

Desenvolvimento Rápido de Aplicação - RAD

- É o modelo sequencial linear mas que enfatiza um desenvolvimento extremamente rápido.
- A “alta velocidade” é conseguida através de uma abordagem de construção baseada em componentes.
- O desenvolvimento rápido de aplicação é adequado para construção de aplicações em curto espaço de tempo, utilizando o método incremental da prototipação, com o ciclo extremamente curto de desenvolvimento, que só é possível quando temos grande entendimento dos requisitos e o projeto é curto e simples.
- É um modelo de processo de desenvolvimento de software iterativo e incremental que enfatiza um ciclo de desenvolvimento extremamente curto (entre 60 e 90 dias).



Prof. Anderson Augusto Bosing

Desenvolvimento Rápido de Aplicação - RAD

- A aplicação deve ser modularizada de forma que cada função deva ser completada em pelo menos 3 meses.
- Cada modulo pode ser alocado a uma equipe distinta e depois serem integradas para formar um todo.
- Criação e reutilização de componentes.
- No RAD, há o sequenciamento existente no modelo cascata e a ideia de apresentações parciais ao cliente para validação e continuação do desenvolvimento, vinda do modelo de prototipação, porém, nesse modelo, o que é apresentado ao cliente é, de fato, parte do software, e não apenas um protótipo.



Prof. Anderson Augusto Bosing

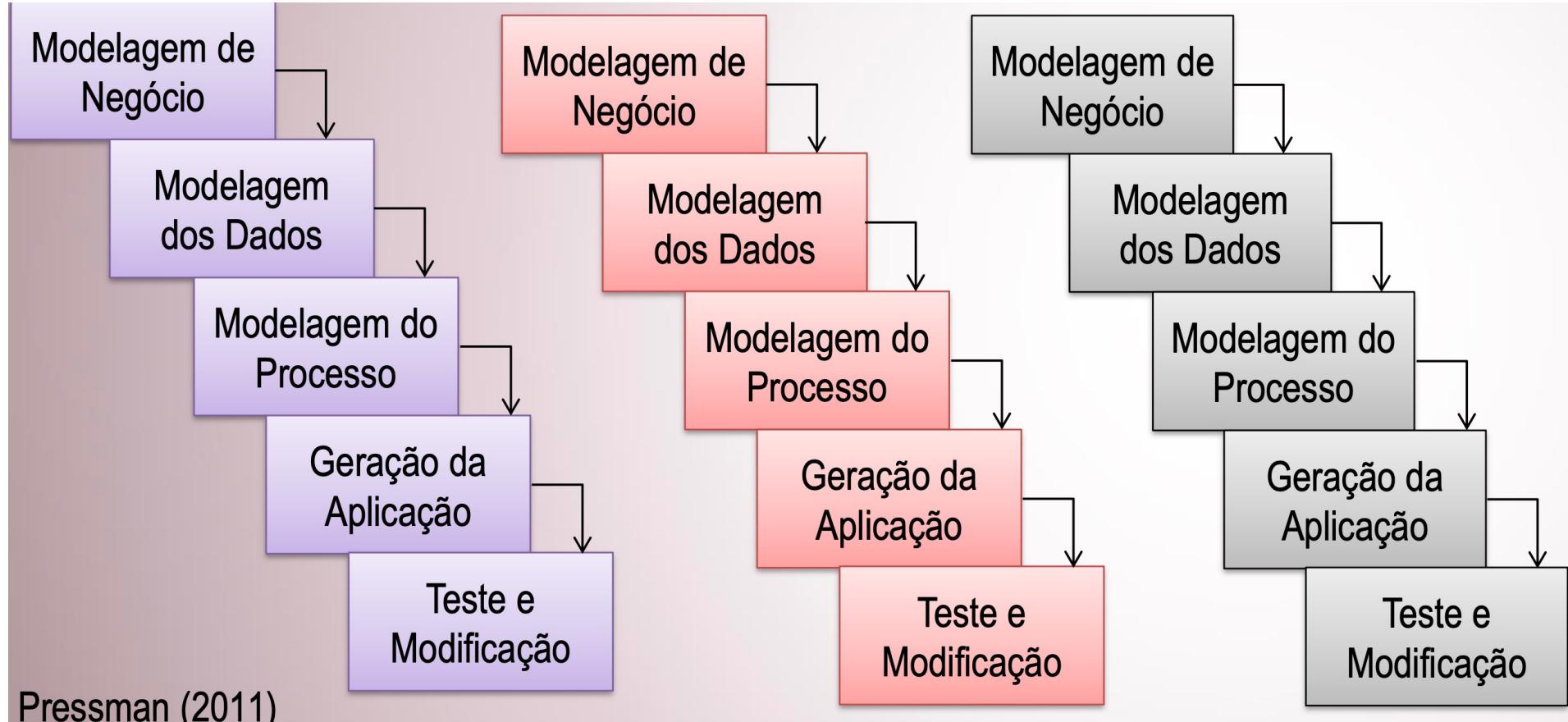
Desenvolvimento Rápido de Aplicação - RAD

