UNIDADE 1
O PRODUTO E O PROCESSO DE SOFTWARE

Professora: Denise Franzotti Togneri

Email: denise.franzotti@faesa.br
YouTube: Denise Togneri
Facebook: profdenisetogneri
Instagram: profdenisetogneri

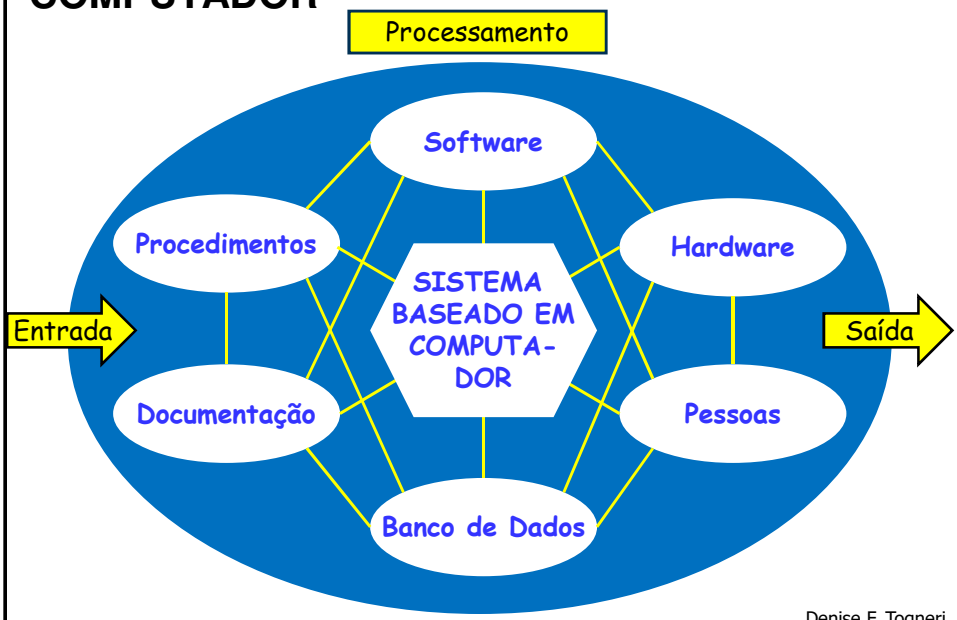
Sumário

- Sistema baseado em computador
- Software ou produto de software
- Campos de aplicação ou tipos de software
- Software legado
- Engenharia de Software
- Projeto
- Processo de software e suas atividades
- A essência da prática da Engenharia de Software
- Os sete princípios gerais da Engenharia de Software
- Mitos do desenvolvimento de software
- Equipes de Software

SISTEMA BASEADO EM COMPUTADOR

- é um conjunto de **elementos** organizados, inter-relacionados, que possuem características comuns, para atingir alguma meta ou objetivo pré-definido por meio do processamento da informação (PRESSMAN, 2006, p. 100).

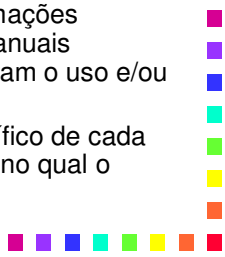
ELEMENTOS DE UM SISTEMA BASEADO EM COMPUTADOR



ELEMENTOS DE UM SISTEMA BASEADO EM COMPUTADOR

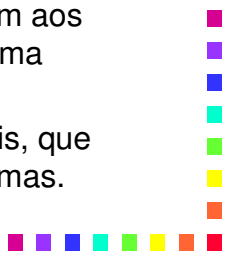
- **Software:** composto de programas de computador, estruturas de dados e produtos de trabalho correlacionados que servem para realizar o método lógico, procedimento ou controle necessário;
- **Hardware**;
- **Pessoas:** usuários e operadores de hardware e software;
- **Bancos de dados:** um coleção grande e organizada de informações que é acessada por intermédio do software e persiste ao longo do tempo;
- **Documentação:** artefatos gerados ao longo do processo de software e que devem ser mantidos atualizados, ou seja, informações descritivas (por exemplo, modelos, especificações, manuais impressos, arquivos de ajuda on-line, sites) que mostram o uso e/ou operação do sistema;
- **Procedimentos:** os passos que definem o uso específico de cada elemento do sistema ou o contexto de procedimentos no qual o sistema reside.

Esses elementos se combinam de diversos modos para transformar a informação (PRESSMAN, 2006, p. 100).



SOFTWARE OU PRODUTO DE SOFTWARE

- O software ou produto de software é o componente lógico de um sistema baseado em computador e é composto de (PRESSMAN; MAXIM, 2016, p. 4):
  - (1) as **instruções** (os programas de computador) que quando executadas fornecem as características, função e desempenho desejados;
  - (2) as **estruturas de dados** que permitem aos programas manipular informações de forma adequada;
  - (3) os **documentos,** impressos ou virtuais, que descrevem a operação e uso dos programas.



