**METODOLOGIAS ÁGEIS E O CICLO DE DESENVOLVIMENTO DE SOFTWARE**

**PROCESSOS DE SOFTWARE**

-O processo de desenvolvimento de um software envolve metodologias para atividades, ações e tarefas necessárias – é como se fosse um roteiro que precisa ser seguido.

-Todos os envolvidos são direta ou indiretamente responsáveis pelo ciclo de desenvolvimento do software. Isso gera controle, estabilidade e organização no contexto.

- Os processos são adaptáveis de acordo com o produto a ser construído.

**PRINCIPAIS ATIVIDADES DO PROCESSO:**

1. Comunicação
2. Planejamento
3. Modelagem
4. Construção
5. Entrega

- O primeiro passo é a organização do processo e suas relações = fluxo

- Para cada estágio do processo há uma interação com as outras etapas = fluxo de processo interativo

- As etapas podem até acontecer sequencialmente, mas é normal que aconteçam paralelamente também.

**PADROES DE PROCESSO**

-Quando se vê um problema dentro do processo, define-se um padrão para resolução: Descreve um problema de processo e sugere soluções de acordo com o contexto. Já existem certos padrões de resoluções – templates – que podem ser usados para cada tipo de problema/contexto.

- Pode ser definido para qualquer nível de abstração

- Quando se trabalha com os padrões, é preciso também avaliar a qualidade desses processos. Para isso são usadas as Abordagens de avaliação e aperfeiçoamento:

. SCAMPI

.CBA IPI

.SPICE (ISSO/IEC 15504)

.ISO 9001:2000

**MODELO PRESCRITIVO (Clássico-Tradicional)**

- Também conhecido como modelo clássico ou tradicional – pois antes não existia modelos ágeis.

- Foco na ordem (começo, meio e fim) e consistência do processo.

-Prescrevem conjuntos de elementos de processos e fluxos –prescreve um roteiro de passos.

**MODELO CASCATA-CLÁSSICO**

Nesse modelo é feito um passo a passo linear, sistemático, na ordem, em sequência. É útil para requisitos bem compreendidos, definidos e estáveis.

1. Comunicação: Início do processo com levantamento de requisitos
2. Planejamento: Estimativas, cronograma e acompanhamento
3. Modelagem: análise e projeto
4. Construção: código, teste
5. Entrega: entrega do projeto, suporte e feedback

**MODELO V**

-É também uma espécie de modelo cascata mas no modelo V, para cada etapa de construção há também uma etapa de verificação e validação dos testes.

PROBLEMAS DOS MODELOS CASCATA:

. Dificilmente se vê em projetos reais fluxos que sejam sequenciais

. Esse modelo não lida bem com adaptação constante de mudanças

. Requisitos não são bem estabelecidos na primeira fase

. Longo tempo para visualizar primeira versão do software

. Gera estados de bloqueio para a equipe

**MODELOS INCREMENTAL, EVOLUCIONÁRIO E CONCORRENTE**

**MODELO DE PROCESSO INCREMENTAL**

- Para situações com requisitos bem definidos, mas ainda não refinados. É feito ainda num modelo sequencial, porém vai-se entregando pequenos incrementos durante o processo

-Pode-se utilizar prototipagem (o uso de protótipo é interessante quando ainda não se tem requisitos refinados ou escopo muito fechado. Ou seja, começa modelando a ideia do projeto com um protótipo e consegue finalizar com um incremento do produto. Depois faz novamente um novo tipo de protótipo e segue.)

- No modelo incremental ainda se tem um modelo em cascata, porém com a diferença de que se tem pequenas entregas através de protótipos e incrementos.

**MODELO EVOLUCIONÁRIO**

- Modelo que possibilita o desenvolvimento de um software que cresce e se adapta constantemente

- São interativos

- Modelos:

. Prototipagem

-Muito útil para refinar requisitos, validar eficiência e interação com usuário, pode ser aplicado isoladamente ou em conjunto com outros processos, o protótipo atua como forma de obtenção de requisitos, podem ser descartáveis ou evoluir.

