



Teste no CVDS



# Revisão

## Qualidade de *software*

*“Qualidade de software é a conformidade com requisitos funcionais e de desempenho explicitamente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas que são esperadas em todo software desenvolvido profissionalmente”*

*Pressman (PRESSMAN, 2006)*



# Revisão

**Requisitos de qualidade:** Conjunto completo das **características** da qualidade de um processo, item, material, produto ou serviço, com valor nominal e seus respectivos limites máximo e/ou mínimo de aceitação (tolerâncias).

**Requisitos de software:** são descrição dos recursos e funcionalidades do sistema alvo. **Os requisitos** transmitem as expectativas dos usuários do produto de **software**.



# Revisão

**Requisitos de qualidade:** Conjunto completo das **características** da qualidade de um processo, item, material, produto ou serviço, com valor nominal e seus respectivos limites máximo e/ou mínimo de aceitação (tolerâncias).

**Requisitos de software:** são descrição dos recursos e funcionalidades do sistema alvo. **Os requisitos** transmitem as expectativas dos usuários do produto de **software**.



# Revisão

Do ponto de vista técnico, temos duas atividades de extrema importância para garantir a qualidade de um software: verificação e validação – conhecidas como atividades de V&V.

VERIFICAÇÃO	Atividades para responder à pergunta: “ <i>Estamos construindo o produto corretamente?</i> ”;
VALIDAÇÃO	Atividades para responder à pergunta: “ <i>Estamos construindo o produto certo?</i> ”.



# Revisão

## TESTE DE SOFTWARE

*“Teste é a simulação de um artefato executável com entradas e pré-condições conhecidas e a comparação entre a saída atual e a saída esperada para determinar se o artefato está em conformidade com os requisitos” (FIRESMITH, 2013).*

*“Teste é o processo de executar um sistema ou componente sob condições específicas, observando e registrando os resultados, avaliando alguns aspectos do sistema ou componente” (ISO/IEC/IEEE 24765, 2010).*

*“Teste é o processo de executar um programa ou sistema com a intenção de encontrar erros” (MYERS, 2004).*



# Revisão

## Casos de teste

**Casos de teste** é uma sequência de passos que devem ser executados no sistema, sendo que os dados de entrada e saída esperada para cada passo são especificados. Os casos de teste devem “direcionar” as ações do testador em uma determinada funcionalidade e fazer com que ele observe se o resultado que ele obtém em cada passo é o resultado esperado, de acordo com os requisitos



# Revisão

## DEFEITOS

O que fazer quando o programa falha e não retorna o resultado esperado?

Ao identificar um defeito, o testador deve reportar o defeito de forma a permitir que o time de desenvolvimento identifique facilmente como eliminar o defeito do sistema.





# Revisão

## Cenário de Testes

De acordo com a norma IEEE 829-2008 (IEEE 829-2008, 2008), cenário de teste é um conjunto de casos de teste relacionados, que comumente testam o mesmo componente ou a mesma funcionalidade do sistema.




# Revisão

ENGANO ( <i>MISTAKE</i> )	Uma ação humana que produz um resultado incorreto, como uma codificação ou modelagem errada;
DEFEITO ( <i>FAULT</i> )	Uma imperfeição ou deficiência em um artefato que faz com que este não esteja em conformidade com os requisitos ou especificações, sendo necessária sua correção ou substituição. Termos como "erro" e " <i>bug</i> " comumente são usados para expressar defeitos;




# Revisão

ERRO ( <i>ERROR</i> )	Resultado incoerente produzido por meio de uma ação no sistema;
FALHA ( <i>FAILURE</i> )	Incapacidade do <i>software</i> exercer a função para a qual foi desenvolvido.



## O teste nas fases de vida e de desenvolvimento de um *software* (CVDS)

Sempre que falamos de alguma atividade relacionada ao desenvolvimento de software, precisamos lembrar dos modelos de processo de desenvolvimento, afinal, é assim que geralmente começa nosso entendimento técnico sobre como desenvolver um software



## Modelos de ciclo de vida de desenvolvimento de *software*

Um modelo de ciclo de vida de desenvolvimento de software descreve os tipos de atividades realizadas em cada estágio de um projeto de desenvolvimento de software e como as atividades se relacionam umas com as outras de forma lógica e cronológica. Há vários modelos de ciclo de vida de desenvolvimento de software, cada um dos quais requer abordagens diferentes para o teste.



## Desenvolvimento de *software* e teste de *software*

É uma parte importante do papel de um testador estar familiarizado com os modelos mais comuns de ciclo de vida de desenvolvimento de *software* para que as atividades apropriadas de teste possam ocorrer;



# Desenvolvimento de *software* e teste de *software*

Em qualquer modelo de ciclo de vida de desenvolvimento de software, existem várias características para um bom teste:

- Para cada atividade de desenvolvimento, existe uma atividade de teste correspondente;
- Cada nível de teste tem objetivos de teste específicos;
- A análise e a modelagem de teste para um determinado nível de teste começam durante a atividade de desenvolvimento correspondente;



# Desenvolvimento de *software* e teste de *software*

Em qualquer modelo de ciclo de vida de desenvolvimento de software, existem várias características para um bom teste:

- Os testadores participam de discussões para definir e refinar os requisitos e a modelagem, e estão envolvidos na revisão dos produtos de trabalho (p. ex., requisitos, modelagem, histórias de usuários etc.) assim que os rascunhos estiverem disponíveis





## Desenvolvimento de *software* e teste de *software*

Independentemente do modelo de ciclo de vida de desenvolvimento de *software* escolhido, as atividades de teste devem começar nos estágios iniciais do ciclo de vida, aderindo ao princípio de testar do início.



# Desenvolvimento de *software* e teste de *software*

O programa de estudo da ISTQB (*International Software Testing Qualifications Board*) categoriza os modelos mais comuns de ciclo de vida de desenvolvimento de *software* em :

- Modelos de desenvolvimento sequencial;
- Modelos de desenvolvimento iterativo e incremental.



# Desenvolvimento Sequencial

**Modelo de Desenvolvimento Sequencial:** descreve o processo de desenvolvimento de software como um fluxo sequencial e linear de atividades. Isso significa que qualquer fase do processo de desenvolvimento deve começar quando a fase anterior estiver concluída. Em teoria, não há sobreposição de fases, mas, na prática, é benéfico ter antecipadamente o feedback da fase seguinte.



