Sistemas de Informação
Disciplina: Engenharia de Software e da Usabilidade
Professor: José Maurício S. Pinheiro

AULA 1 – Introdução a Engenharia de Software

1. Introdução

Há 20 anos, menos de 1% do público poderia descrever de forma inteligível o que significa "software de computador". Hoje, a maioria dos profissionais bem como a maior parte do público, acham que entendem o que é software. Será que entendem?
Uma descrição de software poderia assumir as seguintes formas:

- instruções (programas de computador) que, quando executadas, produzem a função e o desempenho desejados;
- estruturas de dados que possibilitam que os programas manipulem adequadamente a informação;
- documentos que descrevem a operação e o uso dos programas. Não há dúvida de que outras definições, mais completas, poderiam ser oferecidas.

Segundo Pressman (2006), um software é um conjunto composto por instruções de computador, estruturas de dados e documentos. Para Bauer (1969), a Engenharia de Software é “a criação e a utilização de sólidos princípios de engenharia a fim de obter softwares econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais”.
Segundo o IEEE (1992), “Engenharia de software é a aplicação de uma abordagem sistemática, disciplinada e quantificável, para o desenvolvimento, operação e manutenção do software; isto é, a aplicação de engenharia ao software.”
Existem diferentes tipos de sistemas de software, desde os simples sistemas embutidos até os sistemas de informações complexos, de alcance mundial. Não faz sentido procurar notações, métodos ou técnicas universais para a engenharia de software, porque diferentes tipos de software exigem abordagens diferentes. Por exemplo, desenvolver um sistema de informações corporativo é totalmente diferente de desenvolver um controlador para um instrumento cientifico. Nenhum desses sistemas tem muito em comum com um jogo computacional com gráficos intensos. Todas essas aplicações precisam de engenharia de software, embora não necessitem das mesmas técnicas.
Outra consideração é que o software é um elemento de sistema lógico, e não físico. Portanto, o software tem características que são consideravelmente diferentes das do hardware:

- O software é desenvolvido e projetado por engenharia, não manufaturado no sentido clássico;
- O software não se desgasta;
- A maioria dos softwares é feita sob medida em vez de ser montada a partir de componentes existentes.

Produtos de software podem ser:

- Genéricos – desenvolvidos para serem vendidos para uma grande variedade de clientes, por exemplo, softwares para PC, tais como Excel e Word;
- Personalizados – desenvolvidos para um único cliente de acordo com as suas especificações.

2. Breve Histórico

Ao longo da década de 1980, avanços na microeletrônica resultaram em maior poder de computação a um custo cada vez mais baixo. Hoje, o problema é diferente. O principal desafio é melhorar a qualidade (e reduzir o custo) de soluções baseadas em computador - soluções que são implementadas com software.
O software deve, antes de tudo, ser visto como um elemento de um sistema mais amplo, ao qual denominaremos um Sistema Computacional, que é o resultado da união de diferentes componentes, entre os quais se enquadram o software, o hardware e elementos de outras naturezas como por exemplo o elemento humano, representado pelo usuário do sistema. Nestes sistemas, o software vem, há muitos anos, substituindo o hardware como o elemento de mais difícil concepção, com menor probabilidade de sucesso em termos de prazos e custo e de mais difícil administração. Além disso, a demanda por software cresce a cada dia, até como consequência do grande desenvolvimento do hardware dos sistemas computacionais (Fig. 1).

