

# SOFTWARES COMERCIAIS

A Engenharia de Software tem por objetivo apoiar o desenvolvimento profissional de software, visto que a maior parte do DESENVOLVIMENTO DE SOFTWARE É **UMA ATIVIDADE PROFISSIONAL**, mesmo que existam desenvolvedores amadores (pessoas que programam e apenas elas usam o programa).

A Engenharia de Software vai além do software em si, porque para o desenvolvimento de uma aplicação, é necessário estudar diversas técnicas, especializações e documentações para se certificar que o sistema junto com a entrada de dados e as configurações dos aparelhos irá rodar corretamente. (isso tudo na maior parte das vezes não é preciso para o desenvolvimento de um software amador.

Nessa área o **SOFTWARE** é tratado como **PRODUTO**, sendo assim o **PRODUTO DE SOFTWARE** pode ser dividido em:

**PRODUTOS GENÉRICOS:** Dentro dessa classe, existem os: **STAND-ALONE**, são softwares que são produzidos por uma empresa de desenvolvimento e outras organizações (clientes) compram se quiser, como **SISTEMAS OPERACIONAIS, FERRAMENTAS DE EDIÇÃO DE TEXTO E CRIAÇÃO DE PROJETOS (MESMO QUE OUTRAS EMPRESAS COMPREM O PRODUTO, O CONTROLE DA ESPECIFICAÇÃO DELES FICA COM A EMPRESA QUE DESENVOLVEU). E AS APLICAÇÕES VERTICAIS**, que são softwares destinados para alguma área que vai precisar de gestão, como a **ORGANIZAÇÃO DENTRO DE BIBLIOTECAS OU ESCRITÓRIOS DE CONTABILIDADE**.

**PRODUTOS SOB ENCOMENDA:** Nessa classe, um **cliente X** vai **especificar** o que ele quer **para** o **desenvolvedor** e esse vai projetar uma aplicação **ESPECÍFICA** para **ESTE CLIENTE X**. Exemplos: **PROGRAMA PARA CONTROLAR UM DISPOSITIVO ELETRÔNICO, SISTEMA PARA AJUDAR EM UM DETERMINADO PROCESSO OU SISTEMAS DE TRÁFEGO ÁEREO. (AQUI O DESENVOLVEDOR VAI PROJETER O SOFTWARE, MAS O CONTROLE DAS ESPECIFICAÇÕES FICA A CARGO DA EMPRESA QUE ENCOMENDOU O PRODUTO).**

O tipo de PRODUTO DE SOFTWARE mais usados nos dias de hoje tem sido os PRODUTOS GENERICOS, como os sistemas ERP, que são usados para atender a gestão empresarial, como o SAP, que é adaptado para resolver os processos de negócio.

Quando o desenvolvedor projeta o Software, ele deve saber que o aplicativo não vai ser usado só por ele, mas sim por outras pessoas também. Então a qualidade da **APLICAÇÃO** deve estar presente no aspecto de **OQUE O PROGRAMA ESTÁ FAZENDO** e **COMO ELE SE COMPORTA ENQUANTO ESTÁ SENDO EXECUTADO**. Isso tudo vai além dos dados técnicos, **ENVOLVE A DOCUMENTAÇÃO PARA O APP – A ESTRUTURA – A ORGANIZAÇÃO DOS PROGRAMAS DO SISTEMA**. Tudo isso é chamado de **ATRIBUTOS NÃO FUNCIONAIS OU ATRIBUTOS DE QUALIDADE – TEMPO DE RESPOSTA DO PROGRAMA** (esses atributos variam de aplicação para aplicação, como um jogo deve ser ágil, um sistema bancário deve ser seguro). Ele **dever ter MANUTENIBILIDADE** (evolui com as necessidades do cliente) – **CONFIANÇA E PROTEÇÃO** (não pode causar prejuízos físicos ou econômicos e deve ser blindado contra usos maliciosos) – **EFICIÊNCIA** (não desperdiçar recursos do sistema) – **ACEITABILIDADE** (User-Friendly).

