

ENGENHARIA DE SOFTWARE

2 - DEFININDO A ENGENHARIA DE SOFTWARE

Prof. Cleidson de Souza

Profa. Carla Alessandra Lima Reis

Prof. Rodrigo Quites Reis

Causas da Crise de Software

- Dificuldades Essenciais
 - Complexidade;
 - Conformidade;
 - Modificabilidade; e
 - Invisibilidade.
- Dificuldades Acidentais
 - Má qualidade dos métodos, linguagens, ferramentas, processos e modelos de ciclo de vida; e
 - Falta de qualificação técnica.

Causas da Crise de Software (2)



No Silver Bullet

Essence and Accidents of Software Engineering

Frederick P. Brooks, Jr.

University of North Carolina at Chapel Hill

Fashioning complex conceptual constructs is the *essence*; accidental tasks arise in representing the constructs in language. Past progress has so reduced the accidental tasks that future progress now depends upon addressing the essence.

Of all the monsters that fill the nightmares of our folklore, none terrify more than werewolves, because they transform unexpectedly from the familiar into horrors. For these, one seeks bullets of silver that can magically lay them to rest.

The familiar software project, at least as seen by the nontechnical manager, has something of this character; it is usually innocent and straightforward, but is capable of becoming a monster of missed schedules, blown budgets, and flawed products. So we hear desperate cries for a silver bullet—something to make software costs drop as rapidly as computer hardware costs do.

But, as we look to the horizon of a decade hence, we see no silver bullet. There is no single development, in either technology or in management technique, that by itself promises even one order-of-magnitude improvement in productivity, in reliability, in simplicity. In this article, I shall try to show why, by examining both the nature of the software problem and the

throughs—and indeed, I believe such to be inconsistent with the nature of software—many encouraging innovations are under way. A disciplined, consistent effort to develop, propagate, and exploit these innovations should indeed yield an order-of-magnitude improvement. There is no royal road, but there is a road.

The first step toward the management of disease was replacement of demon theories and humours theories by the germ theory. That very step, the beginning of hope, in itself dashed all hopes of magical solutions. It told workers that progress would be made stepwise, at great effort, and that a persistent, unremitting care would have to be paid to a discipline of cleanliness. So it is with software engineering today.

Does it have to be hard?—Essential difficulties

Frederick Brooks. No Silver Bullet - Essence and Accidents of Software Engineering. IEEE Computer, 1987. Imagem de:
https://twitter.com/zeljko_obren/status/909014656802574336

Software em ~2017

- Windows 10: aprox 35-80 milhões LOC
 - Fonte: <https://www.quora.com/How-many-lines-of-code-does-Windows-10-contain>
- Linux 3.10 kernel: 19,5 milhões de LOC
 - Fonte: Wikipedia
- Twitter: 3.000 imagens por segundo
 - Fonte: <http://highscalability.com/blog/category/example>
- Netflix: cada filme é gravado em mais de 120 versões com diferentes resoluções
 - Fonte: <http://highscalability.com/blog/2015/11/9/a-360-degree-view-of-the-entire-netflix-stack.html>
- WhatsApp: 700 milhões usuários, 30 bi de msgs/dia
 - Fonte: https://m.facebook.com/story.php?story_fbid=10152994719980011&id=500035010

Software em ~2024

- Microsoft Windows : 50 milhões de LOC
- Google: 2 bilhões de LOC
 - <https://www.wired.com/2015/09/google-2-billion-lines-codeand-one-place/>

Mundo Real: Para ilustrar a complexidade envolvida na construção de sistemas de software reais, vamos dar alguns números sobre o tamanho desses sistemas, em linhas de código. Por exemplo, o sistema operacional Linux, em sua versão 4.1.3, de 2017, possui cerca de 25 milhões de linhas de código e contribuições de quase 1.700 engenheiros ([link](#)). Para mencionar um segundo exemplo, os sistemas do Google somavam 2 bilhões de linhas de código, distribuídas por 9 milhões de arquivos, em janeiro de 2015 ([link](#)). Nesta época, cerca de 40 mil solicitações de mudanças de código (commits) eram realizadas, em média, por dia, pelos cerca de 25 mil Engenheiros de Software empregados pelo Google nessa época.

- GPT e variantes (como ChatGPT): 175 mil LOC
 - <https://www.quora.com/What-s-the-code-for-Chat-GPT-look-like-How-many-lines>

Software 2021 ...

- Como organizar as atividades tal que 5.000 pessoas possam trabalhar juntas ao mesmo tempo?
 - Processo de software;
 - Ferramentas de Gerência de Configuração;
- Como testar tanto código? Para tantas plataformas diferentes?
- Como projetar um sistema com 80 milhões de linhas de código? Como garantir a integridade deste projeto?

Software em 2024 ...

- *Large Language Models (LLMs)* são os modelos usados em ferramentas de Inteligência Artificial generativa;
- Estes modelos são não-determinísticos, ou seja, dada uma mesma entrada, produzem duas respostas diferentes.
 - Sendo assim, como testar este software?
 - Como definir o que o software precisa fazer?



Definição de Engenharia de Software

Engenharia de Software - Definições

- Engenharia de software é a disciplina que lida com a construção de sistemas de software flexíveis, modulares, robustos, confiáveis, usáveis e adequados ao contexto sócio-técnico onde estes sistemas estão inseridos.
- Uma disciplina que lida com a construção de sistemas de software que são tão grandes que são construídos por um ou vários times de engenheiros. [Ghezzi, Jazayeri, Mandrioli]
- Uma disciplina cujo objetivo é a produção de software sem falhas, entregue dentro de prazo e orçamento e satisfazendo as necessidades dos usuários. Além disso, o software precisa ser facilmente modificável quando as necessidades do usuário mudam.[Schach]

Engenharia de Software - Definições

- Aplicação prática de conhecimento científico no projeto e construção de programas e da documentação requerida para desenvolver, operar e manter esses programas. [Boehm]
- O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente software que seja confiável e que funcione eficientemente em máquinas reais. [Fritz Bauer-1969]
- Construção de um software com várias versões por várias pessoas (*multi-person construction of multi-version software*).[Parnas]

Introdução

- Engenharia de software:
 - “O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente software que seja confiável e que funcione eficientemente em máquinas reais” (Fritz Bauer-1969)
 - Inclui:
 - Métodos e metodologias de desenvolvimento de software;
 - Planejamento, estimativas, gerência de projetos em andamento;
 - Projeto de programas (algoritmos, estruturas de dados);
 - Ferramentas;

O que é a Engenharia de Software?

