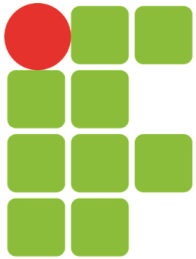


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO SUL
Campus Feliz

Introdução à Engenharia de Software

Profª. Dra. Ana Paula Lemke

Contextualização

Desenvolver software (seguindo os preceitos da Engenharia de Software) é uma atividade que não deve ser confundida com escrever programas para o computador.*

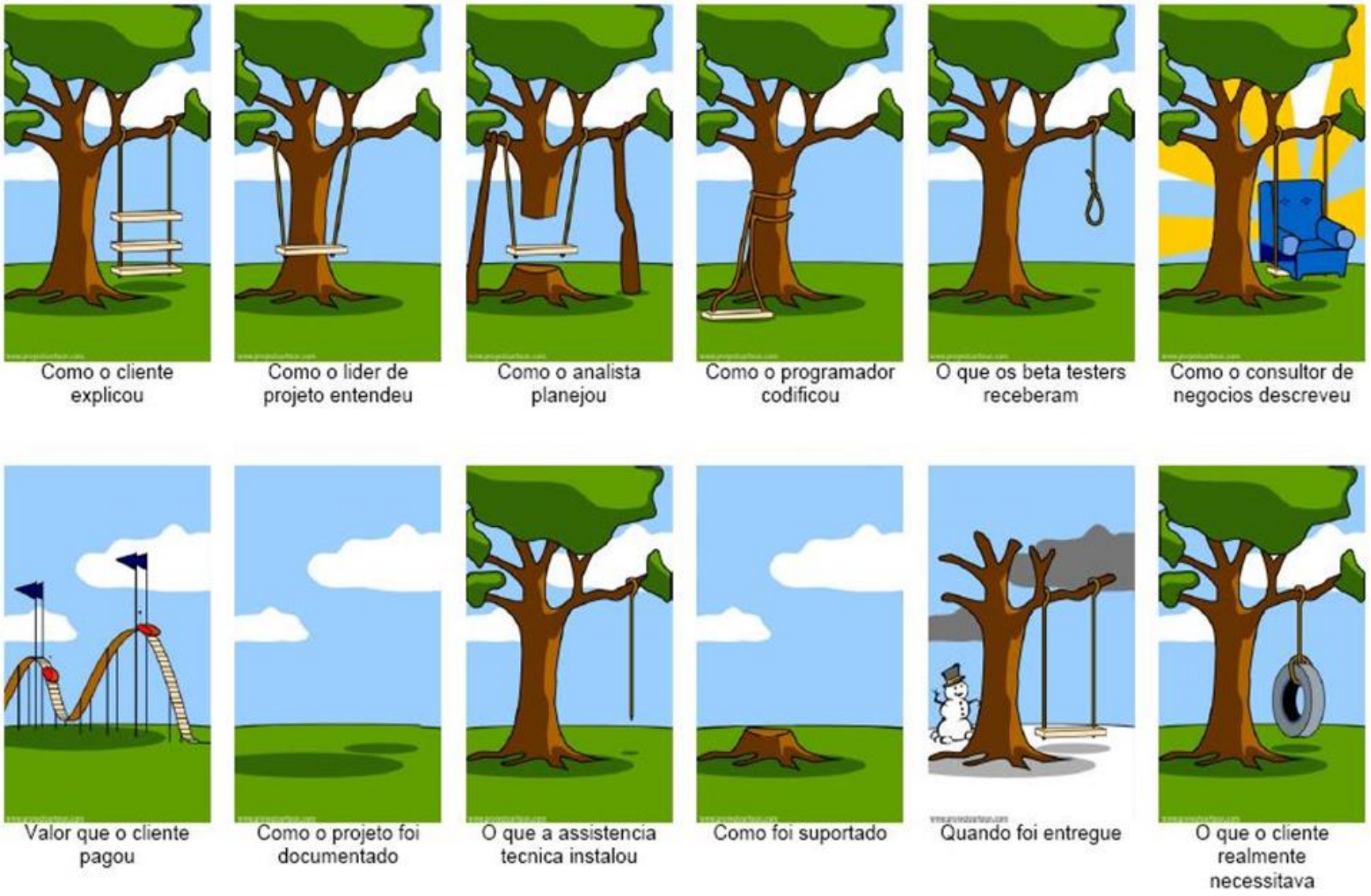
**Adaptado do livro "Engenharia de Software e Sistemas de Informação", escrito por Denis Alcides Rezende, 2005.*

Contextualização

- Necessidade do usuário:

Eu quero um balanço que seja feito de corda e que seja amarrado num galho da árvore, e que eu possa ficar sentado no meio dele.

Contextualização



Contextualização

- O que é software?
  - Programa de computador + documentação associada
  - Podem ser desenvolvidos para um cliente específico ou para o público em geral.
- O que é a Engenharia de Software (ES)?
  - Disciplina de engenharia que se preocupa com todos os aspectos da produção de software.
- Quais são as principais atividades da ES?
  - Especificação, desenvolvimento, validação e evolução de software.

Fonte: Sommerville, I. "Engenharia de Software", 9º ed, 2011 (pág. 4).

Contextualização

- Qual a diferença entre ES e Ciência da Computação?
  - Ciência da Computação foca a teoria e os fundamentos;
  - ES preocupa-se com o lado prático do desenvolvimento e entrega de softwares úteis.
- Qual a diferença entre ES e Engenharia de Sistemas?
  - Engenharia de Sistemas preocupa-se com todos os aspectos do desenvolvimento de sistemas computacionais (inclui engenharia de hardware, software e processo).
- Quais são os principais desafios da ES?
  - Aumento de diversidade, diminuição do tempo para entrega e desenvolvimento de software confiável.

