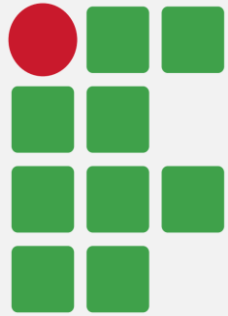
ENGENHARIA DE SOFTWARE

Introdução aos Métodos Ágeis

Prof. Giovani Fonseca Ravagnani Disperati
IFSP – Câmpus Guarulhos

CONTEÚDO PROGRAMÁTICO



- Desenvolvimento Ágil de Software
- Métodos Ágeis
- Técnicas de Desenvolvimento Ágil de Software
- Scrum
- Clean Code
- Code Smells
- TDD na prática

DESENVOLVIMENTO ÁGIL DE SOFTWARE

Desenvolvimento ágil de Software

- Desenvolvimento e entrega rápidos agora são frequentemente o requisito mais importante para sistemas de software

- As empresas operam em um requisito de mudança rápida e é praticamente impossível produzir um conjunto de requisitos de software estáveis;

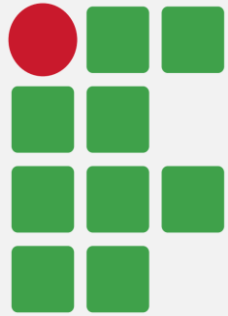
- O software deve evoluir rapidamente para refletir as mudanças nas necessidades de negócios.

Desenvolvimento ágil de Software

- O desenvolvimento orientado a planos é essencial para alguns tipos de sistema, mas não atende a essas necessidades de negócios.

- Os métodos de desenvolvimento ágil surgiram no final da década de 1990, com objetivo de reduzir radicalmente o tempo de entrega de sistemas de software funcionais.

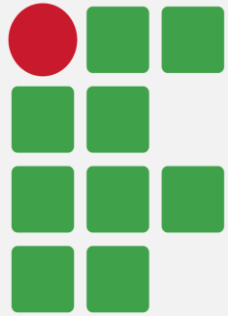
Desenvolvimento ágil de Software



- A especificação, o design e a implementação do programa são intercalados.

- O sistema é desenvolvido como uma série de versões ou incrementos com as partes interessadas envolvidas na especificação e avaliação da versão.

Desenvolvimento ágil de Software



- Entrega frequente de novas versões para avaliação

- Amplo suporte a ferramentas (por exemplo, ferramentas de teste automatizadas) usado para apoiar o desenvolvimento.

- Documentação *necessária* - foco no código de trabalho

Desenvolvimento ágil de Software

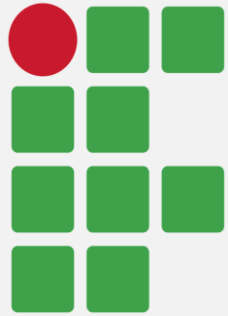
- Desenvolvimento orientado a planos
  - Uma abordagem orientada a planos para a engenharia de software é baseada em estágios de desenvolvimento separados, com as saídas a serem produzidas em cada um desses estágios planejadas com antecedência.
  - Não necessariamente implica o modelo em cascata - o desenvolvimento incremental e orientado por planos é possível
  - A iteração ocorre dentro das atividades.

Desenvolvimento ágil de Software

- Desenvolvimento ágil
  - Especificação, design, implementação e teste são intercalados e as saídas do processo de desenvolvimento são decididas por meio de um processo de negociação durante o processo de desenvolvimento de software.

MÉTODOS ÁGEIS

Métodos ágeis



- A insatisfação com o overhead envolvido nos métodos de design de software das décadas de 1980 e 1990 levou à criação de métodos ágeis.
- Esses métodos
  - Concentram-se no código em vez do design
  - São baseados em uma abordagem iterativa para o desenvolvimento de software
  - Destinam-se a fornecer software funcional rapidamente e evoluí-lo rapidamente para atender aos requisitos de mudança.

Métodos ágeis

- O objetivo dos métodos ágeis é reduzir sobrecargas no processo de software (por exemplo, limitando a documentação) e ser capaz de responder rapidamente às mudanças de requisitos sem retrabalho excessivo.

Manifesto ágil

- 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.

Métodos ágeis

Princípios	Descrição
Envolvimento do cliente	Os clientes devem estar intimamente envolvidos em todo o processo de desenvolvimento. Sua função é fornecer e priorizar novos requisitos de sistema e avaliar as iterações do sistema.
Entrega incremental	O software é desenvolvido em incrementos com o cliente especificando os requisitos a serem incluídos em cada incremento.
Pessoas não processos	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Os membros da equipe devem desenvolver suas próprias maneiras de trabalhar sem processos prescritivos.
Abrace a mudança	Espere que os requisitos do sistema mudem e, assim, projete o sistema para acomodar essas mudanças.
Mantenha a simplicidade	Foco na simplicidade tanto do software que está sendo desenvolvido quanto do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.

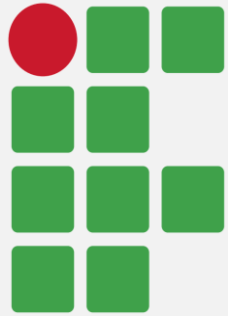
Manifesto ágil

- Praticamente todos os produtos de software e aplicativos agora são desenvolvidos usando uma abordagem ágil.

- Desenvolvimento de sistema personalizado dentro de uma organização, onde há um compromisso claro do cliente de se envolver no processo de desenvolvimento e onde existem poucas regras e regulamentos externos que afetam o software.

TÉCNICAS E PROCESSOS ÁGEIS DE SOFTWARE

Extreme Programming



- Um método ágil muito influente, desenvolvido no final da década de 1990, que introduziu uma variedade de técnicas de desenvolvimento ágil.

- Extreme Programming (XP) tem uma abordagem "extrema" para o desenvolvimento iterativo.
  - Novas versões podem ser construídas várias vezes por dia;
  - Os incrementos são entregues aos clientes a cada 2 semanas;
  - Todos os testes devem ser executados para cada construção e a construção só é aceita se os testes forem executados com sucesso.

Extreme Programming

Práticas do XP	Descrição
Planejamento incremental	Os requisitos são registrados em cartões de história e as histórias a serem incluídas em um lançamento são determinadas pelo tempo disponível e sua prioridade relativa. Os desenvolvedores dividem essas histórias em 'Tarefas' de desenvolvimento.
Pequenos lançamentos	O conjunto mínimo de funcionalidades úteis que fornece valor comercial é desenvolvido primeiro. As versões do sistema são frequentes e adicionam funcionalidades de forma incremental à primeira versão.
Design Simples	O design é realizado o suficiente para atender aos requisitos atuais e nada mais.
Desenvolvimento *test-first*	Um framework de testes é usado para escrever testes para uma nova funcionalidade antes que essa funcionalidade seja implementada.
Refatoração	Espera-se que todos os desenvolvedores refatorem o código continuamente assim que forem encontradas melhorias de código possíveis. Isso mantém o código simples e sustentável.