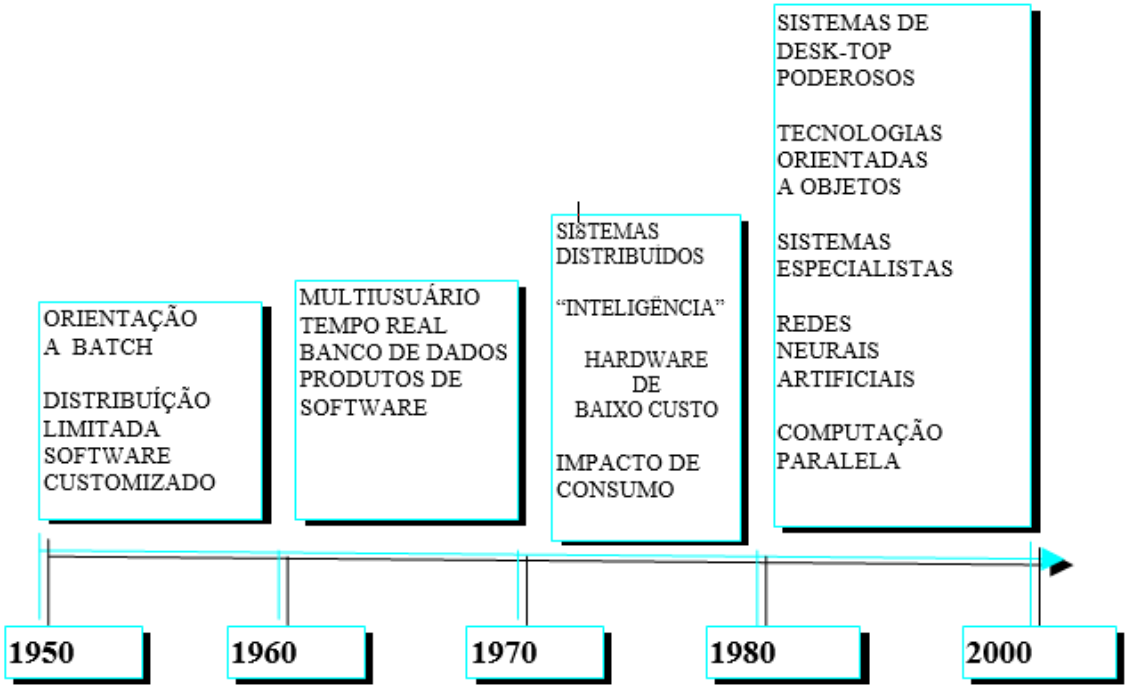

Figura 1 - Evolução do Software

O contexto em que o software foi desenvolvido está estreitamente ligado a quase sete décadas de evolução dos sistemas computadorizados. O melhor desempenho de hardware, menor tamanho e custo mais baixo precipitaram o aparecimento de sistemas baseados em computadores mais sofisticados (Fig.2).

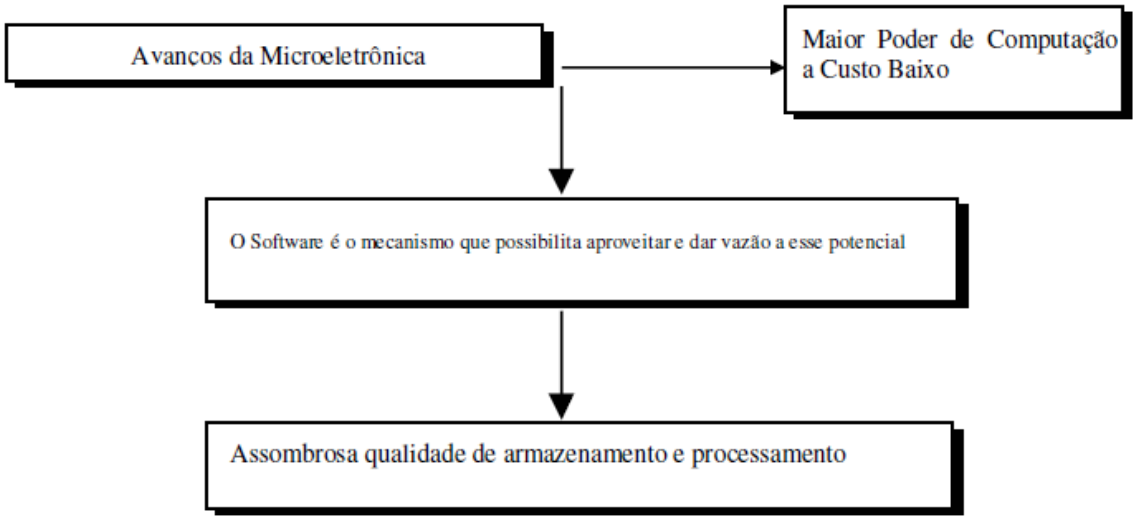

Figura 2 - Desenvolvimento do Software

Diferentes autores trataram nos seus livros sobre a evolução da computação. Osborne caracterizou uma "nova revolução industrial", Toffler chamou o advento da microeletrônica de "a terceira onda de mudança" na história humana e Naisbitt previu que a transformação de uma sociedade industrial em uma "sociedade de informação" terá um profundo impacto sobre nossas vidas. Feigenbaum e McCorduck sugeriram que a informação e o conhecimento (controlados por computador) são o foco principal de poder no século XXI, e Stoll argumentou que a "comunidade eletrônica" criada por redes e software e a chave para a troca de conhecimentos em todo o mundo. No final do século passado (1990), Toffler descreveu uma "mudança de poder", em que as velhas estruturas de poder (governamental, educacional, industrial, econômico e militar) se desintegrarão enquanto os computadores e o software levarão a uma "democratização do conhecimento".
Para fins de avaliação histórica do software, de seus tipos e de seus processos de desenvolvimento, pode-se classificá-los em 4 grandes gerações (Tabela 1). Cada avanço de geração agrega complexidade e aumento de demanda por software.

- 1ª. Geração – 1950 a 1960 – Software para sistemas computacionais de grande porte (Mainframes) com orientação de processamento em lotes (batch), com distribuição limitada apenas a grandes empresas;
- 2ª. Geração – 1960 a 1975 – Software para sistemas de médio porte, com suporte a acessos multiusuários, de tempo real, e bancos de dados. Nesta geração, inicia-se o acesso a softwares por empresas de menor porte;
- 3ª. Geração – 1975 a 1990 – Software para computadores de pequeno porte – microcomputadores – ligados em redes locais. Início da utilização por pessoas físicas e pequenas empresas, o que aumentou grandemente a demanda;
- 4ª. Geração – 1990 a atual – Softwares distribuídos em todas as áreas da sociedade, extremamente poderosos e complexos (sistemas inteligentes, distribuídos, paralelos), rodando em todos os tipos de dispositivos. Chega-se ao pico da complexidade e da demanda.

