SUMÁRIO

SUMÁRIO **1**
Engenharia de Software **3**
Objetivos gerais 3
Livro Referência **4**
Hardware **4**
Software **4**
Elementos de um software 4
Instruções 4
Estruturas de Dados 5
Documentos 5
Características de um Software 5
Principais Características do Software 6
Tipo de software **6**
Tipo de software - Categorias modernas 7
Curva de defeitos de hardware **7**
Curva de defeitos do software **8**
Evolução de Software **8**
A primeira era do desenvolvimento de software - 1950 a 1965 8
A segunda era do desenvolvimento de software - 1963 a 1974 9
A terceira era do desenvolvimento de software - 1973 a 1978 9
A quarta era do desenvolvimento de software - 1985 aos dias atuais 9
Crise de Software **10**
O que é a crise do software? 10
Problemas relacionados a crise de software 10
As estimativas de prazo e de custo frequentemente são imprecisas 10
A produtividade das pessoas da área de software 10
A qualidade de software às vezes é menos que adequada 11
O software existente é difícil de manter 11
Resumo dos problemas associados à crise de software 12
Causas da Crise de Crise de Software 12
Natureza do software 12
Falhas das pessoas responsáveis pelo desenvolvimento de um software 13
Mitos de Software 13
Impacto das Mudanças **13**
Impacto das mudanças 13
Custos Relativos para Corrigir defeitos de Software 14
Você sabia? 14
Espaçonave da NASA 14
Ariane 5 15
Explosão de Gasoduto Soviético 15
Fundamentos da Engenharia de Software **15**
O que é engenharia de software? 15
Preocupação da engenharia de software 16

Principais metas da engenharia de software....16
Camadas da Engenharia de Software....16
Camada de Processo....17
Camada de Métodos....17
Camada de Ferramentas....18

Engenharia de Software

Engenharia de software é a disciplina tecnológica e gerencial preocupada com a produção sistemática e manutenção de produtos de software que são desenvolvidos e modificados no prazo estabelecido e dentro das estimativas de custo (Fairley, 1985). O objetivo da disciplina é ajudar o aluno a compreender e aplicar conhecimento científico para o projeto e construção de programas de computador e a documentação associada necessária para desenvolvê-los, operá-los e mantê-los (Boehm, 2011).

Esta disciplina ajudará o aluno a compreender o processo de engenharia de software, lhe possibilitará adquirir as habilidades e competências necessárias para entender os problemas das organizações, definir soluções tecnológicas, identificar e documentar as características de uma solução.

O processo de engenharia de software geralmente segue um ciclo de vida que inclui várias fases, como análise de requisitos, design, implementação, teste, manutenção e evolução. Cada fase tem seus próprios objetivos e atividades específicas para garantir que o software atenda aos requisitos do cliente, seja confiável, eficiente e fácil de manter.

Existem várias metodologias de desenvolvimento de software, como o modelo cascata, o modelo em espiral e as metodologias ágeis, como Scrum e Kanban. As metodologias ágeis têm ganhado destaque devido à sua abordagem flexível, colaborativa e iterativa, permitindo uma adaptação mais eficiente às mudanças nos requisitos durante o desenvolvimento.

Além disso, a engenharia de software também abrange práticas de garantia de qualidade, gerenciamento de configuração, gerenciamento de projetos e documentação adequada. A colaboração entre os membros da equipe, a comunicação eficaz e a adoção de boas práticas de codificação são fundamentais para o sucesso de projetos de engenharia de software.

No contexto atual, a engenharia de software enfrenta desafios constantes devido à rápida evolução das tecnologias, demandas crescentes por inovação e a necessidade de lidar com sistemas complexos. No entanto, a disciplina continua a ser crucial para o desenvolvimento bem-sucedido de software em diversas áreas, desde aplicações empresariais até sistemas embarcados e soluções de inteligência artificial.

Disciplina ministrada pelo Professor João de Deus Freire Júnior.

Objetivos gerais

- Discernir Software e Hardware, bem como as suas características;
- Conhecer os diversos tipos de software;
- Ter ciência dos diversos problemas que caracterizam a crise do software;
- Conhecer os principais paradigmas da Engenharia de Software;
- Conhecer a essência dos principais processos de desenvolvimento de software;
- Conhecer técnicas e ferramentas eficazes para identificar e documentar as características de uma solução de software.

Antes de estudar a disciplina e o processo de Engenharia de Software, é muito importante compreender os principais elementos e características de um software. Cada elemento de um software é essencial para a operação, evolução e manutenção do software. Entender as características do software possibilita a compreensão de sua natureza peculiar, seu processo de fabricação e tempo de vida. Com esse conhecimento será possível ter um melhor aprendizado da Engenharia de Software.

Livro Referência

PRESSMAN, R. S.(2011) Engenharia de Software: uma abordagem profissional. 7.ed. Porto Alegre: Bookman, 2016.

Hardware

O hardware é toda a parte física de um computador ou outro dispositivo computacional. Ele permite que as instruções enviadas pelos programas sejam executadas, que os usuários interajam com as aplicações e, entre outras coisas, que dados e informações sejam armazenados. Exemplos de hardware são: mouse, teclado, fones de ouvido, monitores e etc. Devido a sua natureza física o hardware se desgasta ao longo do tempo o que o torna passível de reparação ou inutilização.