# Desenvolvimento Sequencial

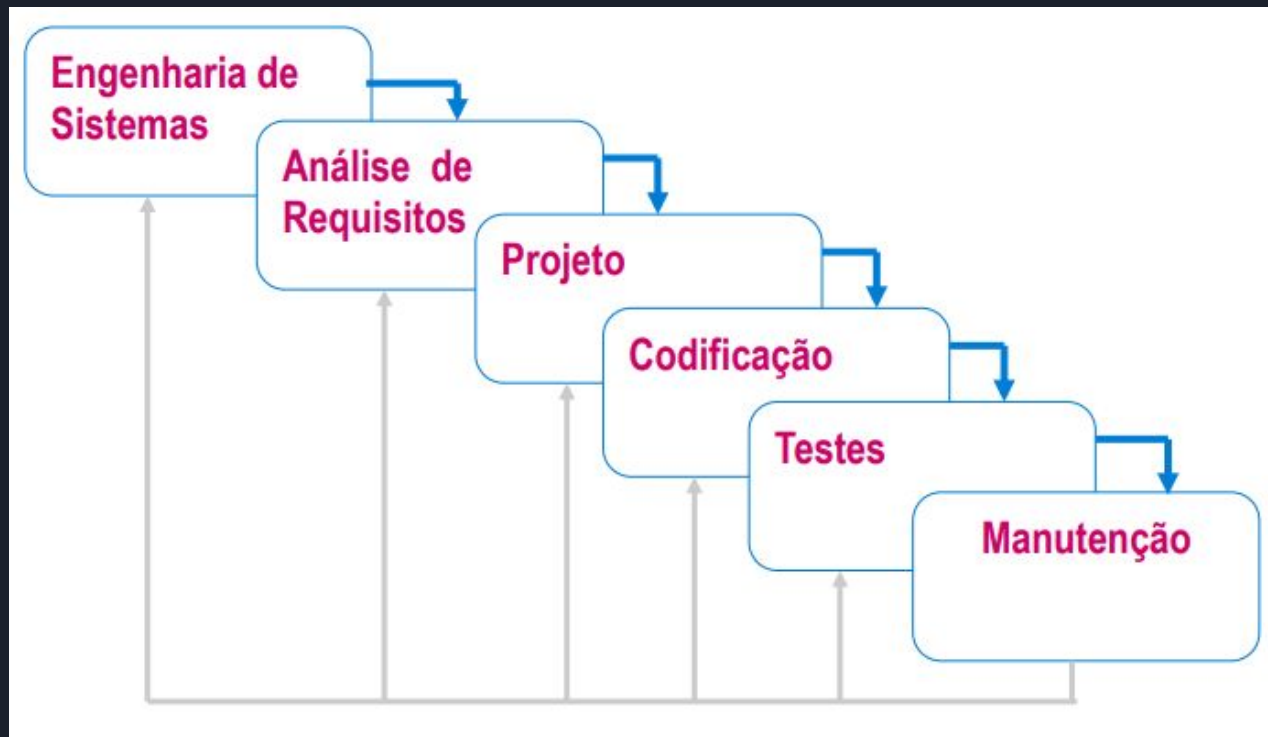
No **Modelo Cascata**, as atividades de desenvolvimento (p. ex., análise de requisitos, projeto, codificação, teste) são concluídas uma após a outra. Nesse modelo, as atividades de teste só ocorrem após todas as outras atividades de desenvolvimento terem sido concluídas.



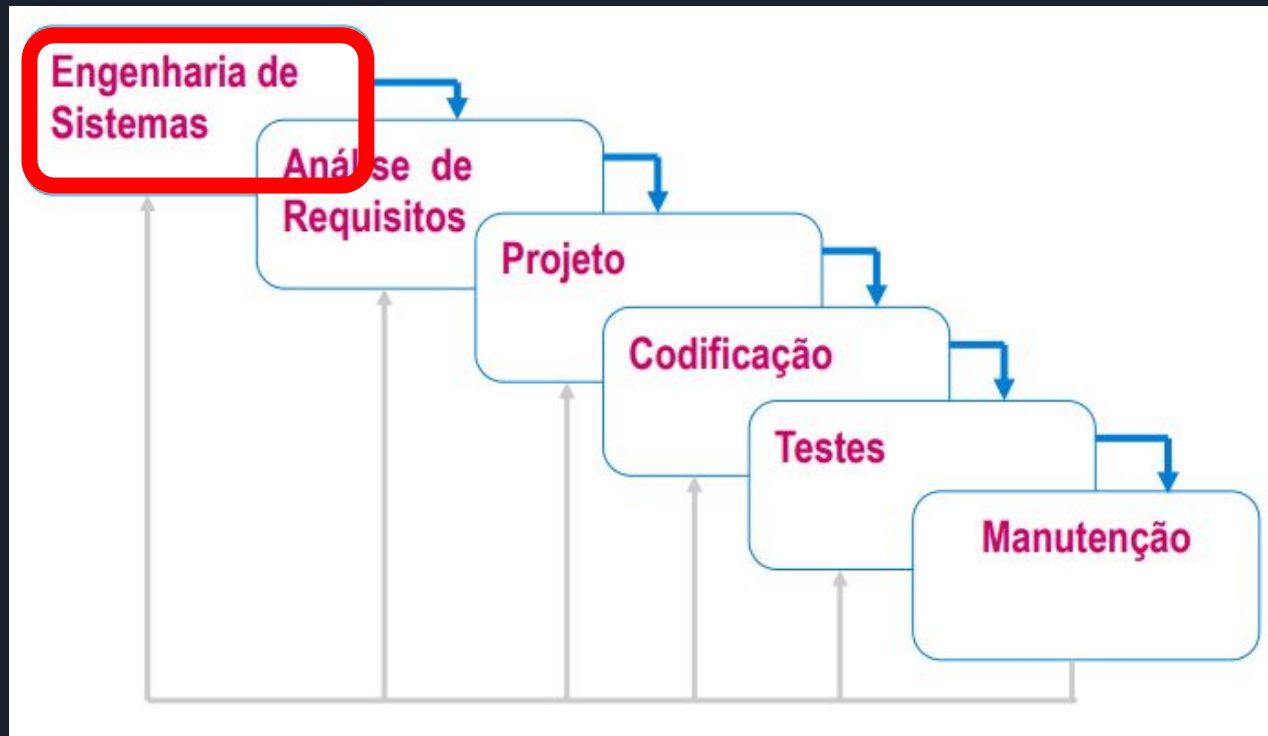
# Modelo Cascata

- modelo mais antigo e o mais amplamente usado da engenharia de software
- modelado em função do ciclo da engenharia convencional
- requer uma abordagem sistemática, seqüencial ao desenvolvimento de software
- O resultado de uma fase se constitui na entrada da outra

