ENGENHARIA DE SOFTWARE
... Síntese...

*"**uma vela não perde sua chama acendendo outra**"*

*"**Apenas 5% dos professores fizeram, fazem e farão a diferença**"*

ENGENHARIA DE SOFTWARE

#Plano de ensino

#Plano de aula

#Metodologia

#Avaliação

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Conceitos

-O que é Software?

-Produtos de software

-O que é Engenharia de Software?

-Qual a diferença entre engenharia de software e Engenharia de sistemas?

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Princípios

1. Entender o que é engenharia de software e porque ela é importante
2. Saber as respostas para questões-chave que fornecem uma introdução à Engenharia de Software
3. Entender questões profissionais e éticas, relevantes para os engenheiros de software
4. Onde os softwares são usados? Qual o impacto do software na sociedade? Quando desenvolver para web ou desktop? Como desenvolver? O que o software produz? O que levar em consideração ao desenvolver um software?

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Processos

-Quais são as razões e critérios para se implantar/utilizar um Processo de Software?

-Importante: -gerenciamento de Projetos; -documentação de Sistemas; -produtividade; -organização do Trabalho

-A evolução dos Processos

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Gestão de Projetos

-O que são Projetos?

-O que são Processos?

-Fatores que afetam a Gerência de Projetos: Absorção de inovações tecnológicas; Especificação do projeto; Metodologias inadequadas; Subestimativa de riscos; Dificuldades de estimar prazos e recursos; Dificuldades de aferir progresso; Fraco acoplamento entre as estratégias empresariais e as de TI; Cultura das organizações; Ambiente típico de desenvolvimento de projetos no Brasil

-PMBOK --- soluções

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Requisitos

-O que são requisitos?

-Qual é a importância dos requisitos?

-Como abstrair os requisitos – técnicas de elicitação

-Requisitos Funcionais e Não Funcionais

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – IHC

-O que é IHC?

-Qual é a importância e os cuidados?

-Alguns Objetivos: -fornecer uma interação pessoa-computador o mais “amigável” possível; -fornecer uma interação clara e consistente facilitando a navegabilidade; -ser fator crítico de apoio ao usuário, não confundindo e gerando insegurança.

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Sistemas Críticos

-Sistema: é um conjunto de componentes inter-relacionados que coletam, que armazenam, que guardam dados para serem processados, manipulados e convertidos em informações. Para que possam ser utilizados como informações para uma tomada de decisão.

1. Propriedades emergentes funcionais: aparecem quando todas as partes de um sistema trabalham juntas para atingir um objetivo.

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Sistemas Críticos

-Propriedades emergentes não funcionais: referem-se ao comportamento do sistema em seu ambiente operacional. Ex.: confiabilidade, desempenho, segurança e proteção. Falhas dessas propriedades podem fazer com que o sistema se torne inutilizável.

-Tipos de Sistemas Críticos: Missão, Negócio e Segurança

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Gerenciamento da Qualidade

-Por que a qualidade de software é tão importante?

-Documentação de qualidade: -Documentar o gerenciamento da qualidade significa registrar o que cada equipe dentro do projeto faz; -Técnica utilizada principalmente no desenvolvimento de sistemas de grande porte e complexos; -Ajuda a verificar se tarefas não foram “esquecidas”.

-Gerenciamento de qualidade de software (SWOT + 5W2H + PDCA + 5S)

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Testes de Software

-Qual é a importância dos Testes de Software?

-As duas atividades fundamentais para testes são:•Testes de componentes (teste de partes do sistema); •Teste de sistema (teste do sistema como um todo).

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Verificação e Validação

-Qual é a importância dos processos de Verificação e Validação?

-Verificação: checar se o software cumpre com suas especificações, ou seja, se o sistema cumpre com seus requisitos funcionais e não funcionais.

-Validação: assegurar que o software atenda às expectativas do cliente, garantindo que ele faça o que o cliente espera que faça.

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Sistemas Embarcados

-É um sistema complexo (microprocessado) no qual o microcomputador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla;

-lógica boolena x lógica difusa

-decimal x hexadecimal

-analógico x digital

-LP compilada x interpretada

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Reuso

-É o processo de incorporar códigos existentes, especificações de requisitos, especificações de projeto e planos de teste.

-Vantagens: 1.Melhores índices de produtividade; 2.Produtos de melhor qualidade; 3.Mais confiáveis, consistentes e padronizados; 4.Redução dos custos e tempo; 5.Maior flexibilidade na estrutura do software;

ENGENHARIA DE SOFTWARE

#Síntese (conteúdos que serão aplicados)

Engenharia de Software – Métricas - conceito e aplicabilidade

-medidas

-prazo

-custo

-tipos de métricas

-proposta comercial

-Métricas Orientadas por Pontos de Função

ENGENHARIA DE SOFTWARE

Fontes:

ROGER S. PRESSMAN. Engenharia de Software. Uma Abordagem Profissional. 7ª Edição.

SOMMERVILLE, IAN. ENGENHARIA DE SOFTWARE. PEARSON BRASIL. 9ª Edição.

Engenharia de Software

Conceitos

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Conceitos

O que é Software?

1. Programas para computadores e equipamentos eletrônicos com documentação associada (requisitos, modelos de projeto e manuais de usuário, etc.)

2. Produtos de software podem ser desenvolvidos para cliente particular ou para mercado geral.

Engenharia de Software
Conceitos

Produtos de software podem ser:

Genéricos – desenvolvidos para qualquer cliente.

Ex.: (Software para PC como Excel ou Word. Especificação é da organização.)

Personalizado (sob encomenda) – desenvolvido para um cliente específico de acordo com as especificações.

Ex.: (Especificação é do cliente - SAP, web services.)

Engenharia de Software
Conceitos

O que é Engenharia de Software?

O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais.”

Engenharia de Software
Conceitos

Engenharia de Software é uma disciplina da engenharia que está concentrada em todos os aspectos da produção de software.

Disciplina que integra processo, métodos e ferramentas para o desenvolvimento de software.

Engenharia de Software
Conceitos

Qual a diferença entre engenharia de software e

Engenharia de sistemas?

Engenharia de sistemas se concentra em todos os aspectos de sistemas baseados em computadores como hardware, software, políticas e processos de engenharia. Engenharia de Software é parte deste processo voltado ao desenvolvimento de software , das aplicações e dos bancos de dados.

Engenharia de Software
Conceitos

Qual a diferença entre engenharia de software e ciência da computação?

1. Ciência da computação se concentra nas teorias e métodos fundamentais da computação;

2. Eng. Software se concentra nos problemas práticos da produção de software.

3. Ideal é que engenheiros de software se apoiassem nas teorias da computação, mas na realidade isso é complicado.

Engenharia de Software
Conceitos

Qual é a importância do Engenheiro de Software?

Engenheiros de software são envolvidos na especificação de sistemas, projeto arquitetural, integração e distribuição.

Engenharia de Software
Conceitos

Engenheiros de Software devem adotar uma abordagem sistemática e organizada para trabalhar e usar ferramentas e técnicas apropriadas dependendo de cada problema a ser solucionado (considerar plataforma, SO, escopo, imediatismo, etc).

Engenharia de Software
Conceitos

O que é um processo de software?

Conjunto de atividades com o objetivo de desenvolver ou evoluir um software.

Engenharia de Software
Conceitos

As atividades de todos os processos de software são:

Especificação – o que o sistema deverá fazer e quais as restrições de desenvolvimento

Desenvolvimento – produção do sistema de software, projetado e programado

Engenharia de Software
Conceitos

As atividades de todos os processos de software são:

Validação – verifica se o software é o que o cliente deseja

Evolução – mudanças do software em função da demanda, que podem ser requisitos dos clientes ou do mercado

Engenharia de Software
Conceitos

O que é um modelo de processo de software?

Uma representação simplificada do processo de software que apresenta a visão do mesmo. Exemplos de perspectivas/modelos de processos são:

1. Modelo de Workflow – sequência de atividades;
2. Modelo de fluxo de dados – fluxo da informação, como entrada é transformada numa saída;
3. Modelo de papel/ação – quem faz o que.

Engenharia de Software
Conceitos

Quais são os custos da engenharia de software?

Aproximadamente 60% do custo é para desenvolvimento, 40% para testes, mas isso pode variar.

Para softwares personalizados os custos com evolução podem exceder os custos de desenvolvimento e são difíceis de planejar.

Engenharia de Software
Conceitos

Quais são os custos da engenharia de software?

Custos variam e dependem do processo e do tipo de sistema a ser desenvolvido, além dos atributos (imediatismo, performance, plataforma, segurança, disponibilidade, etc).

Distribuição do custo depende do modelo que está sendo usado.

Engenharia de Software
Conceitos

O que são métodos de engenharia de software?

Abordagem estruturada para desenvolvimento do software que incluem modelos de sistemas, notações, regras, guias de processos, recomendações, etc.

Descrições de Modelos de sistemas: modelos gráficos que devem ser produzidos. Ex.: Modelo de objetos, fluxo de dados, máquina de estado;

Engenharia de Software
Conceitos

O que são métodos de engenharia de software?

Regras: Restrições aplicadas aos modelos. Ex.: toda entidade deve ter um único nome;

Recomendações: Heurísticas que caracterizam boa prática de projeto nesse método. Ex: nenhum objeto deve ter mais que sete sub-objetos;

Guia de processo: Quais atividades devem ser seguidas. Ex.: atributos de objetos devem ser documentados antes de definir as operações do mesmo.

Engenharia de Software
Conceitos

Quais são os atributos de um bom software?

O software deve dispor das funcionalidades requeridas, além de performance para o usuário e fácil manutenção, usabilidade e confiança.

Fácil manutenção: Software deve ser escrito de modo que possa evoluir de acordo com as mudanças necessárias;

Engenharia de Software
Conceitos

Quais são os atributos de um bom software?

Confiança: Software deve ter trustworthy (confiança). Não deve causar danos físicos ou econômicos em caso de falhas (COMPLICADO);

Eficiência: Software não deve desperdiçar os recursos do sistema;

Usabilidade: Software deve ser usável, sem esforço excessivo, pelo tipo de usuário para o qual foi projetado

Engenharia de Software
Conceitos

Quais são os desafios-chave da engenharia de software?

Heterogeneidade, entrega e confiança.

H: Desenvolver técnicas para operar em sistemas distribuídos em plataformas e ambientes de execução diferentes;

E: Desenvolver técnicas para diminuir os tempos de entrega dos sistemas grandes e complexos sem comprometer a qualidade;

C: Desenvolver técnicas que demonstrem que o software pode ter a confiança dos usuários.