Extreme Programming

Práticas do XP	Descrição
Programação em pares	Os desenvolvedores trabalham em pares, verificando o trabalho uns dos outros e fornecendo o suporte para sempre fazer um bom trabalho.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de forma que nenhuma ilha de especialização se desenvolva e todos os desenvolvedores assumam a responsabilidade por todo o código. Qualquer um pode mudar qualquer coisa.
Integração contínua	Assim que o trabalho em uma tarefa é concluído, ele é integrado a todo o sistema. Depois de qualquer integração, todos os testes de unidade no sistema devem ser aprovados.
Ritmo sustentável	Grandes quantidades de horas extras não são consideradas aceitáveis, pois o efeito líquido geralmente é a redução da qualidade do código e da produtividade a médio prazo.

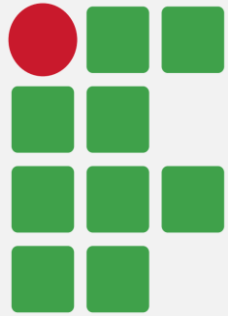
Extreme Programming

Práticas do XP	Descrição
Cliente no local	Um representante do usuário final do sistema (o cliente) deve estar disponível em tempo integral para uso da equipe XP. Em um processo de programação extremo, o cliente é um membro da equipe de desenvolvimento e é responsável por trazer os requisitos do sistema para a equipe para implementação.

Extreme Programming

- O desenvolvimento incremental é suportado por meio de lançamentos de sistema pequenos e frequentes.
- O envolvimento do cliente significa envolvimento do cliente em tempo integral com a equipe.
- Pessoas ao invés de processos, por meio de programação em pares, propriedade coletiva e um processo que evita longas horas de trabalho.
- Mudança suportada por meio de versões regulares do sistema.
- Manter a simplicidade por meio de refatoração constante de código.

Extreme Programming



- A Extreme Programming tem um foco técnico e não é fácil de integrar com a prática de gerenciamento na maioria das organizações.

- Consequentemente, embora o desenvolvimento ágil use práticas do XP, o método originalmente definido não é amplamente utilizado.
  - Principais práticas
  - Histórias de usuários para especificação
  - Refatoração
  - Desenvolvimento test-first (TDD)
  - Programação em pares

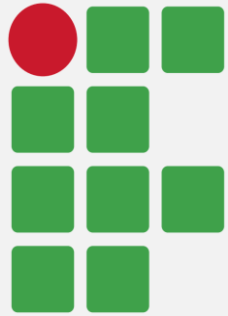
Extreme Programming

- No XP, um cliente ou usuário faz parte da equipe XP e é responsável por tomar decisões sobre os requisitos.
    - Os requisitos do usuário são expressos como histórias de usuário ou cenários.

- Eles são escritos em cartões e a equipe de desenvolvimento os divide em tarefas de implementação. Essas tarefas são a base das estimativas de cronograma e custo.

- O cliente escolhe as histórias a serem incluídas na próxima versão com base em suas prioridades e nas estimativas de cronograma.

Extreme Programming



Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either 'current medication', 'new medication' or 'formulary'.

If you select 'current medication', you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

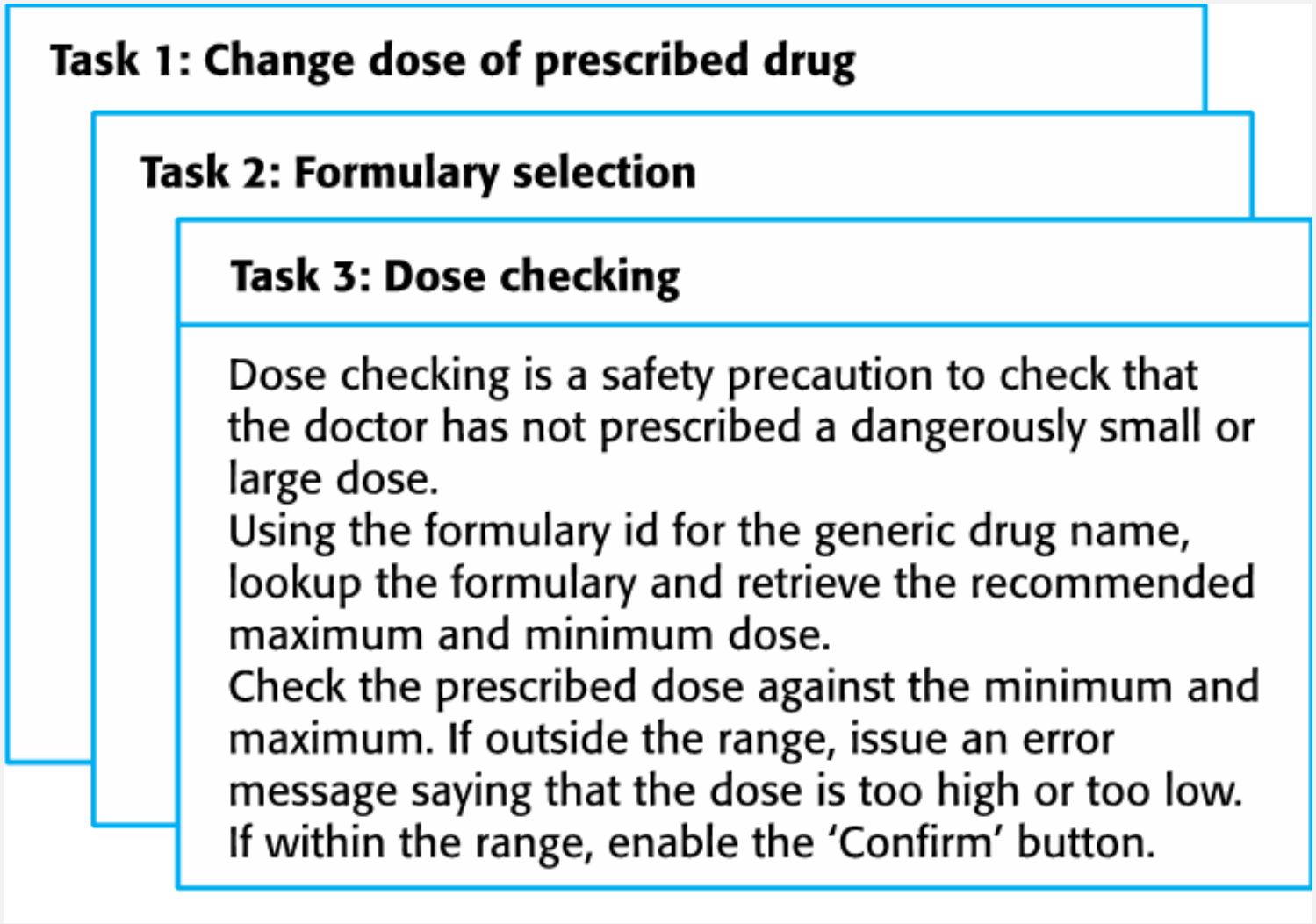
If you choose, 'new medication', the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose 'formulary', you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

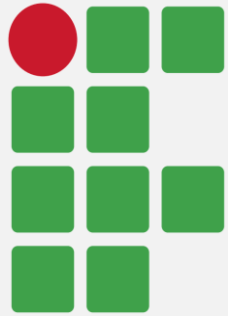
In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click 'OK' or 'Change'. If you click 'OK', your prescription will be recorded on the audit database. If you click 'Change', you reenter the 'Prescribing medication' process.