Piadinha nerd!

Software é aquilo que você xinga.

Hardware é aquilo que você chuta.

Café com Código

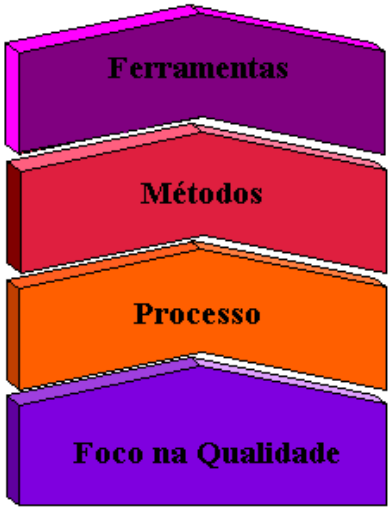
ENGENHARIA DE SOFTWARE

- A **Engenharia de Software** é a criação e a utilização de sólidos princípios de engenharia a fim de se obter softwares econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais (PRESSMAN; MAXIM, 2016, p. 15).

- A **Engenharia de Software (IEEE, 1993):**
  - (1) a aplicação de uma abordagem sistemática, disciplinada e quantificável, para o desenvolvimento, operação e manutenção de software; isto é, a aplicação da engenharia ao software;
  - (2) o estudo de abordagens como as de (1).

CAMADAS DA ENGENHARIA DE SOFTWARE

- A Engenharia de Software é uma tecnologia em camadas.



→ CASE, LPs, Editores de texto.

→ Como fazer?

→ Une as fases. Marcos do Projeto

→ Melhoria contínua

CAMADAS DA ENGENHARIA DE SOFTWARE

- **Foco na Qualidade:** A Eng. Sw. deve se apoiar num compromisso organizacional com a qualidade. Gestão da Qualidade Total, Seis Sigma e filosofias análogas levam à cultura de um processo contínuo de aperfeiçoamento, permitindo o desenvolvimento de abordagens cada vez mais efetivas para a Eng. Sw.

- **Processo:** O alicerce da Eng. Sw. é a camada de processo, que é o adesivo que mantém unidas as camadas de tecnologia e permite o desenvolvimento racional e oportuno de softwares de computador. O processo define um arcabouço que deve ser estabelecido para a efetiva utilização da tecnologia de Eng. Sw. Os processos de software formam a base para o controle gerencial de projetos de software e estabelecem o contexto no qual os métodos técnicos são aplicados, os produtos de trabalho são produzidos, os marcos são estabelecidos, a qualidade é assegurada e as modificações são adequadamente geridas;

CAMADAS DA ENGENHARIA DE SOFTWARE

- **Métodos:** fornecem a técnica de "como fazer" para construir softwares. Abrangem um amplo conjunto de tarefas que incluem: comunicação, análise de requisitos, modelagem de projeto, construção de programas, testes e manutenção;
- **Ferramentas:** fornecem apoio automatizado ou semi-automatizado para o processo e para os métodos. Quando ferramentas são integradas de modo que a informação criada por uma ferramenta possa ser usada por outra, é estabelecido um sistema de apoio ao desenvolvimento de software chamado CASE - *Computer-Aided Software Engineering* (PRESSMAN, 2006, p.17).

O QUE É UM PROJETO?

- É um conjunto de atividades **temporárias**, realizadas em grupo, destinadas a produzir um produto, serviço ou resultado **únicos** (PMI, 2018):
  - é **temporário** no sentido de que tem um início e fim definidos no tempo, e, por isso, um escopo e recursos definidos;
  - é **único** no sentido de que não se trata de uma operação de rotina, mas um conjunto específico de operações destinadas a atingir um objetivo em particular. Assim, uma equipe de projeto, muitas vezes, inclui pessoas que geralmente não trabalham juntas – algumas vezes vindas de diferentes organizações e de múltiplas geografias.

O QUE É UM PROJETO?

- **Exemplos:**
  - O desenvolvimento de um software para um processo empresarial
  - a construção de um prédio ou de uma ponte
  - o esforço de socorro depois de um desastre natural
  - a expansão das vendas em um novo mercado geográfico.

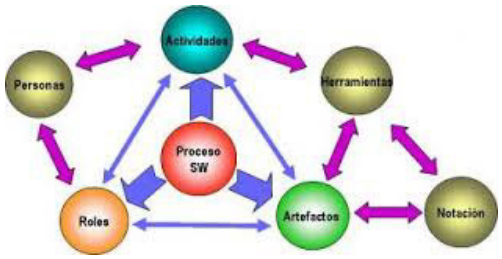
- Todos devem ser gerenciados de forma especializada para apresentarem os resultados, aprendizado e integração necessários para as organizações dentro do prazo e do orçamento previstos.