Engenharia de Software
Conceitos

1. Pesquise alguns, pelo menos 3, problemas causados por erros em software.
2. Pesquise, pelo menos, 3 produtos de categorias diferentes onde o software faz a diferença.
3. Qual a diferença de um software livre para open source?
4. O que é GNU - GPL?
5. Que tipos de licenças de software existem?
6. O software livre impacta na sociedade de que forma?
7. Software livre resolve a pirataria? Explique.
8. Como combater a pirataria?

Fonte

Fonte (texto adaptado) –

PRESSMAN, Roger S.. Engenharia de Software. Uma Abordagem Profissional. 7ª Edição.

SOMMERVILLE, Ian. Engenharia de Software. 9ª Edição.

Engenharia de Software

ESTUDAR ...

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software

Responsabilidade Profissional e Ética
Princípios da Engenharia de Software

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Objetivo

1. Entender o que é engenharia de software e porque ela é importante
2. Saber as respostas para questões-chave que fornecem uma introdução à Engenharia de Software
3. Entender questões profissionais e éticas, relevantes para os engenheiros de software

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Objetivo:

1. Onde os softwares são usados?
2. Qual o impacto do software na sociedade?
3. Quando desenvolver para mobile, web ou desktop?
4. Como desenvolver?
5. O que o software produz?
6. O que levar em consideração ao desenvolver um software?

(Layout, Padronização, LP, BD, Redes, Segurança, Qualidade, Público, Plataforma, SO, Multiusuário, Pensar no hoje e no amanhã)

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Para refletir....

Os tópicos apresentados na aula anterior...

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Evolução do software:

Nos primeiros anos (1950):

- Software desenvolvido sem administração e métodos
- Software projetado sob medida para a aplicação
- Quase não havia "produto de software"
- Quem desenvolvia, usava e alterava o software
- O projeto ficava na cabeça do desenvolvedor
- Não havia documentação
- O que acontecia se um desenvolvedor pedisse demissão?

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Evolução do software:

Segunda fase (meados dos anos 60):

- Interatividade, sistemas de tempo real
- Uso de diferentes bibliotecas
- Necessidade de adaptar (ao cliente ou ao hardware) e manter milhares de linhas de código
- Surgem as atividades de manutenção de software
- A manutenção passou a absorver muitos recursos (tempo e R$)

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Evolução do software:

Terceira fase (anos 70):

- Redes, distribuição e concorrência
- Generalização dos computadores pessoais
- Crescimento das empresas de software
- As empresas passaram a vender até 100 vezes mais produtos de software
- Muitos passaram a gastar mais dinheiro com software do que com o computador

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Evolução do software:

Quarta fase (atualmente):

- Tecnologia orientada a objetos
- Redes neurais (IA)
- A capacidade de construir software não acompanha o ritmo da demanda
- Projetos ruins e o uso de recursos inadequados ameaçam a capacidade de manter os softwares existentes

Engenharia de Software

Responsabilidade profissional e ética

Princípios da Engenharia de Software

Desenvolvimento de software:

- Programação por tentativa e erro
- Falta de métodos para o desenvolvimento
- Hoje seu custo é bem maior que o do hardware

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Evolução do hardware:

Desenvolvimento de hardware:

- Era o maior custo
- Uso de padrões técnicos (análise e projeto)
- Emprego de controles, métodos e ferramentas (engenharia de software)
- Portabilidade
- Aplicabilidade

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Lamentações:

• Por que demora tanto tempo para que os programas sejam concluídos?

• Por que os custos são tão elevados?

• Por que não descobrimos todos os erros antes de entregarmos o software ao nosso cliente?

• Por que temos dificuldade em medir o progresso enquanto o software está sendo desenvolvido?

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Conseqüências:

- Aplicações escritas há 20 anos, modificadas ao longo do tempo, são impossíveis de manter.
- Pequenas alterações podem fazer o sistema falhar.

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Conseqüências:

Não há quem conheça estes sistemas

- Falta de documentação do desenvolvimento

Sistemas críticos (tráfego aéreo) de funcionamento "estranho" não são substituídos

- Não há sistemas para substituí-los

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Mitos do software – Gerente:

Temos um manual repleto de padrões e procedimentos que oferecerá tudo o que os desenvolvedores precisam saber:

- Ele é usado?
- As técnicas são adequadas ao software sendo desenvolvido?
- Ele é completo?

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Mitos do software – Gerente:

Temos os computadores mais modernos do mercado:

•Ferramentas (Engenharia de Software) que auxiliam o desenvolvimento são mais úteis

Se o cronograma está atrasado adicionaremos mais programadores para tirar o atraso

•Perda de produtividade na educação dos novos programadores

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Mitos do software – Cliente:

Uma declaração geral dos objetivos é suficiente para começar a escrever o programa.

Os detalhes podem ser preenchidos mais tarde.

-- Isto causa:

- Definição inicial ruim
- Principal causa do fracasso de projetos de desenvolvimento de software

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Mitos do software – Cliente:

Os requisitos modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas:

- Pouca atenção à definição inicial dos requisitos
- O impacto negativo das mudanças é maior conforme o projeto vai se desenvolvendo

Definição (1x)

Implementação (1,5x a 6x)

Manutenção (60x a 100x) Pode ser mais barato desenvolver outro

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Mitos do software – Desenvolvedor:

Depois de escrever e colocar o programa em funcionamento nosso trabalho estará terminado:

- De 50% a 70% do esforço é gasto depois de entregar a primeira versão do sistema ao cliente

Enquanto não tiver o programa funcionando não posso medir sua qualidade:

- A qualidade não deve ser garantida por testes, mas por todas as atividades de desenvolvimento

A única coisa a ser entregue em um projeto bem-sucedido é o programa funcionando:

- A documentação bem feita é crucial para as manutenções futuras

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Avisos!

Não estabeleça prazos audaciosos demais → prazo é prazo

Sempre ouça o mercado

Usuários odeiam bugs

Experiência em simulação ajuda

Nem toda apresentação será um sucesso

O que serve para um cliente pode não servir para outro

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Avisos:

Busque soluções eficientes

SUPORTE! Ferramentas diferentes para situações diferentes

Ajuda On-Line pode ser útil

Previsão e otimização podem ser complexas

Observe os atributos significativos do seu cliente

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Por quê?

Sintomas:

- Compreensão incompleta ou imprecisa das necessidades do usuário
- Inabilidade de lidar com requisitos que evoluem
- Módulos incompatíveis
- Dificuldades de estender ou manter software
- Descoberta de defeitos graves no projeto em etapas avançadas de desenvolvimento ou mesmo em época de implantação ou uso
- Desempenho inaceitável do software
- Falta de coordenação na equipe

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Por quê?

Causas frequentes:

- Gerência de requisitos sem processo definido
- Comunicação ambígua e imprecisa entre partes envolvidas
- Complexidade crescente
- Inconsistências não detectadas em nível de análise, projeto e implementações
- Testes insuficientes

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Por quê?

Causas frequentes:

- Dificuldade em lidar e gerenciar riscos
- Falta de controle sobre propagação de mudanças
- Automação insuficiente
- Ubiqüidade (disponível o tempo todo em qualquer lugar)
- Diversidade de plataformas
- Comunicação entre o cliente e o desenvolvedor é muito fraca.

Engenharia de Software

Responsabilidade profissional e ética
Princípios da Engenharia de Software

Mas nem tudo está perdido:

- Sistemas complexos e grandes foram, e estão sendo, desenvolvidos.
- Simuladores de aeronaves, veículos
- Telemetria, processamento em tempo real
- Geoprocessamento
- Construção de plataformas, edificações
- etc, etc, etc

Fonte

Fonte (texto adaptado) –

PRESSMAN, Roger S.. Engenharia de Software. Uma Abordagem Profissional. 7ª Edição.

SOMMERVILLE, Ian. Engenharia de Software. 9ª Edição.

Engenharia de Software

ESTUDAR ...

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software

Gerência de Projetos e Análise de Processos I

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Gerência de Projetos e Análise de Processos

O que são Projetos?

É um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. Os projetos e as operações diferem, principalmente, no fato de que os projetos são temporários e exclusivos, enquanto as operações são contínuas e repetitivas.

Os projetos são normalmente autorizados como resultado de uma ou mais considerações estratégicas. Estas podem ser uma demanda de mercado, necessidade organizacional, solicitação de um cliente, avanço tecnológico ou requisito legal.

(Fonte: PMBOK – PMI = Project Management Institute)

Engenharia de Software
Gerência de Projetos e Análise de Processos

O que são Processos?

Consiste em um conjunto de atividades relacionadas e seus relacionamentos, bem como os critérios para indicar o início e término do mesmo.

E a obtenção de informações sobre as atividades individuais, como funções, sistemas, formulários, relatórios e outros.

(Fonte: PMBOK – PMI = Project Management Institute)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Métodos – proporcionam detalhes de “como fazer” para construir o software. Os “métodos” envolvem um amplo conjunto de tarefas que incluem: planejamento e estimativa de projeto, análise de requisitos de software e de sistemas, projeto da estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste e manutenção.

Engenharia de Software
Gerência de Projetos e Análise de Processos

Ferramentas – proporcionam apoio automatizado ou semi-automatizado aos métodos. Atualmente, existem ferramentas para sustentar cada um dos métodos. Quando as ferramentas são integradas de forma que a informação criada por uma ferramenta possa ser usada por outra, é estabelecido um sistema de “suporte” ao desenvolvimento de software.

Engenharia de Software
Gerência de Projetos e Análise de Processos

Procedimentos – constituem o elo de ligação que mantém juntos os métodos e as ferramentas e possibilita o desenvolvimento racional e oportuno do software de computador. Os procedimentos definem a sequencia em que os métodos serão aplicados, “os produtos” que se exige que sejam entregues.

Engenharia de Software
Gerência de Projetos e Análise de Processos

Métricas – São processos utilizados para desenvolver um projeto, bem como, "melhorar" a qualidade deste mesmo projeto.

A evolução das métricas:

Métrica de Halstead
Métrica de McCabe
Métricas de Yin e Winchester
Métricas no Ciclo de Vida
Métricas de Henry e Kafura
Métrica Orientada à Tamanho
Métricas Orientadas à Função
Métricas por contagem de Pontos de Função
Métricas por Casos de Uso

Engenharia de Software
Gerência de Projetos e Análise de Processos

Fonte

Fonte (texto adaptado) –

PRESSMAN, Roger S.. Engenharia de Software. Uma Abordagem Profissional. 7ª Edição.

