FUCAPI

Pós-graduação Lato Sensu em Sistemas de Informações e Aplicações WEB

Módulo de Arquitetura para Internet

Prof. Tiago Eugenio de Melo, M.Sc.
E-mail: tiago@comunidadesol.org

Roteiro

- Introdução à Engenharia de Software
- Processos de desenvolvimento de software
- Metodologias de desenvolvimento de software
- Qualidade de processo e produto de software
- Padrões WEB
- Referências bibliográficas

Engenharia de Software

- A economia de todos os países do mundo dependem do uso de software.
- Cada vez mais o controle dos processos tem sido feito por software.
- A Engenharia de Software consiste no conjunto de **teorias**, **métodos** e **ferramentas** para o desenvolvimento profissional de software.

Engenharia de Software

- Atualmente, os custos de software superam os custos de hardware.
- A manutenção de software é onde se tem os maiores gastos. Principalmente nos sistemas de vida longa.

Questões preliminares

- O que é sistema?
  - É um conjunto de elementos concretos ou abstratos entre os quais se pode encontrar alguma relação.

Questões preliminares

- Quais são os elementos de um sistema?
  - São os elementos que estão dentro da fronteira conceitual.
  - São os elementos que possuem interações fortes entre si.
  - São os elementos que possuem interações fracas com os elementos externos ao sistema.

Questões preliminares

- O que é fronteira conceitual?
  - É o limite que separa o que está dentro do sistema do resto.
    - Os elementos de dentro do sistema devem ser detalhados.
    - Deve-se verificar a interação destes elementos com o ambiente externo.

Questões preliminares

- O que é software?
  - Programas de computador e documentação associada.
  - Existem duas categorias de produtos de software:
    - Produtos genéricos: sistemas produzidos e vendidos no mercado a qualquer pessoa que possa comprá-los.
    - Produtos específicos: sistemas encomendados especialmente por um determinado cliente.

Questões preliminares

- O que é Engenharia de Software?
  - É uma das áreas da Engenharia que trata dos aspectos de produção de software.
  - A Engenharia de Software tem como objetivo estabelecer uma sistemática abordagem de desenvolvimento, através de ferramentas e técnicas apropriadas, dependendo do problema a ser abordado, considerando as restrições e recursos disponíveis.

Questões preliminares

- O que é Engenharia de Software?
  - Em resumo, visa resolver problemas inerentes ao **processo** e ao **produto** de software.

Questões preliminares

- Quais são os princípios da Engenharia de Software?
  - Formalidade: produtos mais confiáveis, podem controlar seu custo e ter mais confiança no seu desempenho.
  - Abstração: identificar os aspectos importantes ignorando os detalhes.
  - Decomposição: subdividir o processo em atividades específicas, atribuídas a diferentes especialistas.

Questões preliminares

- Quais são os princípios da Engenharia de Software?
  - Generalização: sendo mais geral, é possível que a solução possa ser reutilizada.
  - Flexibilização: modificação com qualidade.

Questões preliminares

- Qual é a diferença entre Engenharia de Software e Ciência da Computação?
  - A Ciência da Computação tem como objetivo o desenvolvimento de teorias e fundamentações; enquanto a Engenharia de Software se preocupa com as práticas de desenvolvimento de software.

Questões preliminares

- Qual é a diferença entre Engenharia de Software e Ciência da Computação?
  - Na prática, as ‘elegantes’ teorias da Ciência da Computação não podem ser aplicadas para os problemas reais e complexos que requerem uma solução de software.

Questões preliminares

- Qual é a diferença entre Engenharia de Sistemas e Engenharia de Software?
  - A Engenharia de Sistemas trata dos sistemas baseados em computadores, que inclui hardware e software. Enquanto a Engenharia de Software trata apenas dos aspectos de desenvolvimento de software.

Questões preliminares

- O que é um processo de software?
  - É um conjunto de atividades que objetivam o desenvolvimento e evolução de software.
  - Ele começa na concepção do problema (solicitação do usuário) e termina quando o sistema sai de uso.

Questões preliminares

- Quais são as principais atividades de um processo de desenvolvimento de software?
  - Especificação: define o que o sistema deverá fazer e as suas restrições.
  - Desenvolvimento: produção do software.

Questões preliminares

- Quais são as principais atividades de um processo de desenvolvimento de software?
  - Validação: checagem se o software faz o que o usuário quer.
  - Evolução: mudanças no software para atender às novas demandas.

Questões preliminares

- O que é um modelo de processo de software?
  - É a representação simplificada de um processo de software, apresentada sob uma perspectiva específica.
  - Exemplos de perspectivas:
    - Fluxo de trabalho (workflow): seqüência de atividades.
    - Fluxo de dados: fluxo das informações.
    - Papel/Ação: quem faz o que.

Questões preliminares

- Quais são os principais modelos?
  - Cascata ou sequencial.
  - Modelo evolutivo.
  - Transformação formal.
  - Integração de componentes reusáveis.

Questões preliminares

- Quais são os objetivos dos modelos?
  - Auxiliar no processo de produção, ou seja, gerar produtos com qualidade, produzidos mais rapidamente e a um custo cada vez menor.

Questões preliminares

- Quais são os objetivos dos modelos?
  - Possibilitam:
    - Ao gerente: controlar o processo de desenvolvimento de sistemas de software.
    - Ao desenvolvedor: obter a base para produzir, de maneira eficiente, softwares que satisfaçam os requisitos pré-estabelecidos.

Questões preliminares

- Quais são os custos da Engenharia de Software?
  - Pesquisas mostram que 60% dos custos é para o desenvolvimento e 40% para os testes.
  - O custo de evolução do software, normalmente, excede o custo de desenvolvimento.
  - O custo depende do tipo de sistema a ser desenvolvido e as suas restrições.
  - A distribuição dos custos depende do modelo de desenvolvimento adotado.

Questões preliminares

- O que são os métodos de Engenharia de Software?
  - São as abordagens estruturadas para o desenvolvimento de software que incluem os modelos de software, notações, regras e maneiras de desenvolvimento.

Questões preliminares

- O que são ferramentas CASE?
  - Softwares que têm o objetivo de fornecer um suporte automatizado para as atividades de processo de software.
  - Operam em dois níveis:

- Alto nível: ferramentas que suportam as atividades iniciais de requisitos e projetos.

- Baixo nível: ferramentas que suportam as atividades de programação, depuração e testes.

Questões preliminares

- Quais são os atributos de um bom software?
  - Atender às funcionalidades exigidas.
  - Eficiente.
  - Manutenível.
  - Usável.

Questões preliminares

- Quais são questões atuais?
  - Sistemas legados: os sistemas antigos devem ser mantidos e atualizados (atividades de migração).
  - Heterogeneidade: sistemas são uma combinação de hardware e software.
  - Prazos de entrega: pressão para um menor prazo de entrega.

Evolução do Software

- 1950 – 1965
  - O hardware sofreu contínuas mudanças.
  - O desenvolvimento de software era considerado como uma atividade secundária, em que havia poucos métodos sistemáticos.
  - O hardware era de propósito geral.
  - O software era específico para cada aplicação.
  - Não havia documentação.

Evolução do Software

- 1965 – 1975
  - Surgimento das técnicas de multiprogramação e multiusuários.
  - Técnicas interativas.
  - Sistemas em tempo real.
  - Produtos de software.
  - Bibliotecas de software.
  - Manutenção quase impossível.

Evolução do Software

- 1975 – ontem
  - Sistemas distribuídos.
  - Redes locais e globais.
  - Uso generalizado de microprocessadores.
  - Hardware de baixo custo.

Evolução do Software

- Atualmente
  - Tecnologias orientadas a objetos.
  - Sistemas especialistas e softwares de inteligência artificial usados na prática.
  - Computação paralela.
  - Desenvolvimento de sistemas para a WEB.

Crise de Software

- Conjunto de problemas encontrados no desenvolvimento de software:
  - Estimativas de prazo e de custo são frequentemente imprecisas.
  - A produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços.
  - A qualidade do software está abaixo do esperado.
  - O software existente é muito difícil de manter.

Mitos de Software

- Mito: já temos um manual completo com padrões e procedimentos para a construção de software. Isso não é suficiente ao que a equipe precisa saber?
- Realidade: será que o manual é usado? Será que os profissionais sabem que ele existe? Ele reflete as práticas mais modernas de desenvolvimento? Ele está completo?