Software

O software é a parte lógica que faz com que os computadores ou outros dispositivos computacionais funcionem. O software fornece instruções para o hardware de como ele deve funcionar e quais funções executar. Exemplos de software são: aplicativos para dispositivos móveis, sistemas operacionais, processadores de textos, jogos e diversos outros programas de computador. Pela sua natureza lógica, o software não se desgasta com o tempo, mas pode se deteriorar.

O software pode ser melhor definido pelas partes que o formam. Segue uma definição de software que segue essa lógica:

> *Software consiste em: (1) instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados; (2) estruturas de dados que possibilitam aos programas manipular informações adequadamente; e (3) informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas. (PRESSMAN, 2011, p. 32)*

Esta definição deixa claro os elementos que formam um software, que são: instruções, estrutura de dados e informação descritiva (documentação). Qualquer um desses elementos são essenciais para o funcionamento, evolução e manutenção do software.

O software é um produto produzido por profissionais especializados, ele pode dar suporte a operações computacionais, de negócios, médicas e etc. Ele pode ser disponibilizado gratuitamente para uso em massa, pode servir para divulgação de outros produtos, pode ser o diferencial competitivo de grandes corporações ou pode ser a única interação entre clientes e a empresa.

Elementos de um software

Os principais elementos de um software são: instruções, estrutura de dados e documentos.

Instruções

As instruções, quando executadas, produzem a função e desempenho desejados pelo software. As instruções dizem o que os dispositivos e máquinas devem fazer. O conjunto de instruções de um software é seu código-fonte. Este código pode ser escrito com uso de linguagem de programação como Java, Python e etc.

Estruturas de Dados

As estruturas de dados possibilitam que os programas manipulem adequadamente os dados e produzam informações. As estruturas de dados armazenam dados de forma eficiente. Elas podem ser: filas, pilhas, árvores binárias, objetos relacionais e etc.

Documentos

Os documentos descrevem a operação e uso do programa. A documentação do software detalha a estrutura do software, sua arquitetura, componentes, funcionamento e regras. A documentação bem elaborada é essencial para evolução e manutenção do software.

Características de um Software

Software é desenvolvido ou passa por um processo de engenharia; ele não é fabricado no sentido clássico.

Tanto software como hardware requerem a construção de um produto, porém, seus métodos de fabricação são totalmente diferentes. Ambas as atividades são dependentes de pessoas, mas a relação entre pessoas envolvidas e trabalho realizado é completamente diferente nesses produtos. Os custos de software concentram-se no processo de engenharia. Isso significa que projetos de software não podem ser geridos como se fossem projetos de fabricação.

O Software não “se desgasta”, mas, se deteriora, à medida que o tempo passa, os componentes de um hardware sofrem os efeitos cumulativos de poeira, vibração, impactos, temperaturas extremas e vários outros males ambientais ele começa a desgastar-se.

O software não é suscetível aos males ambientais que fazem com que o hardware se desgaste. Portanto, ele não se desgasta. Mas, o ambiente de negócios, a tecnologia, os processos, as organizações, sociedades, leis, regras mudam e todos esses aspectos podem fazer com que o software fique obsoleto e se deteriore.

De acordo com Pressman (2011), outro aspecto de desgaste ilustra a diferença entre hardware e software. Quando um componente de hardware se desgasta, ele é substituído por uma peça de reposição. Não existem peças de reposição de software. Cada defeito de software indica um erro no projeto ou no processo pelo qual o projeto foi traduzido em código-fonte. Portanto, as tarefas de manutenção de software, que envolvem solicitações de mudanças, implicam em complexidade maior do que a de manutenção de hardware.

Embora a indústria caminhe para a construção com base em componentes, a maioria dos softwares continua a ser construída de forma personalizada (sob encomenda).

À medida que a disciplina de Engenharia de Software evolui, cada vez mais componentes reutilizáveis são criados. Muitas empresas inclusive comercializam componentes ou softwares prontos chamados “pacotes” para outras empresas. Desta forma, as organizações não precisam construir aplicações ou componentes de software já existentes no mercado, elas só precisam pagar pela aquisição ou uso dos serviços.

Principais Características do Software

Intangibilidade: Diferentemente do hardware, o software não tem forma física. Ele é composto por código, dados que são armazenados eletronicamente e documentação.

Flexibilidade: Pode ser alterado e atualizado mais facilmente do que o hardware. Atualizações e correções de bugs podem ser distribuídas de maneira eficiente.

Dependência de Hardware: O software requer hardware para ser executado. Ele interage com o hardware para realizar tarefas específicas.

Abstração: Permite aos usuários interagir com o computador de maneira mais amigável, ocultando detalhes complexos do hardware.

Volatilidade: Pode ser facilmente modificado e atualizado, mas também pode ser perdido se não for armazenado adequadamente.

Linguagens de Programação: Os programas são escritos em linguagens de programação, que servem como meio de comunicação entre os desenvolvedores e o computador.

Tipo de software

O software é a tecnologia única mais importante no cenário mundial. O software se tornou uma tecnologia indispensável para negócios, ciência e engenharia; ele viabilizou a criação de novas tecnologias (por exemplo, engenharia genética e nanotecnologia), a extensão de tecnologias existentes (por exemplo, telecomunicações) e a mudança radical nas tecnologias mais antigas (por exemplo, indústria gráfica).

Se tornou a força motriz por trás da revolução do computador pessoal, já vemos pacotes de software sendo comprados pelos consumidores em lojas de bairro.