Tabela 1 - Eras dos sistemas de computador

Primeiros Anos 1950 a 1960	Segunda Era 1960 a 1970	Terceira Era 1970 a 1980	Quarta Era 1980 a 2000
Orientação Batch	Multiusuário Interativo	Sistemas Distribuídos	Sistemas de Desktop poderosos
Distribuição Limitada	Tempo Real	Hardware de Baixo Custo	Tecnologias Orientadas à Objetos
Software Customizado	Banco de Dados	Microprocessadores	Sistemas Especialistas
Programação Artesanal	Produto de Software	Impacto de Consumo	Computação paralela
Sem Administração Específica	Software House	"inteligência" embutida	Ferramentas CASE
Sem Documentação			Reutilização
			Redes Neurais artificiais

Nos anos 1940, quando se iniciou a evolução dos sistemas computadorizados, grande parte dos esforços, e consequentes custos, era concentrada no desenvolvimento do hardware, em razão, principalmente das limitações e dificuldades encontradas na época. Na década de 1950, quando surgiram os primeiros softwares, a ênfase das pesquisas ainda era voltada para o hardware. O hardware estava disponível apenas nos centros de pesquisa e o software desenvolvido sem utilizar técnicas de engenharia.
À medida que a tecnologia de hardware foi sendo dominada, as preocupações nessa década se voltaram para o desenvolvimento dos sistemas operacionais, onde surgiram então as primeiras realizações destes sistemas, assim como das chamadas linguagens de programação de alto nível, como FORTRAN e COBOL, e dos respectivos compiladores.
Na década de 1960, surgiram os microprocessadores e o hardware deixou de representar um problema. Ocorre uma mudança na direção das pesquisas e o software tornou-se o foco dos pesquisadores e organizações começam a desenvolver grandes sistemas (Fig. 3).

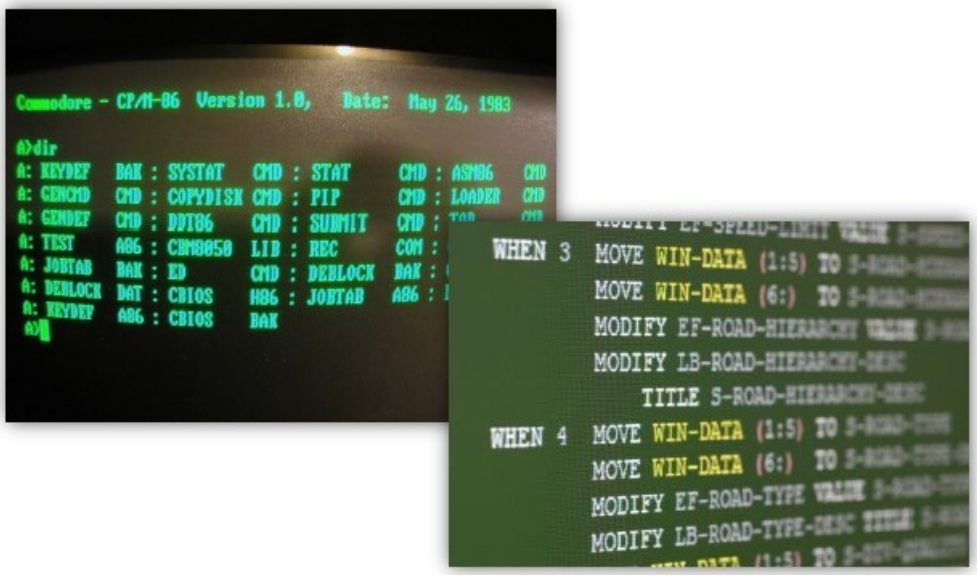

Figura 3 - Desenvolvimento de grandes sistemas

Durante os primeiros anos do desenvolvimento dos sistemas computadorizados, o hardware sofreu contínuas mudanças, enquanto o software era visto por muitos como uma reflexão posterior. A programação de computador era uma arte

"secundaria" para a qual havia poucos métodos sistemáticos. O desenvolvimento do software era feito virtualmente sem administração - até que os prazos começassem a se esgotar e os custos a subir abruptamente. Durante esse período, era usada uma orientação batch (em lote) para a maioria dos sistemas. Na maior parte do tempo, entretanto, o hardware dedicava-se a execução de um único programa que, por sua vez, dedicava-se a uma aplicação específica.
O software, por outro lado, era projetado sob medida para cada aplicação e tinha uma distribuição relativamente limitada. O produto software (isto e, programas desenvolvidos para serem vendidos a um ou mais clientes) estava em sua infância. A maior parte do software era desenvolvida e, em última análise, usada pela própria pessoa ou organização. Ela escrevia-o, colocava-o em funcionamento e, se ele falhasse, era essa pessoa quem o consertava. Uma vez que a rotatividade de empregos era baixa, os gerentes podiam dormir tranquilos com a certeza de que essa pessoa estaria lá se defeitos fossem encontrados. Por causa desse ambiente de software personalizado, o projeto era um processo implícito realizado no cérebro de alguém, e a documentação muitas vezes não existia.
A segunda geração dos sistemas computadorizados estendeu-se de meados da década de 1960 até o final da década de 1970. A multiprogramação e os sistemas multiusuários introduziram novos conceitos de interação homem-máquina. As técnicas interativas abriram um novo mundo de aplicações e novos níveis de sofisticação de software e hardware. Sistemas de tempo real podiam coletar, analisar e transformar dados de múltiplas fontes, controlando processos e produzindo saída em milissegundos, e não em minutos. Os avanços da armazenagem on-line levaram à primeira geração de sistemas de gerenciamento de bancos de dados.
A segunda geração também foi caracterizada pelo uso do produto de software e pelo advento das "software houses". O software era desenvolvido para ampla distribuição num mercado interdisciplinar. Programas para mainframes e minicomputadores eram distribuídos para centenas e, às vezes, milhares de usuários. Empresários da indústria, governos e universidades puseram-se "desenvolver pacotes de software" e a ganhar muito dinheiro.
A terceira geração dos sistemas computadorizados começou em meados da década de 1970 e até o final dos anos 1980. Os sistemas distribuídos - múltiplos computadores, cada um executando funções concorrentemente e comunicando-se um com o outro – aumentaram intensamente a complexidade dos sistemas baseados em computador. As redes globais e locais, as comunicações digitais de largura de banda elevada e a crescente demanda de acesso "instantâneo" a dados exigem muito dos desenvolvedores de software.
A terceira geração também foi caracterizada pelo advento e generalização do uso de microprocessadores, computadores pessoais e poderosas estações de trabalho (workstations) de mesa. O microprocessador gerou um amplo conjunto de produtos inteligentes – de automóveis a fornos de micro-ondas, de robôs industriais a equipamentos médicos.
A quarta geração é marcada por profundas mudanças e as novas tecnologias estão rapidamente ocupando o lugar das abordagens mais convencionais para o desenvolvimento de software em muitas áreas de aplicação. Já estamos numa quinta era, e problemas associados ao software de computador continuam a se intensificar:

- A sofisticação do software ultrapassou nossa capacidade de construir um software que extraia o potencial do hardware;

- Nossa capacidade de construir programas não pode acompanhar o ritmo da demanda de novos programas;
- Nossa capacidade de manter os programas existentes é ameaçada por projetos ruins e recursos inadequados.

A falta de metodologia no desenvolvimento prejudicava a qualidade do produto entregue, pois as equipes de trabalho não tinham um modelo de como desenvolver (Fig.4). Não havia documentação adequada do que estava sendo executado e sempre surgia a pergunta: “E agora: como dar manutenção em um sistema que não tem projeto?”

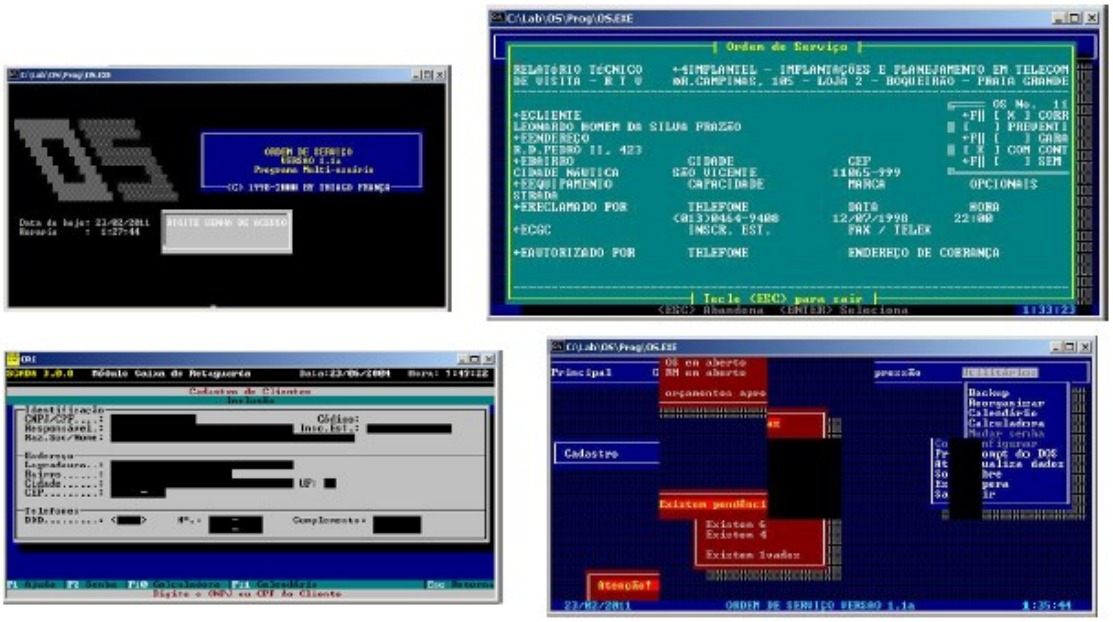

Figura 4 - Falta de metodologia prejudica a qualidade do projeto

3. Definindo Software

Pode-se definir o software, numa forma clássica, como sendo: "um conjunto de instruções que, quando executadas, produzem a função e o desempenho desejados, estruturas de dados que permitam que as informações relativas ao problema a resolver sejam manipuladas adequadamente e a documentação necessária para um melhor entendimento da sua operação e uso".
Entretanto, no contexto da Engenharia de Software, o software deve ser visto como um produto a ser "vendido". É importante dar esta ênfase, diferenciando os "programas" que são concebidos num contexto mais restrito, onde o usuário ou "cliente" é o próprio autor. No caso destes programas, a documentação associada é pequena ou (na maior parte das vezes) inexistente e a preocupação com a existência de erros de execução não é um fator maior, considerando que o principal usuário é o próprio autor do programa, este não terá dificuldades, em princípio, na detecção e correção de um eventual "bug". Além do aspecto da correção, outras boas características não são também objeto de preocupação como a portabilidade, a flexibilidade e a possibilidade de reutilização.
Na Figura 5, temos a exemplificação dos componentes de um software genérico.