Fonte: Sommervile, I. "Engenharia de Software", 9º ed, 2011 (pág. 4).

Contextualização

- Quais as dificuldades em se construir um software?
- Quais os **problemas** mais comuns?
- O que faz um analista de sistemas?
- Como podemos entender o que o cliente quer?
- Como validar/verificar o que o cliente quer?

- Difícil manutenção
- Documentação pobre
- Problemas na qualidade
- Custo e prazos sempre ultrapassam
- Não adequação as necessidades do usuário

Contextualização

- Quais as dificuldades em se construir um software?
- Quais os problemas mais comuns?
- O que faz um analista de sistemas?
- Como podemos entender o que o cliente quer?
- Como validar/verificar o que o cliente quer?

Crise do Software

- Desenvolvimento de software é um contínuo estado de crise:
  - Dificuldade de escrever, compreender e verificar programas computacionais.
  - Projetos ultrapassam orçamentos e cronograma.
  - Atender requisitos e satisfazer usuário (qualidade).
- Geralmente, projetos de software são grandes, complexos, mal especificados, relacionam aspectos não familiares e são vulneráveis a problemas não previstos.
- "*Software é hard*" Donald Knuth
  - Não existe bala de prata, isto é, uma única abordagem para prevenir casos de falha.

Crise do Software

- Problemas recorrentes enfrentados no processo de desenvolvimento (construção, implantação e manutenção) do software:
  - **25%** dos projetos são cancelados.
  - O tempo de desenvolvimento é bem maior do que o estimado.
  - **75%** dos sistemas não funcionam como planejado.
  - Manutenção e reutilização são difíceis e custosas.
  - Os problemas são proporcionais a complexidade dos sistemas.

Mitos e Realidades

Mito de Administrador: “Temos um manual repleto de padrões e procedimentos para a construção de software. Isso oferece ao meu pessoal tudo o que eles precisam saber.

Realidade: Será que o manual é usado? Os profissionais sabem que ele existe? Ele reflete o estado da arte no desenvolvimento de software? Ele é completo?

Mitos e Realidades

Mito de Administrador: “Meu pessoal tem ferramentas de desenvolvimento de software de última geração: compramos os mais novos computadores no mês passado!”.

Realidade: É preciso muito mais do que os mais recentes computadores para se desenvolver software com qualidade.

Mitos e Realidades

Mito de Administrador: “Quando estamos atrasados nos prazos, é só adicionar mais programadores para tirar o atraso”.

Realidade: O desenvolvimento de software não é um processo mecânico igual à manufatura. Acrescentar pessoas em um projeto torna-o ainda mais atrasado.

Pessoas podem ser acrescentadas, mas somente de uma forma planejada.

Mitos e Realidades

Mito de Administrador: “Se eu decidir terceirizar o projeto de software, posso simplesmente relaxar e deixar essa empresa realizá-lo”.

Realidade: Se uma empresa não souber gerenciar e controlar projetos de software , ela irá, invariavelmente, enfrentar dificuldades ao terceirizá-los.

Mitos e Realidades

Mito de Cliente: “Uma declaração geral dos objetivos é suficiente para se começar a escrever programas - podemos preencher os detalhes mais tarde”.

Realidade: Um entendimento inicial ruim do software é uma das causas de fracassos no desenvolvimento de software.

Mitos e Realidades

Mito de Cliente: “Posso modificar os requisitos de projeto continuamente, uma vez que qualquer mudança pode ser facilmente acomodada, afinal, o software é flexível”.

Realidade: Uma mudança, quando solicitada tardiamente num projeto, pode causar um grande impacto no desenvolvimento e nas partes já desenvolvidas do sistema.

Mitos e Realidades

Mito de Profissional: “Assim que escrevermos o programa e o colocarmos em funcionamento nosso trabalho estará completo!”.

Realidade: Os dados da indústria indicam que entre 60 e 80% de todo esforço gasto num sistema serão despendidos depois que ele for entregue pela primeira vez ao cliente.

Mitos e Realidades

Mito de Profissional: “Enquanto o programa não estiver “rodando”, não é possível avaliar a sua qualidade”.

Realidade: A revisão técnica formal é um dos mecanismos mais efetivos de qualidade do software e pode ser aplicada desde o começo de um projeto.

Mitos e Realidades

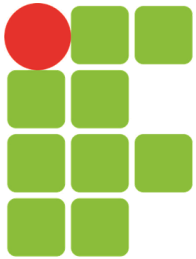
Mito de Profissional: “O único produto passível de entrega é o programa em funcionamento”.

Realidade: Um programa em funcionamento é somente uma parte de uma configuração de software que inclui muitos elementos (modelos, documentos, planos).

Mitos e Realidades

Mito de Profissional: “A Engenharia de Software nos fará criar documentação volumosa e desnecessária e, invariavelmente, irá nos retardar”.

Realidade: A Engenharia de Software não trata de criação de documentos, trata da **criação de um produto de qualidade**. Melhor qualidade conduz à redução do retrabalho, e menos retrabalho resulta em maior rapidez na entrega.

