CÓDIGO BELO VS. LEGADO E QUALIDADE DE SOFTWARE

ENGENHARIA DE SISTEMAS DE INFORMAÇÃO

Daniel Cordeiro

22 de agosto de 2017

Escola de Artes, Ciências e Humanidades | EACH | USP

PERGUNTA

Em geral, qual afirmação sobre a relação entre os custos de consertar bugs no software e de melhorar o software é a mais precisa?

- $(Consertar um bug) $\geq \approx 2\times$ $(Melhorar)
- $(Consertar um bug) $\approx$ $(Melhorar)
- $(Melhorar) $\approx 2\times$ $(Consertar um bug)
- $(Melhorar) $\approx$ 2–4$\times$ $(Consertar um bug)

Manutenção vs. desenvolvimento: gasto com TI pelo governo dos EUA em 2010–2017

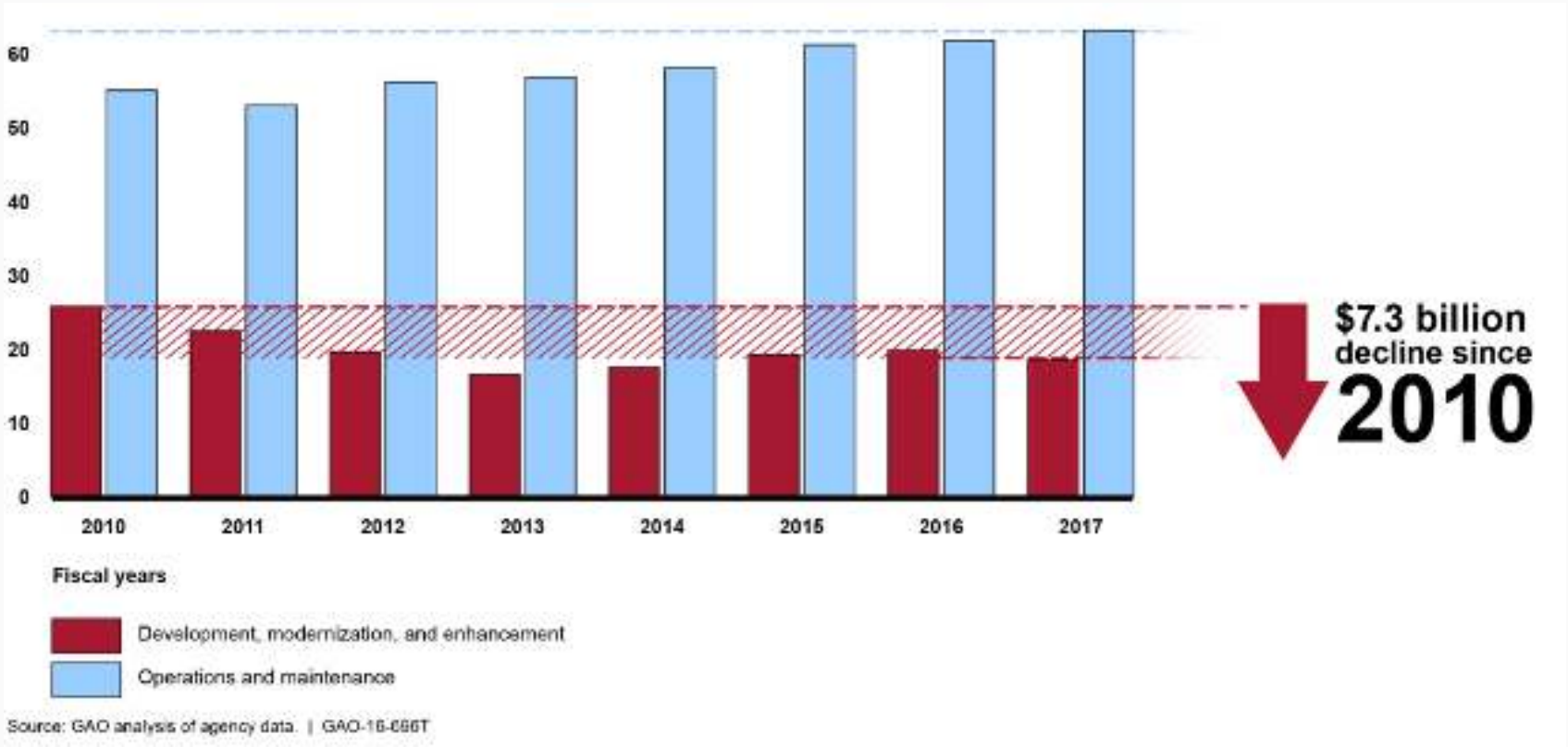


Figura 1: Fonte: *Government Accountability Office (GAO) report GAO-16-696T, Federal Agencies Need to Address Aging Legacy Systems, Testimony Before the Committee on Oversight and Government Reform, House of Representatives.* Publicado em 26 de maio de 2016.