- Para gerenciar de forma correta um projeto é necessário, dentre outros, definir um processo de trabalho a ser executado, que no caso da Engenharia de Software é denominado **processo de software.**

Denise F. Togneri

O PROCESSO DE SOFTWARE

- É o conjunto de **atividades, métodos, técnicas** e **ferramentas** necessárias para o **desenvolvimento** e **manutenção** de um **produto de software** (PRESSMAN; MAXIM, 2016, p. 16).



Denise F. Togneri

O PROCESSO DE SOFTWARE

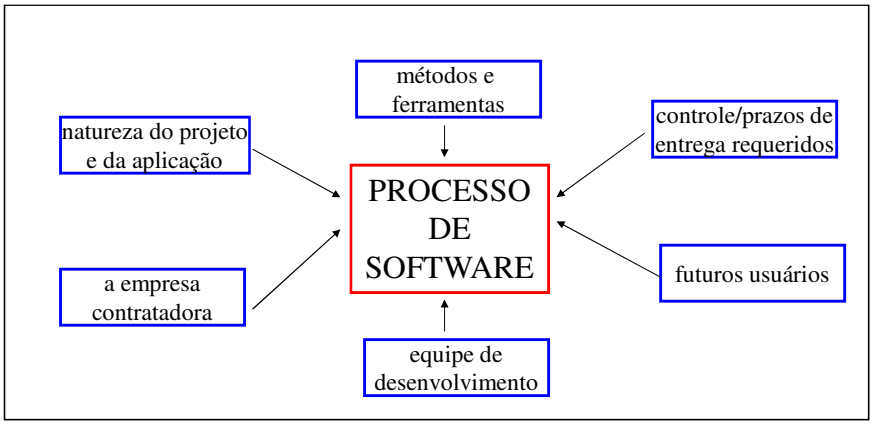
- As atividades a executar são a base do processo, mas também devem ser definidos:
  - recursos
  - detalhamento
  - precedência das atividades
  - métodos e técnicas utilizadas para a construção do software
  - ferramentas computacionais para o processo e métodos.

O PROCESSO DE SOFTWARE

- Não é um prescrição rígida de como desenvolver um software.
- Pelo contrário, ele deve ser **ágil e adaptável** para possibilitar à equipe de software selecionar um conjunto apropriado de atividades, de forma a entregar o software dentro do prazo e do custo acordados, e com qualidade (PRESSMAN; MAXIM, 2016, p. 18).
- **O processo de software é a Engenharia de Software aplicada a um projeto específico.**

O PROCESSO DE SOFTWARE

- Devem, então, ser definidos casos a caso para cada projeto, com o objetivo de atingir o produto, considerando-se



ATIVIDADES DO PROCESSO DE SOFTWARE

Um processo de software é composto por **atividades principais** (*framework activities*) que são aplicáveis a todos os projetos de software, independente de seu tamanho ou complexidade e por **atividades de apoio** (*umbrella activities*) que revestem as atividades principais (PRESSMAN; MAXIM, 2016, p. 16).

Atividades principais

- Comunicação
- Planejamento
- Modelagem
- Construção
- Implantação ou Entrega

Atividades de apoio

- Acompanhamento e controle de projeto de software
- Gestão de risco
- Garantia de qualidade de software
- Revisões técnicas formais
- Medição
- Gestão de configuração de software
- Gestão de reusabilidade
- Preparação e produção do produto do trabalho.

ATIVIDADES DO PROCESSO DE SOFTWARE

- São classificadas em (PRESSMAN; MAXIM, 2016, p. 17):

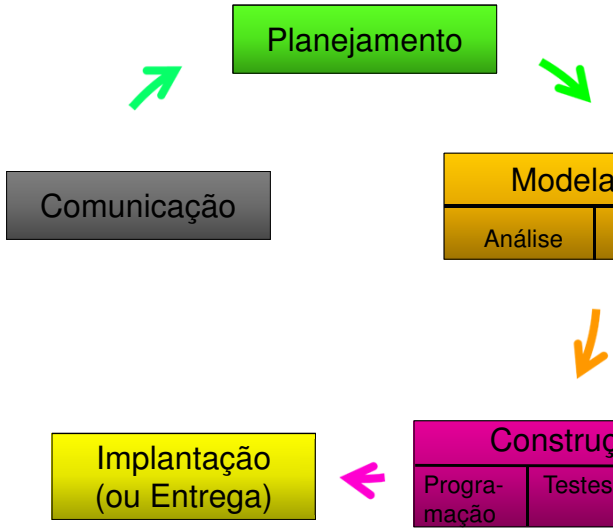
ATIVIDADES PRINCIPAIS

ou *framework activities,* que são aplicáveis a todos os projetos de software, independente de seu tamanho ou complexidade.

ATIVIDADES DE APOIO

ou *umbrella activities*, que complementam as atividades principais e ajudam uma equipe de software a gerenciar e a controlar o andamento, a qualidade, as alterações e os riscos de um projeto.