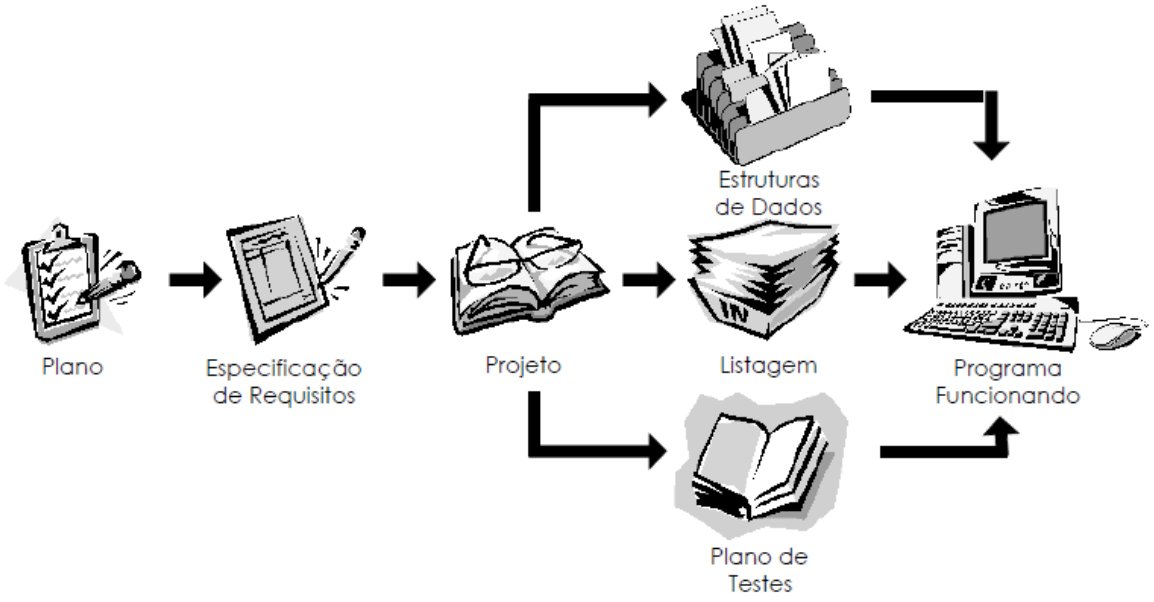

Figura 5 - Componentes do Software

Um software é destinado ao uso por pessoas outras que os seus programadores. Os eventuais usuários podem, ainda, ter formações e experiências diferentes, o que significa que uma grande preocupação no que diz respeito ao desenvolvimento do produto deve ser a sua interface, reforçada com uma documentação rica em informações para que todos os recursos oferecidos possam ser explorados de forma eficiente. Ainda, os produtos de software devem passar normalmente por uma exaustiva bateria de testes, dado que os usuários não estarão interessados (e nem terão capacidade) de detectar e corrigir os eventuais erros de execução.
Resumindo, um programa desenvolvido para resolver um dado problema e um produto de software destinado à resolução do mesmo problema são duas coisas totalmente diferentes. É óbvio que o esforço e o consequente custo associado ao desenvolvimento de um produto serão muito superiores.
Para caracterizar melhor o significado de software é importante levantar algumas particularidades quando comparadas a outros produtos, considerando que o software é um elemento lógico e não físico como a maioria dos produtos de consumo:

- O software é concebido e desenvolvido como resultado de um trabalho de engenharia e não manufaturado no sentido clássico;
- O software não se desgasta, ou seja, ao contrário da maioria dos produtos, o software não se caracteriza por um aumento na possibilidade de falhas à medida que o tempo passa (como acontece com a maioria dos produtos manufaturados);
- A maioria dos produtos de software é concebida inteiramente sob medida, sem a utilização de componentes pré-existentes.

Com o crescimento dos custos de software, em relação aos custos de hardware, no custo total de um sistema computacional, o processo de desenvolvimento de software tornou-se um item de fundamental importância na produção de tais sistemas. Algumas questões que caracterizaram as preocupações com o processo de desenvolvimento de software são:

- Por que o software demora tanto para ser concluído?
- Por que os custos de produção têm sido tão elevados?
- Por que não é possível detectar todos os erros antes que o software seja entregue ao cliente?
- Por que é tão difícil medir o progresso durante o processo de desenvolvimento de software?

