Métodos Ágeis

Engenharia de Software II

Fabio Amorim RA: 151041131
Paulo Vitor de Queiroz Zanele RA: 151044244

Conteúdo desta apresentação

- Introdução histórica
- Ideais Ágeis
- Desenvolvimento Ágil e Dirigido a Planos
- Dificuldade e Limitações
- Modelos Ágeis de Processo
- Discussão Artigo 1
- Discussão Artigo 2
- Conclusão
- Referências

Introdução Histórica

- 1960s/1970s: Crise do Software
  - Problemas com orçamento;
  - Problemas com o prazo;
  - Problemas com qualidade;
  - Problemas com requisitos;
  - Problemas com manutenibilidade.

- 1980s/1990s: Engenharia de Software Tradicional
  - Planejamento cuidadoso;
  - Qualidade formalizada;
  - Uso de métodos de analise e design;
  - Ferramentas CASE (Computer-aided software engineering);
  - Desenvolvimento de software rigoroso e controlado.

- 1980s/1990s: Engenharia de Software Tradicional
  - Desenvolvimento de sistemas robustos e críticos (ex: sistemas aeroespaciais e de governo);
  - Sistemas duradouros, precisam ter muito boa manutenibilidade;
  - Grandes/diferentes empresas;
  - Grandes equipes, podendo estar dispersas geograficamente;
  - Longos períodos de desenvolvimentos.

- 2000s/2010s: Engenharia de Software Ágil
  - Mudanças rápidas;
  - Novas oportunidades, novos mercados, novos produtos, novos concorrentes;
  - Importância do software nesse cenário: precisam ser desenvolvidos rapidamente, com agilidade;
  - Tempo passa a ser o mais importante (em alguns casos mais importante que a qualidade).

- 2000s/2010s: Engenharia de Software Ágil
  - Ambientes corporativos menores (médio e pequeno porte);
  - Processos “pesados” causam muito overhead;
  - Muito tempo gasto com planejamento e análise, comparando com implementação e testes;
  - No final da década de 1990, desenvolvedores propuseram processos mais “leves”: os “métodos ágeis”;
  - Objetivo produzir rapidamente softwares úteis, por meio de incrementos, com cada incremento incluindo novas funcionalidades.

Ideais Ágeis

- Ao final da década de 90, reuniões informais entre experientes da área de desenvolvimento de software começaram a acontecer;

- Em 2001, é lançado do Manifesto Ágil;

- Quatro valores e doze princípios.

“Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

- Indivíduos e interações mais que processos e ferramentas
- Software em funcionamento mais que documentação abrangente
- Colaboração com o cliente mais que negociação de contratos
- Responder a mudanças mais que seguir um plano

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.”

Os Doze Princípios

Estes quatro valores se encontram bem definidos dentro dos doze princípios básicos do Manifesto Ágil:

1- Garantir a satisfação do cliente com a entrega rápida e contínua de um software;
2- Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento do software;
3- Entregar um software funcional em meses ou semanas, preferencialmente em períodos mais curtos;
4- Stakeholders e desenvolvedores devem trabalhar juntos diariamente durante todo o projeto;
5- Construir projetos em torno de indivíduos motivados, dando a eles ambientes e suporte necessários, e confiando em suas habilidades;
6- O melhor método para transmitir informações entre a equipe é através de conversa olho-no-olho;

7- Software funcionando é a medida inicial de progresso;
8- Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;
9- Excelência técnica e bom design aumenta a agilidade do processo;
10- Prezar pela simplicidade;
11- Time autônomos e auto-organizáveis geram melhores soluções;
12- A equipe deve tomar intervalos para avaliar um meio de se tornar mais eficaz e se ajustar de acordo.