ATIVIDADES PRINCIPAIS DO PROCESSO DE SOFTWARE (PRESSMAN; MAXIM, 2016, p. 17)



ATIVIDADES PRINCIPAIS DO PROCESSO DE SOFTWARE

COMUNICAÇÃO:

- Envolve muita comunicação e colaboração com o cliente (e outros interessados - *stakeholders*) para
  - Levantar todas as áreas usuárias, entender os objetivos a serem atingidos com o projeto, os problemas a serem resolvidos, as necessidades a serem atendidas, o processo de negócio a ser automatizado;
  - levantar e documentar os requisitos, de forma a ajudar a definir as funções do software e os recursos (*hardware, software e peopleware*) necessários;
  - Gerar um Documento de Requisitos de "nível macro".

ATIVIDADES PRINCIPAIS DO PROCESSO DE SOFTWARE

PLANEJAMENTO:

- Inicia-se com a aprovação do escopo do software a ser construído, usando como base para a tomada de decisão, o Documento de Requisitos gerado na Comunicação e um Estudo de Viabilidade do projeto.
- Gera um **Plano de Projeto de Software** que define o trabalho de Engenharia de Software a ser realizado, contendo o processo de software que será adotado, as atividades que serão conduzidas, os riscos prováveis, o Plano de Gestão da Qualidade, os recursos que serão necessários, o custo do projeto, os artefatos (produtos de trabalho) a serem produzidos em cada etapa e um cronograma do trabalho, dentre outros;

MODELAGEM:

- Inclui a criação de modelos que permitam ao desenvolvedor e ao cliente entender melhor os requisitos do software e o projeto que vai satisfazer a esses requisitos. É composta de duas ações de Engenharia de Software:
  - **Análise** - modela **o que** o software deve atender e
  - **Projeto** - modela **como** construir o software.

ATIVIDADES PRINCIPAIS DO PROCESSO DE SOFTWARE

CONSTRUÇÃO:

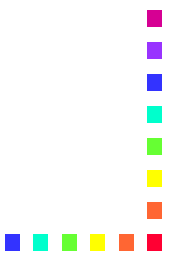
- Engloba a:
  - **Programação** - geração de código (quer manual ou automática);
  - **Testes** - necessários para **encontrar falhas** no código;
  - **Depuração** (*debugging*) - é o processo de procurar, analisar e **remover as causas de falhas** no software.

IMPLANTAÇÃO OU ENTREGA:

- O software é entregue ao cliente (é implantado no "ambiente de produção"), que avalia o produto entregue e fornece feedback com base na avaliação.

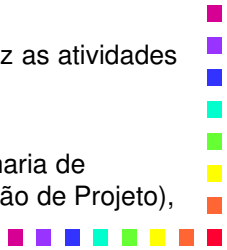
ATIVIDADES DE APOIO TÍPICAS DO PROCESSO DE SW (PRESSMAN; MAXIM, 2016, p. 18)

- Controle e Acompanhamento de Projeto
- Gestão de Riscos
- Garantia da Qualidade de Software
- Revisões Técnicas
- Medição
- Gestão da Configuração de Software
- Gestão de reuso
- Preparo e produção de artefatos de software



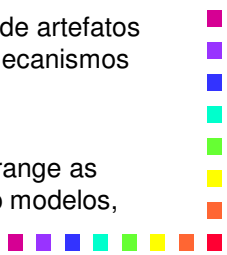
ATIVIDADES DE APOIO DO PROCESSO DE SOFTWARE

- **Acompanhamento e controle de projeto de software.** Permite à equipe de software avaliar o progresso em relação ao plano de projeto e tomar as ações necessárias para cumprir o cronograma.
- **Gestão de risco.** Identifica e avalia os riscos que podem afetar o resultado do projeto ou a qualidade do software, bem como propõe atividades de mitigação, monitoramento e administração dos riscos, ou seja, consiste de sub-atividades que ajudam a entender e administrar a incerteza.
- **Garantia de qualidade de software.** Define e conduz as atividades necessárias para garantir a qualidade do software.
- **Revisões técnicas.** Avaliam os artefatos da Engenharia de Software (ex. Documento de Requisitos, Especificação de Projeto), tentando identificar e remover erros antes que se propaguem para a atividade seguinte.