SOMMERVILLE, Ian. Engenharia de Software. 9ª Edição.

Engenharia de Software
Gerência de Projetos e Análise de Processos

ESTUDAR ...

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software

Gerência de Projetos e Análise de Processos II

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Gerência de Projetos e Análise de Processos

Fatores que afetam a Gerência de Projetos

"o fracasso na implementação de projetos de TI está associado aos seguintes fatores"

1. Absorção de inovações tecnológicas
2. Especificação do projeto
3. Metodologias inadequadas
4. Subestimativa de riscos
5. Dificuldades de estimar prazos e recursos

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Fatores que afetam a Gerência de Projetos

"o fracasso na implementação de projetos de TI está associado aos seguintes fatores"

6. Dificuldades de aferir progresso

7. Fraco acoplamento entre as estratégias empresariais e as de TI

8. Cultura das organizações

9. Ambiente típico de desenvolvimento de projetos no Brasil

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Absorção de Inovações Tecnológicas

A prática têm demonstrado que projetos de tecnologia da informação que incorporam novas tecnologias e inovações ainda não dominadas pela equipe técnica do projeto e nem disseminadas pela organização, não atingem seus objetivos.

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Especificação do Projeto

Qualquer projeto de TI, seja de desenvolvimento de sistemas, de uma rede de teleprocessamento, ou de automação, exige uma necessidade muito acentuada de "interação entre indivíduos", muitas vezes divergentes.

O ato de especificar um projeto, seus objetivos, sua concepção, "requer a agregação" de conhecimentos multidisciplinares, num processo evolutivo, até que a solução para o problema seja satisfatória.

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Metodologias Inadequadas

A maioria dos ambientes de desenvolvimento de projetos de TI não possuem um roteiro (metodologia) definido. Este procedimento tem profundos impactos nos custos, prazos e qualidade de um projeto.

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Subestimativa de Riscos

É muito comum sermos otimistas em relação aos projetos de TI, principalmente no seu início. Esquecemos as catástrofes passadas e planejamos “sem considerar contingências”.

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Dificuldades de Estimar Prazos e Recursos

“As técnicas” utilizadas para estimar prazos são “fracas e ineficientes”. As estimativas tendem a aquilatar o esforço desenvolvido e não o trabalho feito ou o real avanço do projeto.

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Dificuldades de Aferir Progresso

Em projetos de TI, pode-se traduzir “temo transcorrido em custos realizados”, mas não em progresso realizado.

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Fraco Acoplamento entre as Estratégias Empresariais e as de TI

Em organizações em que a TI não é estratégica para a conquista dos objetivos, geralmente não são encontrados os “meios e condições” adequados para o bom desempenho do trabalho de gerência de projetos.

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Cultura das Organizações

Este talvez seja o fator “mais relevante”. Quanto mais conscientes e participativos forem a alta administração, os gerentes e os usuários em geral, nos processos de uso da TI, mais facilitada ficará a tarefa de gerenciar projetos.

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Ambiente Típico de Desenvolvimento de Projetos no Brasil

A ausência de uma infra-estrutura de apoio à condução de projetos de TI na maioria das empresas no Brasil faz com que os gerentes de projetos sejam verdadeiros "malabaristas", e esta ausência é uma fonte inesgotável de argumentos para que se faça um planejamento "exequível" e realista.

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Necessidade de formalização do processo de Gerência de Projetos de TI

Definição de responsabilidades pelos agentes envolvidos no projeto, visando estruturar um ambiente de responsabilidades compartilhadas.

Definição de estratégias de condução do projeto.

-Planejamento do projeto

-Administração do projeto

(Fonte: Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas)

Engenharia de Software
Gerência de Projetos e Análise de Processos

Gerenciamento de Projetos
Gerenciamento da Integração
Gerenciamento de Escopo
Gerenciamento de Custos
Gerenciamento de Qualidade
Gerenciamento das Aquisições
Gerenciamento de Recursos Humanos
Gerenciamento das Comunicações
Gerenciamento de Risco
Gerenciamento de Tempo
Gerenciamento das Partes Interessadas

Fonte (Texto adaptado): PMBOK 5ª edição

Engenharia de Software
Gerência de Projetos e Análise de Processos

Fonte

Agnaldo Aragon Fernandes – Gerência de Projetos de Sistemas

Fonte (texto adaptado) –

PRESSMAN, Roger S.. Engenharia de Software. Uma Abordagem Profissional. 7ª Edição.

SOMMERVILLE, Ian. Engenharia de Software. 9ª Edição.

Engenharia de Software
Gerência de Projetos e Análise de Processos

ESTUDAR ...

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software

PROCESSOS DE SOFTWARE

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Processos de Software

Quais são as razões e critérios para se implantar/utilizar um Processo de Software?

Engenharia de Software
Processos de Software

As razões são:

-a padronização

-a facilidade no treinamento

-a redução da manutenção

-a estipulação de prazos e custos reais

-a eliminação de possíveis erros

-a documentação automática

Engenharia de Software
Processos de Software

Os critérios são:

-análises diversas

-analisar .. tempo x adaptabilidade

-a facilidade no manuseio

-o volume

-verificar as áreas envolvidas

Engenharia de Software
Processos de Software

Importante:

-gerenciamento de Projetos

-documentação de Sistemas

-produtividade

-organização do Trabalho

Engenharia de Software
Processos de Software

Qual é a diferença entre Método e Metodologia

Segundo o dicionário Aurélio...

Método é o caminho pelo qual se atinge um objetivo

Metodologia é o estudo dos métodos e, especialmente, dos métodos das ciências

Engenharia de Software
Processos de Software

E na Tecnologia da Informação? Como é a definição dos mesmos?

De uma forma ampla (global):

Metodologias são os procedimentos para a construção do software.

Métodos especificam a técnica para a elaboração destes procedimentos sendo, possível que determinados métodos sejam utilizados por diferentes metodologias e vice-versa.

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Linear

Engenharia de Software
Processos de Software

A evolução dos Processos: Análise Convencional

Alguns Fatores Críticos:

-falta de Ferramentas adequadas

-falta de Técnicas para abstrair informações dos usuários

-custo de retrabalho

-problemas de comunicação

Engenharia de Software
Processos de Software

A evolução dos Processos: Prototipagem

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Espiral

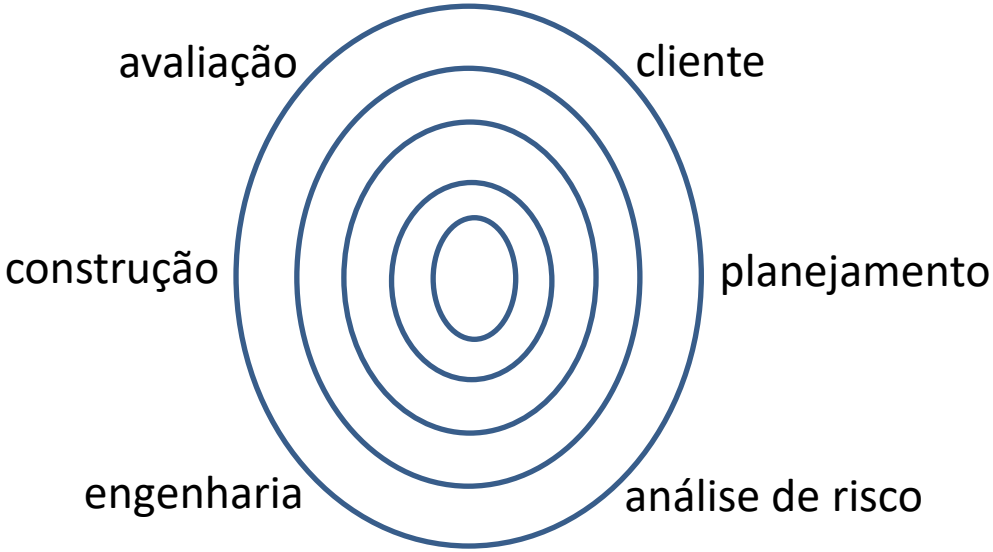
Técnicas emergentes deste modelo:

-desenvolvimento em sequencia

-aumento da complexidade

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Espiral



Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Análise Estruturada

Fatores Críticos de Sucesso:

-refinamentos sucessivos

-ênfase nos processos

-visão global do problema

-voltado para grandes sistemas

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Análise Estruturada

Fatores Críticos de Sucesso:

-D.F.D.
-Diagrama Hierárquico
-Diagrama de Estrutura de Módulos
-Dicionário de Dados
-Português Estruturado
-Abordagem ‘Top-Down’

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Engenharia da Informação

Fatores importantes:

-D.E.R.

-facilidade da transação lógica para física

-visão geral da organização

-ênfase dos dados

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Análise Essencial

-evolução da Análise Estruturada

-abordagem de Eventos

-visão bottom-up

-fase de especificação

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Análise Essencial

-essência do problema

-ênfase nos processos

-D.F.D + D.E.R.

-modelo Estruturado para Orientado a Objetos

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Análise Essencial

Fatores como -- Levantamento de Eventos:

-estímulos

-saídas

-normais e temporais

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Análise Essencial

Fatores como – Refinamentos Sucessivos:

-variação significativa

-oposto

-negação

-sucessor

-precedente

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Orientada a Objetos

-conceitos de Herança

-classes de objetos

-instanciação

-polimorfismo

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo Orientada a Objetos

-resistentes a manutenção

-reutilização do código

-encapsulamento

-GUI = Interface Gráfica

Engenharia de Software
Processos de Software

A evolução dos Processos: ÁGIL

Conceito: (do inglês *Agile software development*) ou Método ágil é um conjunto de metodologias de desenvolvimento de *software*. O desenvolvimento ágil, tal como qualquer metodologia de *software*, providencia uma estrutura conceitual para reger projetos de engenharia de software.

Engenharia de Software
Processos de Software

A evolução dos Processos: Desenvolvimento Ágil de Software

-planejamento

-análise de requisitos

-projetos

-codificação

-testes

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo RAD

RAD – Desenvolvimento Rápido de Aplicativos

-prazo entre 60 e 90 dias

-similar ao Linear (alta velocidade)

-baseado em Componentes

Engenharia de Software
Processos de Software

A evolução dos Processos: Modelo RAD

Fases da Modelagem:

-negócio

-dados

-processo

-geração de aplicação

-testes

Engenharia de Software
Processos de Software

A evolução dos Processos: RUP

Conceito: abreviação de Rational Unified Process (ou Processo Unificado Rational) é um processo proprietário de Engenharia de software criado pela Rational Software Corporation, adquirida pela IBM, ganhando um novo nome IRUP que agora é uma abreviação de IBM Rational Unified Process e tornando-se uma *brand* (marca) na área de Software;

Engenharia de Software
Processos de Software

A evolução dos Processos: SCRUM

Conceito: O Scrum é um processo de desenvolvimento iterativo e incremental para gerenciamento de projetos e desenvolvimento de software.