Exemplos de Métodos Ágeis

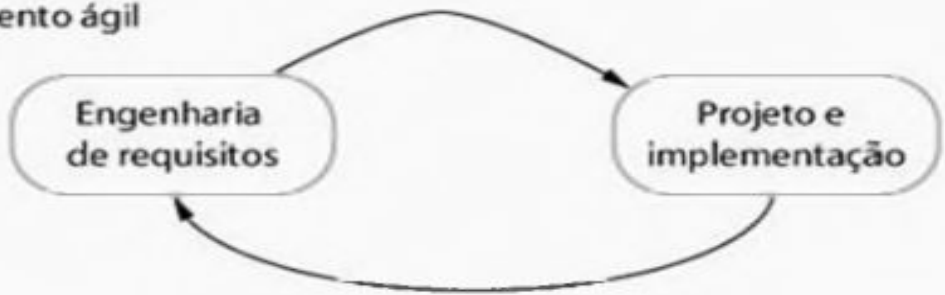
- Extreme Programming (XP)
- Scrum
- Test Driven Development (TDD)
- Crystal
- Desenvolvimento de Software Adaptativo (ASD)
- Desenvolvimento de Sistemas Dinâmicos (FDD)
- Desenvolvimento de Software Enxuto (DSDM)
- Processo Unificado Ágil (AUP)

Princípios	Descrição
Envolvimento do cliente	Os clientes devem estar intimamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar suas iterações.
Entrega incremental	O software é desenvolvido em incrementos com o cliente, especificando os requisitos para serem incluídos em cada um.
Pessoas, não processos	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Membros da equipe devem desenvolver suas próprias maneiras de trabalhar, sem processos prescritivos.
Aceitar as mudanças	Deve-se ter em mente que os requisitos do sistema vão mudar. Por isso, projete o sistema de maneira a acomodar essas mudanças.
Manter a simplicidade	Focalize a simplicidade, tanto do software a ser desenvolvido quanto do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.

Desenvolvimento Ágil e Dirigido a Planos

Desenvolvimento baseado em planos

Desenvolvimento ágil



Para optar por um equilíbrio entre as abordagens, você precisa responder a uma série de questões e técnicas, humanas e organizacionais:

1- É realista uma estratégia de entrega incremental com um rápido feedback?

2- Quão grande é o sistema a ser desenvolvido?

3- Qual é o tempo de vida esperado do sistema?

4- Quais tecnologias estão disponíveis para apoiar o desenvolvimento?

5- Como é organizada a equipe de desenvolvimento?

6- Que tipo de sistema está sendo desenvolvido?

7- Quão bons são os projetistas e programadores na equipe de desenvolvimento?

8- O sistema é sujeito a regulamentação externa?

Dificuldades e Limitações

- Cliente deve estar disposto e capaz de passar o tempo com a equipe de desenvolvimento.
- Cliente deve ser capaz de representar todas as partes interessadas.
- Membros da equipe podem não ter a personalidade adequada.
- A organização pode não ter cultura adequada.

- Priorizar mudanças pode ser extremamente difícil principalmente se há muitas partes envolvidas.
- Manter simplicidade pode ser complicado. Pode-se levar tempo para se fazer as simplificações desejáveis.
- Pode dificultar negociações contratuais (incluindo terceirizações).
- Depende de maturidade de desenvolvedores.

Modelos de Processos Ágeis

Extreme Programming (XP)

Histórico

- Criado em 1996 por Kent Bleck.
- Baseia-se em cinco valores:
  - Comunicação;
  - Simplicidade;
  - Feedback;
  - Coragem;
  - Respeito;

Características

- Os requisitos do sistema são definidos pelos clientes através dos stories;
- O desenvolvimento é realizado em pares pelos desenvolvedores;
- No XP o cliente tem um grande envolvimento no desenvolvimento do projeto, sendo responsável por avaliar se os testes forneceram resultados satisfatórios ou não.
- Grande volume de trabalho em pouco tempo.