- A aplicação deve ser modularizada de forma que cada função deva ser completada em pelo menos 3 meses.
- Cada modulo pode ser alocado a uma equipe distinta e depois serem integradas para formar um todo.
- Criação e reutilização de componentes.
- No RAD, há o sequenciamento existente no modelo cascata e a ideia de apresentações parciais ao cliente para validação e continuação do desenvolvimento, vinda do modelo de prototipação, porém, nesse modelo, o que é apresentado ao cliente é, de fato, parte do software, e não apenas um protótipo.



Prof. Anderson Augusto Bosing

Desenvolvimento Rápido de Aplicação - RAD



Modelo Incremental

- Segundo Pressman (2011) “modelos evolucionários são iterativos. Apresentam características que possibilitam desenvolver versões cada vez mais completas do software”.
- Dessa forma, o primeiro modelo evolucionário que surgiu é o modelo incremental de desenvolvimento de software.
- Para Sommerville (2014) o sistema incremental tem o objetivo de reduzir o retrabalho custoso do modelo cascata, possibilitando ao cliente postergar decisões e requisitos conforme necessidade, mas mantendo o poder de gerenciamento que o modelo oferece.



Prof. Anderson Augusto Bosing

Modelo Incremental

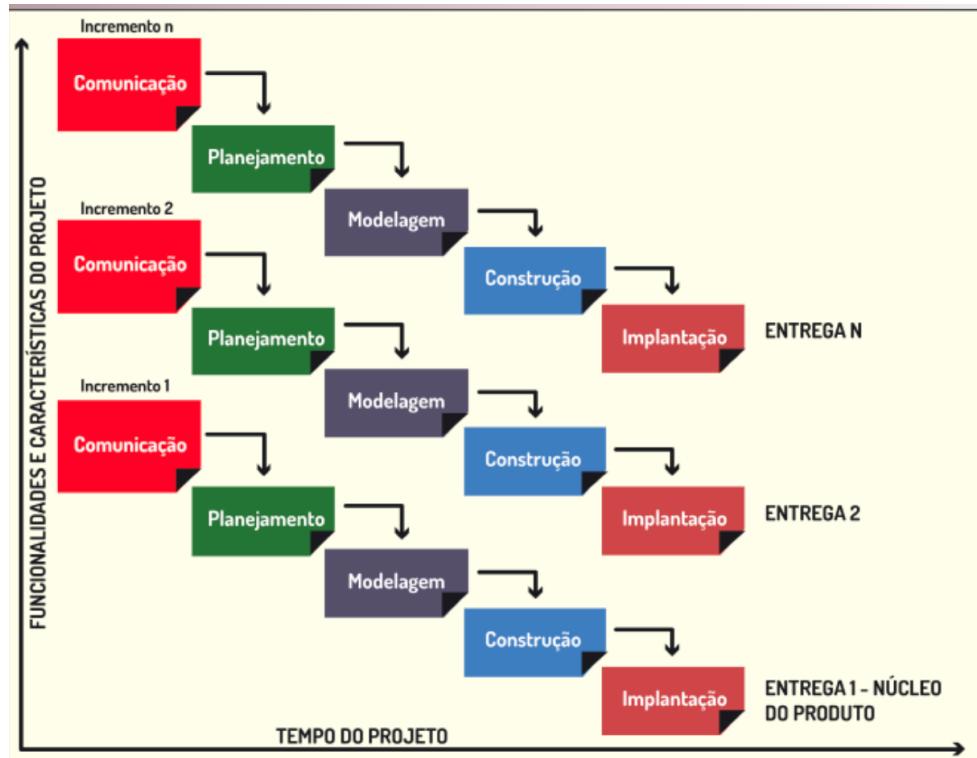
- Essa metodologia divide a fase de desenvolvimento do software em ciclos completos, passando por todas as fases existentes no modelo clássico, levantamento e análise de requisitos, projeto, implementação e testes.

