Ciclo de desenvolvimento de software e metodologias ágeis

Carolina Santana Louzada (Analista QA – Venturus)

Mais sobre mim

- Graduada em Engenharia de Computação- UFS
- Fazendo especialização em qualidade e desenvolvimento de software
- Qualidade de software -> automação
- Educação + tecnologia
- Jogos + música + aprender novas atividades
- LinkedIn -> Carolina Santana Louzada | LinkedIn

Objetivo do curso: Compreender o ciclo de vida do software e seus processos, bem como entender omo a qualidade e os testes atuam nos processos, tendo como foco o pensamento ágil.

Pré-requisitos: ★ Ter completado curso de Fundamento de Qualidade de Software

Percurso : Processos de software; Desenvolvimento ágil; Testes no mundo ágil.

Aula 1: Processos de software

Objetivos:

- Definição de processo, fluxo e padrões de software
- Compreender modelos prescritivos
- Compreender modelos incremental, evolucionário e concorrente
- Compreender modelos mais especializados
- Compreender o processo unificado (RUP)

Aula 1 . Etapa 1 - Definindo processo, fluxo e padrões de software.

"...o desenvolvimento de software é um processo de aprendizado social. Trata-se de um processo iterativo no qual a própria ferramenta em evolução serve como meio de comunicação..."

Citação de Howard Baetjer Jr. (apud Pressman, Roger S., Maxim, Bruce R. Engenharia de Software: uma abordagem profissional, 8ªed)

O que é processo de software?

Metodologia para as atividades, ações e tarefas necessárias para desenvolver um software.

Algumas reflexões...

➔ Todos os envolvidos são diretamente ou indiretamente responsáveis.

➔ Gera estabilidade, controle e organização dentro do contexto.

➔ Processos são adaptáveis de acordo com o produto a ser construído.

Atividades principais do processo (Pressman e Sommerville)

Segundo Pressman:
- ➔ Comunicação
- ➔ Planejamento
- ➔ Modelagem
- ➔ Construção
- ➔ Entrega

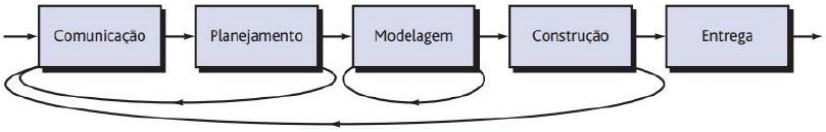
Segundo Sommerville:
- ➔ Especificação
- ➔ Projeto e implementação
- ➔ Validação
- ➔ Evolução

Atividades principais do processo

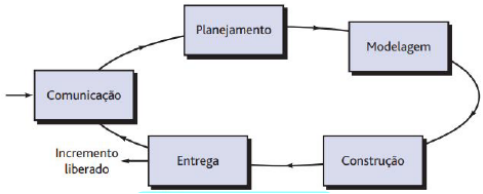
➔ Organização dos processos e suas relações = fluxo do processo

A)
Fuxo de processo linear

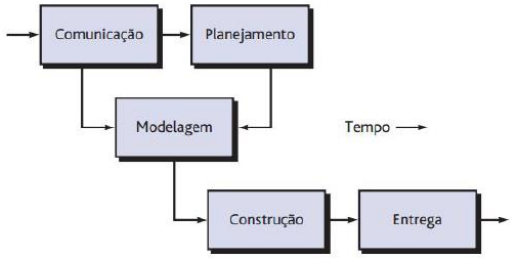
B) **Fluxo de processo interativo**



C) **Fluxo de processo evolucionário**



D) **Fluxo de processo paralelo**



Cada processo é constituído de tarefas!

Padrões de processo

➔ Descreve um problema de processo e sugere soluções de acordo com o contexto
➔ Pode ser definido para qualquer nível de abstração
➔ Abordagens de avaliação e aperfeiçoamento:
◆ SCAMPI(Standard CMMI Assessment Method for Process Improvement)
◆ CBA IPI(CMM-Based Appraisal for internal Process Improvement)
◆ SPICE(ISO/IEC 15504) - requisitos para avaliação de processos de software
◆ ISO 9001:2000

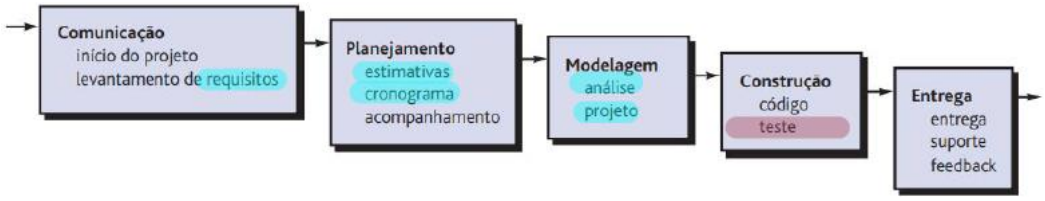
Aula 1 . Etapa 2 - Modelo prescritivo

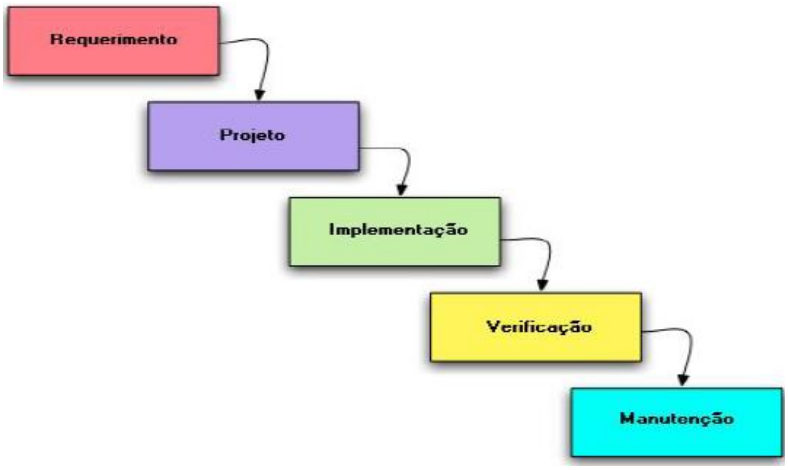
O que é?