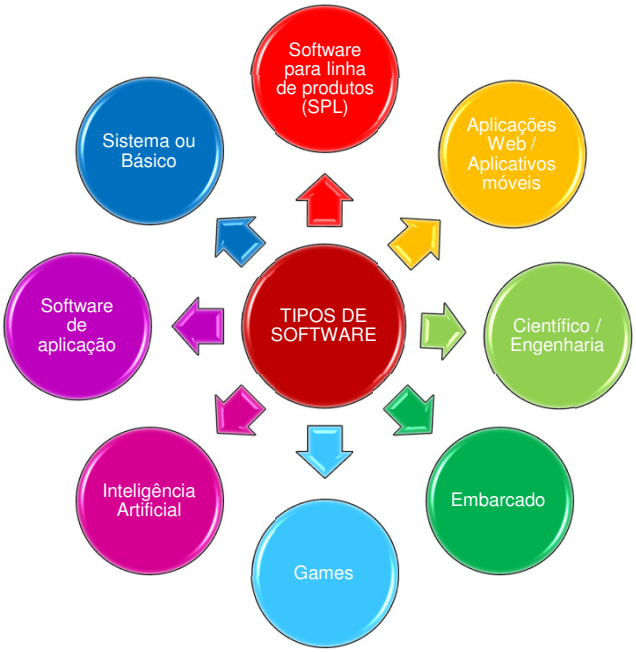


ATIVIDADES DE APOIO DO PROCESSO DE SOFTWARE

- **Medição.** Define e coleta medidas de processo, projeto e produto que ajudam a equipe a implantar um software que satisfaça às necessidades do usuário e que atenda aos requisitos definidos; pode ser usada conjugada com todas as outras atividades principais e de apoio.
- **Gestão de configuração de software.** Gerencia os efeitos das mudanças ao longo de todo o processo de software.
- **Gestão de reuso.** Define critérios pra a reutilização de artefatos (inclusive componentes de software) e estabelece mecanismos para a obtenção de componentes reusáveis;
- **Preparo e produção de artefatos de software.** Abrange as atividades necessárias para criar artefatos tais como modelos, atas, documentos, logs, formulários e listas.



CAMPOS DE APLICAÇÃO OU TIPOS DE SW



CAMPOS DE APLICAÇÃO OU TIPOS DE SW

Alguns campos de aplicação de software são (PRESSMAN; MAXIM, 2016, p. 6):

- **Sistema ou Básico:** feitos para atender outros sistemas (ex. sistemas operacionais, compiladores, gerenciadores de bancos de dados, drivers, editores, software de rede, etc);

- **Software para linha de produtos de software (SPL – *software product lines*):** projetado para fornecer capacidade específica de utilização por muitos clientes diferentes, ou seja, técnica de produção baseada em outras engenharias – fábricas que desenvolvem uma mesma família de produtos com partes e recursos comuns. Consiste em um conjunto de sistemas de software que:
  - Têm uma funcionalidade comum
  - São construídos de uma forma prescrita visando uma missão específica ou segmento de mercado
  - São desenvolvidos utilizando componentes e recursos (ativos) de uma base comum
  - Fornecem uma substancial economia de produção de software e é aplicável em grupos de sistemas similares.

CAMPOS DE APLICAÇÃO OU TIPOS DE SW

Alguns campos de aplicação de software são (PRESSMAN; MAXIM, 2016, p. 6):

- **Aplicações Web/aplicativos móveis:** categoria de software voltada às redes que abrange uma ampla variedade de aplicações, contemplando aplicativos voltados para navegadores e software residente em dispositivos móveis;
- **Científico/Engenharia**: baseado fortemente em processamento numérico (ex. softwares para astronomia, vulcanologia, meteorologia, análise genética, biologia molecular);
- **Embarcado**: são usados para controlar produtos e sistemas para os mercados industriais e de consumo (ex. controle de teclado para forno de microondas, controle de combustível, sistemas de freio, etc).
- **Games**
- **Inteligência Artificial:** Sistemas Especialistas, Sistemas baseados em conhecimento, redes neurais, processamento de linguagem natural;

SOFTWARE LEGADO

- **Foram desenvolvidos décadas atrás e têm sido continuamente modificados para se adequar às mudanças dos requisitos de negócio e a plataformas computacionais.**
- A proliferação de tais sistemas está causando dores de cabeça para grandes organizações que os consideram dispendiosos de manter e arriscados de evoluir.
- **Muitos sistemas legados permanecem dando suporte para funções de negócio vitais e são indispensáveis para o mesmo. Por isso, um software legado é caracterizado pela longevidade e criticidade de negócios** (PRESSMAN; MAXIM, 2016, p. 8).

SOFTWARE LEGADO

SOFTWARE LEGADO

- Alguns sistemas legados, apesar de darem suporte a funções vitais de negócio e serem indispensáveis para ele, podem ser classificados como de **baixa qualidade** (se forem julgados em termos da Engenharia de Software moderna).

- Alguns problemas são (PRESSMAN; MAXIM, 2016, p. 8):
  - Projetos inextensíveis
  - Código de difícil entendimento
  - Documentação deficiente ou inexistente
  - Casos de teste e resultados que nunca foram documentados
  - Histórico de alterações mal gerenciado

- **O QUE FAZER?**

SOFTWARE LEGADO

- TALVEZ A ÚNICA RESPOSTA ADEQUADA SEJA: NÃO FAÇA NADA, pelo menos até que o sistema legado tenha que passar por alguma modificação significativa.
- Se ele atende às necessidades de negócio e funciona de maneira confiável, ele não precisa ser consertado!