O software evoluiu lentamente de produto para serviço, na medida que empresas de software ofereceram funcionalidade imediata (just-in-time), via um navegador Web ou aplicativos móveis.

Companhias de software se tornaram as maiores e mais influentes companhias da era industrial criando a era digital, a Internet evoluiu e modificou tudo: de pesquisa em bibliotecas a compras feitas pelos consumidores, incluindo discurso político, hábitos de namoros de jovens e de adultos não tão jovens.

O software foi incorporado em sistemas de todas as áreas: transportes, medicina, telecomunicações, militar, industrial, entretenimento, máquinas de escritório e etc. Para atender toda essa demanda foram criados vários tipos de software.

Segue na tabela abaixo os tipos mais importantes:

Tipos de Software	Descrição e Exemplos
Básico	Programas de apoio a outros programas. Ex.: Sistema Operacional, Drivers e Compiladores.
De Tempo Real	Monitora, analisa e controla eventos do mundo real. Ex.: Controle de Tráfego Aéreo, Aviões, Trânsito e Usinas.
Comercial	Operações Comerciais e Tomada de Decisões Administrativas. Ex.: Sistemas de Administração Empresarial, Vendas e etc.
Científico e Engenharia	Algoritmos de Processamento de Números. Exemplos: Cálculos e etc.
De Computador Pessoal	Processamento de textos, planilhas eletrônicas, aplicativos de diversões e etc. Ex.: Pacote Office, Jogos e etc.
De Inteligência Artificial	Algoritmos não numéricos para resolver problemas que não sejam favoráveis à computação ou à análise direta. Ex.: Robôs, Chatbots e etc.
Embutido	Controla produtos e sistemas de mercados industriais e de consumo. Ex.: Lavadora, Iluminação e etc.
Mobile	Aplicativo para dispositivos móveis.

Tipo de software - Categorias modernas

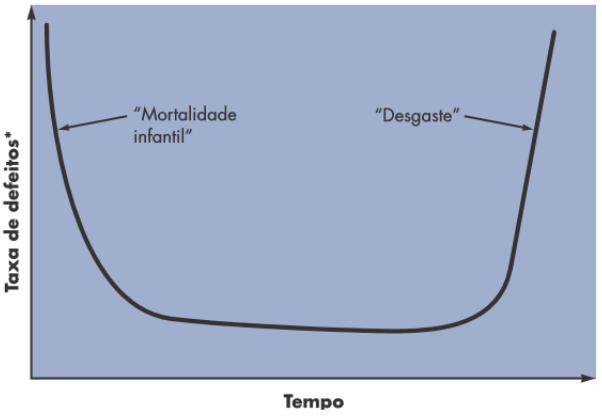
Com a evolução da tecnologia é possível observar novos tipos de software que, mesmo que possam ser classificados nas categorias apresentadas no tópico anterior, devem ser destacados separadamente também devido ao grande número de corporações e usuários que os utilizam. Segue na tabela abaixo esses tipos de software mais modernos:

Tipos de Software	Descrição e Exemplos
SAAS	Software como um Serviço. Ex.: Hospedagem de Sites e etc.
Redes Sociais	Comunicação, Conteúdo colaborativo e propositivo. Ex.: Facebook, Instagram e etc.
Plataformas	Fornecem a base para oferecimento e aquisição de serviços. Ex.: Uber e Airbnb.

Curva de defeitos de hardware

A curva de defeitos de um hardware tem um comportamento bem diferente da curva de defeitos de um software. Ela está relacionada à natureza física do hardware e de muitos outros produtos físicos. O hardware apresenta uma alta taxa de defeitos no início de seu ciclo de vida, isso acontece até que o equipamento e seus componentes sejam ajustados corretamente a sua função e ambiente.

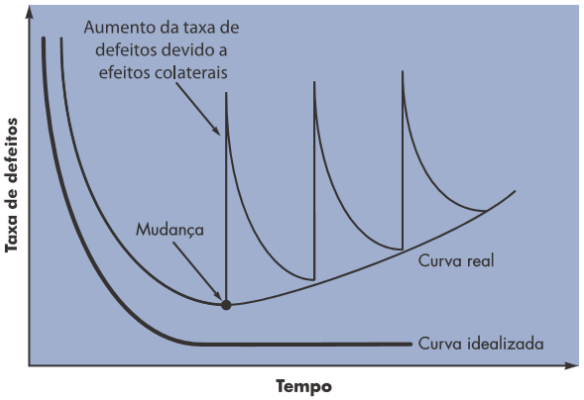
Após a correção desses defeitos e estabilização do produto, a taxa de defeitos do hardware se mantém estável e baixa até o fim de seu ciclo de vida quando é verificada uma alta taxa de defeitos novamente devido ao desgaste dos produtos por efeitos cumulativos de poeira, vibração, impactos e temperaturas extremas e vários outros males ambientais. Essa curva de defeitos do hardware é chamada de “curva da banheira”. Vejam na figura abaixo a curva de defeitos do hardware ao longo de seu ciclo de vida.



Curva de defeitos do software

A curva de defeitos de um software é bastante peculiar e tem um comportamento bem diferente da curva de defeitos de um hardware. Ela também tem uma alta taxa de defeitos no início do ciclo de vida do software assim como o hardware, porém, ela não se estabiliza ao longo do tempo como a do hardware.

