**Anotações 08/03/2024**

**Engenharia Tradicional**

* Civil, mecânica, elétrica, aviação, automobilística etc
* Projeto com duas características:
  + Planejamento detalhado
  + Sequencial
  + Uma tarefa depende da outra
* Isto é (Watterfal)
  + Há milhares de anos

Software é diferente

* Engenharia de software é != Engenharia Tradicional
* Software != carro, ponte, casa, avião)
* Software != produtos físicos
* Software é abstrato

**Dificuldade 1: Requisitos**

* Clientes não sabem o que querem
  + Funcionalidades são infinitas
  + Mundo mudal
* Não da para ficar gastando tempo levantando requisitos, depois projetar e então implementar
* Isso leva a um software obsoleto

**Dificuldade 2: Documentações detalhadas**

* Verbosas e pouco uteis
* Na prática, desconsideradas durante implementação
* Requisitos mudam com o tempo

Manifesto Ágil trouxe um sistema onde o sistema é desenvolvido e entregue gradativamente (sistema interativo).

**Desenvolvimento interativo**

* Suponha um sistema imenso, complexo etc
* Qual o menor “incremento do sistema” eu consigo implementar em 15 dias e validar com usuários?
* Validar é muito importante!!
* **CLIENTE NÃO SABE O QUE QUER!**

**Outros pontos importantes**

* Menor ênfase em documentação
* Menor ênfase em big upfront design
* Envolvimento constante do cliente
* Novas práticas de programação
  + Testes, refactoring, integração continua etc

**Métodos Ágeis**

* Dão mais consistência às ideias ágeis
  + Definem um processo, mesmo que çeve
  + Workflow, eventos, papeis etc..

**Exemplos: Extreme Programming (XP), Scrum, Kanban**

**Scrum**

* Muito utilizado nas empresas locais.
* Princípio proposto em 1995

**Sprints**

* Duração: até 1 mês, normalmente 15 dias
* A cada Sprint, entrega-se uma parte do sistema, diferente do Waterfall que entrega somente no fim do processo
* Implementa-se algumas histórias dos usuários. Funcionalidades acordadas entre cliente e empresa desenvolvedora do software
* Histórias: funcionalidades do sistema
* Exemplo: fórum de perguntas e respostas
* Geralmente quem escreve essas histórias são os POs. Um especialista no domínio do sistema
* Cliente <-> PO <-> desenvolvedores

**Funções de um PO**

* Escrever histórias
* Explicar histórias
* Definir testes de aceitação de histórias
* Priorizar histórias, dando sequência a elas

**Backlog do Produto**

* Lista de historias do usuário
* (e outros itens do trabalho)
* Duas características
  + Priorizada
  + Dinâmica

**Iteração = Sprint**

**Importante**

* **Em um time Scrum, todos tem o mesmo nível hierárquico**
* **PO não é o chefe dos Devs**
* **Devs tem autonomia para dizer que não vão conseguir implementar tudo que o PO quer em um único Sprint**

**Times Scrum**

* Pequenos
* 5 a 11 membros, sendo 1 PO e 1 Scrum Master
* Multidisciplinar: devs, designers, cientistas de dados etc
* Times ágeis são hoje chamados de squads com frequência

**Scrum Master**

* Especialista em Scrum
* Removedor de impedimentos não técnicos
  + Exemplo: desenvolvedores não tem máquinas boas
* Pode também coletar métricas de processo
* Não é o chefe do time, mas um líder servidor
* Pode pertencer a mais de um time

**Reuniões diárias**

* 15 minutos de duração (rápida)
  + O que foi feito no dia anterior
  + O que fazer no dia
  + Se há algum problema atualmente impedindo o desenvolvimento
* Objetivos
  + Melhorar comunicação
  + Antecipar problemas

**Revisão do Sprint**

* Time apresenta o resultado do spring para o PO
* O Sprint pode ser aprovado ou reprovado, onde, nos piores casos a historia volta ao backlog

**Retrospectiva**

* Último evento do sprint
* Time se reúne para decidir o que melhorar
* Modelo mental, melhorias constantes

**Criterios para conclusão de historias**

* Critérios internos para considerar historias prontas
* Também chamados de DoD
* Exemplos:
  + Testes de unidade com cobertura >= 75%
  + Revisão de código por outro membro da equipe
  + Atualizar documentação
  + Teste de performance

**Story Points**

* Usados para estimar o tamanho da historia
* Ajudar a definir o que vai caber na sprint
* Uso não é obrigatório em scrum

**Kanban**

* Origem na deca de 50 no japao
* Sistema de produção da Toyota
* Manufatura lean, produção just/time
* Trabalhar na produção de um projeto através de quadros
* É mais simples
* Não existem sprints
* Ideia central é que as pessoas “puxam” as tarefas para trabalhar ao concluir uma tarefa ela passa para outra etapa no quadro

**Limites WIP**

* Número máximo de tarefas em um passo
* Contando: tarefas em andamento e concluídas

**Objetivos dos limites WIP**

* Criar um fluxotrabalho sustentável
  + Evitar que o time fique sobrecarregado de trabalho
  + WIP = “ACORDO” entre o time e a organização
* Evitar que o trabalho fique concentrado em um passo

**“pare de começar e comece a terminar”**

**- Kanban é mais simples que o Scrum**

**- Kanban é mais adequado para times maduros**

**- Talvez, começar com Scrum e depois ir para Kanban**

**- Kanban é um método evolucionario**

**Transição de Waterfall para Ágil**

* Antes de disseminação dos princípios ágeis, alguns métodos iterativos ou evolucionários foram propostos
* Transição Waterfall ~1970 e ágil ~2000 foi gradativa
* Exemplos
  + Espiral 1986 – Proposto por Barry Boehm – iterações de 6 a 24 meses
  + Rational Unified Process (RUP) 2003 – Proposto pela Rational, depois comprada pela IBM – Usando diragramas UML e ferramentas CASE