MANUTENÇÃO DE SOFTWARE: REFATORAÇÃO NO NÍVEL DE MÉTODOS

ENGENHARIA DE SISTEMAS DE INFORMAÇÃO

Daniel Cordeiro

7 (aula “retrô”) e 10 de novembro de 2017

Escola de Artes, Ciências e Humanidades | EACH | USP

Código Legado & Refatoração: reflexões, falácias e armadilhas

FALÁCIAS E ARMADILHAS

"A maior parte do tempo gasto com o projeto, codificação e teste será usada com refatorações."

Falácia Será mais rápido jogar esse código fora e começar tudo de novo

Armadilha Misturar refatoração com implementação de melhorias

Armadilha Aderência rígida às métricas ou "alergia" a cheiros de código

Armadilha *Dívida Técnica*: esperar muito tempo para fazer uma "grande refatoração" (vs. refatoração contínua)

Um Raio X no fiasco do software do Affordable Care Act

HealthCare.gov

We have a lot of visitors on the site right now.

Please stay on this page.

We're working to make the experience better, and we don't want you to lose your place in line. We'll send you to the login page as soon as we can. Thanks for your patience!

Acontece nas melhoras famílias...

ESTADÃO ECONOMIA & NEGÓCIOS » Problemas com eSocial persistem para patrões e domésticos após seis meses

ECONOMIA ESOCIAL

1 Problemas com eSocial persistem para patrões e domésticos após seis meses

2 Oito dicas para evitar erros com o doméstico no eSocial

Problemas com eSocial persistem para patrões e domésticos após seis meses

Dificuldades técnicas ficaram para trás, mas demissão ainda gera dúvidas; relatos indicam demora na liberação de FGTS e seguro-desemprego

3

Hugo Passarelli
25 Abr | 2016 | 06h45

Foto: Gabriela Biló/Estadão

PERGUNTA

Qual a melhor aproximação para a quantidade de linhas de código escritas para o sistema web do site HealthCare.gov?

1. 0,5 milhão LOC — ônibus espacial: $\approx 0,4$ MLOC
2. 5 milhões LOC — Linux 3.0: ≈ 13 MLOC
3. 50 milhões LOC — Windows Vista: ≈ 50 MLOC
4. 500 milhões de LOC — um sistema bancário: ≈ 100 MLOC

Veja o artigo “The One Disheartening Number That Suggests Healthcare.gov Will Not Be Fixed Anytime Soon”: `http://www.slate.com/blogs/future_tense/2013/10/21/healthcare_gov_problems_why_5_million_lines_of_code_is_the_wrong_way_to.html`

TL;DR:

- Eles **deveriam** ter usado um método Ágil?
- Eles **poderiam** ter usado um método Ágil?
- Independentemente disso, o **código era bom**?

O QUE TEM NO CÓDIGO?

- Lógica para troca de companhias de seguros de 36 estados
- Integração: troca de companhias que já existiam em 14 estados
- Integração: sistemas de dados do IRS (receita federal) & *Social Security* (INSS) — verifica identidade, receita familiar, endereço, etc.
- Gerenciamento de identidade federado (cada sistema tem um conceito diferente de identidade)
- “Front-end”: registro, criação de conta, aplicações para guiar a troca de seguradora

O QUE TEM NO CÓDIGO?

- Aviso: só uma pequena parte do código foi lançado (uma combinação do JavaScript do lado do cliente + Node.js)
- O código JS não passa no JSHint (mau cheio de código)
- Não há diretório de testes
- Não há cache de conteúdo estático (JavaScript); mais de 2 MB de JS servidos em cada requisição de página
- Não usavam um CDN para servir o conteúdo estático
- Não usavam compressão (minificação) de JavaScript
    - poderia ter reduzido o tamanho e tempo de download em $\approx 75\%$

TAMANHO DO PROJETO COMO PREDITOR DE SUCESSO?

- CHAOS report de 2013 (Standish Group): para projetos de software grandes (> $ 10M):
  - 10% terminam no prazo e orçamento
  - 52% têm problemas (atrasado, acima do orçamento ou incompleto)
  - 38% falhou (cancelado ou entregue, mas nunca usado)
- “A verdadeira chave para o sucesso é fazer menos por menos. A chave para fazer menos por menos é dividir os projetos grandes em uma sequência de projetos menores ...”

CONTRATO

- 2017: 16 companhias incluindo a CGI Federal “habilitadas” para concorrer a US$ 4 B por “*Indefinite delivery, indefinite quantity*” nesse e em outros projetos correlatos
- 55 empresas contratadas, US$ 394M no total
- a maior foi a CGI Federal, com US$ 88 M (e outros US$ 8 B em contratos federais)

OPS

Em uma pesquisa com 832 usuários tentando se conectar ao site:

50% mensagem de “Tente mais tarde”

25% “Erro ao criar a conta”

38% “Site fora do ar”

19% sem problemas

ÀS VÉSPERAS DO LANÇAMENTO

"Ele é rápido, construído com HTML estático, totalmente escalável e seguro", disse Bryan Sivak, CIO do HHS em uma entrevista (ao se referir sobre o código do front-end desenvolvido pela Development Seed, Inc.)

