Livros da biblioteca:
1) PRESSMAN, R. S. Engenharia de Software. Pearson. Cap. 1 (1.1, 1.2, 1.3 e 1.4)
2) SOMMERVILLE. Engenharia de Software. Pearson. Cap. 1 e 2.

1. Anúncios históricos sobre a engenharia de software

- No inicio da década de 1980: "Software: A Nova Força Propulsora" (Business Week)

- Em meados da década de 1980: "Uma Crescente Defasagem de Software" (Fortune)

- Ao final da década: "Armadilha do Software - Automatizar ou Não?" (Business Week)

- No começo da década de 1990: "Podemos Confiar em Nosso Software?" (Newsweek)

- "Criar Software Novo: Era Uma Tarefa Agonizante..." (Wall Street Journal)

- A nova compreensão da importância do software de computador - as oportunidades que ele oferece e os perigos que apresenta.

- Atualmente, o software ultrapassou o hardware como a chave para o sucesso de muitos sistemas baseados em computador.

- As informações oferecidas pelo software (e bancos de dados relacionados) diferenciam uma empresa de suas concorrentes. A inteligência e a função oferecidas pelo software muitas vezes diferenciam dois produtos de consumo ou industrias idênticas. E o software que pode fazer a diferença.

A crise do software ocorrida na década de 1970 foi causada praticamente pela inexistência de métodos eficazes aplicados no desenvolvimento de projetos e sistemas de software. Os projetos, naquela época, estouravam prazo e orçamento, os softwares eram de baixa qualidade e não atendiam aos requisitos dos usuários. Por fim, não existia nenhuma solução eficaz que resolvesse os problemas enfrentados pelos profissionais de Tecnologia da Informação (TI). Neste contexto, juntamente com o avanço das tecnologias e o aumento da complexidade dos softwares, surgiu a engenharia de software, com o propósito de fornecer ferramentas, métodos, princípios e padrões para o desenvolvimento dos softwares.

Segundo a proposta de definição de engenharia de software realizada por Fritz Bauer na primeira grande conferência dedicada ao assunto, a engenharia de software 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."

De acordo com Pressman, a engenharia de software é um fruto da engenharia de sistemas e de hardware, composta por um conjunto de três elementos fundamentais:

Métodos – é a definição de como será realizado o processo de desenvolvimento de software em termos de planejamentos, previsões para conclusão, requisitos, arquitetura de programação, algoritmo de processamento, codificação, teste e manutenção;

Ferramentas - são utilizadas como apoio nas etapas dos métodos de desenvolvimento. Um conjunto destas ferramentas pode ser utilizado para gerar informações que serão utilizadas por outras ferramentas. Quando esse processo ocorre auxiliado por computador, tem-se a engenharia de software (*Computer-Aided Software Engineering – CASE*);

Procedimentos - é a união de como usar métodos e ferramentas, ou seja, são as etapas escolhidas para execução do projeto, de forma concisa e lógica. Determina-se como será a sequencia em que os métodos serão aplicados, os requisitos de documentação e relatórios, o controle de qualidade, de mudanças e de andamento do projeto, enfim, tudo que envolve o processo de desenvolvimento.

2. Software

2.1.1. Conceitos

- um conjunto de instruções (programas de computador) que, quando executadas, produzem a função e o desempenho desejados;

- um elemento de sistema lógico, e não físico. Portanto, o software tem características que são consideravelmente diferentes das do hardware: o software não se desgasta.

- além de programa, inclui toda documentação associada e os dados de configuração para fazer com que esses programas operem corretamente.

2.1.2. Problemas do software

- Estimativas de prazos e de custos que são frequentemente imprecisos;
- Produtividade dos profissionais da área que não tem acompanhado a demanda por novos serviços;
- A qualidade do software desenvolvido que é insuficiente.

2.1.3. Evolução do software

Período	Evolução
1950 – 1960	Orientação a Batch
	Software totalmente customizados
	Distribuição limitada
1960 – 1970	Multiusuários
	Tempo Real
	Banco de Dados
	Produto de Software
1980 – 1990	Sistemas distribuídos
	Inteligência embutida
	Hardware de baixo custo
	Impacto de consumo
1990 – 2000	Sistemas de desktop poderosos
	Tecnologia orientada a objeto
	Sistemas Especialistas
	Redes neurais artificiais
	Computação Paralela

Quadro 1 – Evolução do software(fonte Pressmam, pg.5).

