Programação Pragmática

Carla Maria Pinheiro

Agenda

- O que é Programação Pragmática?
- Programador Pragmático
- Antes da Implementação...
- *Tracer Code*
- Boas Práticas de Programação
- Testes
- Bibliografia

O que é Programação Pragmática?

- Criada por Andrew Hunt e David Thomas
- Boas práticas de programação
- Voltada para o programador e para a equipe de programação
- Não tem um processo definido em fases

Programador Pragmático

- Curioso por técnicas e tecnologias
  - Investimento no aprendizado (aulas, cursos)
- Inquisitivo
- Crítico
- Realista
- Assume responsabilidades
  - Assume seus erros
  - Provê opções para resolvê-los

Antes da Implementação...

- Levantamento de Requisitos
  - Trabalhe com o usuário para pensar como ele
  - Dicionário de dados em reuniões com usuários
  - Especificação de mini-linguagem
  - *Use Cases*
    - Documentação formal ou informal?
    - Poucos detalhes
- *Template* para caso de uso - Cockburn

Template

1. CHARACTERISTIC INFORMATION
   - Goal in context
   - Scope
   - Level
   - Preconditions
   - Success end condition
   - Failed end condition
   - Primary actor
   - Trigger

Template

2. MAIN SUCCESS SCENARIO
3. EXTENSIONS
4.VARIATIONS
5. SCHEDULE
6. OPEN ISSUES

Template

8.RELATED INFORMATION

- Priority
- Performance target
- Frequency
- Superordinate use case
- Subordinate use cases
- Channel to primary actor
- Secondary actor
- Channel to secondary actor

Tracer Code

- Usado, principalmente, em sistemas novos no mercado
- Construção de parte do esqueleto do sistema
- Adição de funcionalidades

Tracer Code

- Visão do sistema próxima da realidade
- Não é um protótipo
  - O *tracer code* será preenchido e corrigido com o desenrolar do sistema
  - Protótipos, em geral, são descartáveis

Tracer Code

- Usuários vêem algo funcionando (demonstrações)
- Desenvolvedores constroem estrutura
- Plataforma de integração
- Melhor sentimento de progresso

Boas Práticas de Programação

- Janela Quebrada
  - Projetos ruins, decisões erradas, código “pobre”
  - *Bugs* encontrados -> *Bugs* corrigidos!
  - Maior facilidade de correção no início do desenvolvimento

Boas Práticas de Programação

- *DRY Principle – Don't Repeat Yourself*
  - Toda parte do conhecimento deve ter uma simples e não-ambígua representação dentro do sistema!

- Evite duplicação de Informação
  - Documentação em código
    - Códigos ruins precisam de comentários!?
    - Desatualização dos comentários

Boas Práticas de Programação

- Ortogonalidade
  - Componentes com funcionalidades bem definidas
  - Mudanças localizadas
  - Facilita o reuso
  - Reduz risco do desenvolvimento

Boas Práticas de Programação

- Scripts
  - Tarefas Repetitivas
- Editores de Texto
  - Uso de editores como IDE!!!
- Controle de Código Fonte

Boas Práticas de Programação

- Geração de Código
  - Geradores Passivos
  - Geradores Ativos

Boas Práticas de Programação

- *Refactoring*
  - *Refactoring* automático
    - Ferramentas
- *Wizards*
  - Não use se você não entende o código produzido!!

Testes

- Unitários
  - Componentes
- Integração
  - Subsistemas
- Validação e Verificação
  - Dados
  - Regressão
- Teste os testes!!!
  - Cause *bugs*

Bibliografia

- Andrew Hunt, David Thomas, The Pragmatic Programmer, Addison-Wesley, 2000
- www.pragmaticprogrammer.com
- http://alistair.cockburn.us/

Programação Pragmática