- O modelo trabalha com 5 (cinco) fases distintas: comunicação, planejamento, modelagem, construção e implantação.



Prof. Anderson Augusto Bosing

Modelo Incremental



1. **Comunicação:** trata das informações do negócio, como são geradas, processadas e quem utilizará. Define os requisitos.
2. **Planejamento:** define os objetos, suas características e como se relacionarão entre si.
3. **Modelagem:** desenha o processo, visando atender à função do negócio, e define os procedimentos necessários para a manipulação dos objetos de dados definidos na etapa anterior.
4. **Construção:** ocorre a geração da aplicação, normalmente por meio de reutilização de componentes que serão integrados posteriormente.
5. **Implantação:** acontece alguns pequenos testes e a entrega de parte do produto.

Modelo espiral

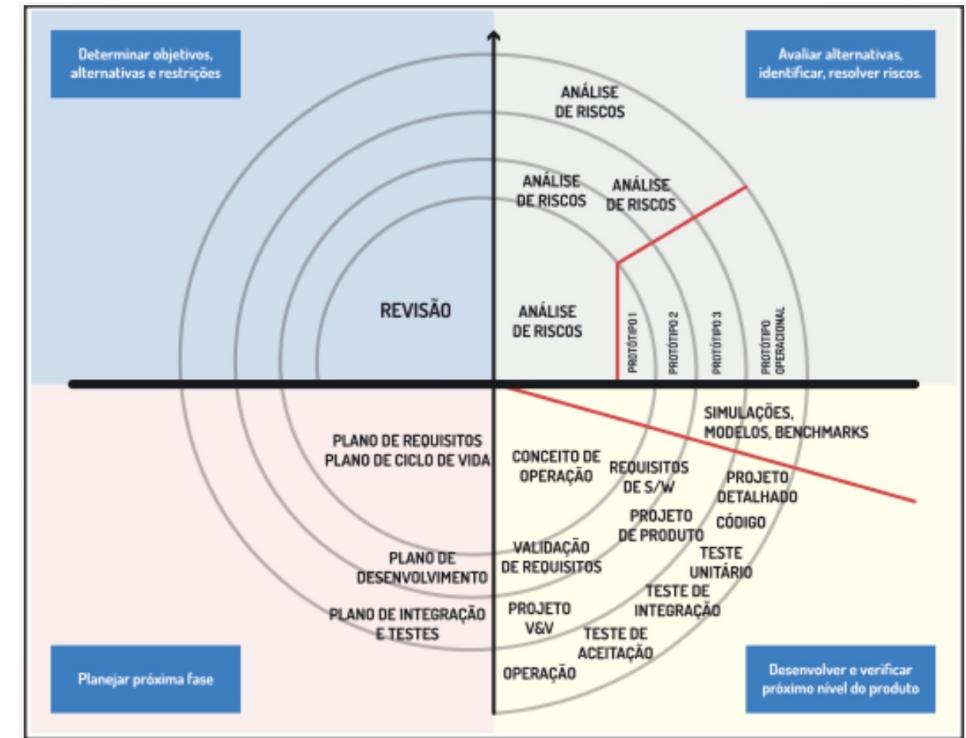
- Pressman (2011) define que esse é um modelo de processo de software evolucionário que acopla a natureza **iterativa da prototipação** com os **aspectos sistemáticos e controlados do modelo cascata**. Fornece potencial para o rápido desenvolvimento de versões cada vez mais completas de software.
- Cada loop na espiral representa uma fase do processo de software.
- As atividades acontecem em forma de espiral, no sentido horário, começando do centro, no ponto indicado como início. O software será construído em versões evolutivas, sendo que, em cada iteração da espiral, temos a entrega de um protótipo ou de uma versão mais completa do sistema.



