Envelhecimento de Software

Rodrigo Rocha Gomes e Souza

9 de julho de 2008

envelhecimento, decaimento, erosão

VISÃO GERAL

Software aging

- Parnas, 1994
- Metáfora médica/biológica
  - Velhice inevitável (a não ser que morra)
  - Recursos oferecidos vs. Expectativas dos usuários
- Sintomas
  - Difícil acompanhar o mercado
  - Deterioração da performance
  - Muitos bugs

- Causas
  - Ausência de movimento
  - Presença de movimento
- Problema: compreensão
- Programador
  - Não gosta de documentar
  - Gosta de resultados a curto prazo (ver o programa rodando)

- Solução
  - documentação e reviews
  - refactoring
  - Projetar para mudanças
    (Como prever as mudanças? É possível?)
  - Diluir a preocupação com o 1º release
- Discussão
  - Projetar para mudanças (flexibilidade) vs.
  - Keep it simple, Sam. (compreensão)
  - Adiar modularização (economia)

VISÃO FUNCIONAL
VISÃO ESTRUTURAL

Functional Paleontology

- Antón e Potts, 2001
- Como prever as mudanças?
  (enfoque em função, requisitos)
- Normal science vs. Paradigm shifts (Kuhn)
- Hierarquia de funcionalidades
  - Benefícios, ônus
  - Rebound effect
  - Benefícios ativos e reativos (+ acessibilidade, - privacidade)
  - Agente precede outros
  - Nível objeto precede nível meta
- (The Black Swan)

VISÃO FUNCIONAL
VISÃO ESTRUTURAL

Design Principles and Design Patterns

- Robert C. Martin, 2000
- Sintomas do apodrecimento do design
  - **Rigidez** – é difícil mudar
  - **Fragilidade** – se mudar, quebra
  - **Imobilidade** – não dá pra reusar
  - **Viscosidade**
    - Do design: é mais fácil fazer gambiarra
    - Do ambiente: por exemplo, desenvolvedores privilegiam mudanças que minimizem o tempo de compilação
- Designs resilientes
  - Diminuir a dependência. “Firewalls de dependência”
  - Princípios de design: OCP, LSP, DIP, ISP. Princípios de arquitetura...

Does Code Decay?

- Eick et al., 2001
- Enfoque estrutural
- Code decay provocado por mudanças
- Code decay: mudanças ficam difíceis
  - Custo, intervalo, qualidade
- Fatores de risco
  - tamanho de módulo, idade cronológica, complexidade inerente, mudança na organização, reuso, muitos requisitos, desenvolvedores inexperientes

- Mudanças
  - DELTA, ADD, DATE, INT, DEV
- CDI (code decay indices)
  - CHNG, FREQ, FILES, NCSL, AGE
  - Preditivas: FP (fault), EFF (effort)

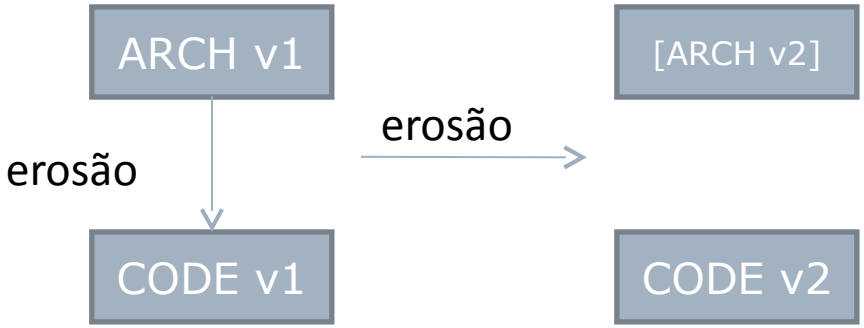
- Resultados
  - Rigidez crescente no tempo
  - Modularidade (localidade) decrescente no tempo
- Discussão
  - Complexidade do código (R) quantidade de faltas

OUTROS

Design Erosion: Problems & Causes

- Pesquisa “de varejo”
- Como evitar erosão: design patterns, refactoring, separation of concerns
- Erosão: você pode adiar, mas não evitar.
- Estratégias
  - Estratégia ótima (???)
  - Estratégia mínima (???)
- Opinião: Idéia boa, realização ruim
  - Teoria da informação
  - Net Option Value (Design Rules: The Power of Modularity. Baldwin e Clark)

Lightweight Prevention of Architectural Erosion



- Descrição arquitetural: DAG
- Como garantir que as pessoas entendem e implementam as decisões arquiteturais?
  - Monitorar as mudanças e comparar com a arquitetura
- Opinião: idéia boa, exposição superficial