- “Engenharia de Software é a aplicação de uma **abordagem sistemática**, disciplinada e quantificável ao **desenvolvimento, operação e manutenção de software**”
 - *IEEE Std 610.12 (1990)*

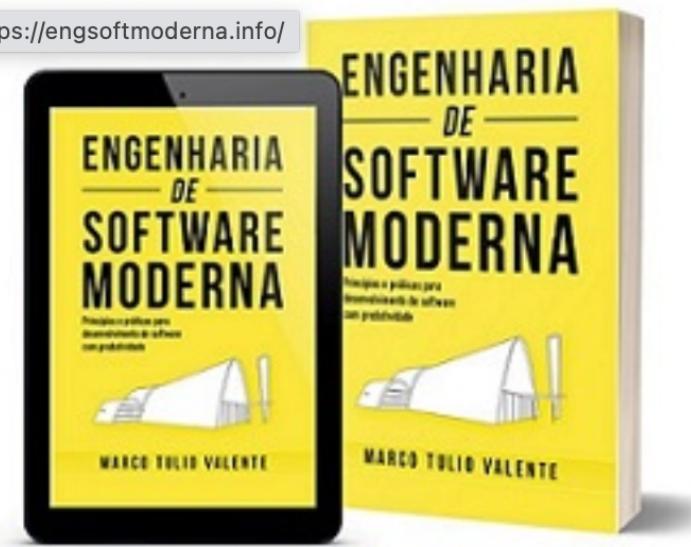
Objetivos da Engenharia de Software

- Controle sobre o desenvolvimento de software dentro de **custos, prazos** e níveis de **qualidade** desejados
- Produtividade no desenvolvimento, operação e manutenção de software
- Qualidade versus Produtividade
- Permitir que profissionais tenham controle sobre o desenvolvimento de software dentro de custos, prazos e níveis de qualidade desejados

Definição de Engenharia de Software

- ❑ Área da Computação destinada a **investigar os desafios e propor soluções** que permitam **desenvolver sistemas de software**;
- ❑ Principalmente aqueles mais complexos e de maior tamanho – de forma produtiva e com qualidade;

<https://engsoftmoderna.info/>



- O software que satisfaz os **requisitos** solicitados pelo usuário. Deve ser fácil de manter, ter boa performance, ser confiável e fácil de usar
- Alguns atributos de qualidade
 - Manutenibilidade
 - O software deve evoluir para **atender os requisitos que mudam**
 - Eficiência
 - O software não deve desperdiçar os recursos do sistema
 - Usabilidade
 - O software deve ser fácil de usar pelos usuários para os quais ele foi projetado

Importância da Engenharia de Software

- Qualidade de software e produtividade garantem:
 - Disponibilidade de serviços essenciais
 - Segurança de pessoas
 - Competitividade das empresas
 - Produtores
 - Consumidores

Introdução

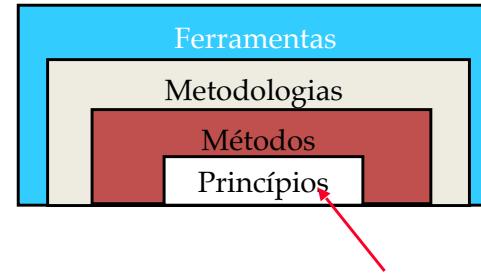
- Conceitos de Engenharia de Software
 - Objetivo: descrever e exemplificar, de forma geral, os princípios, métodos, metodologias e ferramentas de ES

[Guezzi-1994]

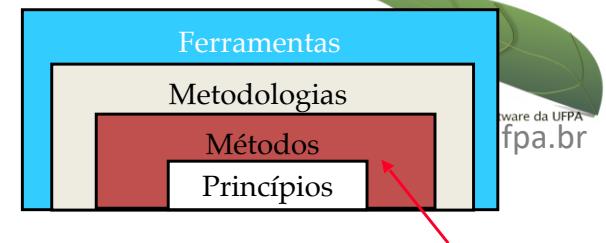


Introdução

- Conceitos de Engenharia de Software
 - Princípios
 - Modularidade
 - Consistência
 - Completude
 - Confiabilidade
 - Corretude
 - Manutenibilidade
 - Reusabilidade
 - Eficiência
 - Abstração
 - Formalismo

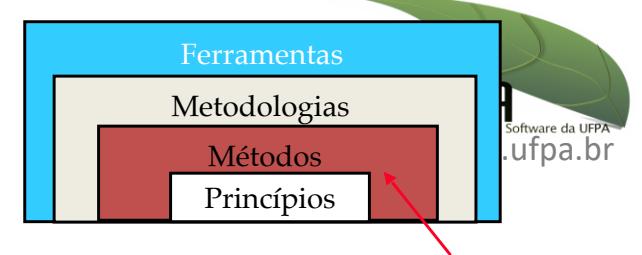


Introdução

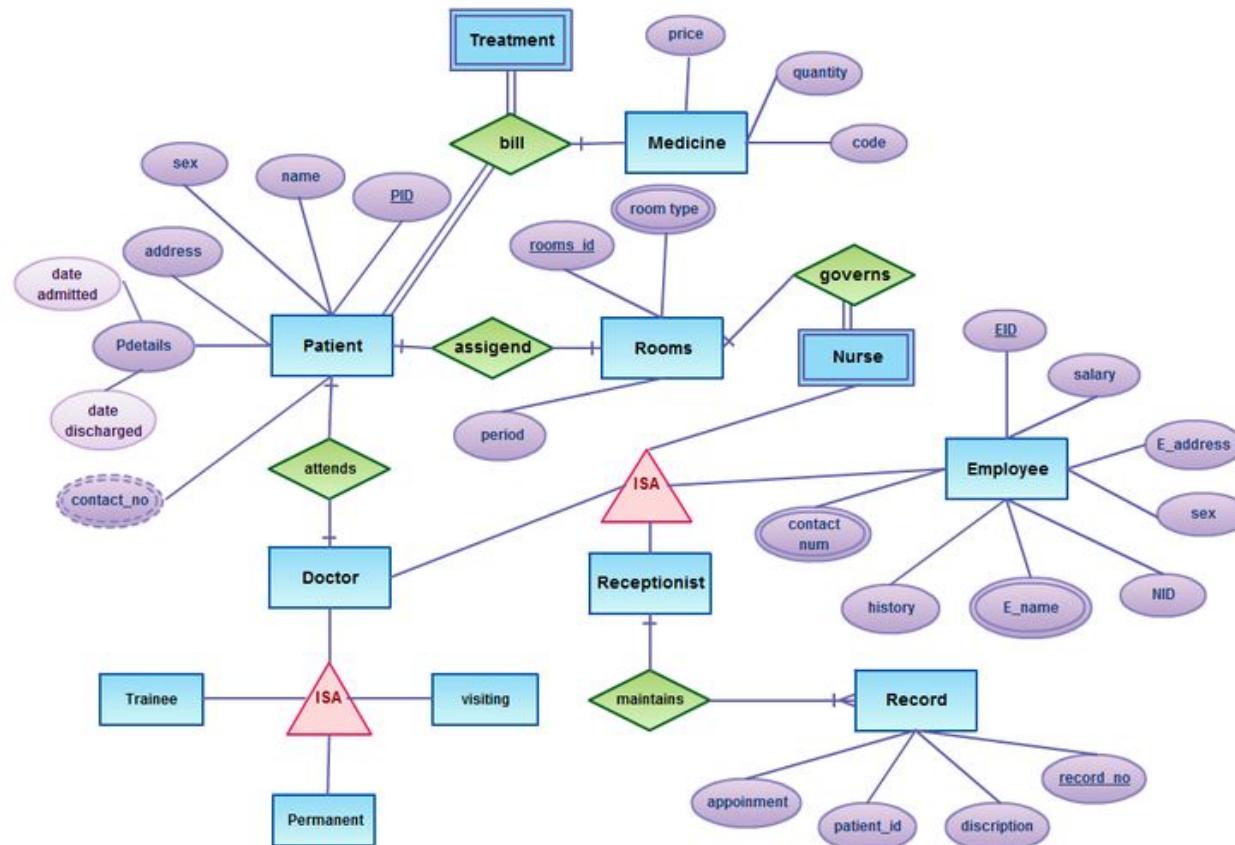


- Conceitos de Engenharia de Software
 - Métodos ou Modelos:
 - Modelos com algum formalismo para:
 - compreender
 - comunicar
 - Exemplos:
 - Diagramas: UML, ER...
 - Especificações textuais
 - Diferentes níveis de formalismo (informais, semi-formais ou formais)

