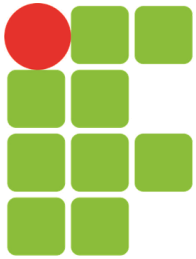
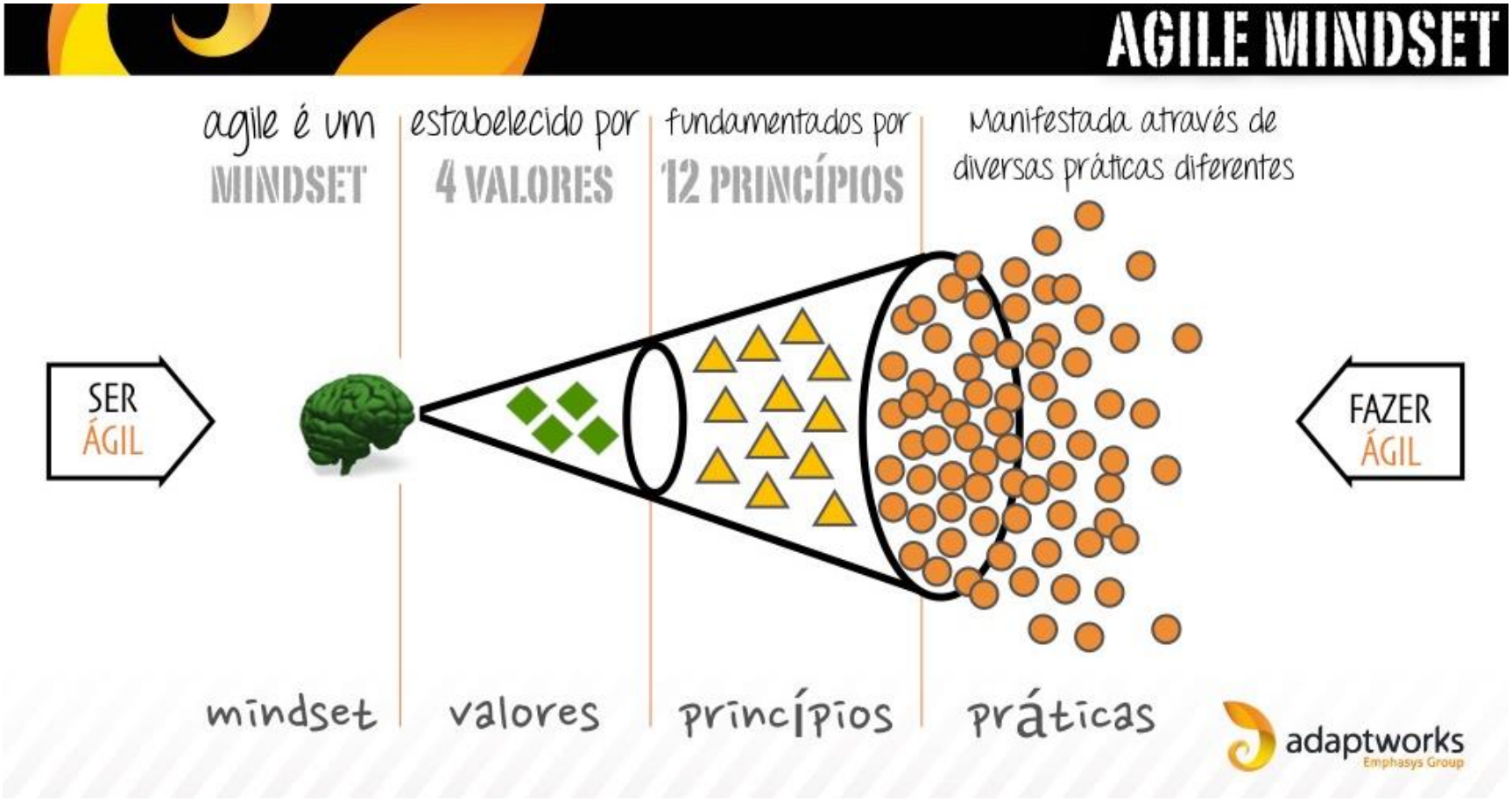


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO SUL
Campus Feliz

Desenvolvimento Ágil de Software

Desenvolvimento Ágil de Software

Desenvolvimento Ágil de Software



Desenvolvimento Ágil de Software

- As metodologias ágeis (ou leves) são processos de desenvolvimento de software, em geral, empregados por organizações que dão **ênfase à colaboração** baseada numa **abordagem flexível**.
- Ideais para projetos nos quais os **requisitos mudam constantemente**, em decorrência do mercado, da organização, do projeto e do conhecimento.
- O Manifesto Ágil foi redigido em 2001.

Desenvolvimento Ágil de Software

Manifesto Ágil

- Valores Fundamentais do Manifesto Ágil:

Indivíduos e **interações** ao invés de processos e ferramentas.

