Módulo I - Introdução, Sistemas e Aplicações

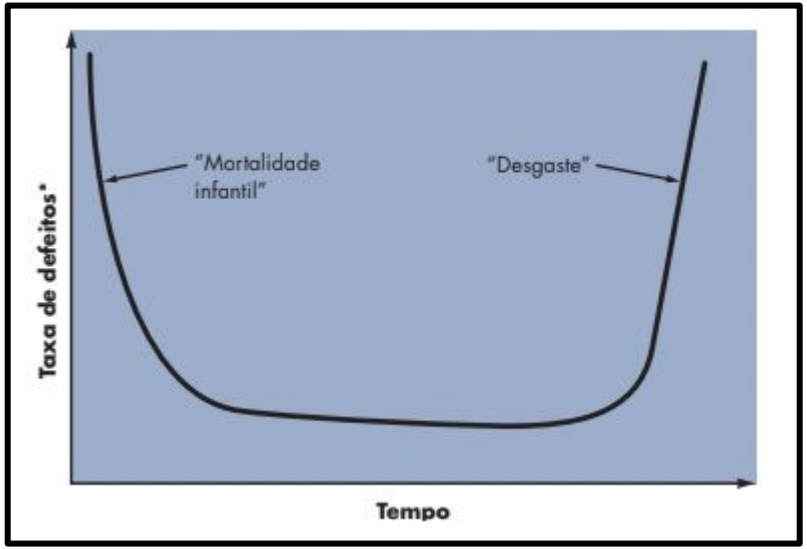
Tópico 8 - Engenharia de Software e boas práticas

.StartSe

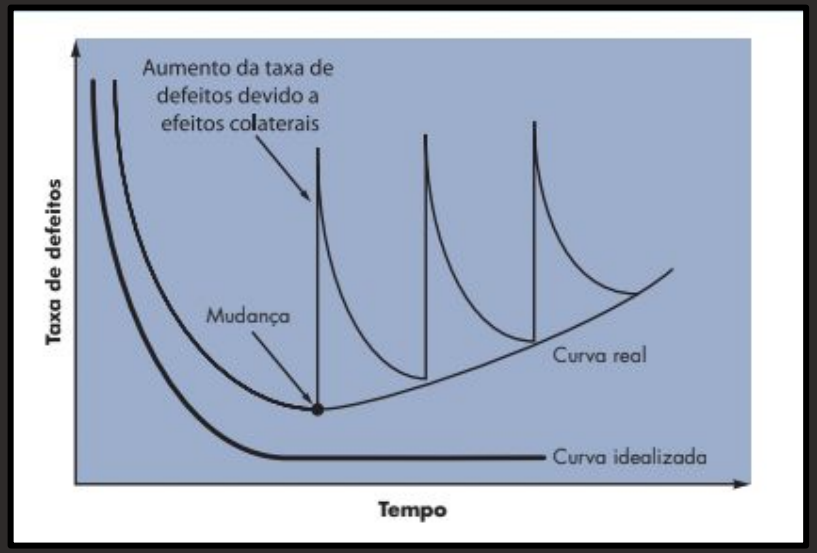
Como funciona o desenvolvimento de software?

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

Embora existam algumas similaridades entre o desenvolvimento de software e a fabricação de hardware, as duas atividades são fundamentalmente diferentes. Ambas requerem a construção de um "produto", entretanto, as abordagens são diferentes. 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.



Hardware (e muitos outros produtos convencionais)



Software

Software não se desgasta, ele se deteriora.

Software não é suscetível aos males ambientais que fazem com que o hardware se desgaste. Portanto, teoricamente, a curva da taxa de defeitos para software deveria assumir a forma da "curva idealizada". Defeitos ainda não descobertos irão resultar em altas taxas logo no início da vida de um programa. Entretanto, esses serão corrigidos e a curva se achata como mostrado. A curva idealizada é uma simplificação grosseira de modelos de defeitos reais para software. Porém, a implicação é clara: software não se desgasta, mas sim se deteriora!