Estas são algumas das questões que a Engenharia de Software pode ajudar a resolver. Aqui alguns dos problemas que as originam:

- Raramente, durante o desenvolvimento de um software, é dedicado tempo para coletar dados sobre o processo de desenvolvimento em si;
- Devido à pouca quantidade deste tipo de informação, as tentativas em estimar a duração/custo de produção de um software têm conduzido a resultados bastante insatisfatórios;
- A falta destas informações impede uma avaliação eficiente das técnicas e metodologias empregadas no desenvolvimento;
- A insatisfação do cliente com o sistema "concluído" ocorre frequentemente, devido, principalmente, ao fato de que os projetos de desenvolvimento são baseados em informações vagas sobre as necessidades e desejos do cliente (problema de comunicação entre cliente e fornecedor);
- A qualidade do software é quase sempre suspeita, problema resultante da pouca atenção que foi dada, historicamente, às técnicas de teste de software (o conceito de qualidade de software é algo relativamente recente);
- O software existente é normalmente muito difícil de manter em operação, o que significa que o custo do software acaba sendo incrementado significativamente devido às atividades relacionadas à manutenção; isto é um reflexo da pouca importância dada à manutenibilidade no momento da concepção dos sistemas.

As causas principais dos problemas levantados anteriormente podem ser resumidas em quatro aspectos:

- Falta de experiência dos profissionais na condução de projetos de software;
- Falta de treinamento no uso de técnicas e métodos formais para o desenvolvimento de software;
- “Cultura de programação” que ainda é difundida e facilmente aceita por estudantes e profissionais de Ciências da Computação;
- Resistência às mudanças que os profissionais normalmente apresentam ao uso de novas técnicas de desenvolvimento de software.

4. Crise do Software

A crise do software foi um termo que surgiu nos anos 1970, quando a engenharia de software era praticamente inexistente. O termo expressava as dificuldades do desenvolvimento de software devido ao rápido crescimento da demanda por softwares, a complexidade dos problemas a serem resolvidos e a inexistência de técnicas de desenvolvimento.

Quando o desenvolvimento de softwares começou a utilizar os princípios das linguagens estruturadas e modulares as empresas de softwares começaram a falhar constantemente nos prazos de entrega, apresentando resultados insatisfatórios e os orçamentos ultrapassavam o cronograma predeterminado.
Ainda existem muitos relatos e projetos de software que deram errado e resultaram em falhas de software. Muitas dessas falhas são consequência de dois fatores:

- **Aumento de demanda** - Conforme novas técnicas de desenvolvimento de software nos auxiliam a construir sistemas maiores e mais complexos, as demandas mudam. Os sistemas têm de ser construídos e entregues mais rapidamente. Sistemas maiores e mais complexos são requeridos; sistemas devem ter novas capacidades que antes eram consideradas impossíveis.
- **Expectativas baixas** - É relativamente simples escrever programas computacionais sem usar técnicas e métodos. Muitas empresas foram forcadas a desenvolver softwares à medida que seus produtos e serviços evoluíram. Elas não usam métodos de engenharia de software no dia a dia. Consequentemente, seu software torna-se mais caro e menos confiável do que deveria ser.

As causas da crise do software estão ligadas a complexidade do processo de software e a relativa imaturidade da engenharia de software como profissão:

- As estimativas de prazo e de custo frequentemente são imprecisas;
- Não é dedicado tempo para coletar dados sobre o processo de desenvolvimento de software;
- Com poucos dados históricos como guia as estimativas são empíricas, com resultados previsivelmente ruins;
- A produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços;
- Os projetos de desenvolvimento de software normalmente são efetuados apenas com um vago indício das exigências do cliente;
- A qualidade de software está aquém do mínimo recomendável. No final dos anos 1990 que começam a surgir conceitos quantitativos sólidos de garantia de qualidade de software;
- O software existente é muito difícil de manter. A tarefa de manutenção consome o orçamento destinado ao desenvolvimento do software. A facilidade de manutenção não é enfatizada como um critério importante.

Uma das consequências da crise do software foi o surgimento de vários métodos de desenvolvimento, os quais devido a sua grande quantidade caracterizou-se por se tornar mais um problema frente a inexistência de um método genérico e confiável. Outro aspecto está também na aceitação de novos métodos uma vez que devido a sua imaturidade no mercado eles dificilmente são incorporados devido à pouca confiabilidade que os desenvolvedores tem em relação aos mesmos, assim preferindo usar métodos antigos mas que tem sua eficiente garantida pela experiencia de uso.

5. Desenvolvimento de Software

O processo de desenvolvimento de software suscita algumas importantes questões:

- Software é desenvolvido, não fabricado – Software, diferentemente de outros produtos, como celulares e sapatos, normalmente é desenvolvido sob encomenda para clientes e aplicações específicas. Cada software tende a ser, em menor ou maior grau, diferente dos demais. Isso impede que técnicas de fabricação em escala (industriais) possam ser utilizadas em seu desenvolvimento, que poderiam diminuir custos e prazos;
- Software não desgasta – Software, ao contrário do hardware – em que se pode facilmente perceber o desgaste de peças e engrenagens – não desgasta no sentido literal da palavra. Entretanto, instalações de software "desgastam" ao longo do tempo, e, portanto, necessitam de manutenção e de reparos.

Um exemplo clássico é a instalação de um sistema operacional em um microcomputador. No início, tem-se velocidade e estabilidade, mas, com o passar do tempo (e pela instalação e desinstalação de programas, jogos etc.) a velocidade do sistema cai e os problemas aparecem, como apresentado na Figura 6.

Figura 6 - Desgaste do software

6. Conceitos de Engenharia de Software

Na literatura, pode-se encontrar diversas definições da 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" [NAU 69].

