ENGENHARIA DE SOFTWARE

Introdução

AGENDA

- Conceitos de Engenharia de Software
- Processo de desenvolvimento de software

ENGENHARIA DE SOFTWARE

CONCEITOS

CENÁRIO INICIAL

- Desenvolvimento informal e não suficiente
- Software tinham anos de atraso no desenvolvimento
- Custo geralmente acima do previsto
- Desempenho insatisfatório

CRISE DO SOFTWARE (1968)

- Custo de hardware caia
- Custo de software subia

Surgia a ...

Engenharia de Software

ENGENHARIA DE SOFTWARE

"Disciplina da engenharia relacionada com todos os aspectos da produção de software desde estágios iniciais de especificação até a manutenção"

Disciplina de engenharia:

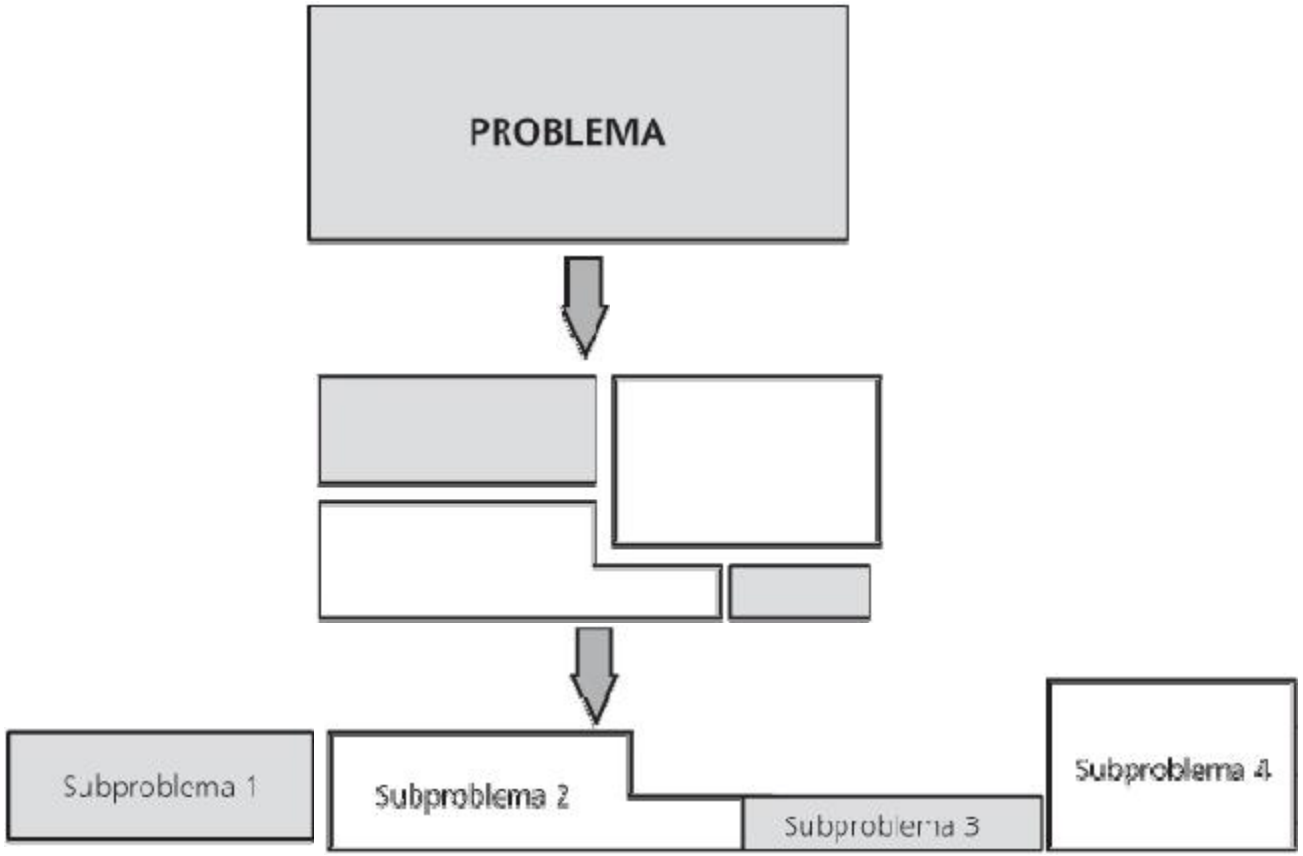
- aplicação de teoria, métodos e ferramentas;
- descoberta de soluções;
- trabalha com restrições organizacionais e financeiras

Todos os aspectos da produção de software:

- técnicos;
- gerenciais;
- desenvolvimento de métodos, teorias e ferramentas de apoio.