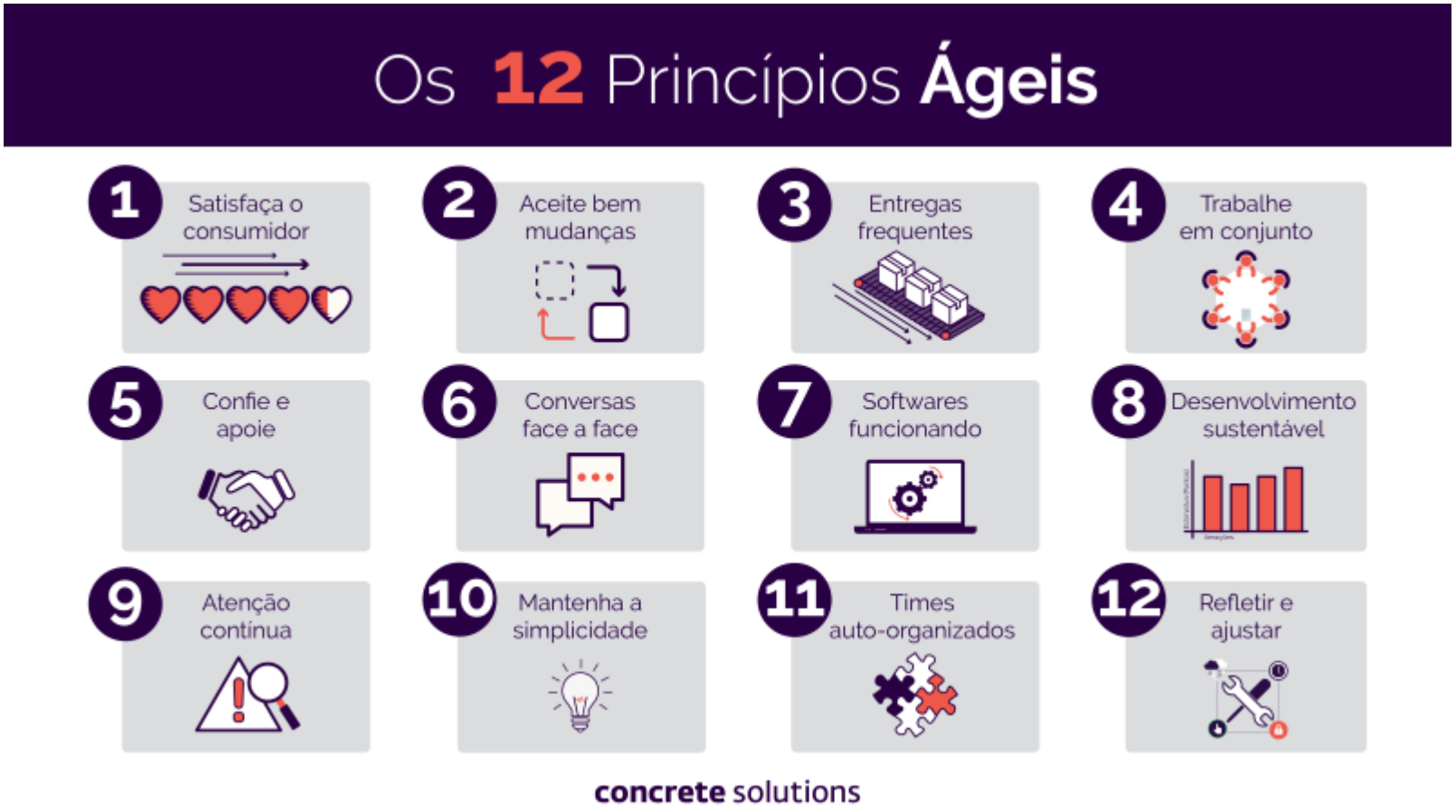
Software funcionando ao invés de documentação abrangente.

Colaboração com o cliente ao invés de negociação de contratos.

Resposta a mudanças ao invés de obediência a um plano.

Desenvolvimento Ágil de Software

Princípios Ágeis



Desenvolvimento Ágil de Software

Manifesto Ágil

Os 12 princípios por trás do Manifesto Ágil:

1. Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
3. Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
4. Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
5. Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.

Fonte: http://agilemanifesto.org/principles.html (em inglês)
http://www.manifestoagil.com.br/principios.html (em português)

Desenvolvimento Ágil de Software

Manifesto Ágil

Os 12 princípios por trás do Manifesto Ágil (continuação):

6. O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
7. Software funcional é a medida primária de progresso.
8. Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
9. Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
11. As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis.
12. Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

Fonte: http://agilemanifesto.org/principles.html (em inglês)
http://www.manifestoagil.com.br/principios.html (em português)

Desenvolvimento Ágil de Software

Características Fundamentais

Características fundamentais compartilhadas pelas abordagens para o desenvolvimento ágil:

1. Os processos de especificação, projeto e implementação são intercalados.
   - Não há especificação detalhada do sistema, a documentação é minimizada e o documento de requisitos apenas define as características mais importantes.
2. O sistema é desenvolvido em uma série de versões.
   - Usuários finais e *stakeholders* estão envolvidos na especificação e avaliação de cada versão (alterações e novos requisitos são tratados em uma versão posterior).
3. Interfaces de usuário são geralmente projetadas com sistemas interativos que permitem desenhar e posicionar ícones.
   - O sistema pode gerar uma interface baseada na Web ou para Windows, por exemplo, com base neste projeto de interface.