Extreme Programming



Extreme Programming

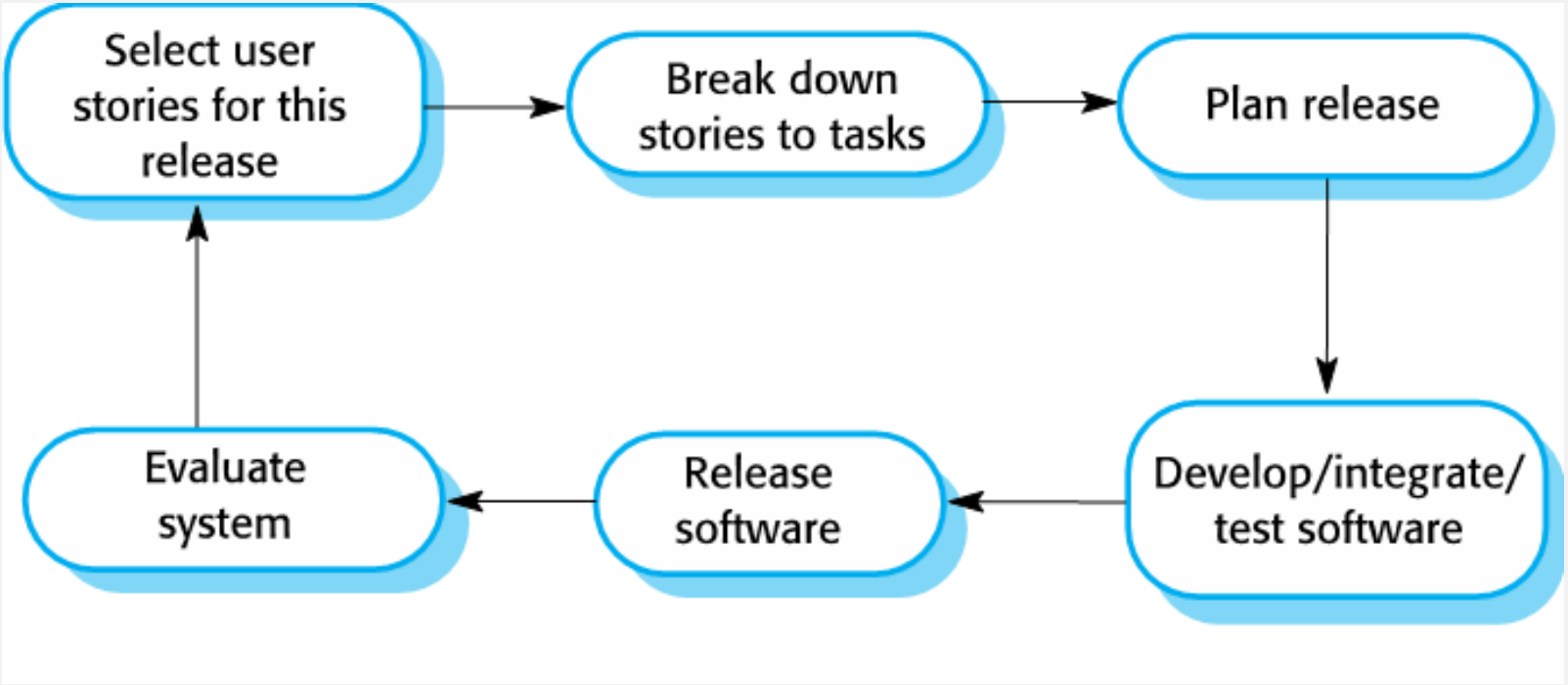


- A sabedoria convencional em engenharia de software é projetar para a mudança. Vale a pena gastar tempo e esforço antecipando as mudanças, pois isso reduz os custos mais tarde no ciclo de vida.

- O XP, no entanto, afirma que isso não vale a pena, pois as alterações não podem ser antecipadas com segurança.

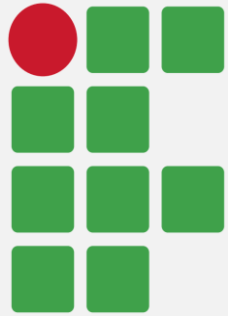
- Em vez disso, ele propõe a melhoria constante do código (refatoração) para facilitar as alterações quando elas precisam ser implementadas.

Extreme Programming



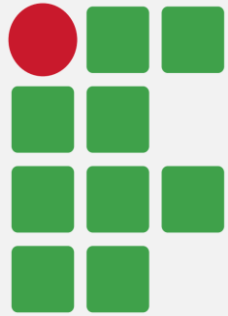
REFATORAÇÃO

Refatoração



- A equipe de programação busca possíveis melhorias no software e faz essas melhorias mesmo quando não há necessidade imediata delas.
    - Isso melhora a compreensão do software e, portanto, reduz a necessidade de documentação.
- As alterações são mais fáceis de fazer porque o código é bem estruturado e claro.
    - No entanto, algumas mudanças requerem refatoração de arquitetura e isso é muito mais caro.