Engenharia de Software
Processos de Software

A evolução dos Processos: SCRUM

Algumas características de Scrum:

Clientes se tornam parte da equipe de desenvolvimento;
Entregas frequentes e intermediárias de funcionalidades 100% desenvolvidas;
Planos frequentes para redução de riscos desenvolvidos pela equipe;

Engenharia de Software
Processos de Software

A evolução dos Processos: SCRUM

<u>Algumas características de Scrum:</u>

Discussões diárias de status com a equipe;
A discussão diária na qual cada membro da equipe responde às seguintes perguntas:
O que fiz desde ontem?
O que estou planejando fazer até amanhã?
Existe algo me impedindo de atingir minha meta?
Transparência no planejamento e desenvolvimento;

Engenharia de Software
Processos de Software

A evolução dos Processos: SCRUM

<u>Algumas características de Scrum:</u>

Reuniões frequentes com os *stakeholders* (todos os envolvidos no processo) para monitorar o progresso;

Problemas não são ignorados e ninguém é penalizado por reconhecer ou descrever qualquer problema não visto;

Locais e horas de trabalho devem ser energizadas, no sentido de que "trabalhar horas extras" não necessariamente significa "produzir mais".

Engenharia de Software
Processos de Software

A evolução dos Processos: SCRUM

Algumas características de Scrum:

Papéis principais:

Scrum é um esqueleto de processos que contém grupos de práticas e papéis pré-definidos. Os principais papéis são:

o **ScrumMaster**, que mantém os processos (normalmente no lugar de um gerente de projeto);
o **Proprietário do Produto**, ou **Product Owner**, que representa os *stakeholders* e o negócio;
a **Equipe**, ou **Team de Desenvolvimento**, um grupo multifuncional entre 3 a 9 pessoas e que fazem a análise, projeto, implementação, teste etc.

Engenharia de Software
Processos de Software

A evolução dos Processos: Extreme Programming

Conceito: Desenvolvimento simultaneo de aplicativos

Princípios: Comunicação, Simplicidade, Feedback, Coragem

Práticas: Planejamento, entregas frequentes, metáfora, projeto simples, testes, programação em pares, refatoração, propriedade coletiva, integração contínua, 40h de trabalho semanal, cliente presente, código padrão.

Fonte

Fonte (texto adaptado) –

PRESSMAN, Roger S.. Engenharia de Software. Uma Abordagem Profissional. 7ª Edição.

SOMMERVILLE, Ian. Engenharia de Software. 9ª Edição.

Engenharia de Software

ESTUDAR ...

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software

REQUISITOS

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Software:
- São programas para computadores e equipamentos eletrônicos com documentação associada (requisitos, modelos de projeto e manuais de usuário, etc.).
- É o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais.
- Eles refletem as necessidades dos clientes de um sistema que ajuda a resolver algum problema.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Software:
- Os requisitos de um sistema são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais.
- O termo requisito pode ser usado com variações que vão desde uma simples declaração de um serviço que o sistema deve fornecer, ou uma restrição do sistema, até a uma definição formal e detalhada de uma função do sistema.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Requisitos funcionais: correspondem à listagem de tudo o que o sistema deve fazer. São declarações das funções que o sistema deve fornecer, como ele deve reagir a entradas específicas e como deve se comportar em determinadas situações. Podem, inclusive, declararem o que o sistema NÃO deve fazer.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Requisitos do usuário: correspondem aos requisitos abstratos de alto nível. São declarações escritas em linguagem natural, ou representadas em diagramas, sobre as funções e restrições do sistema.
- Requisitos de sistema: indicam uma descrição detalhada do que o sistema deverá fazer, assim como das restrições. O documento de requisitos pode ser chamado de especificação funcional.
- Requisitos de domínio: são requisitos que se originam do domínio de aplicação do sistema e que refletem características desse domínio. Podem ser funcionais ou não-funcionais.
- Requisitos evidentes: são efetuados com conhecimento do usuário. Corresponderão aos eventos e respostas do sistema (troca de informações).
- Requisitos ocultos: são efetuados pelo sistema sem o conhecimento explícito do usuário.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Requisitos não-funcionais: são restrições colocadas sobre como o sistema deve realizar seus requisitos funcionais. Por exemplo: regras de negócio, restrições de tempo, restrições sobre o processo de desenvolvimento, etc.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Requisitos externos: abrange todos os requisitos procedentes de fatores externos ao sistema e a seu processo de desenvolvimento. Ex.: requisitos de interoperabilidade, requisitos legais, requisitos éticos (garantir que o sistema será aceitável para seus usuários e o público em geral).
- Requisitos de produtos: especificam o comportamento do produto. Ex.: requisitos de desempenho (rapidez com que o sistema deve operar, memória requerida), requisitos de confiabilidade (taxa aceitável de falhas), requisitos de portabilidade, requisitos de facilidade de uso.
- Requisitos organizacionais: procedentes de políticas e procedimentos nas organizações do cliente e do desenvolvedor. Ex.: padrões de processo a serem utilizados, os requisitos de implementação (linguagem de programação, método do projeto a ser utilizado), requisitos de fornecimento (quando os produtos e documentos devem ser entregues).

Engenharia de Software

Requisitos de software

Documento de Requisitos:

1. **Introdução**

   - Projeto do documento de requisitos
   - Escopo do produto
   - Definições, acrônimos e abreviações
   - Referências
   - Visão geral do restante do documento

Engenharia de Software

Requisitos de software

Documento de Requisitos:

2. Descrição geral

- Perspectiva do produto
- Funções do produto
- Características do produto
- Restrições gerais
- Suposições e dependências

Referências

SOMMERVILLE, I. **Engenharia de Software.** 6ª ed. São Paulo: Addison Wesley, 2003. 592p.

WAZLAWICK, R. S. **Análise e Projeto de Ciência da Computação Orientados a Objetos.** 3ª tiragem. Rio de Janeiro: Elsevier, 2004.

Engenharia de Software

ESTUDAR ...

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software

Interface Humano Computador

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Interface Humano Computador

O que é IHC?

IU –Interface com o Usuário é uma parte fundamental de um software.

É a parte do sistema visível para o usuário, através da qual, ele se comunica para realizar suas tarefas.

Engenharia de Software
Interface Humano Computador

Qual é a importância?

Pode se tornar uma fonte de motivação e até, dependendo de suas características, uma grande ferramenta para o usuário.

No entanto, se mal projetada, pode se transformar em um ponto decisivo na rejeição de um sistema.

Engenharia de Software
Interface Humano Computador

Alguns Objetivos:

-fornecer uma interação pessoa-computador o mais "amigável" possível.

-fornecer uma interação clara e consistente facilitando a navegabilidade.

-ser fator crítico de apoio ao usuário, não confundindo e gerando insegurança.

Engenharia de Software
Interface Humano Computador

Necessidades:

É um mecanismo por meio do qual se estabelece um diálogo entre o programa e o ser humano.

O usuário e a interface são altamente comuns no tratamento referente às questões humanas ou tecnológicas.

Engenharia de Software
Interface Humano Computador

Observações:

Quem é o usuário?

Quem faz o usuário aprender a interagir com o computador?

Como o usuário interpreta o conjunto de telas do sistema?

Engenharia de Software
Interface Humano Computador

Observações: -Humanas

Tipos de raciocínio;

Percepção visual;

Psicologia cognitiva;

Engenharia de Software
Interface Humano Computador

Observações: -Humanas

Usuário e seu comportamento;

Tarefas obrigatórias do usuário;

Tarefas que o software executa para o usuário;

Engenharia de Software
Interface Humano Computador

Percepção Humana:

Visual: Relatórios, Gráficos, Telas;

Percepções processadas com base em tamanho, forma, cor;

Engenharia de Software
Interface Humano Computador

VERDE	AZUL	LARANJA
ROXO	VERMELHO	PRETO
AMARELO	CINZA	ROSA
LARANJA	VERDE	AZUL
VERMELHO	ROXO	CINZA
ROSA	PRETO	AMARELO

Engenharia de Software
Interface Humano Computador

Confusão no cérebro:

Lado direito ---diz a cor

Lado esquerdo ---diz a palavra

Engenharia de Software
Interface Humano Computador

Percepção Humana –Comunicação Visual

Gráfica mais agradável;

Apresentação em forma textual;

Engenharia de Software
Interface Humano Computador

Qualidade de Interface

-tarefas orientadas ao computador e Humano;

-Nem todos os critérios são relevantes em todo tipo de software;

-Não se deve fazer uma avaliação com novatos;

Engenharia de Software
Interface Humano Computador

Formulário de Julgamento:

Facilidade de Uso: difícil/fácil

Navegação: difícil/fácil

Carga Cognitiva: Confusa/Intuitiva

Engenharia de Software
Interface Humano Computador

Formulário de Julgamento:

Mapeamento: Nenhum/Poderoso

Desenho de Tela: Sim/Não

Compatibilidade: Sim/Não

Engenharia de Software
Interface Humano Computador

Qualidade do Projeto

A NBR 13596 lista um conjunto de características que deve ser verificado em um software para que ele seja considerado um “software de qualidade”.

Engenharia de Software
Interface Humano Computador

Qualidade do Projeto:

-Funcionalidade

-Confiabilidade

-Usabilidade

-Eficiência

-Manuteniblidade

-Portabilidade

Engenharia de Software
Interface Humano Computador

Disponibilidade: em execução, capaz de fornecer serviços úteis a qualquer momento.

Confiabilidade: que o sistema forneça corretamente os serviços, conforme esperado pelo usuário.

Engenharia de Software
Interface Humano Computador

Segurança: que um sistema não cause danos para pessoas ou para o ambiente.

Proteção: que um sistema possa resistir a invasões acidentais ou intencionais.

Engenharia de Software
Interface Humano Computador

Integridade: garantia de que os programas e dados do sistema são sejam danificados.

Confidencialidade: garantia de que as informações possam ser acessadas apenas pelas pessoas autorizadas.