Desenvolvimento Ágil de Software

Fatores Humanos [1..2]

- Fatores humanos são fundamentais para o sucesso do desenvolvimento ágil de software.
  - "*O processo se molda às necessidades das pessoas e equipes*, e não o caminho inverso."
- Características de uma equipe ágil:
  - **Competência** (talento inato, habilidades específicas relacionadas a software e conhecimento generalizado do processo)
  - **Foco comum** (entregar o incremento dentro do prazo)
  - **Colaboração**
  - **Habilidade na tomada de decisão**
  - *Continua...*

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pp. 86-87).

Desenvolvimento Ágil de Software

Fatores Humanos [2..2]

- Características de uma equipe ágil (continuação):
  - **Habilidade de solução de problemas confusos** (a equipe precisa saber lidar continuamente com ambiguidade e mudança)
  - **Confiança mútua e respeito**
  - **Auto-organização**
    - A equipe se organiza para o trabalho a ser feito
    - A equipe organiza o processo de forma adequada ao seu ambiente
    - A equipe organiza o cronograma de trabalho

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pp. 86-87).

Desenvolvimento Ágil de Software

Exemplos

- Scrum
- *Extreme Programming* – XP
- Desenvolvimento de Software Adaptativo (ASD – *Adaptive Software Development*)
- Método de desenvolvimento de Sistemas Dinâmicos (DSDM – *Dynamic Systems Devolopment Method*)
- Desenvolvimento dirigido a funcionalidades (FDD – *Feature Driven Development*)
- Crystal
- Desenvolvimento guiado por testes (TDD - *Test Driven Development*)
- Desenvolvimento guiado por comportamento (BDD - *Behavior Driven Development*)

Scrum

Visão Geral [1..3]

- É um *framework* de desenvolvimento ágil concebido no início dos anos 90.
  - "Scrum não é um processo ou uma técnica para construir produtos; em vez disso, é um *framework* dentro do qual você pode empregar vários processos ou técnicas" [Guia Scrum, 17]
- Seus princípios são consistentes com o Manifesto Ágil.
- Atividades estruturais:
  - Requisitos
  - Análise
  - Projeto
  - Evolução
  - Entrega

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pp. 95-96).

Scrum

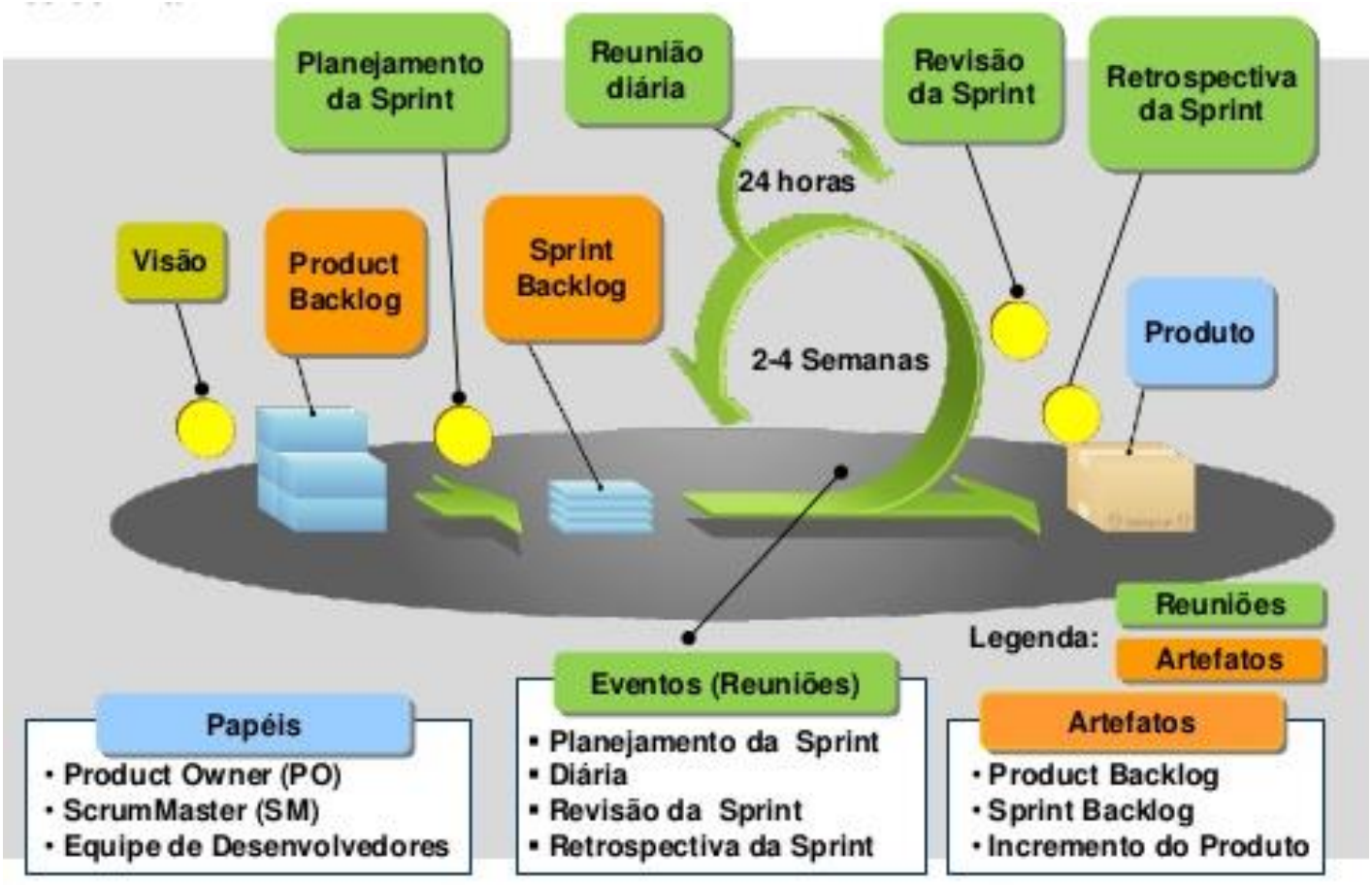
Visão Geral [2..3]