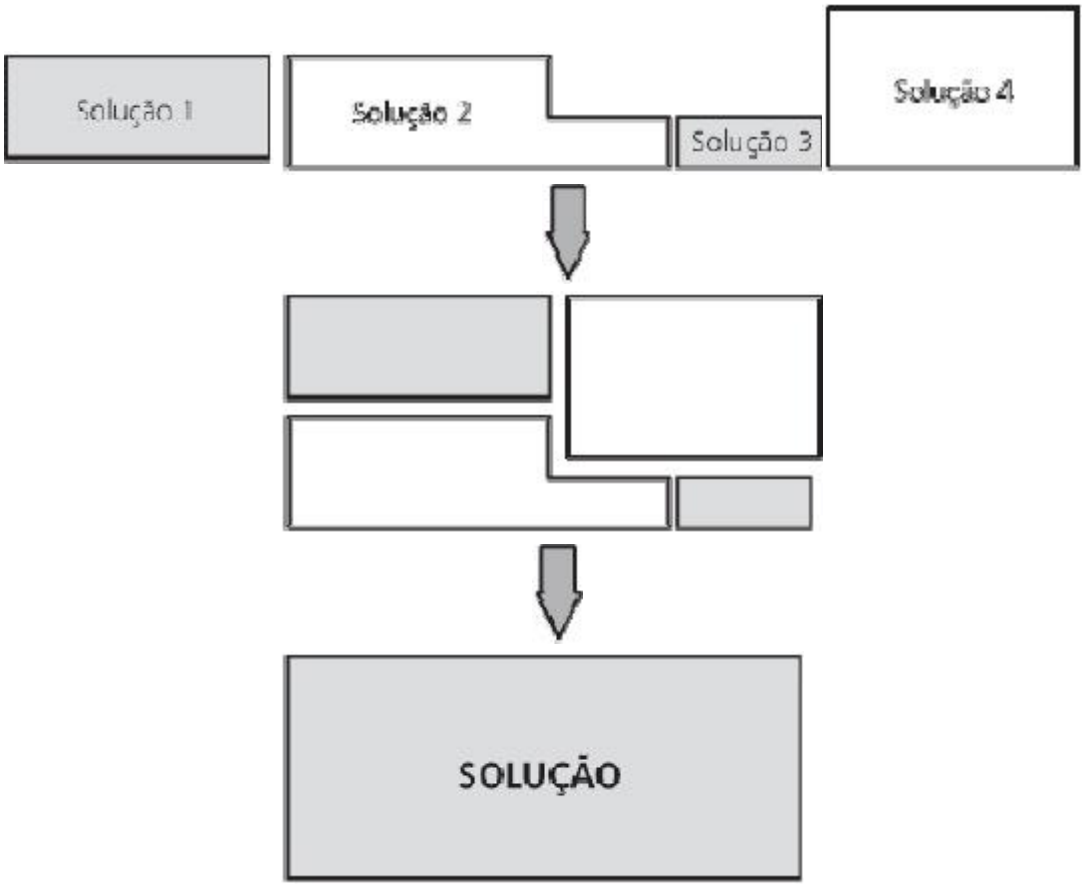
ENGENHARIA DE SOFTWARE

Análise versus síntese de um problema



ENGENHARIA DE SOFTWARE

Análise versus síntese de um problema



O processo de síntese

Por que a engenharia de software?

Método ou técnica: procedimento para o resultado.

E: combinando ingredientes em uma ordem e momentos específicos.

Ferramenta: instrumento ou sistema

Ex.: Máquina de escrever, tesoura, etc.

Procedimento: receita de combinação de ferramentas e técnicas.

Ex.: plano de testes.

Paradigma: estilo de fazer algo, representa uma abordagem ou filosofia para a construção de software

Ex.: cozinhas: francesa, chinesa, orientado a objetos, procedural.

ENGENHARIA DE SOFTWARE

"Desenvolvimento de software com alta qualidade dentro de custos adequados"

O QUE É UM SOFTWARE?

Programa, dados, documentos, configuração, sites, ou seja, todos os elementos necessários para que o programa funcione.

Podem ser:

- genéricos;
- por encomenda.

CARACTERÍSTICAS DO SOFTWARE

Software é intangível e abstrato

Não é limitado por materiais, controlado por leis da física ou processos de manufatura

- Não existe limitação física ou potencial para software.
- Pode facilmente se tornar extremamente complexo e difícil de ser compreendido.