A Engenharia de Software inclui todos os **ASPECTOS DE DA PRODUÇÃO DE SOFTWARE**, desde os estágios iniciais, gerenciamento do projeto de software e desenvolvimento técnico de ferramentas. Por isso a produção de um programa **É MAIS BARATA AO LONGO PRAZO**.

Engenheiros podem ter uma **ABORDAGEM SISTEMÁTICA E ORGANIZADA** para fazer um produto de alta qualidade, **CHAMADO DE PROCESSOS DE SOFTWARE**, envolve:

- ➡ **ESPECIFICAÇÃO DE SOFTWARE:** cliente + engenheiro definem as restrições do software;
- ➡ **DESENVOLVIMENTO DE SOFTWARE:** o software é projetado e programado;
- ➡ **VALIDAÇÃO DE SOFTWARE:** ele vai ser avaliado para ver se bate com o pedido do cliente;
- ➡ **EVOLUÇÃO DE SOFTWARE:** ele vai ser atualizado de acordo com as necessidades do cliente ou do mercado.

Alguns tipos de **SOFTWARES**:

- ⚡ **HETEROGENEIDADE:** vai ser um software “mais versátil”, operando em **COMPUTADORES** e **DISPOSITIVOS MÓVEIS**. E mesmo sendo um software mais novo/moderno pode ser necessário ter que integrá-lo ao um sistema +antigo;
- ⚡ **MUDANÇA DE NEGÓCIO SOCIAL:** a mudança no software deve ser feita o + rápido possível. É importante evoluir mais para entregar um software de qualidade em menos tempo;
- ⚡ **SEGURANÇA E CONFIANÇA:** como o software está presente em todos os aspectos da vida, ele deve ser seguro e confiável, portanto, deve ser **ROBUSTO E BLINDADO CONTRA-ATAQUES MALICIOSOS**, principalmente em sistemas remotos ou em páginas Web ou Web service.

Existem alguns tipos de Aplicações:

- 1- Aplicações **STAND-ALONE**: são aplicações que rodam em um computador pessoal e que não precisam estar conectados à rede, só instalar e usar, COMO EDITOR DE FOTOS;
- 2- Aplicações **BASEADAS EM TRANSAÇÕES**: vai ter um servidor e o cliente vai acessar os serviços do servidor através de um computador pessoal como aplicações WEB para compra de produtos.  
Aplicações Remotas: aplicações que trabalham remotamente
- 3- Sistemas de **CONTROLE EMBUTIDO**: é um sistema que vai controlar e gerenciar um hardware, são sistemas leves, simples com linha de código fina e não vão rodar em um computador como o SOFTWARE DE UM MICROONDAS;
- 4- Sistemas de **PROCESSAMENTO EM LOTES**: são sistemas que fazem o processamento de grandes lotes, vai pegar um lote com grande quantidade de informações/dados e processar aquilo, como por exemplo SISTEMAS DE COBRANÇA E PAGAMENTO (FINANCEIROS);
- 5- Sistemas de **ENTRETENIMENTO**: responsáveis em entreter o usuário, como PLAYER DE VÍDEOS, PLAYER DE MÚSICA, JOGOS;
- 6- Sistemas para **MODELAGEM E SIMULAÇÃO**: usados para fazer a projeção de processos ou que fazem simulação de algum fato, como o AUTOCAD, SISTEMAS DE PREVISÃO (PREDIÇÃO DE TEMPO);
- 7- Sistemas de **COLETAS DE DADOS**: usam sensores para receber dados do ambiente e enviam para um sistema fazer o processamento;
- 8- Sistemas **de SISTEMAS**: são sistemas muito grandes, robustos, como SIMULADOR;

# TIPOS DE SISTEMAS

No começo a Internet era um armazenamento de informações acessível em qualquer lugar do Mundo e podiam ser executados em computadores locais dentro de uma organização. A partir do ano de 2.000 ela começou a evoluir, surgindo **SISTEMAS WEB**, esses não eram programas específicos, mas sim serviços que poderiam ser acessados via Navegador. Isso desencadeou uma série de desenvolvimento de variados softwares que realizavam diversos serviços através da Internet. E graças isso também começou a produção de **NAVEGADORES WEB** capazes de suportar esses softwares. Isso tudo facilitou para todas as pessoas, pois o comércio migrou para a Interação Web usando softwares da empresa, o que cortava custos, pois o programador não precisava desinstalar ou instalar o programa de cada computador para atualizar.

