Teste de Software

Gilmar Ferreira Arantes
- gilmar.arantes@goias.gov.br

GPS - SCTI

Goiânia, 24 e 25 de Junho de 2019

Sumário I

Quem Sou Eu...

1. Nome: Gilmar Ferreira Arantes
2. Gradução - Análise de Sistemas - Universo - 1999-2003;
3. MBA Gestão de TI - Alfa - 2004-2006;
4. Especialista em Análise e Projeto de Software - UFG - 2006-2007;
5. Mestre em Ciência da Computação - UFG - 2009-2012;
6. Gestor de Tecnologia da Informação - SEDI - 01/02/2007;
7. Professor Assistente - INF/UFG - 28/11/2013.
8. Emails: gilmar.arantes@goias.gov.br, gilmar.arantes@ufg.br e gilmar.arantes@gmail.com

Superintendência de Integração Tecnológica da Informação | Secretaria de Estado da Educação | Somos todos GOIÁS GOVERNO DO ESTADO

E Vocês?

1. Nome.
2. Atividades na Seduc.
3. Experiência com Testes.

Filosofando...

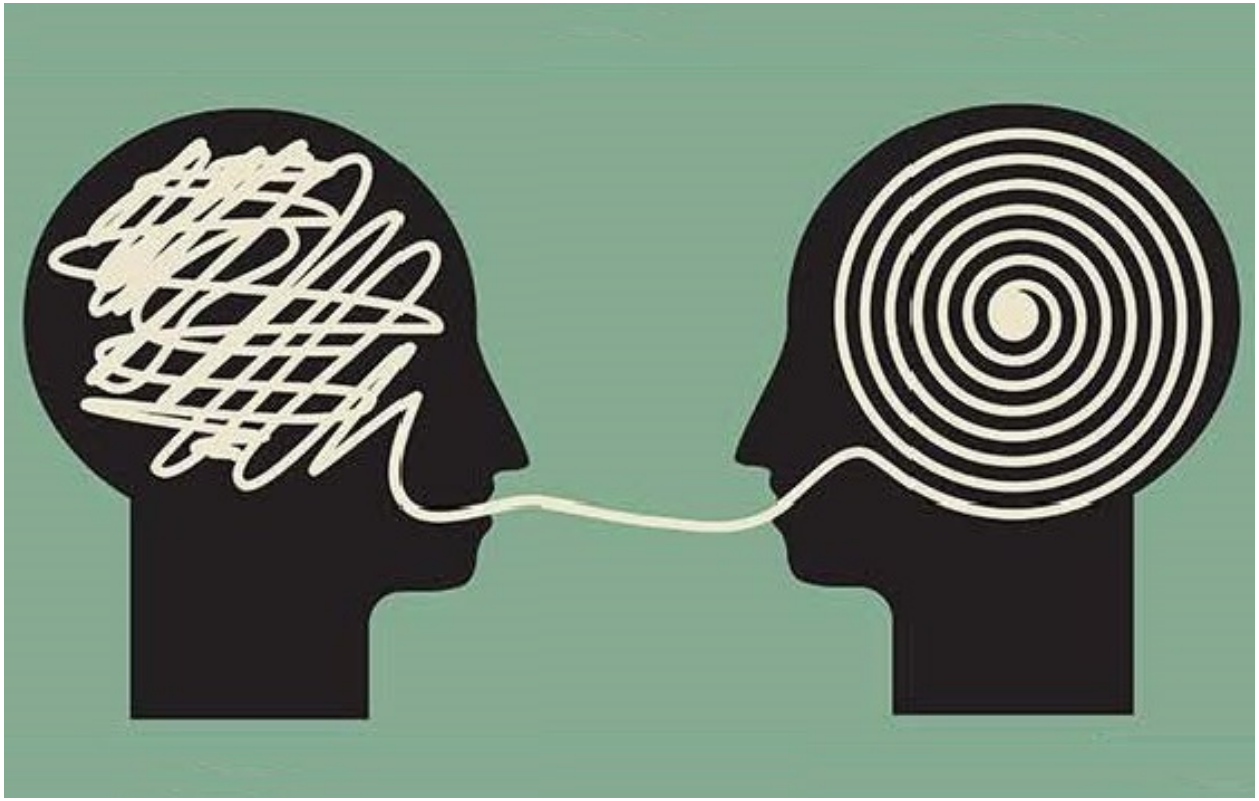

Figure: 1

- Filosofando. [Google, 2019]