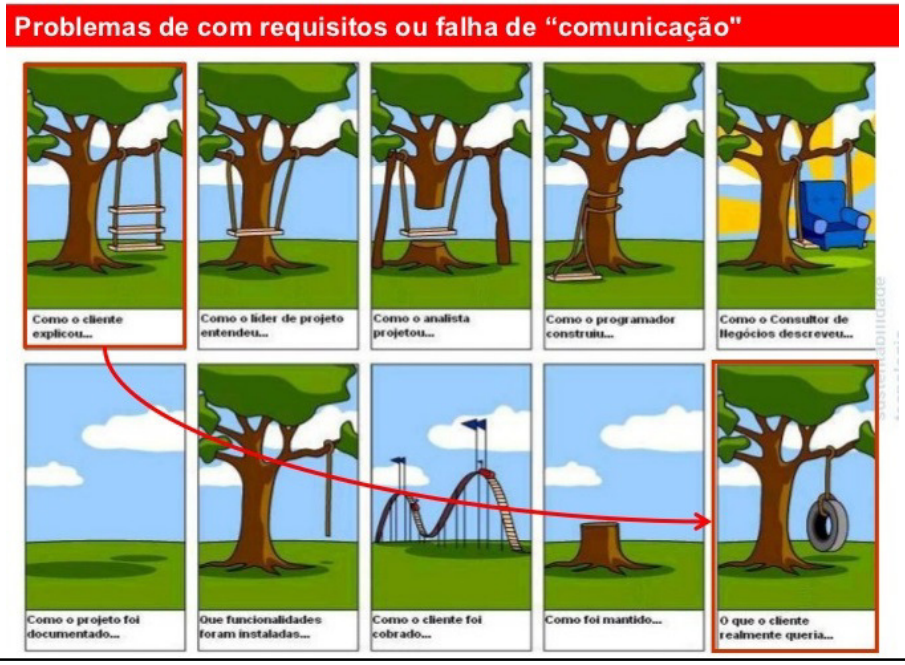
SOFTWARE LEGADO

- Com o passar do tempo, no entanto, esses sistemas evoluem devido a uma ou mais das seguintes razões (PRESSMAN; MAXIM, 2016, p. 8):
  - O software deve ser adaptado para atender às necessidades de novos ambientes ou de novas tecnologias computacionais;
  - O software deve ser aperfeiçoado para implementar novos requisitos de negócio
  - o software deve ser expandido para torná-lo capaz de funcionar com outros bancos de dados ou com sistemas mais modernos.
  - O software deve ser rearquitetado para torná-lo mais viável dentro de um ambiente computacional em evolução.

SOFTWARE LEGADO – REENGENHARIA DE SOFTWARE

- Quando essas modalidades de evolução ocorrem, um sistema legado deve passar por **reengenharia de software** cujas principais subatividades são (PRESSMAN; MAXIM, 2016, p. 803):
  - Análise de inventário (tamanho, idade, criticalidade nos negócios)
  - Reestruturação dos documentos
  - Engenharia Reversa – para recuperar os modelos do projeto. Ex: modelo relacional de Banco de Dados, projeto da arquitetura e dos componentes, gerados a partir do software já existente
  - Reestruturação de código na mesma linguagem de programação ou em outra mais moderna
  - Reestruturação de dados
  - Engenharia direta – ferramentas que recuperam as informações do projeto de software existente e as utilizam para alterar ou reconstituir o sistema, de forma a para melhorar a qualidade geral.

Problemas no desenvolvimento de software



A ESSÊNCIA DA PRÁTICA DA ENGENHARIA DE SOFTWARE

Em linhas gerais, a essência da solução de problemas e, consequentemente, a essência da prática da Engenharia de Software consiste em (PRESSMAN; MAXIM, 2016, p. 19):

- **Compreender o problema** (comunicação e análise)
- **Planejar uma solução** (modelagem e projeto de software)
- **Executar o plano** (geração do código)
- **Examinar o resultado para ter precisão** (testes e garantia da qualidade).

A ESSÊNCIA DA PRÁTICA DA ENGENHARIA DE SOFTWARE

- **Compreender o problema** (comunicação e análise)
  - *Quem tem interesse na solução do problema?*
  - *Quais são as incógnitas? Que dados, funções e recursos são necessários para resolver o problema?*
  - *O problema pode ser compartimentalizado?*
  - *O problema pode ser representado graficamente?*

- **Planejar uma solução** (modelagem e projeto de software)
  - *Você já viu problemas semelhantes anteriormente?*
  - *Algum problema semelhante já foi resolvido?*
  - *É possível definir subproblemas?*
  - *É possível representar uma solução de maneira que conduza a uma implementação efetiva?*