# Modelo Cascata



# Modelo Cascata





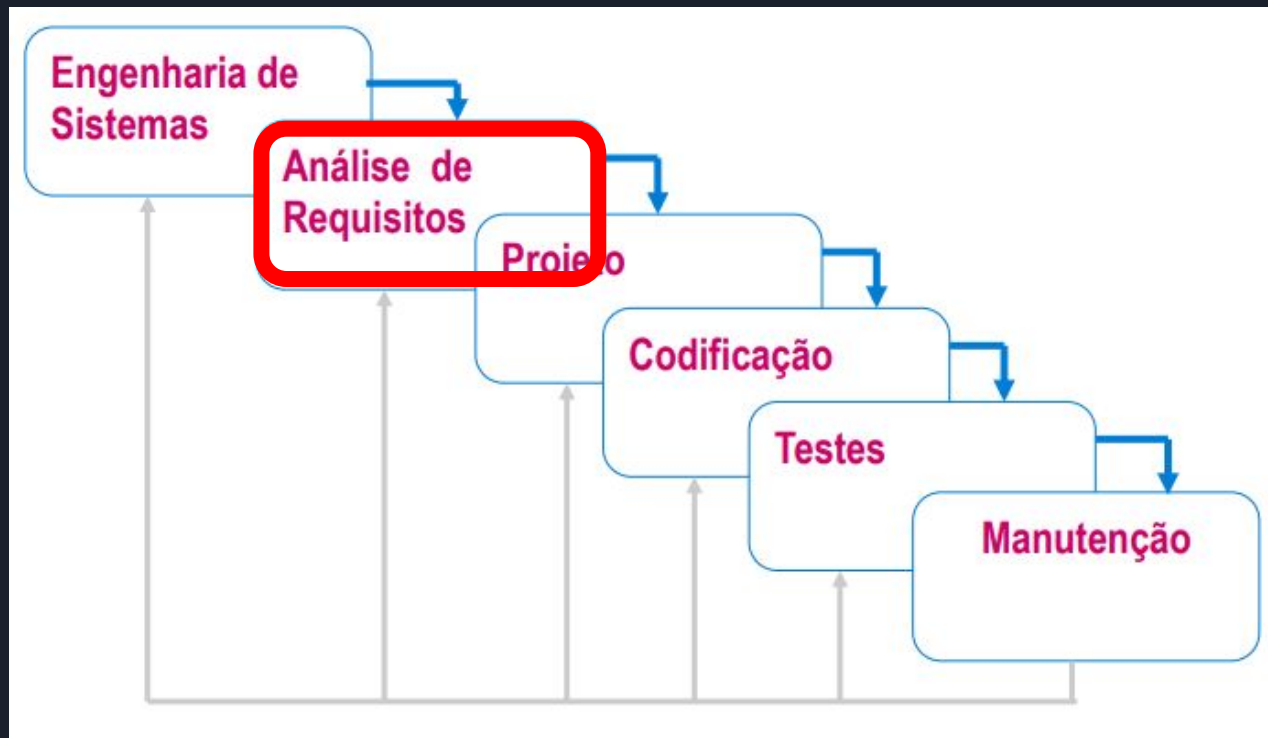
# Modelo Cascata

## Engenharia de Sistemas / Informação e Modelagem:

- envolve a coleta de requisitos em nível do sistema, com uma pequena quantidade de projeto e análise de alto nível,
- esta visão é essencial quando o software deve fazer interface com outros elementos (hardware, pessoas e banco de dados)



# Modelo Cascata



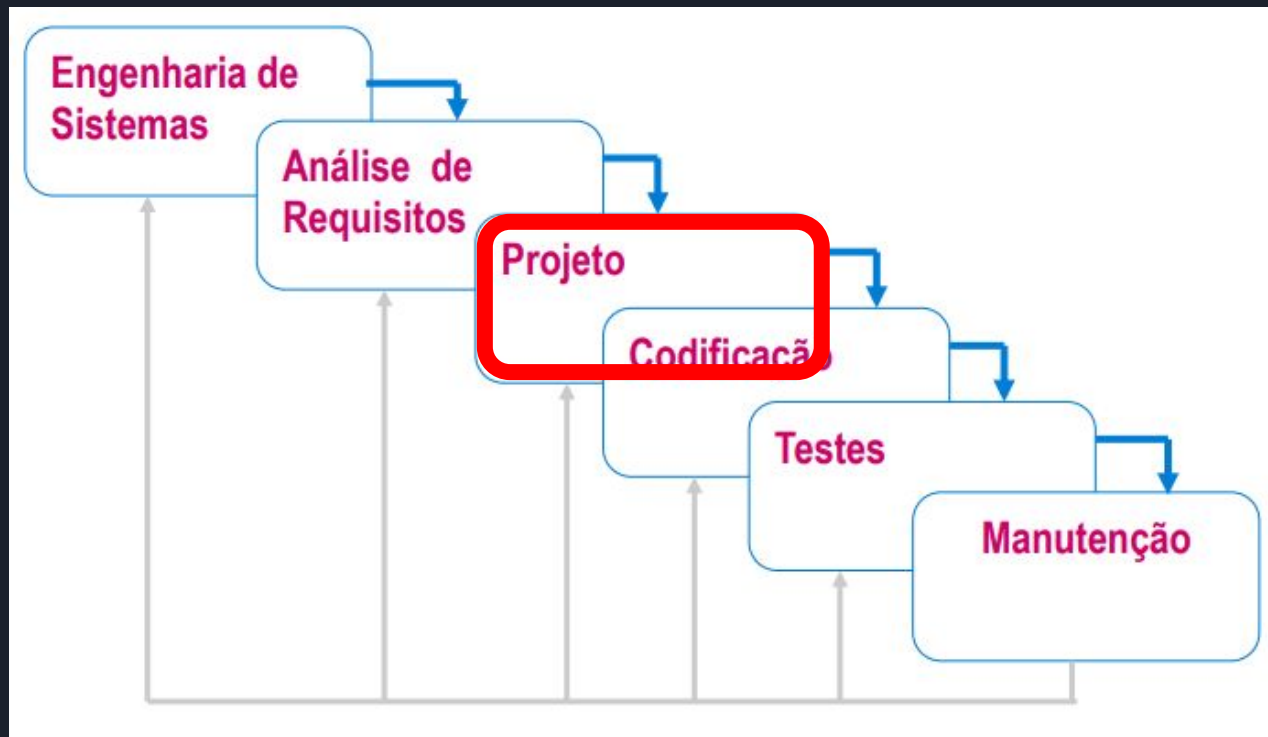


# Modelo Cascata

## Análise de Requisitos de Software

- o processo de coleta dos requisitos é intensificado e concentrado especificamente no software
- deve-se compreender o domínio da informação, a função, desempenho e interfaces exigidos
- os requisitos (para o sistema e para o software) são documentados e revistos com o cliente