Diferente do hardware, o software sofre diversas mudanças durante seu ciclo de vida. Essas mudanças acontecem para corrigir defeitos identificados tardiamente, para desenvolver melhorias no software, adaptações e evoluções. A cada mudança, como efeito colateral, há um aumento na taxa de defeitos daquele software. O software é estabilizado, mas o processo se reinicia com uma nova mudança. Vejam na figura abaixo a curva de defeitos do software ao longo de seu ciclo de vida.



Evolução de Software

A década de 1950 é um marco para o desenvolvimento de software. É nela que se inicia a primeira era do desenvolvimento de software. Da década de 1950 até os nossos dias tivemos quatro eras de desenvolvimento de software. (PFLEEGER, 2003)

Segue nos tópicos abaixo as principais características de cada era do software:

A primeira era do desenvolvimento de software - 1950 a 1965

A lista abaixo apresenta as principais características desta era de desenvolvimento do software:

- O desenvolvimento de software era considerado uma arte;
- Haviam poucos métodos sistemáticos para o desenvolvimento;
- O desenvolvimento de software não era gerenciado;
- O hardware sofria contínuas mudanças e era o centro das atenções;
- O software era customizado, ou seja, adequado às necessidades do usuário final, e a sua distribuição era limitada;
- O software era desenvolvido e utilizado pela mesma pessoa ou organização;
- Não havia documentação, todas as informações necessárias sobre o software estavam na cabeça das pessoas que o desenvolveram (one's head);
- O processamento de dados era em lote (batch).

A segunda era do desenvolvimento de software - 1963 a 1974

A lista abaixo apresenta as principais características desta era de desenvolvimento do software:

- Surgimento da multiprogramação e dos sistemas multiusuários;
- Desenvolvimento de técnicas interativas homem -máquina;
- Utilização de sistemas de tempo real;
- Surgimento da 1ª geração de Sistema Gerenciadores de Banco de Dados;
- Nascem as software houses e os produtos de software;
- O software era produzido para ampla distribuição em um mercado multidisciplinar, em várias áreas de conhecimentos;
- Surge o conceito de biblioteca de software;
- Devido à falta de metodologias de desenvolvimento e de documentação, a manutenção era praticamente impossível.

A terceira era do desenvolvimento de software - 1973 a 1978

A lista abaixo apresenta as principais características desta era de desenvolvimento do software:

- Surgimento dos sistemas distribuídos e paralelos;
- Desenvolvimento das redes locais e globais de computadores;
- Necessidade de elevada demanda por acesso imediato a dados por parte dos usuários;
- Criação dos computadores de uso pessoal (PC - personal computers) e estações de trabalho (workstations);
- Uso generalizado de microprocessadores;
- Grande consumos de computadores;
- Os computadores se tornam acessíveis.

A quarta era do desenvolvimento de software - 1985 aos dias atuais

A lista abaixo apresenta as principais características desta era de desenvolvimento do software:

- Tecnologias orientadas a objetos;
- Sistemas especialistas e software de inteligência artificial usados na prática;
- Software de rede neural artificial;

- Computação Paralela[1];
- Internet;
- Dispositivos móveis;
- Redes sociais.

Crise de Software

O processo de desenvolvimento de software é complexo e a demanda por software cresceu e ainda cresce exponencialmente. O software traz diferenciais competitivos para as organizações, serviços base e suportam as operações. Ele é essencial. Porém, há muitos problemas encontrados nos projetos de construção, melhorias e manutenção de software. De fato, vivemos uma crise do software.

O que é a crise do software?

A Crise do Software foi um termo que surgiu nos anos 70 que expressava as dificuldades do desenvolvimento de software frente ao rápido crescimento da demanda por software, da complexidade dos problemas a serem resolvidos e da inexistência de técnicas de desenvolvimento de sistemas.

Problemas relacionados a crise de software

Há diversos problemas associados à Crise do Software que são percebidos pelos usuários, desenvolvedores e outros envolvidos com a atividade de criação, evolução e manutenção de software. Esses problemas são as "dores" sentidas por todos envolvidos com software. Eles têm prejudicado muito as corporações, as empresas de tecnologia, entre outros. Segue abaixo quatro exemplos desses problemas:

As estimativas de prazo e de custo frequentemente são imprecisas

As estimativas de prazo e custo necessários para desenvolvimento de software geralmente falham e são muito maiores do que o previsto. Os prazos maiores do que os previstos afetam muito as corporações, pois, frequentemente o prazo pode estar associado ao início de uma nova operação de negócios, lançamento de novo produto e etc.; as corporações são afetadas também pelos custos maiores que os orçados causando cancelamento de projetos, prejuízos, perdas financeiras e etc. As falhas nas estimativas estão associadas às causas abaixo:

- As equipes de desenvolvimento de software não dedicam tempo para coletar dados sobre o processo de desenvolvimento de software.
- As equipes de desenvolvimento de software não tem nenhuma indicação sólida de produtividade, não podendo avaliar com precisão a eficácia de novas ferramentas, métodos ou padrões.