- Conceito central: ***Sprint***
  - Um *sprint* é uma unidade de planejamento onde se avalia o que precisa ser feito (a partir do *backlog** do produto), seleciona-se os recursos e implementa-se o software.
  - A funcionalidade completa é entregue aos *stakeholders* no final do *sprint*.
  - *Sprints* tem comprimento fixo, variando de duas a quatro semanas.

* Lista do trabalho a ser feito no projeto.

Scrum

Visão Geral [3..3]



Fonte da imagem: https://www.slideshare.net/Ridlo/scrum-o-tutorial-definitivo

Scrum

Time Scrum

- O Time Scrum é composto por:
  - ***Product Owner*** *– PO*
  - **Time de Desenvolvimento** (de 3 a 9 integrantes)
  - **Scrum *Master***

- São times auto organizáveis e multifuncionais.
  - Times auto organizáveis **escolhem** qual a melhor forma para completarem seu trabalho.
  - Times multifuncionais **possuem todas as competências necessárias** para completar o trabalho sem depender de outros que não fazem parte da equipe.

Fonte: SCHWABER, K., SUTHERLAND, J. Guia do Scrum. 2016 (pág. 5)

Scrum

Eventos (ou Cerimônias)

- São definidos os seguintes eventos no Scrum:
  - **Sprint** (máx. **30 dias**)
  - **Sprint *Planning*** (máx. **8 horas**)
  - ***Daily* Scrum** (máx. **15 min**)
  - **Sprint *Review*** (máx. **4 horas**)
  - **Sprint *Retrospective*** (máx. **3 horas**)
- Todos os eventos são ***time-boxed*** (tem uma duração máxima).

Fonte: SCHWABER, K., SUTHERLAND, J. Guia do Scrum. 2016 (pág. 5)

Scrum
Artefatos

- São definidos os seguintes artefatos no Scrum:
  - ***Product Backlog***
  - ***Sprint Backlog***
  - **Incremento**
    - **Definição de pronto**: quando o item do Backlog do Produto ou um incremento é descrito como “Pronto”, todos devem entender o que o “Pronto” significa.

Fonte: SCHWABER, K., SUTHERLAND, J. Guia do Scrum. 2016 (pág. 5)

Scrum

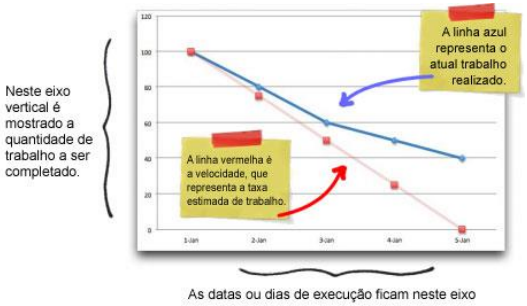
Técnicas e Artefatos utilizados com frequência

- Times Scrum utilizam com frequência as/os seguintes técnicas/artefatos*:

Planning Poker

Quadro de Kanban

Gráfico de Burndown

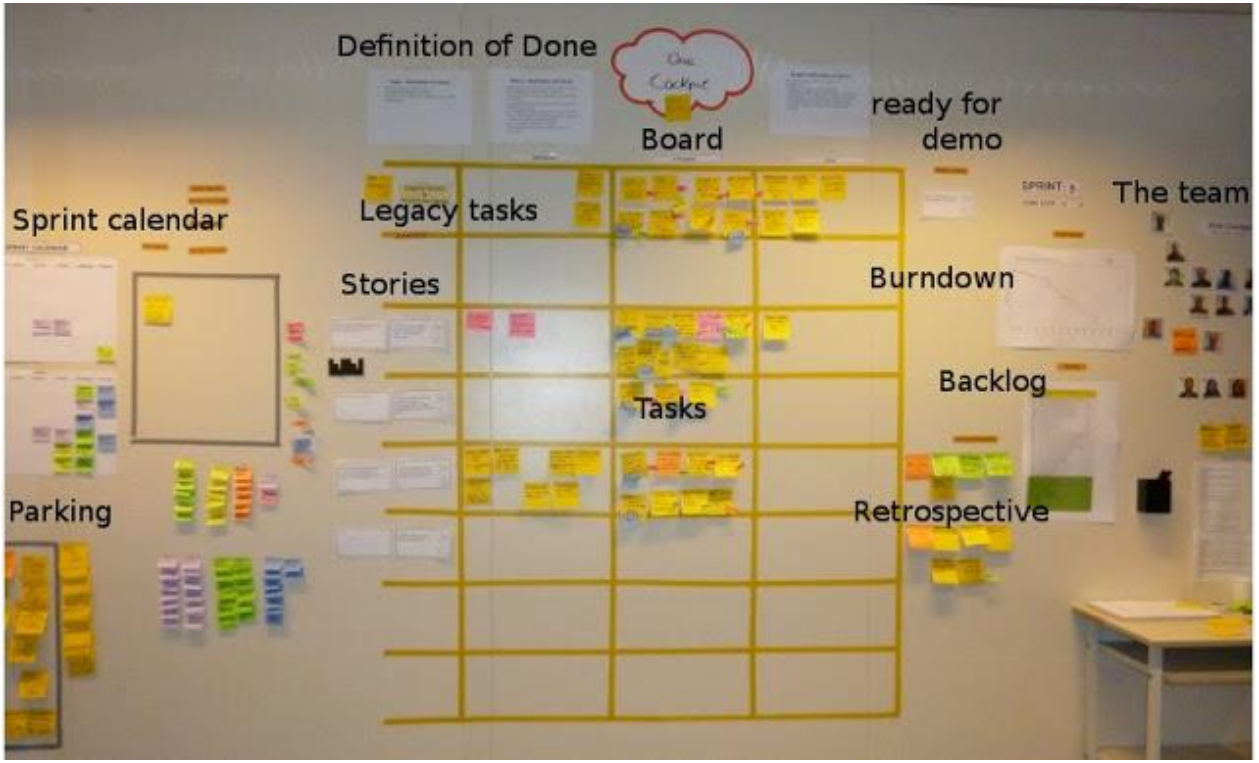


* O uso de tais técnicas/artefatos são narrados em sítios especializados na Web.

Scrum

Técnicas e Artefatos utilizados com frequência

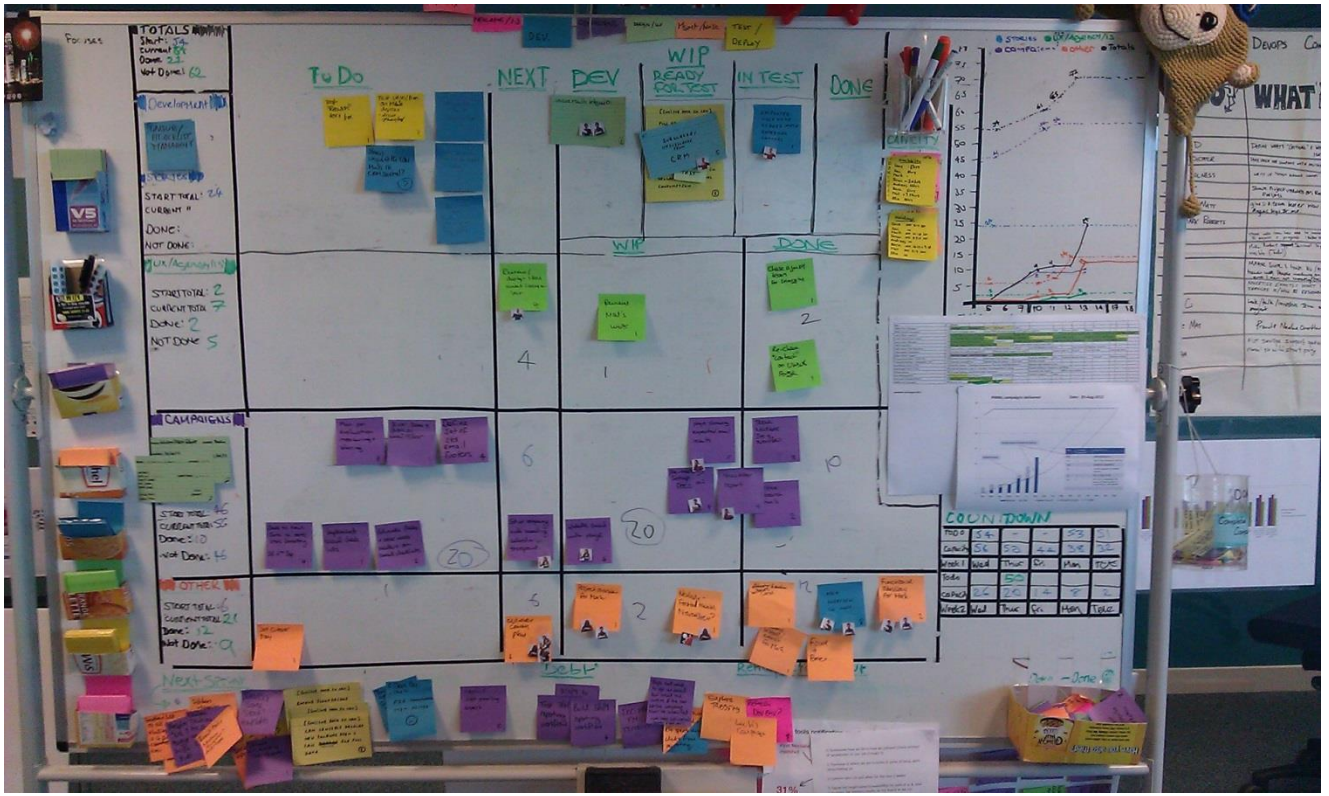
Exemplos de Scrum *Boards*



Scrum

Técnicas e Artefatos utilizados com frequência

Exemplos de Scrum *Boards*



Extreme Programming – XP [1..6]