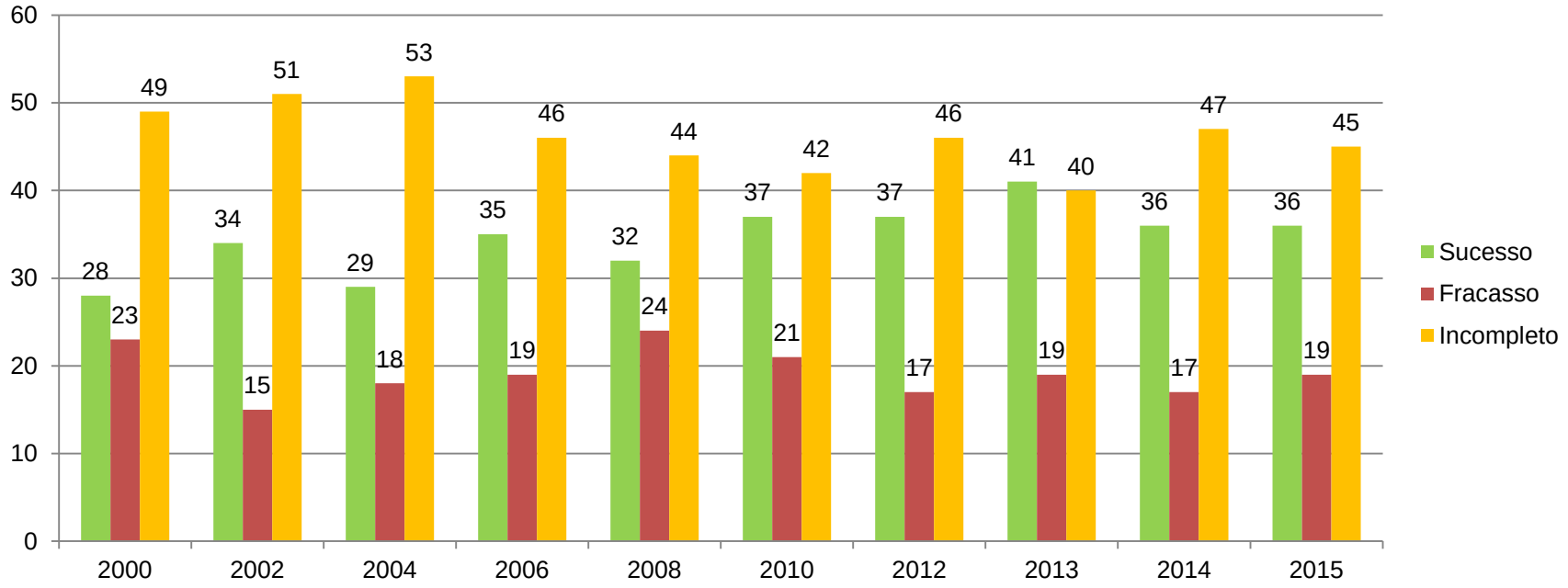


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO SUL
Campus Feliz

Evidências da Crise de Software

Chaos Report Tradicional

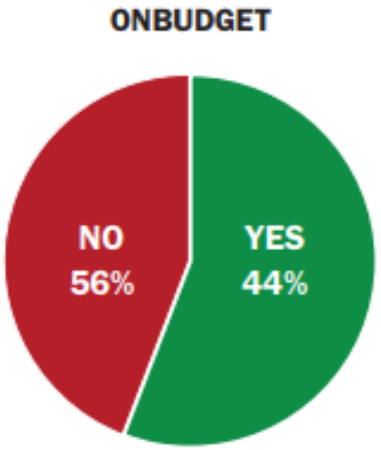
THE STANDISH GROUP
CHAOS REPORT



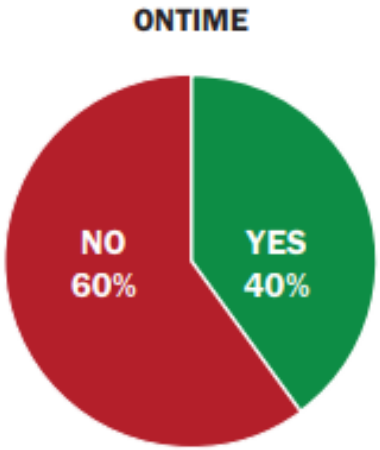
Sucesso: quando o projeto foi entregue dentro do prazo e do custo contendo todas as características e funções requeridas (ou seja, **OnTime**, **OnBudget** e **OnTarget**).
Fracasso/Cancelamento: quando o projeto é cancelado durante o seu desenvolvimento ou foi entregue mas nunca usado.
Incompleto/deficiente: quando o projeto não foi entregue no prazo, custou mais do que o previsto ou não contempla as características e funções requeridas inicialmente.

Fontes: CHAOS MANIFESTO 2013 e 2015 (https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf).

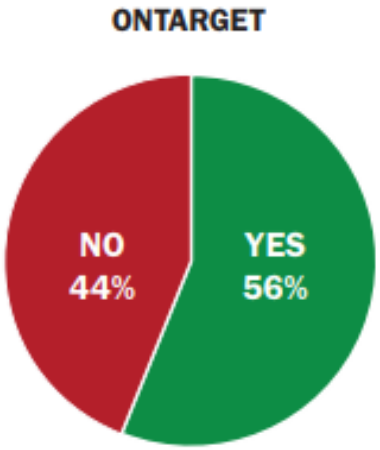
Chaos Report Tradicional



The percentage of projects that were OnBudget from FY2011–2015 within the new CHAOS database.



The percentage of projects that were OnTime from FY2011–2015 within the new CHAOS database.