# Modelo Cascata



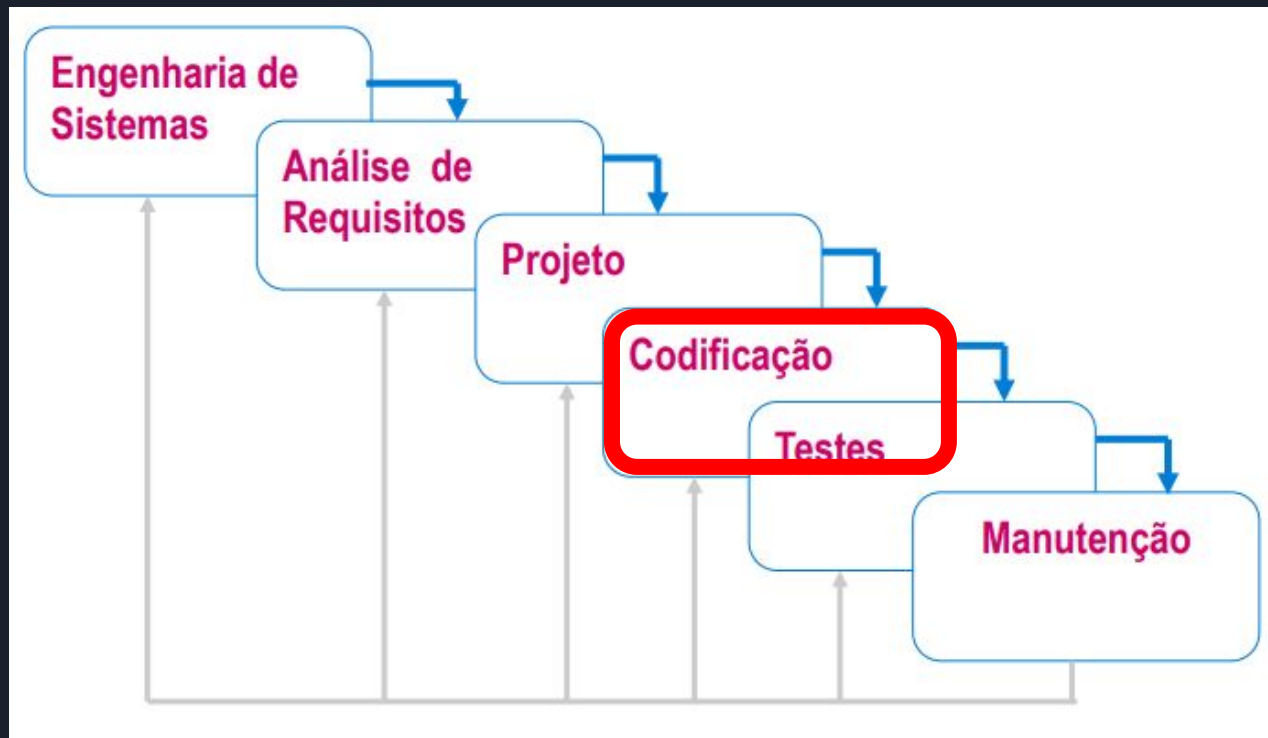


# Modelo Cascata

## Projeto

- tradução dos requisitos do software para um conjunto de representações que podem ser avaliadas quanto à qualidade, antes que a codificação se inicie,

# Modelo Cascata



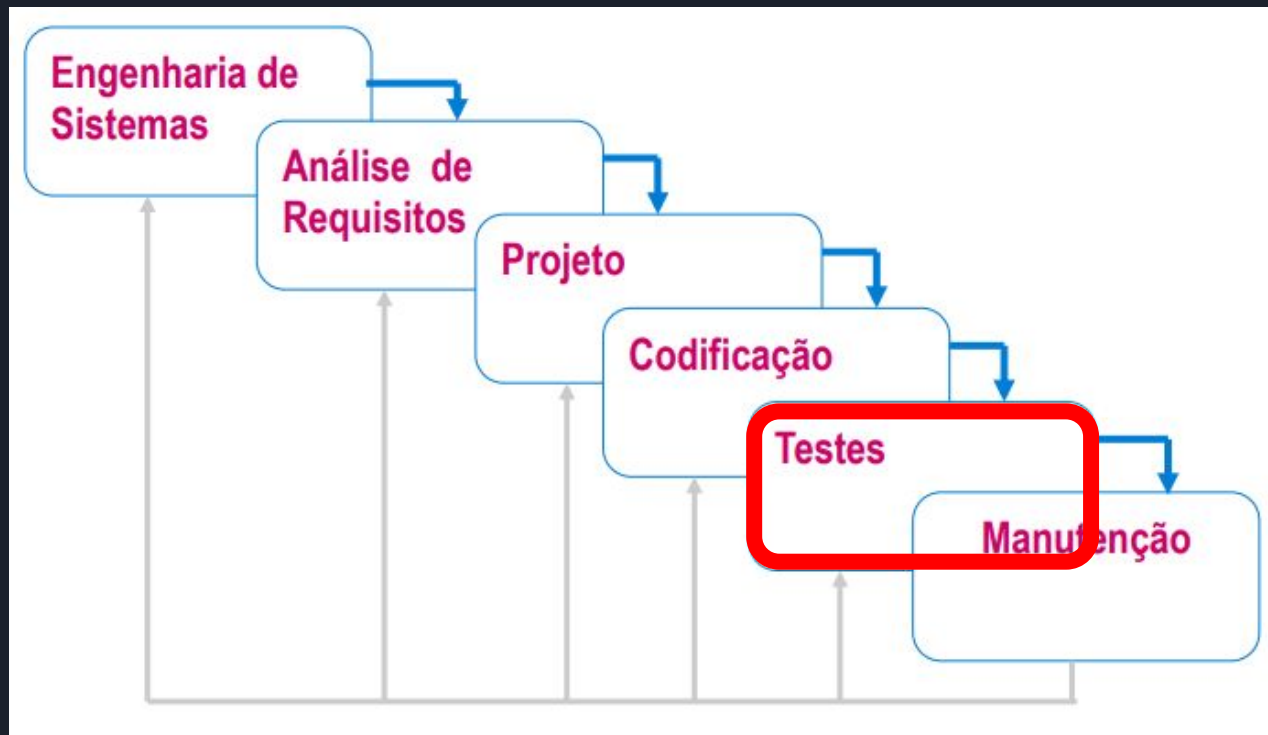


# Modelo Cascata

## Codificação

- tradução das representações do projeto para uma linguagem “artificial” resultando em instruções executáveis pelo computador

# Modelo Cascata





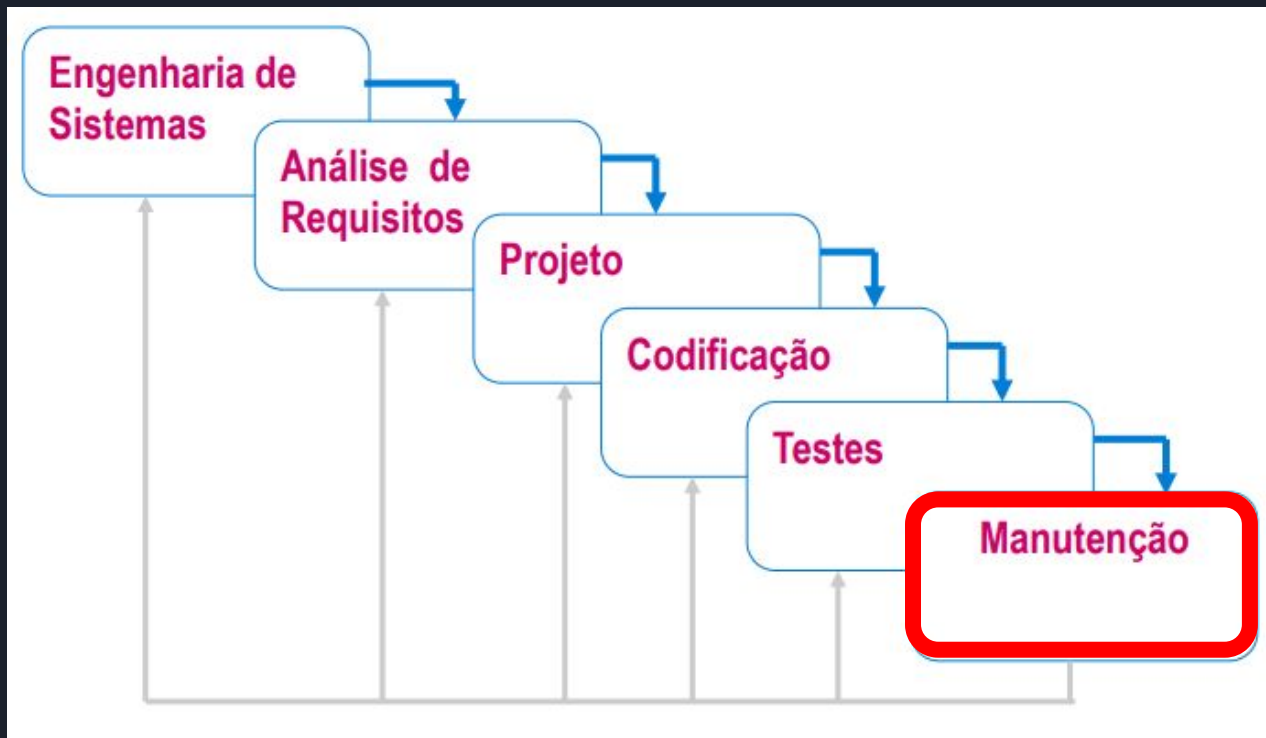
# Modelo Cascata

Testes concentra-se:

- nos aspectos lógicos internos do software, garantindo que todas as instruções tenham sido testadas ;
- nos aspectos funcionais externos, para descobrir erros e garantir que a entrada definida produza resultados que concordem com os esperados.



