejar a fase de construção em detalhes

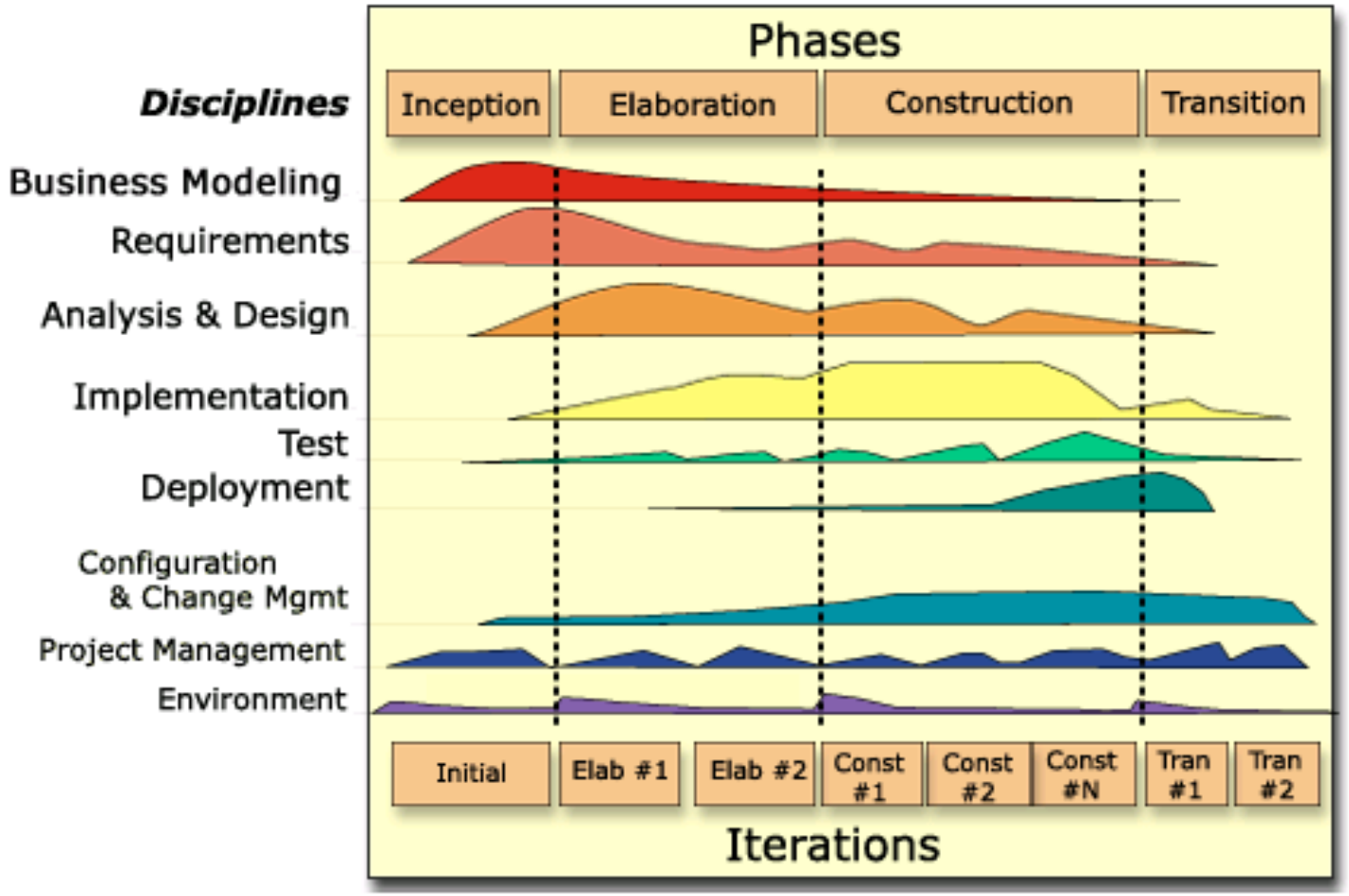
Fase do RUP (3/4)