Abordagem de sistemas

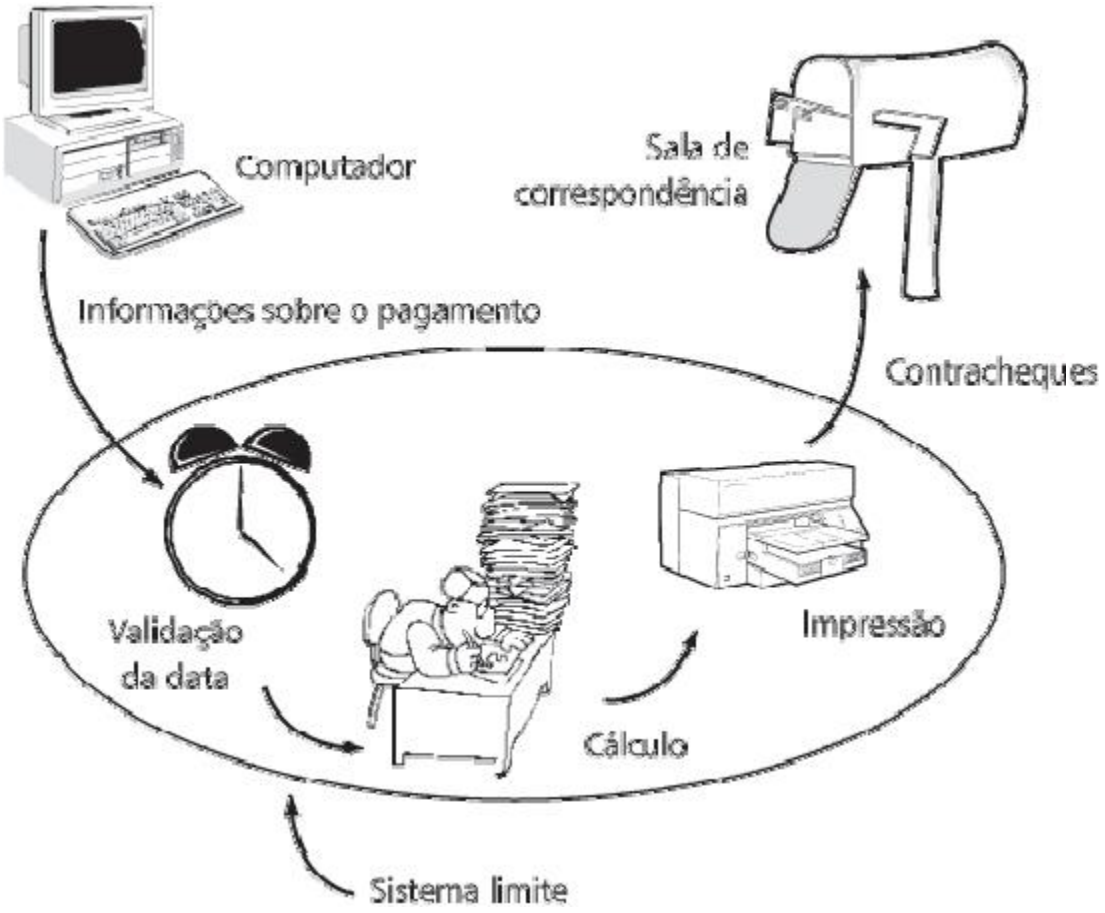
Identificar atividades e objetos.

Definir as relações e fronteiras do sistema.

Considerar sistemas inter-relacionados.