# Modelo Cascata





# Modelo Cascata

## Manutenção

- provavelmente o software deverá sofrer mudanças depois que for entregue ao cliente;
- causas das mudanças: erros, adaptação do software para acomodar mudanças em seu ambiente externo e exigência do cliente para acréscimos funcionais e de desempenho.



## Modelo Cascata

O modelo Cascata trouxe contribuições importantes para o processo de desenvolvimento de software:

- Imposição de disciplina, planejamento e gerenciamento
- a implementação do produto deve ser postergada até que os objetivos tenham sido completamente entendidos



# Modelo Cascata

## Problemas com o Modelo Cascata

- Projetos reais raramente seguem o fluxo seqüencial que o modelo propõe;
- Logo no início é difícil estabelecer explicitamente todos os requisitos. No começo dos projetos sempre existe uma incerteza natural ;
- O cliente deve ter paciência. Uma versão executável do software só fica disponível numa etapa avançada do desenvolvimento.



## Desenvolvimento de *software* e teste de *software*

Ao contrário do Modelo Cascata, o **Modelo V** integra o processo de teste ao longo do processo de desenvolvimento, implementando o princípio de testar do início.

O **Modelo V** inclui níveis de teste associados a cada fase de desenvolvimento correspondente, ainda suportando o princípio de testar do início.



## Modelo V

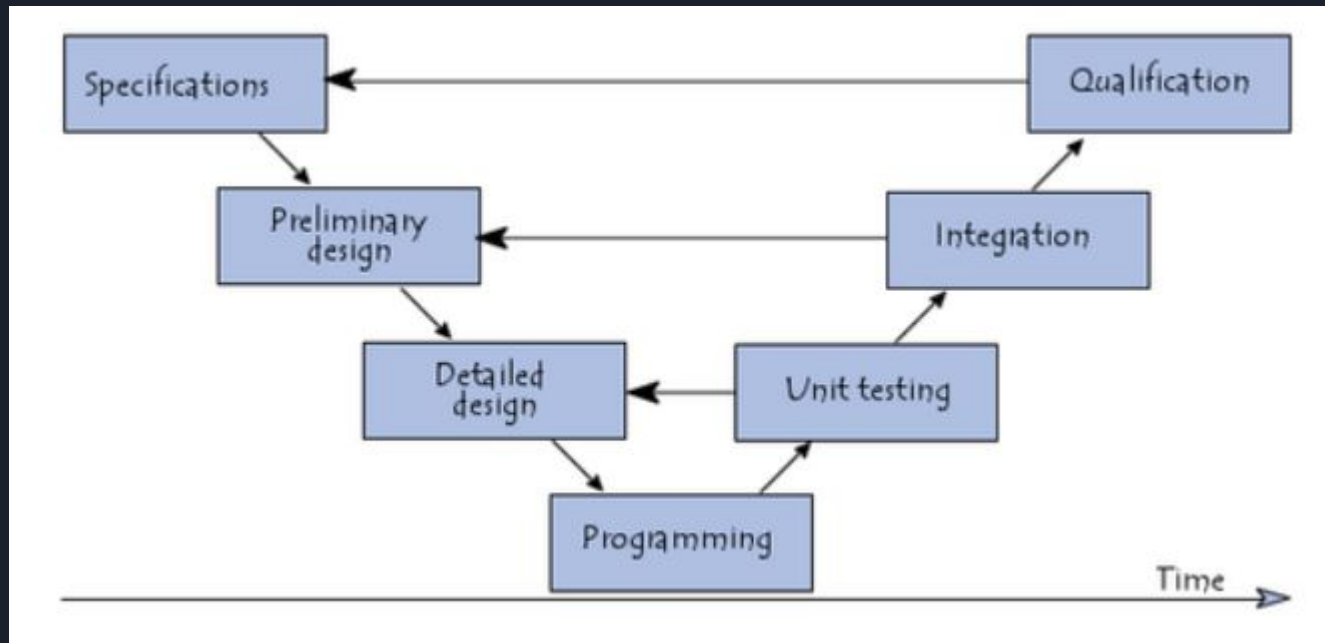
O **Modelo V** é uma variação do modelo cascata, que demonstra como as atividades de testes estão relacionadas com análise e projeto.



## Modelo V

Este modelo propõe que durante os testes de unidade e de integração, os programadores e a equipe de testes devem garantir que todos os aspectos do projeto foram implementados corretamente no código.

# Modelo V







## Modelo V

A conexão entre os lados esquerdo e direito do modelo em V implica que, caso sejam encontrados problemas durante a verificação e a validação, o lado esquerdo do V pode ser executado novamente para corrigir e melhorar os requisitos, o projeto e a codificação, antes da execução das etapas de testes que estão no lado direito.



## Modelo V

O modelo V torna mais explícitas algumas iterações e repetições do trabalho, ocultas no modelo cascata. Enquanto o enfoque do modelo cascata está nos documentos e nos artefatos, o enfoque do V está na atividade e na correção.



# Modelo V

## Principais características:

- Enfatiza a importância de considerar as atividades de testes durante o processo, ao invés de um teste posterior após o término do processo;
- Pode-se obter a retroalimentação mais rapidamente;
- Ajuda a desenvolver novos requisitos;
- Melhora a qualidade do produto resultante



## Modelo V

### Desvantagens:

- Dificuldade em seguir o fluxo sequencial do modelo;
- Dificuldade para o cliente poder especificar os requisitos explicitamente.