★ Também chamado de modelos “tradicionais”
★ Foco na ordem e consistência do processo
★ Prescrevem conjunto de elementos de processo e fluxos

Modelo Cascata – Clássico

➔ Útil para requisitos bem compreendidos,definidos e estáveis
➔ Processo linear e sistemático





Modelo Cascata - Problemas

➔ Projetos reais não seguem fluxos sequenciais

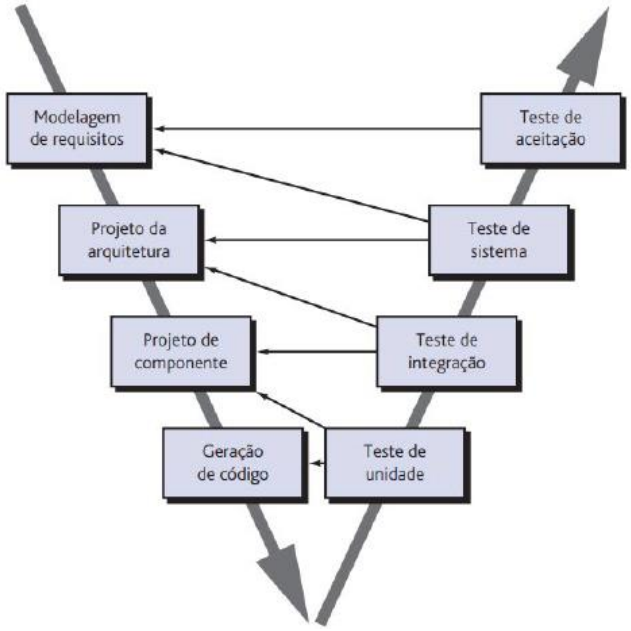
➔ Não lida bem com adaptação constante de mudanças

➔ Requisitos não são bem estabelecidos na primeira fase

➔ Longo tempo para visualizar primeira versão do software

➔ Gera estados de bloqueio para a equipe

Modelo V



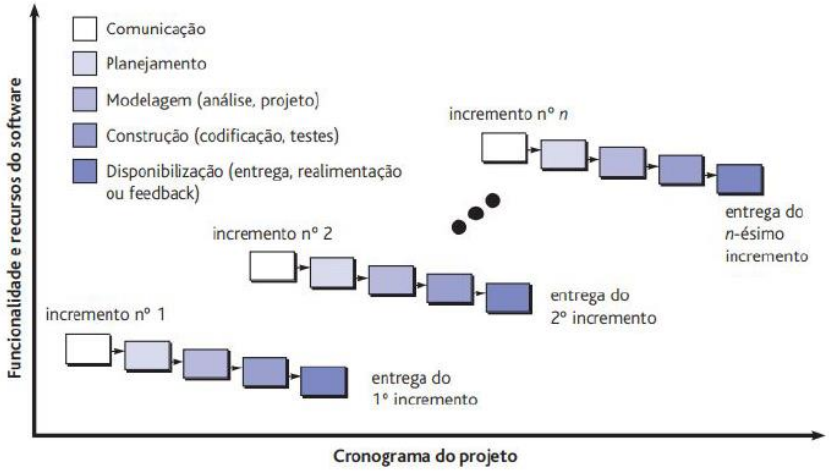
➔ Relação entre atividades de garantia de qualidade e atividades restantes do processo

➔ Não há diferença fundamental entre o Cascata e V

Aula 1 . Etapa 3 - Modelos incremental, evolucionário e concorrente

Modelos de processo incremental:

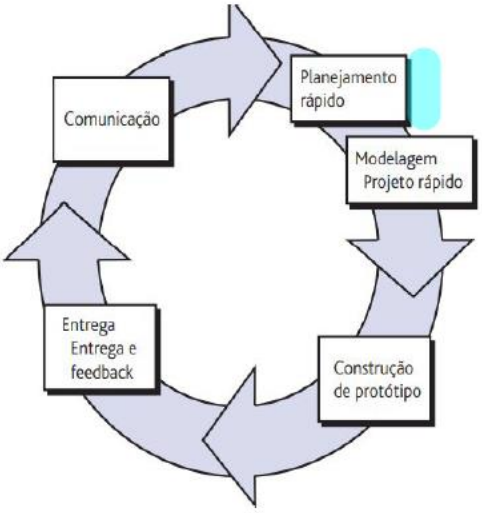
➔ Situações com requisitos iniciais bem definidos, mas não Refinados (bem definidos e não refinados; sequencia linear. prototipagem: modelar a ideia com uso de protótipo; não deixa de ser cascata, mas pequenas versões de software sao entregues)



★ Pode-se utilizar prototipagem

Modelo evolucionário:

★ Modelo que possibilita o desenvolvimento de um software que cresce e se adapta constantemente
★ São iterativos
★ Modelos: Prototipagem e Espiral

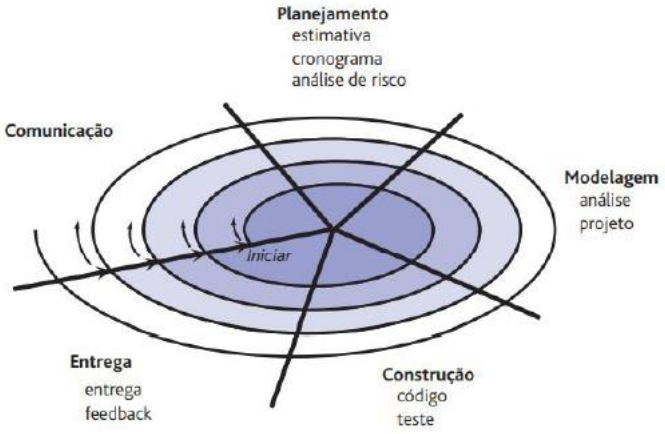


Modelo evolucionário – Prototipação

★ 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 podem evoluir

Problemas da Prototipação

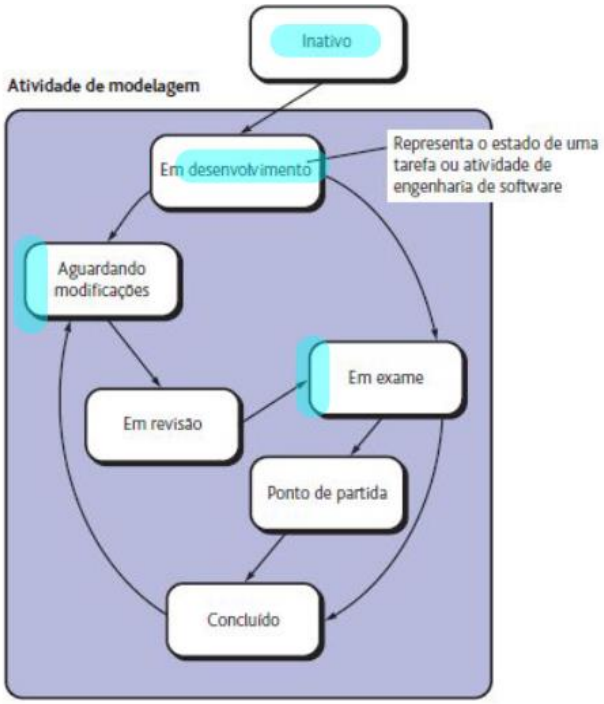
★ Falta de consideração da qualidade global do software após prototipação
ser considerada "funcional"
★ Acomodar com escolhas iniciais da prototipação (Não tá bom, mas funciona)



Modelo evolucionário – Espiral

★ Natureza iterativa da prototipação + aspectos sistemáticos do cascata
★ Estratégia cíclica incremental com foco em diminuir riscos
(utilizado em sistemas complexos; prototipação; replanejamento para o próximo ciclo, onde pode haver melhorias de acordo com o feedback do cliente)