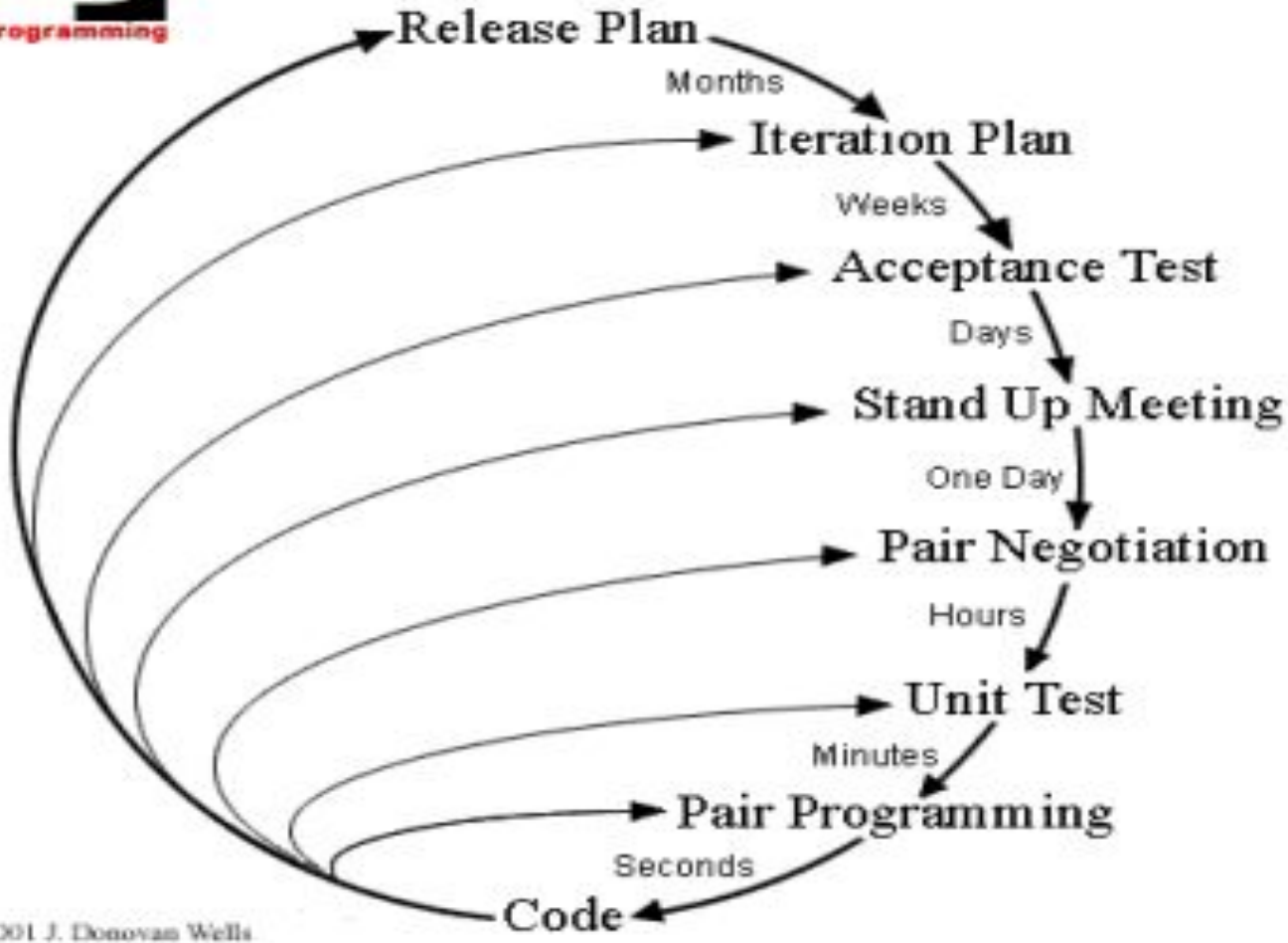
- Comunicação constante, envolvendo o time de desenvolvimento e os stakeholders;
- Testes são realizados o tempo todo. São feitos de forma automatizada, com integração constante;
- Sempre buscando o máximo de simplicidade possível.

Stories

- Fornecem uma visão dos requisitos desejados pelo usuário.
- Podem ser bem detalhadas ou mais simples.
- São uma das principais ferramentas de interação com o usuário.
- Exemplo: “Eu enquanto comprador de livros, quero encontrar um livro de sei o título para poder comprá-lo.”

Extreme Programming

Planning/Feedback Loops



XP Industrial (IXP)

- Acrescenta um novo modelo de organização, para se adaptar a uma escala industrial de produção.

- Em reação ao XP normal seis novas práticas foram incorporadas ao modelo industrial.