- O nome foi cunhado em 2000 (mas há trabalhos desde o final dos anos 80).
- Valores da XP: comunicação, simplicidade, *feedback*, coragem (disciplina) e respeito. [Pressman, 2011].
- Características [Sommerville, 2011]:
  - Os requisitos são expressos como cenários (chamados de estória do usuário).
  - Os programadores trabalham em pares e desenvolvem os testes para cada tarefa ANTES de escrevem o código.
  - Quando um novo código é integrado ao sistema, todos os testes devem ser executados com sucesso.
  - O intervalo de tempo entre as *releases* é curto.

Extreme Programming – XP [2..6]

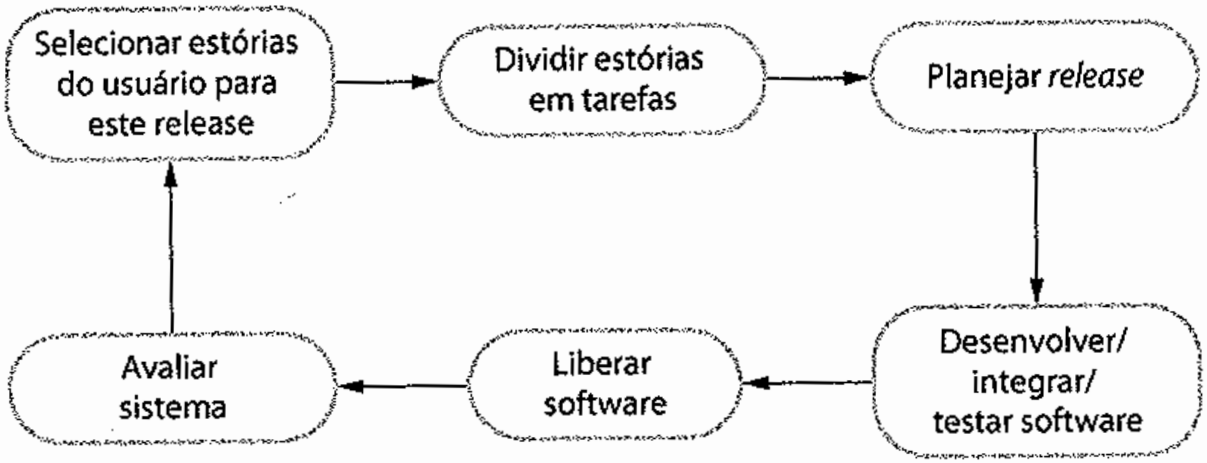
Tabela 3.2 Práticas de Extreme Programming

Princípio ou prática	Descrição
Planejamento incremental	Os requisitos são gravados em cartões de estória e as estórias que serão incluídas em um release são determinadas pelo tempo disponível e sua relativa prioridade. Os desenvolvedores dividem essas estórias em 'Tarefas'. Veja os quadros 3.1 e 3.2.
Pequenos *releases*	Em primeiro lugar, desenvolve-se um conjunto mínimo de funcionalidades útil, que fornece o valor do negócio. *Releases* do sistema são frequentes e gradualmente adicionam funcionalidade ao primeiro *release*.
Projeto simples	Cada projeto é realizado para atender às necessidades atuais, e nada mais.
Desenvolvimento *test-first*	Um *framework* de testes iniciais automatizados é usado para escrever os testes para uma nova funcionalidade antes que a funcionalidade em si seja implementada.
Refatoração	Todos os desenvolvedores devem refatorar o código continuamente assim que encontrarem melhorias de código. Isso mantém o código simples e manutenível.
Programação em pares	Os desenvolvedores trabalham em pares, verificando o trabalho dos outros e prestando apoio para um bom trabalho sempre.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de modo que não se desenvolvam ilhas de *expertise*. Todos os conhecimentos e todos os desenvolvedores assumem 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 ao sistema como um todo. Após essa integração, todos os testes de unidade do sistema devem passar.
Ritmo sustentável	Grandes quantidades de horas-extra não são consideradas aceitáveis, pois o resultado final, muitas vezes, é a redução da qualidade do código e da produtividade a médio prazo.
Cliente no local	Um representante do usuário final do sistema (o cliente) deve estar disponível todo o tempo à equipe de XP. Em um processo de Extreme Programming, o cliente é um membro da equipe de desenvolvimento e é responsável por levar a ela os requisitos de sistema para implementação.

Extreme Programming – XP [3..6]

- O ciclo de um *release* [Sommerville, 2011]:

Figura 3.2 O ciclo de um *release* em Extreme Programming



Extreme Programming – XP [4..6]

Quadro 3.1 Uma estória de prescrição de medicamentos

Prescrição de medicamentos

Kate é uma médica que deseja prescrever medicamentos para um paciente de uma clínica. O prontuário do paciente já está sendo exibido em seu computador, assim, ela clica o campo 'medicação' e pode selecionar 'medicação atual', 'nova medicação', ou 'formulário'.