2.1.4. Aplicação/Tipos dos softwares

2.1.4.1) Software de Sistema: coleção de programas escritos para servir outros programas.
- Características: interação intensa com hardware do computador; uso intenso por múltiplos usuários; compartilhamento de Recursos. Ex: Compiladores, editores e componentes de sistemas operacionais

2.1.4.2) Software Comercial: processamento de Informações comercial. Aplicações dessa área reestruturam os dados existentes de modo a facilitar operações comerciais ou tomada de decisão de gestão de negócios . Ex: Folha de pagamentos, contas a pagar/receber , controle estoque e outros.

2.1.4.3) Software Científico e de engenharia: esse tipo de software é caracterizado por processar números e também simulação de sistemas. Ex: Astronomia, biologia molecular

2.1.4.4) Software para Inteligência Artificial: faz uso de algoritmos não numéricos para resolver problemas complexos que não são passiveis de computação ou analise direta. Ex: Sistemas Especialistas, também chamados sistemas baseados em conhecimento, de reconhecimento de padrões (de imagem e de voz)

2.1.4.5) Software de Tempo Real – são programas que monitoram, analisam e controlam eventos do mundo real, devendo responder aos estímulos do mundo externo com restrições de tempo pré-determinadas.

2.1.4.6) Software Embutido – são desenvolvidos para executar atividades muito específicas e inseridos em produtos inteligentes tanto para atividades comerciais como para atividades domesticas.

2.1.4.7) Outros
- E-commerce (B2B, B2C, B2M, e outros)

- Aplicações móveis (iPhone, iPad, Tablets em geral, Android)

- Jogos e entretenimento

- Robótica

- Sistemas de Hipermídia (Exemplo: mídias ricas)

- Redes sociais

3. Engenharia de software

A engenharia de software surgiu da necessidade de se construir software com mais qualidade em menor tempo, antigamente se produzia software de uma maneira muito desordenada sem preocupação com o que realmente o software deveria fazer ou se erra possível construir um software para executar tal tarefa, com isto surgiu à famosa crise do software que fez com que as empresas ou fábricas de software pensassem em uma maneira de como desenvolver os softwares de maneira confiável e rápida. O software teve uma grande evolução no decorrer de sua existência ocasionada principalmente pelo barateamento do hardware e a evolução das técnicas de desenvolvimento

3.1. Conceitos

"O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que funcione eficientemente cm máquinas reais." (Fritz Bauer)

É um conjunto integrado de métodos e ferramentas utilizadas para especificar, projetar, implementar e manter um sistema.

“Uma disciplina que reúne metodologias, métodos e ferramentas a Engenharia de Software ser utilizados, desde a percepção do problema até o momento em que o sistema desenvolvido deixa de ser operacional, visando resolve problemas inerentes ao processo de desenvolvimento e ao produto de software.” (Ariadne Carvalho & Thelma Chiossi no livro Introdução à Computação).

O estabelecimento e uso de princípios de engenharia para a produção economicamente viável de software de qualidade que funcione em máquinas reais.

A engenharia de software é a disciplina envolvida com a produção e manutenção sistemática de software que são desenvolvidos com custos e prazos estimados.

Disciplina que aborda a construção de software complexo - com muitas partes interconectadas e diferentes versões - por uma equipe de analistas, projetistas, programadores, gerentes, "testadores", etc.

Disciplina que utiliza um conjunto de métodos, técnicas e ferramentas para analisar, projetar e gerenciar desenvolvimento e manutenção de software (é abstrato e intangível).

3.2. Fundamentos

Engenharia de software relaciona-se à teorias, métodos e ferramentas para o desenvolvimento profissional de software. Ela é uma derivação da engenharia de sistemas e de hardware. Ela abrange um conjunto de três elementos fundamentais - métodos, ferramentas e procedimentos - que possibilita ao gerente o controle do processo de desenvolvimento do software e oferece ao profissional uma base para a construção de software de alta qualidade produtivamente.

Os métodos de engenharia de software proporcionam os detalhes de "como fazer" para construir o software. Os métodos envolvem um amplo conjunto de tarefas que incluem: planejamento c 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. Os métodos da engenharia de software muitas vezes introduzem uma notação gráfica ou orientada a linguagem especial e introduzem um conjunto de critérios para a qualidade do software.

As ferramentas de engenharia de software proporcionam apoio automatizado ou semiautomatizado aos métodos. Atualmente, existem ferramentas para sustentar cada um dos métodos anotados anteriormente. 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 chamado engenharia de software auxiliada por computador (CASE - Computer-Aided Software Engineering). O CASE combina software, hardware e um banco de dados de engenharia de software (uma estrutura de dados contendo importantes informações sobre analise, projeto, codificação e teste) para criar um ambiente de engenharia de software que seja análogo ao projeto auxiliado por computador/engenharia auxiliada por computador para o hardware.