- As empresas Ford e Chrysler utilizam essa modelo.

Práticas IXP

- Avaliação de prontidão;
- Análise contínua de risco;
- Comunidade do projeto;
- Projeto de Afretamento;
- Gerenciamento orientado a testes;
- Retrospectivas;
- Aprendizado contínuo.

Resumo XP

- Já que testar é bom, que todos testem o tempo todo.
- Já que revisão é bom, que se revise o tempo todo.
- Se projetar é bom, então refatorar o tempo todo.
- Se teste de integração é bom, então que se integre o tempo todo.
- Se simplicidade é bom, desenvolva uma solução não apenas que funcione, mas que seja a mais simples possível.
- Iterações curtas são boas, então mantenha-as realmente curtas.

Modelos de Processos Ágeis

Scrum

Scrum

- Criado em 1993, por Jeff Sutherland;
- Três ideias principais:
  - Transparência: Fluxo de informações fácil, criando uma cultura de trabalho aberta.
  - Inspeção: Verificar o progresso e os artefatos Scrum.
  - Adaptação: Junção do conhecimento das ideias anteriores, para melhorar o processo de desenvolvimento.

Conceitos

- **Backlog**: É a lista de requisitos da aplicação que será necessária para o seu entendimento e desenvolvimento. O responsável por ela é o *Product Owner*.

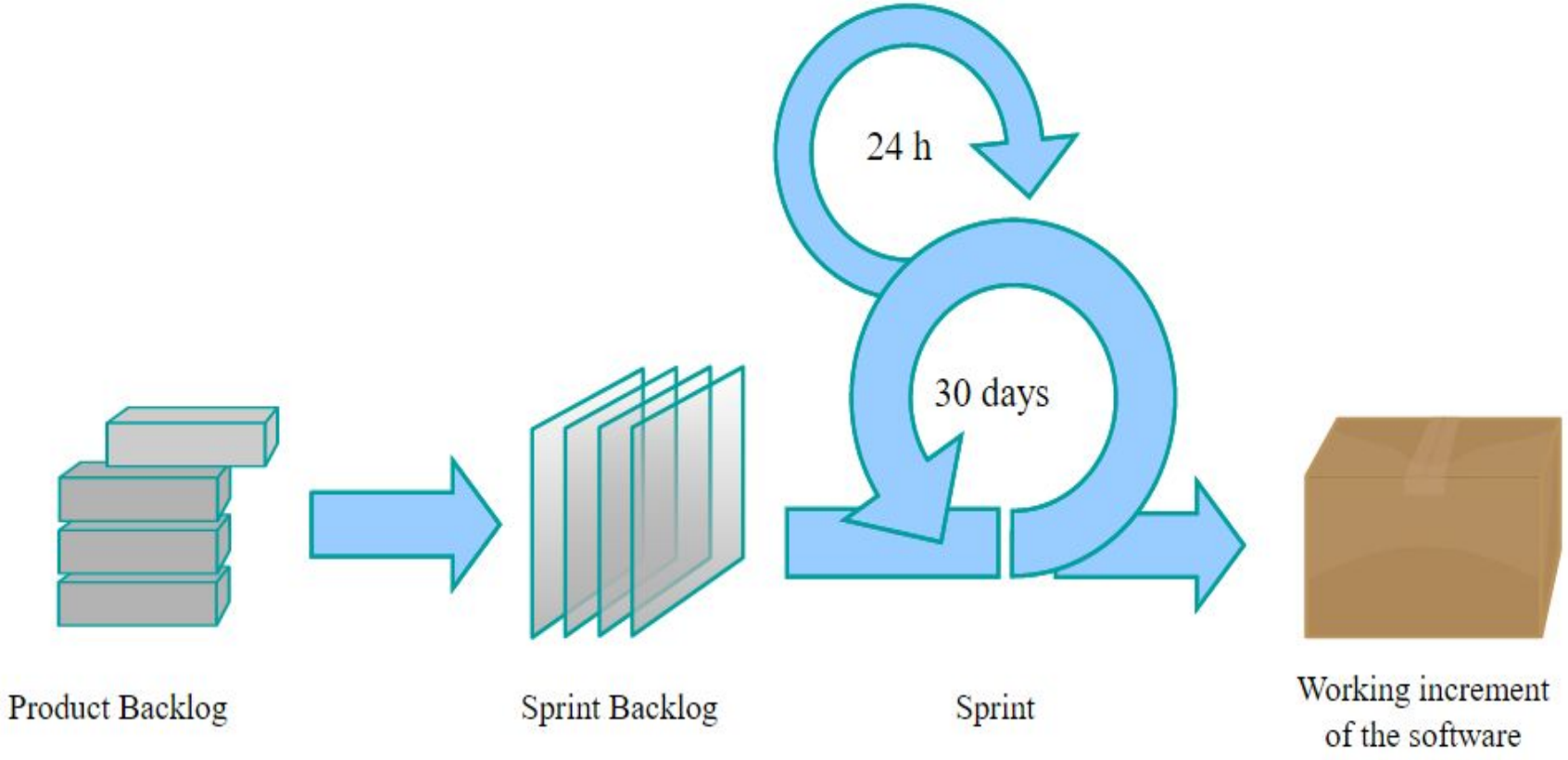
- **Product Backlog**: É a lista de requisitos da aplicação, nela estão contidas todas as informações pertinentes ao projeto e tudo que precisa ser desenvolvido para que a aplicação seja entregue.