Se ela selecionar 'medicação atual', o sistema pede que ela verifique a dose. Se ela quiser mudar a dose, ela altera esta e em seguida, confirma a prescrição.

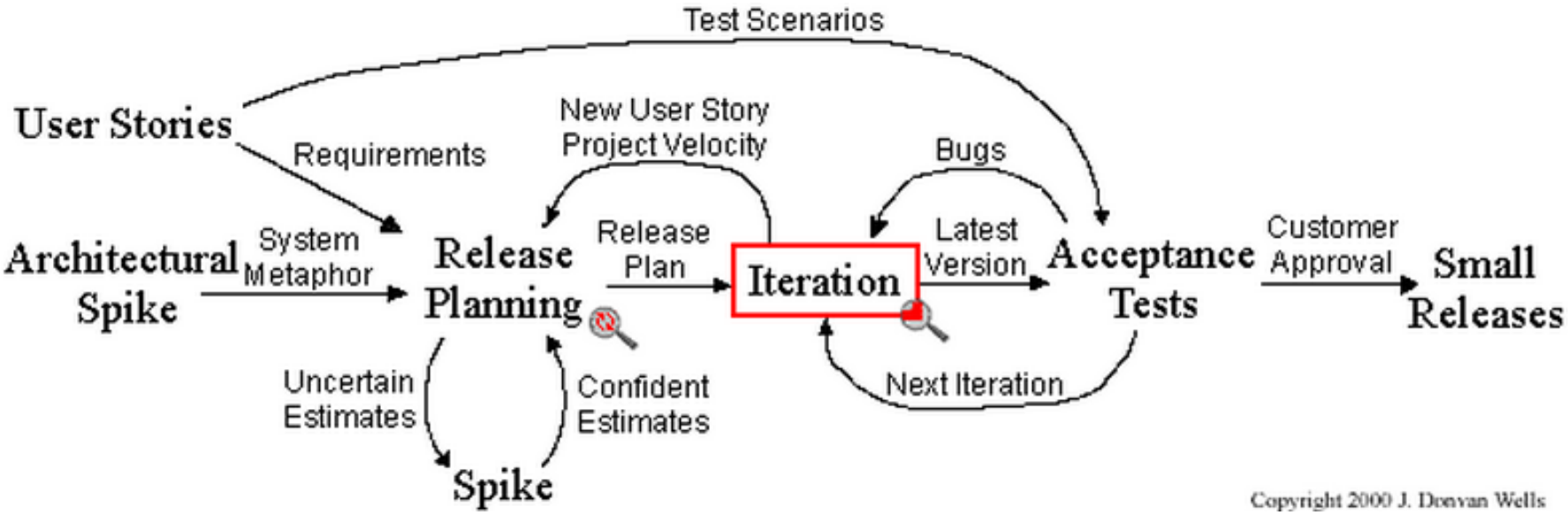
Se ela escolher 'nova medicação', o sistema assume que ela sabe qual medicação receitar.
Ela digita as primeiras letras do nome do medicamento. O sistema exibe uma lista de possíveis fármacos que começam com essas letras. Ela escolhe a medicação requerida e o sistema responde, pedindo-lhe para verificar se o medicamento selecionado está correto.
Ela insere a dose e, em seguida, confirma a prescrição.

Se ela escolhe 'formulário', o sistema exibe uma caixa de busca para o formulário aprovado.
Ela pode, então, procurar pelo medicamento requerido. Ela seleciona um medicamento e é solicitado que verifique se a medicação está correta. Ela insere a dose e, em seguida, confirma a prescrição.

O sistema sempre verifica se a dose está dentro da faixa permitida. Caso não esteja, Kate é convidada a alterar a dose.

Após Kate confirmar a prescrição, esta será exibida para verificação. Ela pode escolher 'OK' ou 'Alterar'. Se clicar em 'OK', a prescrição fica gravada nos bancos de dados da auditoria.
Se ela clicar em 'Alterar', reinicia o processo de 'Prescrição de Medicamentos'.

Extreme Programming – XP [5..6]



Fonte: http://www.extremeprogramming.org/map/project.html.

Extreme Programming – XP [6..6]