Participantes no desenvolvimento de software

Abordagem de sistemas

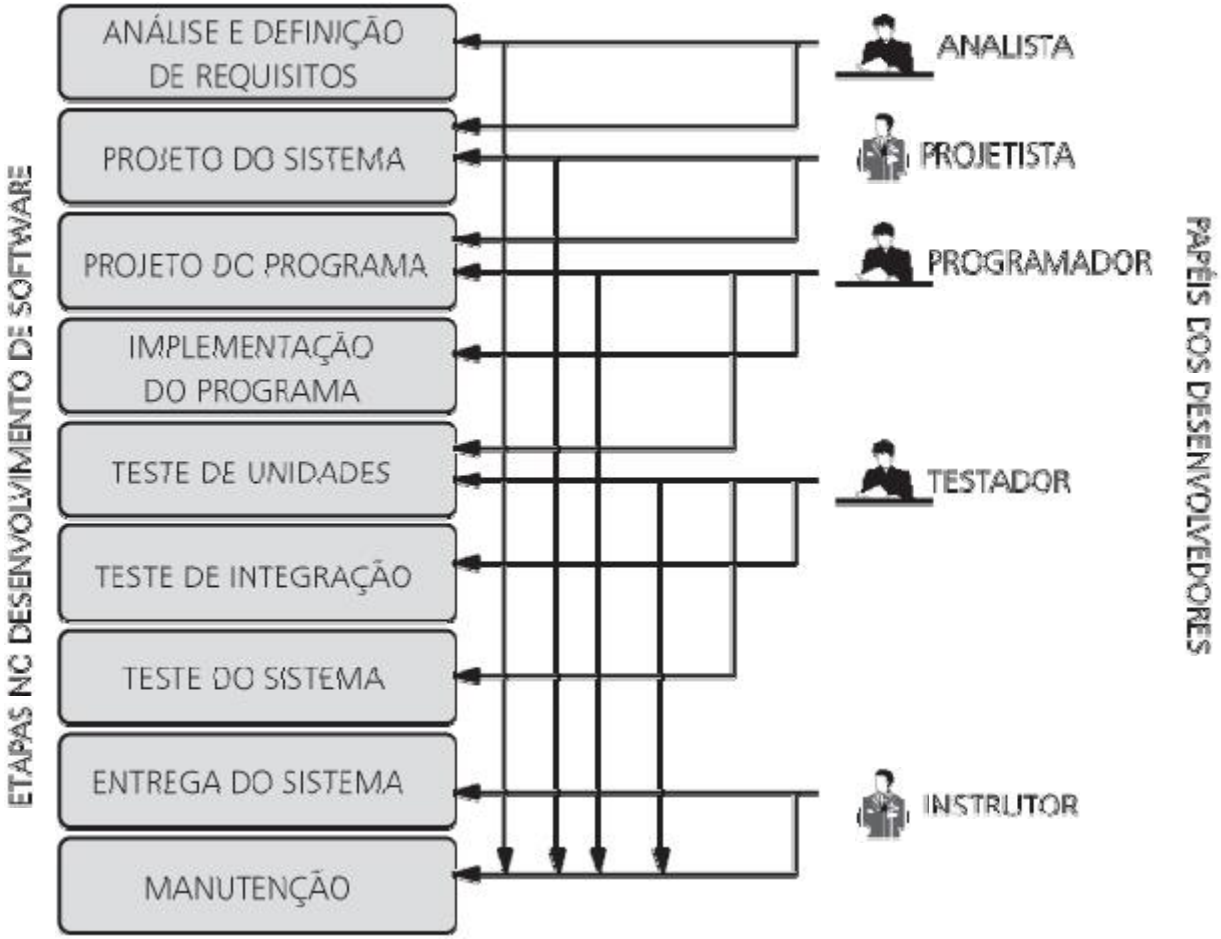


Definição do sistema de produção de contracheques

CICLO DE VIDA DE UM SOFTWARE

Um software possui um ciclo de vida. Começa desde os primeiros estágios de especificação, passa pela sua construção, manutenção e termina quando este é desativado.

Membros de uma equipe de desenvolvimento



Os papéis da equipe de desenvolvimento

CUSTOS DA ENGENHARIA DE SOFTWARE

60 % no desenvolvimento

40% nos testes

Custos de evolução (software de encomenda) excedem o custo do software

DESAFIOS DA ENGENHARIA DE SOFTWARE

- Heterogeneidade:
  - inclui execução em plataformas distintas, redes heterogêneas, sistemas legados.
  - Desenvolver técnicas para construção de software que seja flexível para adaptar-se a heterogeneidade
- Entrega:
  - técnicas demandam tempo e são necessárias para obter qualidade
- Confiança:
  - sistemas críticos, sistemas web, etc.

ATRIBUTOS DE UM BOM SOFTWARE

- Facilidade de manutenção
  - permitir evoluções (as empresas são dinâmicas).

- Confiança: confiabilidade, proteção e segurança
  - Não deve causar danos físicos ou econômicos no caso de falha do sistemas.
- Eficiência:
  - não deve desperdiçar recursos do sistema.
  - Eficiência inclui tempo de resposta, tempo de processamento e utilização de memória.
- Usabilidade:
  - usável, sem esforço excessivo. Apresentar boa interface com o usuário e documentação adequada.

MODULARIDADE (1)

Software é dividido em componentes

Os componentes são nomeados e chamados de módulos

Integra-se estes componentes para satisfazer os requisitos do sistema.

Dividir para conquistar

Cuidado: Dividir demais pode ser prejudicial

Aumenta o número de módulos => aumenta o esforço de integração

MODULARIDADE (2)

Como saber se um módulo está no tamanho adequado?

- Decompor o módulo de maneira que ele diminua a complexidade do problema.
- Permitir que módulos possam ser integrados para serem reaproveitados.
- Ser uma unidade autônoma.
- Permitir a continuidade e reduzir efeitos colaterais.
  - É possível fazer evoluções localizadas.
- Oferecer proteção.
  - Problemas devem ser contidos no módulo.