- **Sprint Backlog**: É o conjunto de informações necessárias para a finalização de um *Sprint*, ele é criado a partir do *Product Backlog*. Dentro da *Sprint Backlog* é que são criadas as tarefas que precisarão ser entregues no final do *Sprint*.

- **Sprints:** unidades de trabalho necessárias para cumprir algum requisito na lista de pendências. O tempo necessário para se completar um sprint varia de acordo com a complexidade e tamanho do projeto.

- **Reuniões Scrum:** reuniões curtas, diárias, realizadas pela equipe Scrum e lideradas pelo *Scrum master* para responder três perguntas básicas:
  - O que você fez desde a última reunião?
  - Está encontrando alguma dificuldade?
  - O que você pretende fazer até a próxima reunião?

- **Demos:** incremento de software entregue ao cliente, com algumas funções prontas dentro de um intervalo pré-estabelecido.


24 h
30 days
Product Backlog
Sprint Backlog
Sprint
Working increment
of the software

Modelos de Processos Ágeis

Desenvolvimento de Software Adaptativo (ASD)

Desenvolvimento de Software Adaptativo (ASD)

- Proposto em 2000, por Jim Highsmith;
- Construção de software e sistemas complexos;
- Baseado no modelo RAD(Rapid Application Development):
  - Ciclo de desenvolvimento entre 60 e 90 dias;
  - Reutilização de componentes;
  - Criação de componentes reutilizáveis.

Ciclo de Vida

- **Especulação:**
  - Declaração de missão, restrições do projeto, requisitos básicos, plano de entrega.

- **Colaboração:**
  - Alto fluxo de informação, confiança entre a equipe.

- **Aprendizado:**
  - Foco no cliente, revisões de qualidade pela gerência.

Características

- Enfoque na missão, com o objetivo de entregar resultados ao cliente;
- Períodos fechados (time-boxes), com a construção focada na evolução do produto;
- Dirigido a riscos;
- Tolerante a mudanças.

Modelos de Processos Ágeis

Desenvolvimento de Software Lean (LSD)

Lean

- Originalmente desenvolvida na Toyota para guiar os processos industriais de linha de montagem, atuando fortemente na eliminação de desperdícios, aumento da velocidade de processos e excelência em qualidade.

- *Lean Development* traz os conceitos de *Lean* para o universo do desenvolvimento de software.