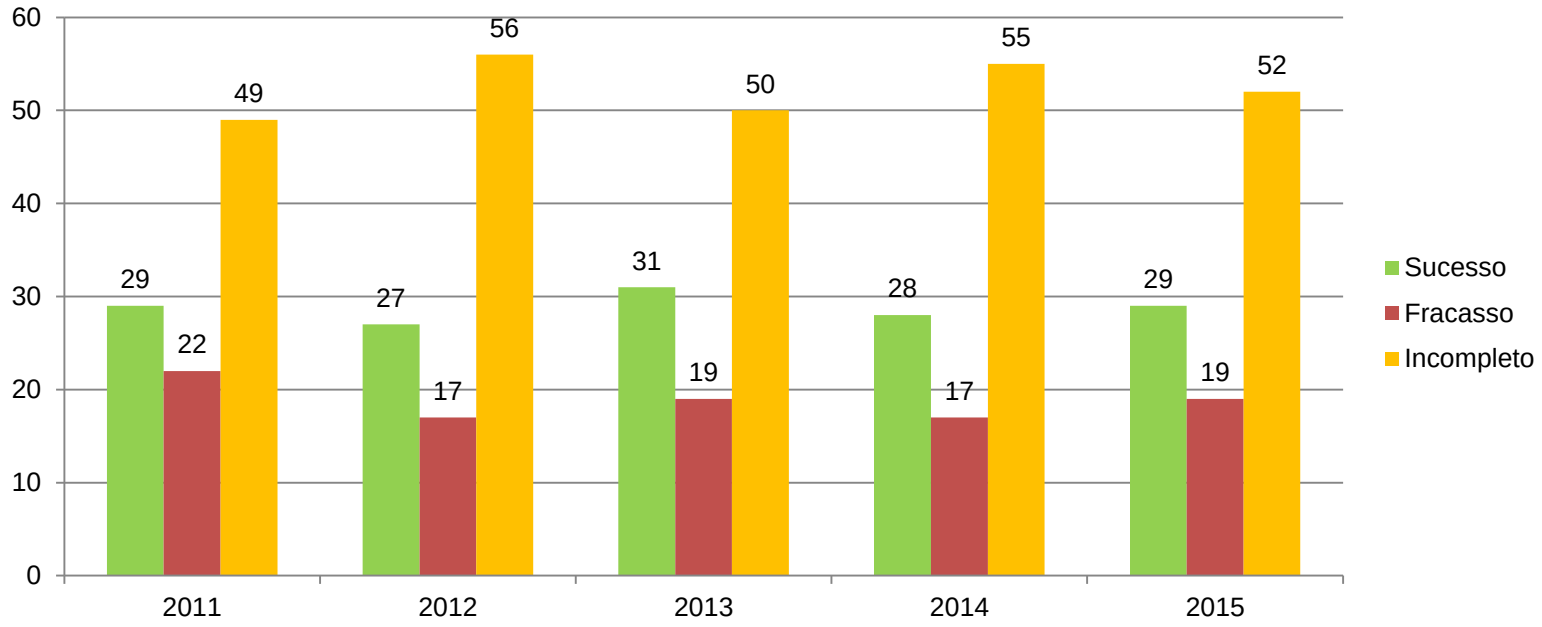


The percentage of projects that were OnTarget from FY2011–2015 within the new CHAOS database.

Fonte: gráficos extraídos do documento CHAOS Report 2015.
https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Chaos Report "Moderno"