O próximo passo, foi o desenvolvimento de **WEB SERVICE**, ele permite que aplicações cooperem e compartilhem informações e dados umas com as outras. O WEB SERVICE é uma aplicação que vai pegar uma informação que um COMPUTADOR está requisitando e consultar um **BANCO DE DADOS**, o **BANCO DE DADOS** vai retornar essa pesquisa com a resposta e o **WEB SERVICE** vai converter em **FORMATO XML** e informar o dispositivo.



Com o avanço da Internet, a forma de acessar conteúdo e serviço também mudou, pois antes os programas eram rodados em computadores isolados e a Internet possibilitou o acesso de serviços via "Nuvem", por exemplo, e-mail. O que ampliou o alcance para todo mundo.

Algumas dessas mudanças consiste em:

1-REUTILIZAÇÃO DE SOFTWARE: para montar UM SISTEMA WEB, os programadores pensam em usar componentes de sistemas que já existem;

2-ENTEGA INCREMENTAL: hoje em dia é difícil o cliente especificar exatamente o que ele quer, por isso de usa muito o MÉTODO INCREMENTAL para no DESENVOLVIMENTO de um SISTEMA WEB;

3-INTERFACE RESTRITA: interfaces de usuário são restringidas pela capacidade dos navegadores, por mais que tecnologias como o ALAX permite criar interfaces ricas e diferenciadas, é difícil usar elas. A INTERFACE de um SISTEMA WEB tem a tendência de ser mais pobre em relação a INTERFACE de UM PROGRAMA desenvolvido para rodar no COMPUTADOR.

# PROCESSO DE SOFTWARE

Processo de Software, inclui um conjunto de atividades que levam a produção de um software, envolve técnicas que vão além de programar em uma determinada linguagem. Todo Processo de Software deve incluir as seguintes atividades:

- 1- **ESPECIFICAÇÃO DO SOFTWARE:** o programador deve conversar com o cliente, para saber o que o cliente deseja que o software faça, definindo **OQUE O PROGRAMA IRÁ FAZER!** E no final do processo verificarem se as **EXPECTATIVAS** foram **BATIDAS**;
- 2- **PROJETO E IMPLEMENTAÇÃO:** pode fazer um mapa mental detalhando o passo a passo o que o programa irá fazer. Depois elaborando um **PROTÓTIPO** para ter uma base de como vai ficar e começar a programar, desse modo caso identificar um erro, é mais fácil consertar durante o andamento do projeto;
- 3- **VALIDAÇÃO:** depois que o software estiver pronto, deve ser implementado nos computadores do cliente, para checar se o programa não tem nenhum erro de codificação ou se a máquina vai conseguir rodar o programa (às vezes a hardware pode ser muito antiga). E assim, o **PROGRAMA VAI SENDO IMPLEMENTADO DEFINITIVAMENTE** na empresa;
- 4- **MANUTENÇÃO:** depois que o software foi implementado, o sistema deve ser atualizado, seja porque o governo mudou alguma legislação ou o cliente achou uma alguma parte não intuitiva.

Essas atividades fazem parte de todos os **PROCESSOS DE SOFTWARE** e incluem subatividades (**VALIDAÇÃO DE REQUISITOS, PROJETO DE ARQUITETURA, TESTE etc.**). E existem atividades para apoiar o desenvolvimento de um software (**DOCUMENTAÇÃO e GERENCIAMENTO DE CONFIGURAÇÕES**).