Engenharia de Software
Interface Humano Computador

Correção: garantia de que os serviços do sistema estejam conforme o especificado.

Precisão: garantia de que a informação seja liberada em um nível apropriado de detalhes.

Tempo certo: garantia de que as informações sejam liberadas quando solicitadas.

Engenharia de Software
Interface Humano Computador

Mais algumas características:

Facilidade de reparos;

Facilidade de manutenção;

Capacidade de sobrevivência;

Tolerância a erros;

Engenharia de Software
Interface Humano Computador

Fonte:

Pressman, Roger S. **Uma Abordagem Profissional**. 7ª edição

SOMMERVILLE, I. **Engenharia de Software.** 9ª edição

Engenharia de Software

Sistemas Críticos

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Sistemas Críticos

Sistema: é um conjunto de componentes inter-relacionados que coletam, que armazenam, que guardam dados para serem processados, manipulados e convertidos em informações. Para que possam ser utilizados como informações para uma tomada de decisão.

Engenharia de Software
Sistemas Críticos

Características dos sistemas críticos

1. Possuem propriedades emergentes, ou seja, propriedades do sistema como um todo, que dependem dos componentes do sistema e dos relacionamentos entre eles.

2. São ditos *não determinísticos*. Isso significa que, para uma determinada entrada, a saída produzida pode não ser sempre a mesma. Seu comportamento pode ser alterado pela criação de novos relacionamentos entre seus componentes.

3. Os objetivos organizacionais não dependem apenas do sistema, mas também da estabilidade dos objetivos, dos relacionamentos e dos conflitos entre os objetivos, além da interpretação dos objetivos pelas pessoas.

Engenharia de Software
Sistemas Críticos

Propriedades emergentes de Sistemas Críticos

São duas:

1. *Propriedades emergentes funcionais*: aparecem quando todas as partes de um sistema trabalham juntas para atingir um objetivo.

2. *Propriedades emergentes não funcionais*: referem-se ao comportamento do sistema em seu ambiente operacional. Ex.: confiabilidade, desempenho, segurança e proteção. Falhas dessas propriedades podem fazer com que o sistema se torne inutilizável.

São exemplos, ainda, de propriedades emergentes: volume, confiabilidade, proteção (capacidade de resistir a ataques), facilidade de manutenção, usabilidade.

Engenharia de Software
Sistemas Críticos

Influências relacionadas à confiabilidade

São três:

1. *Confiabilidade de hardware*: qual é a probabilidade de falha de um componente de hardware, e qual é o tempo gasto para o reparo dele?

2. *Confiabilidade de software*: qual a probabilidade de um componente de software produzir uma saída errada?

3. *Confiabilidade do operador*: qual é a probabilidade de que o operador cometa um erro?

Engenharia de Software
Sistemas Críticos

Sistemas Legados

São sistemas baseados em computadores, desenvolvidos no passado, por meio de tecnologias antigas e obsoletas.

Muitas vezes é preferível que esse tipo de sistema continue a existir, com mudanças que o adaptem aos novos requisitos e tecnologias, a descartá-los e iniciar o desenvolvimento de um novo sistema.

Engenharia de Software
Sistemas Críticos

São aqueles cujas falhas podem resultar em perdas econômicas significativas, danos físicos ou ameaças à vida humana.

Classificam-se em três tipos:

1. *De segurança*: uma falha pode resultar em prejuízo, perda de vida ou danos sérios ao ambiente. Ex.: sistema de controle de uma fábrica de produto químico.
2. *De missão*: uma falha pode ocasionar problema em atividade dirigida a metas. Ex.: sistema de navegação de nave espacial.
3. *De negócios*: uma falha pode resultar em custos muito altos para o negócio que o utiliza. Ex. sistema controle bancário.

Engenharia de Software
Sistemas Críticos

Confiança: propriedade importante em sistemas críticos

Em sistemas críticos, **confiança** está relacionada à disponibilidade, confiabilidade, segurança e proteção.

1. *Quanto ao uso do sistema*: sistemas não confiáveis, inseguros ou desprotegidos são, com frequência, rejeitados pelos usuários.
2. *Custos de falhas altos:* o custo de falha do sistema é maior do que o custo de um sistema de controle.
3. *Perda de informações*: os dados são muito valiosos, o que faz com que seja justificável o que se gasta de esforço e dinheiro para duplicá-los e impedir que sejam corrompidos.

Engenharia de Software
Sistemas Críticos

Onde os sistemas críticos podem falhar?

1-No hardware: por causa de erros no projeto, falhas de componentes, erros de fabricação, fim da vida útil de componentes.

2-No software: erros na especificação, projeto ou implementação.

3-Nos operadores humanos: são, atualmente, a maior causa de falhas de sistema

Engenharia de Software
Sistemas Críticos

Sistema crítico de segurança simples

São de vários tipos: desde sistemas de controle de dispositivos e máquinas até sistemas de informações e de comércio eletrônico.

Confiança no sistema

Está relacionada ao sistema de computador (hardware), que pode falhar, não fornecendo os serviços solicitados.

Devido à falha, o sistema de software pode não operar conforme o esperado, ou ainda, corromper dados.

O grau de confiança dos usuários em que o sistema irá operar conforme sua expectativa, sem falhas, é expresso sob a forma de *não confiável, muito confiável* e *ultra confiável.*

Engenharia de Software
Sistemas Críticos

Dimensões de confiança

1. *Disponibilidade*: é a probabilidade de que o sistema esteja pronto, em execução, capaz de fornecer serviços úteis a qualquer momento.
2. *Confiabilidade*: é a probabilidade, em um dado período de tempo, de que o sistema forneça corretamente os serviços, conforme esperado pelo usuário.
3. *Segurança*: é um julgamento da probabilidade de que um sistema cause danos para pessoas ou para o ambiente.
4. *Proteção*: é um julgamento da probabilidade de que um sistema possa resistir a invasões acidentais ou intencionais.

Engenharia de Software
Sistemas Críticos

Dimensões de confiança

Além das já citadas, podemos considerar, ainda:

- *Integridade*: garantia de que os programas e dados do sistema são sejam danificados.
- *Confidencialidade*: garantia de que as informações possam ser acessadas apenas pelas pessoas autorizadas.
- *Correção*: garantia de que os serviços do sistema estejam conforme o especificado.
- *Precisão*: garantia de que a informação seja liberada em um nível apropriado de detalhes.
- *Tempo certo*: garantia de que as informações sejam liberadas quando solicitadas.

Engenharia de Software
Sistemas Críticos

Outras propriedades do sistema sob o aspecto da confiança

- *Facilidade de reparos*: as falhas existem, mas podem ser minimizadas se o sistema puder ser reparado rapidamente.
- *Facilidade de manutenção*: refere-se à facilidade em incorporar novos requisitos ao sistema, com baixa probabilidade de que essas mudanças introduzam novos erros.
- *Capacidade de sobrevivência*: refere-se à capacidade do sistema em continuar a fornecer o serviço quando estiver sob ataque e, potencialmente, quando parte do sistema estiver desativada.
- *Tolerância a erros*: propriedade que faz parte da usabilidade e reflete até onde o sistema evita e tolera erros de entrada do usuário.

Engenharia de Software
Sistemas Críticos

Disponibilidade e confiabilidade

- *Disponibilidade de um sistema* é a probabilidade de que o sistema estará pronto e funcionando, para fornecer serviços aos usuários quando eles solicitarem.
- *Confiabilidade de um sistema*: é a probabilidade de que os serviços serão fornecidos corretamente, conforme especificado. É a probabilidade de operação livre de falhas durante um período especificado, em um dado ambiente, para um objetivo específico.

Essas duas propriedades são comprometidas pelas falhas de sistema.

Engenharia de Software
Sistemas Críticos

Segurança

Sistemas de segurança críticos são aqueles em que é essencial que a operação seja sempre segura, isto é, que nunca cause danos a pessoas ou ao ambiente, mesmo que ocorra falha no sistema. Ex. sistema de controle de aeronaves.

Confiabilidade e *segurança* são propriedades relacionadas. Porém, são atributos separados de *confiança*.

Engenharia de Software
Sistemas Críticos

Por que os sistemas confiáveis de software não são, necessariamente, seguros?

1. Especificação pode estar incompleta e, assim, não descreve o comportamento exigido do sistema em situações críticas (*dificuldades com requisitos*).
2. O mau funcionamento de software pode fazer o sistema se comportar de maneira imprevisível, de forma que o software fique exposto a uma situação inesperada.
3. Os operadores podem falhar ao gerarem entradas incorretas individualmente, mas que em certas situações, podem levar a um mau funcionamento do sistema.

Engenharia de Software
Sistemas Críticos

Garantia de segurança

O essencial para se garantir a segurança, é garantir que o acidente não ocorra, ou que as consequências sejam mínimas. De que forma?

1. *Prevenção de perigos*: o sistema deve ser projetado de forma que os perigos sejam evitados.

2. *Detecção e remoção dos* perigos: os perigos são detectados e removidos antes de causarem um acidente.

3. *Limitação de danos*: trata-se de recursos de proteção que o sistema possui para minimizar os danos resultantes de um acidente.

Engenharia de Software
Sistemas Críticos

Proteção

Reflete a capacidade do sistema de se proteger de ataques externos acidentais ou propositais.

Três tipos de danos podem ser causados por ataques externos:

1. *Recusa de serviço*: o sistema é forçado a um estado em que seus serviços normais tornam-se indisponíveis.
2. *Corrupção de programas ou dados*: os componentes de software podem ser alterados de maneira não autorizada, afetando o comportamento do sistema e, consequentemente, sua confiabilidade e segurança.
3. *Vazamento de informações confidenciais*: o ataque externo pode expor informações para pessoas não autorizadas, afetando a segurança do sistema, e permitindo posteriores ataques.

Engenharia de Software
Sistemas Críticos

Fonte:

Engenharia de Software – Ian Sommerville – 8ª ed.

Engenharia de Software

Gerenciamento da Qualidade

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Gerenciamento da Qualidade

Por que a qualidade de software é tão importante?

Engenharia de Software
Gerenciamento da Qualidade

Qualidade de Software

-Aprimoramento nos último anos (adoção de novas técnicas e tecnologias).

-Conscientização da importância do gerenciamento de qualidade de software.

-Adoção de técnicas de gerenciamento de qualidade provenientes da manufatura de software.

-Noção de qualidade é diferente em produtos manufaturados e em sistemas de software.

Engenharia de Software
Gerenciamento da Qualidade

Por que a qualidade de software é diferente da qualidade de produtos manufaturados?