Código legado vs. código belo

- Código legado: software antigo que continua a satisfazer as necessidades do cliente, mas que é difícil de evoluir devido a um projeto deselegante ou tecnologia antiquada
  - 60% do custo de manutenção do software causado pela adição de novas funcionalidades a código legado
  - 17% do custo é usado na correção de bugs
- **Problema vital** mas ignorado na maioria dos cursos de ES
- Diferente de código belo: satisfaz o cliente e é fácil de evoluir

QUALIDADE DE SOFTWARE

- Qualidade do produto (em geral): “adequação para o uso”
  - Valor de negócio para o cliente && para o fabricante
  - Garantia de qualidade: processos/padrões
- Qualidade do software:
  1. Satisfaz as necessidades do cliente: facilidade de usar, dá as respostas corretas, não para de funcionar (*crash*), etc.
  2. É fácil de ser depurado e melhorado pelos desenvolvedores
- Software QA: garante a qualidade e melhora os processos na organização do software

GARANTIAS

- Verificação: você construiu a coisa da maneira *certa*?
  - você seguiu a especificação?
- Validação: você construiu a coisa *certa*?
  - era isso o que o cliente queria?
  - a especificação estava certa?
- Hardware: geralmente foca a verificação
- Software: geralmente foca a validação
- Testes garantem a Qualidade do Software

TESTAR EXAUSTIVAMENTE É IMPRATICÁVEL

- Dividir para conquistar: realize testes diferentes em cada fase do desenvolvimento do software
  - o nível superior não refaz os testes do nível inferior
- *Cobertura*: várias medidas que indicam o quanto do código foi exercitado pelo conjunto de testes

Tipos de testes

Teste de sistema ou aceitação verifica se o programa integrado cumpre a especificação

Teste de integração verifica se as interfaces entre diferentes unidades tem as mesmas hipóteses; se elas se comunicam corretamente

Módulo ou teste funcional verifica diversas unidades individuais

Teste de unidade verifica se um único método faz aquilo que é esperado

PERGUNTA

Qual afirmação não é verdadeira sobre testes?

- Testes que não são mais tão úteis devem ser descartados
- Mesmo que seja difícil conseguir, uma cobertura de 100% garante que o código está correto
- Testes de alto nível normalmente delegam testes mais detalhados para os níveis mais baixos
- Testes de unidade trabalham dentro de uma única classe e testes funcionais trabalham com várias classes