OS PROBLEMAS DE SEMPRE

- “Mais usuários que o esperado” (> 20 milhões)
  - também aconteceu com o FarmVille! De 3 para 150 servidores na cloud em questão de semanas
- “Interfaces com sistemas legados” (Receita Federal, INSS)
  - companhias aéreas, bancos, corretoras de ações todos resolveram isso bem
  - será que alguém criou stubs para as APIs para testar o sistema?
- “Banco de dados pode ser um gargalo” (se comunica com companhias de seguro em 14 estados e serve 36 outros)
  - use uma fila para enfileirar as tarefas e entregue os resultados depois
- “Tempo insuficiente para testar”
  - na verdade isso vira um risco se deixado pro último momento (*Waterfall* se os requisitos mudam durante o desenvolvimento

LEI DE BROOKS

> *"Adicionar mais gente em um projeto de software atrasado faz com que o projeto atrase mais" — Fred Brooks Jr., The Mythical Man-Month*

Resposta: um "pico de tecnologia" com os "melhores e mais brilhantes" chamados para consertar o site

- engenheiros da Verizon
- "empreiteiros & experts da indústria de seguros"
- "veteranos das companhias mais importantes da Silicon Valley"
- "Essa nova infusão... irá trazer um vasta gama de expertise e habilidades em tópicos diferentes, incluindo uma vasta experiência em escalabilidade de sistemas de informação"

REVISÕES DE CÓDIGO?

- “Dados olhos suficientes, todos os erros são óbvios” — Eric S. Raymond, *The Cathedral & The Bazaar*
- Healthcare.gov: tiraram o código que inicialmente tinham colocado no GitHub
    - felizmente alguém tinha feito um *fork* (STRML/Healthcare.gov-Marketplace) e começou a inspecionar e corrigir

DIVISÃO DE TRABALHO?

- Armadilha: dividir o trabalho em front-end/back-end vs. ter um dono de uma história end-to-end
- Healthcare.gov: empresas que desenvolviam o front- & back-end
  - não conseguiam falar entre eles
  - front-end foi entregue antes, com código e UI boas
  - não havia um responsável pela UI end-to-end ou teste de stress
  - integrações de back-end eram difíceis e aparentemente foram insuficientemente testadas

CLIENTE QUE MUDA DE IDEIA?

- Armadilha: falhar ao construir a coisa certa, mesmo que tenha construído certo a coisa
- Healthcare.gov: “Só nos últimos 10 meses, (...) oficiais do governo mudaram os requisitos de hardware e software sete vezes”. *Insurance site seen needing weeks to fix, NY Times, 21/10/2013*

ESTIMATIVA DE CUSTOS ÁGIL

- O mundo real precisa estimar os custos antes que o cliente concorde com o projeto
- Ágil: qualidade, cronograma, escopo $\Rightarrow$ escolha 2
- Healthcare.gov: definiu a data da implantação, pediu um plano antecipado, empenhou o dinheiro para o empreiteiro

IMPLANTAÇÃO INCREMENTAL?

- Uso de “Feature flags” para implantação incremental
  - algumas poucas funcionalidades por vez
  - para poucos usuários por vez
  - todo grande site SaaS de sucesso faz isso, crescendo “organicamente” ao longo do tempo
- Healthcare.gov: a coisa toda foi pro ar em todos os estados, para todos os usuários, em um único dia

SIM: PLANEJE-E-DOCUMENTE
NÃO: MÉTODOS ÁGEIS

1. É necessário criar uma especificação?
2. Os clientes estarão indisponíveis durante o desenvolvimento?
3. O sistema a ser construído é muito grande?
4. O sistema a ser construído é muito complexo (ex: sistema de tempo real)?
5. O sistema terá uma vida útil muito longa?
6. Você usa ferramentas de softwares pobres?
7. O time está geograficamente distribuído?
8. A cultura do time (seu modo de pensar e planejar) é orientado à documentação?
9. O time possui habilidades de programação fracas?
10. O sistema a ser construído está sujeito a regulações e normas?

SISTEMA DE LICITAÇÃO QUEBRADO?

58 empresas “preferidas” tendem a dominar os contratos de IT nos EUA

- 6500 páginas de regulamentações
- Processo herdado do departamento de defesa, que requer a especificação completa antecipadamente
- Empresa licitada teme ser processada por não respeitar a regulamentação e acabar perdendo o contrato pro concorrente
- O escritório de licitações do governo teme ser acusada de não seguir as regras pelas empresas que não foram escolhidas
- (Essas empresas gastam milhões de dólares em *lobbying*)

AS REGRAS DE LICITAÇÃO FEDERAL TRABALHAM CONTRA ÁGIL

- “Se eu fosse dar um lance em um projeto completo, eu precisaria de mais advogados e de escritores de propostas do que de engenheiros para construir o projeto... Se as pessoas não estão vendo a necessidade de realizar uma reforma no sistema de licitações, então temos um problema”
  - Eric Gundersen, Development Seed, Inc.
  - Development Seed foi, na verdade, terceirizado pela Aquilent
- Não é possível mudar o curso no meio da corrente sem quebrar as regras do contrato
- Modelo Cascata é usado porque é o processo herdado de projetos de hardware, e “é o mais fácil de conseguir”

RESUMO

- Desafios técnicos que não são triviais, mas que não são novos
- Dividir em projetos menores poderia ter ajudado
- Métodos Ágeis poderiam ter ajudado (e teriam funcionado com equipes menores), mas as regras de licitação federal fazem com que Ágil seja basicamente ilegal
- A qualidade do código se revelou, no mínimo, embaraçosa