Refatoração

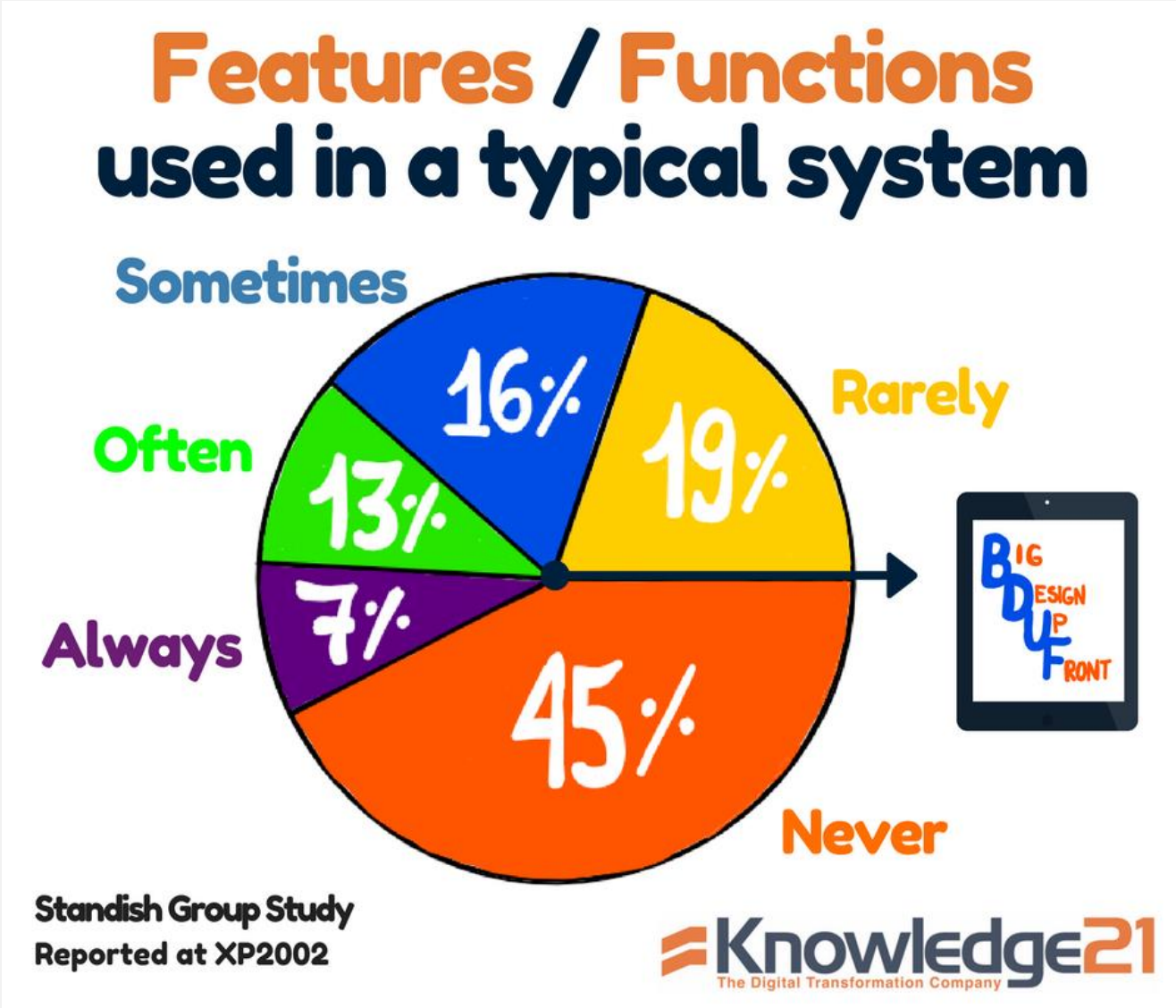


- Reorganização de uma hierarquia de classes para remover código duplicado.

- Organizar e renomear atributos e métodos para torná-los mais fáceis de entender.

- A substituição do código embutido por chamadas para métodos que foram incluídos em uma biblioteca de programa.

Extreme Programming



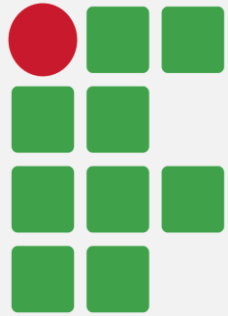
DESENVOLVIMENTO DIRIGIDO A TESTES

Desenvolvimento Dirigido a Testes

- O teste é fundamental para o XP e o XP desenvolveu uma abordagem em que o programa é testado após cada mudança ser feita.

- Recursos de teste do XP:
  - Desenvolvimento de teste inicial.
  - Desenvolvimento de teste incremental a partir de cenários.
  - Envolvimento do usuário no desenvolvimento e validação do teste.
  - Os conjunto de testes automatizados é usado para executar todos os testes de componentes sempre que uma nova versão é criada.

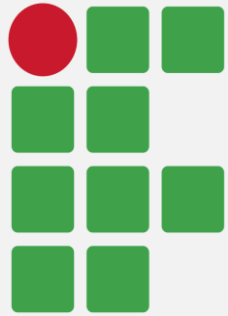
Desenvolvimento Dirigido a Testes



- Escrever testes antes do código esclarece os requisitos a serem implementados.

- Os testes são escritos como programas em vez de dados para que possam ser executados automaticamente. O teste inclui uma verificação de que foi executado corretamente.

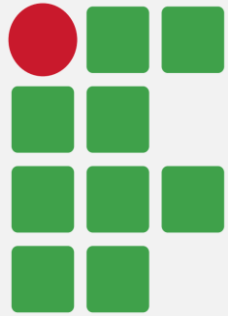
Desenvolvimento Dirigido a Testes



- Normalmente depende de uma estrutura de teste como o Junit.

- Todos os testes anteriores e novos são executados automaticamente quando uma nova funcionalidade é adicionada, verificando assim se a nova funcionalidade não introduziu erros.

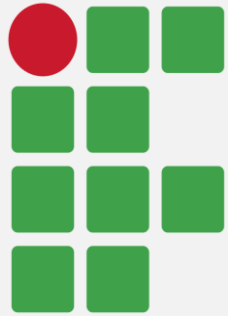
Desenvolvimento Dirigido a Testes



- A função do cliente no processo de teste é ajudar a desenvolver testes de aceitação para as histórias que serão implementadas na próxima versão do sistema.

- O cliente que faz parte da equipe escreve os testes à medida que o desenvolvimento prossegue. Todo código novo é, portanto, validado para garantir que é o que o cliente precisa.

Desenvolvimento Dirigido a Testes



- No entanto, as pessoas que adotam a função de cliente têm tempo disponível limitado e, portanto, não podem trabalhar em tempo integral com a equipe de desenvolvimento.

- Eles podem sentir que fornecer os requisitos foi uma contribuição suficiente e, portanto, podem relutar em se envolver no processo de teste.

Descrição de casos de testes

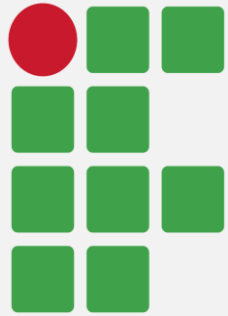
Test 4: Dose checking

Input:
1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:
1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose * frequency is too high and too low.
4. Test for inputs where single dose * frequency is in the permitted range.

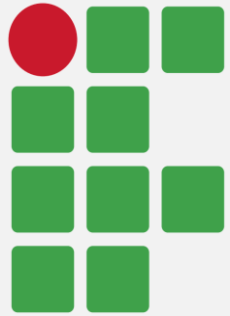
Output:
OK or error message indicating that the dose is outside the safe range.

Desenvolvimento Dirigido a Testes



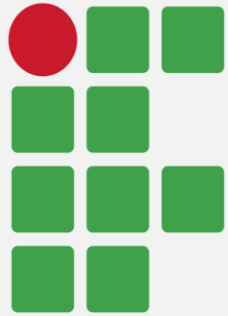
- Automação de teste significa que os testes são escritos como componentes executáveis antes que a tarefa seja implementada
    - Esses componentes de teste devem ser independentes, devem simular o envio da entrada a ser testada e devem verificar se o resultado atende à especificação de saída. Uma estrutura de teste automatizado (por exemplo, Junit) é um sistema que torna mais fácil escrever testes executáveis e enviar um conjunto de testes para execução.

Desenvolvimento Dirigido a Testes



- Como o teste é automatizado, há sempre um conjunto de testes que podem ser executados de forma rápida e fácil
    - Sempre que qualquer funcionalidade é adicionada ao sistema, os testes podem ser executados e os problemas que o novo código introduziu podem ser detectados imediatamente.