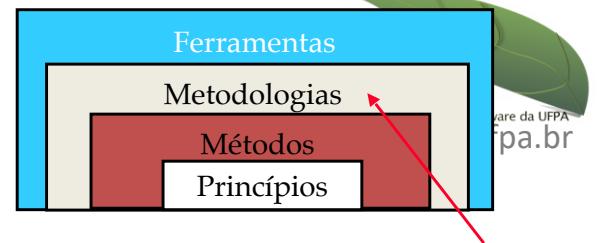
Introdução



- Conceitos de Engenharia de Software
 - Métodos ou Modelos
 - Gráficos, semi-formais



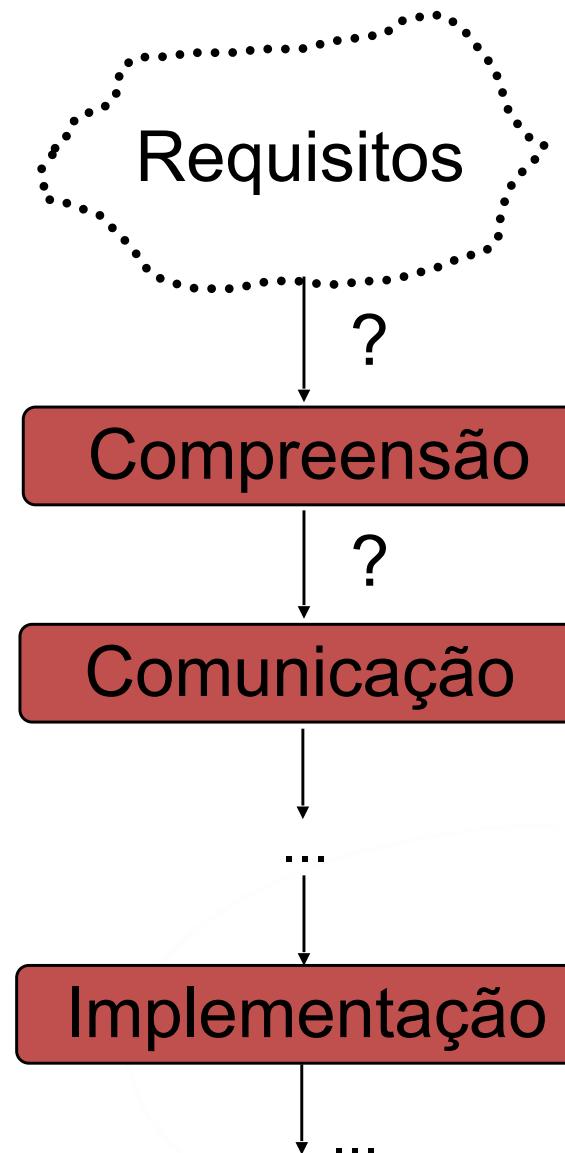
Introdução



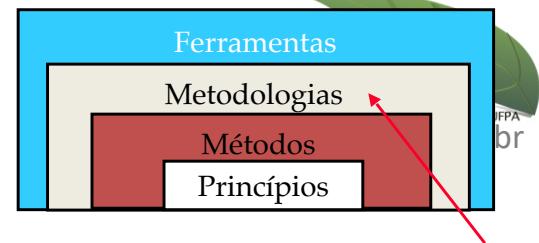
- Conceitos de Engenharia de Software
 - Metodologia, Ciclo de vida ou Processo:
 - Considerados sinônimos nesta disciplina
 - Conjunto de passos que visam a construção de software
 - Gerenciável
 - Completo
 - Em vários níveis de abstração
 -

Introdução

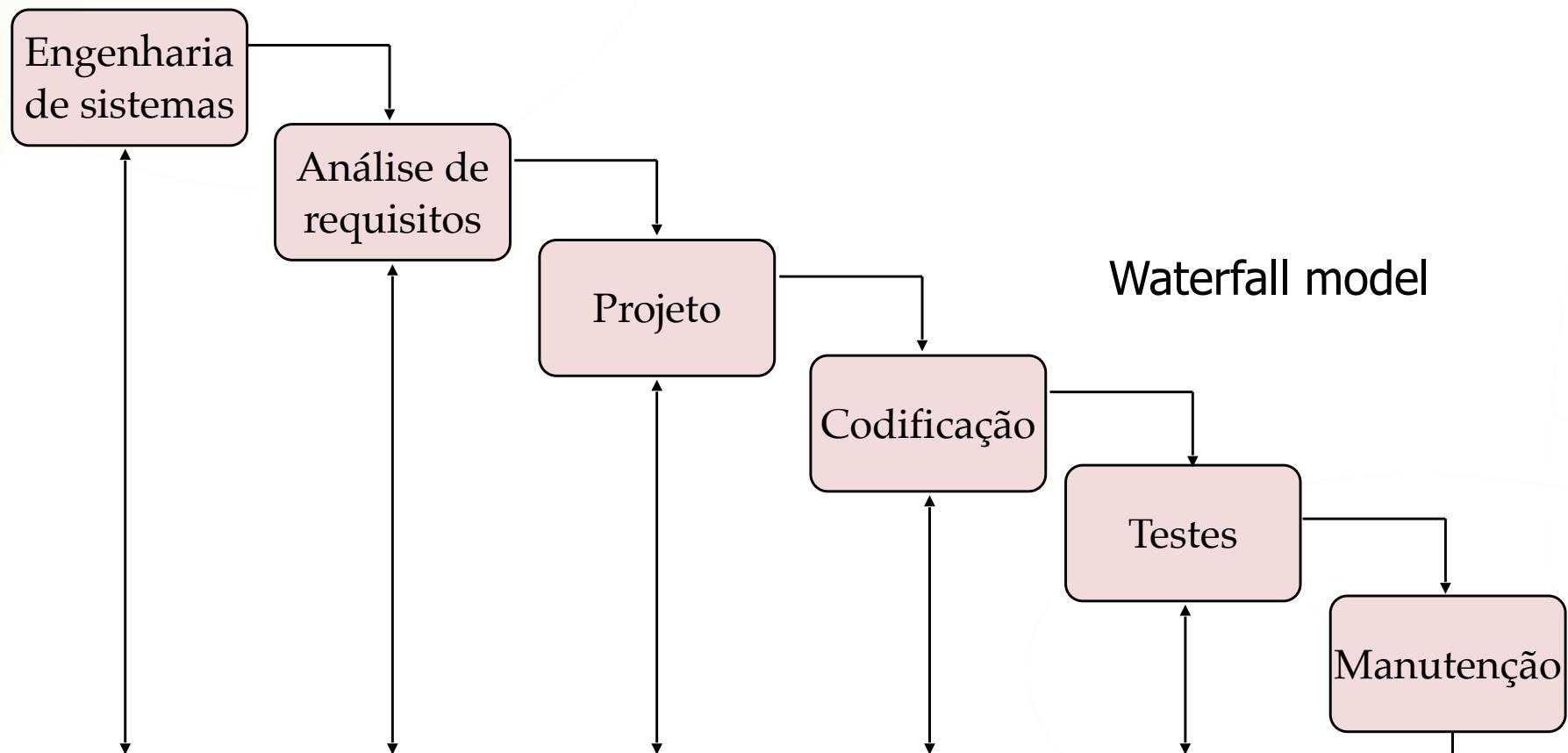
- Processo de Software:
descrição genérica