"A aplicação prática do conhecimento científico para o projeto e a construção de programas computacionais e a documentação necessária à sua operação e manutenção." [Boehm, 76]

"Abordagem sistemática para o desenvolvimento, a operação e a manutenção de software" [Afnor, 83]

"Conjunto de métodos, técnicas e ferramentas necessárias à produção de software de qualidade para todas as etapas do ciclo de vida do produto." [Krakowiak, 85]

Sob um ponto de vista formal, a Engenharia de Software pode ser definida como sendo a aplicação da ciência e da matemática através das quais os equipamentos

computacionais são colocados à disposição do homem por meio de programas, procedimentos e documentação associada.
Podemos dizer que a Engenharia de Software busca prover a tecnologia necessária para produzir software de alta qualidade a um baixo custo. Os dois fatores motivadores são essencialmente a qualidade e o custo. A qualidade de um produto de software é um parâmetro cuja quantificação não é trivial, apesar dos esforços desenvolvidos nesta direção. A Engenharia de Software também é multidisciplinar e podemos dizer que une as seguintes áreas:

- Ciências da Computação: abrange arquitetura de computadores, lógica de programação, estrutura de dados algoritmos etc.;
- Administração: o engenheiro de software atua como gestor de um projeto, administrando prazos, equipe, custos, resultados etc.;
- Comunicação: habilidade para saber se expressar com clientes ou usuários;
- Técnicas de solução de problemas: o engenheiro de software deve ser um solucionador de problemas, um gerador de soluções integradas e inteligentes.

Visando melhorar a qualidade dos produtos de software e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de Software, que trata de aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de software. A Engenharia de Software almeja inserir as mesmas sistemáticas existentes em outras áreas da engenharia:

- Custos aceitáveis;
- Gerenciamento do processo de desenvolvimento;
- Garantia do trabalho em equipe e;
- Desenvolvimento de softwares com qualidade.

Alguns princípios devem ser trabalhados na Engenharia de Software como forma de manter o bom funcionamento do produto entregue:

- Evitar dependência de determinadas pessoas ou processos;
- Abstrair aspectos importantes;
- Subdividir problemas complexos;
- Reutilizar resultados (código) e;
- Flexibilização e modularização para facilitar a manutenção.

Os Engenheiros de software devem, dependendo do problema a ser resolvido, das restrições de desenvolvimento e dos recursos disponíveis, adotar uma abordagem sistemática e organizada para seu trabalho, além de usar ferramentas e técnicas apropriadas (Fig. 7).

Figura 7 - Uso de recursos sistemáticos

Assim como em outras áreas, em uma abordagem de engenharia de software, inicialmente o problema a ser tratado deve ser analisado e decomposto em partes menores, em uma abordagem “dividir para conquistar”. Para cada uma dessas partes, uma solução deve ser elaborada. Solucionados os subproblemas isoladamente, é necessário integrar as soluções. Para tal, uma arquitetura deve ser estabelecida. Para apoiar a resolução de problemas, procedimentos (métodos, técnicas, roteiros etc.) devem ser utilizados, bem como ferramentas para parcialmente automatizar o trabalho.

7. Qualidade de Software

O fator qualidade é um dos aspectos importantes que deve ser levado em conta quando do desenvolvimento do software. Para isto, é necessário que se tenha uma definição precisa do que é um software de qualidade ou, pelo menos, quais são as propriedades que devem caracterizar um software desenvolvido segundo os princípios da Engenharia de Software. Portanto, qualidade atualmente não é apenas um diferencial de mercado para a empresa conseguir vender e lucrar mais, é um pré requisito que a empresa deve conquistar para conseguir colocar o produto no Mercado Global. Na área de desenvolvimento de software, há uma urgente necessidade de uma maior preocupação sobre o tema, mas afinal, o que é qualidade?
Primeiramente, é importante discutir o conceito de software de qualidade. Segundo a Associação Francesa de Normalização, AFNOR, a qualidade é definida como "a capacidade de um produto ou serviço de satisfazer às necessidades dos seus usuários". Esta definição, de certa forma, é coerente com as metas da Engenharia de Software, particularmente quando algumas definições são apresentadas. É o caso das definições de Verificação e Validação introduzidas por Boehm, que associa a estas definições as seguintes questões:

- Verificação: "Será que o produto foi construído corretamente?"
- Validação: "Será que este é o produto que o cliente solicitou?"

O problema que surge quando se reflete em termos de qualidade é a dificuldade em se quantificar este fator.

Existem outras definições para qualidade. Algumas pessoas que tentaram uma definição simples chegaram a frases como:

- Qualidade é estar em conformidade com os requisitos dos clientes;
- Qualidade é antecipar e satisfazer os desejos dos clientes;
- Qualidade é escrever tudo o que se deve fazer e fazer tudo o que foi escrito.

Segundo a norma brasileira NBR ISO 8402, qualidade é: “A totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas”.
Nota-se que esta definição formal exige alguns complementos, principalmente para definir o que são as entidades, as necessidades explícitas e as necessidades implícitas.

- A entidade é o produto do qual estamos falando, que pode ser um bem ou um serviço.
- As necessidades explícitas são as próprias condições e objetivos propostos pelo produtor.
- As necessidades implícitas incluem as diferenças entre os usuários, a evolução no tempo, as implicações éticas, as questões de segurança e outras visões subjetivas.