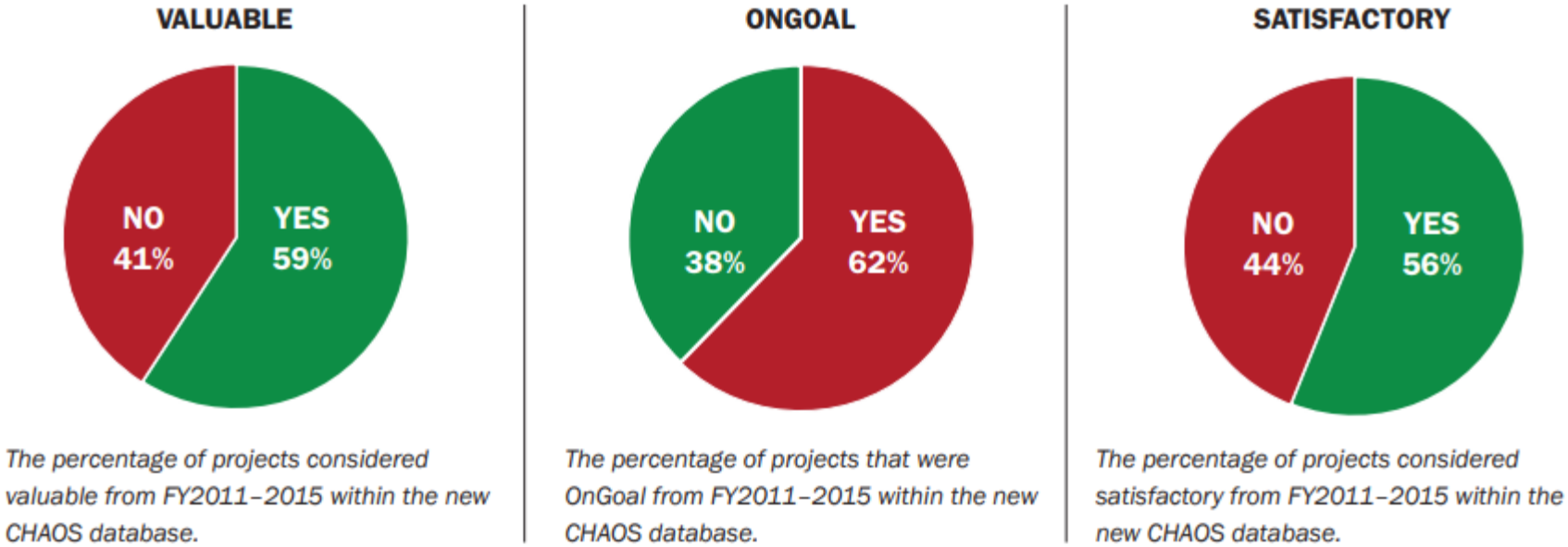
THE STANDISH GROUP
CHAOS REPORT



Sucesso: quando o projeto foi entregue dentro do prazo e do custo contendo todas as características e funções requeridas e satisfaz os clientes/investidores (ou seja, **OnTime**, **OnBudget** e **Valuable**, **Ongoal** e **Satisfactory**).
Fracasso/Cancelamento: quando o projeto é cancelado durante o seu desenvolvimento ou foi entregue mas nunca usado.
Incompleto/deficiente: quando o projeto não foi entregue no prazo, custou mais do que o previsto ou não contempla as características e funções requeridas inicialmente.

Fonte: CHAOS Report 2015 (https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf).

Chaos Report "Moderno"



The percentage of projects considered valuable from FY2011–2015 within the new CHAOS database.

The percentage of projects that were OnGoal from FY2011–2015 within the new CHAOS database.

The percentage of projects considered satisfactory from FY2011–2015 within the new CHAOS database.

Fonte: gráficos extraídos do documento CHAOS Report 2015.
https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Chaos Report "Moderno"

PROJECT SIZE BY CHAOS RESOLUTION

	SUCCESSFUL	CHALLENGED	FAILED	TOTAL
Grand	6%	51%	43%	100%
Large	11%	59%	30%	100%
Medium	12%	62%	26%	100%
Moderate	24%	64%	12%	100%
Small	61%	32%	7%	100%

The size of software projects by the Modern Resolution definition from FY2011–2015 within the new CHAOS database.

Fonte: gráficos extraídos do documento CHAOS Report 2015. https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Chaos Report "Moderno"

SIZE-COMPLEXITY MATRIX

		COMPLEXITY				
		Very Simple	Simple	Average Complexity	Complex	Very Complex
SIZE	Small	100	250	400	550	625
	Moderate	175	325	475	625	775
	Medium	250	400	550	700	925
	Large	325	475	625	775	925
	Grand	400	550	700	850	1000

The Size-Complexity Matrix provides guidelines for categorizing a project in order to assess the risk and effort. The Size-Complexity Matrix has many uses. It uses a 5-point scale for both size and complexity. The lowest-point project is a simple, small project and would have 100 points. The largest and most complex project would have 1,000 points. Green means low risk and effort, yellow means medium risk and effort, and red means high risk and effort.

SIZE GUIDELINES

Size Description		Size
Under $1 million labor	6 or fewer team members/months	Small
$1 million to $3 million	7 to 12 team members/months	Medium
$3 million to $6 million	13 to 24 team members/months	Medium
$6 million to $10 million	25 to 50 team members/months	Large
Over $10 million	Over 50 team members/months	Grand

The table shows the guidelines on how to measure the size of a project.

COMPLEXITY GUIDELINES

Environment	Points
Diverse User Base	1
Multiple Team Locations	1
Multiple Stakeholder Locations	1
Uncooperative Peers	2
Uncooperative Stakeholders	3
Scope	**Points**
Many Requirements-Large Scope	1
Ambiguous scope	1
Fuzzy Undefined Requirements	1
Diverse and Multifaceted Objectives	2
Breaking New Ground	3

The table shows the guidelines on how to measure the complexity of a project.

Fontes:
https://www.infoq.com/articles/standish-chaos-2015/
https://www.standishgroup.com/TimeCalculating

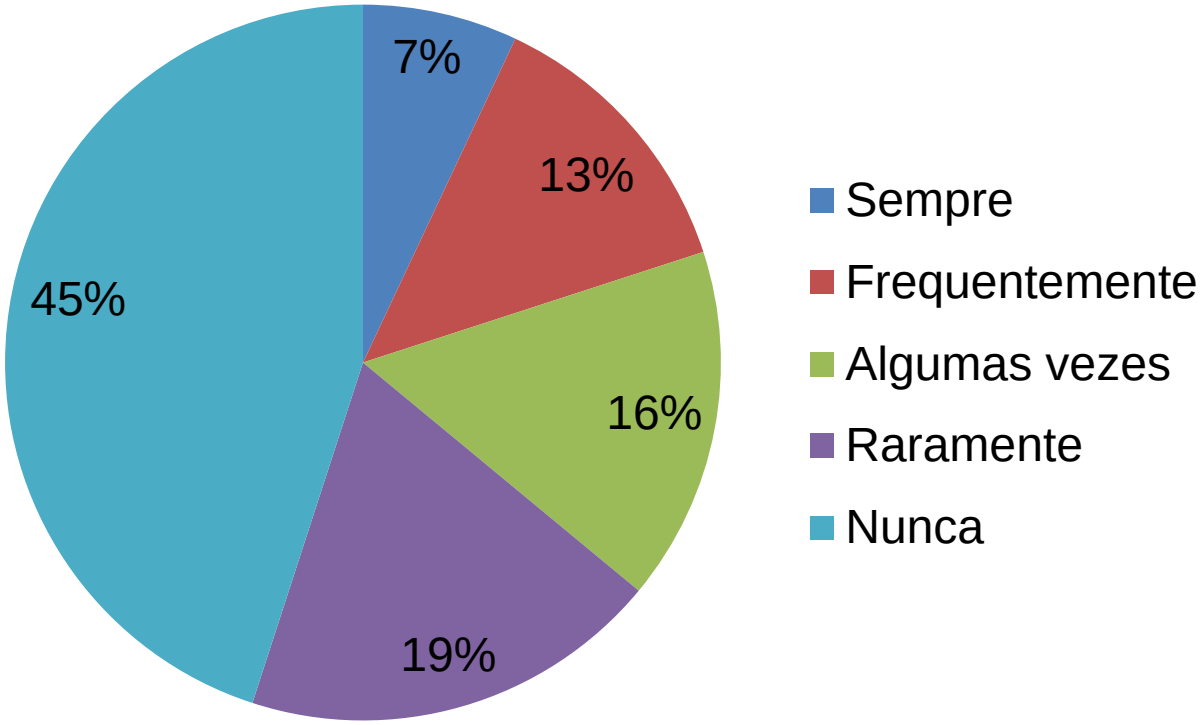
Chaos Report "Moderno"