-A especificação deve ser orientada para as características do produto que o cliente deseja. Mas, o desenvolvedor também pode ter requisitos que não são incluídos na especificação.

-Dificuldade em especificar determinadas características de qualidade de maneira não ambígua.

Engenharia de Software
Gerenciamento da Qualidade

Por que a qualidade de software é diferente da qualidade de produtos manufaturados?

-A dificuldade em escrever especificações de software completas pode fazer com que um produto não seja considerado de alta qualidade, por não corresponder às expectativas do cliente.

Engenharia de Software
Gerenciamento da Qualidade

Por que a qualidade de software é diferente da qualidade de produtos manufaturados?

-Alguns atributos de software como facilidade de manutenção, proteção ou eficiência não podem ser especificados explicitamente, mas são percebidos no sistema.

-"Cultura de qualidade": desenvolvida pelos gerentes de qualidade.

Engenharia de Software
Gerenciamento da Qualidade

Por que a qualidade de software é diferente da qualidade de produtos manufaturados?

-A criação de padrões e procedimentos são a base do gerenciamento de qualidade. Entretanto, é reconhecido que existem aspectos que não podem ser incorporados em padrões (aspectos intangíveis – elegância do software, capacidade de leitura).

Engenharia de Software
Gerenciamento da Qualidade

Documentação de qualidade

-Documentar o gerenciamento da qualidade significa registrar o que cada equipe dentro do projeto faz.

-Técnica utilizada principalmente no desenvolvimento de sistemas de grande porte e complexos.

-Ajuda a verificar se tarefas não foram "esquecidas".

Engenharia de Software
Gerenciamento da Qualidade

Documentação de qualidade

-É um meio de comunicação ao longo da existência de um sistema.

-Permite que os responsáveis pela evolução do sistema acompanhem o que a equipe de desenvolvimento está fazendo.

Engenharia de Software
Gerenciamento da Qualidade

O que é qualidade?

-"Grau de conformidade de um sistema, componente ou processo com os respectivos requisitos." (Glossário do IEEE).

-"Grau de conformidade de um sistema, componente ou processo com as necessidades e expectativas de clientes ou usuários."

Engenharia de Software
Gerenciamento da Qualidade

O que é qualidade?

-A qualidade é definida por uma coleção de atributos, tais como funcionalidade, confiabilidade, satisfação do usuário e desempenho, considerados importantes, mas parciais.

Engenharia de Software
Gerenciamento da Qualidade

Garantia de qualidade

-"Conjunto planejado e sistemático de ações necessárias para estabelecer um nível adequado de confiança de que um item ou produto está em conformidade com seus requisitos técnicos." (Glossário do IEEE).

-"Conjunto planejado e sistemático de meios para garantir à gerência que os padrões, métodos, práticas e procedimentos definidos por um processo são aplicados." (definição constante do CMMI).

Engenharia de Software
Gerenciamento da Qualidade

Gerenciamento de qualidade de software

-Garantia de qualidade: estabelecimento de um framework de procedimentos organizacionais e padrões que conduzem a um software de alta qualidade.

-Planejamento de qualidade: seleção de procedimentos e padrões apropriados deste framework, adaptados para um projeto de software específico.

Engenharia de Software
Gerenciamento da Qualidade

Gerenciamento de qualidade de software

-Controle de qualidade: definição e aprovação de processos que assegurem que a equipe de desenvolvimento de software tenha seguido os procedimentos e os padrões de qualidade de projeto.

Engenharia de Software
Gerenciamento da Qualidade

Gerenciamento de qualidade de processo envolve...

1. Definição de padrões de processo (como e quando as revisões devem ser conduzidas).

2. Monitoramento do processo de desenvolvimento para assegurar que os padrões estão sendo seguidos.

3. Relato do processo de software para a gerência de projeto e para o comprador do software.

Engenharia de Software
Gerenciamento da Qualidade

Importância dos padrões de software

-São baseados no conhecimento sobre as melhores e as mais apropriadas práticas para a empresa.

-Eles provêm um framework para a implementação do processo de garantia de qualidade.

-Eles ajudam na continuidade (adoção das mesmas práticas), reduzindo o esforço do aprendizado quando se inicia um novo trabalho.

Engenharia de Software
Gerenciamento da Qualidade

Importância dos padrões de software

-Principais instituições nacionais e internacionais que desenvolvem padrões de projetos:

-ANSI (American National Standards Institute).

-BSI (British Standards Institution).

-NATO (North Atlantic Treaty Organization).

-IEEE (Institute of Electrical and Eletronic Engineers).

-MPS-BR (Melhoria de Processo de Software Brasileiro)

Engenharia de Software
Gerenciamento da Qualidade

Importância dos padrões de software

-As equipes de garantia de qualidade que desenvolvem padrões para uma empresa devem tomar como base os padrões nacionais e internacionais.

-Prover ferramentas de software para apoiar os padrões.

-Conjunto internacional de padrões que pode ser usado no desenvolvimento de um sistema de gerenciamento de qualidade em todas as indústrias.

Engenharia de Software
Gerenciamento da Qualidade

Fonte: SOMMERVILLE, I. **Engenharia de Software.** 8ª edição

Engenharia de Software

Verificação e Validação

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Verificação e Validação

Qual é a importância dos processos de Verificação e Validação?

Engenharia de Software
Verificação e Validação

É o nome dado aos processos de verificação e análise, a fim de assegurar que o software cumpra com suas especificações e atenda às necessidades dos clientes que estão pagando por ele.

Engenharia de Software
Verificação e Validação

Incluem as tarefas:

1. revisões dos requisitos;
2. revisões de projeto;
3. inspeções de código;
4. testes de produto.

Engenharia de Software
Verificação e Validação

Verificação: checar se o software cumpre com suas especificações, ou seja, se o sistema cumpre com seus requisitos funcionais e não funcionais.

Validação: assegurar que o software atenda às expectativas do cliente, garantindo que ele faça o que o cliente espera que faça.

Engenharia de Software
Verificação e Validação

Nível de confiabilidade depende de:

Função do software: o quão crítico o software é para a organização (objetivo para o qual ele foi desenvolvido).

Expectativas do usuário: usuário vem se tornando mais exigente e, hoje, é menos aceitável entregar sistemas não confiáveis, o que implica em maior dedicação aos processos.

Engenharia de Software
Verificação e Validação

Nível de confiabilidade depende de:

Ambiente de mercado: o esforço a ser empregado neste processo depende de vários fatores, tais como: concorrência, preço que o cliente está disposto a pagar, cronograma.

Engenharia de Software
Verificação e Validação

Abordagens para a verificação e análise de sistemas neste processo

- *Inspeções de software* ou *revisões por pares*: analisam e verificam as representações do sistema (documento de requisitos, diagramas de projeto, código-fonte do programa). Podem ser aplicadas em todos os estágios do processo. São técnicas estáticas, pois não requerem que o sistema seja executado.

Engenharia de Software
Verificação e Validação

Abordagens para a verificação e análise de sistemas neste processo

- Testes de software: executam uma implementação do software com os dados de teste, a fim de examinar as saídas e o comportamento operacional, verificando-se, assim, se o sistema se comporta conforme o esperado. São técnicas dinâmicas.

Engenharia de Software
Verificação e Validação

Estrutura do plano de testes

1. Processo de teste: descrição das fases principais do processo de teste.

2. Rastreabilidadede requisitos: todos os requisitos devem ser individualmente testados.

3. Itens testados: especificação dos produtos do processo de software.

Engenharia de Software
Verificação e Validação

Estrutura do plano de testes

4. Cronograma de testes: apresentar um cronograma geral de testes e alocação de recursos para ele.

5. Procedimentos de registro de testes: trata-se do registro dos resultados dos testes. O processo de teste deve ser auditado para verificar que foi conduzido corretamente.

Engenharia de Software
Verificação e Validação

Estrutura do plano de testes

6. Requisitos de hardware e de software: ferramentas de software necessárias e a utilização estimada de hardware.

7. Restrições: o que afeta o processo de teste (por exemplo: falta de pessoal, falta de recursos).

Engenharia de Software
Verificação e Validação

Principais vantagens da Verificação e Validação em relação aos testes:

1. Inspeções

2. Revisões

Engenharia de Software
Verificação e Validação

Análise estática automatizada

Analisadores estáticos de programa são ferramentas de software que analisam o código-fonte de um programa e detectam possíveis defeitos e anomalias.

Não requerem que o programa seja executado.

Percorrem o texto do programa e reconhecem os diferentes tipos de declarações.

Engenharia de Software
Verificação e Validação

Análise estática automatizada

Detecção de anomalias no programa (variáveis sem iniciação, varáveis não utilizadas, dados com valores excedidos, etc.), podendo resultar em erros de programação e de omissões). Análise estática automatizada é mais bem utilizada com as inspeções de software.

Engenharia de Software
Verificação e Validação

Os estágios envolvidos incluem:

Análise do fluxo de controle: destaca loops com múltiplos pontos de saída ou de entrada e código inacessível.

Engenharia de Software
Verificação e Validação

Os estágios envolvidos incluem:

Análise da utilização de dados: detecta variáveis que são utilizadas sem prévia iniciação, variáveis declaradas mas nunca utilizadas, etc.

Engenharia de Software
Verificação e Validação

Os estágios envolvidos incluem:

Análise de interface: verifica a consistência das declarações de rotinas e procedimentos e seu uso; funções e procedimentos que são declarados e nunca chamados ou resultados de funções que nunca são utilizados.

Engenharia de Software
Verificação e Validação

Os estágios envolvidos incluem:

Análise do fluxo de informações: identifica as dependências entre as variáveis de entrada e as de saída.

Análise de caminho: identifica todos os caminhos possíveis no programa, e exibe as declarações executadas nesse caminho.

Engenharia de Software
Verificação e Validação

Fonte: SOMMERVILLE, I. **Engenharia de Software.** 8ª ed.

Engenharia de Software

Testes

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Testes

Qual é a importância dos Testes de Software?

Engenharia de Software
Testes

As duas atividades fundamentais para testes são:

- Testes de componentes (teste de partes do sistema).
- Teste de sistema (teste do sistema como um todo).

Engenharia de Software
Testes

As duas atividades fundamentais para testes são:

•Os testes de componentes objetivam descobrir defeitos, testando componentes individuais do programa (funções, objetos, componentes reusáveis).

•O teste de sistema integra esses componentes, para formar subsistemas ou sistemas completos, com o objetivo de verificar se o sistema atende aos requisitos funcionais e não funcionais, além de verificar, também, se o sistema não se comporta de maneira inesperada.