- Construção
  - O software deve ser construído completamente, ou seja, deve-se adicionar a musculatura ao esqueleto (arquitetura)
  - Visa-se a capacidade operacional do software

Fase do RUP (4/4)

- Transição
  - Esta fase envolve a realização de testes com o usuário, corrigir defeitos encontrados e realizar treinamento

Rational Unified Process (RUP)



Número de iterações

- Projeto muito simples
  - Concepção: 1
    - ✓Protótipo GUI ou prova de conceito
  - Elaboração: 1
    - ✓Protótipo arquitetural
  - Construção: 1
    - ✓Montagem do produto
  - Transição: 1
    - ✓Finalização do produto

Número de iterações

- Projetos maiores
  - Concepção: 1
    - ✓Protótipo GUI ou prova de conceito
  - Elaboração: 2
    - ✓Protótipo arquitetural
    - ✓Finalização da arquitetura

Número de iterações

- Projetos maiores
  - Construção: 2
    - ✓Montagem parcial do produto
    - ✓Maturação e montagem final do produto
  - Transição: 1
    - ✓Finalização do produto

Número de iterações

- Projetos maiores ainda com muitas tecnologias desconhecidas
  - Concepção: +1 iteração
    - ✓Para mais protótipos
  - Elaboração: +1 iteração
    - ✓Para mais explorações arquiteturais
  - Construção: +1 iteração
    - ✓Devido ao tamanho do produto
  - Transição: +1 iteração
    - ✓Para mais *feedback* operacional