CHAOS FACTORS OF SUCCESS

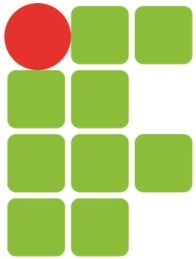
FACTORS OF SUCCESS	POINTS	INVESTMENT
Executive Sponsorship	15	15%
Emotional Maturity	15	15%
User Involvement	15	15%
Optimization	15	15%
Skilled Resources	10	10%
Standard Architecture	8	8%
Agile Process	7	7%
Modest Execution	6	6%
Project Management Expertise	5	5%
Clear Business Objectives	4	4%

The 2015 Factors of Success. This chart reflects our opinion of the importance of each attribute and our recommendation of the amount of effort and investment that should be considered to improve project success.

Uso das Funcionalidades do Software



Baseado em http://www.martinfowler.com/articles/xp2002.html e http://www.featuredrivendevelopment.com/node/614.



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO SUL
Campus Feliz

Esforços no desenvolvimento de Software

Desenvolvimento e Manutenção de Software

- Desenvolvimento (**20%**):
  - Início: quando a necessidade do produto é identificada;
  - Fim: quando o teste do produto implantado é concluído e o produto é entregue para a operação/produção.
- Manutenção (**80%**):
  - Todas as atividades após a entrega:
    - Aumento da capacidade do produto (60%)
    - Adaptação do produto a novos ambientes (20%)
    - Correção de erros (20%)

Desenvolvimento e Manutenção de Software

- Por exemplo, uma pesquisa de Fjeldstad e Hamlen (1979) verificou que 25 empresas gastavam:
  - 31% do seu esforço em desenvolvimento
  - 61% em manutenção (incluindo suporte a usuário).
- Outra pesquisa (Parikh e Zvegintzov, 1983) verificou que a manutenção era cerca de 3 vezes maior que o desenvolvimento.
- Recentemente, as pesquisas estimam que 20% do esforço das empresas é gasto em desenvolvimento e 80% em manutenção.

Desenvolvimento e Manutenção de Software

- Manutenção é cara e inevitável, pois:
  - Há software legado (mais de 5 anos de uso);
  - Implantação é incompleta, necessitando ajustes;
  - Mudanças nos sistemas, refletindo a mudança de negócio da empresa.
- Como atenuar o problema?
  - Melhoria da qualidade do software
  - **Melhoria do processo de produção de SW**
- Ambas associadas à uma *mudança de cultura* no desenvolvimento do software

Mudança de Cultura de Desenvolvimento de SW

- Barateamento do *hardware*;
- Fazer software ainda é atividade intelectual:
  - Mão de obra intensiva ($);
  - Baseada em experiência;
  - Baixos níveis de reuso e automação.
- Usuários mais exigentes: qualidade, escopo e prazo;
- Software mais complexo: metodologias tradicionais já não são suficientes.

Mudança de Cultura de Desenvolvimento de SW

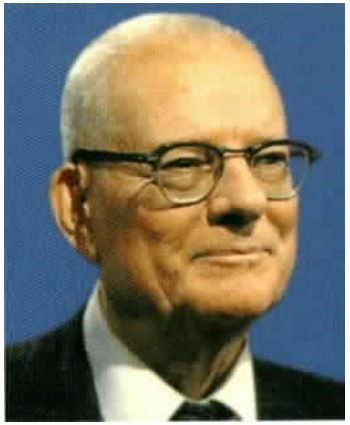
- Como atenuar a crise do software?
  - Disciplina para desenvolvimento (Engenharia de Software).
    - Combina métodos e ferramentas adequados ao processo de desenvolvimento;
    - Utiliza técnicas para a garantia da qualidade;
    - Aplica uma filosofia de coordenação, controle e administração.

Conclusão

Programar não é suficiente!

“Você sabe que fazer o melhor que pode não é o suficiente? Você deve primeiro saber o que precisa ser feito. Então deve fazer o melhor”.

W. Edwards Deming



O que é Engenharia de Software?

- O conceito de Engenharia de Software foi inicialmente proposto em 1968 (Sommerville, 2007).