Quais são os desafios no desenvolvimento de software?

- Devemos fazer um **esforço concentrado para compreender o problema antes de desenvolver uma solução de software**;

- Dado o aumento da complexidade do software em todos âmbitos, **estar constantemente atualizado** e a par das novas tecnologias é fundamental. Pelo mesmo motivo, **projetar tornou-se uma atividade-chave**;

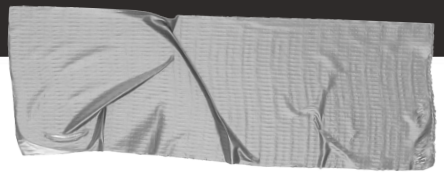
- Em um mundo cada vez mais dependente de software, **um software deve apresentar qualidade elevada**;

- À medida que sua base de usuários e seu tempo em uso forem aumentando, a demanda por adaptação e aperfeiçoamento também irá aumentar. Portanto, um **software deve ser passível de manutenção**.

Dessa forma, software, em todas as suas formas e em todos os seus campos de aplicação, deve passar pelos processos de engenharia.

O que é engenharia de software?

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**. Bem como o estudo de abordagens.



O que é engenharia de software?

A engenharia de software é uma tecnologia em camadas. A pedra fundamental que sustenta a engenharia de software é o foco na qualidade. 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.

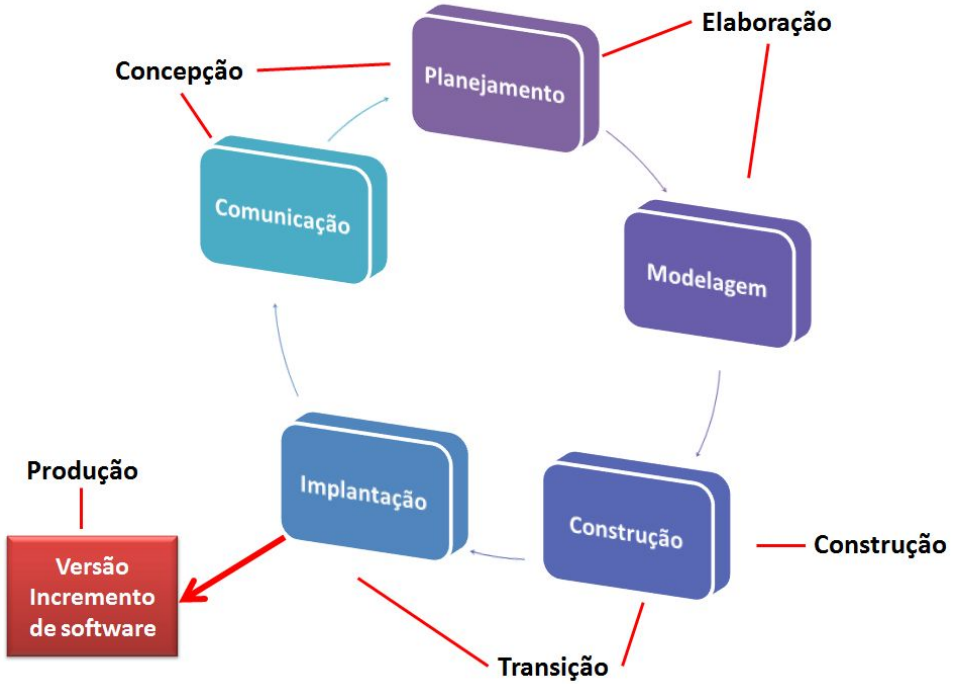
Os **métodos** da engenharia de software fornecem as informações técnicas para desenvolver software: comunicação, análise de requisitos, modelagem de projeto, construção de programa, testes e suporte.

As **ferramentas** da engenharia de software fornecem suporte automatizado ou semi-automatizado para o processo e para os métodos.

E o que é o processo de software?

