# GESTÃO DE PROJETOS: SCRUM E PROGRAMAÇÃO PAREADA

## ACH2006 – ENGENHARIA DE SISTEMAS DE INFORMAÇÃO
## SIN5005 – TÓPICOS EM ENGENHARIA DE SOFTWARE

Daniel Cordeiro

Escola de Artes, Ciências e Humanidades | EACH | USP

# 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...



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 Abril 2016 | 06h48

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: $\approx 13$ MLOC
3. 50 milhões LOC — Windows Vista: $\approx 50$ MLOC
4. 500 milhões de LOC — um sistema bancário: $\approx 100$ 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)

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

> *"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

# Gestão de Projetos

# ENGENHARIA DE SOFTWARE É UM ESPORTE COLETIVO

- Estamos na era pós-desenvolvedor-super-herói :)
- Expectativas sobre funcionalidades/qualidade mais altas
  - um único programador brilhante não é mais capaz de construir sozinho software completamente inovador
- Carreira de sucesso em SW $\Rightarrow$ excelente programador && trabalha bem com outras pessoas && consegue ajudar uma equipe a ser bem sucedida
- *"Não há vencedores em uma equipe mal sucedida, nem perdedores em uma equipe bem sucedida."*
  *–Fred Brooks Jr.*

# EXEMPLO: VIDEOGAMES

| Jogo | Ano | Plataforma | Equipe de Devs. |
|---|---|---|---|
| Space Invaders | 1981 | Arcade | 1 |

# EXEMPLO: VIDEOGAMES

| Jogo | Ano | Plataforma | Equipe de Devs. |
|---|---|---|---|
| Space Invaders | 1981 | Arcade | 1 |
| Super Mario Bros. | 1985 | NES (Nintendo) | 8 |

# EXEMPLO: VIDEOGAMES

| Jogo | Ano | Plataforma | Equipe de Devs. |
|---|---|---|---|
| Space Invaders | 1981 | Arcade | 1 |
| Super Mario Bros. | 1985 | NES (Nintendo) | 8 |
| Sonic the Hedgehog | 1999 | Sega Dreamcast | 30 |

# EXEMPLO: VIDEOGAMES

| Jogo | Ano | Plataforma | Equipe de Devs. |
|---|---|---|---|
| Space Invaders | 1981 | Arcade | 1 |
| Super Mario Bros. | 1985 | NES (Nintendo) | 8 |
| Sonic the Hedgehog | 1999 | Sega Dreamcast | 30 |
| Resident Evil 6 | 2012 | PC, PS3, Xbox 360 | 600 |

# Organização alternativa da equipe?

- Planeje-e-Documente requer documentação e planejamento extensivos e depende de um gerente experiente
- Como deveríamos organizar uma equipe Ágil?
- Há alguma alternativa a organização hierárquica, com um gerente que executa o projeto?

# SCRUM: ORGANIZAÇÃO DA EQUIPE

- Equipe de “2 pizzas” (4 a 9 pessoas)
- “Scrum” inspirado em reuniões curtas e frequentes:
  - reuniões de 15 minutos sempre na mesma hora e lugar
  - para aprender mais: *Agile Software Development with Scrum*, de Schwaber & Beedle

# REUNIÃO DIÁRIA DE SCRUM

- Todos devem responder a três perguntas:
  1. O que você fez desde a reunião de ontem?
  2. O que você planeja fazer hoje?
  3. Existe algum impedimento ou obstáculo?
- Ajuda cada membro da equipe a identificar o que ele precisa

# PAPÉIS NO SCRUM

- Equipe: uma equipe de tamanho “2 pizzas” que entrega o software
- ScrumMaster: o membro da equipe que:
  - protege a equipe de distrações externas
  - mantém a equipe focada no trabalho em questão
  - impõe regras (ex: padrões de código)
  - remove os obstáculos que impedem a equipe de progredir
- Proprietário do produto (*product owner*): um membro da equipe (que não o ScrumMaster) que representa a voz do cliente e prioriza as histórias de usuário

O scrum baseia-se na auto-organização e os membros da equipe frequentemente se revezam entre funções diferentes.

# RESOLUÇÃO DE CONFLITOS I

Conflitos podem ocorrer (como em qualquer trabalho em equipe).
Ex: visões contrárias sobre a direção técnica que deverá ser seguida.

1. 1º liste os itens nos quais todos concordam:
   - não comece listando as divergências
   - muitas vezes percebe-se que eles estão mais de acordo do que imaginavam
2. cada lado tenta articular os argumentos do outro, mesmo que não concorde com tudo
   - evita confusão sobre os termos ou hipóteses, o que muitas vezes é a causa do conflito

# RESOLUÇÃO DE CONFLITOS II

3. Confronto construtivo (Intel) — se você discorda veemente de uma proposta, você é obrigado a contestá-la (mesmo para os seus chefes)
4. Discordar, mas se comprometer (Intel)
    - uma vez que a decisão for tomada, você deve abraçá-la e seguir em frente
    - “Eu discordo, mas vou ajudar a fazer mesmo que eu discorde”.