MODULARIDADE : COESÃO

Realizar uma única tarefa
sem muito relacionamento com o mundo externo.

MODULARIDADE : ACOPLAMENTO

- Medida de interconexão entre os módulos.
- Depende da complexidade de interação entre os módulos.

SITUAÇÃO IDEAL

Independência modular

alta coesão e
baixo acoplamento

MODULARIDADE (3)

Como saber se definimos os módulos corretos?

- Definir módulos cuja função seja previsível:
  - caixas pretas.
- Minimizar estrutura de alta convergência.
- Não deixe que um módulo tenha muita influencia sobre outro.
- Avaliar as interfaces para reduzir a complexidade.

Disciplina de engenharia de software

Noções fundamentais que formam a base de uma disciplina de engenharia de software efetiva (algumas)

- Abstração
- Métodos e notações de análise e projeto
- Protótipo da interface com o usuário
- Arquitetura de software
- Processo de software
- Reuso
- Medição
- Ferramentas e ambientes integrados

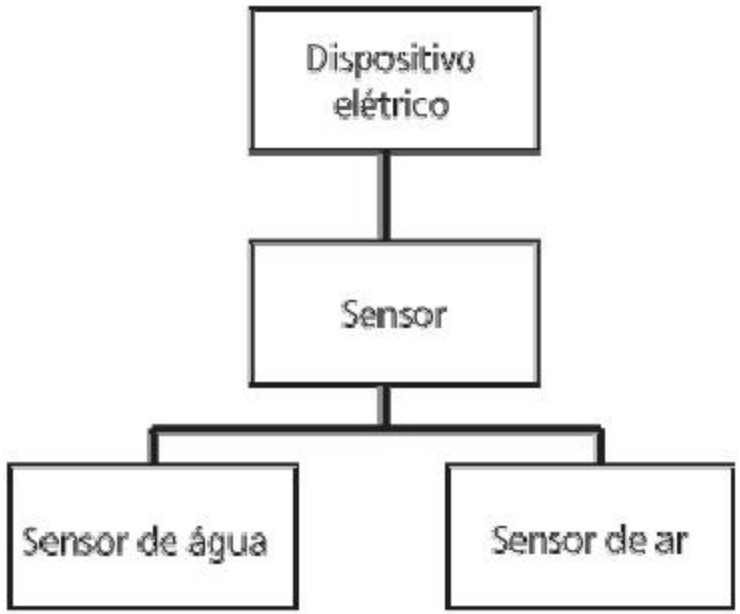
Disciplina de engenharia de software

Abstração

Descrição de um problema

nível de generalização

permite concentrar nos aspectos principais do problema, sem se perder nos detalhes



Hierarquia simples de monitoração de equipamento

Disciplina de engenharia de software

Arquitetura de software

Descreve o sistema em termos de um conjunto de unidades arquitetônicas

Mapa de como essas unidades se relacionam entre si

Componentes e conectores

1. decomposição modular – baseada na atribuição de funções aos módulos
2. decomposição orientada a dados – baseada em estruturas de dados externas
3. decomposição orientada a eventos – baseada nos eventos com os quais o sistema deve lidar
4. projeto 'de fora para dentro' – baseado nas entradas dos usuários no sistema
5. projeto orientado a objetos – baseado na identificação de classes de objetos e suas inter-relações

PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

O que é processo?

Conjunto de tarefas ordenadas

Uma série de etapas

Envolvem:

atividades, restrições e recursos para alcançar a saída desejada

Quando envolve a elaboração de um produto

podemos chamar de ciclo de vida

Processo de Software

Um conjunto de atividades e resultados associados que geram um produto de software.