Alguns modelos que podem ser utilizados na etapa de PROJETO E IMPLEMENTAÇÃO:

**MODELO CASCATA:** usado quando a cliente especifica exatamente o que ele quer (mais usado no caso do governo porque a licitação não muda) é o modelo mais antigo, na qual o desenvolvimento vai seguir em uma única direção. Sendo o 1º a colocar organização, possui começo, meio e fim. Possui algumas fases divididas em:

**1ºANÁLISE E DEFINIÇÃO DE REQUISITOS:** o cliente fala suas necessidades e assim vai ser definido o que o programa vai fazer;

**2ºPROJETO DE SISTEMA E SOFTWARE:** a equipe faz um esboço da arquitetura geral e passa para as linhas de código;

**3ºIMPLEMENTAÇÃO E TESTE UNITARIO:** o código do sistema vai ser gerado;

**4ºINTEGRAÇÃO E TESTE DE SISTEMA:** o sistema vai ser testado pelo cliente;

**5ºOPERAÇÃO E MANUTENÇÃO:** com o sistema já implementado, deve cuidar das atualizações e corrigir possíveis erros que o cliente pode achar.

As **VANTAGENS DO MODELO CASCATA** é que como o projeto não volta atrás, a documentação gerada é totalmente certa. Porém como o projeto não volta atrás, tentar alguma mudança pode provocar confusão e adição de funcionalidades podem gerar mais custos e gastos com mais documentações.

**MODELO INCREMENTAL:** tem a mesma essência que o modelo cascata, porém aqui é entregue o “coração” do programa/a principal função planejada a fazer, ou seja, mesmo que todas as funcionalidades não estejam incluídas, o cliente ainda consegue usar e obter ganho mais rápido se comparado ao modelo cascata. **O MODELO INCREMENTAL** também segue a ideia de cascatas, mas ao fim de cada ciclo apresenta para o cliente (feedback). E a quantidade de documentos feitos é menor se for comparado ao Cascata. E o modelo mais usado atualmente.

Ele também apresenta algumas **DESVANTAGENS**, visto que o processo não é visível, então se precisa de entregas regulares para medir o progresso. Se o programa for desenvolvido com rapidez, não compensa gastar com documentos e se muitas mudanças forem adicionadas com o andamento do projeto.

**ENGENHARIA ORIENTADA A REÚSO:** acontece muitas vezes de forma informal, se baseia no programador usar **LINHAS de CÓDIGOS** já **EXISTENTES** para montar o **SEU PROGRAMA**. Ou seja, dependem muito de ter uma base de componentes reusáveis, e as vezes alguns componentes são sistemas completos (COTS ou DE PRATELEIRA), que fornecem uma funcionalidade específica. Possui algumas fases como:

- 1- **ESPECIFICAÇÃO DOS REQUISITOS:** o cliente vai definir o que ele vai querer
- 2- **ANÁLISE DE COMPONENTES:** o programador irá procurar componentes para solucionar a ESPECIFICAÇÃO DE REQUISITOS, é difícil achar o componente **EXATO** para a especificação **EXATA**, nesse caso ele vai possuir alguma funcionalidade.
- 3- **MODIFICAÇÃO DE REQUISITOS:** os requisitos irão ser analisados depois de encontrarem os componentes e se for o caso, eles passarão por alguns ajustes para “alinhar” com os componentes (se for impossível, é procurada um componente para uma solução alternativa);
- 4- **PROJETO DO SISTEMA COM REÚSO:** framework é montado usando os componentes;
- 5- **DESENVOLVIMENTO E INTEGRAÇÃO:** o software que não pode ser reusado é desenvolvido e os componentes e sistemas COTS são integrados -> **ORIGINANDO UM NOVO SISTEMA**

WebService, coleções de objetos (.NET ou J2EE), sistemas de software stand-alone configurados para uso particular – são exemplos de tipos de componentes que podem ser usados.