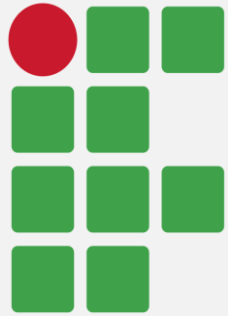
Problemas com o Desenvolvimento Dirigido a Testes



- Os programadores preferem programar a testar e às vezes tomam atalhos ao escrever testes.

- Por exemplo, eles podem escrever testes incompletos que não verificam todas as exceções possíveis que podem ocorrer.

Problemas com o Desenvolvimento Dirigido a Testes

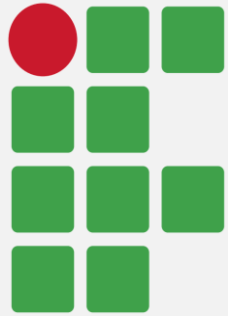


- Alguns testes podem ser muito difíceis de escrever de forma incremental. Por exemplo, em uma interface de usuário complexa, muitas vezes é difícil escrever testes de unidade para o código que implementa a "lógica de exibição" e o fluxo de trabalho entre as telas.

- É difícil julgar a integridade de um conjunto de testes. Embora você possa ter muitos testes de sistema, seu conjunto de teste pode não fornecer uma cobertura completa.

PROGRAMAÇÃO EM PARES

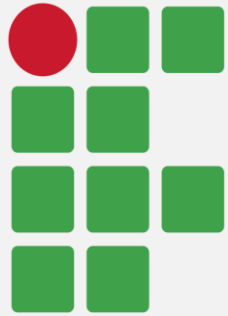
Programação em pares



- A programação em pares envolve programadores trabalhando em pares, desenvolvendo código juntos.

- Isso ajuda a desenvolver a propriedade comum do código e dissemina o conhecimento pela equipe.

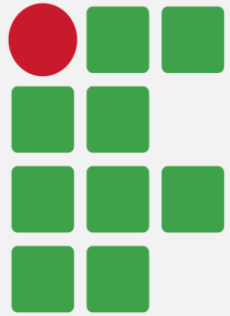
Programação em pares



- Serve como um processo de revisão informal, pois cada linha de código é examinada por mais de 1 pessoa.

- A programação em pares incentiva a refatoração, pois toda a equipe pode se beneficiar com o aprimoramento do código do sistema.

Programação em pares



- Na programação em pares, programadores sentam-se juntos no mesmo computador afim de desenvolver o software.

- Pares são criados dinamicamente, de forma que todos os membros trabalhem com os demais durante o processo de desenvolvimento.

Programação em pares

- O compartilhamento de conhecimento que acontece durante a programação em pares é muito importante pois reduz o risco de um projeto quando um membro deixa a equipe.

- Alguns argumentam que programação em pares é ineficiente; há evidências contrárias, entretanto, mostrando que dois programadores trabalhando conjuntamente são mais produtivos que ambos trabalhando separados.

GESTÃO DE PROJETOS ÁGEIS

Gestão de projetos ágeis

- A principal responsabilidade dos gerentes de projeto de software é gerenciar o projeto de forma que o software seja entregue no prazo e dentro do orçamento planejado para o projeto.

- A abordagem padrão para gerenciamento de projetos é orientada a planos. Os gerentes elaboram um plano para o projeto mostrando o que deve ser entregue, quando deve ser entregue e quem vai trabalhar no desenvolvimento das entregas do projeto.

Gestão de projetos ágeis

- O gerenciamento ágil de projetos requer uma abordagem diferente, que é adaptada ao desenvolvimento incremental e às práticas utilizadas nos métodos ágeis.

SCRUM

Scrum

- Scrum é um método ágil que se concentra no gerenciamento de desenvolvimento iterativo ao invés de práticas ágeis específicas

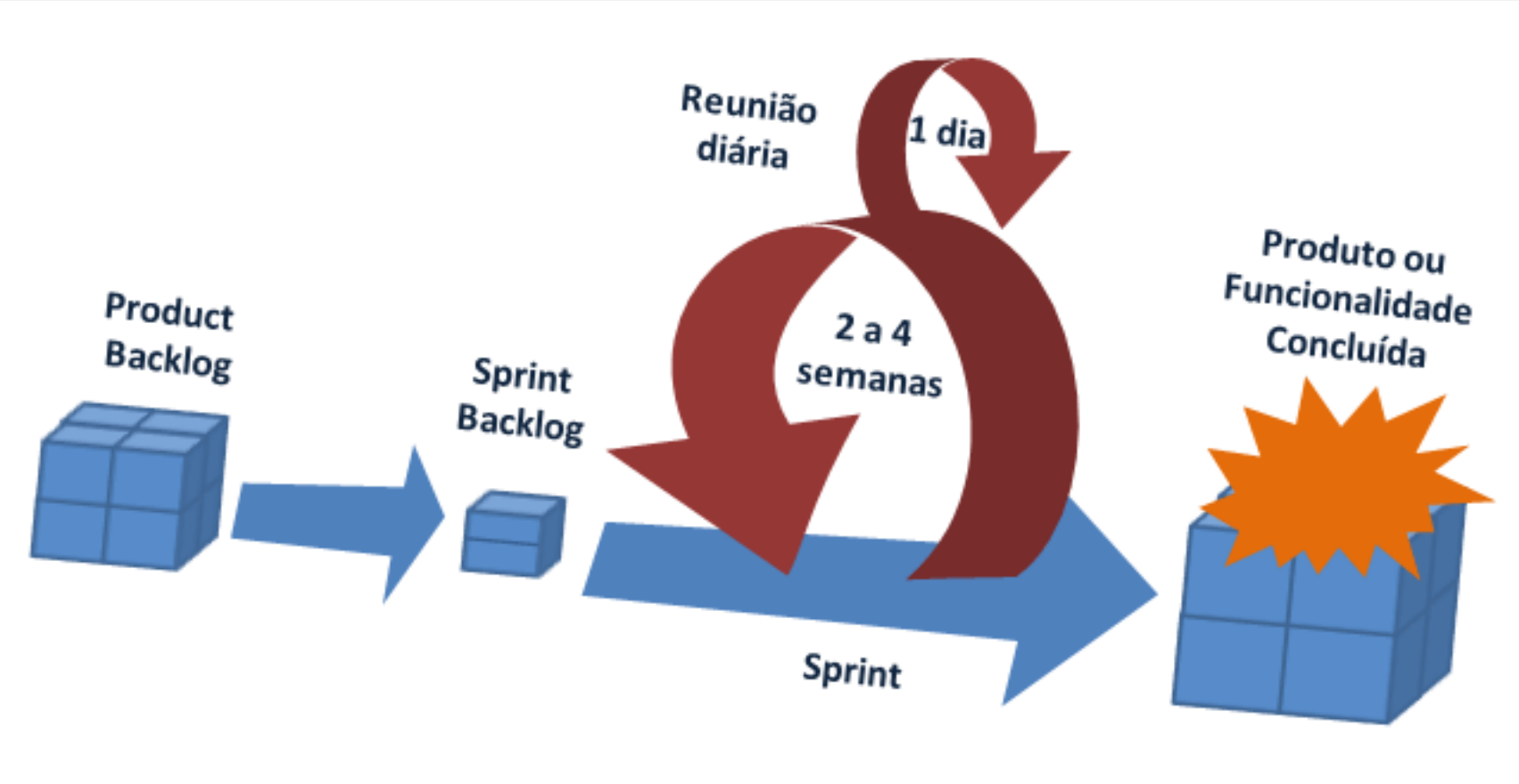
- Existem três fases no Scrum.
    - A fase inicial é uma fase de planejamento de esboço, onde você estabelece os objetivos gerais do projeto e projeta a arquitetura de software.