- Problemas: falta de consideração da qualidade global do software após prototipação ser considerada funcional; acomodar com escolhas iniciais da prototipação. Ex: a arquitetura não está tão boa mas está funcional e o cliente gostou de primeira, e fica assim..

. Espiral

- Natureza interativa da prototipação + aspectos sistemáticos do cascata

- Estratégia cíclica incremental com foco em diminuir riscos

**MODELO CONCORRENTE**

-Representação concorrente de atividades de qualquer processo

**MODELOS ESPECIALIZADOS**

-Modelos usados em contextos bem específicos

**BASEADO EM COMPONENTES** (é comum usa-los)

-Desenvolvimento baseado em componentes com interfaces definidas p serem integradas ao software -> COTS / módulos ou pacotes de classes – usando bibliotecas. Ex: vai montando e encaixando como peças esses componentes ao software-produto

-Evolucionário por natureza

-Foco em reutilização – redução no tempo de desenvolvimento e custos

**MODELO DE MÉTODOS FORMAIS** (não é muito comum usar)

-Atividades baseadas em condução à especificação matemática formal do software – utilização de notação matemática

- Analise matemática auxilia na descoberta de ambiguidades ou inconsistências

- Desenvolvimento consome tempo e dinheiro

-Complexidade exige formação e treinamento

-Bem visto para softwares com fatores críticos

**MODELO ORIENTADO A ASPECTOS**

- Modelo que vem de um paradigma que oferece abordagem metodológica e de processos para definir, especificar, projetar, e construir aspectos que são pontos de interesse que se propagam e entrecortam outras partes da aplicação.

**PROCESSO UNIFICADO**

1. FASE DE CONCEPCAO

Comunicação e planejamento

Requisitos são descritos em conjunto de casos de uso de preliminares

Identificação de recursos, riscos e cronograma

1. FASE DE ELABORACAO

Planejamento e modelagem

Refinamento e expansão de casos de uso

Ampliação de representação arquitetural- ou seja, diagramas mais definidos: casos de uso, modelo de análise, modelo de projeto, modelo de implementação, modelo de disponibilização

1. FASE DE CONSTRUÇÃO

Desenvolvimento de software com base nos modelos

Uso dos modelos para gerar suíte de testes de aceite

Utilização de testes conforme desenvolvimento

1. FASE DE TRANSIÇÃO

Construção e entrega – é um processo de transição final

É comum entrega de testes beta para recebimento de feedbacks

O incremento torna-se uma versão utilizável do software

1. FASE DE PRODUÇÃO

Entrega

Monitoramento de uso continuo

Suporte

Relatórios para defeitos e mudanças

**DESENVOLVIMENTO ÁGIL**

**AGILIDADE** é mais uma cultura do que um processo. Condutores da agilidade: adaptação + comunicação + auto-organização

A adaptação auxilia na diminuição de custos por alterações.

**EXTREME PROGRAMMING – XP** (programação extrema)

-Criado baseado no paradigma de orientação a objetos

-Envolve regras e práticas constantes durante o processo de software

PLANEJAMENTO:

-Ouvir o cliente, foco na comunicação a partir do “planning poker”. Levantamento de requisitos, critérios, etc.

-A atividade leva a criação de histórias do usuário pelo cliente, que também as prioriza

-Membros estimam com base em semanas de desenvolvimento – o ideal é no máximo 3 semanas

- Há uma flexibilização para escrita de novas historias

- Clientes e devs trabalham lado a lado – compromisso básico

- Velocidade: nº de histórias entregues (isso é avaliado como passar dos ciclos)

PROJETO:

- Principio KISS (Keep it simple, stupid!)

- Estimulo no uso de cartões CRC (Classe-responsabilidade-colaborador) – interessante para entender como vai funcionar o fluxo do código

- Solução pontual a partir de protótipos

CODIFICAÇÃO:

- Uso de TDD (Test Driven Development) – é um tipo de teste de unidade

- Refatoração (aperfeiçoamento de código)

- Programação em pares (quando uma pessoa programa e a outra vai acompanhando, revisando o código, dando novas ideias, é possível fazer um rodizio.)