Indivíduo x Sociedade

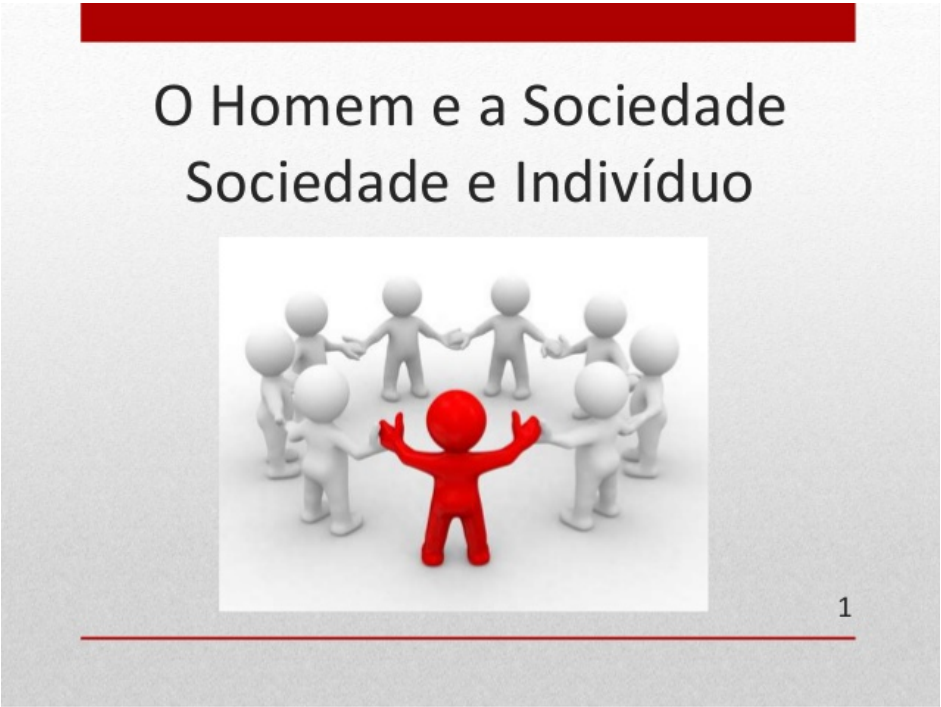

Figure: 2

- Indivíduo x Sociedade. [Google, 2019]

Indivíduo x Sociedade

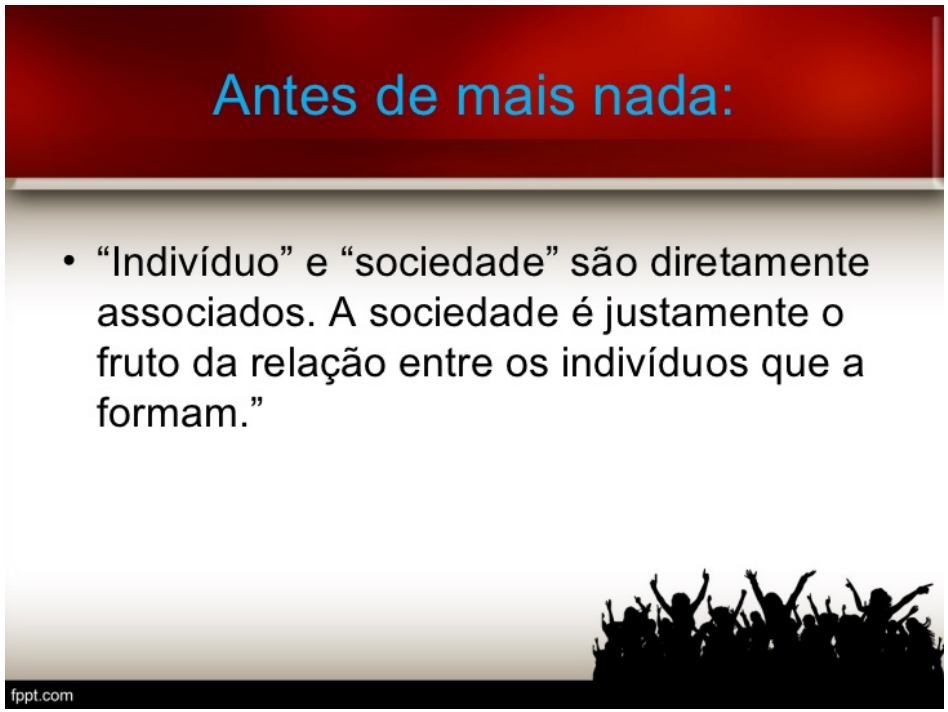

Figure: 3

- Indivíduo x Sociedade. [Google, 2019]

- No Desenvolvimento de Software...