Princípios

- **Inclua a qualidade no processo:** Verificar a qualidade durante o processo, encontrar rápido o problema, corrigir defeitos imediatamente;

- **Crie conhecimento:** Desenvolvimento de software é melhor concebido se este fizer parte de um processo de aprendizado similar ao de criar uma nova receita;

- **Adiar decisões e comprometimentos:** Diminuir as incertezas retardando as decisões até que possam serem feitas com base em acontecimentos mais firmes, previsíveis e conhecidos;

Desenvolvimento de Software Lean

- O objetivo de um sistema de produção *Lean* é “ter as coisas certas, no lugar certo, na hora certa, desde a primeira vez, enquanto elimina-se o desperdício estando aberto a mudanças”;

- Baseado em um conjunto de princípios para adaptar ferramentas, técnicas e métodos a seus contextos e capacidades específicas, esses princípios são:

    - **Elimine desperdícios:** tudo aquilo que não agrega valor para cliente final e que não são percebidos pelo cliente;

- **Entregar rápido:** Velocidade na entrega garante que o cliente receberá o que ele precisa hoje;

- **Respeitar as pessoas:** Envolver os desenvolvedores nos detalhes das decisões técnicas é fundamental para o atingimento da excelência;

- **Otimizar o todo:** Otimizar desde o começo até o final.

Modelos de Processos Ágeis

Desenvolvimento Guiado por Comportamento (BDD)

Desenvolvimento Guiado por Comportamento (BDD)

- Técnica em que os membros da equipe discutem o comportamento esperado de um sistema para gerar um compartilhado do funcionamento esperado.

- Aprofunda técnicas utilizadas no desenvolvimento orientado a testes (TDD) e do desenvolvimento orientado a testes de aceitação (ATDD).

- É centrado em fornecer funcionalidades específicas de software.

Desenvolvimento Orientado a Testes (TDD)

- Um estilo de programação que intercala três atividades:
  - Codificação;
  - Testes;
  - Design na forma de refatoração do código.

Desenvolvimento Orientado a Testes de Aceitação (ATDD)

- Como o TDD essa técnica envolve os membros do time de diferentes perspectivas (clientes, desenvolvedores e testadores).
- Os testes são realizados antes da implementação da funcionalidade.
- A principal diferença está nas discussões do projeto que são chamadas de ”Três amigos”. Envolvem os clientes, nessas discussões são procuradas responder três perguntas:
  - Qual problema estamos tentando resolver?
  - Como resolver?
  - Resultado.

Características

- O BDD utiliza três pontos principais:
  - Conversas internamente entre a equipe de desenvolvimento, com os clientes e partes interessadas;
  - Exemplos concretos para facilitar o entendimento das partes interessadas;
  - Automação de testes.

- A documentação funcional do BDD é fornecida pelos stories de usuários.

Modelos de Processos Ágeis

Crystal

Crystal

- Criado em 1991, por Alistair Cockburn.
- Foi idealizado após diversas consultas a equipes de desenvolvimento.
- Tem como fundamentos dois princípios:
  - Times podem agilizar os processos como o seu trabalho e tornar-se mais otimizados;
  - Projetos são únicos e dinâmicos e requerem métodos específicos.

Características

- O foco principal do Crystal é a interação entre pessoas, sendo precedente as ferramentas e os processos.

- O desenvolvimento do produto deve ser visto como um jogo.

- O método Crystal tem diferentes tipos de aproximação para diversos tipos de projetos.

- O fator humano:
  - As pessoas são parte essencial do projeto e os processos devem ser adaptados para suas necessidades.

- Adaptabilidade:
  - Os processos e ferramentas não são fixos.

- O mais leve possível:
  - O mínimo de documentação, gerenciamento e relatórios.

Escolha Da Família Crystal

- A escolha de uma determinada família para o projeto depende de três fatores:
  - Tamanho do time.
  - Nível Critíco.
  - Qual a prioridade do projeto?

- Esses fatores são geralmente caracterizados por cores, essas cores variam de acordo com o tamanho da equipe:
  - Branco: oitos ou menos pessoas.
  - Amarelo: Entre dez ou vinte pessoas.
  - Laranja: Entre vinte e cinquenta pessoas.
  - Vermelho: Entre cinquenta e cem pessoas.

Propriedades Obrigatórias do Crystal Clear

- Entrega frequente:
  - Evitar trabalho em um produto que não vai ser lucrativo.

- Aprimoramento reflexivo:
  - Não importa o nível do seu código sempre pode ser melhorado.

- Comunicação osmótica:
  - A informação flui de maneira natural no ambiente de trabalho.