TESTES:

- Uso de TDD

-Integração continua

-Inclusão de testes de aceite – histórias do usuário

**METODOLOGIA SCRUM**

- Funciona para qualquer tipo de projeto

-Baseado no lean thinking (pensamento enxuto). É interativo e incremental

- Eventos formais para inspeção e adaptação

-Pilares: transparência, adaptação e inspeção

PROCESSO:

1. História do cliente: Junto ao PO, a squad começa com uma ideia do produto, ouvindo as histórias do usuário para entender as funcionalidades requeridas para o produto e vai colocar tudo isso no BACKLOG DO PRODUTO.
2. PLANNING PRODUCT: Um evento onde o time se reúne para definir quais são as metas da Sprint: o que vão entregar, o que precisa ser feito, quais são os prazos (uma Sprint leva em média 2-4 semanas, etc.)
3. DAILYS: Reuniões por dia para ir alinhando o trabalho
4. SPRINT REVIEW + SPRINT RETROSPECTIVE
5. ENTREGA: Após todas as sprints entregues e dailys feitas, o incremento é entregue para a produção já validado pelo cliente.

**SCRUM TEAM**: Pequeno time de pessoas trabalhando juntas em um projeto sem hierarquia

-Scrum Master: se preocupa em manter a metodologia Scrum – direcionando, alinhando, ajustando, treinando, liderando e mantendo o time.

-Product Owner: maximiza o valor do produto, desenvolve a meta do produto, gerenciando o product backlog, cria comunicações existentes dos itens do backlog para o time, ordena itens.

-Developers: Criam as sprints de backlog ( o que precisa ser feito? Metas), alinhamentos, adaptações.

. Responsáveis por todas as atividades relacionadas aos produtos

SPRINTS: Eventos com duração fixa com objetivo de gerar incremento

-Atividades: Sprint planning, daily, Sprint review, Sprint retrospective

- NÃO se faz mudanças que coloque em risco a meta da Sprint

-Foco na qualidade

-Refinamento conforme necessário

-SOMENTE O PO PODE CANCELAR A SPRINT

NO FINAL: há que se gerar artefatos/documentos como o PRODUC BACKLOG (meta do produto), SPRINT BACKLOG (Meta da Sprint) e o INCREMENTO (Definição de pronto)

**KANBAN**

-FOCO NO ASPECTO VISUAL – COM CARTÕES

- TO DO /DOING/ DONE

-É simples e deve ser aliado com outros frameworks para gerenciamento do projeto

**TESTES NO MUNDO ÁGIL**

-Para cada etapa do ciclo de desenvolvimento do software, tem uma fase de teste também

- Cada nível de teste tem um objetivo especifico

- A análise e modelagem de testes começam durante a fase de desenvolvimento

- Participação no processo de requisitos, modelagem, refinamentos.

**A PARTICIPACAO DO QA NA HISTÓRIA DO USUARIO:**

História do usuário: é a mão na massa de como o usuário vê aquela funcionalidade – o que e como ele deve fazer. Estarão envolvidos aqui tanto requisitos funcionais como não-funcionais. Na perspectiva de testes, para se fazer a verificação de cenários, é importante o QA estar envolvido desde a parte de refinamento, analise de requisitos, etc.

Quando o QA desenvolve cenários, ele já faz pensando nos critérios do cliente e extras (critérios além).

- CONCEITO 3C:

Cartão

Conversação

Confirmação -> os critérios do cliente foram de fato estabelecidos? Critérios de aceite

A perspectiva de quem testa é diferente da do cliente, da do PO, do desenvolvedor...

**ATIVIDADES ENVOLVIDAS NO PLANEJAMENTO DO QA:**

-Análise detalhada das histórias do usuário

-Determinar testabilidade da história (saber testar e executar uma funcionalidade mas também fazer uma previsão do que pode ser testado é importante)

-Criar testes de aceite (vem diretamente da história do usuário).