Por exemplo, a qualidade de um prato de comida (a entidade, o produto) está relacionada com a satisfação de necessidades (requisitos) tais como: sabor, aparência, temperatura, rapidez no serviço, preço, higiene, valor nutricional etc. Para avaliar a qualidade de um produto, deve-se fazer uma lista destas necessidades e analisar cada uma destas necessidades.

Então, podemos definir alguns princípios de qualidade:

- Tentar prevenir defeitos ao invés de consertá-los;
- Ter certeza dos defeitos que forem encontrados, serem corrigidos o mais rápido possível;
- Estabelecer e eliminar as causas, bem como os sintomas dos defeitos;
- Auditar o trabalho de acordo com padrões e procedimentos previamente estabelecidos;

E os princípios da Engenharia:

- Analisar o problema antes de desenvolver a solução;
- Quebrar problemas complexos em problemas menores;
- Garantir que subproblemas se unam pelo controle de seus relacionamentos.

Uma vez que um dos objetivos da Engenharia de Software é melhorar a qualidade do software desenvolvido, uma questão deve ser analisada: O que é qualidade de software? Um usuário provavelmente dirá que um software é de boa qualidade se ele satisfizer suas necessidades, sendo fácil de usar, eficiente e confiável.
Essa é uma perspectiva externa de observação pelo uso do produto. Por outro lado, para um desenvolvedor, um produto de boa qualidade tem de ser fácil de manter, sendo o produto de software observado por uma perspectiva interna. Já para um cliente, o software deve agregar valor a seu negócio (qualidade em uso).

Em última instância, podemos perceber que a qualidade é um conceito com múltiplas facetas (perspectivas de usuário, desenvolvedor e cliente) e que envolve diferentes características (por exemplo, usabilidade, confiabilidade, eficiência, manutenibilidade, portabilidade, segurança, produtividade) que devem ser alcançadas em níveis diferentes, dependendo do propósito do software. Por exemplo, um sistema de tráfego aéreo tem de ser muito mais eficiente e confiável do que um editor de textos. Por outro lado, um software educacional a ser usado por crianças deve primar muito mais pela usabilidade do que um sistema de venda de passagens aéreas a ser operado por agentes de turismo especializados.
A qualidade é crítica para a sobrevivência e o sucesso do mercado de software que está se desenvolvendo de forma global. Dentre as principais razões podemos destacar:

- Qualidade de software é competitividade: a única maneira de diferenciar o produto do competidor é pela qualidade do software e do suporte que é fornecido juntamente. Como o mercado amadurece, usuários não querem apenas que a empresa fale que tem qualidade, mas que mostre a todos a sua qualidade através de Certificação internacional. Não ter certificação pode acarretar desvantagem competitiva;
- Qualidade é essencial para a sobrevivência: Clientes estão pedindo por qualidade. Se a empresa não tiver habilidade de sobreviver em um mercado altamente competitivo, ela está em débito com o mercado. A maioria das grandes organizações está reduzindo o número de fornecedores, e um meio de escolher os fornecedores é verificando quais deles têm certificações de qualidade;
- Qualidade é essencial para o mercado internacional: O mercado de software está, cada vez mais, se tornando global. A habilidade das empresas de mostrarem qualidade, eventualmente as colocam no mercado global. O mercado local é vulnerável a produtos importados que, normalmente, têm mais qualidade;
- Qualidade é custo/benefício: um sistema de qualidade direciona para o aumento da produtividade e permanentemente reduz custos, habilitando o gerenciamento para reduzir a correção de defeitos dando ênfase à prevenção. Todas as empresas sabem que corrigir defeitos após o desenvolvimento do software é mais dispendioso do que os corrigir depois. Prevenir defeitos primeiramente pode resolver muita coisa depois e economizar bastante;
- Qualidade retém consumidores e aumenta lucros: pouca qualidade normalmente custa muito mais do que contratar mais desenvolvedores e ainda continuar sem qualidade. A maioria dos consumidores não tolerarão falta de qualidade e irão procurar outros desenvolvedores. Mais qualidade aumenta a satisfação dos consumidores e assegura os que já são clientes a mais tempo.

O processo de desenvolvimento do software é onde os desenvolvedores traduzem os pré-requisitos em software. É correto afirmar que a qualidade do software está diretamente ligada à qualidade dos processos utilizados para o desenvolvimento. Um bom desenvolvimento de software deve capacitar à organização a definição da consistência dos produtos de qualidade.
O ciclo de vida do produto é agora um negócio crítico para muitos desenvolvedores. Os consumidores de software necessitam de produtos cada vez melhores e mais rápidos de serem desenvolvidos para aumentar a sua

competitividade no mercado global. Se estes objetivos forem cumpridos, o desenvolvimento de software deve:

- Utilizar as melhores práticas da engenharia de software;
- Ser operado por pessoal treinado com responsabilidades e instruções;
- Dar ênfase na prevenção de defeitos assim que forem detectados
- Gerar registro para demonstrar efetividade e eficiência;
- Utilizar destes registros para aumentar a performance no futuro.

Questões:

1. 0 que é software?
2. 0 que é engenharia de software?
3. Quais são as principais atividades da engenharia de software?
4. Quais são os principais desafios da engenharia de software?
5. O que foi a crise do software?