Processo de desenvolvimento de software

Ciclo de vida do software

Descreve a 'vida' do produto de software

Concepção, implementação, entrega, utilização e manutenção

Processo de Software

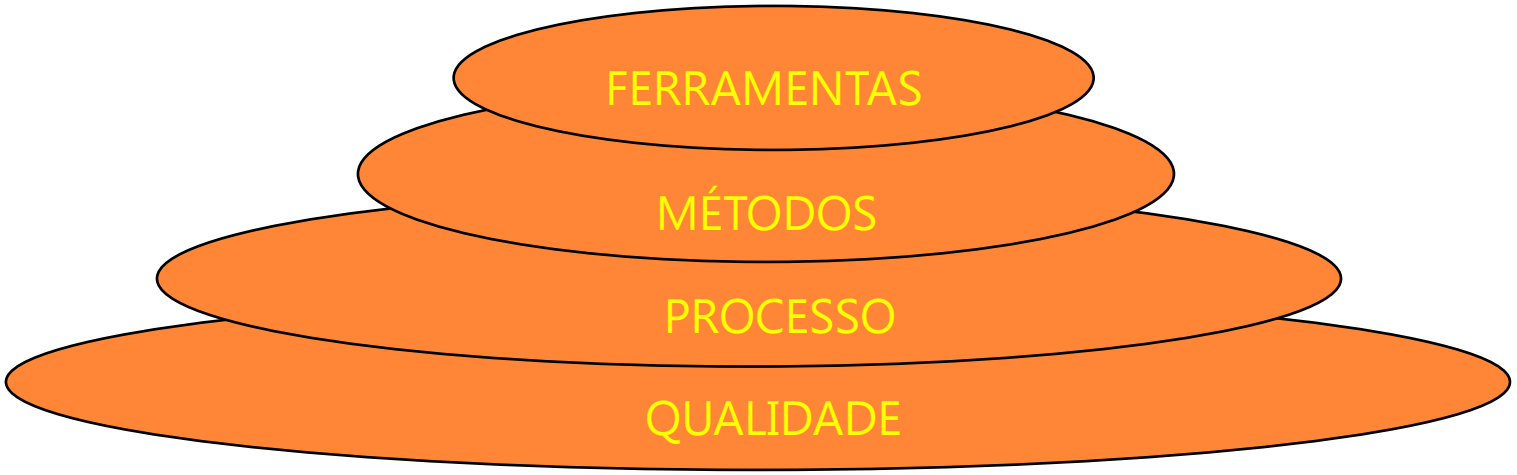
Organizações Diferentes
Estruturas Diferentes → Processos Diferentes

As Atividades Fundamentais são as mesmas:

- Especificação
- Projeto e implementação (design)
- Validação
- Evolução

PROCESSO DE SOFTWARE

Conjunto de atividades e resultados associados que produzem um produto de software



ATIVIDADES PRINCIPAIS DE UM PROCESSO DE SOFTWARE

- Especificação de software
  - definição de funcionalidades e restrições.
- Desenvolvimento de software
  - o software é projetado e programado de acordo com a especificação.
- Validação de software
  - o software é verificado para garantir que é o que o cliente deseja.
- Evolução de software
  - o software é adaptado à mudanças.

ASPECTOS IMPORTANTES

- Diferentes tipos de sistemas precisam de diferentes processos.

- Atividades são organizadas de diferentes maneiras e descritas em diferentes níveis de detalhe.

Processos - Elementos

Todas as principais atividades do processo

Recursos

está sujeito a um conjunto de restrições (como um cronograma)

Produtos intermediários e finais

Subprocessos

com hierarquia ou organizados de algum modo

Possibilidade de modificar tendo a receita para documentar o processo.

Critérios de entrada e saída para cada atividade

Seqüência de atividades

de modo que a ordem de execução de uma para outra seja clara

Conjunto de diretrizes que explicam os objetivos de cada atividade

Restrições e controles para cada atividade, recurso ou produto

EXEMPLO DE PROCESSO (1)

Bolo de chocolate com cobertura

Processo: fazer um bolo de chocolate com cobertura.

Procedimentos: comprar os ingredientes, encontrar os utensílios de cozinha apropriados.