Prof. Anderson Augusto Bosing

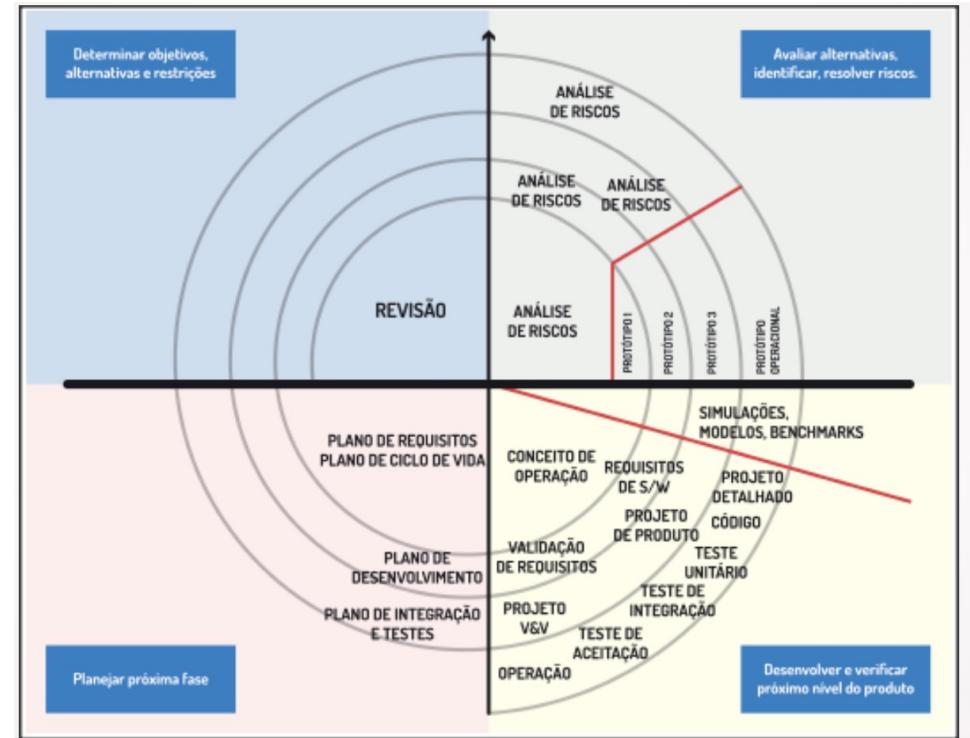
Modelo espiral

- 1. Planejamento e Requisitos** - nessa fase, é feito o levantamento de requisitos junto ao cliente, estabelecendo as restrições e as diretrizes do software, bem como os riscos. É criado um plano de gerenciamento, com cronogramas e estimativas de custos.
- 2. Análise de Riscos** - nesse ponto, é realizado uma análise para cada um dos riscos levantados e estudado maneiras de reduzi-los.
- 3. Desenvolvimento e Validação** – o protótipo é desenvolvido e validado.
- 4. Entrega e Avaliação de Continuidade** - nessa fase, ocorre uma entrega para o cliente e uma avaliação, para decidir se o projeto continuará para o próximo loop da espiral. Caso seja decidida a continuidade, o custo, o cronograma e o planejamento da quantidade de iterações planejadas anteriormente serão refinados de acordo com os feedbacks recebidos do usuário.



Modelo espiral

- Sommerville (2014) “A importante distinção entre o modelo espiral e outros modelos de processo de software é a explícita consideração dos **riscos** no modelo em espiral”.
- Segundo o **PMBOK**: riscos são efeitos negativos nos objetivos do projeto, como escopo, qualidade, custo e prazo.
- Exemplo:
Durante a primeira espiral, é utilizado prototipação, para tirar dúvidas e realizar alinhamento com o cliente, visando à redução dos riscos, e, nos próximos loops da espiral, é utilizado o modelo cascata.