Mitos de Software

- Mito: a equipe já tem as melhores ferramentas de desenvolvimento de software. Isto já é o suficiente.
- Realidade: é necessário muito mais que computadores recentes e programas atualizados para fazer um desenvolvimento de software de alta qualidade.

Mitos de Software

- Mito: se houver um atraso no prazo, basta adicionar mais programadores para recuperar.
- Realidade: o desenvolvimento de software não é um processo mecânico igual à manufatura. Acrescentar pessoas em um projeto pode torná-lo ainda mais atrasado.

Mitos de Software

- Mito: uma declaração geral dos objetivos é suficiente para se começar a programar. Posteriormente pode-se preencher os detalhes.
- Realidade: uma definição inicial incipiente é a principal causa de fracassos dos esforços de desenvolvimento de software. É fundamental uma descrição formal e detalhada do domínio da informação, função, desempenho, interfaces, restrições de projeto e critérios de validação.

Mitos de Software

- Mito: os requisitos de projeto modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas, pois o software é flexível.
- Realidade: uma mudança, quando solicitada tardiamente num projeto, pode ser maior do que a ordem de magnitude mais dispendiosa da mesma mudança solicitada nas fases iniciais.

Mitos de Software

- Mito: assim que os programadores terminarem o programa o trabalho da equipe estará completo.
- Realidade: os dados da indústria indicam que entre 50 e 70% de todo esforço gasto num programa serão dispendidos depois que ele for entregue pela primeira vez ao cliente.

Mitos de Software

- Mito: enquanto não tiver o programa funcionando, eu não terei nenhuma maneira de avaliar sua qualidade.
- Realidade: um programa funcionando é somente uma parte de uma configuração de software que inclui todos os itens de informação produzidos durante a construção e manutenção do software.

Questões de Ética

- Confidencialidade
  - Os engenheiros devem respeitar a confidencialidade dos projetos dos seus empregadores e clientes.
  - Existência de contratos de confidencialidade.

Questões de Ética

- Surgimento de dilemas
  - Quando o analista não está de acordo com as políticas do engenheiro sênior.
  - Quando o empregador age de maneira não ética e as atualizações de sistemas críticos terminam sem a realização dos devidos testes.
  - Participação em projetos que contrariam os princípios do analista. Exemplo: desenvolvimento de projetos militares.

Questionário

- Qual é a importância da Engenharia de Software na sociedade atual?
- O que você entende por software?
- Comente sobre dois princípios da Engenharia de Software.
- Quais são as principais atividades de um processo de desenvolvimento de software?

Questionário

- Qual é a importância da etapa de manutenção no processo de desenvolvimento de software?
- Qual é o papel das ferramentas CASE no desenvolvimento de software?
- O que são sistemas legados? Qual é a dificuldade em trabalhar com sistemas legados ao se desenvolver um novo software?

Questionário

- Por que há os gastos na manutenção de software são altos?
- Qual é a diferença entre software genérico e software específico?
- Comente dois princípios da Engenharia de Software.
- Qual é a diferença entre validação e verificação?

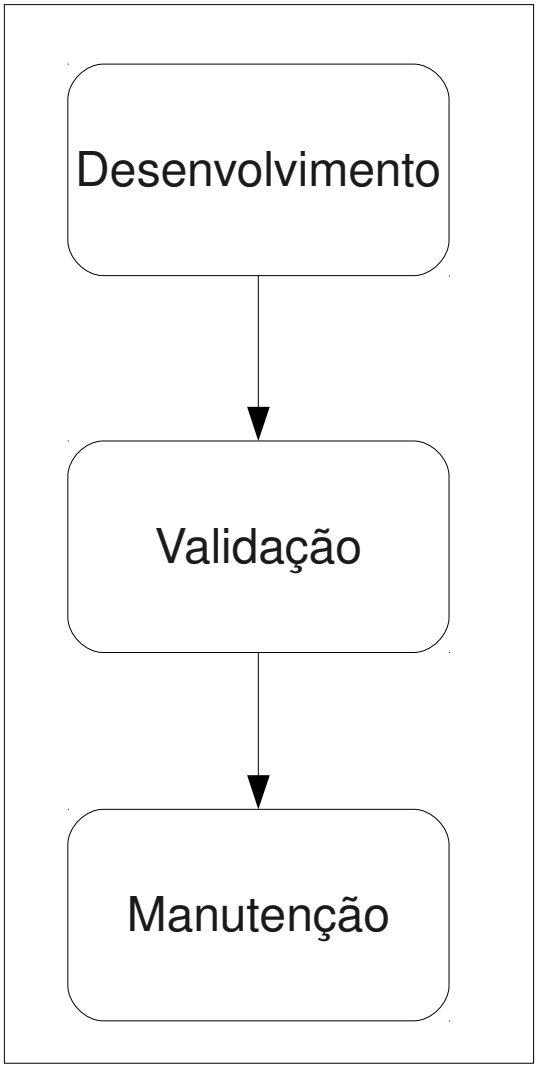
Processos de Desenvolvimento de Software

Processos de desenvolvimento de software

- Definição:
  - Conjunto de atividades para especificar, projetar, implementar e testar sistemas de software.
- As atividades necessárias para o desenvolvimento de software são:
  - Especificação.
  - Projeto.
  - Validação.
  - Evolução.

Processos de desenvolvimento de software

- Etapas:



Processos de desenvolvimento de software

- Ciclo de Vida Clássico (Modelo Cascata)
  - Diferentes fases da especificação e desenvolvimento.
- Desenvolvimento Evolutivo
  - Especificação e desenvolvimento são alternados.
- Desenvolvimento Espiral
  - Faz uma combinação do ciclo de vida clássico e evolutivo.

Processos de desenvolvimento de software

- Desenvolvimento Formal
  - Uso de modelo matemático é formalmente transformado em uma implementação.
- Desenvolvimento Baseado em Reuso
  - O sistema é montado a partir de componentes.

Ciclo de vida clássico

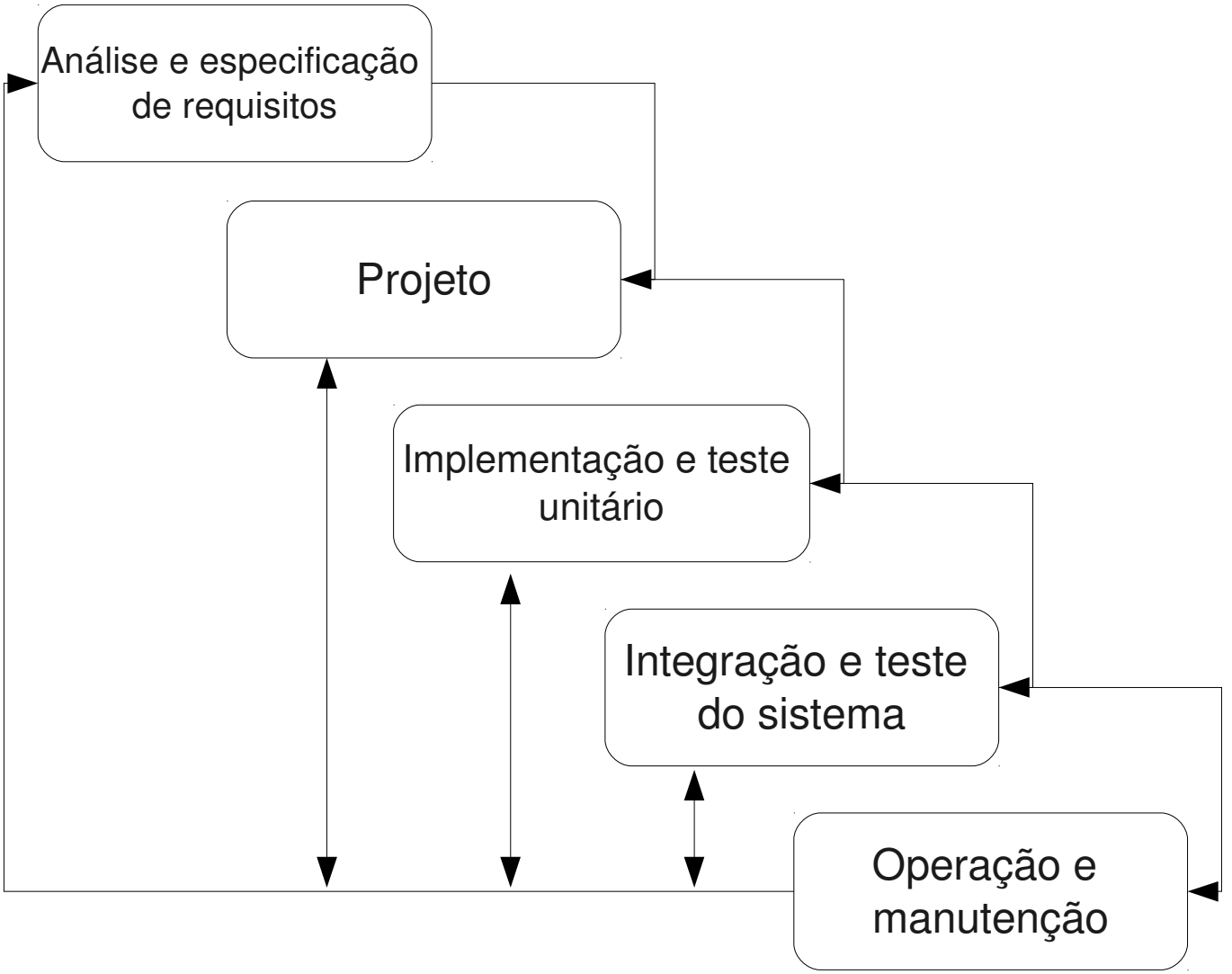
- Método é sistemático e seqüencial.
- O resultado de uma fase se constitui na entrada de outra.
- Também conhecido com cascata.
- Cada fase é estruturada como um conjunto de atividades que podem ser executadas por pessoas diferentes, simultaneamente.