-Estimar esforço

-Identificar aspectos funcionais e não-funcionais a serem avaliados

- Participar do processo de automação (participar dos testes de unidade)

**DETALHANDO A ABORDAGEM DE TESTE**

-Determinar escopo, extensão, objetivos e razões para testes (isso é feito através de comunicação entre a equipe com uma avaliação inicial.)

-Membros que irão atuar

-Ambiente e dados necessários

-Tempo, dependência E pré-requisitos

-Ricos envolvidos

**DIFERENÇAS NAS ABORDAGENS DE TESTES**

-As atividades de teste estão sempre relacionadas ao desenvolvimento, portanto, é importante conhecer os diversos processos e ciclos de vida e como a atividade de testes e qualidade pode se inserir

-Cada empresa adota um processo e customiza de acordo com suas necessidades

-Adaptação é a palavra-chave

- As atividades de qualidade e testes podem ocorrer paralelamente e em constante interação com outras atividades (Ou seja, desde a reunião de planning isso já começa. As atividades não acontecem de forma sequencial, é sempre preciso conversar com o time para saber qual o próximo passo).

-Cada pessoa do time tem atuação direta na validação e verificação

-Foco nos testes de segurança, performance e exploratórios

-Uso de automação para testes de regressão

- Documentação suficiente para manutenção e garantia de qualidade

**PRODUTOS DE TRABALHO COMUNS**

-Testes automatizados -> resultados

-Planos de testes

-Análise de risco (conversar com o PO ou com os devs e avaliar os riscos de determinada abordagem)

-Evidências de testes manuais

-Relatórios de defeitos

**NÍVEIS DE TESTES**

-São sobrepostos

-O Foco na agilidade é:

Testes de unidade (por ex testes de API, testes de sistema)

Testes de aceite

Verificação

Validação

-Uso de integração e entrega continua + automação de testes (Alinhar a pipeline do código com automação)

**STATUS DE TESTES NO MODELO ÁGIL**

-Adaptações e análise crítica p definir o que efetivamente está concluído (ver o que está “done”)

-Atualização frequente de testes manuais e automatizados

-Monitorar status de todas as atividades da equipe -> foco no feedback

-Reuniões diárias p comunicação (dailys)

**MÉTODOS DE TESTES NOS MODELOS ÁGEIS**

TDD (Desenvolvimento orientado por teste) – é um método de desenvolvimento de software que se inicia por testes

-É difícil mas muito bem visto. Começa pelo teste (que inicialmente vai ser falho) e depois vai escrevendo o código, até ir passando nos testes.

ATDD (Desenvolvimento orientado por teste de aceite)

BDD (Desenvolvimento orientado a comportamento)

DEVOPS

É uma cultura que afeta diretamente a agilidade

**PRÁTICAS ÚTEIS PARA TESTES**

-Teste assistido (junto ao PO ou dev ou outro QA)

-Testes incrementais (inicia de forma mais simplifica e vai refinando com o passar da Sprint, fazendo melhorias ao longo)