Introdução



- Conceitos de Engenharia de Software
 - Metodologia, Ciclo de vida ou Processo

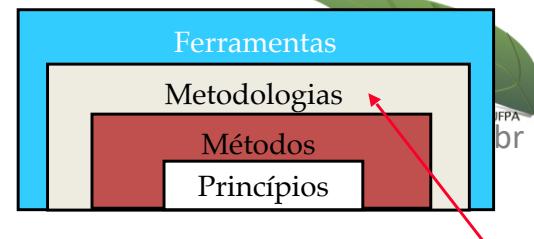


Introdução

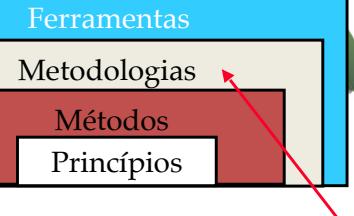
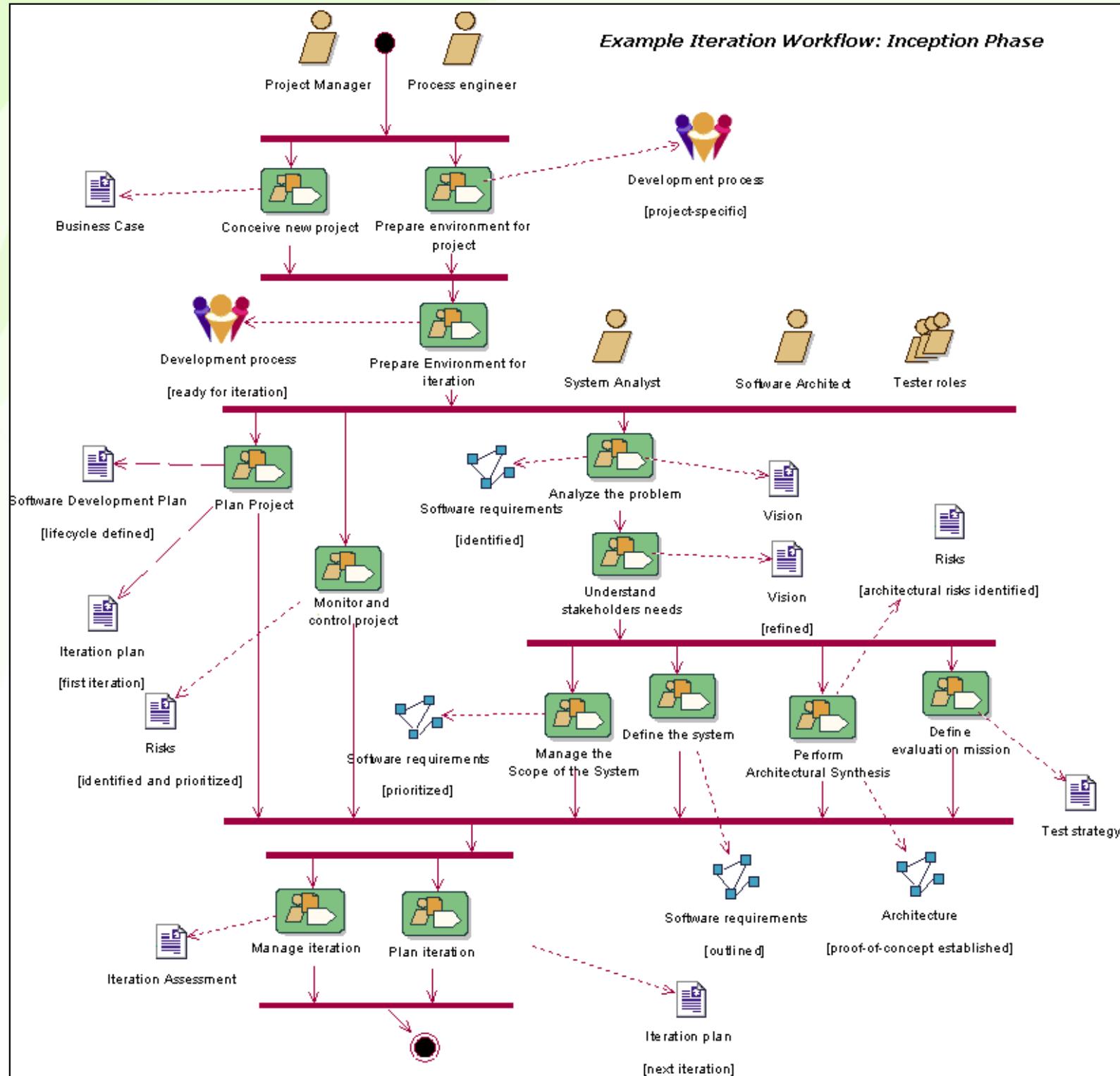


- Conceitos de Engenharia de Software
 - Metodologia, Ciclo de vida ou Processo
 - Montagem de um sistema a partir de componentes reutilizáveis
 - Desenvolvimento Evolucionário e Prototipação
 - Intercala atividades de especificação, desenvolvimento e validação
 - Transformação formal
 - Baseia-se na especificação formal do sistema e na transformação dessa especificação em um programa (usando modelos matemáticos ou ferramentas)
 - ...

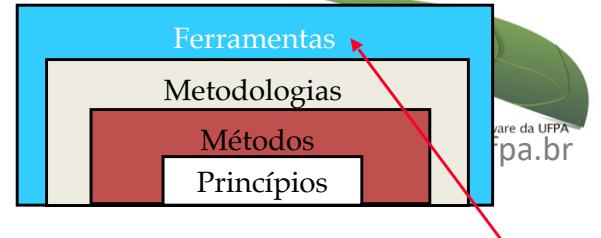
Introdução



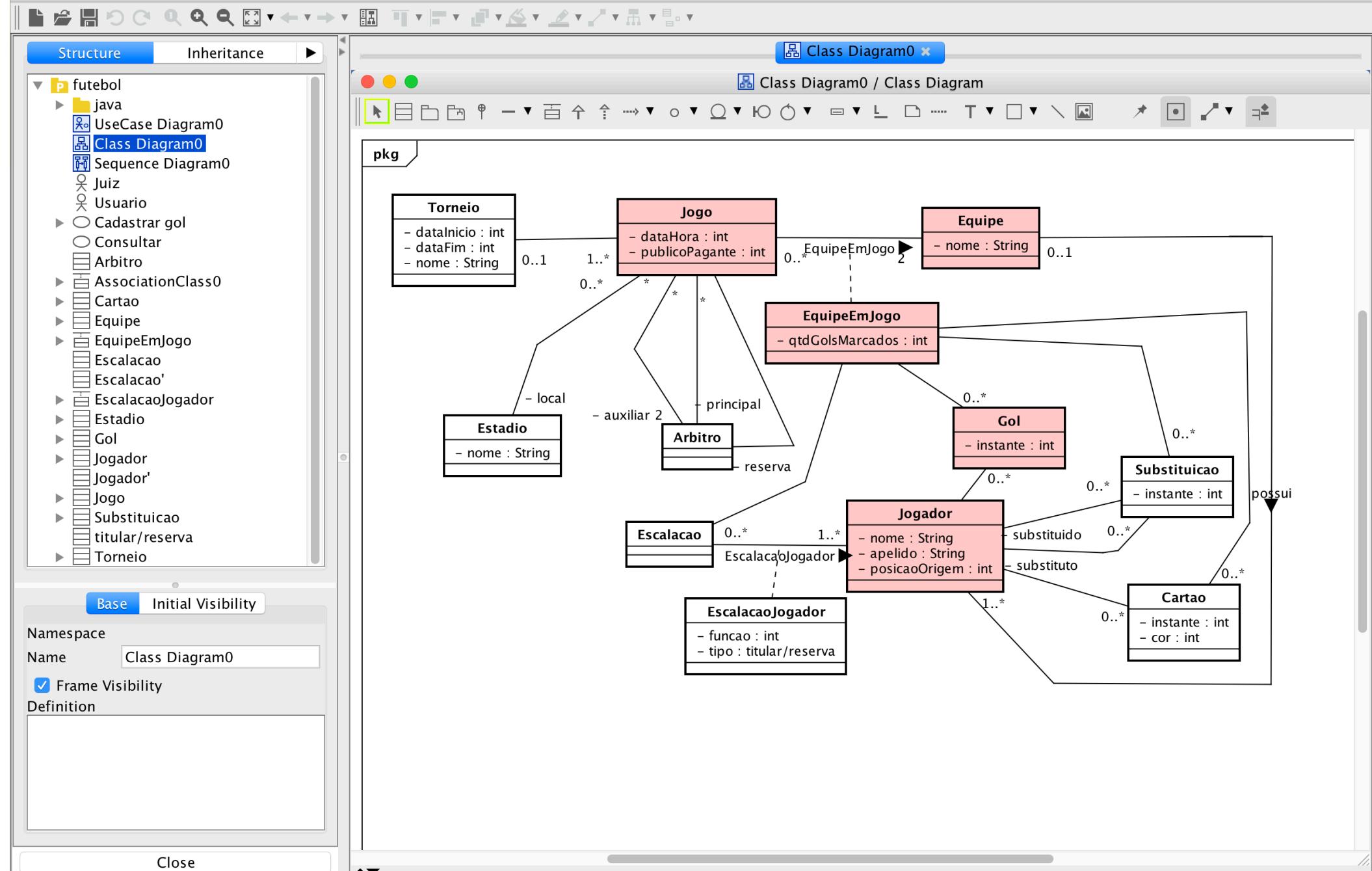
- Conceitos de Engenharia de Software
 - Metodologia, Ciclo de vida ou Processo
 - Modelagem do Processo de Software
 - Utiliza linguagens especializadas (Process Modeling Languages - PMLs)
 - Um modelo de processo pode ser usado para
 - » Descrever um processo que já ocorreu na organização, e/ ou
 - » Prescrever um processo futuro, a ser desempenhado