Modelo concorrente



★ Representação concorrente de atividades de qualquer processo

Aula 1 . Etapa 4 - Modelos especializados

Baseado em componentes

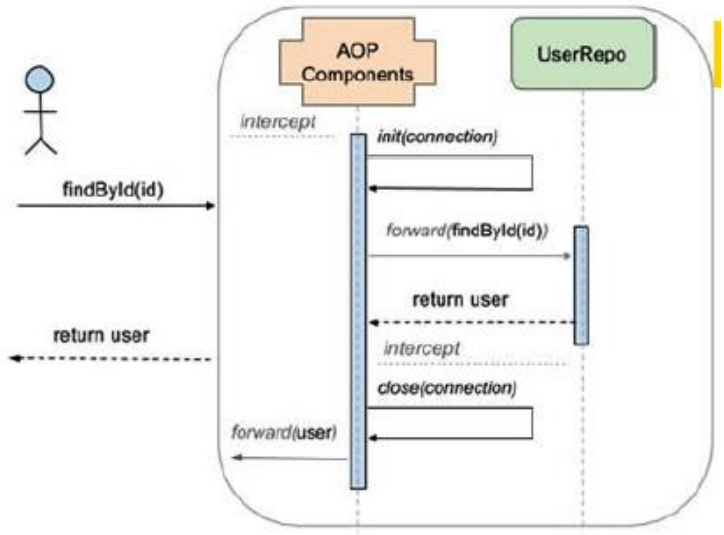
★ Desenvolvimento com base em componentes com interfaces definidas para serem integradas ao software -> COTS(commercial off-the-shelf) o módulos ou pacotes de classes
★ Evolucionário por natureza
★ Foco em reutilização -> Redução no tempo de desenvolvimento e custos

Modelo de métodos formais

★ Atividades baseadas em condução à especificação matemática formal do software -> utilização de notação matemática Engenharia de software sala limpa
★ Análise matemática auxilia na descobertas 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

★ Paradigma que oferece uma 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.



Aula 1 . Etapa 5 - Processo Unificado

Um pouco de história

★ No início dos anos 90 , James Rumbaugh, Grady Booch e Ivar Jacobson começaram a trabalhar em um "método unificado" que combinasse as melhores características de outros processos -> UML (`Unified Modeling Language - para a elaboração da estrutura de projetos de software`)

★ Necessidade de um processo de software dirigido a casos de uso, centrado na arquitetura, iterativo e incremental.

Fases do processo unificado

1. Fase de Concepção: Comunicação e Planejamento

➔ Requisitos são descritos em conjunto de casos de uso preliminares.

➔ Identificação de recursos, riscos, cronograma...

2. Fase de Elaboração Planejamento e Modelagem

➔ Refinamento e expansão de casos de uso

➔ Ampliação de representação arquitetural:

◆ casos de uso
◆ modelo de análise
◆ modelo de projeto
◆ modelo de implementação
◆ modelo de disponibilização

3. 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

4. Fase de Transição: Construção e Entrega

➔ Comum entrega com testes beta para recebimento de feedbacks;

➔ O incremento torna-se uma versão utilizável do software

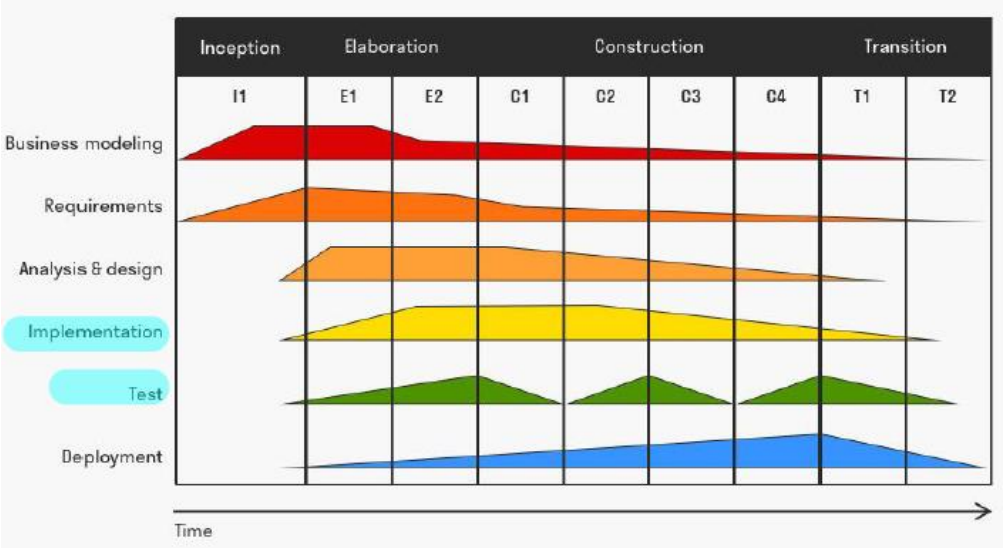
5. Fase de Produção: Entrega

➔ Monitoramento de uso contínuo

➔ Suporte

➔ Relatórios para defeitos e mudanças

Fases do processo unificado



Para saber mais:

➔ Modelo de processo pessoal e de equipe <https://www.geeksforgeeks.org/personal-software-process-psp/>

➔ Pesquise sobre os autores do Manifesto Ágil!

➔ Programação orientada a Aspectos

Aula 2 - Desenvolvimento ágil

Objetivos:

- Contexto da criação do manifesto ágil e seus conceitos
- Compreender o Extreme Programming - XP
- Compreender o Scrum
- Compreender a existência e conceitos de outros modelos ágeis