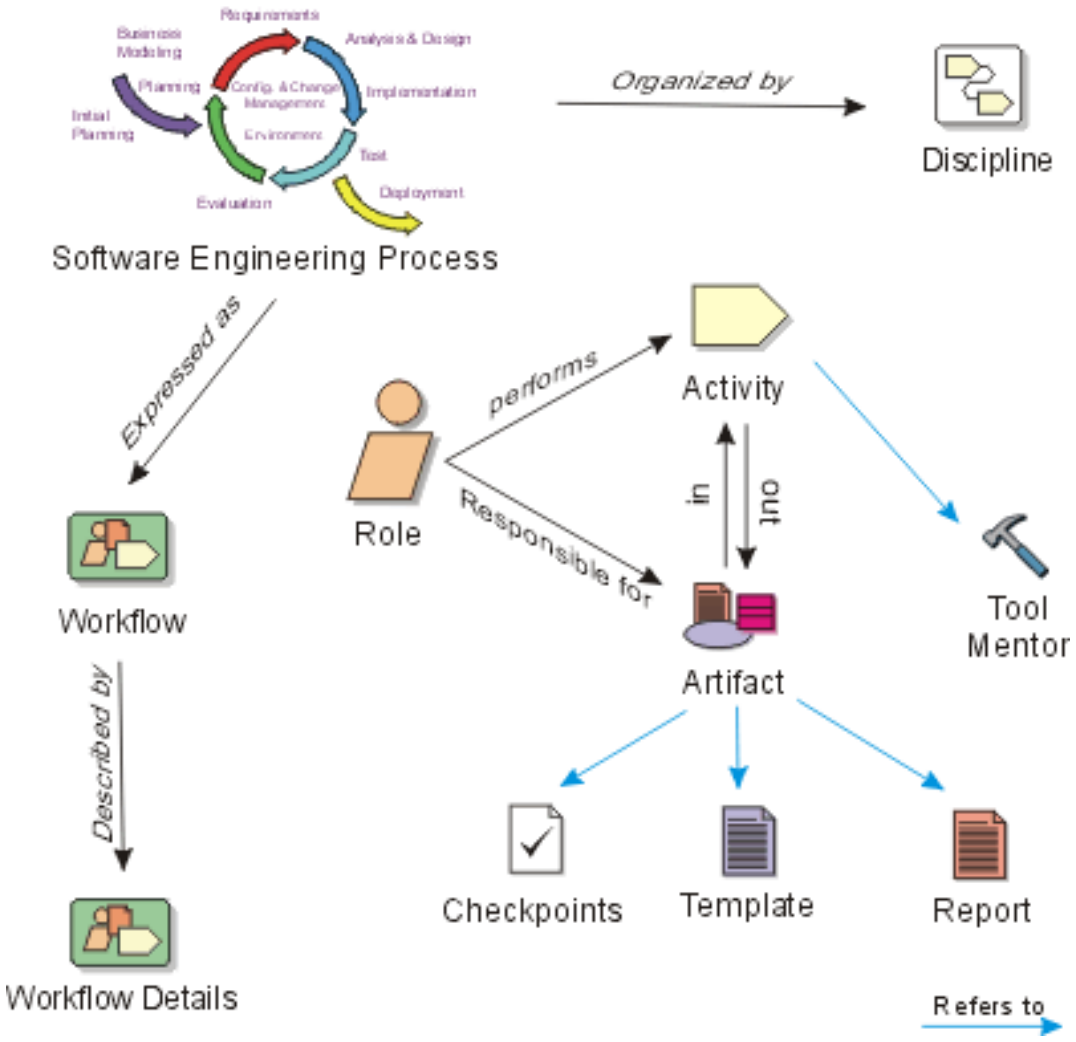
Variação nas iterações

- Modelo de domínio completamente diferente
  - Consolidação de conceitos
  - Mais iterações na concepção
- Nova arquitetura precisa ser montada ou existem muitos riscos
  - Mais iterações na elaboração
- Produto grande e complexo
  - Mais iterações na construção
- Organização com pouca experiência em projetos iterativos
  - Três iterações que produzem software [0,1,1,1]

Consideração sobre as iterações

- Projetos normalmente possuem entre 6 e 8 iterações em projetos típicos em empresas com experiência em projetos iterativos
- Tamanho de uma iteração depende do número de pessoas e da cultura de desenvolvimento
- Iterações menores que 1 mês devem ser definidas cuidadosamente e não devem ser maiores que 3 meses

Estrutura do RUP



Comparação entre modelos de processos – Modelo em Cascata

- Prós do modelo em cascata
  - É fácil de entender e planejar.
  - Funciona para projetos pequenos e bem compreendidos.
  - A análise e o teste são simples e diretos.
- Contras do modelo em cascata
  - Não se adapta bem a mudanças.
  - O teste ocorre nas fases finais do processo.
  - A aprovação do cliente vem no final.

Comparação entre modelos de processos – Prototipação

- Prós da prototipação
  - O impacto das alterações aos requisitos é reduzido.
  - O cliente se envolve bastante e desde o início.
  - Funciona bem para projetos pequenos.
  - A probabilidade de rejeição do produto é reduzida.
- Contras da prototipação
  - O envolvimento do cliente pode causar atrasos.
  - Pode haver a tentação de “embalar” o protótipo.
  - Desperdiça-se trabalho em um protótipo descartável.
  - É difícil de planejar e gerenciar.

Comparação entre modelos de processos – Modelo Espiral

- Prós do modelo espiral
  - Há envolvimento contínuo dos clientes.
  - Os riscos de desenvolvimento são gerenciados.
  - É apropriado para modelos grandes e complexos.
  - Funciona bem para artefatos extensíveis.
- Contras do modelo espiral
  - Falhas de análise de risco podem fadar o projeto ao fracasso.
  - O projeto pode ser difícil de gerenciar.
  - Exige uma equipe de desenvolvimento especializada.

Comparação entre modelos de processos – Processo Unificado

- Prós do processo unificado
  - A documentação de alta qualidade é enfatizada.
  - Há envolvimento contínuo dos clientes.
  - Adapta-se a alterações aos requisitos.
  - Funciona bem para projetos de manutenção.
- Contras do processo unificado
  - Os casos de uso nem sempre são precisos.
  - A integração de incrementos de *software* é complicada.
  - A sobreposição das fases pode causar problemas.
  - Exige uma equipe de desenvolvimento especializada.