Ciclo de vida clássico

- Fases:
  - Análise e especificação dos requisitos.
  - Projeto do software.
  - Implementação e teste unitário.
  - Integração e teste do sistema.
  - Operação e manutenção.

Ciclo de vida clássico

- Fases:



Ciclo de vida clássico

- Fases de análise e especificação de requisitos:
  - Durante essa fase são identificados, através de consultas aos usuários, os serviços e as metas a serem atingidas, assim como as restrições a serem respeitadas.

Ciclo de vida clássico

- Fases de análise e especificação de requisitos:
  - O resultado dessa fase consiste num documento de especificação de requisitos que tem como objetivos:
    - Ser analisado e confirmado pelo usuário para verificar se ele satisfaz todas as suas expectativas.
    - Ser usado pelos desenvolvedores de software para obter um produto que satisfaça os requisitos.

Ciclo de vida clássico

- O documento gerado
  - Deve ter as seguintes características:
    - Inteligível.
    - Preciso.
    - Completo.
    - Consistente.
    - Não ambíguo.
    - Facilmente modificável.

Ciclo de vida clássico

- Tipos de especificação
  - Funcionais: documenta o que o produto de software faz, usando notações informais, seminformais, formais ou uma combinação delas.
  - Não funcionais: documenta a confiabilidade, acurácia dos resultados, desempenho, problemas de interface, restrições físicas e operacionais, questões de portabilidade, entre outras.

Ciclo de vida clássico

- Tipos de especificação
  - De desenvolvimento e manutenção: documenta os procedimentos de controle de qualidade (teste, prioridades das funções desejadas, mudanças prováveis nos procedimentos de manutenção do sistema, entre outras).

Ciclo de vida clássico

- Fase de projeto
  - A fase de projeto envolve a representação das funções do sistema em uma forma que possa ser transformada em um ou mais programas executáveis.
  - É definida a solução do problema
    - Decomposição do produto sub-sistemas e/ou componentes.
    - Representação das funções do sistema em uma forma que possa ser transformada em programas.

Ciclo de vida clássico

- Fase de projeto
  - Definição de ***como*** o produto deve ser implementado.
  - Pode ser dividido em:
    - Projeto de alto nível (geral).
    - Projeto detalhado.
  - Tem como resultado um documento de especificação do projeto.

Ciclo de vida clássico

- Fase de implementação e teste unitário
  - Na fase de implementação o software é transformado em um programa, ou em unidades de programa, através de uma determinada linguagem de programação.
  - Teste unitário: cada unidade satisfaz suas especificações (planos e casos de teste pré-estabelecidos).
  - Resultado: coleção de programas implementados e testados.

Ciclo de vida clássico

- Fase de integração e teste de sistema
  - Programas ou unidades de programa são integrados e testados como um sistema.
  - Integração incremental: programas ou unidades são integrados à medida em que forem sendo desenvolvidos.
  - Resultado: produto pronto para ser entregue ao cliente.

Ciclo de vida clássico

- Contribuições do modelo
  - O processo de desenvolvimento de software deve ser sujeito à disciplina, planejamento e gerenciamento.
  - A implementação do produto deve ser adiada até que os objetivos tenham sido completamente entendidos.
  - Deve ser utilizado especialmente quando os requisitos estão bem claros no início do desenvolvimento.

Ciclo de vida clássico

- Problemas
  - Utiliza método sistemático e seqüêncial, em que a entrada de uma fase é o resultado da anterior.
  - O reinício do modelo é a dificuldade de acomodar mudanças depois que o processo está no final.
  - Dificuldade em atender às mudanças exigidas posteriormente pelo cliente.

Ciclo de vida clássico

- Problemas
  - Modelo mais adequado quando os requisitos estão muito bem entendidos.
  - Modelo ideal para aplicações maiores e mais complexas.
  - Rigidez, mas o desenvolvimento não é linear.
  - A meta continua sendo tentar a linearidade, para manter o processo previsível e fácil de monitorar.

Ciclo de vida clássico

- Problemas
  - Qualquer desvio é desencorajado, pois vai representar um desvio do plano original e, portanto, requerer um replanejamento.
  - Todo o planejamento é orientado para a entrega do produto de software em uma única data.
  - O processo de desenvolvimento pode ser longo e a aplicação pode ser entregue quando as necessidades do usuário já tiverem sido alteradas.

Desenvolvimento Evolutivo

- Baseado no desenvolvimento e implementação de um produto inicial, que é submetido aos comentários e críticas do usuário.
- O produto vai sendo refinado, através de múltiplas versões, até que o produto de software almejado tenha sido desenvolvido.
- As atividades de desenvolvimento e validação são desempenhadas paralelamente, com uma rápida resposta entre elas.

Desenvolvimento Evolutivo

- Categorias
  - Desenvolvimento Exploratório
    - O objetivo é desenvolver o sistema com o contínuo acompanhamento dos clientes desde da especificação até a entrega do produto. Os requisitos precisam ser bem entendidos.
  - Prototipação
    - O objetivo é entender os requisitos do sistema.