- Definição amplamente utilizada:

*"Abrange um conjunto de três elementos fundamentais - **métodos**, **ferramentas** e **procedimentos** – que possibilita ao gerente controlar o processo de desenvolvimento de software e oferece ao profissional uma base para a construção produtiva de software de alta qualidade"* (Pressman, 1995).

O que é Engenharia de Software?

- Métodos
  - Proporcionam os detalhes de “como fazer” para construir o software. Envolvem um amplo conjunto de tarefas que incluem: planejamento, estimativas de projeto, análise de requisitos, etc.
- Ferramentas
  - Proporcionam apoio automatizado ou semi-automatizado aos métodos.
- Procedimentos
  - Definem a sequência em que os métodos serão aplicados, os produtos que serão disponibilizados, como será feito o controle de qualidade, entre outros.

O que é Engenharia de Software?

"É uma disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até a sua manutenção, quando o sistema já está sendo usado" *(Sommerville, 2011, pág. 5).*

O que é Engenharia de Software?

"*Engenharia de Software: (1) A aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software; isto é, a aplicação de engenharia ao software. (2) O estudo de abordagens como definido em (1).*" (Pressman, 2011 apud **IEEE**, página 39).

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.
  - Geralmente é mais barato, a longo prazo, usar métodos e técnicas da ES para sistemas de software, em vez de escrever os programas como se fossem algum projeto pessoal. **Para a maioria dos sistemas, a maior parte do custo é mudar o software depois que ele começa a ser usado**. (Sommerville, 2011).

Objetivos da Engenharia de Software

- O que é um software de qualidade?
  - O software que satisfaz os requisitos solicitados pelo usuário. Deve ser fácil de manter, ter boa *performance*, ser confiável e fácil de usar.
- O que é produtividade?
  - Custo de desenvolvimento reduzido:
    - A empresa consumidora quer investir pouco em software.
    - A empresa produtora tem que oferecer “software barato”.
  - Tempo de desenvolvimento reduzido:
    - Suporte rápido às necessidades do mercado.

Importância da Engenharia de Software

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

Importância da Engenharia de Software

Erros de software tem grande impacto?

Importância da Engenharia de Software

- À medida que o software invade todos os setores, riscos ao público (devido a programas com imperfeições) passam a ser uma preocupação cada vez maior.
- **Questão para discussão**: Crie um cenário o mais catastrófico possível, porém realista, cuja falha de um programa de computador poderia causar um grande dano (em termos econômico ou humano).

Importância da Engenharia de Software

- **1985**: Máquina Medicinal mata
  - **Custo:** Três mortos e três seriamente feridos
  - **Desastre:** A máquina de radiação canadense Therac-25 irradiou doses letais em pacientes.
  - **Causa:** Por causa de um bug sutil chamado de "*condição de corrida*", um técnico acidentalmente configurou o Therac-25 de modo que o feixe de elétrons seria como um fogo de alta potência.

- **2000**: Tratamento contra Câncer mortal
  - **Custo:** 8 pessoas mortas e 20 seriamente feridas
  - **Desastre:** O software de radiação calculou mal a dosagem de radiação que deveria ser enviada, expondo pacientes a níveis fatais de radiação. Os físicos que foram indicados para checar as máquinas foram condenados a morte.
  - **Causa:** O software calculava a dosagem de radiação baseando-se na ordem de entrada dos dados, e algumas vezes enviava o dobro da dose do que deveria.

Fonte: http://www.devtopics.com/20-famous-software-disasters/

Importância da Engenharia de Software

- **1990**: Linhas da AT&T "morrem"
  - **Custo:** 75 milhões de ligações perdidas e 200 reservas aéreas perdidas
  - **Desastre:** Um *switch* dos 114 centros de *switches* da AT&T sofreu um problema mecânico que fez com que todo o seu centro fosse desligado.
  - **Causa:** Uma única linha de código em uma atualização de software implementada para acelerar chamadas causou um efeito cascata que desligou a rede.

Fonte: http://www.devtopics.com/20-famous-software-disasters/

Importância da Engenharia de Software

- **1991**: *Patriot* acaba com Soldados
  - **Custo:** 28 soldados mortos e 100 feridos.
  - **Desastre:** Durante a primeira Guerra do Golfo, um sistema (*Patriot*) americano de mísseis na Arábia Saudita falhou ao interceptar um míssil vindo do Iraque. O míssil destruiu acampamentos americanos.
  - **Causa:** Um erro de arredondamento no software calculou incorretamente o tempo, fazendo com que o sistema *Patriot* ignorasse os mísseis Scud de entrada.

Fonte: http://www.devtopics.com/20-famous-software-disasters/

Importância da Engenharia de Software

- **1993**: Pentium falha em uma operação de divisão
  - **Custo:** $475 milhões e a credibilidade de uma empresa
  - **Desastre:** O Pentium cometeu erros ao dividir números de ponto flutuante em um intervalo específico. Por exemplo, dividindo 4195835.0/3145727.0 obtinha 1,33374 ao invés de 1,33382, um erro de 0,006%. Com uma estimativa de 5 milhões de chips defeituosos em circulação, a Intel se ofereceu para substituir os chips Pentium apenas para os consumidores que poderiam provar que eles precisavam de alta precisão. Contudo a Intel acabou substituindo os chips de qualquer um que reclamou.
  - **Causa:** O divisor na unidade de ponto flutuante do Pentium tinha uma tabela de divisão falha, faltando cerca de cinco mil entradas, resultando nos erros de arredondamento.