Montagem de Componente

- Faz reuso de software visando o ganho em agilidade e confiabilidade, utilizando tecnologias de orientação a objetos.
- Sendo que agilidade desse modelo se deve ao fato do **reuso dos componentes**, obtido por meio de adaptações e de combinações.
- *Pressman (2011)* define que “a engenharia de software baseada em componentes identifica, constrói, cataloga e dissemina um conjunto de componentes de software em determinado domínio de aplicação. Esses componentes são então qualificados, adaptados e integrados para uso em um novo sistema..”



Prof. Anderson Augusto Bosing

Montagem de Componente

➤ Exemplo:

Um software para controle de loja de materiais de construção.
Para ele, é criado o componente de login e o componente de
cadastro de usuários.

➤ Um software para controle de uma loja de moveis.

Pode ser reutilizado os dois componentes criados para o software de controle da loja de materiais de
construção para compor o software da loja de moveis, visto que já estão em uso, funcionando e testados.



Prof. Anderson Augusto Bosing

Técnicas de 4a Geração

- As técnicas de quarta geração englobam um **conjunto de ferramentas de softwares** que visa permitir ao desenvolvedor especificar as características do software em alto nível.
- A técnica de quarta geração tem o objetivo de que os requisitos descritos pelo cliente sejam capazes de ser introduzidos em uma ferramenta que automaticamente geraria a estratégia do projeto, as codificações e realizaria os testes.



Prof. Anderson Augusto Bosing

Engenharia de Software

**Aula: Metodologias Ágeis
de Desenvolvimento de
software.**



Prof. Anderson Augusto Bosing

Por que agil ? O que é agilidade?



O que é agilidade ?

Em fevereiro de 2001, dezessete representantes de diversas práticas e metodologias de desenvolvimento se reuniram em uma estação de esqui, em Utah nos EUA para discutir métodos mais leves de desenvolvimento do que o tradicional desenvolvimento orientado a documentos.

Auto denominados de “The Agile Alliance” criaram o Manifesto for Agile Software Development ou simplesmente Manifesto Ágil para definir a abordagem hoje conhecida como desenvolvimento ágil.

“Estamos descobrindo maneiras melhores de desenvolver software fazendo nós mesmos e ajudando outros a fazê- lo.



Prof. Anderson Augusto Bosing

O que é agilidade ?

- Agilidade:
Rapidez, desembaraço;
Qualidade de quem é veloz;
- Capacidade de responder rapidamente a mudanças:
Mudanças de tecnologias, de equipe, de requisitos...
- Entregar valor ao cliente quando se lida com
imprevisibilidade e dinamismo dos projetos;



Prof. Anderson Augusto Bosing

Metodologias ágeis

- “Linha de pensamento” revolucionária:
 - Precisamos parar de tentar evitar mudanças;
- Metodologias ágeis:
 - “Filosofia” onde muitas “metodologias” se encaixam;
 - Definem um conjunto de atitudes e não um processo prescritivo.



Prof. Anderson Augusto Bosing

Metodologias ágeis

Reação às metodologias tradicionais;

- “Manifesto ágil” (2001):<https://agilemanifesto.org/>
 - Movimento iniciado por programadores experientes e consultores em desenvolvimento de software;
 - Questiona e se opõe a uma série de mitos/práticas adotadas em abordagens tradicionais de Engenharia de Software e Gerência de Projetos.



Prof. Anderson Augusto Bosing

Metodologias ágeis

Princípios do Manifesto Ágil:

- Indivíduos e interações são mais importantes que processos e ferramentas;
- Software funcionando é mais importante que documentação completa (abrangente);
- Colaboração do cliente é mais importante que negociação de contratos;
- Adaptação às mudanças é mais importante que seguir um plano;



Prof. Anderson Augusto Bosing

Metodologias ágeis

Segundo Pressman:

- A engenharia de software ágil combina uma filosofia e um conjunto de diretrizes de desenvolvimento;
- A filosofia encoraja a satisfação do cliente e a entrega incremental de software logo no início;
- Equipes de projetos pequenas e altamente motivadas;
- Métodos informais;
- Produtos de trabalho de engenharia de software mínimos e simplicidade global de desenvolvimento;
- As diretrizes de desenvolvimento enfatizam a entrega em contraposição à análise e ao projeto (apesar destas atividades não serem desencorajadas);
- Comunicação ativa e contínua entre desenvolvedores e clientes.