A produtividade das pessoas da área de software

O software é a tecnologia única mais importante no cenário mundial. O software se tornou uma tecnologia indispensável para negócios, ciência e engenharia; ele viabilizou a criação de novas tecnologias (por exemplo, engenharia genética e nanotecnologia), a extensão de tecnologias existentes (por exemplo, telecomunicações) e a mudança radical nas tecnologias mais antigas (por exemplo, indústria gráfica); se tornou a força motriz por trás da revolução do computador pessoal; já vemos pacotes de software sendo comprados pelos consumidores

[1] A computação paralela é uma abordagem em que várias tarefas ou partes de um programa são executadas simultaneamente em múltiplos processadores ou núcleos de um sistema, com o objetivo de acelerar o processamento e aumentar a eficiência computacional.

em lojas de bairro; o software evoluiu lentamente de produto para serviço, na medida que empresas de software ofereceram funcionalidade imediata (just-in-time), via um navegador Web ou aplicativos móveis; companhias de software se tornaram as maiores e mais influente companhias da era industrial criando a era digital; a Internet, iria evoluiu e modificou tudo: de pesquisa em bibliotecas a compras feitas pelos consumidores, incluindo discurso político, hábitos de namoros de jovens e de adultos não tão jovens. (PRESSMAN, 2011)

Os sistemas têm de ser construídos e entregues mais rapidamente devido a todo esse aumento exponencial da demanda; sistemas maiores e até mais complexos são requeridos; sistemas devem ter novas capacidades que antes eram consideradas impossíveis. Como os métodos de engenharia de software existentes não conseguem lidar com isso, novas técnicas de engenharia de software precisam ser desenvolvidas para atender a essas novas demandas. A produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços. (SOMMERVILLE, 2011).

A qualidade de software às vezes é menos que adequada

Com o aumento da demanda de desenvolvimento de software exposto no tópico anterior, as empresas do setor de Tecnologia da Informação têm sido pressionadas a oferecer soluções de maior qualidade em prazos cada vez menores.

Em contrapartida, uma pesquisa recente publicada em 2012 e feita com organizações nacionais revelou que 43% dos projetos de TI, em média, foram cancelados ou entregues com falhas comprometedoras no processo e/ou produto. (RIBEIRO et al., 2011).

A qualidade de software é comumente menor que adequada. Os problemas de qualidade entre outros fatores estão associados a falta de conceitos qualitativos sólidos de garantia de qualidade e utilização de práticas e ferramentas para assegurar a qualidade do software entregue.

O software existente é difícil de manter

De acordo com a IEEE a definição de engenharia de software é:

> "A aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software; isto é, a aplicação de engenharia ao software." (IEEE, 1993)

Essa definição deixa claro a importância da manutenção de software no processo de engenharia de software. A manutenção começa logo no começo do uso do software. Logo que o software é liberado para os usuários finais, e em alguns dias, erros começam a ser relatados à equipe de desenvolvimento do software. Em adição, mudanças são solicitadas por grupos de usuários para adaptar o software às suas necessidades e para melhor atender suas operações de negócios e clientes. Os usuários sempre precisarão de algumas melhorias para fazer o software funcionar em seu mundo.

Com isso, o desafio da manutenção do software começou. As equipes de sustentação e desenvolvimento do software terão que trabalhar paralelamente na correção de bugs, solicitações de adaptação e melhorias que devem ser planejadas, programadas e, por fim, executadas. Logo, a fila já cresceu muito e o trabalho ameaça devorar os recursos disponíveis. Com o passar do tempo, sua organização descobre que está gastando mais tempo e dinheiro com a manutenção dos programas do que criando novas aplicações. De fato, não é raro uma

organização de software despender de 60% a 70% de todos os recursos com manutenção de software. (PRESSMAN, 2011)

Os motivos de muito trabalho na manutenção de software são muitos. Osborne e Chikofsky fornecem uma resposta parcial:

> "Muitos softwares dos quais dependemos hoje têm em média de 10 a 15 anos. Mesmo quando esses programas foram criados, usando as melhores técnicas de projeto e codificação conhecidas na época [e muitos não foram], o tamanho do programa e o espaço de armazenamento eram as preocupações principais. Eles então migraram para novas plataformas, foram ajustados para mudanças nas máquinas e na tecnologia dos sistemas operacionais e aperfeiçoados para atender a novas necessidades dos usuários – tudo isso sem grande atenção na arquitetura geral. O resultado são estruturas mal projetadas, mal codificadas, de lógica pobre e mal documentadas em relação aos sistemas de software, para os quais somos chamados a fim de mantê-los rodando."(OSBORNE, 1990, p.10-11)

As más práticas de desenvolvimento e má elaboração ou inexistência de documentação de software levam a grande dificuldade na manutenção do software.

Resumo dos problemas associados à crise de software

A tabela abaixo apresenta um resumo dos problemas associados à crise de software.

Problema	Status na Crise
Prazos e Custos em relação às estimativas	Mais altos
Produtividade das Pessoas	Baixa
Qualidade de Software	Baixa
Dificuldade de manter software	Alta

Causas da Crise de Crise de Software

As causas associadas à crise de software são diversas. Vamos ressaltar neste tópico as três principais.

Natureza do software

A própria natureza do software é uma das causas da crise do software devido às suas próprias características. Segue essas características:

- O software é um elemento de sistema lógico e não físico (produto intangível). Consequentemente, o sucesso é medido pela qualidade de uma única entidade e não pela qualidade de muitas entidades manufaturadas.
- O software não se desgasta, mas se deteriora!

À medida que o tempo passa, os componentes de um hardware sofrem os efeitos cumulativos de poeira, vibração, impactos, temperaturas extremas e vários outros males ambientais ele começa a desgastar-se.