Introdução



- Conceitos de Engenharia de Software
 - Ferramentas CASE
 - Software que atua em pontos específicos do processo
 - Exemplo:
 - Análise (modelagem)
 - Projeto (modelagem, simulação e geradores de código)
 - Testes (geradores de testes)
 - Planejamento (cálculo de estimativas)



Plug-in Development - NbBundleTest.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Plug-in Dev... >

Package Explorer Plug-ins

com.aramco.powers2.ui powers2gui.product GenericBranch.java NbBundleTest.java

com.aramco.powers2.ui

src

- com.aramco.powers2.ui
 - AppActionBarAdvisor.java
 - Application.java
 - AppWorkbenchAdvisor.java
 - AppWorkbenchWindowAdvis
 - ICommandIds.java
 - MessagePopupAction.java
 - NbBundle.java
 - OpenViewAction.java
 - Perspective.java
 - PluginConstants.java
 - Powers2Plugin.java
 - ProjectView.java
 - TableEditor.java
 - TableView.java
 - Bundle.properties
- com.aramco.powers2.ui.action
- com.aramco.powers2.ui.forms
- com.aramco.powers2.ui.project.r
- com.aramco.powers2.ui.table
- com.aramco.powers2.ui.wizards
- com.aramco.powers2.xyplot.data

test

- com.aramco.powers2.internal.ui
- com.aramco.powers2.ui.test
 - NbBundleTest.java
- com.aramco.powers2.xyplot.data
- samples

JRE System Library [jdk1.5.0_06]

Plug-in Dependencies

JUnit 4

doc

icons

META-INF

build.properties

com.aramco.powers2.ui.project.moc

IPlotDataModel.violet

plugin_customization.ini

Outline

com.aramco.powers2.ui.test

import declarations

NbBundleTest

- main(String[])
- testExistingResource()
- testNonExistingResource()

com.aramco.powers2.ui.test.NbBundleTest

27 import com.aramco.powers2.ui.NbBundle;

28

29 /**

30 * Tests the behavior of utility class NbBundle.

31 * Tests need to run against the background of a known set of objects.

32 * This set of objects is called a test fixture. (Refer to <http://www.junit.org>)

33 *

34 * @author Guanglin Du (dugl@petrochina.com.cn), Software Engineering Center, RIPED, PetroChina

35 */

36 public class NbBundleTest {

37

38 /**

39 * Uses the Bundle.properties to test NbBundle's behavior.

40 */

41 @Test

42 public void testExistingResource() {

43 String s1 = NbBundle.getMessage(ProjectView.class, "add_new_pvt_sat");

44 assertEquals("Add New PVT or SAT table", s1);

45 }

46

47 /**

48 * Uses the Bundle.properties to test NbBundle's behavior.

49 */

50 @Test

51 public void testNonExistingResource() {

52 String s1 = NbBundle.getMessage(ProjectView.class, "non-existing");

53 assertEquals("%non-existing", s1);

54 }

55

56 /**

57 * Method main to run this class directly.

58 * Can be run this way also on a command line:

59 * java org.junit.runner.JUnitCore samples.SimpleTestFixture

60 */

61 public static void main(String args[]) {

62 JUnitCore.main("com.aramco.powers2.ui.util.test.NbBundleTest");

63 }

64}

65

Error Log Tasks Problems Console Properties Search JUnit

Finished after 0.129 seconds

Runs: 2/2 Errors: 0 Failures: 0

com.aramco.powers2.ui.test.NbBundleTest [Runner: JUnit 4] Failure Trace

Read Me Trim (Bottom)

开始 file:/home - Konqueror Plug-in Development - NbBu 08:26



O que se estuda em Engenharia de Software?

Alguns temas da ES

- 1. Engenharia de Requisitos
- 2. Projeto de Software
- 3. Construção de Software
- 4. Testes de Software
- 5. Manutenção de Software
- 6. Gerência de Configuração
- 7. Gerência de Projetos
- 8. Processos de Software
- 9. Modelos de Software
- 10. Qualidade de Software
- 11. Prática Profissional
(Colaboração e Ética)
- 12. Aspectos Econômicos



Requisitos de Software

- **Definição:** o que sistema deve fazer para atender aos seus clientes com qualidade de serviço
- **Tipos**
 - **Funcionais:** "o que" um sistema deve fazer
 - Funcionalidades ou serviços ele deve implementar
 - **Não-funcionais:** "como" um sistema deve operar
 - Sob quais restrições e com qual qualidade de serviço

Exemplos de Requisitos Funcionais

- Informar o saldo de uma conta corrente
- Informar o extrato de uma conta corrente para um determinado período
- Pagar com PIX
- Receber com PIX
- Aceitar depósitos
- Fazer transferências

Exemplos de Requisitos Não-Funcionais

- Desempenho: dar o saldo da conta em 5 segundos
- Disponibilidade: estar no ar 99.99% do tempo
- Capacidade: armazenar dados de 1M de clientes
- Tolerância a falhas: continuar operando se São Paulo cair
- Segurança: criptografar dados trocados com as agências

Exemplos de Requisitos Não-Funcionais (2)

- Privacidade: não armazenar localização dos usuários
- Interoperabilidade: se integrar com os sistema do BACEN
- Manutenibilidade: bugs devem ser corrigidos em 24 hs
- Usabilidade: versão para celulares e tablets

Testes de Software

- Verificam se um programa apresenta um resultado esperado ao ser executado com casos de teste
- Podem ser:
 - Manuais
 - Automatizados (novo foco)

Falha Famosa:



30 segundos depois

Custo do foguete e satélite:
US\$ 500 milhões

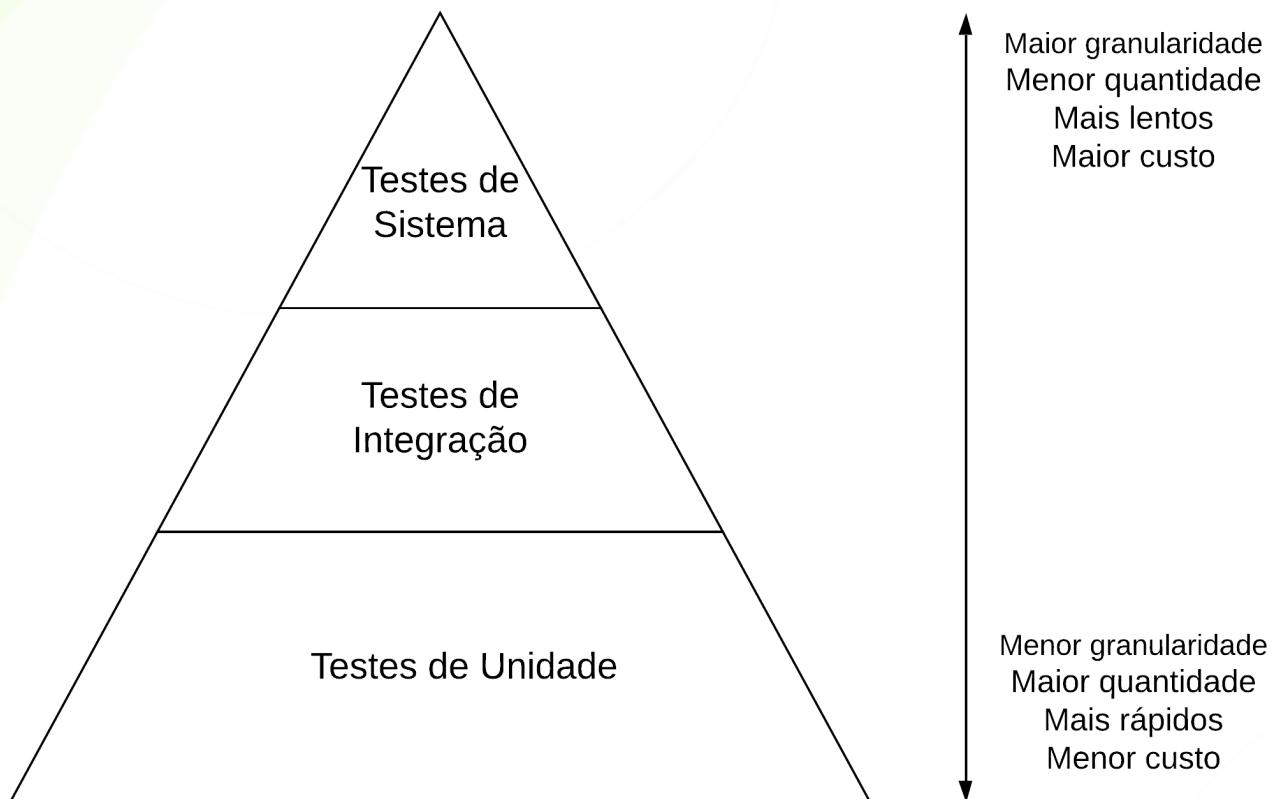


Photo of Ariane 501 Flight, a few seconds after explosion (Credits ESA 1996)

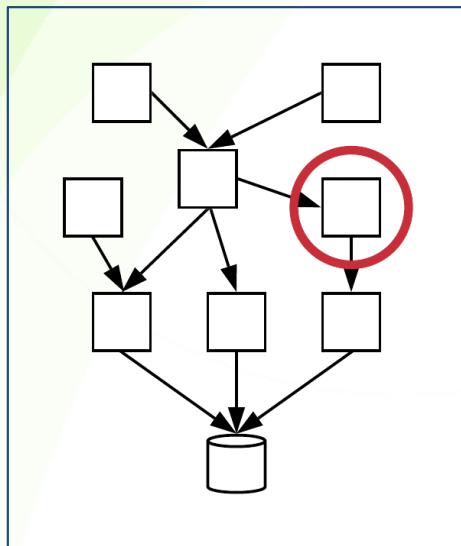
Relatório do Comitê de Investigação

- Explosão foi causada por uma falha de software;
- Conversão de um real de 64 bits para um inteiro de 16 bits;
- Como o real não "cabia" em 16 bits, a conversão falhou;

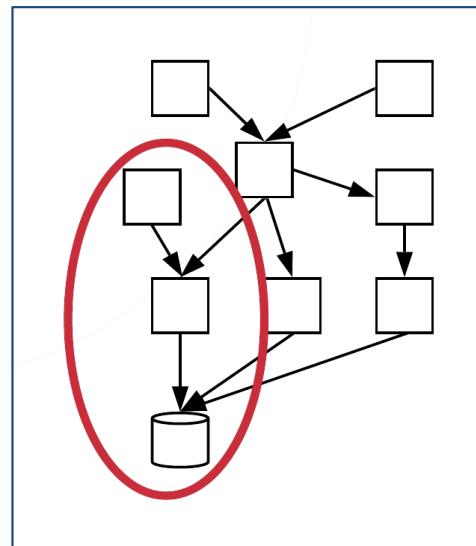
Pirâmide de Testes



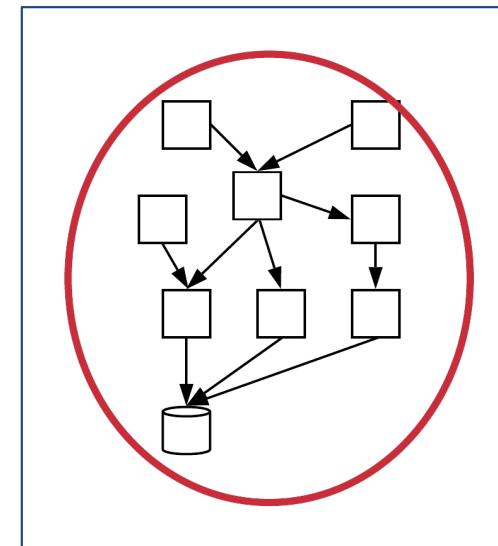
Tipos de Teste



Unidade



Integração

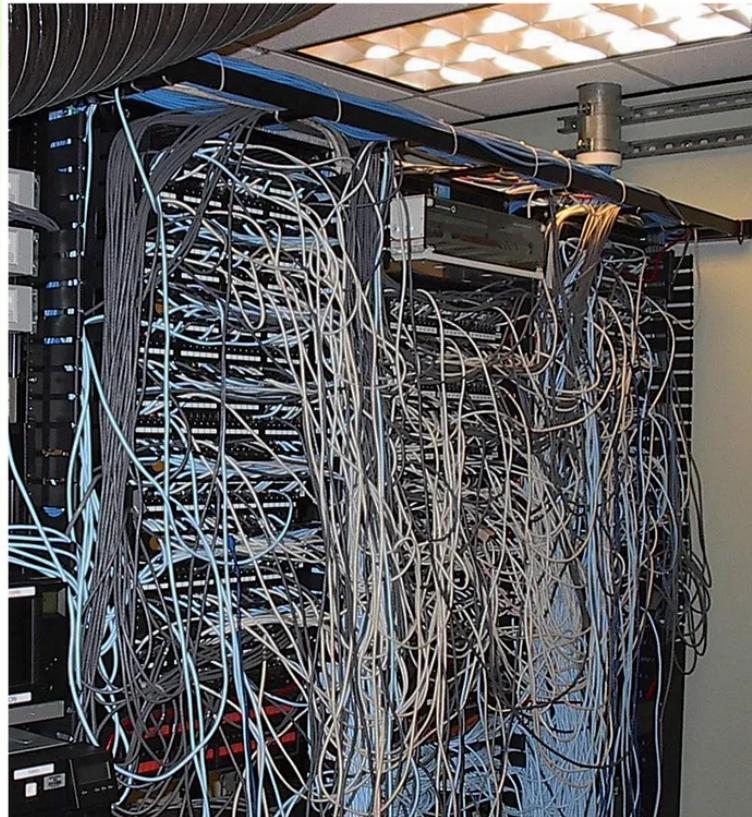


Sistema

Manutenção de Software

- Corretiva
- Preventiva
- Adaptativa
- Evolutiva
- *Refactoring*

Refactoring em 1 slide



Before



After

Software Legado

- “Sistemas de software legado... foram desenvolvidos décadas atrás e têm sido continuamente modificados para se adequar às mudanças dos requisitos de negócio e a plataformas computacionais. A proliferação de tais sistemas está causando dores de cabeça para grandes organizações que os consideram dispendiosos de manter e arriscados de evoluir.” (Dayani-Fard et al., 1999)
- “muitos sistemas legados permanecem dando suporte a funções importantes do negócio e são indispensáveis ao negócio” (Liu e colegas, 1998).
- Ou seja, sistemas legados tem como características a longevidade e criticalidade para o negócio.