Prof. Anderson Augusto Bosing

Princípios da agilidade

1. A mais alta prioridade é a satisfação do cliente, por meio da liberação mais rápida e contínua de software de valor;
2. Receba bem as mudanças de requisitos, mesmo em estágios tardios do desenvolvimento. Processos ágeis devem admitir mudanças que trazem vantagens competitivas para o cliente;
3. Libere software freqüentemente (em intervalos de 2 semanas até meses), dando preferência para uma escala de tempo mais curta;
4. Mantenha pessoas ligadas ao negócio (clientes) e desenvolvedores trabalhando juntos a maior parte do tempo do projeto;



Prof. Anderson Augusto Bosing

Princípios da agilidade

5. Construa projetos com indivíduos motivados, dê a eles o ambiente e suporte que precisam e confie neles para ter o trabalho realizado;
6. O método mais eficiente e efetivo para repassar informação entre uma equipe de desenvolvimento é pela comunicação face-a-face;
7. Software funcionando é a principal medida de progresso de um projeto de software;
8. Processos ágeis promovem desenvolvimento sustentado. Assim, patrocinadores, desenvolvedores e usuários devem ser capazes de manter conversação pacífica indefinidamente;



Prof. Anderson Augusto Bosing

Princípios da agilidade

9. A atenção contínua para a excelência técnica e um bom projeto (design) aprimoram a agilidade;
10. Simplicidade - a arte de maximizar a quantidade de trabalho não feito – é essencial, devendo ser assumida em todos os aspectos do projeto;
11. As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas;
12. Em intervalos regulares, as equipes devem refletir sobre como se tornarem mais efetivas, e então refinarem e ajustarem seu comportamento de acordo.



Prof. Anderson Augusto Bosing

Características gerais

- Procuram minimizar riscos desenvolvendo software em pequenos espaços de tempo (iterações);
- Cada iteração é como um pequeno projeto:
 - Planejamento, requisitos, projeto, codificação, testes...
 - Propõem o desenvolvimento de software de forma mais rápida, com um grande número de ciclos, mas com qualidade;
- Objetivo de cada iteração:
 - Produzir componentes de software (incrementos);
 - Arquitetura vai sendo desenhada a partir da refatoração dos componentes;



Prof. Anderson Augusto Bosing

Características gerais

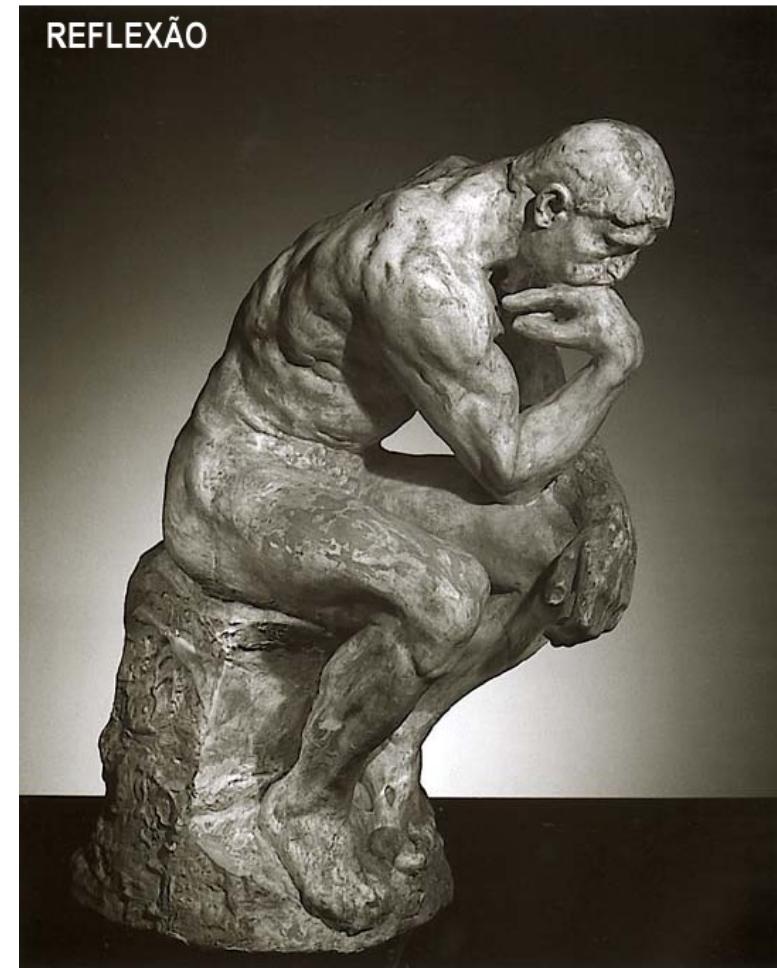
- Um catalizador efetivo para o feedback do cliente é um protótipo executável ou parte de um sistema operacional (“incrementos de software”);
- “Incrementos de software” devem ser entregues em curtos períodos de tempo de modo que a adaptação acerte o passo com as modificações (imprevisibilidade);
- Essa abordagem iterativa habilita o cliente a avaliar o incremento de software regularmente, fornecer feedback necessário à equipe de software e influenciar as adaptações do processo feitas para acomodar o feedback.