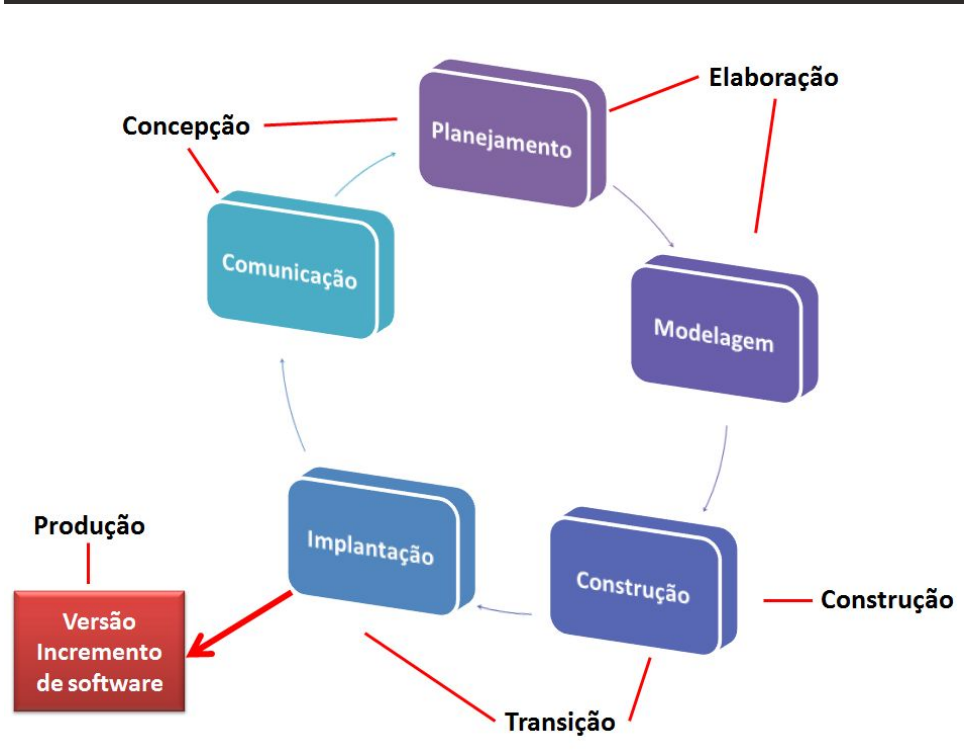
As cinco atividades genéricas do processo de software:

- **Comunicação**: levantamento das necessidades que ajudarão a definir as funções e características do software.
- **Planejamento:** define o trabalho de engenharia de software, descrevendo as tarefas técnicas a ser conduzidas, os riscos prováveis, os recursos que serão necessários, os produtos resultantes a ser produzidos e um cronograma de trabalho.
- **Modelagem:** Cria-se um "esboço" da coisa, de modo que se possa ter uma ideia do todo — qual será o seu aspecto em termos de arquitetura, como as partes constituintes se encaixarão e várias outras características.
- **Construção:** Essa atividade combina geração de código e testes necessários para revelar erros na codificação.
- **Emprego (Implantação):** O software é entregue ao cliente, que avalia o produto entregue e fornece feedback, baseado na avaliação.



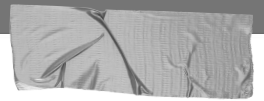
A essência da prática:

- ➔ **Compreenda o problema;**
- ➔ **Planeje a solução;**
- ➔ **Execute/leve adiante o plano;**
- ➔ **Examine o resultado.**



Os 7 princípios da engenharia de software

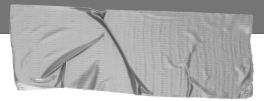
1 - A razão de existir!



Um sistema de software existe por uma única razão: gerar valor a seus usuários. Todas as decisões deveriam ser tomadas tendo esse princípio em mente. Antes de especificar uma necessidade de um sistema, antes de indicar alguma parte da funcionalidade de um sistema, antes de determinar as plataformas de hardware ou os processos de desenvolvimento, pergunte a si mesmo: “Isso realmente agrega valor real ao sistema?”. Se a resposta for “não”, não o faça.

