Engenharia de Software II

Aula 5

http://www.ic.uff.br/~bianca/engsoft2/

Dúvidas da aula passada

- RUP (Rational Unified Process) é uma ferramenta ou um processo? Resposta: os dois.
  - O RUP é uma instância específica e detalhada de um processo genérico descrito por Jacobson, Booch and Rumbaugh no livro "*The Unified Software Developmente Process*".
  - O produto RUP é uma ferramenta desenvolvida e mantida pela Rational (agora parte da IBM) de apoio ao RUP.
- Na fase de concepção do RUP qual a diferença entre "Documento de visão" e "Caso de negócio inicial"?
  - Documento de visão: dá uma visão geral dos requerimentos e das características mais importantes do produto.
  - Caso de negócio inicial: fornece as informações necessárias para determinar se vale a pena investir no projeto, como os critérios de sucesso, projeção de lucros e previsões financeiras.
- Mais informações sobre o RUP:
  - http://www.wthreex.com/rup/
  - ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/rup_bestpractices.pdf

Desenvolvimento Ágil

- 2001: Manifesto para o Desenvolvimento Ágil de Software (http://www.agilemanifesto.org).
  - 17 figuras proeminentes na engenharia de software (a Aliança Ágil) se reuniram e declararam importante a valorização de:
    - **Indivíduos e interações** em vez de processos e ferramentas.
    - **Softwares funcionando** em vez de documentação abrangente.
    - **Colaboração do cliente** em vez de negociação de contratos.
    - **Resposta a modificações** em vez de seguir um plano.

O que é o desenvolvimento ágil?

- É uma filosofia e um conjunto de diretrizes que encorajam:
  - A entrega incremental do software logo de início.
  - Equipes de projeto pequenas e motivadas.
  - Métodos informais de comunicação ao invés de documentos escritos.
  - Enfatizar a entrega em contraposição à análise e ao projeto.
  - Adotar o cliente como parte da equipe.

Quando deve ser usado o desenvolvimento ágil?

- O desenvolvimento ágil é particularmente indicado em situações onde os requisitos são imprevísiveis ou mudam rapidamente.
- Ele funciona melhor para equipes pequenas (até 10 desenvolvedores) trabalhando no mesmo local.

Doze princípios da agilidade

1. Ter como maior prioridade satisfazer o cliente por meio da entrega de software.
2. Modificações contínuas são bem-vindas e levam à competitividade para o cliente.
3. Entrega de software funcionando em períodos de duas semanas a dois meses.
4. O pessoal de negócio e os desenvolvedores devem trabalhar juntos diariamente.
5. Construção de projetos em torno de indivíduos motivados e talentosos.
6. Usar métodos informais de comunicação como conversar pessoalmente.
7. Software funcionando é a principal medida de progresso.
8. Promover o desenvolvimento sustentável, mantendo um ritmo cde produção.
9. Atenção contínua à excelência técnica e ao bom projeto.
10. Simplicidade é essencial.
11. As melhores arquiteturas, requisitos e projetos surgem de equipes auto-organizadas.
12. A equipe deve, em intervalos regulares, refletir sobre como se tornar mais efetiva.

O que é um Processo Ágil?

- É um processo que atende a três suposições-chave sobre a maioria dos projetos de software:
  1. É difícil prever antecipadamente quais requisitos de software e prioridades do cliente vão persistir e quais serão modificadas.
  2. É difícil prever o quanto de projeto é necessário antes que a construção seja usada para comprovar o projeto.
  3. Análise, projeto, construção e testes não podem ser tão bem planejados como gostaríamos.
- Para atender a essas suposições, o processo ágil deve ser adaptável incrementalmente.

Fatores humanos

- Características-chave de uma equipe ágil:
  1. Competência
  2. Foco comum
  3. Colaboração
  4. Capacidade de tomada de decisão
  5. Habilidade de resolver problemas vagos
  6. Respeito e confiança mútua
  7. Auto-organização

Modelos ágeis de processo

- Extreme Programming (XP)
- Desenvolvimento Adaptativo de Software (DAS)
- Método de Desenvolvimento Dinâmico de Sistemas (DSDM)
- Scrum
- Crystal
- Desenvolvimento Guiado por Características (FDD)
- Modelagem Ágil (AM)

Extreme Programming (XP)

- Trabalho pioneiro sobre o assunto escrito em 1999 por Kent Beck.
- Usa uma abordagem orientada a objetos como seu paradigma de desenvolvimento.
- Inclui um conjunto de regras e práticas que ocorrem no contexto de quatro atividades de arcabouço:
  - Planejamento
  - Projeto
  - Codificação
  - Teste

XP - Planejamento

- Criação de um conjunto de histórias de usuário.
  - Parecidas com casos de uso, mas com bem menos detalhes.
  - Cada história é escrita em poucas linhas pelos clientes, que lhe atribuem um valor, e deve poder ser implementadas em menos de três semanas.
  - Exemplo: “Quando a aplicação começa, o último documento em que o cliente estava trabalhando deve ser aberto automaticamente.”
- A equipe XP e os clientes trabalham juntos para definir um plano que determina as histórias que serão desenvolvidas primeiro levando em consideração valores e riscos.
- Depois que o primeiro incremento é entregue, a equipe XP calcula a velocidade do projeto = número de histórias implementadas.
- Ao longo do tempo, o cliente pode adicionar histórias, mudar o valor de uma história, subdividir histórias ou eliminá-las,

XP - Projeto

- Segue o princípio KIS (*keep it simple*).
  - Se restringe a implementar as histórias de usuário.
- Usa cartões CRC (Class-Responsability-Colaborator) para identificar e organizar as classes que são relevantes para o incremento atual.
- Se um problema de projeto difícil é encontrado, o XP recomenda a criação de uma solução de ponta para diminuir o risco.
  - Solução de ponta = um protótipo operacional daquela parte do projeto, que depois será descartado.
- O XP encoraja a refabricação = modificar o sistema de tal modo que o comportamento externo não seja alterado, mas aperfeiçoe a estrutura interna.
  - A refabricação ocorre durante a codificação, mas altera o projeto.

XP - Codificação

- Antes de partir para o código, a equipe deve desenvolver testes unitários para verificar a funcionalidade que será desenvolvida.
- A programação é feita em pares.
  - Isso fornece um mecanismo de solução de problemas e de garantia de qualidade em tempo real.
  - Uma pessoa pensa nos detalhes do código enquanto a outra garante as normas de codificação e que o código gerado vai se encaixar no resto do sistema.
- O código gerado vai sendo integrado imediatamente ao trabalho de outros, o que ajuda a evitar problemas de compatibiilidade e interface.

XP - Teste

- Os testes unitários são criados de tal forma que eles possam ser automatizados, e aplicados diariamente.
- O XP usa uma estratégia de teste de regressão: testes são aplicados de novo mesmo que anteriormente eles não tenham apresentado problema.
  - Isso permite a refabricação.
- Testes de aceitação são especificados pelo cliente (derivados das histórias do usuário) e focalizam nas características e funcionalidades do sistema global.
  - Devem ser automatizados e aplicados frequentemente.
  - O sistema só é considerado aceitável quando todos passar em todos os testes de aceitação.