Scrum

- Isso é seguido por uma série de ciclos de sprint, onde cada ciclo desenvolve um incremento do sistema.

- A fase de encerramento do projeto encerra o projeto, completa a documentação necessária, como estruturas de ajuda do sistema e manuais do usuário, e avalia as lições aprendidas com o projeto.

Scrum



Retirado de https://mindmaster.com.br/scrum/

Scrum

- Scrum é dividido em papéis, cerimônias (reuniões, encontros) e artefatos.

- Os papéis são as funções que os participantes assumem durante o desenvolvimento do projetos; os papéis do Scrum são o Scrum Master, Product Owner e o Team.

Scrum

- As cerimônias são as reuniões, que na terminologia Scrum se dividem em Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective.

- Os artefatos, por sua vez, são os documentos produzidos através do processo: Product Backlog, Sprint Backlog, Definição de pronto e incremento, além de auxiliares como o Burndown Chart.

Scrum

- As cerimônias do Scrum incluem:

- Sprint Planning, uma reunião onde estão presentes o Scrum Master, Product Owner, Team e todos os demais stakeholders interessados. Nesta define-se o que será feito na Sprint atual

- Daily Scrum, uma reunião diária, rápida, onde são feitas três perguntas ao time:
  - O que você fez ontem?
  - O que vai fazer hoje?
  - Está com alguma dificuldade?

Scrum

- Sprint Review, onde estão presentes todos os participantes, afim de apresentar o incremento desenvolvido durante a Sprint.

- Sprint Retrospective, onde estão presentes o Scrum Master e o Team. Nesta reunião discute-se sobre a Sprint que acabou de ser entregue, pontos a melhorar e o que poderia ter sido feito diferentemente.

Scrum

- O Scrum Master é a pessoa responsável por garantir que o processo do Scrum seja seguido; em geral, é a pessoa que detém maior conhecimento sobre o framework.

- O Product Owner, por sua vez, é um representante do lado do cliente; é a pessoa responsável por direcionar o projeto de acordo com as necessidades dos Stakeholders.

- O Team, por sua vez, é composto pelos desenvolvedores que estão trabalhando no projeto.

Scrum

- O Scrum define alguns artefatos, que são documentos de apoio ao processo. Entre eles temos:

- Product Backlog, o conjunto de requisitos que deverá ser desenvolvido durante o projeto;

- Sprint Backlog, uma lista de requisitos a ser desenvolvido na atual Sprint;

- Definição de pronto, um documento de comum acordo entre Time, Scrum Master e PO, contendo uma definição do que significa “pronto” para o time

ESCALONAMENTO DE MÉTODOS ÁGEIS

Escalonando métodos ágeis

- Métodos ágeis provaram ser bem-sucedidos para projetos de pequeno e médio porte que podem ser desenvolvidos por uma pequena equipe co-localizada.

- Às vezes, argumenta-se que o sucesso desses métodos vem devido à melhoria das comunicações, que é possível quando todos estão trabalhando juntos.

- Aumentar a escala de métodos ágeis envolve alterá-los para lidar com projetos maiores e mais longos, onde há várias equipes de desenvolvimento, talvez trabalhando em locais diferentes.

Escalonando métodos ágeis

- Há distintas formas de escalonar métodos tais como o Scrum;

- Podemos escalonar o desenvolvimento de um sistema grande, inviável de ser desenvolvido por uma equipe pequena; Sommerville chama isto de “scalling up”;

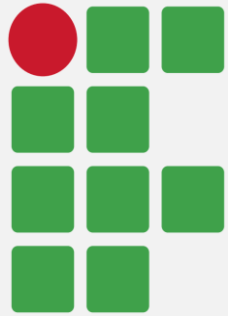
- Uma segunda forma de escalonar é pensando na forma como os ágeis podem ser introduzidos em uma grande organização (empresa), onde já há uma base estabelecida de desenvolvimento estabelecida; Sommerville chama isto de “Scalling out”;

Escalonando métodos ágeis

- Ao escalonar os ágeis, entretanto, é sempre importante manter-se fiel aos princípios fundamentais:
  - Planejamento flexível;
  - Releases frequentes (software funcionando como medida primário do progresso)
  - Integração contínua
  - Desenvolvimento dirigido a testes;
  - Boa comunicação entre os integrantes do time;

PROBLEMAS PRÁTICOS COM MÉTODOS ÁGEIS

Problemas práticos com métodos ágeis



- A “informalidade” do desenvolvimento ágil é incompatível com a abordagem jurídica para a definição de contratos comumente usada em grandes empresas.

- Os métodos ágeis são mais apropriados para o desenvolvimento de novo software, em vez da manutenção do software.

- No entanto, a maioria dos custos de software em grandes empresas vem da manutenção de seus sistemas de software existentes.

Problemas práticos com métodos ágeis

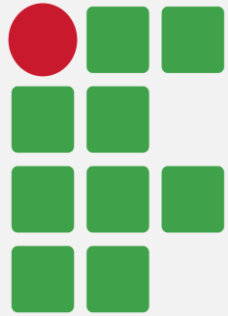
- Os métodos ágeis são projetados para pequenas equipes co-localizadas, mas muito do desenvolvimento de software agora envolve equipes distribuídas em todo o mundo.

Problemas práticos com métodos ágeis

- A maioria dos contratos de software para sistemas customizados é baseada em uma especificação, que define o que deve ser implementado pelo desenvolvedor do sistema para o cliente do sistema.

- No entanto, isso impede a especificação e o desenvolvimento de intercalação, como é a norma no desenvolvimento ágil.

Problemas práticos com métodos ágeis



- É necessário um contrato que paga pelo tempo do desenvolvedor em vez da funcionalidade.
    - No entanto, isso é visto como um alto risco para meus muitos departamentos jurídicos, porque o que deve ser entregue não pode ser garantido

Problemas práticos com métodos ágeis

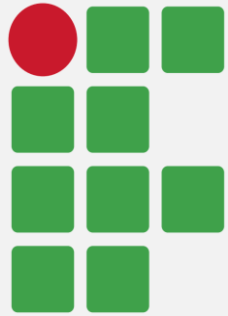
- A maioria das organizações gasta mais na manutenção do software existente do que no desenvolvimento de um novo software.

- Portanto, para que os métodos ágeis tenham sucesso, eles devem suportar tanto a manutenção quanto o desenvolvimento original.