Engenharia de Software
Testes

Metas do processo de testes de software

Demonstrar ao desenvolvedor e ao cliente que o software atende aos requisitos (deve haver pelo menos um teste para cada requisito contido nos documentos de usuário e de sistema).

Engenharia de Software
Testes

Metas do processo de testes de software

Descobrir falhas ou defeitos no software que apresenta comportamento incorreto, não desejável ou em não conformidade com sua especificação (o teste de defeitos está relacionado à remoção de todos os tipos de comportamentos indesejáveis de sistema, como travamentos, interações indesejáveis com outros sistemas, cálculos incorretos e corrompimento de dados).

Engenharia de Software
Testes

Metas do processo de testes de software

- Para o teste de validação: um teste bem sucedido é aquele em que o sistema funciona corretamente.

- Para o teste de defeitos: um teste bem sucedido é o que expõe um defeito que causa o funcionamento incorreto do sistema.

Engenharia de Software
Testes

Metas do processo de testes de software

Os testes não podem demonstrar que um software é livre de defeitos ou que se comportará conforme o especificado, em todas as circunstâncias.

-A meta do teste de software é convencer os desenvolvedores e clientes do sistema de que o software é bom o suficiente para o uso operacional.

-O teste é um processo com o objetivo de se atingir a confiabilidade do software.

Engenharia de Software
Testes

Quem realiza os testes

- Os programadores têm a responsabilidade de testar os componentes que desenvolveram.

- Na próxima etapa, uma equipe é responsável por integrar os módulos de diferentes desenvolvedores e testar o sistema como um todo.

- Para sistemas críticos, pode-se utilizar um processo mais formal, onde testadores independentes são responsáveis por todos os estágios do processo de teste.

Engenharia de Software
Testes

Tipos de testes

1. Integração

2. Componentes

3. Projeto de casos de teste

4. Automação de testes

Engenharia de Software
Testes

Os testes podem mostrar somente a presença de erros em um programa. Eles não podem demonstrar que não existam defeitos remanescentes.

O teste de componentes é de responsabilidade do desenvolvedor de componentes. Outra equipe de testes geralmente executa os testes do sistema.

Engenharia de Software
Testes

Ao testar sistemas, deve-se tentar "quebrar" o sistema usando experiência e diretrizes para escolher os tipos de casos de teste eficazes na descoberta de defeitos em outros sistemas.

O teste de interfaces tenta descobrir defeitos nas interfaces dos componentes compostos. Defeitos de interface podem surgir devido a erros de leitura da especificação, má interpretação da especificação ou erros ou suposições inválidas de *timming*.

Engenharia de Software
Testes

O teste estrutural baseia-se na análise de um programa para determinar seus caminhos, e no uso dessa análise para ajudar na seleção dos casos de teste.

A automação reduz os custos de teste pelo apoio ao processo de teste com uma variedade de ferramentas de software.

Engenharia de Software
Testes

Fonte: SOMMERVILLE, I. **Engenharia de Software.** 8ª ed.

Engenharia de Software

Reuso de Software

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Reuso de Software

Nesta aula veremos de forma simples, objetiva e eficiente alguns recursos sobre Reuso de Software: Conceitos; Objetivos; Vantagens e Desvantagens; Quais os tipos possíveis; Quais são os Requisitos para reutilização; Quais são as Técnicas de Reuso; O que são Frameworks.

Engenharia de Software
Reuso de Software

É o processo de incorporar códigos existentes, especificações de requisitos, especificações de projeto e planos de teste.

Engenharia de Software
Reuso de Software

Objetivos:

Abstrair soluções bem sucedidas de projetos;

Reuso de software –embutidos em um sistema gerador;

Aplicar coleções de objetos concretos e abstratos que são projetadas para reuso por meio de especialização;

Engenharia de Software
Reuso de Software

Vantagens:

1. Melhores índices de produtividade;
2. Produtos de melhor qualidade;
3. Mais confiáveis, consistentes e padronizados;
4. Redução dos custos e tempo;
5. Maior flexibilidade na estrutura do software;

Engenharia de Software
Reuso de Software

Desvantagens:

1. Identificação, recuperação e modificação de artefatos reutilizáveis;

2. Aumento nos custos de manutenção;

3. Barreiras legais e econômicas;

4. Necessidade da criação de incentivos à reutilização;

Engenharia de Software
Reuso de Software

Quais os tipos possíveis?

Reuso de aplicação de sistema;

Reuso de componentes;

Reuso de objeto e função;

Engenharia de Software
Reuso de Software

Requisitos para reutilização:

Deve ser possível encontrar componentes reutilizáveis adequados;

Catalogação e documentação externa efetivas;

Deve-se ter certeza de que o componente se comportará conforme especificado e que será confiável;

Documentação interna detalhada;

Engenharia de Software
Reuso de Software

Alguns Fatores de Planejamento de Reuso:

O cronograma de desenvolvimento;

O ciclo de vida previsto do software;

O conhecimento, habilidades e experiência da equipe de desenvolvimento;

Requisitos funcionais e não funcionais;

O domínio da aplicação;

A plataforma de execução para o software;

Engenharia de Software
Reuso de Software

Quais são as Técnicas de Reuso?

As duas principais abordagens para são:

1.Design Patterns;

2.Geradores de programas;

Engenharia de Software
Reuso de Software

O que são Frameworks?

Frameworks são entidades moderadamente grandes que podem ser reusados.

São projetos de subsistemas para uma coleção de classes abstratas e concretas e as interfaces entre elas.

Engenharia de Software
Reuso de Software

Aplicabilidade:

Um sistema ERP (Enterprise Resource Planning) é um sistema genérico que apoia processos comuns de negócio, tais como pedidos e faturas, manufatura, etc.

São muito difundidos em empresas grandes –eles representam, provavelmente, a forma mais comum de reuso de software.

Engenharia de Software
Reuso de Software

Fonte: SOMMERVILLE, I. **Engenharia de Software.** 8ª ed.

Engenharia de Software

Sistema Embarcado

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Sistema Embarcado

Nesta aula veremos de forma simples, objetiva e eficiente alguns recursos sobre Sistema Embutido: Conceitos; Objetivos; Características; Vantagens; Disponibilidade; Processamento; Aplicabilidade; Periféricos.

Engenharia de Software
Sistema Embarcado

É um sistema complexo (microprocessado) no qual o microcomputador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla;

Engenharia de Software
Sistema Embarcado

Objetivo:

Realizar um conjunto de tarefas predefinidas, geralmente com requisitos específicos. Já que o sistema é dedicado a tarefas específicas, através de engenharia pode-se otimizar o projeto reduzindo tamanho, recursos computacionais e custo do produto.

Engenharia de Software
Sistema Embarcado

Vantagem:

Já que o sistema é dedicado a tarefas específicas, através de engenharia pode-se otimizar o projeto reduzindo tamanho, recursos computacionais e custo do produto.

Engenharia de Software
Sistema Embarcado

Características:

São desenvolvidos para uma tarefa específica. Por questões como segurança e usabilidade, alguns inclusive possuem restrições para computação em tempo real.

Engenharia de Software
Sistema Embarcado

Características:

O software escrito para sistemas embarcados é muitas vezes chamado firmware, e armazenado em uma memória ROM ou memória flash ao invés de um disco rígido.

Engenharia de Software
Sistema Embarcado

Características:

Por vezes o sistema também é executado com recursos computacionais limitados: sem teclado, sem tela e com pouca memória.

Engenharia de Software
Sistema Embarcado

Disponibilidade:

Sistemas embarcados residem em máquinas que, espera-se, possam trabalhar continuamente por anos ininterruptamente, e que possam por vezes recuperarem-se sozinhas após erros.

Engenharia de Software
Sistema Embarcado

Processamento:

Em geral os sistemas embarcados possuem uma capacidade de processamento reduzida em comparação com computadores desktops. Ao invés de utilizar microprocessadores, os desenvolvedores preferem utilizar microcontroladores, pois estes já possuem diversos periféricos integrados no mesmo chip.

Engenharia de Software
Sistema Embarcado

Periféricos:

Os sistemas embarcados comunicam-se com o meio externo através de periféricos. Estes periféricos podem ser combinados com o processador (como no caso dos sistemas microcontrolados) ou associados no sistema.

Engenharia de Software
Sistema Embarcado

Aplicabilidade:

Aviônicos; Telefones celulares e centrais telefônicas; roteadores, hubs, switches e firewalls; Impressoras; Dispositivos de armazenamento; Controladores do motor e do antibloqueio em automóveis: freios ABS e controle de tração;

Engenharia de Software
Sistema Embarcado

Aplicabilidade:

Calculadoras; Fornos microondas; Máquinas de lavar; Aparelhos de TV, DVD players; Equipamentos médicos; Videogames; PDAs; Tratores e implementos agrícolas; Urna eletrônica

Engenharia de Software
Sistema Embarcado

Fonte: SOMMERVILLE, I. **Engenharia de Software.** 8ª ed.

Engenharia de Software

ESTUDAR ...

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software

Métricas

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"

Engenharia de Software
Métricas

Métricas Orientadas ao Tamanho

-são medidas diretas do software e do processo por meio do qual ele é desenvolvido.

-se uma organização de software mantiver registros simples, uma tabela de dados orientados ao tamanho poderá ser criada. A tabela relaciona cada projeto de desenvolvimento de software que foi concluído no decorrer dos últimos anos.

Engenharia de Software
Métricas

Métricas Orientadas ao Tamanho

-A tabela relaciona cada projeto de desenvolvimento de software que foi concluído no decorrer dos últimos anos aos correspondentes dados orientados ao tamanho desse projeto.

-A partir dos dados brutos contidos na tabela, um conjunto de métricas de qualidade e de produtividade orientadas ao tamanho pode ser desenvolvido para cada projeto.

Engenharia de Software
Métricas

Métricas Orientadas ao Tamanho

-Médias podem ser computadas levando-se em conta todos os projetos.

-Para ocorrer uma implementação completa à nível de obtermos alguns resultados através de cálculos orientados ao tamanho.