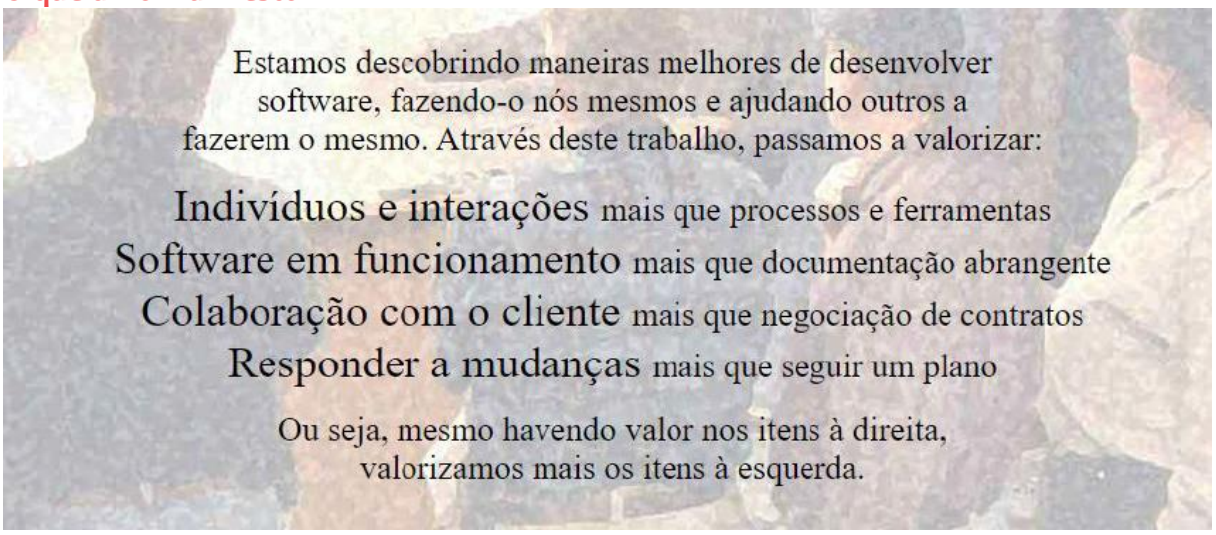
Aula 2 . Etapa 1 - O manifesto ágil

Contexto e surgimento

★ Em contramão aos ditos métodos tradicionais (prescritivos) ou "pesados" , 17 profissionais que já praticavam os "métodos leves" se reuniram em Utah no ano de 2001 e chegaram ao consenso de métodos e práticas para o desenvolvimento de software -> MANIFESTO ÁGIL (documento)

Seus autores:

O que diz o manifesto?



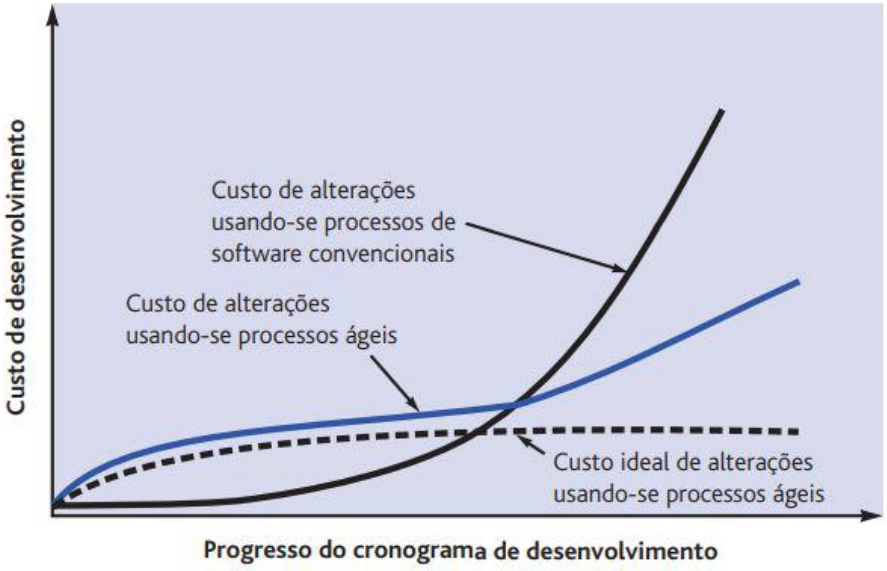
Fonte: agilemanifesto.org

Os 12 princípios

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
7. Software funcionando é a medida primária de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Contínua atenção à excelência técnica e bom design aumenta a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

O que significa ser ágil ?

★ Métodos prescritivos não deixam de ser úteis, porém tem um ponto que pode torná-lo falho: as fraquezas e falhas de quem desenvolve o software!



➔ Condutores da agilidade: adaptação + comunicação + auto-organização

➔ A adaptação auxilia na diminuição de custos por alterações

Motivações

➔ Difícil prever requisitos de softwares e suas possíveis alterações
➔ Difícil prever priorizações do cliente
➔ Análise, projeto e testes não são previsíveis
➔ As atividades de construção do software não são facilmente estimadas

Imprevisibilidade

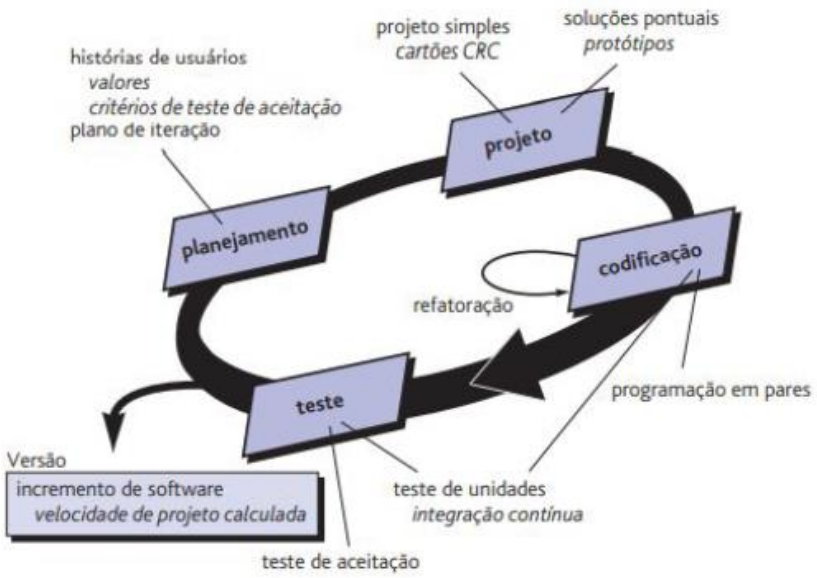
Aula 2 . Etapa 2 - Extreme programming – XP

Surgimento 'Programação Extrema'

➔ Primeiros trabalhos e métodos associados : 1980
➔ Trabalho originário por Kent Beck
➔ Existe variante com refinamentos para grandes organizações: IXP - Industrial Extreme Programming

Extreme Programming Explained: Embrace Change < https://www.amazon.com.br/Extreme-Programming-Explained-Embrace-Change/dp/0321278658>

O processo



➔ Paradigma foco: orientação a objetos
➔ Envolve regras e práticas constantes durante processo de software

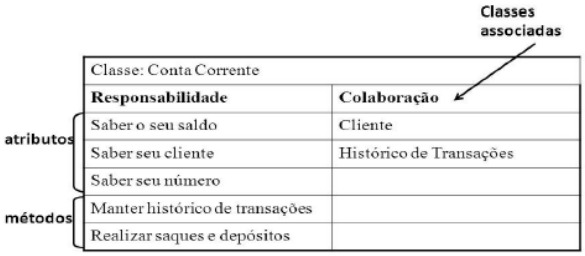
O processo: Planejamento

➔ Foco na comunicação, no 'ouvir' a partir do planning poker (jogo).