O software não é suscetível aos males ambientais que fazem com que o hardware se desgaste. Portanto, ele não se desgasta. Mas, o ambiente de negócios, a tecnologia, os processos, as organizações, sociedades, leis, regras mudam e todos esses aspectos podem fazer com que o software fique obsoleto e se deteriore.

Falhas das pessoas responsáveis pelo desenvolvimento de um software

As falhas das pessoas responsáveis pelo desenvolvimento de um software é uma das causas da crise do software. Isso acontece porque:

- Alguns gerentes de TI não tem nenhum background em software;
- Os profissionais da área de software têm recebido pouco treinamento formal em novas técnicas para o desenvolvimento de software;
- Alguns profissionais e gestores de TI são resistentes a mudanças.

Mitos de Software

Os mitos de software são inverdades sobre o processo de engenharia de software que propagam desinformação e confusão e eles são também uma das causas da crise do software. Os mitos podem ser:

- Administrativos;
- De clientes;
- Profissionais.

Impacto das Mudanças

Uma mudança, quando solicitada tardiamente num projeto, pode ser maior do que mais do que uma ordem de magnitude mais dispendiosa do que a mesma mudança solicitada nas fases iniciais. As mudanças impactam muito o andamento dos projetos podem inclusive inviabilizar a continuidade de um projeto devido ao grande impacto no todo. Por isso, as fases iniciais de levantamento de requisitos de um projeto são essenciais para seu sucesso.

Posso solicitar mudanças no software que está sendo desenvolvido quando eu quiser? As mudanças são facilmente recebidas e absorvidas sem impacto pela equipe de desenvolvimento? O impacto das mudanças diferem de acordo com o momento que são solicitadas? É vital entendermos qual o impacto das solicitações de mudanças no ciclo de desenvolvimento de software.

Impacto das mudanças

O conhecimento acumulado de desenvolvimento de software (baseada em décadas de experiência) afirma que o impacto de mudanças aumentam de forma não linear conforme o projeto avança (Figura 1.2, curva em preto contínuo). É de certa forma fácil absorver uma mudança quando uma equipe de software está levantando requisitos (no início de um projeto). Pode-se ter de alterar um detalhamento do uso, de uma funcionalidade, ampliar uma lista de funções, alterar algo relacionado ao desenho das interfaces com o usuário ou editar uma especificação por escrito. Os custos de tal trabalho são mínimos e o tempo demandado não afetará negativamente o resultado do projeto. Mas, se adiantarmos alguns meses, o que aconteceria? O time de desenvolvimento estará em meio aos testes de validação (que ocorrem relativamente no final do projeto) e um importante interessado está requisitando uma mudança funcional grande, que muitas vezes provoca mudanças estruturais no software. A mudança requer uma alteração no projeto da arquitetura do software, o projeto e desenvolvimento de vários novos componentes, modificações em vários outros componentes, especificação de

novos testes, e assim por diante. Os impactos crescem rapidamente e não serão triviais o tempo e custos necessários para assegurar que a mudança seja feita sem efeitos colaterais inesperados. (PRESSMAN, 2011)

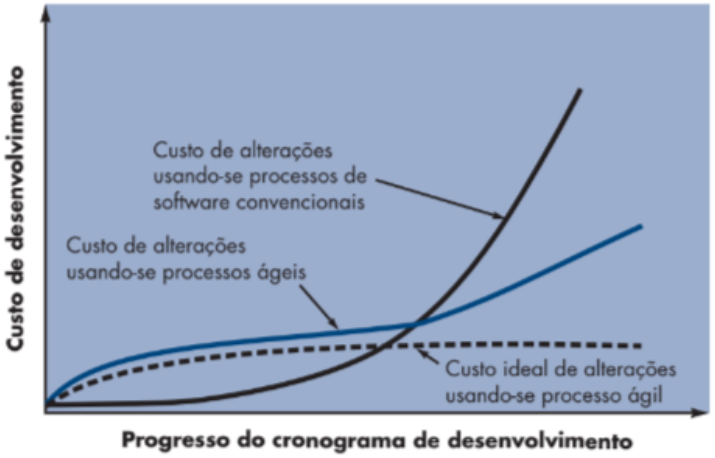
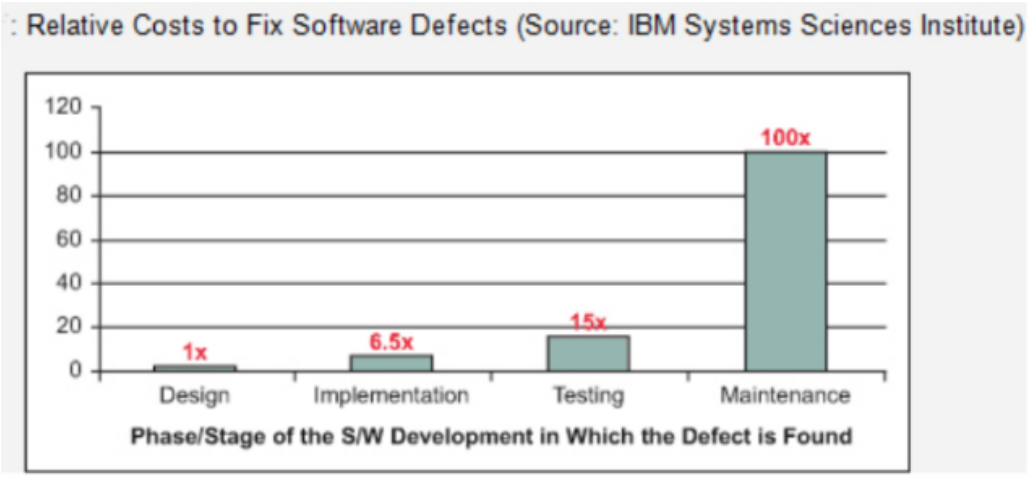


Figura 1.2. Custos de alterações como uma função do tempo em desenvolvimento

Custos Relativos para Corrigir defeitos de Software



Fonte: IBM, 2015

- Defeito de software na fase de desenho o custo é 1 vez maior.
- Defeito de software na fase de implementação/construção/codificação custo 6.5 vez maior.
- Defeito de software na fase de teste custo 15 vezes maior.
- Defeito de software na fase de produção/cliente usando custo 100 vezes maior.