Engenharia de Software
Métricas

Métricas Orientadas ao Tamanho

-Enfim, com a tabela montada é possível executar vários cálculos. Vale ressaltar, que a tabela geral foi criada com todos os projetos.

projeto	esforço	R$	Kloc/Loc	págs	erros	pessoas
teste01	1h	0	9	1	0	1
teste02	05 dias	240,00	6.000	120	32	2

Engenharia de Software
Métricas

Métricas Orientadas ao Tamanho

-Projeto = teste02

projeto	esforço	R$	Kloc/Loc	págs	erros	pessoas
teste02	05 dias	240,00	6.000	120	32	2

Calcular a produtividade: Kloc / esforço (6.000 / 5 = 1.200 Loc/dia)

Calcular a qualidade: erros / Kloc (32 / 6000 = 0,005)

Calcular o custo: R$ / Kloc (240,00 / 6.000 = R$ 0,04 por linha)

Calcular a Produt./Pessoa: Kloc / pessoas (6.000 / 2 = 3.000 linhas/pessoa)

Calcular o valor pago por pessoa: Kloc / pessoas x custo (6.000 / 2 x R$ 0,04 = 120,00)

Engenharia de Software
Métricas

Métricas Orientadas ao Tamanho

-Projeto = exercício

projeto	esforço	R$	Kloc/Loc	págs	erros	pessoas
exercício	2.540h	24.100,00	29.900	323	51	5

Calcular a produtividade: Kloc / esforço

Calcular a qualidade: erros / Kloc

Calcular o custo por Loc: R$ / Kloc

Calcular a Produt./Pessoa: Kloc / pessoas

Calcular o valor pago por pessoa: Kloc / pessoas x custo

Engenharia de Software
Métricas

Métricas Orientadas ao Tamanho

Agora, a pergunta que talvez tenhamos que fazer é: como Gerente de Projetos e não tendo nenhum valor como ponto de referência (Kloc, esforço, etc), como calcular e contratar uma empresa para desenvolver determinado sistema?

Resposta: utilizando métodos de “Estimativas” = esforço, prazo e custo

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

Talvez uma das tarefas mais cruciais para um Gerente de Projeto seja a elaboração de estimativas de esforço de desenvolvimento, prazo e custo de um projeto de sistemas. Negociar prazos e recursos para um projeto sempre deve acontecer quando do seu planejamento, jamais no meio ou no fim do projeto.

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

Estaremos abordando o Método – FPA (Function Point Analysis – Análise por Pontos de Função) – Os FP's são uma unidade de medida utilizada para determinar o 'tamanho' de uma aplicação (IBM, 1974).

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

São executadas algumas etapas, a 1ª é a parametrização de medidas através de valores definidos, e a 2ª através da pontuação por fator de complexidade do sistema, e os cálculos desejados.

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

Seus objetivos são:

-medir o que foi requisitado e recebido do usuário

-medir independente da tecnologia utilizada para a implementação

-prover uma métrica de medição para apoiar a análise de produtividade e qualidade

-prover uma forma de estimar o tamanho do software

-prover um fator de normalização para comparação de software

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

1ª etapa:

Identificar e enumerar as funções da aplicação: (simples – médio – complexo)

-número de entradas do usuário

-número de saídas do usuário

-número de consultas do usuário

-número de arquivos

-número de interfaces externas

Engenharia de Software
Métricas

Métricas Orientadas à Função

TABELA – ENTRADA

CAMPOS ARQUIVOS	1 a 4 itens de dados referenciados	5 a 15 itens de dados referenciados	16 ou mais itens de dados referenciados
0 ou 1 tipo de arquivo referenciado	SIMPLES	SIMPLES	MÉDIO
2 tipos de arquivos referenciados	SIMPLES	MÉDIO	COMPLEXO
3 ou mais tipos de arquivos referenciados	MÉDIO	COMPLEXO	COMPLEXO

Engenharia de Software
Métricas

Métricas Orientadas à Função

TABELA – SAÍDA

CAMPOS ARQUIVOS	1 a 5 itens de dados referenciados	6 a 19 itens de dados referenciados	20 ou mais itens de dados refere
0 ou 1 tipo de arquivo referenciado	SIMPLES	SIMPLES	MÉDIO
2 ou 3 tipos de arquivos referenciados	SIMPLES	MÉDIO	COMPLEXO
4 ou mais tipos de arquivos referenciados	MÉDIO	COMPLEXO	COMPLEXO

Engenharia de Software
Métricas

Métricas Orientadas à Função

TABELA – CONSULTA

CAMPOS ARQUIVOS	1 a 4 itens de dados referenciados	5 a 15 itens de dados referenciados	16 ou mais itens de dados referenciados
0 ou 1 tipo de arquivo referenciado	SIMPLES	SIMPLES	MÉDIO
2 tipos de arquivos referenciados	SIMPLES	MÉDIO	COMPLEXO
3 ou mais tipos de arquivos referenciados	MÉDIO	COMPLEXO	COMPLEXO

Engenharia de Software
Métricas

TABELA – ARQUIVO

CAMPOS REGISTROS	1 a 19 itens de dados referenciados	20 a 50 itens de dados referenciados	51 ou mais itens de dados referenciados
1 tipo de registro lógico	SIMPLES	SIMPLES	MÉDIO
2 a 5 tipos de registro lógico	SIMPLES	MÉDIO	COMPLEXO
6 ou mais tipos de registro lógico	MÉDIO	COMPLEXO	COMPLEXO

Engenharia de Software
Métricas

TABELA – INTERFACE

CAMPOS REGISTROS	1 a 19 itens de dados referenciados	20 a 50 itens de dados referenciados	51 ou mais itens de dados referenciados
1 tipo de registro lógico	SIMPLES	SIMPLES	MÉDIO
2 a 5 tipos de registro lógico	SIMPLES	MÉDIO	COMPLEXO
6 ou mais tipos de registro lógico	MÉDIO	COMPLEXO	COMPLEXO

Engenharia de Software
Métricas

Função	Nº de ocorrência	Complexidade	Peso	Resultado
Nº de entradas do usuário	0	Simples	3	0
	9	Médio	4	36
	4	complexo	6	24
Nº de saídas do usuário	0	Simples	4	0
	14	Médio	5	70
	0	complexo	7	0
Nº de consultas do usuário	0	Simples	3	0
	10	Médio	4	40
	4	complexo	6	24
Nº de arquivos	0	Simples	7	0
	13	Médio	10	130
	0	complexo	15	0
Nº de interfaces externas	1	Simples	5	5
	0	Médio	7	0
	0	complexo	10	0
(Fonte: Roger Pressman – Engenharia de Software)			Total de FP'b	329

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

2ª etapa: para identificar o Fator de Ajuste, deve-se estimar o Nível de Influência para cada uma das características das aplicações à seguir:

Comunicação de dados	**Processamento distribuído**
Performance	**Utilização de equipamento**
Volume de transações	**Entrada de dados on-line**
Eficiência do usuário final	**Atualização on-line**
Processamento complexo	**Reutilização de código**
Facilidade de implantação	**Facilidade operacional**
Múltiplos locais	**Facilidade de mudanças**

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

2ª etapa: o Nível de Influência de cada característica é dado por uma escala de 0 a 5.

0 = não existe, ou nenhuma influência (0%)

1 = pouca influência (1-20%)

2 = influência moderada (21-40%)

3 = influência média (41-60%)

4 = influência significativa (61-80%)

5 = grande influência (81-100%)

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

2ª etapa:

Comunicação de dados = 5	**Processamento distribuído = 3**
Performance = 5	**Utilização de equipamento = 5**
Volume de transações = 5	**Entrada de dados on-line = 5**
Eficiência do usuário final = 2	**Atualização on-line = 4**
Processamento complexo = 5	**Reutilização de código = 1**
Facilidade de implantação = 2	**Facilidade operacional = 3**
Múltiplos locais = 2	**Facilidade de mudanças = 5**
	Total de NI = 52

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

2ª etapa: utilizando o Nível de Influência Total, podemos determinar, então, o Fator de Ajuste (FA), de acordo com a fórmula: 0,65 + (0,01 x NI) = FA

0,65 + (0,01 x 52) = FA

0,65 + (0,52) = FA

FA = 1,17

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

Encontrando o FP = Pontos de Função (conforme citação no slide 8)

FP = FP’b x FA

FP = 329 x 1,17

FP = 375 (arredondar o resultado)

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

Com este valor de Pontos de Função (FP = 375) é possível executar os mais variados cálculos: o esforço, o prazo e o custo de um Projeto de Sistema (Software).

Cabe ao Gerente de Projeto decidir qual a Linguagem de Programação que será desenvolvido o Projeto.

Engenharia de Software
Métricas

Estimativas do número médio de LOC por FP

Cobol = 100

Pascal = 90

Linguagens Orientadas a Objeto (C++) = 30

Linguagens de 4ª e 5ª Geração

(Java + Delphi + Visual Basic) = 20

Geradores de Codigo

(SQL + HTML) = 15

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

Por exemplo, uma Linguagem de 4ª Geração:

-a tabela (NBR 13596 ... ISO 9126) aponta 20 Loc por FP

20 x 375 = 7.500 Kloc

Engenharia de Software
Métricas

Produtividade (Pessoa-Mês-Kloc)

Tipo de Sistema	Produtividade
Sistema Comercial	2.500 Kloc/Loc
Comércio Eletrônico	3.600 Kloc/Loc
Sistema Web	3.300 Kloc/Loc

Fonte: Método COCOMO

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

-a tabela (NBR 13596 ... ISO 9126) aponta para um sistema WEB um esforço de 3.300 Kloc/mês

7.500 / 3.300 = 2 meses e 3 dias

será o prazo para o desenvolvimento do Projeto

Engenharia de Software
Métricas

Métricas Orientadas por Pontos de Função

Estimativas = esforço, prazo e custo

-a tabela (NBR 13596 ... ISO 9126) aponta 132h/mês

132h x 2 meses = 264h

-por ex., se o custo/hora para desenvolvimento é R$ 90,00

O custo total do Projeto será de R$ 23.760,00

Engenharia de Software
Métricas

Qualidade do Projeto

A NBR 13596 lista um conjunto de características que deve ser verificado em um software para que ele seja considerado um “software de qualidade”.

-Funcionalidade

-Confiabilidade

-Usabilidade

-Eficiência

-Manuteniblidade

-Portabilidade

Engenharia de Software
Métricas

Referências

- Fonte:

- PRESSMAN, Roger S. **Uma Abordagem Profissional**. 7ª edição
- SOMMERVILLE, I. **Engenharia de Software.** 8ª ed.
- VAZQUEZ, Carlos Eduardo. **Análise de Pontos de Função**. 13ª edição.

Engenharia de Software
Métricas

ESTUDAR ...

"uma vela não perde sua chama acendendo outra"

"Apenas 5% dos professores fizeram, fazem e farão a diferença"