Sistemas Legados

- Sistemas antigos, usando linguagens, SOs, BDs antigos;
- Manutenção custosa e arriscada; e
- Muitas vezes, são importantes (legado ≠ irrelevante).

COBOL é muito comum em bancos

- Estima-se que existam ~200 bilhões de LOC em COBOL
- Maioria são sistemas de bancos
 - 95% das transações em ATMs são em COBOL
 - Um único banco europeu tem 250 MLOC em COBOL

Fonte: palestra de Vadim Zaytsev na SLE 2020 (<https://youtu.be/sSkIUTdfDjs>)

Software Legado (2)

Our Government Runs on a 60-Year-Old Coding Language, and Now It's Falling Apart

Retired engineers are coming to the rescue



Dave Gershorn Apr 8, 2020 · 4 min read ★

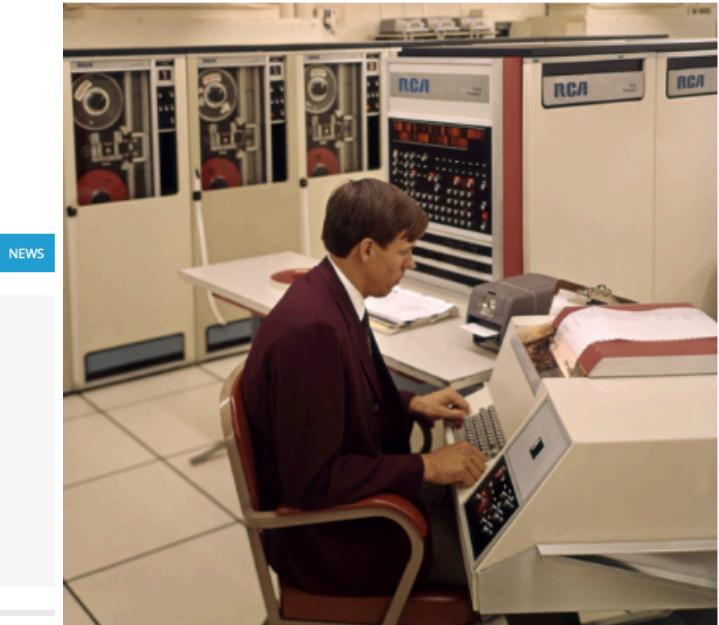


Início » Aplicativos e Software » IBM busca programadores de COBOL após aumento de demanda

IBM busca programadores de COBOL após aumento de demanda

IBM explica que pandemia está levando a "sobrecargas" em sistemas baseados em COBOL

Por Emerson Alecrim
13/04/2020 às 18:32



O **COBOL** é uma linguagem de programação tão antiga — surgiu em 1959 — que, hoje, quase não é lembrada. Mas quem decidiu aprendê-la está vantagem nas circunstâncias atuais: a pandemia de coronavírus (Covid-19) fez a procura por programadores de COBOL aumentar. A **IBM** é uma das companhias que estão buscando esses profissionais.

- PIX começa a ser testado com pagamentos instantâneos simulados
- Python e Java empata em ranking de linguagens de programação

Qual a relação do coronavírus com a linguagem? Basicamente, a pandemia tem levado a um aumento no acesso a sistemas抗igos que foram desenvolvidos em COBOL e que são mantidos assim até hoje em várias partes do mundo.

EM DESTAQUE

- <https://tecnoblog.net/333957/ibm-procura-programadores-cobol-aumento-demanda/>
- <https://onezero-medium-com.cdn.ampproject.org/c/s/onezero.medium.com/amp/p/61ec0bc8e121>

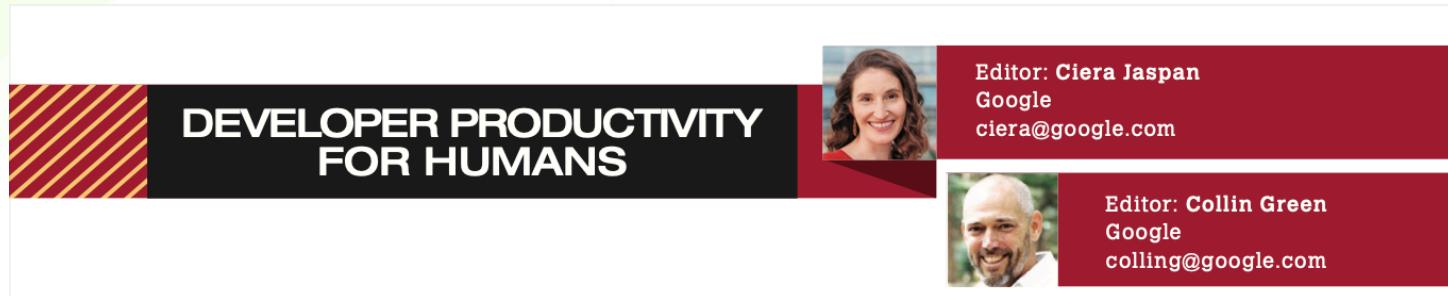


Over the weekend, New Jersey governor, Phil Murphy, made an unusual public plea during his daily coronavirus briefing: The state was seeking volunteer programmers who know COBOL, a 60-year old programming language that the state's unemployment benefits system is built on. Like every state across the nation, New Jersey was being flooded with unemployment claims in the wake of the coronavirus pandemic. And New Jersey's data processing systems were unprepared.

Processos de Desenvolvimento de Software

- Processo de software: define as atividades que devem ser seguidas para construir um sistema de software
- Dois principais modelos:
 - Waterfall ("cascata")
 - Ágil

Desenvolver software é uma atividade colaborativa



The image shows the cover of an article titled "DEVELOPER PRODUCTIVITY FOR HUMANS". The cover features a red and black design with two editor portraits. The top editor, Ciera Jaspan, is a woman with long brown hair, smiling. The bottom editor, Collin Green, is a man with a beard, also smiling. Both portraits are accompanied by their names and affiliations: "Editor: Ciera Jaspan Google ciera@google.com" and "Editor: Collin Green Google colling@google.com".

Software Development Is a Team Sport

Claire Taylor^{ID}, Marie Huber, Qiao Ma, Rayven Plaza, Alison Chang^{ID}, and Jie Chen^{ID}

A WELL-COORDINATED, HIGH-FUNCTIONING team is critical to developing and delivering quality product experiences at a competitive pace. However, the field of software engineering has historically emphasized individuals when studying outcomes like productivity and velocity. Understanding these outcomes at a team or product level requires more than simply aggregating individual-level measures: Teamwork is necessary to avoid counterproductive

but it's difficult to define specifics: Who works with whom? When and how do they interact? What patterns exist across the organization? At a large, global company like Google, the answers to these questions are continuously shifting, as projects begin and end, individuals change roles or join the company, and organizational priorities evolve. We already had existing data sources like management chain (e.g., who reports to

collaborations with other employees to narrow the problem space. We have plans to extend the metric to all Googlers in the future.)