Problemas práticos com métodos ágeis

- Duas questões principais:
  - Os sistemas desenvolvidos com uma abordagem ágil podem ser mantidos, dada a ênfase no processo de desenvolvimento de minimizar a documentação formal?
  - Os métodos ágeis podem ser usados de forma eficaz para desenvolver um sistema em resposta às solicitações de mudança do cliente?
- Podem surgir problemas se a equipe de desenvolvimento original não puder ser mantida.

Problemas práticos com métodos ágeis



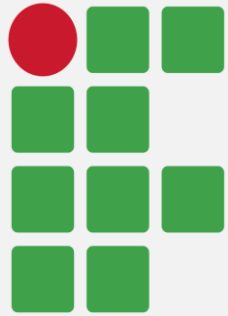
- Os principais problemas são:
  - Falta de documentação do produto
  - Manter os clientes envolvidos no processo de desenvolvimento
  - Manter a continuidade da equipe de desenvolvimento

Problemas práticos com métodos ágeis

- O desenvolvimento ágil depende da equipe de desenvolvimento saber e compreender o que deve ser feito.

- Para sistemas de longa duração, este é um problema real, pois os desenvolvedores originais nem sempre trabalharão no sistema.

Problemas práticos com métodos ágeis



- As práticas do XP, entretanto, podem nos ajudar a mitigar tais riscos. Um software com casos de testes auto-documentados por meio de TDD, ou BDD, pode mitigar a falta de continuidade do time;

- A produção da documentação necessária, conforme apontado por Robert C. Martin em seu livro “Princípios, Padrões e Práticas Ágeis” pode mitigar a suposta falta de documentação

MÉTODOS ÁGEIS VS DIRIGIDOS A PLANOS

Métodos ágeis vs dirigidos a planos

- A maioria dos projetos inclui elementos de processos ágeis e orientados a planos. A decisão sobre o equilíbrio depende de:
  - É importante ter uma especificação e um projeto muito detalhados antes de passar para a implementação? Neste caso, você provavelmente precisará usar uma abordagem orientada a planos.
  - É realista adotar uma estratégia de entrega incremental, onde você entrega o software aos clientes e obtém um feedback rápido deles? Nesse caso, considere o uso de métodos ágeis.

Métodos ágeis vs dirigidos a planos

- Também é necessário considerar qual é o tamanho do sistema que está sendo desenvolvido

- Os métodos ágeis são mais eficazes quando o sistema pode ser desenvolvido com uma pequena equipe co-localizada que pode se comunicar informalmente.

- Isso pode não ser possível para grandes sistemas que requerem equipes de desenvolvimento maiores, portanto, uma abordagem orientada a planos pode ter que ser usada.

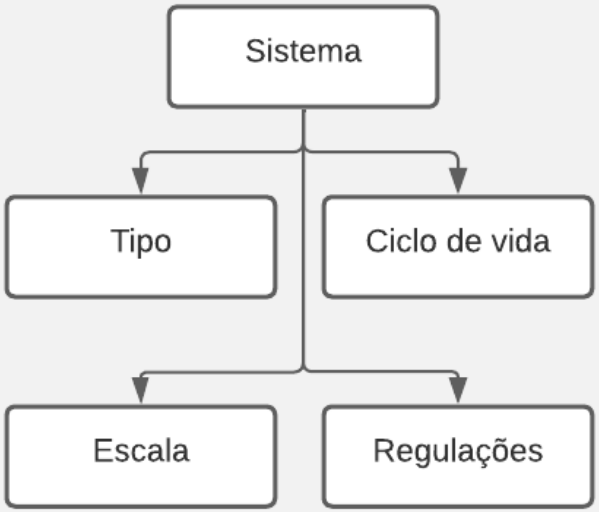
Métodos ágeis vs dirigidos a planos

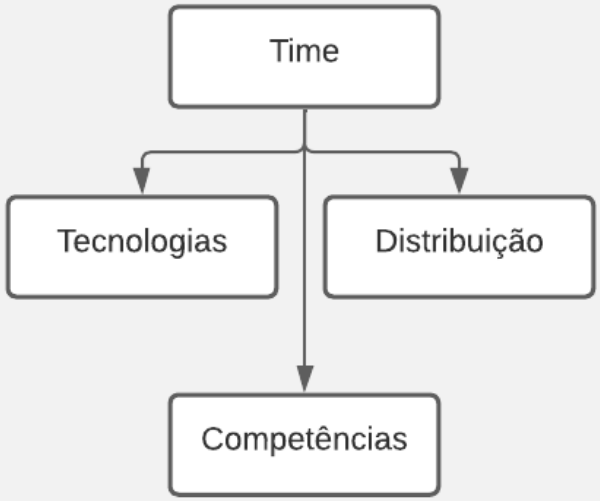
Princípio	Prática
Envolvimento do cliente	Isso depende de ter um cliente que deseja e é capaz de passar tempo com a equipe de desenvolvimento e que pode representar todas as partes interessadas do sistema. Frequentemente, os representantes do cliente têm outras demandas em seu tempo e não podem desempenhar um papel completo no desenvolvimento do software. Onde há stakeholders externos, como reguladores, é difícil representar seus pontos de vista para a equipe ágil.
Abraçar a mudança	Priorizar as mudanças pode ser extremamente difícil, especialmente em sistemas para os quais há muitos interessados. Normalmente, cada parte interessada dá prioridades diferentes para mudanças diferentes.

Métodos ágeis vs dirigidos a planos

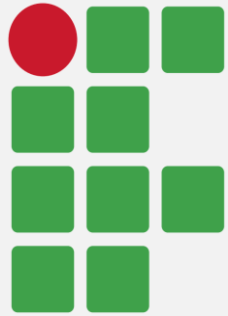
Princípio	Prática
Entrega incremental	As iterações rápidas e o planejamento de curto prazo para o desenvolvimento nem sempre se encaixam nos ciclos de planejamento de longo prazo do planejamento de negócios e marketing. Os gerentes de marketing podem precisar saber quais são as características do produto com vários meses de antecedência para preparar uma campanha de marketing eficaz.
Manter a simplicidade	Sob pressão dos cronogramas de entrega, os membros da equipe podem não ter tempo para realizar as simplificações desejáveis do sistema.
Pessoas, não processos	Membros individuais da equipe podem não ter personalidades adequadas para o envolvimento intenso que é típico dos métodos ágeis e, portanto, podem não interagir bem com outros membros da equipe.

Métodos ágeis vs dirigidos a planos





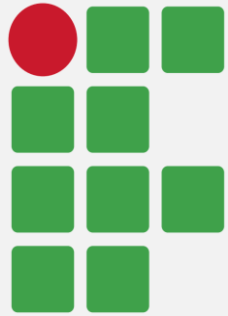
Métodos ágeis vs dirigidos a planos



- Qual é o tamanho do sistema que está sendo desenvolvido?
  - Os métodos ágeis são mais eficazes quando uma equipe co-localizada relativamente pequena pode se comunicar informalmente.

- Que tipo de sistema está sendo desenvolvido?
  - Os sistemas que requerem muita análise antes da implementação precisam de um design bastante detalhado para realizar essa análise.