# Desenvolvimento sequencial

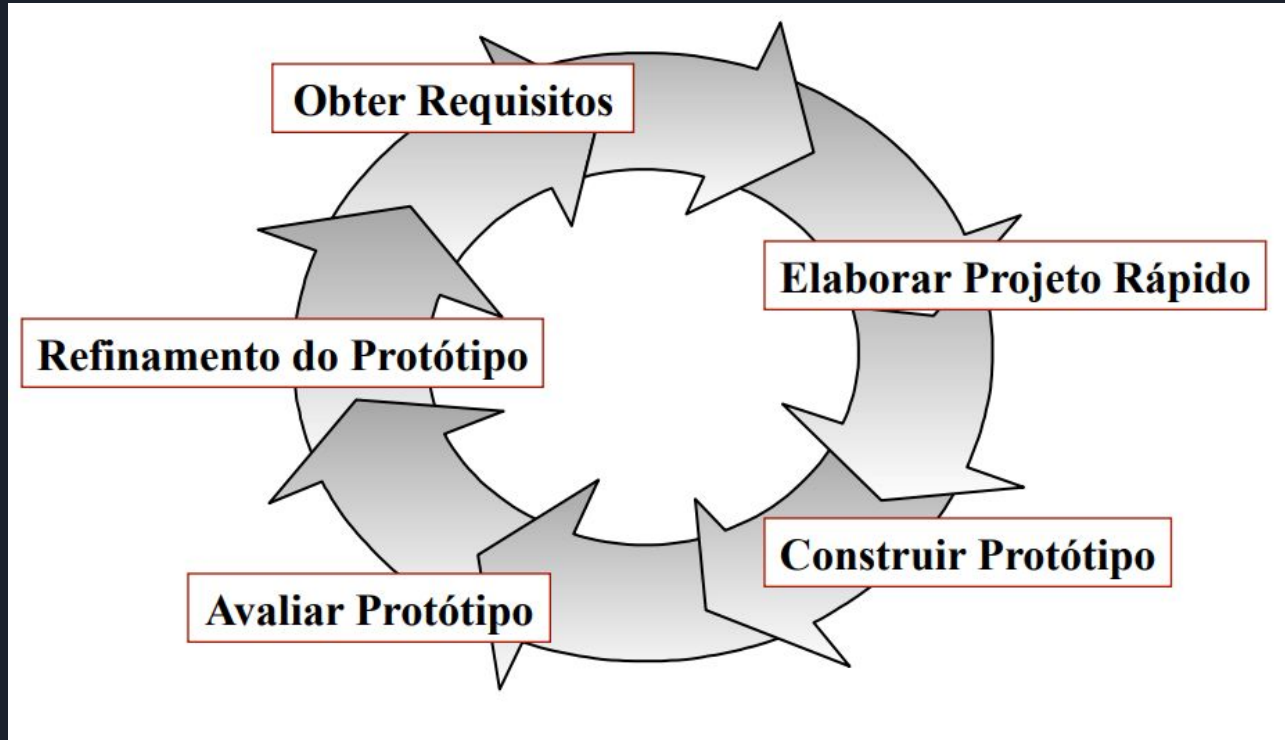
Os modelos de desenvolvimento sequencial fornecem softwares que contêm o conjunto completo de recursos, mas normalmente exigem meses ou anos para serem entregues aos stakeholders e aos usuários



# Modelo de Prototipação

- O objetivo é entender os requisitos do usuário e, assim, obter uma melhor definição dos requisitos do sistema.
- Possibilita que o desenvolvedor crie um modelo (protótipo) do software que deve ser construído
- Adequado para quando o cliente não definiu detalhadamente os requisitos.

# Modelo de Prototipação





# Modelo de Prototipação

## Obtenção dos requisitos:

- Desenvolvedor e cliente definem os objetivos gerais do software, identificam quais requisitos são conhecidos e as áreas que necessitam de definições adicionais.

## Projeto rápido:

- Representação dos aspectos do software que são visíveis ao usuário (abordagens de entrada e formatos de saída)





# Modelo de Prototipação

## Construção do Protótipo:

- Implementação rápida do projeto;

## Avaliação do protótipo:

- Cliente e desenvolvedor avaliam o protótipo



# Modelo de Prototipação

## Refinamento do protótipo:

- cliente e desenvolvedor refinam os requisitos do software a ser desenvolvido.

## Construção do produto

- identificados os requisitos, o protótipo deve ser descartado e a versão de produção deve ser construída considerando os critérios de qualidade.



# Modelo de Prototipação

- Ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente.
- A chave é definir-se as regras do jogo logo no começo.
- O cliente e o desenvolvedor devem ambos concordar que o protótipo seja construído para servir como um mecanismo a fim de definir os requisitos



# Modelo de Prototipação

## Problemas com a Prototipação:

- cliente não sabe que o software que ele vê não considerou, durante o desenvolvimento, a qualidade global e a manutenibilidade a longo prazo
- desenvolvedor freqüentemente faz uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo



# Desenvolvimento Incremental

O **desenvolvimento incremental** envolve o estabelecimento de requisitos, a modelagem, a construção e o teste de um sistema em partes, o que significa que os recursos do software crescem de forma incremental.



# Desenvolvimento Incremental

O tamanho desses incrementos de recurso varia, com alguns métodos tendo partes maiores e algumas partes menores.

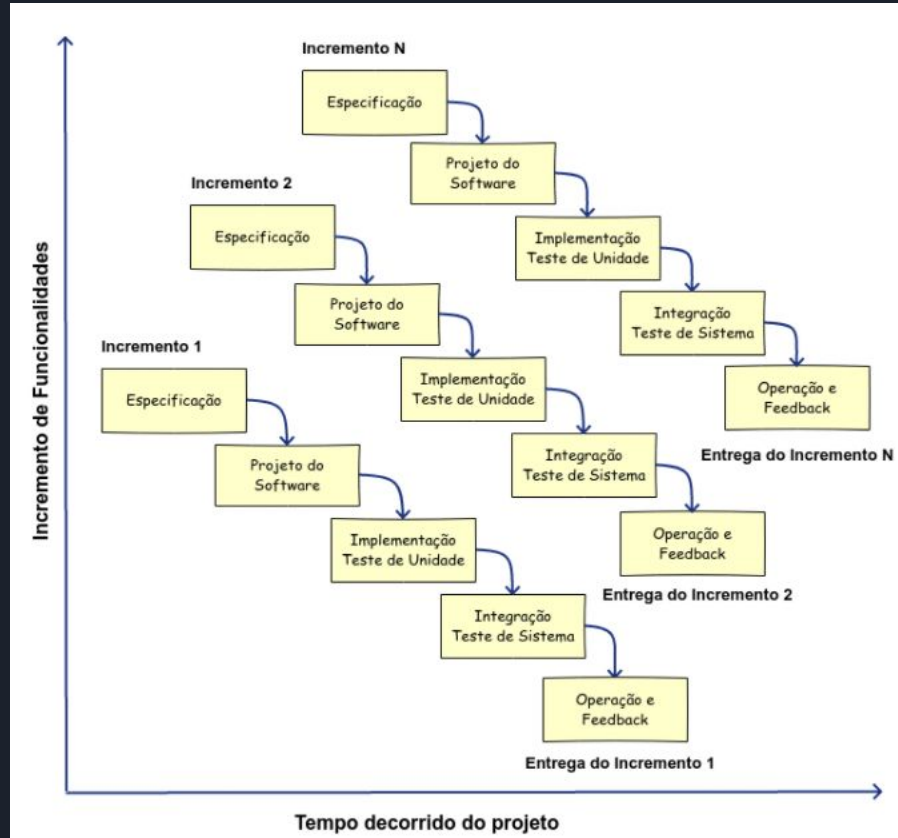
Os incrementos de recurso podem ser tão pequenos quanto uma única alteração para uma tela de interface do usuário ou uma nova opção de consulta.



# Desenvolvimento Incremental

O *Modelo Incremental* surge como uma melhoria do *Modelo em Cascata*. Ao invés de especificar e desenvolver tudo de uma só vez, este modelo trabalha com incrementos, ou seja, pequenos pedaços de software entregues de cada vez. Este modelo combina elementos do *Modelo em Cascata* aplicados de maneira iterativa, ou seja, de forma que o progresso aconteça através de sucessivos refinamentos, melhorados a cada iteração.

# Desenvolvimento Incremental







# Desenvolvimento Incremental

O primeiro incremento é frequentemente chamado de “núcleo do produto” (PRESSMAN, 2006) e contém a implementação dos requisitos básicos para que o sistema possa funcionar e atender minimamente às necessidades do cliente.



# Desenvolvimento Incremental

Cada aprimoramento é lançado como uma versão. Novas versões são criadas até que o sistema fique completo e adequado, para então, ser lançada a versão final.