➔ A atividade leva a criação de histórias do usuário pelo cliente, que também as prioriza.
➔ Membros estimam com base em semanas de desenvolvimento -> máximo ideal de 3 semanas (*metodologia XP não existe conceito sprint)
➔ Flexibilização para escrita de novas histórias
➔ Clientes e desenvolvedores trabalham lado a lado = compromisso básico
➔ Velocidade: nº histórias entreges

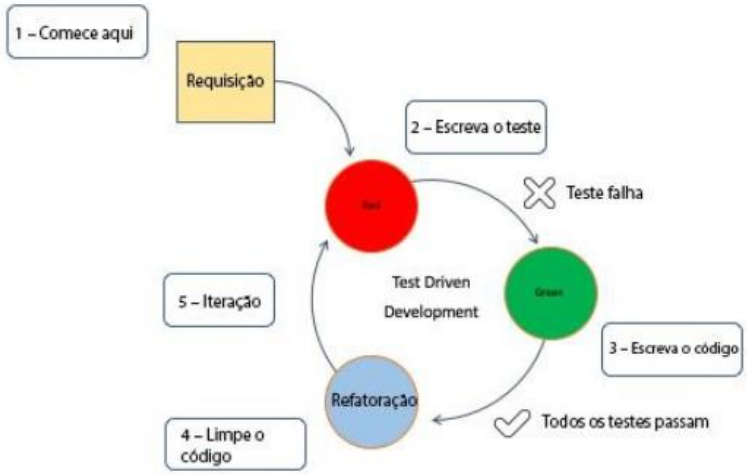
O processo: Projeto

➔ Princípio KISS(*keep it simples, stupid!*)
➔ Estímulo no uso de cartões CRC(classe-responsabilidade-colaborador)



➔ Solução pontual a partir de protótipos

O processo: Codificação



➔ Uso de TDD(*Test Driven Development*)
➔ Refatoração: aperfeiçoamento de código
➔ Programação em pares

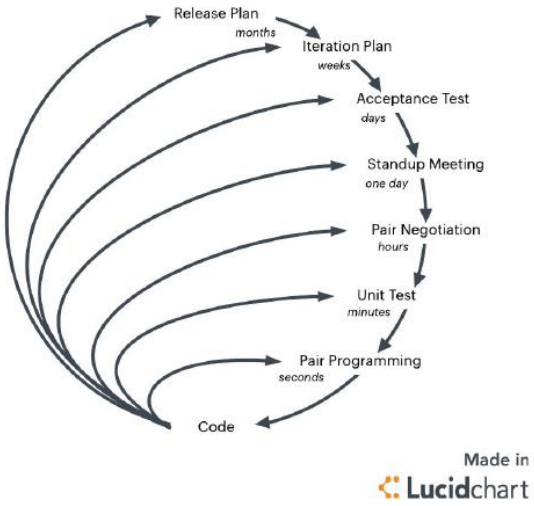
Fonte: Tecmundo(2020) <https://www.tecmundo.com.br/software/155909-5-passos-desenvolvimento-orientado-testes-tdd.htm>

O processo: Testes

➔ Uso de TDD (*Test Driven Development*)
➔ Integração contínua
➔ Inclusão de testes de aceite -> histórias do usuário

Valores

Fonte: RedSpark(2016)



Planning and Feedback Loops

Extreme Programming: A Gentle Introduction. <http://www.extremeprogramming.org>

Aula 2 . Etapa 3 - Scrum

Scrum: Nome provém de uma ação em partida de rugby: jogadores dos dois times se juntam com a cabeça abaixada e se empurram para obter posse de bola.

Surgimento e teoria

➔ Criado por Jeff Sutherland no início dos anos 1990 e desde então vem sendo revisado

➔ "Framework leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos."

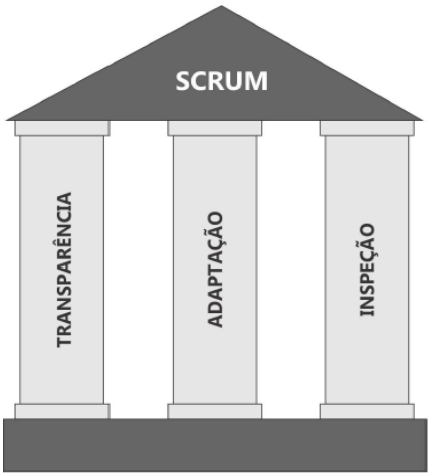
➔ Baseado no empirismo e lean thinking

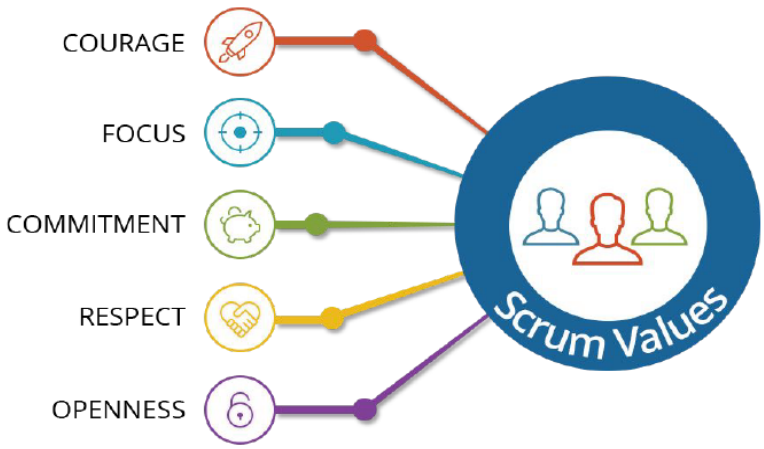
➔ Iterativo e incremental

➔ Eventos formais para inspeção e adaptação

Scrum Guide <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100>

Pilares e valores





Processo

O Scrum Team

★ Pequeno time de pessoas sem hierarquia -> Produto
o *Scrum master*
o *Product owner*
o *Developers*
★ São multifuncionais e autogerenciáveis
★ Responsáveis por todas as atividades relacionadas ao produtos

Product Owner

★ Maximiza valor do produto
★ Gerenciamento do Product Backlog:
o Desenvolver e expressar meta do produto
o Criar e comunicar itens do Backlog
o Ordenar itens
o Garantir que Product Backlog seja transparente, visível e compreensível

Desenvolvedores

★ Criação de incremento utilizável a cada sprint
★ Habilidades amplas de acordo com domínio do trabalho
★ Responsabilidades:
o Criar Sprint Backlog
o Alinhar a definição de Pronto
o Adaptação com direção à meta da Sprint
o Responsabilizar-se como profissionais

Scrum Master

★ Guardião do Scrum -> eficácia da metodologia
★ Liderança que serve à organização
★ Responsabilidades:
o Treinar membros para auto-gerenciamento
o Concentração do time
o Remoção de impedimentos
o Manutenção de eventos
o Auxilia PO com técnicas e melhorias no gerenciamento do Backlog

Eventos: A Sprint

★ Eventos de duração fixa com objetivo de gerar incremento
★ Atividades = Sprint Planning + Daily + Sprint Review e Sprint Retrospective
★ Não se faz mudanças que coloque em risco a meta da Sprint
★ Foco na qualidade
★ Refinamento conforme necessário
★ Somente PO pode cancelar a Sprint

Eventos : Planning
★ Inicia sprint : definição do trabalho a ser realizado
★ Porque essa sprint é valiosa?
★ O que pode ser feito nesta Sprint?
★ Como o trabalho será realizado?
★ Criação do Sprint Backlog