# ESPECIFICAÇÃO DE SOFTWARE



## ENGENHARIA DE REQUISITOS:

A Engenharia de Requisitos é o processo de descobrir os requisitos de um sistema, ou seja, **descobrir o que** o software **deve fazer** para saciar a necessidade do cliente, descobrir **qual são os serviços e funcionalidade** que o sistema deve ter. **Depois** de descobrir esses requisitos, deve **analisar e documentar para verificar**.

**Tem como objetivo, gerar um documento de requisitos que foram acordados com o cliente , sendo a parte +detalhada dos requisitos para o cliente e a parte +detalhada do sistema para o desenvolvedor .**

**É dividida em 4 partes:**

**1-ESTUDO DE VIABILIDADE:** é feito um estudo para determinar se os requisitos levantados irão resolver os problemas do usuário, se os requisitos serão rentáveis levando em conta o hardware e software disponível para ser negócio e conforme os resultados dessa análise decidir se o projeto vai avançar ou não;

**2-ELICITAÇÃO E ANÁLISE DE REQUISITOS:** nessa etapa é feita um entendimento dos requisitos observando os que já existem e pode envolver um ou mais modelos de protótipos para entender o sistema ( 🧐 já sei o que o sistema deve fazer, mas como ele pode fazer isso? Então vou olhar o que já existe para ter uma ideia);

**3-ESPECIFICAÇÃO DE REQUISITOS:** traduz as informações obtidas na análise e converte em um documento de requisitos ( 🧐 já sei como eu vou fazer meus requisitos, agora vou descrever eles nesse documento);

**4-VALIDAÇÃO DE REQUISITOS:** os requisitos descritos anteriormente são analisados em relação com as necessidades, para ver se eles estão alinhados, se resolvem o problema e caso haja algum erro , já é arrumado ( 🧐 vou pegar esse documento que eu fiz com os requisitos que eu fiz depois de estudar outros já existentes, e comparar com o documento de necessidades, para ver se eles estão se resolvendo).



**PARA A IMPLANTAÇÃO** do projeto, é pego a **ESPECIFICAÇÃO DOS REQUISITOS** e começado a desenvolver o programa (algoritmo, estrutura de rede, modelo de desenvolvimento, interface) , normalmente usando o modelo ITERATIVO (cada processo depende do anterior) pois é difícil de chegar em um projeto final de imediato!

**É DIVIDIDO EM ALGUMAS ETAPAS:**



**1- PROJETO DE ARQUITETURA** é definido a estrutura geral do sistema, como os componentes principais vão se relacionar com os componentes secundários e como serão distribuídos;

**2- PROJETO DE INTERFACE** é definido uma interface para cada componente, de maneira que cada componente **tenha sua interface independente** (não necessite de outra), **uma interface precisa** para cada componente/processo;

**3- PROJETO DE COMPONENTE** é projetado cada componente/ação no sistema

**4- PROJETO DE BANCO DE DADOS** é projetado a estrutura do banco de dados (como ele vai ser apresentado).



**A VALIDAÇÃO DE SOFTWARE**, é realizada para verificar se os requisitos planejados nos documentos funcionam no software já feito, ou seja, é colocado o programa em ação e rodado testes e feito uma revisão em todos os mecanismos. Quando o programa é pequeno ele pode ser testado de uma forma geral, porém quando **o sistema é grande**:

**Primeiro é feito um teste em componentes isolados – TESTE DE DESENVOLVIMENTO**

Depois os **componentes/ações junto com o sistema em si**, ou seja, depois de testados isoladamente, eles são juntados na estrutura e testado como um sistema completo e assim acha erros de funcionalidade e de interface ou algumas recomendações de melhorias são feitas – **TESTE DE SISTEMAS**

Depois é testado o **sistema em si + com os dados inseridos**, pode revelar erros ou omissões do sistema com dados reais, pois a forma do processamento de informações vai ser diferentes dos valores testados – **TESTE DE ACEITAÇÃO**



Para ocorrer a **EVOLUÇÃO DO SOFTWARE**, é necessário ter uma flexibilidade de sistemas, para que as mudanças no programa possam ser feitas, pois alterar alguma coisa no software é caro, mas não tão caro quanto uma alteração no hardware.