Todos os demais princípios se apoiam neste primeiro.

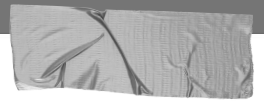
2 - KISS (Keep It Simple Stupid!)



Esse princípio contribui para um sistema mais fácil de compreender e manter. Isso **não** significa que características, até mesmo as internas, devam ser descartadas em nome da simplicidade.

De fato, frequentemente os projetos mais elegantes são os mais simples, o que não significa "rápido e malfeito" — na realidade, simplificar exige muita análise e trabalho durante as iterações, sendo que o resultado será um software de fácil manutenção e menos propenso a erros.

3 - Mantenha a visão

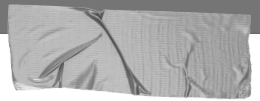


Uma visão clara é essencial para o sucesso. Sem ela, um projeto se torna ambíguo. Sem uma integridade conceitual, corre-se o risco de transformar o projeto numa colcha de retalhos de projetos incompatíveis, unidos por parafusos inadequados... Comprometer a visão arquitetônica de um sistema de software debilita e até poderá destruir sistemas bem projetados. Ter um arquiteto responsável e capaz de manter a visão clara e de reforçar a adequação ajuda a assegurar o êxito de um projeto.

4 - O que um produz os outros consomem

Sempre especifique, projete e implemente ciente de que alguém mais terá de entender o que você está fazendo. O público para qualquer produto de desenvolvimento de software é potencialmente grande. Especifique tendo em vista os usuários; projete, tendo em mente os implementadores; e codifique considerando aqueles que terão de manter e estender o sistema. Alguém terá de depurar o código que você escreveu e isso o faz um usuário de seu código; facilitando o trabalho de todas essas pessoas você agrega maior valor ao sistema.

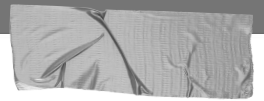
5 - Esteja aberto para o futuro



Nos ambientes computacionais de hoje, em que as especificações mudam de um instante para outro e as plataformas de hardware se tornam rapidamente obsoletas, a vida de um software, em geral, é medida em meses, mas precisamos de produtos mais duráveis que isso!

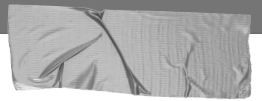
Jamais faça projetos limitados, sempre pergunte “e se” e prepare-se para todas as possíveis respostas, criando sistemas que resolvam o problema geral, não apenas aquele específico. Isso muito provavelmente conduziria à reutilização de um sistema inteiro.

6 - Planeje com antecedência, visando a reutilização



Planejar com antecedência para o reuso reduz o custo e aumenta o valor tanto dos componentes reutilizáveis quanto dos sistemas aos quais eles serão incorporados.

7 - Pense!



Este último princípio é, provavelmente, aquele que é mais menosprezado. **Pensar bem e de forma clara antes de agir quase sempre produz melhores resultados.** Quando se analisa alguma coisa, provavelmente esta sairá correta. Ganha-se também conhecimento de como fazer correto novamente. Se você realmente analisar algo e mesmo assim o fizer da forma errada, isso se tornará uma valiosa experiência. Um efeito colateral da análise é aprender a reconhecer quando não se sabe algo, e até que ponto poderá buscar o conhecimento.

O que são boas práticas?

A ideia central por trás do conceito de boas práticas é: **não se faz código para máquinas, mas sim para pessoas!**

Boas práticas são toda ação no ambiente de desenvolvimento que corrobora para garantir os sete princípios que mencionamos: **que visam a durabilidade, facilidade de manutenção, clareza e compreensão do código, bem como a postura de documentar e compartilhar conhecimento sobre o projeto e tecnologias**.

Livros de Robert C. Martin, o Uncle Bob