Resolução de conflitos também pode ser útil na vida pessoal!

# SCRUM: RESUMO

- Basicamente, uma pequena equipe auto-organizada com uma reunião curta (e de pé) todos os dias
- Trabalho dividido em “sprints” de 2–4 semanas
- Sugere que os membros troquem os papéis entre si (especialmente o proprietário do produto) a cada iteração

Créditos: https://commons.wikimedia.org/wiki/File:ST_vs_Gloucester_-_Match_-_23.JPG

# DICA PARA USO DE SCRUM

☺

- “As reuniões em pé de 5 minutos realmente nos ajudaram a ficar na linha e a compartilhar conhecimento quando estávamos empacados”

☹

- “O maior desafio para nós era a comunicação/coordenação do time”
- “Tenha um líder scrum por vez, rotacione o cargo”
- “1 reunião por semana não era suficiente”

# Programação pareada

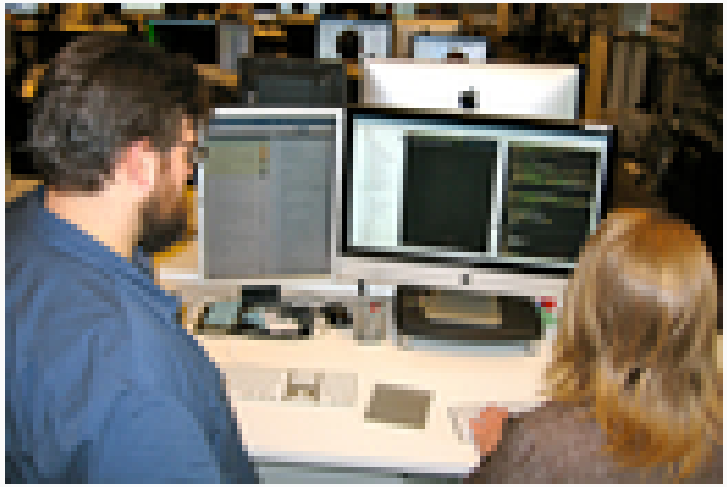
# DUAS CABEÇAS PENSAM MELHOR DO QUE UMA?

- Estereótipo: um “lobo solitário” trabalhando a noite toda à base de energéticos
- Será que há uma maneira mais sociável de programar?
  - Quais seriam os benefícios de termos várias pessoas programando juntas?
- Como você faria para evitar que uma pessoa fizesse todo o trabalho, enquanto a outra pega um café e checa seu Facebook?

**Objetivo**
Melhorar a qualidade do software e reduzir o tempo para fazer uma tarefa ao manter 2 pessoas desenvolvendo o mesmo código.

- Algumas pessoas (e empresas) adoram
- Alguns alunos também gostavam e aplicavam isso em seus projetos

# PROGRAMAÇÃO PAREADA



- Sente lado a lado colocando os monitores voltados para si
- Os computadores não são “pessoais”; são para uso dos pares
- Para evitar distrações, nada de ler e-mails ou navegar na web

- O piloto digita o código e pensa taticamente em como completar a tarefa atual, explicando seus pensamentos em voz alta enquanto digita
- O navegador revê cada linha de código assim que ela é digitada, agindo como uma rede de segurança para o piloto
- O navegador pensa estrategicamente sobre os problemas futuros, fazendo sugestões para o piloto
- Requer bastante conversa e concentração
- **Os pares alternam os papéis**

# AVALIAÇÃO DE PROGRAMAÇÃO PAREADA

- PP é mais rápida quando a tarefa é mais simples
- PP alcança maior qualidade quando a tarefa é mais difícil
  - … e código mais legível também
- Mas requer mais esforço de concentração do que programação solo
- Além disso, funciona como ferramenta de transferência de conhecimento
  - expressões idiomáticas de linguagens de programação, truques de ferramentas, processos da empresa, tecnologias mais novas, etc.
  - algumas equipes forçam a troca dos pares a cada nova tarefa; eventualmente todo mundo é pareado com todo mundo (“pareamento promíscuo”)

# Programação pareada: o que (não) fazer

- Não fique mexendo no seu *smartphone* enquanto estiver de navegador
- Considere formar um par com alguém que não tenha o mesmo nível de experiência que você — vocês dois irão aprender com isso!
  - tentar explicar é uma ótima forma de entender algo
- Troque o papel frequentemente — cada papel exercita um conjunto de habilidades diferentes e você aprenderá algo com ambos
  - o navegador aprimora sua capacidade de explicar suas ideias ao piloto

☺

- “Nos ajudou a evitar erros bobos que poderiam ter nos custado bastante tempo para depurar”
- “A troca de pares frequentemente acabou deixando o time mais coeso”