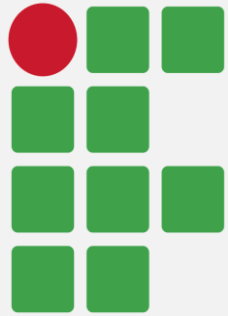
Métodos ágeis vs dirigidos a planos



- Qual é a expectativa de vida do sistema?
  - Os sistemas de longa duração requerem documentação para comunicar as intenções dos desenvolvedores do sistema à equipe de suporte.

- O sistema está sujeito a regulamentação externa?
  - Se um sistema for regulamentado, você provavelmente precisará produzir documentação detalhada como parte do caso de segurança do sistema.

Métodos ágeis vs dirigidos a planos



- Os designers e programadores da equipe de desenvolvimento são bons?
    - Às vezes, é argumentado que os métodos ágeis requerem níveis de habilidade mais altos do que as abordagens baseadas em planos, nas quais os programadores simplesmente traduzem um projeto detalhado em código.
- Como a equipe de desenvolvimento está organizada?
    - Documentos de projeto podem ser necessários se a equipe for distribuída.
- Quais tecnologias de suporte estão disponíveis?
    - O suporte de uma IDE para visualização e análise do programa é essencial se a documentação do projeto não estiver disponível.

Métodos ágeis vs dirigidos a planos

- As organizações tradicionais de engenharia têm uma cultura de desenvolvimento baseado em planos, pois esta é a norma na engenharia.

- É prática organizacional padrão desenvolver uma especificação detalhada do sistema?

- Os representantes do cliente estarão disponíveis para fornecer feedback sobre os incrementos do sistema?

- O desenvolvimento ágil “informal” pode se encaixar na cultura organizacional de documentação detalhada?

Métodos ágeis vs dirigidos a planos

- Os grandes sistemas são geralmente conjuntos de sistemas separados de comunicação, onde equipes separadas desenvolvem cada sistema. Frequentemente, essas equipes estão trabalhando em lugares diferentes, às vezes em fusos horários diferentes.

- Grandes sistemas são "sistemas brownfield", isto é, eles incluem e interagem com uma série de sistemas existentes. Muitos dos requisitos do sistema estão preocupados com essa interação e, portanto, não se prestam realmente a flexibilidade e desenvolvimento incremental.

Métodos ágeis vs dirigidos a planos

- Onde vários sistemas são integrados para criar um sistema, uma fração significativa do desenvolvimento está relacionada à configuração do sistema, e não ao desenvolvimento do código original.

- Grandes sistemas e seus processos de desenvolvimento são frequentemente limitados por regras e regulamentos externos que limitam a maneira como podem ser desenvolvidos.

Métodos ágeis vs dirigidos a planos

- Grandes sistemas têm um longo tempo de aquisição e desenvolvimento. É difícil manter equipes coerentes que conheçam o sistema durante esse período, pois, inevitavelmente, as pessoas passam para outras funções e projetos.

- Os grandes sistemas geralmente possuem um conjunto diversificado de partes interessadas. É praticamente impossível envolver todas essas diferentes partes interessadas no processo de desenvolvimento.

Métodos ágeis vs dirigidos a planos

MÉTODOS ÁGEIS PARA SISTEMAS DE LARGA ESCALA

Métodos ágeis para sistemas de larga escala

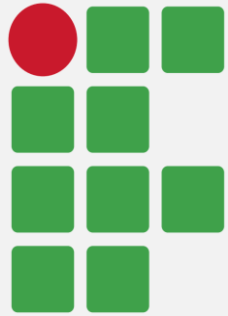
- Para sistemas de larga escala uma abordagem totalmente incremental para a engenharia de requisitos é impossível.
  - Não pode haver um único proprietário do produto ou representante do cliente.
  - Para o desenvolvimento de grandes sistemas, não é possível focar apenas no código do sistema.
  - Os mecanismos de comunicação entre equipes devem ser projetados e usados.
  - A integração contínua é praticamente impossível. No entanto, é essencial manter compilações e versões regulares do sistema.

Métodos ágeis para sistemas de larga escala

- É possível, contudo, escalonar métodos como o Scrum para projetos maiores e multi-times. Isso se dá por meio de:

- Replicação de papéis
    - Cada equipe tem um Product Owner para seu componente de trabalho e um Scrum Master.
- Arquitetos de produto
    - Cada equipe escolhe um arquiteto de produto e esses arquitetos colaboram para projetar e desenvolver a arquitetura geral do sistema.

Métodos ágeis para sistemas de larga escala



- Alinhamento de liberação
  - As datas de lançamento dos produtos de cada equipe são alinhadas para que um sistema demonstrável e completo seja produzido.
- Scrum de Scrums
  - Há um Scrum de Scrums diário, onde os representantes de cada equipe se encontram para discutir o progresso e planejar o trabalho a ser feito.

Métodos ágeis para sistemas de larga escala

- Os gerentes de projeto que não têm experiência com métodos ágeis podem relutar em aceitar o risco de uma nova abordagem.

- As grandes organizações costumam ter procedimentos e padrões de qualidade que todos os projetos devem seguir e, devido à sua natureza burocrática, são provavelmente incompatíveis com métodos ágeis.

Métodos ágeis para sistemas de larga escala

- Os métodos ágeis parecem funcionar melhor quando os membros da equipe têm um nível de habilidade relativamente alto. No entanto, em grandes organizações, é provável que haja uma ampla gama de habilidades e habilidades.

- Pode haver resistência cultural aos métodos ágeis, especialmente nas organizações que têm um longo histórico de uso de processos convencionais de engenharia de sistemas.

PONTOS CHAVE

Pontos chave

- Os métodos ágeis são métodos de desenvolvimento incrementais que se concentram no desenvolvimento rápido de software, lançamentos frequentes do software, reduzindo sobrecargas de processo ao minimizar a documentação e produzir código de alta qualidade

Pontos chave

- As práticas de desenvolvimento ágil incluem
  - Histórias de usuário para especificação do sistema
  - Versões frequentes do software,
  - Melhoria contínua de software
  - Desenvolvimento test-first
  - Participação do cliente na equipe de desenvolvimento.

Pontos chave

- Scrum é um método ágil que fornece uma estrutura de gerenciamento de projetos.
    - É centrado em torno de um conjunto de sprints, que são períodos de tempo fixos quando um incremento do sistema é desenvolvido.

- Muitos métodos práticos de desenvolvimento são uma mistura de desenvolvimento ágil e baseado em planos.

Pontos chave

- É difícil escalar métodos ágeis para grandes sistemas.
    - Grandes sistemas precisam de um projeto inicial e alguma documentação. Práticas organizacionais podem entrar em conflito com a informalidade das abordagens ágeis.

Referências

- SOMMERVILLE, Ian. Engenharia de software 9 ed. São Paulo, SP: Pearson Prentice Hall, 568p. ISBN: 9788579361081, 2011.

- PRESSMAN, R. S Engenharia de Software - Uma abordagem Profissional - 8 ed. Porto Alegre, RS: Bookman/Amgh Editora, 968 p. ISBN: 9788580555332, 2016.