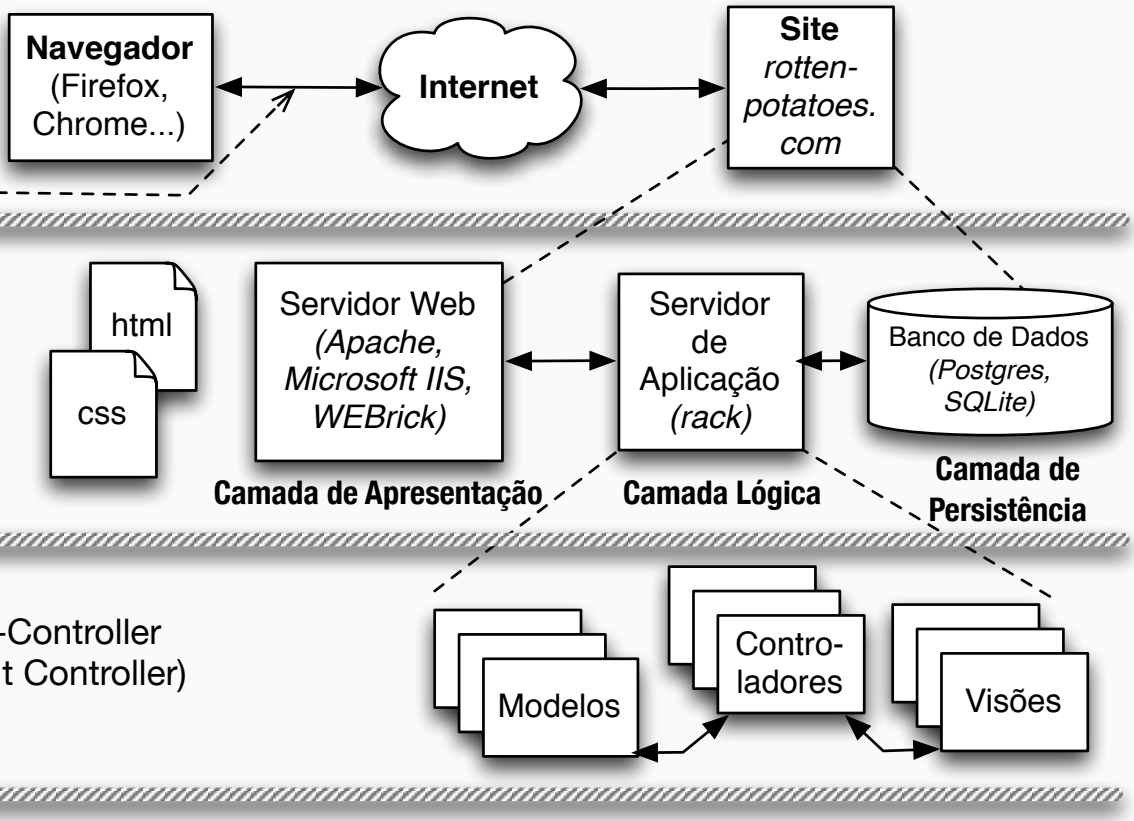
A ARQUITETURA DE APLICAÇÕES SAAS

§2.1 100.000 pés
• Cliente-servidor (vs. P2P)

§2.2 50.000 pés
• HTTP e URIs

§2.3 10.000 pés
• XHTML e CSS

§2.4 5.000 pés
• Arquitetura de 3 camadas
• Escalabilidade horizontal



§2.5 1.000 pés—Model-View-Controller
(vs. Page Controller, Front Controller)

§2.6 500 pés: Modelos Active Record (vs. Data Mapper)
§2.7 500 pés: Controladores REST (*Representational State Transfer* para ações auto-contidas)
§2.8 500 pés: Template View (vs. Transform View)

• **Active Record** • **REST** • **Template View**
• Data Mapper • Transform View

Visão de 100.000 pés

- A web segue uma arquitetura cliente–servidor
- Processos fundamentalmente orientados a requisições–resposta

PROTOCOLOS TCP/IP

- Endereços IP (*Internet Protocol*) identificam uma interface física de rede com quatro octetos (`200.144.183.244`, 32 bits) ou com oito grupos de quatro dígitos hexadecimal (`2001:12d0:c000:91::41`, 128 bits)
  - o endereço especial `127.0.0.1` (`0:0:0:0:0:0:0:1`) aponta para "este computador", chamado de `localhost`, mesmo se não estiver conectado a Internet!
- TCP/IP (*Transmission Control Protocol / Internet Protocol*)
  - IP: não há garantias, pacotes são enviados tão bem quanto possível (*best-effort*) de um endereço IP para outro
  - TCP: torna o IP confiável ao detectar problemas no envio de pacotes (que não chegaram, que chegam fora de ordem, erros de transmissão, lentidão na rede, etc.) e se recuperar desses problemas
  - *Portas* TCP permitem múltiplas aplicações TCP no mesmo computador

Visão de 100.000 pés

- A web segue uma arquitetura cliente–servidor
- Processos fundamentalmente orientados a requisições–resposta
- DNS (*Domain Name System)* é outro tipo de servidor que mapeia nomes a endereços IP

- Protocolo ASCII de requisição–resposta para transferência de informação na Web
- Requisições HTTP incluem:
  - método de requisição (GET, POST, etc.)
  - Uniform Resource Identifier (URI)
  - versão do protocolo HTTP entendida pelo cliente
  - cabeçalhos — informação extra referente à requisição
- Resposta HTTP do servidor:
  - versão do protocolo & código de status
  - cabeçalhos da resposta
  - corpo da resposta

Códigos de status HTTP

2xx tudo ocorreu bem

3xx o recurso foi movido

4xx problema no acesso

5xx erro no servidor

PERGUNTA

Assumindo que “>” significa “depende de”, qual afirmação não é correta?

1. DNS > IP
2. HTTP > TCP > IP
3. TCP > DNS
4. Todas estão corretas

COOKIES

- Obs: HTTP é *stateless*
- Problema antigo: como guiar um usuário “através” de uma sequência de páginas?
  - usar o IP para identificar o usuário? Ruim, usuários compartilham um mesmo IP em uma rede compartilhada
  - embutir informação do usuário na URI? Ruim, quebra os caches
- Rapidamente substituído por *cookies*; assista: `http://screencast.saasbook.info`
- Arcabouços como o Rails gerenciam a adulteração de cookies para você

USOS DE COOKIES

- A maioria dos sites perceberam que manter um estado para cada usuário poderia ser útil para muitas coisas:
  - personalização (“My Yahoo!”)
  - rastreamento de clicks/fluxo
  - autenticação (logado ou não)
  - Quais desses podem ser implementados no lado do usuário? Quais não devem ser e por quê?
- Regra de ouro: não confie no cliente! Deve ser possível identificar alterações nos cookies

PERGUNTA

Um(a) _____ pode criar e modificar cookies; o(a) _____ é responsável por incluir o cookie correto em cada requisição

1. Navegador / app SaaS
2. app SaaS / navegador
3. requisição HTTP / navegador
4. app SaaS / resposta HTTP