Measuring Collaboration

In reviewing the literature, we saw that surveys,² interviews,³ and other user-centered data provided rich information about collaborations, but they weren't scalable over time. In contrast, logs-based approaches enabled ongo-

Aspectos Éticos

- Engenheiros de Software começam a questionar o uso que as empresas fazem do software desenvolvido por eles

Cybersecurity

Google Engineers Refused to Build Security Tool to Win Military Contracts

A work boycott from the Group of Nine is yet another hurdle to the company's efforts to compete for sensitive government work.

<https://www.bloomberg.com/news/articles/2018-06-21/google-engineers-refused-to-build-security-tool-to-win-military-contracts>

Em 2015, descobriu-se que o software instalado em milhões de carros da Volkswagen comportava-se de forma diferente quando testado em um laboratório de certificação. Nessas situações, o carro emitia poluentes dentro das normas. Fora do laboratório, ele emitia mais poluentes... Ou seja, o código incluía um “if” como o seguinte (meramente ilustrativo). O que você faria se seu chefe pedisse para escrever um if como esse?

```
if "Carro sendo testado em um laboratório"  
    "Emita poluentes dentro das normas"  
else  
    "Emita poluentes fora das normas"
```

Tipos ABC de sistemas

Tipos ABC de sistemas

- Classificação proposta por Bertrand Meyer
- Três tipos de software:
 - Sistemas C (Casuais)
 - Sistemas B (Business)
 - Sistemas A (Acute)

Sistemas C (Casuais)

- Tipo muito comum de Sistema;
- Sistemas pequenos, sem muita importância;
- Podem ter bugs; às vezes, são descartáveis;
- Desenvolvidos por 1-2 engenheiros;
- **Não se beneficiam tanto do que vamos estudar;**
- Risco: "*over-engineering*";

Sistemas B (Business)

- Sistemas importantes para uma organização;
- **Sistemas que se beneficiam do que veremos no curso;**
- Risco: não usarem técnicas de ES e se tornarem um passivo, em vez de um ativo para as organizações.

Sistemas A (Acute)

- Sistemas onde nada pode dar errado, pois o custo é imenso, em termos de vidas humanas e/ou \$\$\$;
- Sistemas de missão crítica;



Metrô



Aviação



Medicina

Sistemas A (Acute)

- Requerem certificações
 - **Fora do escopo do nosso curso**

Document Title	DO-178C - Software Considerations in Airborne Systems and Equipment Certification
Description	This document provides recommendations for the production of software for airborne systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements. Compliance with the objectives of DO-178C is the primary means of obtaining approval of software used in civil aviation products.
Document Number	DO-178C
Format	Hard Copy
Committee	SC-205
Issue Date	12/13/2011

Temas estudados em Engenharia de software

- 1. Engenharia de Requisitos
- 2. Projeto de Software
- 3. Construção de Software
- 4. Testes de Software
- 5. Manutenção de Software
- 6. Gerência de Configuração
- 7. Gerência de Projetos
- 8. Processos de Software
- 9. Modelos de Software
- 10. Qualidade de Software
- 11. Prática Profissional
(Colaboração e Ética)
- 12. Aspectos Econômicos



Qual a implicação disto?

- Existem múltiplos papéis essenciais no processo de desenvolvimento de software. Cada papel exige habilidades diferentes.
- Uma lista ilustrativa (*não exaustiva e não científica*) de papéis, áreas e habilidades:
 - Requisitos → Product Owner → interação com o cliente, negócios
 - Interface com usuário → UI Designer → entender os usuários, psicologia, arte
 - Implementação → dev ou developer → programação++
 - Verificação e validação → eng. de qualidade → atenção a detalhes, programação+
- Todos estes papéis são extremamente desejados!

Indústria de Software

- Atualmente, a enorme indústria de software tornou-se fator dominante nas economias do mundo industrializado.
- Vários governos têm interesse em replicar o Vale do Silício em suas cidades
- Equipes de especialistas em software, cada qual concentrando-se numa parte da tecnologia necessária para distribuir uma aplicação complexa, substituíram o programador solitário de antigamente.

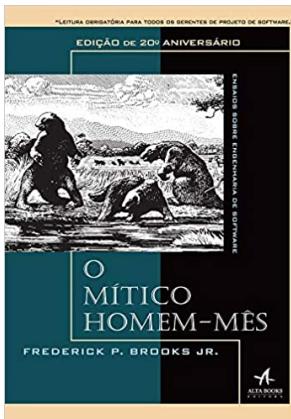
Resumo

- Software
- Dificuldade em desenvolver software
- Crise do Software
- Definição da Engenharia de Software
- Temas em Engenharia de Software
 - Requisitos, Implementação, Manutenção (refatoração, sistemas legados), etc

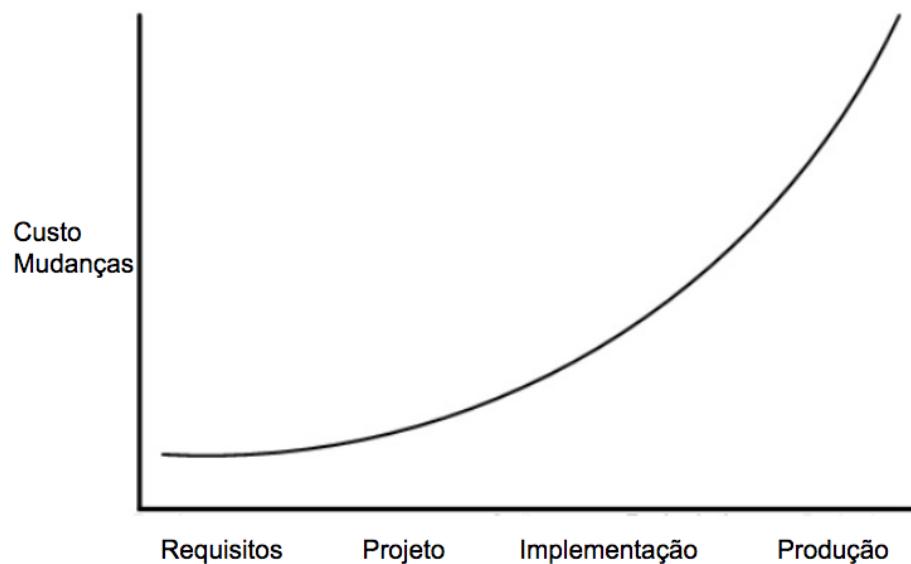
Exercícios

1. Os custos com manutenção podem alcançar 80% dos custos totais alocados a um projeto de software durante o seu ciclo de vida. Por que esse valor é tão alto?
2. Suponha que você tenha que construir uma ponte. Como seria o projeto dessa ponte usando:
 - a. Um método waterfall
 - b. Um método ágil
3. Refactoring é uma transformação de código que preserva comportamento. Qual o significado da expressão preservar comportamento?

5. Em gerência de projetos de software, existe uma lei empírica muito famosa, chamada **Lei de Brooks**, que diz que
- “incluir novos devs em um projeto que está atrasado, vai deixá-lo mais atrasado ainda.”**
- Por que essa lei tende a ser verdadeira?



6. Seja o seguinte gráfico, que mostra — para um sistema — como os custos de mudanças variam conforme a fase do desenvolvimento. Qual método de desenvolvimento você recomendaria para esse sistema?



Fim