Figure: 4

- Equipe de Desenvolvimento de Software. [Google, 2019]

Qualidade de Software

- Nossos Processos tem qualidade?
- Nossos Produtos tem qualidade?

Figure: 5

- Qualidade de Software. [Google, 2019]

Qualidade de Software

- Segundo a norma brasileira NBR ISO 8402, qualidade é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.

Qualidade de Software

- Segundo a norma brasileira NBR ISO 8402, qualidade é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.
- Conquistar um produto de qualidade requer planejamento de atividades que darão suporte na obtenção da mesma.

Qualidade de Software

- Segundo a norma brasileira NBR ISO 8402, qualidade é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.
- Conquistar um produto de qualidade requer planejamento de atividades que darão suporte na obtenção da mesma.
- Essa prática requer gerenciamento, documentação, padrões de convenções e métricas, auditorias, relatórios e ferramentas.

Qualidade de Software

- Segundo a norma brasileira NBR ISO 8402, qualidade é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.
- Conquistar um produto de qualidade requer planejamento de atividades que darão suporte na obtenção da mesma.
- Essa prática requer gerenciamento, documentação, padrões de convenções e métricas, auditorias, relatórios e ferramentas.
- Obviamente a qualidade se adapta ao estilo do produto e do processo.

Verificação e Validação

Figure: 6

- Verificação x Validação. [Google, 2019]

Importância do Teste de Software

- Por que o teste é importante?
- Porque objetiva melhorar a qualidade do produto entregue ao nosso cliente, ou seja, objetiva garantir que o produto entregue correspondente às necessidades explicitadas pelo cliente.
- Nem mais, nem menos.

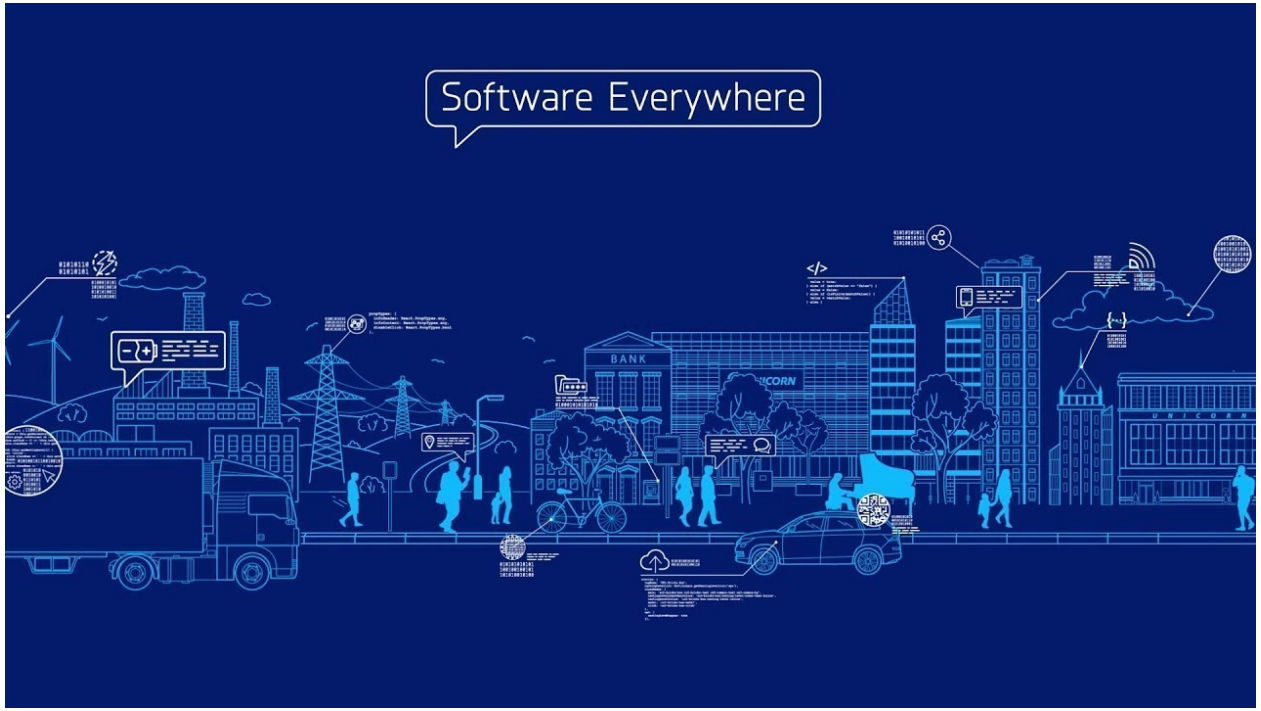

Figure: 7

- Onipresença do Software. [Google, 2019]