Para evitar mudanças pode ser feita a **PROTOTIPAÇÃO DO SISTEMA**: na qual é feita uma versão do sistema rapidamente para o cliente experimentar e ver se bate com suas necessidades e ver o que pode ser feito mais (refinarem os requisitos);

E pode ser usado a **ENTREGA INCREMENTAL**, na qual incrementos do sistema é entregue para o cliente ver e experimentar e comentar sobre, evitando mudanças futuras, comprometimento de requisitos e caso exista algum erro, no próximo incremento esse erro já vem consertado, com custos baixos.

# ENGENHARIA DE REQUISITOS



## ENGENHARIA DE REQUISITOS:

A Engenharia de Requisitos é o processo de descobrir os requisitos de um sistema, ou seja, **descobrir o que o software deve fazer** para saciar a necessidade do cliente, descobrir **qual são os serviços e funcionalidade** que o sistema deve ter. **Depois** de descobrir esses requisitos, deve **analisar e documentar para verificar**.

**Tem como objetivo, gerar um documento de requisitos que foram acordados com o cliente , sendo a parte +detalhada dos requisitos para o cliente e a parte +detalhada do sistema para o desenvolvedor .**

Como os requisitos vão ser colocados em documentos, pessoas diferentes vão ler, sendo assim podem ser divididos em:



**REQUISITOS DE USUÁRIO:** é o documento destinado para o cliente ou para qualquer pessoa que não seja de TI, é escrito com uma linguagem natural/+acessível ou uma tabela. Nele vai ter as funções e restrições do sistema **de forma bem simples e geral/abstrata – ALTO NÍVEL;**



**REQUISITOS DE SISTEMAS:** é o documento destinado para os desenvolvedores, especialistas em TI, com uma linguagem +detalhada e padronizada, sendo o mais preciso possível para o programador não fazer nada errado. As **informações vão estar +detalhadas/+específicas – BAIXO NÍVEL;**

**REQUISITOS FUNCIONAIS:** são ações específicas que eu quero que o sistema execute, trata-se de funcionalidades do sistema, como ele deve reagir depois de uma entrada de dados, qual é o comportamento dele e o que não pode fazer. Deve ser **COMPLETO** (todos os serviços pedidos devem ser feitos) e **CONSISTENTE** (não pode ter definições ambíguas ou contraditórios) que é difícil em projetos grandes.

**REQUISITOS NÃO FUNCIONAIS:** está relacionado com a operação do sistema como um todo e não a **funcionalidades específicas**, por exemplo a segurança, a eficácia, a performance, custo, usabilidade, adaptabilidade, ou seja, são **qualidades** que o sistema tem e **não das ações específicas. Podem ser também regra inegociável/uma imposição que o cliente faz** (deve ser programado em java ou deve ser orientado a objeto).

**Requisito de produto** – relacionado ao comportamento do software;

**Requisito Organizacional** – relacionado ao processo de desenvolvimento do software/operacional, ex: linguagem usada;

**Requisito Externo** – regra que vem acima do cliente, como legislação, questões éticas

Como a Engenharia de Requisitos elabora um documento com os requisitos pedido pelo cliente pode ocorrer a **DOCUMENTAÇÃO DE REQUISITOS**, quando os requisitos estão mudando tão rápido, que a documentação não acompanha o desenvolvimento de software, então tem a **VERIFICAÇÃO DE REQUISITOS**, tendo a coleta de requisitos de forma incremental **PRIORIZANDO OS REQUISITOS** que devem ser implementados na próxima incrementação.

REQUISITOS NÃO FUNCIONAIS: atributos de qualidade (maneira de mensurar a qualidade do software, permite ter critérios para avaliar se um software é bom ou não), não precisa entregar 100% dos atributos mas deve-se entregar o que cada sistema requisitar, porém todos os software DEVEM TER DESEMPENHO, é o único requisito que aparece em todos os softwares

e restrição(inegociável)