Desenvolvimento Evolutivo

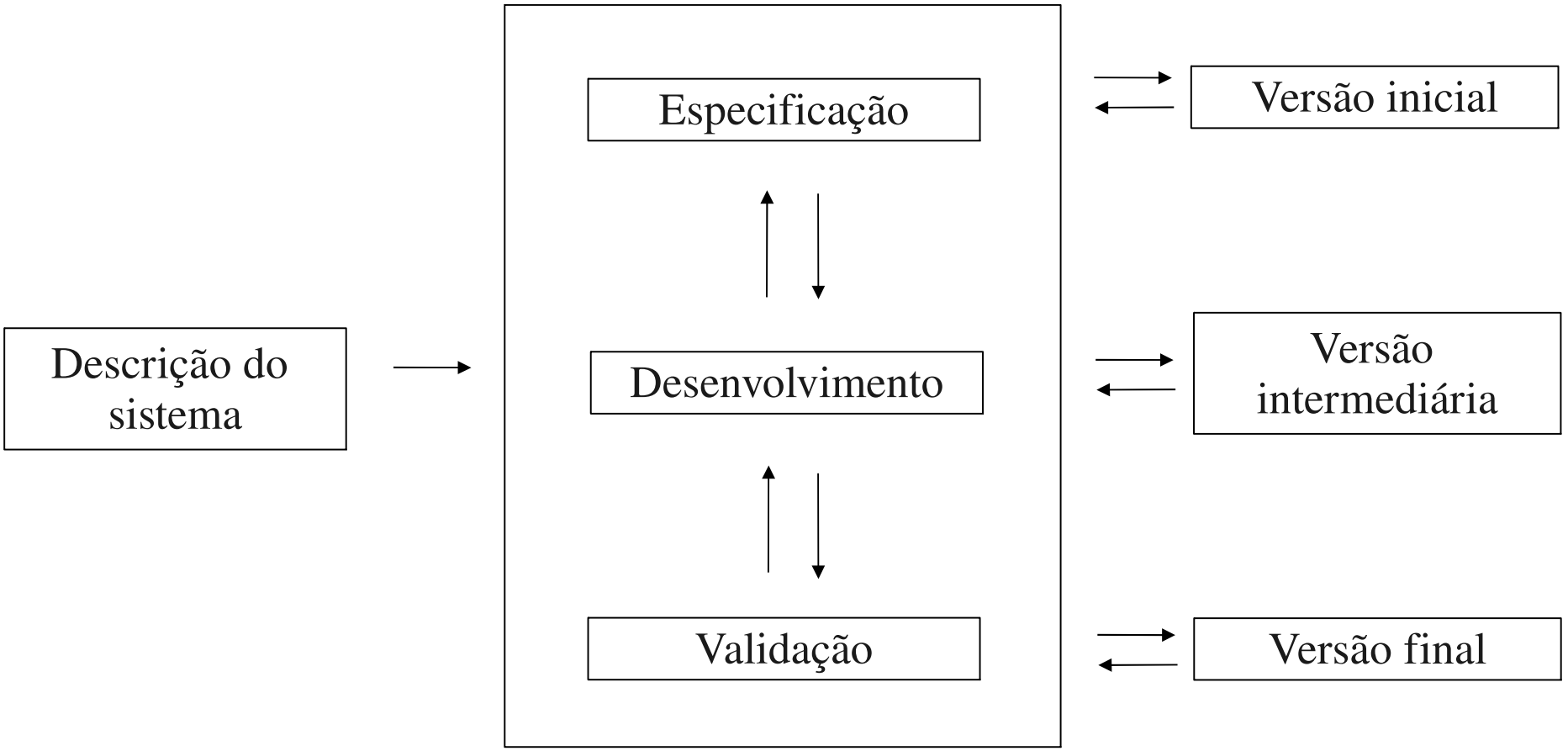
- Desenvolvimento Exploratório
  - O objetivo é trabalhar junto do usuário para descobrir os seus requisitos, de maneira incremental, até que o produto final seja obtido.
  - O desenvolvimento começa com as partes do produto que são melhor entendidas.
  - A evolução acontece quando novas características são adicionadas à medida que são sugeridas pelo usuário.

Desenvolvimento Evolutivo

- Desenvolvimento Exploratório
  - É indicado o seu uso quando é difícil, ou mesmo impossível, estabelecer uma especificação detalhada dos requisitos do sistema a *priori*.
  - A versão inicial do produto é submetida a uma avaliação inicial do usuário.
  - Essa versão é refinada, gerando várias versões, até que o produto almejado tenha sido desenvolvido.

Desenvolvimento Evolutivo

- Desenvolvimento Exploratório



Desenvolvimento Evolutivo

- Prototipação
  - Entender os requisitos do usuário e obter uma melhor definição dos requisitos do sistema.
  - Usado para fazer experimentos com os requisitos que não estão bem entendidos.
  - Enolve projeto, implementação e teste, mas não de maneira formal ou completa.

Desenvolvimento Evolutivo

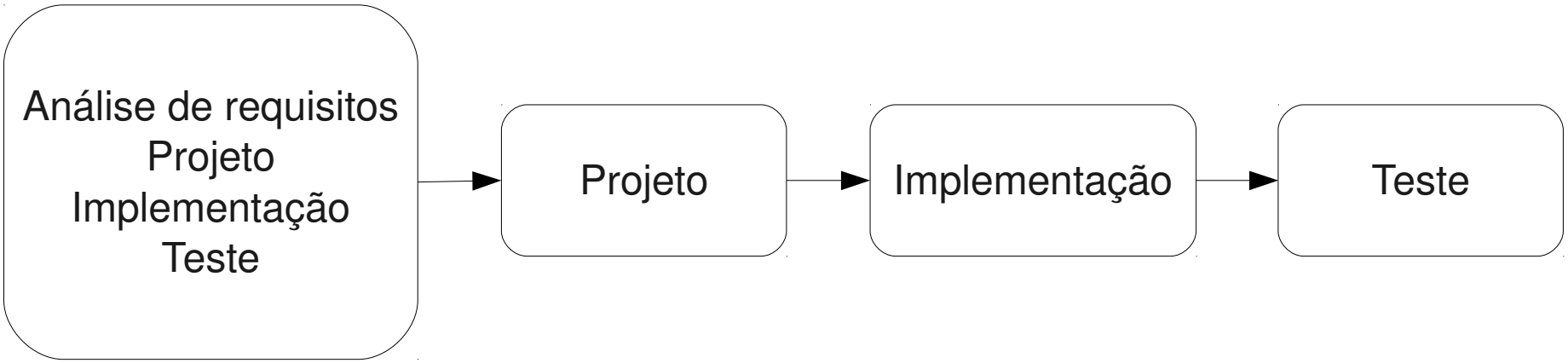
- Prototipação
  - O usuário define uma série de objetivos, mas não consegue identificar detalhes de entrada, processamento ou requisitos de saída.
  - O desenvolvedor está incerto sobre a eficiência de um algoritmo, a adaptação de um sistema operacional, ou ainda sobre a forma de interação homem-máquina.

Desenvolvimento Evolutivo

- Prototipação
  - Possibilita ao desenvolvedor criar um modelo do software que será construído.
  - O desenvolvedor pode perceber as reações iniciais do usuário e obter sugestões para mudar ou inovar o protótipo.
  - O usuário pode relacionar o que vê no protótipo diretamente com os seus requisitos.

Desenvolvimento Evolutivo

- Protótipo descartável
  - O objetivo é entender os requisitos do usuário e, consequentemente, obter uma melhor definição dos requisitos do sistema.



Desenvolvimento Evolutivo

- Problemas
  - Ausência de visibilidade do processo
    - Como o desenvolvimento acontece de maneira rápida, não compensa produzir documentos que reflitam cada versão do produto de software.
  - Sistemas são fracamente estruturados
    - Mudanças constantes tendem a corromper a estrutura do software.

Desenvolvimento Evolutivo

- Problemas
  - Necessidade de ferramentas de rápido desenvolvimento.
  - O usuário vê o que parece ser uma versão em funcionamento do produto de software.
  - O desenvolvedor, muitas vezes, assume certos compromissos de implementação.

Desenvolvimento Evolutivo

- Aplicabilidade:
  - Sistemas de pequeno e médio porte
    - Pois os problemas de mudanças no sistema atual podem ser contornados através da reimplementação do sistema todo quando mudanças substanciais se tornam necessárias.
  - Como parte de um sistema grande (ex.: a interface do usuário).

Desenvolvimento Evolutivo

- Aplicabilidade:
  - Sistema de curta duração.
  - Testes podem ser mais efetivos, visto que testar cada versão do sistema é provavelmente mais fácil do que testar o sistema todo no final.

Desenvolvimento Espiral

- Desenvolvido para englobar as melhores características do ciclo de vida clássico e do paradigma evolutivo.
- Adiciona um novo elemento – a análise de risco – que não existe nos dois paradigmas anteriores.
- Os riscos são as circunstâncias adversas que podem atrapalhar o processo de desenvolvimento e a qualidade do produto a ser desenvolvido.

Desenvolvimento Espiral

- Prevê a eliminação de problemas de alto risco através de um planejamento e projeto cuidadosos.
- As atividades podem ser organizadas como uma espiral que tem muitos ciclos.
- Cada ciclo na espiral representa uma fase do processo de desenvolvimento do software.

Desenvolvimento Espiral

- O primeiro ciclo pode estar relacionado com o estudo de viabilidade e com a operacionalidade do sistema; o segundo ciclo com a definição dos requisitos; o próximo com o projeto do sistema e assim por diante.
- Não existem fases fixas.