Fonte: http://www.devtopics.com/20-famous-software-disasters/

Importância da Engenharia de Software

- **1996**: Desintegração do foguete Ariane
  - **Custo:** $500 milhões
  - **Desastre:** Ariane 5 foi intencionalmente destruído segundos após seu lançamento em seu voo inaugural. Também foram destruídos quatro satélites científicos para estudar como o campo magnético da Terra interage com os ventos solares.
  - **Causa:** O desligamento ocorreu quando o computador de orientação tentou converter a velocidade do foguete de 64 bits para um formato de 16 bits. O número era muito grande, o que resultou em erro de estouro (*overflow*). Quando o sistema de orientação desligou, o controle passou para uma unidade idêntica redundante, que também falhou porque nele estava correndo o mesmo algoritmo.

    Vídeo: https://youtu.be/kYUrqdUyEpI
    Mais informações: http://www.ime.uerj.br/~demoura/Especializ/Ariane/

Importância da Engenharia de Software

- **1999**: Passaportes britânicos para lugar nenhum
  - **Custo:** £12.6 milhões
  - **Desastre:** A agência de passaportes do Reino Unido implementou um sistema da Siemens que falhou ao emitir documentos para meio milhão de cidadãos britânicos. A agência teve que pagar milhões ao governo para compensar a raiva da população.
  - **Causa:** A Agência lançou seu novo sistema sem testá-lo de forma adequada ou treinar seus funcionários. Ao mesmo tempo, uma mudança na lei exigia que todos os menores de 16 anos viajando ao exterior deveriam obter um passaporte, resultando em um aumento enorme na procura de passaportes, o que sobrecarregou o sistema.

Fonte: http://www.devtopics.com/20-famous-software-disasters/

Importância da Engenharia de Software

- **2017**: Exclusão de dados de processos no Tribunal de Contas do Estado do Amazonas
  - **Custo:** R$ 33 mil, pagos pelos dois funcionários que cometeram o erro
  - **Desastre*:** exclusão de dados de dois softwares, atingindo 16,5 mil processos (80% de processos de aposentadoria). *Os dados foram recuperados.
  - **Causa:** execução indevida de um script (sequência de comados) que um administrador de sistemas rodou sem fazer a checagem. De acordo com superiores, o problema ocorreu por “excesso de confiança e falta de responsabilidade” por parte dos dois servidores. Um que mexia diretamente com o PostgreSQL, que alimentava os sistemas, e outro por saber das limitações do backup e da necessidade de ampliação do mesmo e não tomar as providências necessárias como gestor.

Fonte: https://www.cbsi.net.br/2017/10/profissional-de-ti-erra-script-e-apaga-16-mil-processos-do-banco-de-dados-doTCE-AM.html/

Atividade

- Você já ouviu falar do vírus **Stuxnet**? Clique aqui para ler a história.

Diversidade na Engenharia de Software

- A forma como a ES é aplicada varia dramaticamente de acordo com:
  - A organização que está desenvolvendo o software;
  - O tipo de software que será desenvolvido;
  - As pessoas envolvidas no processo de desenvolvimento.

Fonte: Sommervile, I. “Engenharia de Software”, 9º ed, 2011 (pp 6-7).

Diversidade na Engenharia de Software

- Exemplos de tipos de aplicações de acordo com Sommerville (pág. 7):
  - Aplicações *stand-alone*
  - Aplicações interativas baseadas em transações
  - Sistemas de controle embutidos
  - Sistemas de processamento em lotes
  - Sistemas de entretenimento
  - Sistemas para modelagem e simulação
  - Sistemas de coleta de dados
  - Sistemas de sistemas

Diversidade na Engenharia de Software

- Categorias de Software de acordo com Pressman (pp. 34-35):
  - Software de sistema
  - Software de aplicação
  - Software científico/de engenharia
  - Software embutido
  - Software para linha de produtos
  - Aplicações para Web (*Webapps*)
  - Software de Inteligência Artificial

Atividade

- **Ler** as páginas indicadas dos livros do Pressman e Sommerville e **fazer o mapeamento** entre os tipos/categorias de software propostos por cada autor.
- **Indicar exemplos** de software para cada categoria.

Diversidade na Engenharia de Software

- Fundamentos da ES que se aplicam a todos os tipos de sistemas:
  - Usar um processo gerenciado e compreendido (saber o que será produzido e quando estará finalizado).
  - Desenvolver software confiável e com bom desempenho.
  - Entender e gerenciar a especificação e os requisitos de software.
  - Fazer o melhor uso possível dos recursos existentes.

Fonte: Sommervile, I. "Engenharia de Software", 9º ed, 2011 (pág. 8).

Diversidade na Engenharia de Software

O caso das WebApps

- Sistemas e aplicações baseadas na Web "*envolvem uma mistura de publicação impressa e desenvolvimento de software, de marketing e computação, de comunicações internas e relações externas e de arte e tecnologia*".
- Atributos encontrados na grande maioria das WebApps:
  - Uso intensivo de redes;
  - Simultaneidade (muitos usuários acessando ao mesmo tempo);
  - Carga não previsível;
  - Desempenho (ninguém espera muito);
  - Disponibilidade;
  - Orientadas a dados;
  - Sensibilidade no conteúdo (estética de apresentação);
  - Evolução contínua;
  - Imediatismo (para disponibilização no mercado);
  - Segurança;
  - Estética.

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pág. 37).

Bibliografia

- Fontes utilizadas para a confecção desse material:
  - PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. Rio de Janeiro: McGraw-Hill, 2011.
  - SOMMERVILLE, I. Engenharia de Software. 9 ed. Addison-Wesley, 2011.