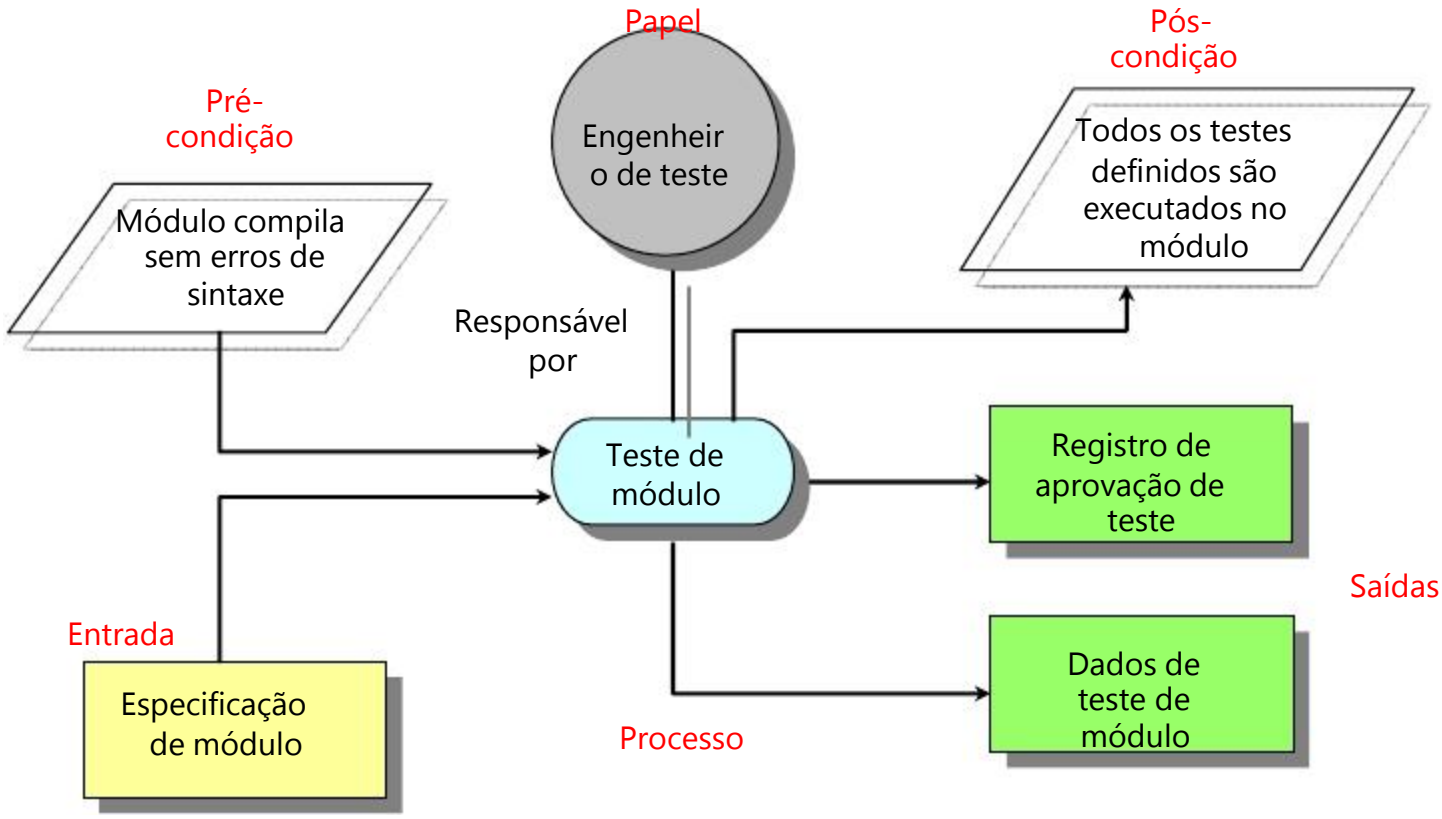
Receita: descrição do procedimento de fazer a massa e assar o bolo. Contém atividades, restrições e recursos.

Atividades: bater os ovos antes de misturar os outros ingredientes.

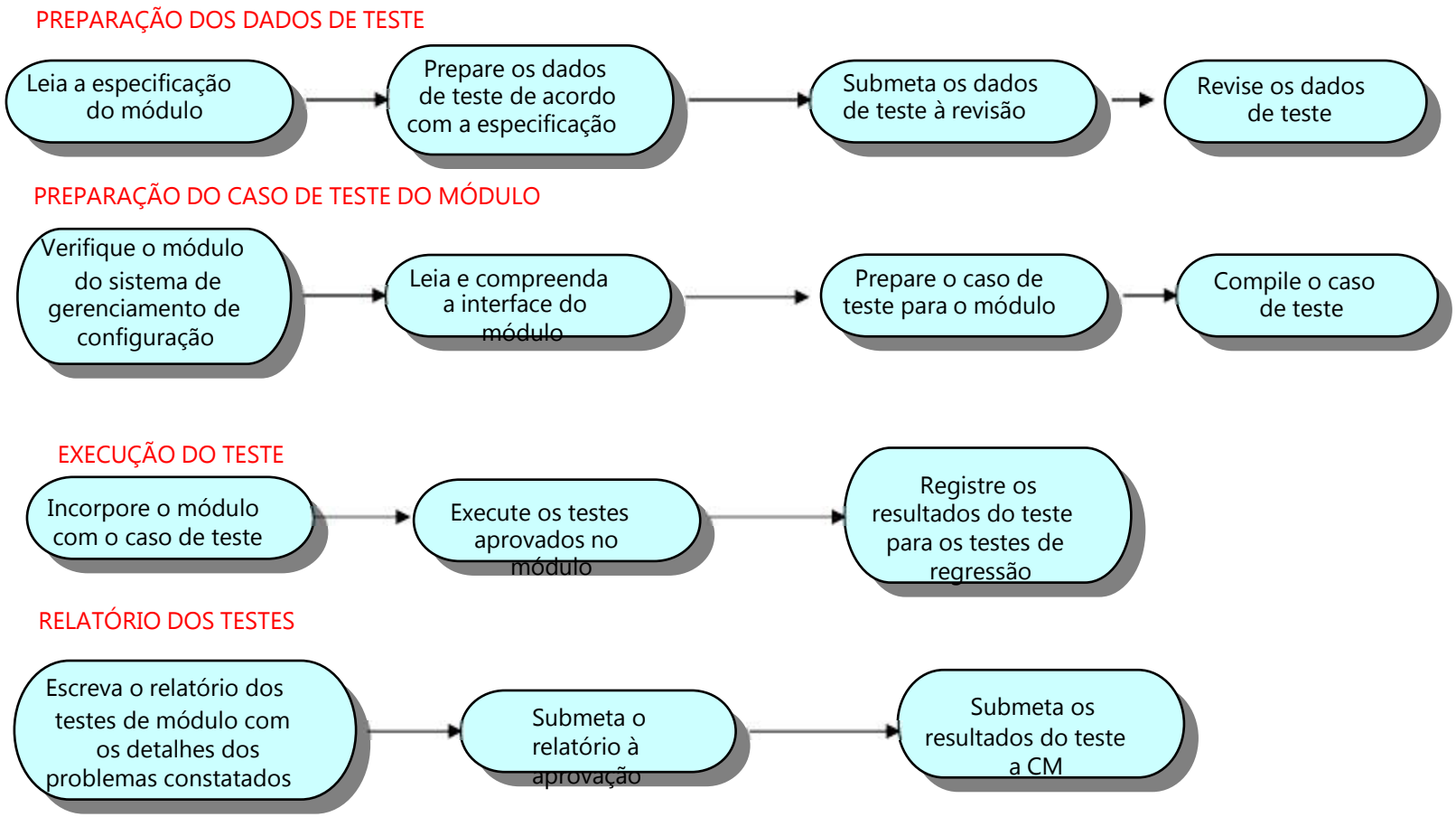
Restrições: especificação da temperatura 'esquente o chocolate até derreter antes de misturar o açúcar'.