Características gerais

- Encorajamento de atitudes reflexivas e contínuo aprendizado;
- Lições aprendidas: as lições aprendidas de qualquer atividade de solução de problema (inclusive aquelas que resolvem o problema errado) podem ser benéficas para a equipe mais adiante no projeto ou em outros projetos;

REFLEXÃO



Características gerais

- Fatores Humanos: Segundo Cockburn e Highsmith, “o desenvolvimento ágil enfoca os talentos e habilidades dos indivíduos moldando o processo a pessoas e equipe específicas”;
- Auto-organização no contexto de desenvolvimento ágil:
 1. A equipe ágil organiza-se para o trabalho a ser feito;
 2. A equipe organiza o processo para melhor acomodar seu ambiente local;
 3. A equipe organiza o cronograma de trabalho para conseguir melhor entrega do incremento de software;
- Uma equipe auto-organizada está no controle do trabalho que realiza. A equipe estabelece os seus próprios compromissos e define os planos para cumpri-los;



Prof. Anderson Augusto Bosing

Características gerais

Segundo Ken Schwaber:

“A equipe seleciona quanto trabalho acredita que pode realizar dentro da iteração e a equipe se compromete com o trabalho. Nada desmotiva tanto uma equipe quanto alguém de fora assumir compromissos por ela. Nada motiva tanto uma equipe quanto a aceitação da responsabilidade de cumprir seus compromissos que ela própria estabeleceu.”



Prof. Anderson Augusto Bosing

Conclusões

Pró-agilidade x Pró-engenharia tradicional segundo HighSmith:

- “Os metodologistas tradicionais são um punhado de bitolados que preferem produzir documentação perfeita a um sistema funcionando que satisfaça às necessidades do negócios.”;
- “Os metodologistas levianos, quer dizer, ‘ágiles’, são um punhado de gloriosos hackers que terão uma grande surpresa quando tiverem de ampliar seus brinquedos para chegar a um software que abranja toda a empresa.”;



Prof. Anderson Augusto Bosing

Conclusões

- Não fique limitado a uma “arma” ou técnica em particular;
- Metodologias diferentes são necessárias para diferentes tipos de projetos
 - Fatores a serem considerados:
 - Número de Pessoas envolvidas no Projeto;
 - Criticidade do Sistema;
 - Prioridades do Projeto;
- Construa a sua “caixa de ferramentas”;



Prof. Anderson Augusto Bosing

Conclusões

- Em cada modelo ágil (XP, SCRUM, Crystal, FDD) há um conjunto de “idéias” (“tarefas de trabalho”) que representam um afastamento significativo da engenharia de software convencional;
- Segundo Pressman, muitos conceitos ágeis são simples adaptações de bons conceitos de engenharia de software;
- Conclusão segundo Pressman: “há muito a ser ganho considerando o melhor de ambas as escolas, e quase nada a ser ganho denegrindo qualquer uma dessas abordagens.”