Eventos : Daily
★ Inspeção em direção à meta da sprint
★ Adaptação do Sprint Backlog
★ Curta duração/diariamente
★ Plano de trabalho
★ Comunicação
★ Remoção de impedimento

Eventos : Sprint review
★ Apresentar e inspecionar resultados
★ Determinar adaptações
★ Ajuste no Product Backlog
★ Scrum Team + Stakeholders
★ Penúltimo evento da sprint

Eventos : Sprint retrospective
★ Planejamento voltado para qualidade e eficácia
★ Inspeção de processos, interações, ferramentas...
★ O que funcionou?
★ O que não funcionou?
★ Como foi resolvido?
★ Conclusão da sprint

Artefatos
★ Product Backlog -> Meta do produto
★ Sprint Backlog -> Meta da sprint
★ Incremento -> Definição de pronto

Aula 2 . **Etapa 4 -** Outros modelos ágeis

Método de Desenvolvimento de Sistemas Dinâmicos(DSDM)
➔ Foco na construção e manutenção de sistemas que satisfaçam restrições de prazo curto por meio da prototipação em ambiente controlado
➔ Analogia com Princípio de Pareto: 80% de uma aplicação pode ser entregue em 20% do tempo que levaria para entregar a aplicação completa

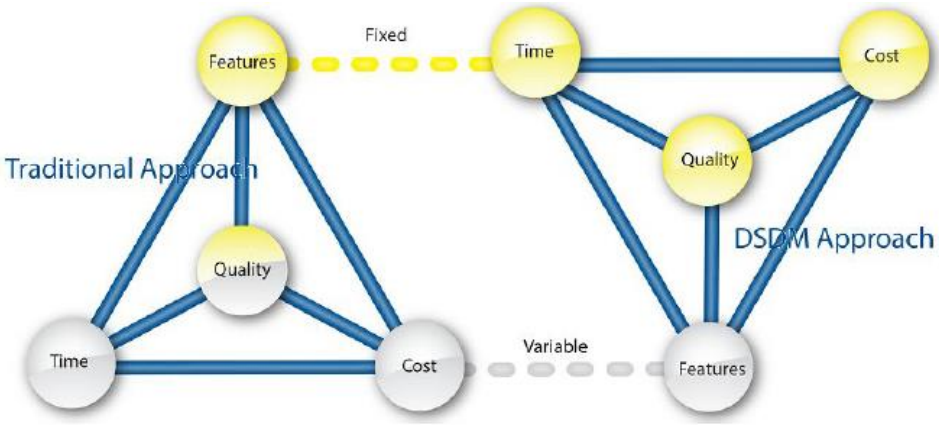
Método de Desenvolvimento de Sistemas Dinâmicos (DSDM)

➔ É iterativo e incremental

➔ Somente o trabalho suficiente é requisitado para cada incremento

➔ Mantenedor -> Agile Business Consortium <https://www.agilebusiness.org/business-agility/what-is-dsdm.html>

➔ Pode ser combinado com XP



Método de Desenvolvimento de Sistemas Dinâmicos (DSDM)

➔ Princípios:

◆ Focar na necessidade do negócio
◆ Entregar dentro do prazo
◆ Colaborar
◆ Nunca comprometer a qualidade
◆ Construir incrementalmente a partir de bases sólidas
◆ Desenvolver iterativamente
◆ Comunicar de forma contínua e clara
◆ Demonstrar controle.

Método de Desenvolvimento de Sistemas Dinâmicos(DSDM)

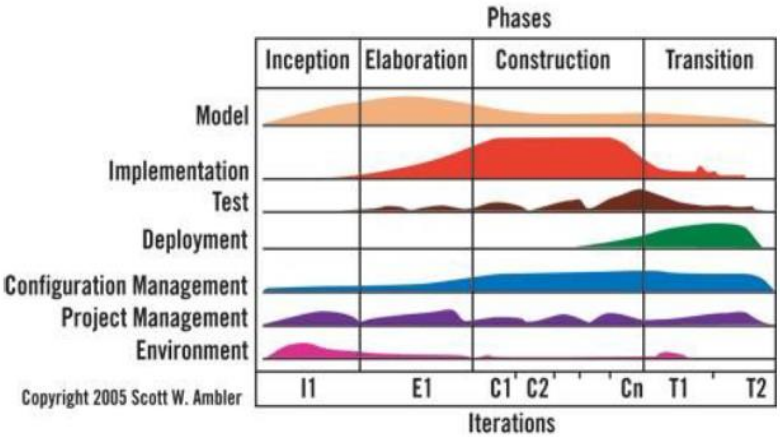
➔ Fases:

◆ Pré-projeto : Orçamento, contrato e projeto candidatos
◆ Ciclo de vida: Desenvolvimento do produto
● Análise de viabilidade
● Iteração de modelo funcional
● Iteração de design e construção
● Implantação
◆ Pós-projeto: Manutenção, melhorias e ajustes

➔ Papéis:

- ◆ Gerente executivo
- ◆ Visionário
- ◆ Intermediador
- ◆ Anunciante
- ◆ Gerente de projeto
- ◆ Coordenador técnico
- ◆ Líder de time
- ◆ Desenvolvedor
- ◆ Testador
- ◆ Escrivão
- ◆ Facilitador

Processo Unificado Ágil

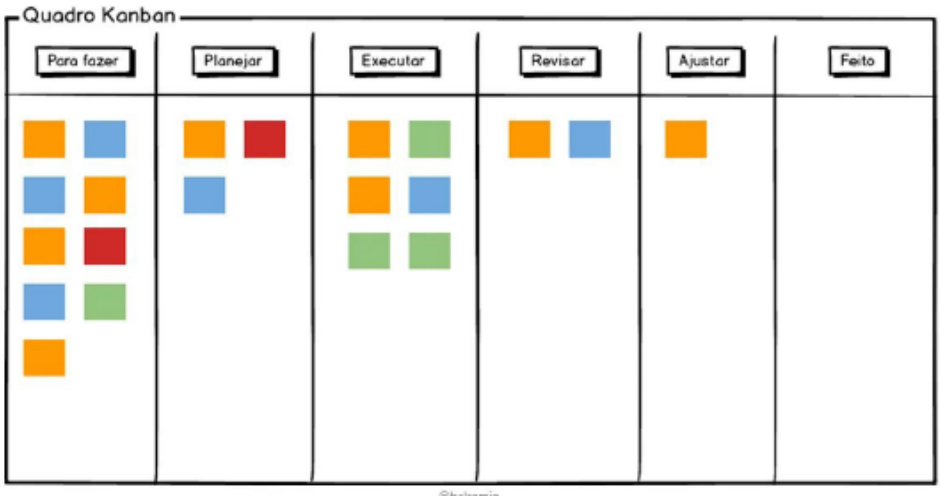


★ Filosofia: Sequencial para o que é amplo e iterativa para o que é particular

★ Atividades:

- o Modelagem
- o Implementação
- o Testes
- o Entrega
- o Configuração e gerenciamento
- o Gerenciamento de ambiente