Você sabia?

Há vários exemplos conhecidos de grandes projetos impactados por mudanças ou defeitos identificados na fase de manutenção do software em produção. Segue nos próximos tópicos alguns exemplos.

Espaçonave da NASA

Em sua missão a Marte em 1998, a espaçonave Climate Orbiter acabou perdida no espaço. Embora a falha tenha confundido os engenheiros por algum tempo, foi revelado que um subcontratado da equipe de engenharia não conseguiu fazer uma conversão simples de unidades inglesas para métricas. Um lapso embaraçoso que

enviou a nave de US $125 milhões fatalmente para perto da superfície de Marte, após tentar estabilizar sua órbita muito baixa. Os controladores de voo acreditam que a espaçonave invadiu a atmosfera de Marte, onde as tensões associadas prejudicaram suas comunicações, deixando-a voando pelo espaço em uma órbita ao redor do sol.

Ariane 5

O mais novo foguete de lançamento de satélite não tripulado da Europa utilizou o software funcional de seu antecessor, o Ariane 4. Infelizmente, os motores mais rápidos do Ariane 5 exploraram um bug que não foi encontrado nos modelos anteriores. Trinta e seis segundos após o lançamento inicial, os engenheiros do foguete apertaram o botão de autodestruição após várias falhas do computador. Em essência, o software tentou amontoar um número de 64 bits em um espaço de 16 bits. As condições de estouro resultantes travaram os computadores principal e de backup (que estavam executando exatamente o mesmo software).

O Ariane 5 havia custado quase US $8 bilhões para ser desenvolvido e carregava uma carga útil de satélite de US $500 milhões quando explodiu.

Explosão de Gasoduto Soviético

O oleoduto soviético tinha um nível de complexidade que exigiria um software de controle automatizado avançado. A CIA (Central de Inteligência dos Estados Unidos) foi informada das intenções soviéticas de roubar os planos do sistema de controle. Trabalhando com a empresa canadense que projetou o software de controle de dutos, a CIA fez com que os projetistas criassem deliberadamente falhas na programação para que os soviéticos recebessem um programa comprometido. Alega-se que, em junho de 1982, falhas no software roubado levaram a uma explosão massiva ao longo de parte do oleoduto, causando a maior explosão não nuclear da história do planeta.

Fundamentos da Engenharia de Software

A Engenharia de Software foi criada com o intuito de resolver ou minimizar os impactos da crise de software. Ela é o emprego de boas práticas no desenvolvimento de software, a aplicação de processos, métodos e ferramentas adequadas para o desenvolvimento econômico e de qualidade do software. As preocupações e metas de engenharia de software miram a evolução e sistematização do processo de desenvolvimento de software.

O que é engenharia de software?

Uma boa definição de Engenharia de software é que ela é o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter software de maneira econômica, que seja confiável e funcione de forma eficiente em máquinas reais. (Bauer apud Pressman, 2011)

Outra boa definição também citada pelo Pressman e oriunda do dicionário internacional de engenharia é que Engenharia de software é a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software; isto é, a aplicação de engenharia ao software. (IEEE apud Pressman, 2011)

Um resumo e simplificação dessas duas definições poderia ser que a engenharia de software prescreve a aplicação de abordagens e processos sistemáticos para possibilitar o desenvolvimento de software de maneira econômica e com qualidade.

Preocupação da engenharia de software

A engenharia de software se preocupa com a sistematização do processo de criação e manutenção de software. Ela tem por objetivo apoiar o desenvolvimento profissional de software. Ela inclui técnicas que apoiam especificação, projeto e evolução de programas. (Sommerville, 2011).

Principais metas da engenharia de software

As principais metas da engenharia de software são:

1. Melhorar a qualidade de produtos de software;
2. Aumentar a produtividade do pessoal técnico e;
3. Aumentar a satisfação dos clientes.

Por isso, a disciplina de engenharia de software se preocupa com todos os aspectos da produção de software desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado. (Sommerville, 2011)

Segue abaixo uma afirmação de Sommerville (2011, p.5) sobre as preocupações da engenharia de software:

> "A engenharia de software não se preocupa apenas com os processos técnicos do desenvolvimento de software. Ela também inclui atividades como gerenciamento de projeto de software e desenvolvimento de ferramentas, métodos e teorias para apoiar a produção de software. Engenharia tem a ver com obter resultados de qualidade requeridos dentro do cronograma e do orçamento."