Os procedimentos da engenharia de software 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 seqü.ncia em que os métodos serão aplicados, os produtos (deliverables) que se exige que sejam entregues (documentos, relatórios, formulários etc.), os controles que ajudam a assegurar a qualidade e a coordenar as mudanças, e os marcos de referencia que possibilitam aos gerentes de software avaliar o progresso.

Essas etapas - métodos, ferramentas e procedimentos - muitas vezes são citadas como paradigmas da engenharia de software. Um paradigma de engenharia de software e escolhido tendo-se como base a natureza do projeto e da aplicação, os métodos e as ferramentas a serem usados, os controles e os produtos que precisam ser entregues.

3.3. Objetivos

- Melhorar a qualidade do software e aumentar a produtividade e satisfação profissional de engenheiros de software.

- Aplicar a teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para o **desenvolvimento** sistemático de software.

- Produzir **documentação** formal destinada a comunicação entre os membros da equipe de desenvolvimento bem como aos usuários.

3.4. Características

- Custos com software geralmente são os mais altos em sistemas de computação. O custo de um software é geralmente maior que o custo do hardware.

- Gasta-se mais para manter o software do que para desenvolvê-lo. Nos sistemas de longa vida útil, os custos de manutenção podem ser várias vezes maiores que os custos de seu desenvolvimento.

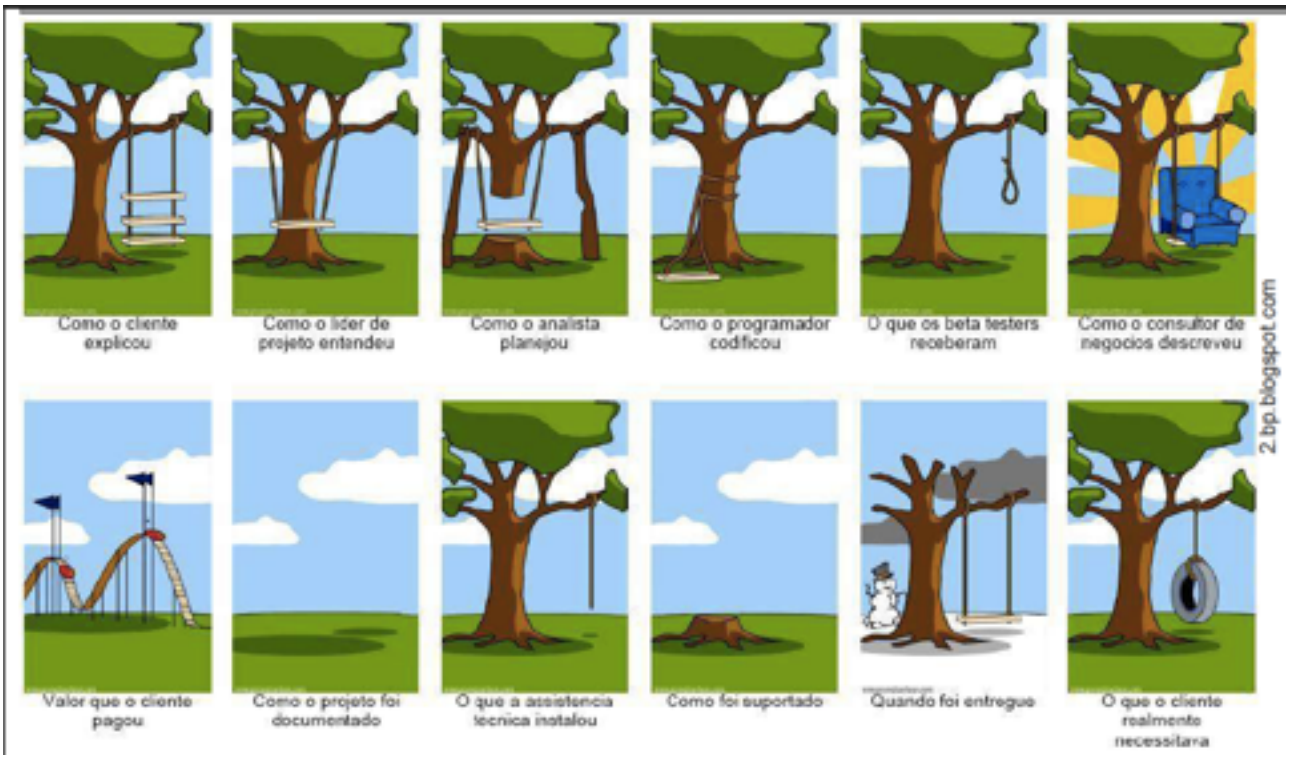
- Fatores que devem ser considerar no desenvolvimento/evolução de software do "mundo real" : escopo; custo; prazo e qualidade.