A ESSÊNCIA DA PRÁTICA DA ENGENHARIA DE SOFTWARE

- **Executar o plano** (geração do código)
  - *A solução é adequada ao plano? O código-fonte pode ser atribuído ao modelo de projeto?*
  - *Todas as partes componentes da solução estão provavelmente corretas? O projeto e o código foram revistos, ou melhor, provas da correção foram aplicadas ao algoritmo?*
- **Examinar o resultado para ter precisão** (testes e garantia da qualidade).
  - *É possível testar cada parte componente da solução? Foi implementada uma estratégia de testes razoável?*
  - *A solução produz resultados adequados aos dados, às funções e às características necessários?*

OS SETE PRINCÍPIOS GERAIS DA ENGENHARIA DE SOFTWARE

São eles (PRESSMAN; MAXIM, 2016, p. 21):

1 – A razão de existir → agregar valor para seus usuários!

2 – KISS (Keep it simple, Stupid!) → não complique!

3 – Mantenha a visão → respeite a visão arquitetural projetada para o sistema

4 – O que um produz outros consomem → ao especificar, projetar ou implementar pense que outro terá que entender o que você está fazendo

5 – esteja aberto para o futuro → adaptabilidade a mudanças

6 – Planeje com antecedência, visando o reuso

7 – Pense! → Pensar bem e de forma clara antes de agir quase sempre produz melhores resultados.

MITOS DO DESENVOLVIMENTO DE SOFTWARE

- São **crenças infundadas** sobre o software e o processo de software e remontam aos primórdios da computação.

- Podem ser (PRESSMAN; MAXIM, 2016, p. 23):
  - **Do gerenciamento**
  - **Dos clientes**
  - **Dos profissionais da área**

- **Conclusão:** o problema não está no software (**produto**) e sim em como ele é desenvolvido (**processo**).

MITOS DE GERENCIAMENTO

- **MITO:** Já temos um livro cheio de padrões e procedimentos para desenvolver software. Ele não supriria meu pessoal com tudo que precisam saber?

  - **REALIDADE:** O livro com padrões pode existir, mas ele é usado? Os praticantes da área estão cientes de que ele existe? Esse livro reflete a prática moderna da Eng.ª de software? É completo? É adaptável? Está alinhado para melhorar o tempo de entrega, mantendo ainda o foco na qualidade?
    - Em muitos casos, a resposta para todas essas perguntas é não!

MITOS DE GERENCIAMENTO

- **MITO:** Se um cronograma atrasar, poderemos acrescentar mais programadores e ficar em dia (algumas vezes denominado conceito da “horda mongol”).
  - **REALIDADE:** O desenvolvimento de software não é um processo mecânico como o de fabricação. Acrescentar pessoas num projeto de software atrasado só o tornará mais atrasado ainda, pois quando novas pessoas entram, as que já estavam terão de gastar tempo situando os recém-chegados, reduzindo, consequentemente, o tempo destinado ao desenvolvimento produtivo. Pode-se adicionar pessoas, mas somente de forma planejada e bem coordenada.

MITOS DE GERENCIAMENTO

- **MITO:** Se eu decidir terceirizar o projeto de software, posso simplesmente relaxar e deixar a outra empresa realizá-lo.
  - **REALIDADE:** Se uma organização não souber gerenciar e controlar projetos de software, ela irá, invariavelmente enfrentar dificuldades ao terceirizá-los.

MITOS DOS CLIENTES

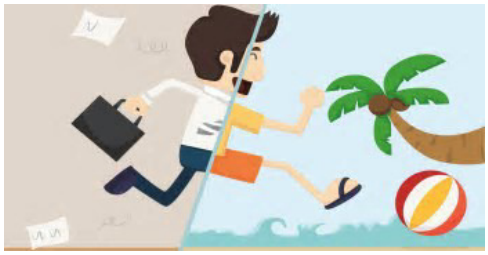
- **MITO:** Uma definição geral dos objetivos é suficiente para começar a escrever os programas – podemos preencher os detalhes mais tarde.
  - **REALIDADE:** Embora nem sempre seja possível uma definição ampla e estável dos requisitos, uma definição de objetivos ambígua é receita para um desastre.
    - Requisitos não ambíguos (normalmente derivados da iteratividade) são obtidos somente pela comunicação contínua e eficaz entre cliente e desenvolvedor.

MITOS DOS CLIENTES

- **MITO:** Os requisitos de software mudam continuamente, mas as mudanças podem ser facilmente assimiladas, pois o software é flexível.
  - **REALIDADE:** É verdade que os requisitos de software mudam, mas o impacto da mudança varia dependendo do momento em que ela foi introduzida.
    - Quando as mudanças dos requisitos são solicitadas cedo (antes do projeto ou da codificação terem começado), o impacto sobre os custos é relativamente baixo.
    - Entretanto, conforme o tempo passa, ele aumenta rapidamente - recursos foram comprometidos, uma estrutura de projeto foi estabelecida e mudar pode causar recursos adicionais e modificações fundamentais no projeto.