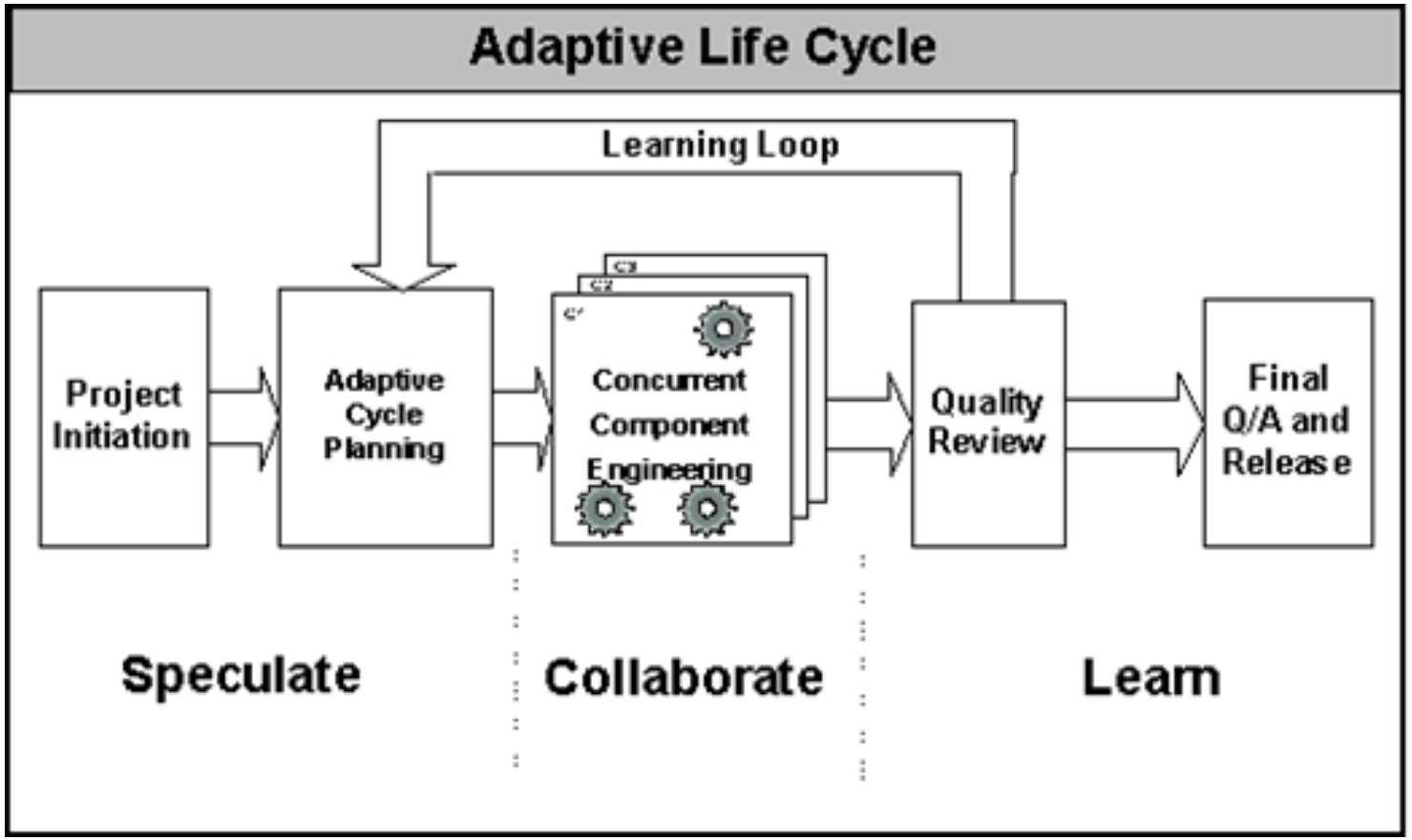
- Críticas feitas ao XP [Pressman, 2011, pág. 92]:
  - Volatilidade de requisitos
  - Necessidades conflitantes de clientes
  - Requisitos identificados informalmente
  - Falta de projeto formal

Desenvolvimento de Software Adaptativo [1..2]

- Proposto em 2000 como uma técnica para construção de software e sistemas complexos.
- Bases: colaboração humana e auto-organização das equipes.
- Incorpora três fases:
  - **Especulação**: é onde se define o conjunto de ciclos de versão (incrementos) a partir da missão do cliente, das restrições do projeto e dos requisitos básicos.
  - **Colaboração**: envolve comunicação e trabalho em equipe. Foco na confiança entre os membros da equipe.
  - **Aprendizagem**: aumenta os níveis reais de entendimento (desenvolvedores geralmente superestimam seu próprio entendimento de tecnologia, do processo e do projeto).

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pp. 94-95).

Desenvolvimento de Software Adaptativo [2..2]



Método de desenvolvimento de Sistemas Dinâmicos [1..2]

- Baseia-se numa versão modificada do princípio de Pareto: 80% de uma aplicação pode ser entregue em 20% do tempo que levaria para entregar a aplicação completa.
- É um processo iterativo, onde cada iteração segue a regra do 80% (ou seja, somente o trabalho suficiente é requisitado).

- Para saber mais, acesse: dsdm.org

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pp. 96-97).

Método de desenvolvimento de Sistemas Dinâmicos [2..2]

- Ciclo de vida DSDM:
  - Estudo da viabilidade
  - Estudo do negócio
  - Iteração de modelos funcionais
  - Iteração de projeto e desenvolvimento
  - Implementação

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pp. 96-97).

Bibliografia

- Sommerville, Ian. **Engenharia de Software**. São Paulo: Pearson Addison-Wesley. 2007, 8ª edição.

Bibliografia adicional:

- Pressman, Roger. **Engenharia de Software**. São Paulo: McGraw-Hill, 2006, 6ª edição.
- Kruchten, Philippe. **Introdução ao RUP - *Rational Unified Process***. Ciência Moderna. 2003, 1ª edição.