Camadas da Engenharia de Software

Podemos dividir a atuação da engenharia de software em camadas. Contudo, todas essas camadas devem ter como foco o comprometimento organizacional com a qualidade, a promoção de uma cultura de aperfeiçoamento contínuo de processos, e é esta cultura que, no final das contas, leva ao desenvolvimento de abordagens cada vez mais efetivas na engenharia de software. A pedra fundamental que sustenta a engenharia de software é o foco na qualidade. (PRESSMAN, 2011)

As camadas da engenharia de software são: Processo, Métodos e Ferramentas. Todas essas camadas como afirmado acima tem o foco de proporcionar mais qualidade ao software. Elas servem como base e suportam umas às outras.

Segue abaixo uma figura que exibe as camadas da engenharia de software:

Camada de Processo

A camada base da engenharia de software é a camada de processo. Ela constitui o elo de ligação entre os métodos e ferramentas e define a sequência em que os métodos serão aplicados.

Segue uma explicação do Pressman (2011, p. 40) sobre essa camada:

> "A base para a engenharia de software é a camada de processos. O processo de engenharia de software é a liga que mantém as camadas de tecnologia coesas e possibilita o desenvolvimento de software de forma racional e dentro do prazo. O processo define uma metodologia que deve ser estabelecida para a entrega efetiva de tecnologia de engenharia de software. O processo de software constitui a base para o controle do gerenciamento de projetos de software e estabelece o contexto no qual são aplicados métodos técnicos, são produzidos produtos derivados (modelos, documentos, dados, relatórios, formulários etc.), são estabelecidos marcos, a qualidade é garantida e mudanças são geridas de forma apropriada."

Define o conjunto de atividades, métodos e ferramentas que serão utilizados para o desenvolvimento do software. Essa camada estabelece o "como" do desenvolvimento, definindo o ciclo de vida do software, as práticas de gerenciamento de projeto e as técnicas de qualidade a serem utilizadas.

Esta camada lida com os processos e metodologias utilizados para desenvolver software de forma eficaz. Isso inclui a definição de modelos de ciclo de vida, como o modelo cascata, modelo em espiral[2] e metodologias ágeis, como Scrum e Kanban. A camada de processo também engloba práticas de gestão de projetos, planejamento, estimativa, acompanhamento e controle de atividades de desenvolvimento de software.

Constituem o elo de ligação entre os métodos e ferramentas e definem a sequência em que os métodos serão aplicados.

Alguns Processos:

- Desenvolvimento em Espiral
- Entrega

Camada de Métodos

A segunda camada da engenharia de software é a camada de métodos. Ela fornece os detalhes de como fazer para construir o software. Abrange as técnicas e práticas específicas para a construção do software.

Segue uma explicação do Pressman (2011, p. 40) sobre essa camada:

> "Os métodos da engenharia de software fornecem as informações técnicas para desenvolver software. Os métodos envolvem uma ampla gama de tarefas, que incluem: comunicação,

[2] O modelo em espiral é um modelo de desenvolvimento de software que combina elementos do modelo de cascata com iterações típicas de metodologias ágeis. Ele foi proposto por Barry Boehm em 1986 e é adequado para projetos de grande escala ou onde os requisitos são incertos ou sujeitos a mudanças significativas ao longo do tempo. O modelo em espiral é representado graficamente como uma espiral, na qual cada ciclo ou volta da espiral representa uma fase do processo de desenvolvimento.

> análise de requisitos, modelagem de projeto, construção de programa, testes e suporte. Os métodos da engenharia de software baseiam-se em um conjunto de princípios básicos que governam cada área da tecnologia e inclui atividades de modelagem e outras técnicas descritivas."

Esta camada trata dos métodos, técnicas e ferramentas utilizadas para realizar atividades específicas durante o desenvolvimento de software. Isso inclui métodos de análise de requisitos, design de software, implementação, teste, manutenção e gestão de configuração. Além disso, nesta camada estão presentes práticas de modelagem, como diagramas UML (Unified Modeling Language), padrões de projeto e abordagens de desenvolvimento orientado a objetos.

Proporcionam os detalhes de como fazer para construir o software.
Alguns Métodos:

- Planejamento e estimativa de projeto
- Análise de requisitos de sistemas e de software
- Projeto da estrutura de dados
- Algoritmo de processamento
- Codificação
- Teste
- Manutenção

Camada de Ferramentas

A terceira camada da engenharia de software é a camada de ferramentas. As ferramentas são suporte automatizado aos métodos. Compõe o conjunto de ferramentas de software que auxiliam nas diversas etapas do desenvolvimento.

Segue uma explicação do Pressman (2011, p. 40) sobre essa camada:

> "As ferramentas da engenharia de software fornecem suporte automatizado ou semi automatizado para o processo e para os métodos. Quando as ferramentas são integradas, de modo que as informações criadas por uma ferramenta possam ser usadas por outra, é estabelecido um sistema para o suporte ao desenvolvimento de software, denominado engenharia de software com o auxílio do computador."

Nesta camada estão as ferramentas de software utilizadas para auxiliar nas atividades de desenvolvimento, teste, depuração, documentação e gestão de projetos de software. Isso inclui ambientes de desenvolvimento integrado (IDEs), sistemas de controle de versão (como Git), ferramentas de automação de teste, sistemas de gerenciamento de requisitos e ferramentas de gestão de projetos. As ferramentas nesta camada visam aumentar a eficiência, produtividade e qualidade do processo de engenharia de software.

Dão suporte automatizado aos métodos.

Algumas Ferramentas:

- Eclipse e outras IDEs (Codificação)
- SONAR Code (Qualidade de Código)

- Selenium (Testes)