MITOS DOS PROFISSIONAIS DA ÁREA

- **MITO:** Uma vez que o programa foi feito e colocado em uso, nosso trabalho está terminado.
  - **REALIDADE:** Uma vez alguém já disse que "o quanto antes se começar a codificar, mais tempo levará para terminá-lo".
    - Levantamentos indicam que entre 60 e 80% de esforço será despendido após a entrega do software ao cliente pela primeira vez.



MITOS DOS PROFISSIONAIS DA ÁREA

- **MITO:** Até que o programa esteja "em execução", não há como avaliar sua qualidade.
  - **REALIDADE:** Um dos mecanismos de garantia da qualidade de software pode ser aplicado desde a concepção de um projeto – a revisão técnica.
    - Revisores de software são um "filtro de qualidade" que mostram ser mais eficientes do que testes para encontrar certas classes de defeitos de software.

MITOS DOS PROFISSIONAIS DA ÁREA

- **MITO:** O único produto passível de entrega é o programa em funcionamento
  - **REALIDADE:** Um programa funcionando é somente uma parte de uma configuração de software que inclui muitos elementos.
    - Uma variedade de produtos derivados (exemplo, modelos, documentos, planos) constitui uma base para uma engenharia bem sucedida e, mais importante, uma orientação para suporte de software.

MITOS DOS PROFISSIONAIS DA ÁREA

- **MITO:** A Engenharia de Software nos fará criar documentação volumosa e desnecessária e, invariavelmente, vai nos retardar.

  - **REALIDADE:** A Engenharia de software não trata de criação de documentos, trata da criação de um produto de qualidade.
    - Melhor qualidade conduz à redução de retrabalho e, menos retrabalho resulta em maior rapidez na entrega.

EQUIPES DE SOFTWARE

- **Uma equipe consistente é um grupo de pessoas tão coesas, que o todo é maior que a soma das partes.** Em uma equipe consistente, a probabilidade de sucesso aumenta e não é preciso gerenciá-la do modo tradicional (PRESSMAN; MAXIM, 2016, p. 90).

- Os membros de equipes consistentes são
  - **mais produtivos** e **mais motivados** do que a média
  - compartilham de um **objetivo** e de uma **cultura comum** e
  - possuem um **senso de pertencimento** a uma equipe de elite que os torna únicos.

CARACTERÍSTICAS DAS EQUIPES DE SOFTWARE EFICIENTES

Uma equipe de software eficiente deve (PRESSMAN; MAXIM, 2016, p. 90):

- estabelecer um **senso de propósito** → objetivo comum
- incorporar um **senso de envolvimento** → os membros sentem que suas qualidades e contribuições são valiosas
- promover um **senso de confiança** → nas habilidades e na competência dos colegas e do gerente
- estimular um **senso de melhoria** → buscando sempre maneiras de melhorar a forma de trabalhar.

CINCO FATORES DAS EQUIPES TÓXICAS

Os cinco fatores que promovem um ambiente potencialmente tóxico nas equipes são (PRESSMAN; MAXIM, 2016, p. 91):

- Uma atmosfera de trabalho frenética
- Alto grau de frustração que causa atrito entre os membros da equipe
- Um processo de software fragmentado ou coordenado de forma deficiente
- Uma definição nebulosa dos papéis dentro da equipe de software
- Contínua e repetida exposição a falhas.

REFERÊNCIAS

FALBO, Ricardo de A. **Automatização do Processo de Desenvolvimento de Software**. In: ROCHA, Ana R. C. da. (Org.). Qualidade de Software: seleção de textos. Curitiba: CITS, 1996. p. 71-88.

_____. **Notas de aula da disciplina Engenharia de Software -** Mestrado em Informática. UFES, 1998.

_____. **Desenvolvimento orientado a objetos**: notas de aula. Disponível em: <http://www.inf.ufes.br/~falbo>. Acesso em: 04 ago. 2000.

IEEE Standards Collection: Software Engineering, IEEE Standard 610.12-1990, IEEE, 1993.

MCCONNELL, Steve. **Rapid Development**. Washington: Microsoft Press, 1996.

PAULK, Mark C., WEBER Charles V., CURTIS, Bill et al. (Org.). **The Capability Maturity Model** - Guidelines for Improving the Software Process. Reading, Massachusetts: Addison-Wesley, 1998.

PMI. **O que é gerenciamento de projetos?** Disponível em: <https://brasil.pmi.org/brazil/AboutUS/WhatIsProjectManagement.aspx>. Acesso em: 06 fev. 2018.

PRESSMAN, R. S; **Engenharia de Software**. 6.ed. Rio de Janeiro: McGraw-Hill, 2006.

PRESSMAN, R. S; MAXIM, B.R. **Engenharia de Software**. 8.ed. Rio de Janeiro: McGraw-Hill, 2016.