Todavia, diferente do *Modelo em Cascata*, onde cada etapa tem sua vez para acontecer e, ao término de todas, o projeto termina, no *Modelo Incremental* as atividades de Especificação, Projeto, Implementação e Validação são intercaladas, acontecendo **em cada nova versão**, com rápido feedback entre todas as atividades (SOMMERVILLE, 2011).



# Desenvolvimento Incremental

## Vantagens:

- É particularmente útil quando não há mão-de-obra disponível para uma implementação completa, dentro do prazo comercial de entrega estabelecido pelo projeto (PRESSMAN, 2006);
- O cliente não precisa receber todo o sistema para poder usá-lo. Como a implementação é feita por pedaços ordenados por prioridades, o cliente pode ter acesso às partes mais importantes antes, podendo utilizá-las imediatamente;



# Desenvolvimento Incremental

## Desvantagens:

- Os problemas são particularmente críticos para os sistemas grandes, complexos e com vida-longa, onde várias equipes desenvolvem diferentes partes do sistema;
- Sistemas de grande porte necessitam de um framework ou arquitetura estável, e as responsabilidades das diferentes equipes de trabalho do sistema precisam ser claramente definidas, respeitando essa arquitetura.



# O desenvolvimento iterativo

Ocorre quando grupos de recursos são especificados, projetados, construídos e testados juntos em uma série de ciclos, geralmente com duração fixa.

Iterações podem envolver mudanças em recursos desenvolvidos em iterações anteriores, juntamente com mudanças no escopo do projeto.



# O desenvolvimento iterativo

Cada iteração fornece software funcional que é um subconjunto crescente do conjunto geral de recursos até que o software final seja entregue ou o desenvolvimento seja interrompido.



# O desenvolvimento iterativo

Modelos de desenvolvimento iterativos:

- Espiral: envolve a criação de incrementos experimentais, alguns dos quais podem ser muito retrabalhados ou até mesmo abandonados em trabalhos subsequentes de desenvolvimento.



# O desenvolvimento iterativo

## Modelos de desenvolvimento iterativos:

- Scrum: cada iteração tende a ser relativamente curta (p. ex., horas, dias ou algumas semanas) e os incrementos de recursos são correspondentemente pequenos, como alguns aprimoramentos e / ou dois ou três novos recursos;



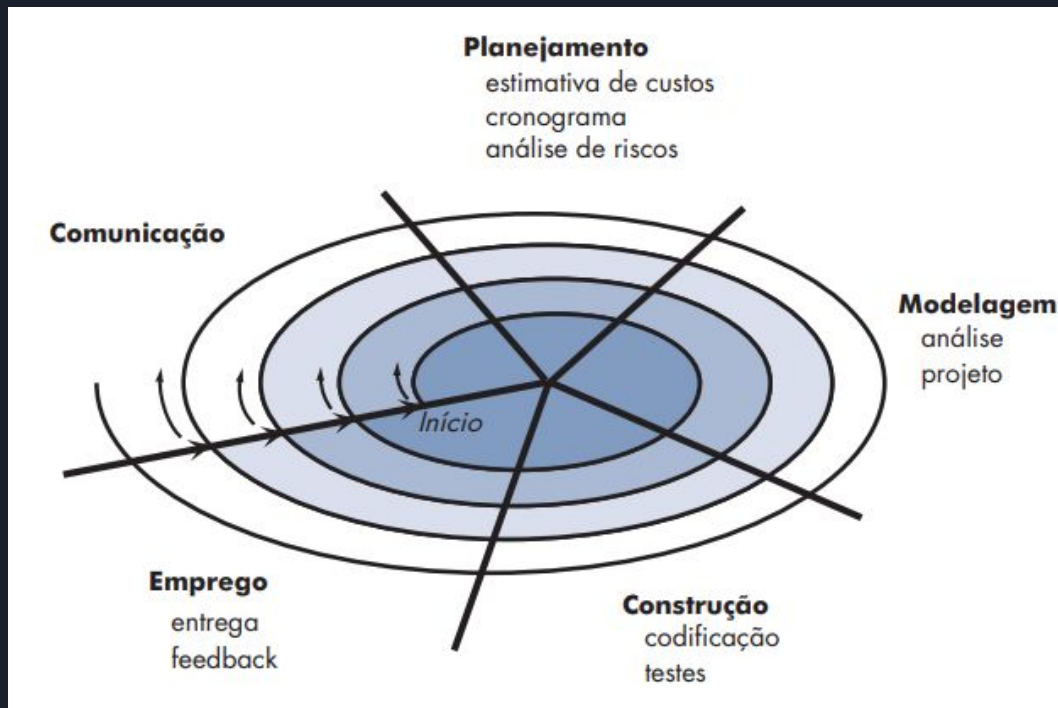


# Modelo Espiral

Modelos de desenvolvimento iterativos:

O **modelo espiral** é um modelo de processo de software evolucionário que acopla a natureza iterativa da prototipação com os aspectos sistemáticos e controlados do modelo cascata (PRESSMAN, 2011).

# Modelo Espiral





# Modelo Espiral

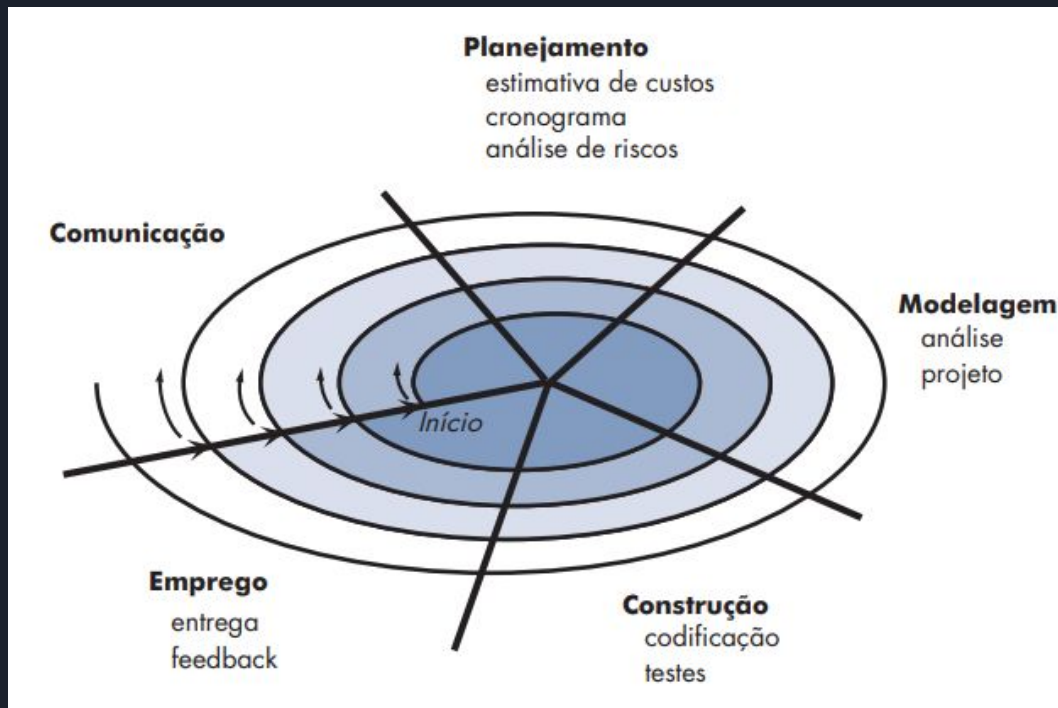
Usando-se o modelo espiral, o software será desenvolvido em uma série de versões evolucionárias. Nas primeiras iterações, a versão pode consistir em um modelo ou em um protótipo. Já nas iterações posteriores, são produzidas versões cada vez mais completas do sistema que passa pelo processo de engenharia.