- Dependendo do tamanho do software, tais fatores se tornam mais difíceis de garantir.

3.5. Engenharia de Sistemas x Engenharia de Software

- A engenharia de sistemas se ocupa de todos os aspectos relacionados ao desenvolvimento de sistemas com base em computadores, incluindo hardware, software e engenharia de processos;
- A engenharia de Software é parte desse processo.

3.6. Diálogo para desenvolvimento de um software!!!!!



4. Perfil do Analista de sistemas

_Capacidade de gerir a alocação dos recursos humanos e materiais diante da disponibilidade financeira e de tempo.

• Capacidade de especificar sistemas de informação, utilizando ferramentas de modelagem, conhecimento da linguagem de implementação, do banco de dados, da arquitetura de sistemas e de redes.

• Capacidade de aprender.

Um analista de sistemas, além de saber construir modelos, deve ser conhecedor ou aprofundar-se no que está modelando, seja um sistema de matrícula, vendas, controle de estoque, bancário, etc. Durante a modelagem o analista muitas vezes se torna um especialista na área.

Um analista de sistemas deve ter:
- habilidade com pessoas
- conhecimento de aplicações
- habilidade em tecnologia
- mente lógica e organizada.

5. Desenvolvimento de software

"Pode ser definido como um conjunto de atividades, métodos, práticas e transformações que pessoas utilizam para desenvolver ou dar manutenção em softwares ou seus produtos associados (projetos, manuais, código, etc.)".
(SEI-Carnegie Mellon University. The Capability Maturity Model: guidelines for improving the software process.Addison Wesley. 1995 p.8)

5.1. Solução de problemas complexos

Decompor o problema em subproblemas de menor complexidade, desde que seja possível reconstituir o todo. **Ex**: Sistema venda - dividir em subprocessos:
- Venda a vista (dinheiro, cheque, vale, cartão débito);
- Venda a prazo (cartão crédito, convênio, crediário), etc.

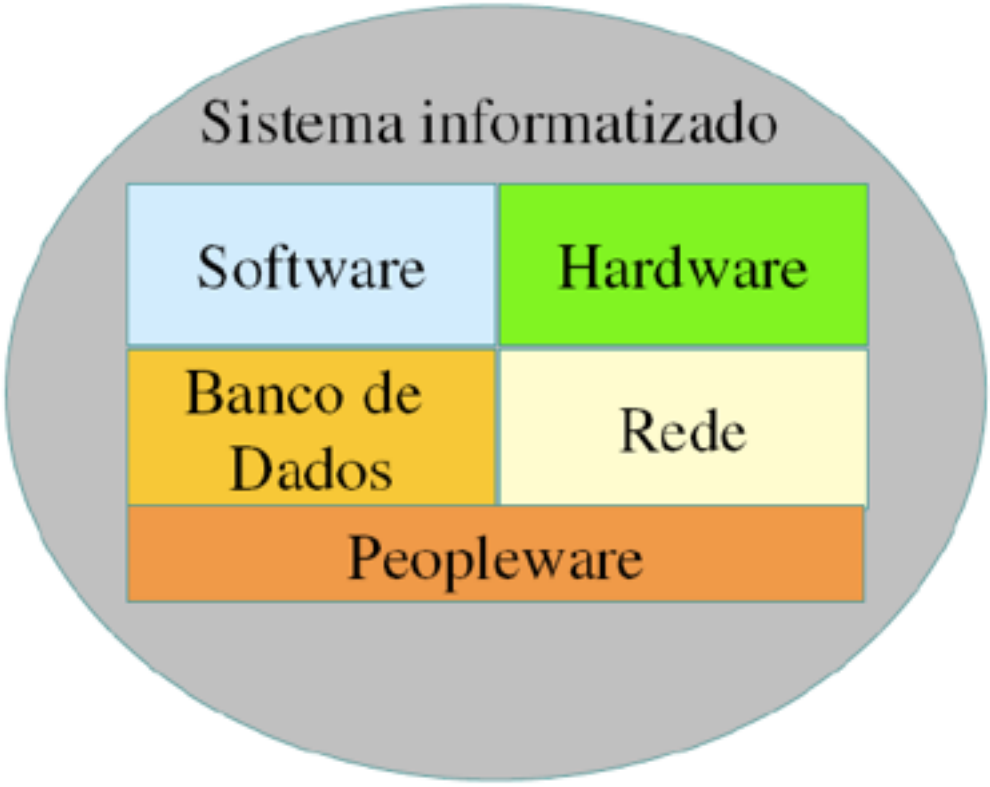
2º - Decompor o problema por pontos de vista diferentes. **Ex**: Sistema venda - dividir em foco de análise:
- Fluxo do processo, interface com os usuários;
- Integração com sistemas afins, etc.