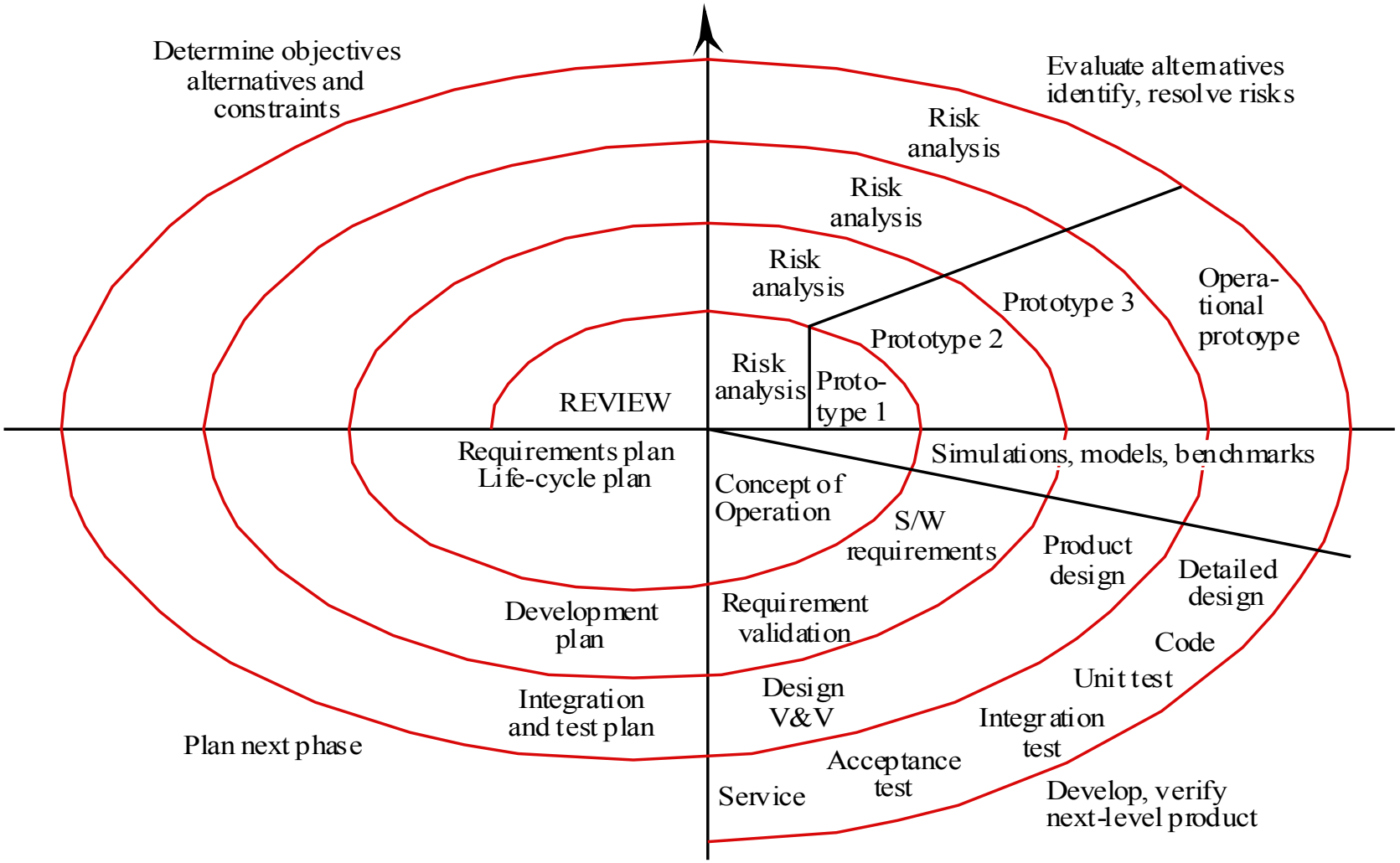
Desenvolvimento Espiral

- Fases do Modelo Espiral:
  - Definição dos objetivos, alternativas e restrições: um plano inicial é esboçado e os riscos do projeto são identificados.
  - Análise de risco: para cada um dos riscos identificados é feita uma análise cuidadosa.

Desenvolvimento Espiral

- Fases do Modelo Espiral:
  - Desenvolvimento e validação: após a avaliação dos riscos, um paradigma de desenvolvimento é escolhido.
  - Planejamento: o projeto é revisado e a decisão de percorrer ou não mais um ciclo na espiral é tomada. Se a decisão for percorrer mais um ciclo, então o próximo passo do desenvolvimento do projeto deve ser planejado.

Desenvolvimento Espiral



Desenvolvimento Formal de Sistemas

- Baseado na transformação de uma especificação matemática através de diferentes representações para um programa executável.
- Consegue alcançar os requisitos da especificação mais facilmente.

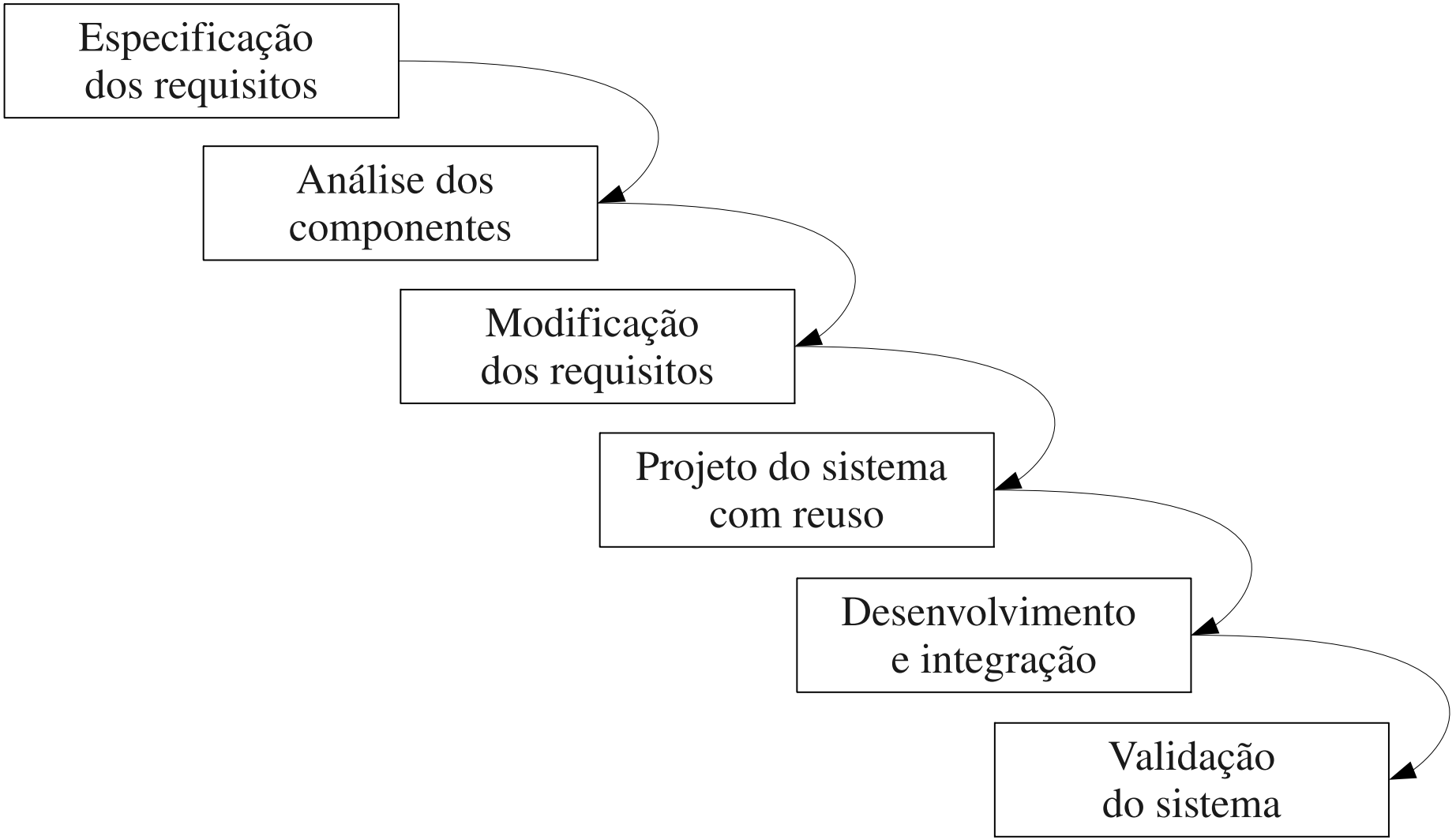
Desenvolvimento Formal de Sistemas

- Problemas:
  - Dificuldade em encontrar profissionais especializados.
  - Dificuldade em especificar determinados aspectos do sistemas, como a interface do usuário.
- Aplicabilidade:
  - Principalmente para sistemas críticos onde não são toleradas falhas.