Propriedades Não Obrigatórias do Crystal Clear

- Segurança pessoal:
  - Manter sempre uma comunicação honesta e clara.
- Foco:
  - Concentração em uma única atividade por vez, evitam troca de atividades constantes.
- Acesso fácil a usuários experts:
  - Manter sempre uma comunicação constante com os usuários.
- Testes automáticos.

Processo Crystal

- Utiliza um ciclo de processo aninhado de vários tamanhos;
- O que cada membro da equipe faz em qualquer momento depende da sua localização em cada ciclo;
- Cada projeto divide-se normalmente em sete ciclos.

Ciclos

- Projeto;
- Entrega;
- Iteração;
- Semana de trabalho;
- Integração;
- Dia de trabalho;
- Episódio.

Resumo

- Comunicação é o ponto central.
- Flexibilidade.
- Modelos únicos para cada projeto.

Discussão Artigo I

Evolução dos Bancos de Dados no Ambiente de Desenvolvimento de Processos Ágeis

Introdução

- Avanço nos métodos desenvolvimento de software;
- Problemas em evoluir esquemas de bancos de dados;
- Proposta evolutiva baseada em refatoração;
- Estratégias que auxiliam no controle e compartilhamento das alterações.

Refatoração

- Reestrutura o corpo do código já existente;
- Altera internamente, sem alterar seu comportamento externo;
- Pequenas mudanças individualmente;
- Em conjunto, grandes resultados;

- Resultados como:
  - Melhor otimização;
  - Mais compreensivel;
  - Maior flexibilidade.

- “ A refatoração não resolve todos os problemas, mas sem dúvida é uma ferramenta valiosa.”

Refatoração em Bancos de Dados

- Resistência dos desenvolvedores;
- Defendem que esquema seja arquitetado de forma tradicional;
- Compreensão completa do domínio;
- Necessidades do cliente mudam ao decorrer do projeto;
- Problemas de acoplamento;

- “A consolidação dos processos de desenvolvimento de software modernos permitiu que o conceito do desenvolvimento iterativo e evolutivo fosse também empregado a projetos de bancos de dados.”

- Refatoração aplicada durante o processo de desenvolvimento;

- Correção de falhas;

- Sintonia com a evolução dos requisitos;

- Baseado em técnicas como:
  - Modelagem ágil;
  - Refatoração em bancos de dados;
  - Testes de regressão;
  - Desenvolvimento orientado a testes.

- Surgimento dos “bancos de dados evolutivos”, em resposta ao desenvolvimento de software ágil.

Refatoração no Ambiente de Desenvolvimento

- É defendido que todo desenvolvedor tenha uma *sandbox;*
- Realizar alterações sem correr risco;
- O sistema funcionará mesmo após as alterações;
- Outra prática é a integração contínua;
- Menor período de tempo entre elas, menor a possibilidade de conflitos;
- Utiliza-se a técnica *check-out*;

- Estratégia de controle de versão;
- Possibilita registrar as alterações;
- Auxilia na identificação e retificação de eventuais falhas;

Tipos de Refatoração em Bancos de Dados

- Foi definido quatro tipos de refatoração: Estrutural, Qualidade dos Dados, Arquitetural e Integridade Referencial;

- **Refatoração Estrutural**: Modificar os esquemas do banco de dados;
  - Remover, inserir, alterar, unificar, dividir e renomear colunas e tabelas;
  - Modificar relacionamentos;
  - Substituir campos complexos por tabelas.

- **Refatoração de Qualidade de Dados**: Visa uma melhoria das informações, a fim de assegurar uma coerência de dados;
  - Criação de tabela de pesquisa;
  - Remover e inserir restrição de coluna;
  - Mover dados;

- **Refatoração de Integridade Referencial**: Garante integridade dos dados, preservando os relacionamentos entre as tabelas;
  - Exclusão em cascata, fisica ou lógica;
  - Adicionar ou remover restrições na chave lógica;

- **Refatoração Arquitetural**: Modificar o acesso às tabelas de forma organizada e padronizada a fim de obter uma otimização, é preciso analisar bem sua empregabilidade;
  - Adicionar métodos;
  - Inserção de índices;
  - Inserção de tabelas somente leitura.