Recursos: farinha, açúcar, ovos e chocolate.

EXEMPLO DE PROCESSO (2)



EXEMPLO DE PROCESSO (3)



(SOMMERVILLE, 2007)

Ainda sobre processo....

- Cada estágio é por si só um processo
  - ou coleção de processos
  - podem ser descritos como conjunto de atividades
- Cada atividade envolve restrições, resultados e recursos
  - Exemplo.
    - A análise e definição dos requisitos precisa ter como entrada inicial uma declaração das funções e características desejadas para o produto, expressas pelo usuário
- Cada processo pode ser descrito de várias maneiras
  - utilizando texto, figuras ou uma combinação desses recursos.
- Existem várias maneiras de se fazer essa descrição
  - geralmente organizada como modelo que contém as principais características do processo

Razões para modelar um processo

Representações abstratas de processos

Formar um entendimento comum

Encontrar inconsistências, redundâncias e omissões

Encontrar e avaliar

- atividades propostas mais adequadas aos objetivos

Fazer um processo geral para uma situação particular na qual ele será utilizado

Reuso e Customização

Modelos de Processo

PSP (*Personal Software Process* – Processo Pessoal de Software)

Profissional controla a qualidade do produto

Cinco atividades: planejamento, projeto de alto nível, revisão de projeto de alto nível, desenvolvimento, pós-conclusão.

Planejamento: define os requisitos e realiza estimativa (tamanho e recursos)

Projeto de alto nível: especificações dos componentes

Revisão do projeto de alto nível: verificação formal para descobrir erros

Desenvolvimento: projeto do componente é refinado (e revisado)

Pós-conclusão: verifica a efetividade através de métricas (processo pode ser modificado)

Modelos de Processo

TSP (*Team Process Software* – Processo de Equipe de Software)

Objetivos: (i) construir equipes auto dirigidas; (ii) mostrar aos gerentes como acompanhar e motivar suas equipes; (iii) acelerar o aperfeiçoamento do processo de software.

Exercício: Considerando o TSP, apresente os benefícios (quantitativos e qualitativos) que podemos obter com a utilização do processo

Modelos de Processo

TSP – Benefícios de uma equipe autodirigida:

Entendimento consistente de metas e objetivos

Monitora produtividade e qualidade

Identifica um processo de equipe apropriado

Define normas

Avalia continuamente o risco e ...

Emite relatórios

Trabalho 1 – Entrega 5/4/16

- Pesquise e faça um texto dando a descrição e características dos modelos de processos de processos de softwares em Engenharia de Software

  - Modelo em Cascata
  - Modelo Incremental
  - Modelo Evolucionário

  No final, faça uma análise sobre quais vantagens e desvantagens cada modelo possui.