Desenvolvimento Baseado em Reuso

- Sistemas baseados em componentes já existentes. Semelhantes ao desenvolvimento de hardware.
- Fases do processo:
  - Análise do componente.
  - Modificação dos requisitos.
  - Projeto do sistema com reuso.
  - Desenvolvimento e integração.
- Método que vem crescendo bastante nos últimos tempos.

Desenvolvimento Baseado em Reuso



Metodologias de Desenvolvimento de Software

Surgimento

- A crise de software da década de 70 foi um dos principais motivos para o criação de metodologias para desenvolvimento de software.
- Surgem diversas metodologias:
  - Análise estruturada.
  - Análise essencial.
  - Análise orientada a objetos.
- A análise auxilia na comunicação entre as pessoas envolvidas no processo, no gerenciamento da complexidade e na redução dos custos de desenvolvimento.

Características

Técnica	Enfoque	Abordagem
Análise estruturada	Processos e dados	Top-down (decomposição)
Análise essencial	Controle, processos e dados	Middle-out (lista de eventos)
Análise orientada a objetos	Dados, controle e processos	Definição de objetos

Análise estruturada

- Prioriza as funções e processos.
- Utiliza as seguintes ferramentas:
  - Diagrama de fluxo de dados (DFD).
  - Dicionário de dados (DD).
  - Especificação da lógica dos processos.
- A análise estruturada clássica não modela o comportamento temporal, nem complexos relacionamentos de dados.

Análise essencial

- É uma evolução da análise estruturada por adicionar a preocupação com o controle.
- Usa uma lista de eventos externos como base para o particionamento do sistema.
- O modelo essencial é construído sem considerar restrições de implementação – essência do sistema.

Análise essencial

- Esse modelo é formado por:
  - Modelo ambiental: define a fronteira entre o sistema e o ambiente.
  - Modelo comportamental: descreve o comportamento interno do sistema.
  - Modelo de informação: modelo os dados necessários às atividades essenciais do sistema.
  - Modelo de implementação: extensão do modelo essencial com restrições de implementação.

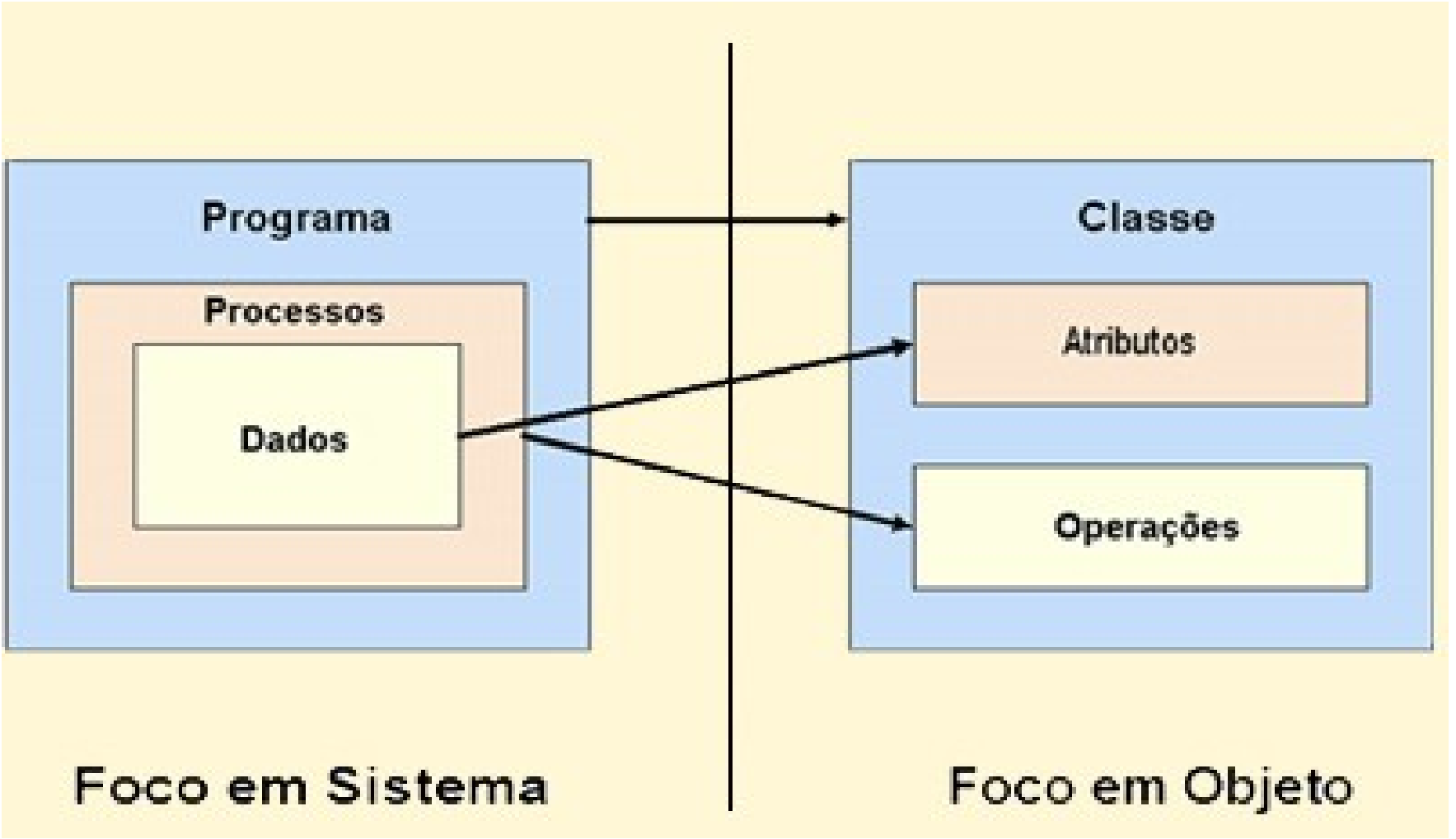
Análise essencial

- Modelo ambiental
  - Diagrama de contexto: define as interfaces entre o sistema e o ambiente. São identificadas informações externas e as produzidas como saídas.
  - Lista de eventos: identifica os eventos que ocorrem no ambiente e como o sistema deve reagir.
- Modelo comportamental
  - Mostra o comportamento interno do sistema.
  - Uso como ferramenta DFD com abordagem diferente.
  - Constrói um DFD para cada evento.

Análise orientada a objetos

- Mudança do enfoque das funções para os dados.
- Preocupação em modelar de forma mais detalhada o sistema.
- Análise mais próxima da realidade.
- Facilidade na comunicação com o usuário:
  - Objetos como entidades do mundo real.
  - Objetos com estrutura e comportamento e que se comunicam.
- Dificuldades em fazer alterações nas estruturas de dados nas abordagens tradicionais.

Comparação



Qualidade de processo e produto de software

O que é qualidade?

- As pessoas, em geral, tem uma noção de qualidade, mas tem dificuldade em definir o conceito.
- Qualidade é um conceito relativo e diversos aspectos devem ser considerados. Por exemplo, como é possível medir a qualidade de um carro? Conforto, segurança, desempenho, beleza e custo representam a qualidade do carro?

O que é qualidade?

- Os dicionários definem qualidade como:
  - Atributo, condição natural, propriedade pela qual algo ou alguém se individualiza, distinguindo-se dos demais; maneira de ser, essência, natureza; grau de perfeição, de conformidade a um certo padrão.

O que é qualidade?

- Qualidade está fortemente relacionada à conformidade com os requisitos.
- Qualidade diz respeito à satisfação do cliente.