5.2. Sistema informatizado

"É um mecanismo composto por um conjunto de partes inter-relacionadas"

"Disposição das partes ou dos elementos de um todo coordenado entre si e que funcionam como estrutura organizada".

- Componentes de um sistema de informação: hardware, software, pessoas, dados e procedimentos.



5.3. Ciclo de vida de um software (FALBO, R. de A. Engenharia de Software: notas de aula. Universidade Federal do Espírito Santo - EFES. 2005)

"*Planejamento*: O objetivo do planejamento de projeto é fornecer uma estrutura que possibilite ao gerente fazer estimativas razoáveis de recursos, custos e prazos. Uma vez estabelecido o escopo de software, uma proposta de desenvolvimento deve ser elaborada, isto é, um plano de projeto deve ser elaborado configurando o processo a ser utilizado no desenvolvimento de software. À medida que o projeto progride, o planejamento deve ser detalhado e atualizado regularmente. Pelo menos ao final de cada uma das fases do desenvolvimento (análise e especificação de requisitos, projeto, implementação e testes), o planejamento como um todo deve ser revisto e o planejamento da etapa seguinte deve ser detalhado. O planejamento e o acompanhamento do progresso fazem parte do processo de gerência de projeto. "

" *Análise e Especificação de Requisitos*: Nesta fase, o processo de levantamento de requisitos é intensificado. O escopo deve ser refinado e os requisitos identificados. Para entender a natureza do software a ser construído, o engenheiro de software tem de compreender o domínio do problema, bem como a funcionalidade e o comportamento esperados. Uma vez identificados os requisitos do sistema a ser desenvolvido, estes devem ser modelados, avaliados e documentados. Uma parte vital desta fase é a construção de um modelo descrevendo *o que* o software tem de fazer (e não *como* fazê-lo). "

“*Projeto*: Esta fase é responsável por incorporar requisitos tecnológicos aos requisitos essenciais do sistema, modelados na fase anterior e, portanto, requer que a plataforma de implementação seja conhecida. Basicamente, envolve duas grandes etapas: projeto da arquitetura do sistema e projeto detalhado. O objetivo da primeira etapa é definir a arquitetura geral do software, tendo por base o modelo construído na fase de análise de requisitos. Esta arquitetura deve descrever a estrutura de nível mais alto da aplicação e identificar seus principais componentes. O propósito do projeto detalhado é detalhar o projeto do software para cada componente identificado na etapa anterior. Os componentes de software devem ser sucessivamente refinados em níveis de maior detalhamento, até que possam ser codificados e testados. “

“*Implementação*: O projeto deve ser traduzido para uma forma passível de execução pela máquina. A fase de implementação realiza esta tarefa, isto é, cada unidade de software do projeto detalhado é implementada. “

“ *Testes*: inclui diversos níveis de testes, a saber, teste de unidade, teste de integração e teste de sistema. Inicialmente, cada unidade de software implementada deve ser testada e os resultados documentados. A seguir, os diversos componentes devem ser integrados sucessivamente até se obter o sistema. Finalmente, o sistema como um todo deve ser testado. “

“ *Entrega e Implantação*: uma vez testado, o software deve ser colocado em produção. Para tal, contudo, é necessário treinar os usuários, configurar o ambiente de produção e, muitas vezes, converter bases de dados. O propósito desta fase é estabelecer que o software satisfaz os requisitos dos usuários. Isto é feito instalando o software e conduzindo testes de aceitação (validação). Quando o software tiver demonstrado prover as capacidades requeridas, ele pode ser aceito e a operação iniciada. “

“*Operação*: nesta fase, o software é utilizado pelos usuários no ambiente de produção. “

“ *Manutenção*: Indubitavelmente, o software sofrerá mudanças após ter sido entregue para o usuário. Alterações ocorrerão porque erros foram encontrados, porque o software precisa ser adaptado para acomodar mudanças em seu ambiente externo, ou porque o cliente necessita de funcionalidade adicional ou aumento de desempenho. Muitas vezes, dependendo do tipo e porte da manutenção necessária, essa fase pode requerer a definição de um novo processo, onde cada uma das fases precedentes é re-aplicada no contexto de um software existente ao invés de um novo“.