Kanban



★ Significa cartão/sinalização -> Japão

★ Metodologia para organização de tarefas

- o To do
- o Doing
- o Done

★ É simples e deve ser aliado com outros frameworks para gerenciamento do Projeto

Para saber mais

Microsoft Word - AMPanfleto.doc (agilemodeling.com)
Agile Business Consortium < https://www.agilebusiness.org/business-agility/what-is-dsdm.html>
O que é Kanban? Definição e Detalhes Explicados | Kanbanize < https://kanbanize.com/pt/recursos-kanban/primeiros-passos/o-que-e-kanban>
Scrum Guide < https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf>
Extreme Programming Explained: Embrace Change

Aula 3 - Testes no mundo ágil

Objetivos:

- Contextualizando a atividade de teste
- Entender os testes nas abordagens ágeis e suas diferenças dos modelos tradicionais
- Métodos e práticas de testes no Ágil

Aula 3 . Etapa 1 - Contextualizando a atividade de teste

A adoção de teste nos ciclos de vida do software:

★ Para cada atividade de desenvolvimento existe uma atividade de teste
★ Cada nível de teste tem objetivos específicos
★ A análise e modelagem de testes começam durante a atividade de desenvolvimento
★ Participação no processo de requisitos, modelagem, refinamento...

A participação do QA na história do usuário:

★ Histórias de usuário = requisitos funcionais + não-funcionais
★ Conceito 3C:
o Cartão
o Conversação
o Confirmação -> Critérios de aceite
★ Perspectiva de quem testa difere do cliente, do PO e do desenvolvedor

Atividades envolvidas no planejamento:

★ Análise detalhada has histórias
★ Determinar testabilidade da história
★ Criar testes de aceite
★ Criar tarefas para teste
★ Estimar esforço
★ Identificar aspectos funcionais e não funcionais a serem avaliados
★ Participar do processo de automação

Detalhando a abordagem de teste:

★ Determinar escopo, extensão, objetivos e razões para testes
★ Membros que irão atuar
★ Ambiente e dados necessários
★ Tempo, dependência e pré-requisitos
★ Riscos envolvidos

Aula 3 . Etapa 2 - Testes nas abordagens ágeis

Diferenças nas abordagens de testes:

★ As atividades de testes estão sempre relacionadas com o desenvolvimento, portanto, é importante conhecer os diversos processos e ciclos de vida e como a atividade de testes e qualidade se insere!
★ Cada empresa adota um processo e o customiza de acordo com necessidade
★ Adaptação é palavra-chave
★ As atividades de qualidade e teste estão embutidas em cada iteração podendo ocorrer paralelismo e sobreposição com outras atividades
★ 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
★ Evidências de testes manuais
★ Relatórios de defeitos

Níveis de teste no modelo ágil
★ São sobrepostos
★ Foco
o Testes de unidade
o Testes de aceite
■ Verificação
■ Validação
★ Uso de integração e entrega contínua + automação de testes

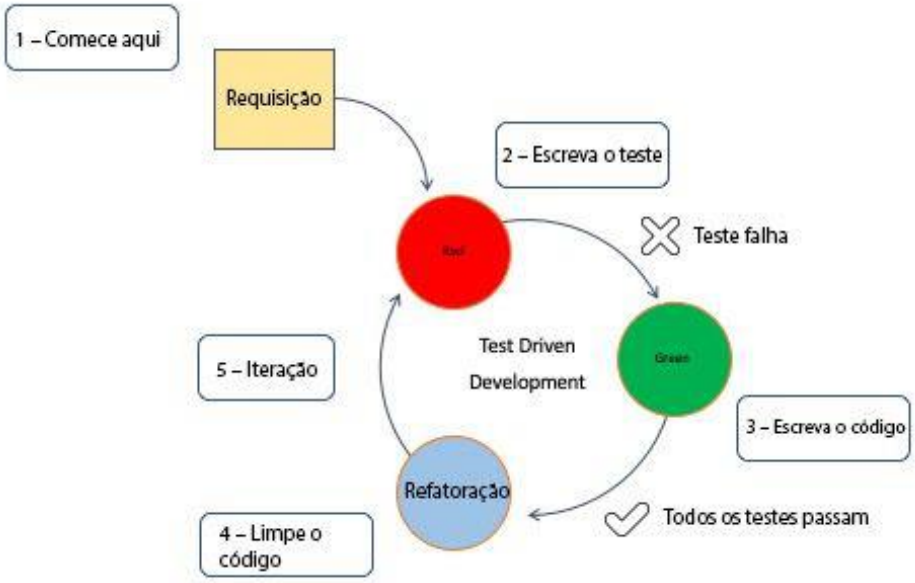
Status de testes no modelo ágil:
★ Adaptação do modelo existe evolução e análise crítica para definir o que está efetivamente concluído
★ Atualização frequente de testes manuais e automatizados
★ Monitorar status de todas as atividades da equipe -> foco no feedback
★ Reuniões diárias para comunicação

Atitudes e habilidades para agilidade:
1. Positividade e pensamento na solução com todos da equipe
2. Pensamento crítico com foco em qualidade
3. Comunicação constante com cliente
4. Feedbacks constantes
5. Avaliação de cenários que representem os critérios de aceites
6. Colaboração em tempo integral com programadores
7. Adaptação à mudança
8. Organização e planejamento

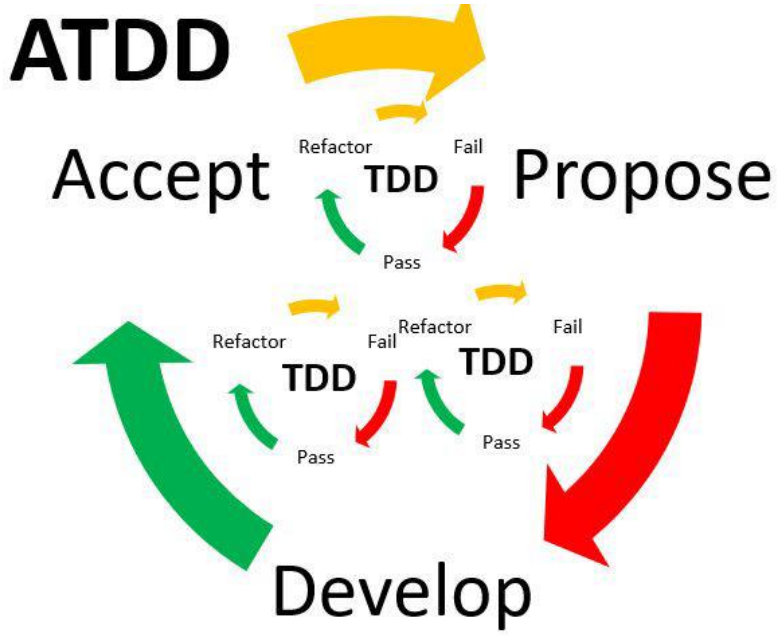
Outras atividades na equipe ágil:
➔ Compreender e atualizar estratégias de teste
➔ Medir e informar cobertura de teste
➔ Garantir uso de ferramentas de forma adequada
➔ Gerenciar ambientes de teste e seus dados
➔ Relatar defeitos e gerenciá-los
➔ Assegurar tarefas de forma adequada e suas estimativas
➔ Esclarecimento contínuo de requisitos junto à equipe
➔ Sugerindo melhorias