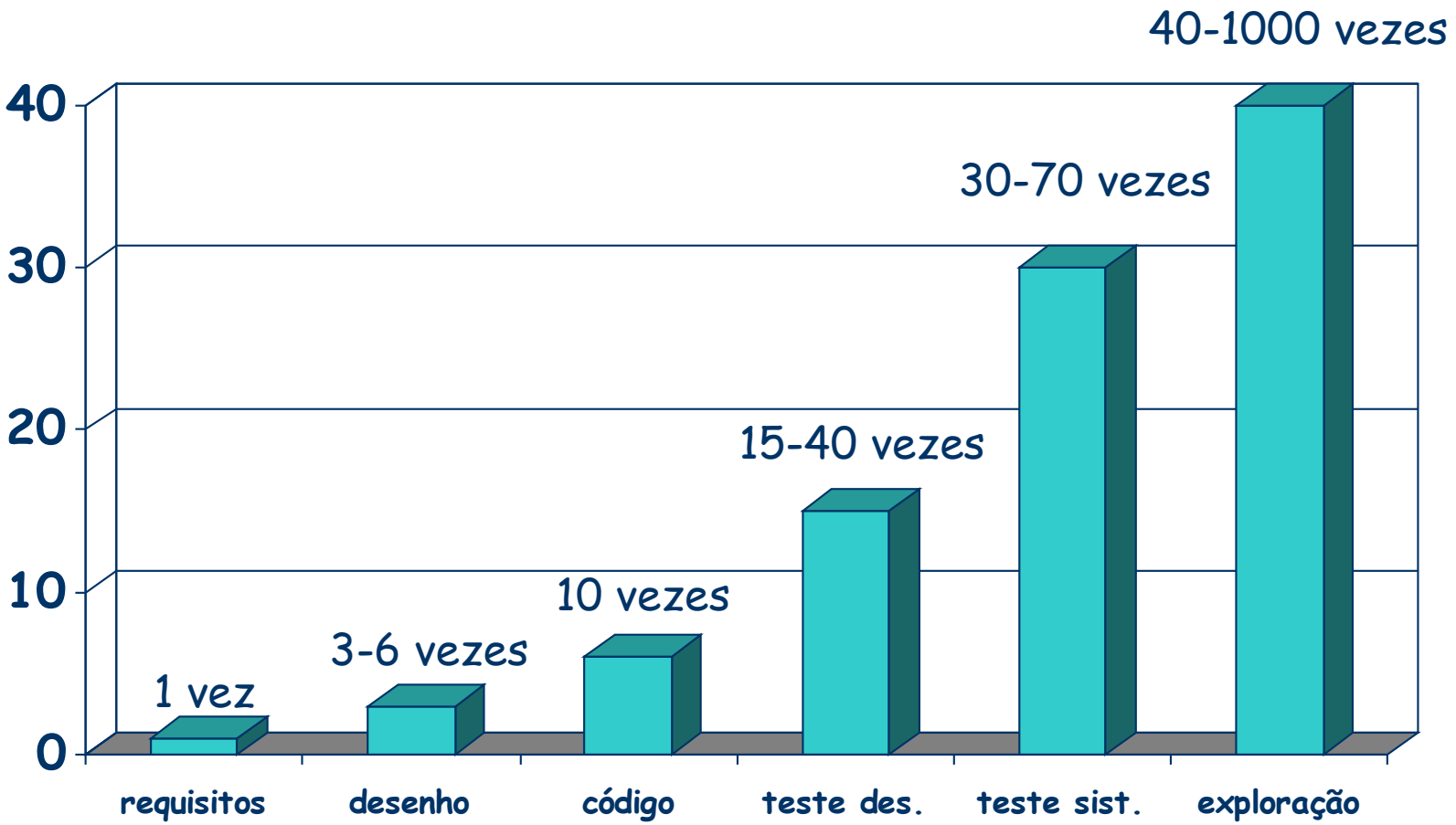
O que é qualidade de software?

- Conjunto de características a serem satisfeitas em um determinado grau, de modo que o software satisfaça às necessidades de seus usuários.
- Para Pressman, um software de qualidade é aquele que está em concordância com os requisitos funcionais e de performance, com padrões de desenvolvimento explicitamente documentados e com as características implícitas em todo software desenvolvido profissionalmente.

Quem participa da qualidade?

- Todos os envolvidos no processo de software:
  - Cliente
  - Analista
  - Programador
  - Gerente
  - Diretor

Custos relativos à correção do software



Qualidade de produto versus qualidade de processo de software

- A qualidade do produto de software não se atinge de forma espontânea.
- A qualidade do produto de software depende da qualidade do processo de software.

O que o processo de software deve estabelecer?

- As atividades a serem realizadas durante o processo, sua estrutura, organização, incluindo a definição de um modelo de ciclo de vida quando pertinente.
- Artefatos requeridos e produzidos por cada uma das atividades do processo.
- Procedimentos a serem adotados na realização das atividades.
- Recursos necessários para a realização das atividades.

Qualidade do processo de software

- Um bom processo de software não garante que os produtos gerados serão de boa qualidade, mas é um indicativo que a organização é capaz de produzir bons produtos.

Motivos para buscar a qualidade do processo de software

- Aumento da qualidade do produto.
- Diminuição do retrabalho.
- Maior produtividade.
- Redução do tempo para atender o mercado.
- Maior competitividade.
- Maior precisão nas estimativas.
- Produtos de software são complexos.
- Software não tem produção em série. Custo está no projeto e desenvolvimento.

Motivos para buscar a qualidade do processo de software

- Software é invisível. Sua representação é feita através de grafos e diagramas.
- A Engenharia de Software ainda não está madura o suficiente.
- Não há um acordo entre os profissionais sobre o que é qualidade de software.
- Uma das principais causas do excessivo tempo gasto (custo) ao desenvolver software é o retrabalho inútil. Cerca de 40 a 50% do esforço de desenvolvimento é retrabalho inútil.

Organizações

- ISO – International Organization for Standardization
- IEEE – Instituto de Engenharia Elétrica e Eletrônica
- ABNT – Associação Brasileira de Normas Técnicas

Padrão de qualidade

- A ISO 9001 define, de maneira genérica, os padrões de qualidade para empresas que projetam, desenvolvem e fazem manutenção de produtos.
- Há necessidade de desenvolver um manual específico para a empresa.
- Existem instituições que certificam se o manual desenvolvido segue os padrões da ISO 9001.

Principais normas nacionais e internacionais na área de software

ISO 9126	Características da qualidade de produtos de software
NBR 13596	Versão brasileira da ISO 9126
ISO 14598	Guias para avaliação de produtos de software, baseados na ISO 9126
ISO 12119	Características de qualidade de pacotes de software (software de prateleiras)
ISO 12207	Norma para a qualidade do processo de desenvolvimento de software.
NBR ISO 9001	Modelo para garantia de qualidade em projeto, desenvolvimento, intalação e assistência técnica (processo)
CMM	Modelo da *Software Engineering Institute* (SEI) para avaliação da qualidade do processo de desenvolvimento de software. Não é uma norma ISO, mas é muito bem aceita no mercado.
SPICE (ISO 15504)	Projeto da ISO/IEC para avaliação de processo de desenvolvimento de software. Ainda não é uma norma oficial ISO, mas o processo está em andamento.

Importância dos padrões

- Encapsulam as melhores ou mais adequadas práticas.
- As melhores práticas são adquiridas depois de várias tentativas ou erros.
- O conhecimento dentro de um padrão evitar a repetição de erros.
- Representam valor para a empresa.