# Modelo Espiral

- O modelo espiral é dividido em uma série de atividades de trabalho ou regiões de tarefa.
- Existem tipicamente de 3 a 6 regiões de tarefa, estas são definidas pela equipe de engenharia de software;
- Cada uma dessas atividades representa um segmento do caminho espiral

# Modelo Espiral





# Modelo Espiral

- Assim que esse processo evolucionário começa, a equipe de software realiza atividades indicadas por um circuito em torno da espiral no sentido horário, começando pelo seu centro
- Os riscos são considerados à medida que cada revolução é realizada.

Risco : Segundo o Project Management Body of Knowledge (PMBOK, 2017), é um evento ou condição incerta que, se ocorrer, terá um efeito positivo (oportunidade) ou negativo (ameaça) sobre pelo menos um objetivo do **projeto** envolvendo tempo, custo, escopo ou qualidade.



# Modelo Espiral

- O primeiro circuito em volta da espiral pode resultar no desenvolvimento de uma especificação de produto;
- Passagens subsequentes em torno da espiral podem ser usadas para desenvolver um protótipo e, então, progressivamente, versões cada vez mais sofisticadas do software;
- Cada passagem pela região de planejamento resulta em ajustes no planejamento do projeto.



# Modelo Espiral

- Custo e cronograma são ajustados de acordo com o feedback (a realimentação) obtido do cliente após entrega





# Modelo Espiral

- engloba as melhores características do ciclo de vida Clássico e da Prototipação, adicionando um novo elemento: a Análise de Risco
- segue a abordagem de passos sistemáticos do Ciclo de Vida Clássico incorporando-os numa estrutura iterativa que reflete mais realisticamente o mundo real
- usa a Prototipação, em qualquer etapa da evolução do produto, como mecanismo de redução de riscos



# Modelo Espiral

## Vantagens:

- é, atualmente, a abordagem mais realística para o desenvolvimento de software em grande escala.
- usa uma abordagem que capacita o desenvolvedor e o cliente a entender e reagir aos riscos em cada etapa evolutiva



# Modelo Espiral

## Desvantagens

- pode ser difícil convencer os clientes que uma abordagem "evolutiva" é controlável ;
- exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso.



# Scrum

Por definição, o Scrum é um *framework* desenvolvido para que as pessoas possam tratar e resolver problemas complexos e adaptativos, mas ainda assim de maneira produtiva e criativa, entregando produtos que agregam valor ao cliente. Sendo assim, ele não define um processo de desenvolvimento, mas sim boas práticas e atividades a serem seguidas.

Um framework em desenvolvimento de software, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.



# Scrum

Três pontos importantes em relação a teste no Scrum que merecem destaque é:

- Ao fim de cada Sprint (iterações, comumente, 2 ou 4 semanas de desenvolvimento), é esperado que se tenha um “entregável” do produto, ou seja, uma versão pronta para ser usada (o que não significa, necessariamente, que será entregue ao cliente);



# Scrum

Três pontos importantes em relação a teste no Scrum que merecem destaque é:

- O time Scrum deve ser multidisciplinar e todos devem seguir o conceito de pronto, que é um conceito definido pelo próprio time a fim de garantir que o trabalho está completo no incremento da Sprint;

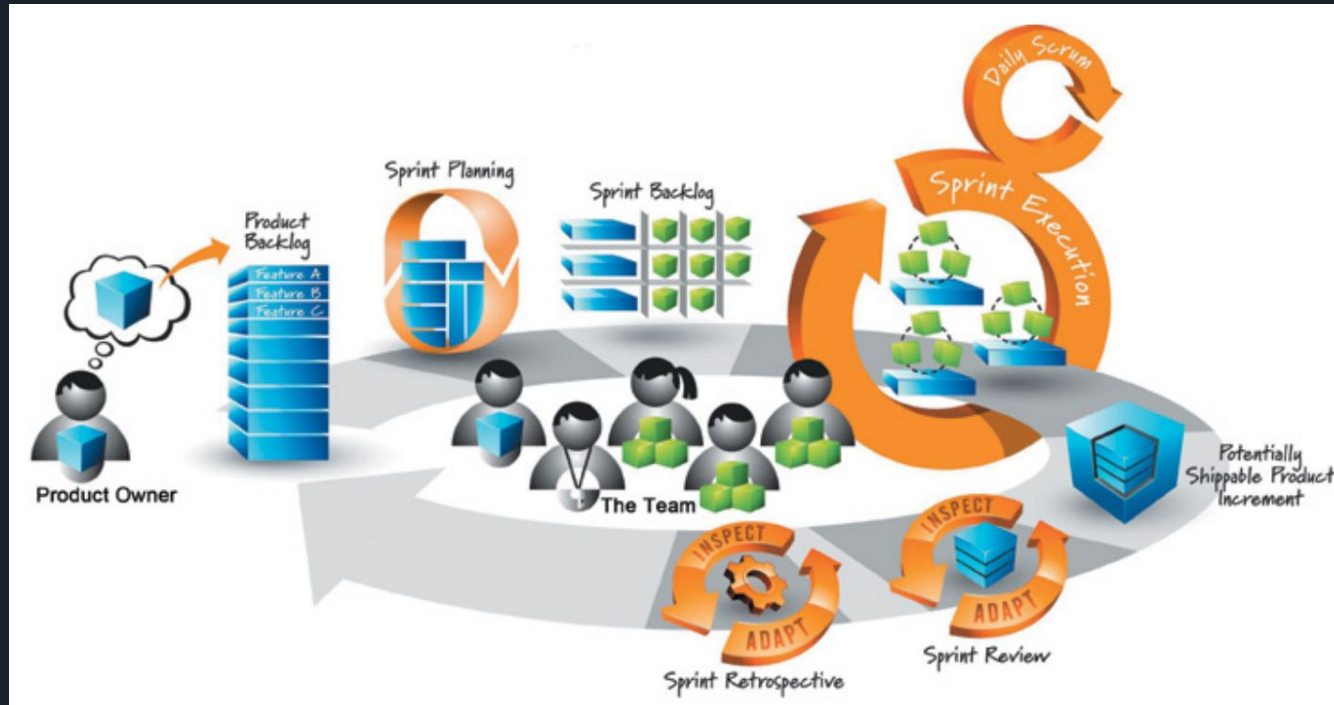


# Scrum

Três pontos importantes em relação a teste no Scrum que merecem destaque é:

- Considerando que todos estão envolvidos com o teste, o empenho em garantir que apenas requisitos testáveis e bem especificados façam parte do Sprint backlog, que é compartilhada por todos, afinal, eles terão que testar aquele requisito assim que ele estiver pronto, antes de terminar a Sprint

# Scrum







# Questões

- 1-Quais são os tipos de ciclos de vida mais utilizados em desenvolvimento de software?
- 2-Explique de maneira breve 2 modelos de desenvolvimento de software ensinados na aula de hoje, citando suas vantagens e desvantagen.
- 3-Cite um modelo de desenvolvimento não mencionado na aula de hoje junta uma breve explicação.