Aula 3 . **Etapa 3 -** Métodos de testes no modelo ágil

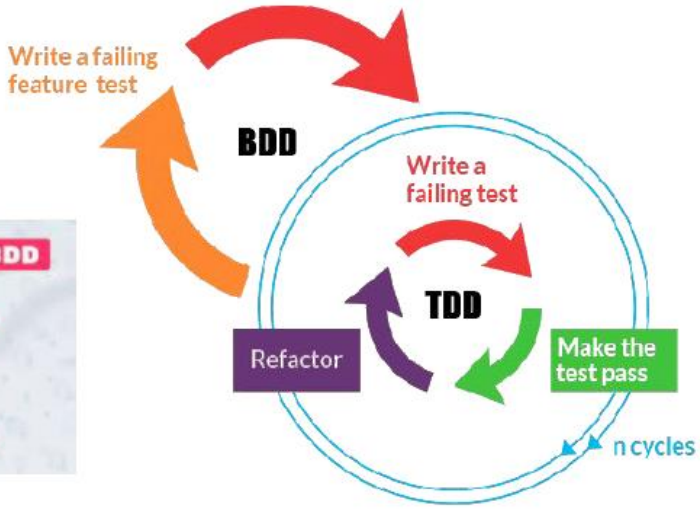
TDD - Desenvolvimento orientado por teste



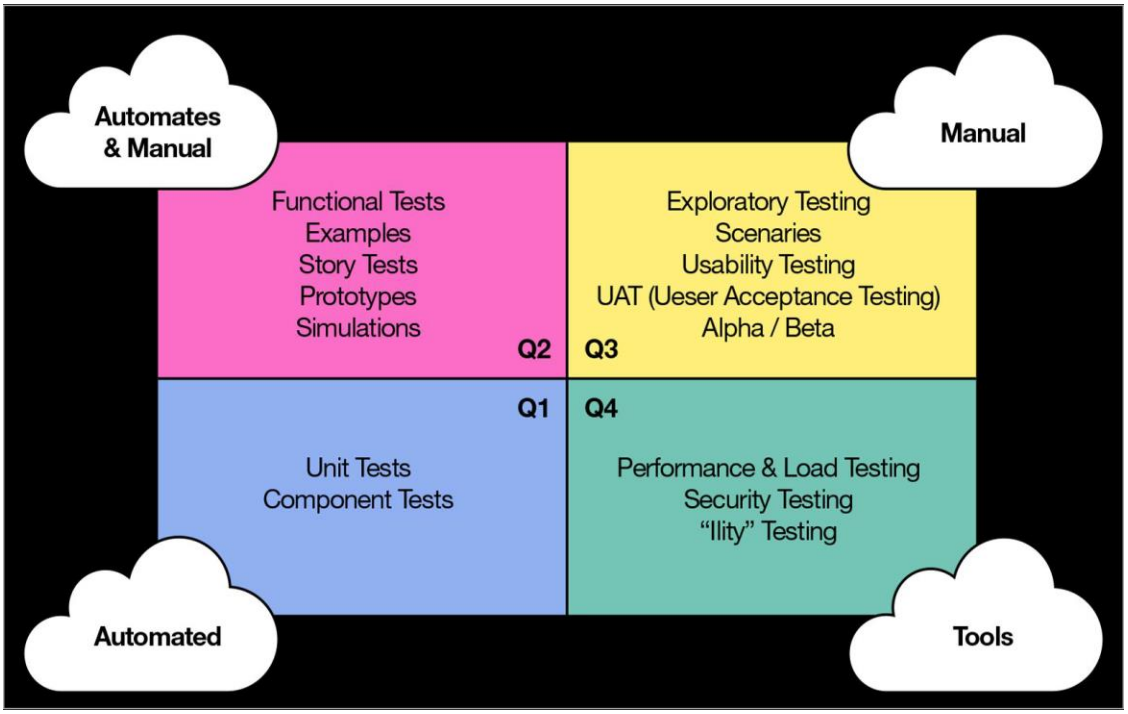
ATDD - Desenvolvimento orientado por teste de aceite

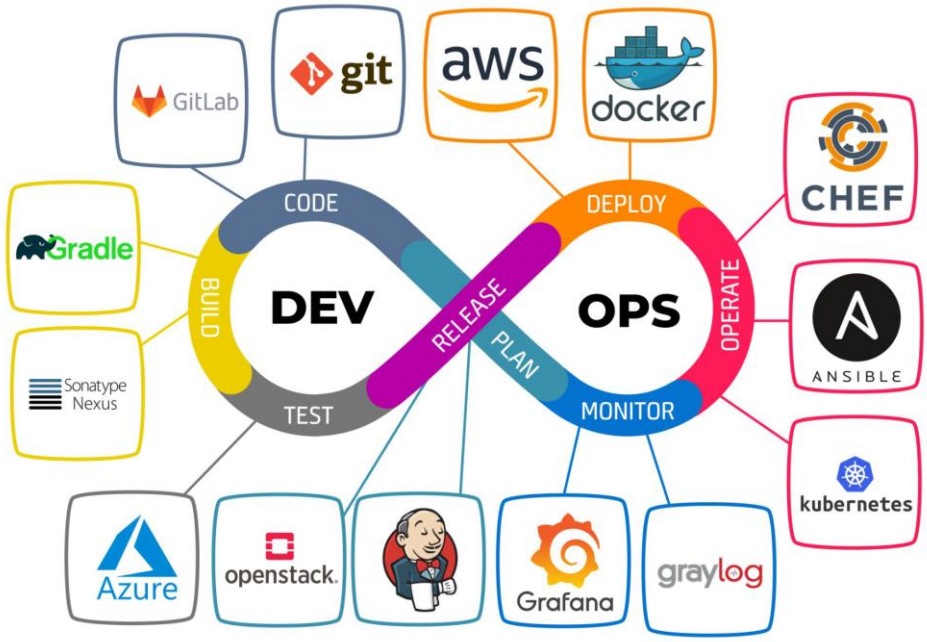


BDD- Desenvolvimento orientado a comportamento



Quadrantes de testes ágeis





Práticas úteis para testes:

★ Teste assistido
★ Testes incrementais
★ Mapa mental
o Estratégias
o Cenários
o Dados

Certificações para testes ágeis:

➔ CTFL-AT Agile Tester
➔ CTFL-ATT Agile Technical Tester

Para saber mais:

★ TDD
★ ATDD

★ BDD na prática
★ CTFL-AT (bstqb.org.br)
★ CTFL-ATT (bstqb.org.br)
★ Cultura DevOps: entenda o que é quais os seus benefícios (profissionaisti.com.br)