Importância dos padrões

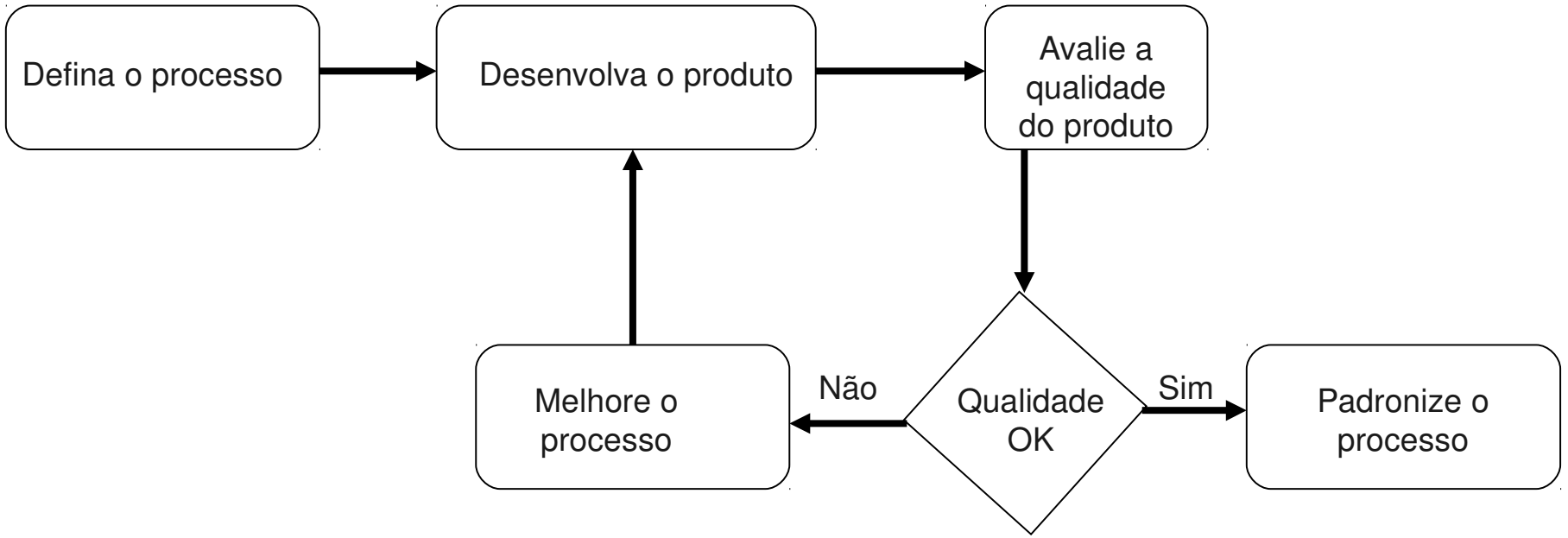
- Fornecem uma estrutura para implantar o processo de garantia de qualidade.
- Facilitam a continuidade de um trabalho.
- Reduzem o esforço de aprendizado em um novo projeto.

Tipos de padrões

- De produto
  - Documentos.
  - Documentação.
  - Codificação.
- De processo
  - Especificação.
  - Projeto.
  - Validação.

Qualidade Processo versus Produto

- A qualidade do processo afeta diretamente a qualidade do produto:



Qualidade do processo versus produto

- Os comentários anteriores valem para produtos manufaturados.
- No desenvolvimento de software, a relação entre qualidade de processos de software e os produtos é mais complexa.
- Melhorando o processo nem sempre conduz à melhoria da qualidade.

Qualidade do processo versus produto

- O software não é manufaturado, mas sim projetado.
- É um processo criativo e não mecânico.
- Depende de habilidades e experiências individuais.
- No entanto, a qualidade do processo tem influência significativa na qualidade do software.

Planejamento da qualidade

- O planejamento da qualidade deve começar antes do desenvolvimento.
- O plano de qualidade deve estabelecer as qualidades do produto.

Planejamento da qualidade

- Segundo Humphrey (1989), o plano de qualidade deve incluir:
  - Introdução sobre o produto (descrição, mercado, distribuição).
  - Plano para o produto (datas, responsabilidades, distribuição).
  - Descrição de processo.
  - Metas de qualidade.
  - Riscos e gerenciamento de riscos.
  - Processo de avaliação da qualidade.
- Não deve ser um documento extenso.

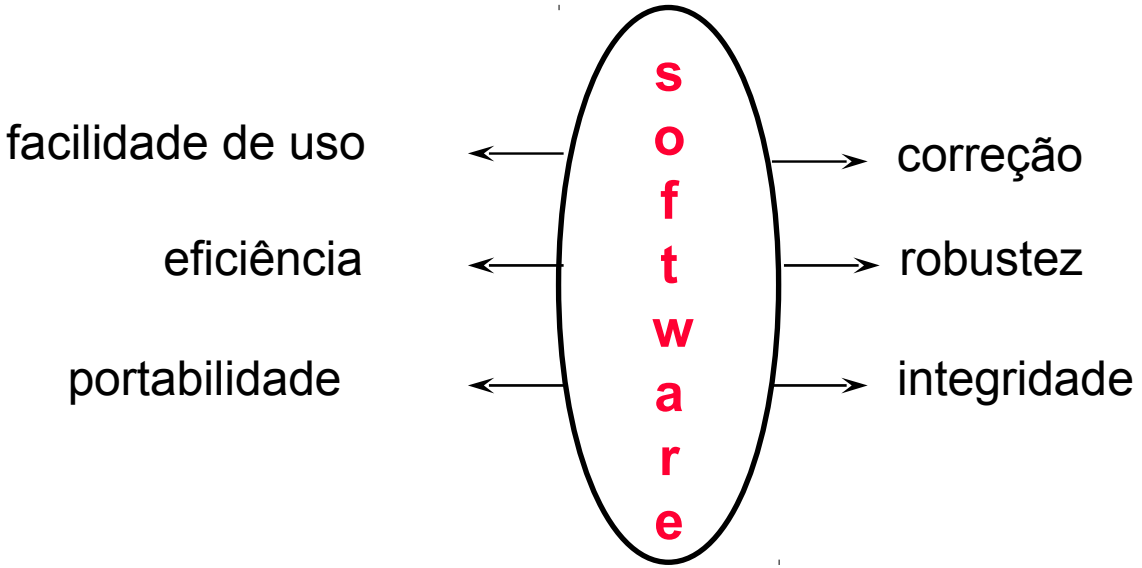
Fatores da qualidade de software

- A noção da qualidade de software pode ser descrita por um grupo de fatores, requisitos ou atributos, tais como: confiabilidade, eficiência, facilidade de uso, modularidade, legibilidade etc.
- Pode-se classificar esses fatores em internos e externos.

Fatores da qualidade de software

- Fatores externos são percebidos tanto pelas pessoas que desenvolvem software como pelos usuários. Por exemplo, confiabilidade, eficiência e facilidade de uso são fatores externos.
- Fatores internos são percebidos apenas pelas pessoas que desenvolvem o software. Por exemplo, modularidade e legibilidade.
- Se os fatores internos forem observados, os fatores externos serão observados.

Fatores da qualidade de software



Revisões de qualidade

- Um grupo de pessoas examina parte ou todo o processo de software, o sistema ou a sua documentação, a fim de descobrir desvios em relação ao plano de qualidade.
- É aconselhável convidar pessoas do desenvolvimento para participar da revisão, para dicas e esclarecimento.
- Cada revisão deve gerar um documento.

Padrões WEB

Padrões WEB

- Existe um padrão para desenvolvimento de software para a WEB.
- O organismo responsável por essa padronização é o W3C e que consiste em um consórcio de várias empresas ligadas à internet e à tecnologia, no ano de 1994. Tim Berners-Lee foi o idealizador do W3C.
- Essa padronização é destinada a desenvolvedores WEB, fabricantes de software e programadores que criem sites.

Início da WEB

- A internet foi criada para a troca de informações.
- Os primeiros navegadores liam apenas textos simples.
- Surgimento da guerra dos navegadores (Netscape versus Microsoft).
- A popularidade da internet e o aumento do volume de dados exigiam a criação de padrões também para a WEB.

Problemas atuais

- Manutenibilidade.
- Existência de outros dispositivos (celulares, smart phones).
- Maior quantidade de navegadores.
- Separação do trabalho de programador e do trabalho de designer.
- O uso de tabelas, por exemplo, é um problema de desinger.

Acessibilidade

- Acessibilidade está relacionada com a possibilidade (impedimento) de alguém fazer alguma coisa.

Referência Bibliográfica

- [1] Carvalho, Ariadne M.B. & Chiossi, Thelma C. dos Santos. ***Introdução à Engenharia de Software***. Editora da Unicamp, 2001.
- [2] Jalote, Pankaj. ***Integrated Approach To Software Engineering***. 2 ed. Springer-Verlag, 1997.
- [3] Pressman, Roger S. ***Software engineering: a practioner's approach***. 5 ed. McGraw Hill, 2001.
- [4] Sommerville, Ian. ***Software Engineering***. 6 ed. Addison Wesley, 2001.