Aplicação

- Sistema de Alocação de Espaços, desenvolvido para a Universidade Federal de São Carlos (UFSCar);
- Requisitos inconsistentes;
- Adoção do Scrum, para o gerenciamento do desenvolvimento do projeto;
- Aplicação dos exemplos de refatoração em bancos de dados.

Conclusão

- O estudo foi desenvolvido como uma solução as frequentes mudanças no processo de desenvolvimento;
- Aplicação no estudo de caso;
- Forma fácil e segura, garantiu o controle das alterações;
- O emprego dessas técnicas de fatoração, permite atender as reais necessidades do cliente, mesmo com as mudanças.

Discussão Artigo II

Evolução do desenvolvimento ágil no Brasil

Introdução

- Falta de estudos sobre o desenvolvimento ágil no Brasil.

- Grande impacto dos métodos ágeis nos currículos dos cursos, organização de empresas de desenvolvimento.

Histórico

- Em 2000, foi realizado a Primeira Conferência Internacional de Extreme Programming e Desenvolvimento Ágil de Software.

- Em 2002 ocorreu o primeiro evento nacional de desenvolvimento ágil, Extreme Programming Brasil. Com a presença de Kent Bleck.

Questões Abordadas

- Como o ensino de métodos ágeis evoluiu no Brasil?
- Como a pesquisa evoluiu no Brasil?
- Como a prática dos métodos ágeis afetou a indústria brasileira?

Ensino de Métodos Ágeis

- Criação de cursos específicos para ensino de métodos ágeis.
- Oferecimento de certificações.

Pesquisa em Métodos Ágeis

- Crescimento constante no número de dissertações e teses apresentadas nos últimos anos;
- O nível de colaboração aumentou, entretanto continua limitado;
- Um grande avanço no número de publicações em revistas internacionais.

Métodos Ágeis na Indústria

- Aumento no uso dos métodos ágeis;
- Aumento de produtividade de desenvolvimento;
- Redução de custos de desenvolvimento.

Conclusão

- Aumento no interesse pelos métodos ágeis;
- Grande aceitação por parte da indústria no uso das soluções ágeis;
- Falta de pessoal especializado.

Conclusão

- Os métodos ágeis buscam diminuir a documentação e focar no desenvolvimento do produto.
- Comunicação direta com o cliente.
- Desenvolvimento dinâmico.

Referências

- AGILE ALLIANCE. Agile Practice Guide. 1th Edition, 2017.

- BECK, K et al. Manifesto for Agile Software Development. Disponível em: < http://agilemanifesto.org/>. Acesso em: 17 out. 2018.

- Claudete Florêncio, Marilde Terezinha Prado Santos. “Evolução dos Bancos de Dados no Ambiente de Desenvolvimento de Projetos Ágeis”. Universidade Federal de São Carlos, Departamento de Computação, 2015.

- Claudia de O. Melo, Viviane Santos, Eduardo Katayama, Hugo Corbucci, Rafael Prikladnicki, Alfredo Goldman, Fabio Kon, "The Evolution of Agile Software Development in Brazil." Journal of Brazilian Computer Society, 2013.

- COCKBURN,Alistair. Crystal clear a human-powered methodology for small: 1. ed. Estados Unidos:Pearson , 2004.

- FILHO, H. R. Scrum Alliance. Disponível em: <https://www.scrumalliance.org/>. Acesso em: 02 out. 2018.

- Lucas Layman, Laurie Williams, Lynn Cunningham, "Exploring Extreme Programming in Context: An Industrial Case Study." North Carolina State University, Department of Computer Science.

- MAXIM, R Bruce; PRESSMAN, Roger. Software Engineering - A Practitioner's Approach: 8. ed. Nova York: Editora McGraw-Hill Education, 2015

- SOMMERVILLE, Ian. Software Engineering: 10. ed. Estados Unidos: Pearson, 2016.

- STEFFEN, JULIANA. Lean Development. Disponível em: https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/lean_para_desenvolvimento_de_sw_o_que__c3_a9_isso_afinal12?lang=en>. Acesso em: 27 nov. 2018.