-Mapa mental: fazer um fluxo mental de estratégias, rascunhos p os testes

. Estratégias

. Cenários

. Dados

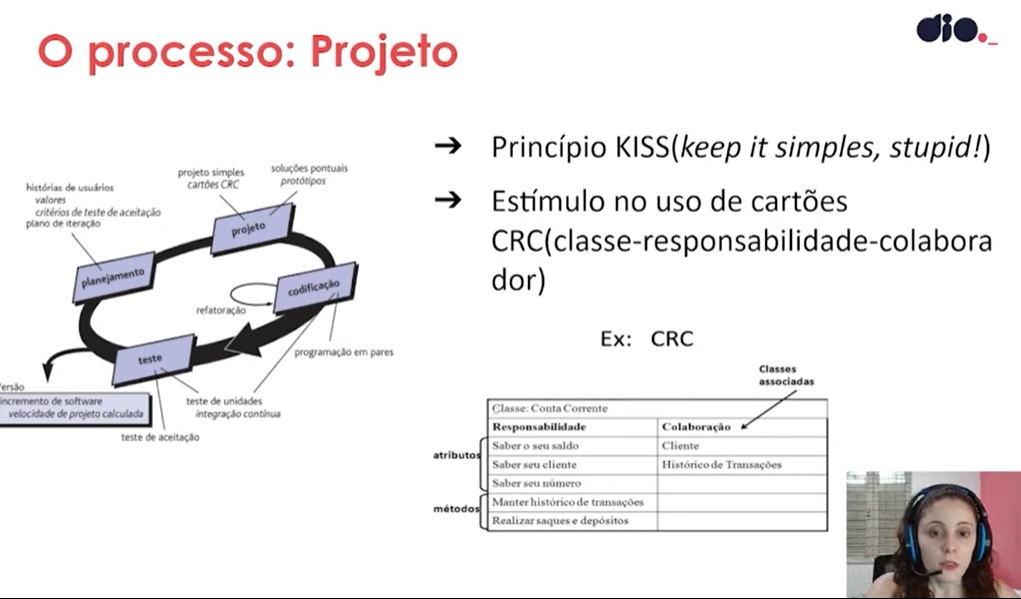
**CERTIFICAÇÕES:**

-CTFL-AT: Agile Tester (+ focado na parte teórica, conceitual de agilidade)

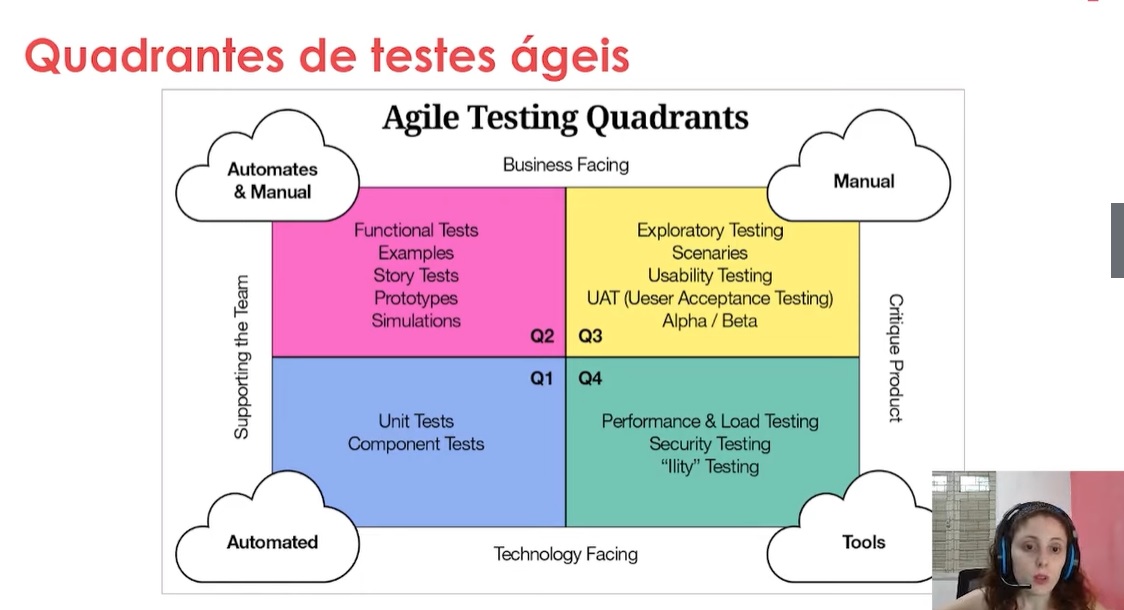
-CTFL-ATT: Agile Tecnichal tester (mais focado na parte prática e técnica)

-

1. **Exemplo cartão CRC para modelagem:**

****

1. **Quadrante de testes ágeis:**

****