Teste x Debug

- Testar e Debugar é a mesma coisa?
- a Resposta é...
- Não.
- Teste objetiva revelar a presença de defeitos.
- Debug é o processo de encontrar o corrigir os defeitos.

Teste x Homologação

- Testar e homologar é a mesma coisa?
- a Resposta é...
- Não.
- os objetivos são diferentes.
- Teste objetiva revelar a presença de defeitos.
- Homologar objetiva aferir que o funcionamento esteja adequado às necessidades explícitadas.

Custo do Teste

- Por que é difícil a implantação de um processo de teste?
- Custo.

Custo dos Defeitos

- O site `https://raygun.com/blog/costly-software-errors-history/`, apresenta um relatório atualizado (2019) do ranking dos maiores custos associados a defeitos de software, que serão descritos a seguir:

- Prejuízo de US$ 125 milhões.

Figure: 8

- NASA's Mars Climate Orbiter. [HARLEY, 2019]

- O Ariane 5 custou cerca de US $ 8 bilhões para ser desenvolvido e carregava uma carga útil de US $ 500 milhões em satélites quando explodiu.

Figure: 9

- Ariane 5 Flight 501. [HARLEY, 2019]

Therac 25

- O Therac-25 é o nome de uma máquina de radiografia fabricada pela Atomic Energy of Canada (AECL) em 1985.
- Esse dispositivo “assassino” foi responsável pela morte de três pacientes entre 1985 e 1987.
- A causa da morte dessas pessoas foi o envenenamento por radiação.
- O problema estava na quantidade de radiação emitida em seu funcionamento.
- Enquanto um paciente deveria receber cerca de 200 rads, a Therac-25 bombardeava as pessoas com o valor absurdo de 15 mil e 20 mil rads.

Figure: 10

- Therac 25. [Google, 2019]

O que é teste?

Figure: 11

- ???

O que é teste?

Segundo a norma [IEEE 610.12.1990]

"Teste é o processo de operar um sistema ou componente sob condições específicas, observando e registrando os resultados, avaliando alguns aspectos do sistema ou componente."

Níveis de Maturidade

- Segundo [Beizer, 1990], são cinco:

Níveis de Maturidade

- Segundo [Beizer, 1990]
- Nível 0 - Não há diferença entre teste e depuração (debugging).

Níveis de Maturidade

- Segundo [Beizer, 1990]
- Nível 0 - Não há diferença entre teste e depuração (debugging).
- Nível 1 - O propósito do teste é mostrar que o software funciona.

Níveis de Maturidade

- Segundo [Beizer, 1990]
- Nível 0 - Não há diferença entre teste e depuração (debugging).
- Nível 1 - O propósito do teste é mostrar que o software funciona.
- Nível 2 - O propósito do teste é mostrar que o software não funciona.

Níveis de Maturidade

- Segundo [Beizer, 1990]
- Nível 0 - Não há diferença entre teste e depuração (debugging).
- Nível 1 - O propósito do teste é mostrar que o software funciona.
- Nível 2 - O propósito do teste é mostrar que o software não funciona.
- Nível 3 - O propósito do teste não é provar nada, mas reduzir o risco de não funcionamento a um valor aceitável.

Níveis de Maturidade

- Segundo [Beizer, 1990]
- Nível 0 - Não há diferença entre teste e depuração (debugging).
- Nível 1 - O propósito do teste é mostrar que o software funciona.
- Nível 2 - O propósito do teste é mostrar que o software não funciona.
- Nível 3 - O propósito do teste não é provar nada, mas reduzir o risco de não funcionamento a um valor aceitável.
- Nível 4 - Teste não é uma ação, mas sim uma disciplina mental (institucionalizada na empresa) que resulta em software de baixo risco sem que seja empregado muito esforço de teste.

Taxonomia

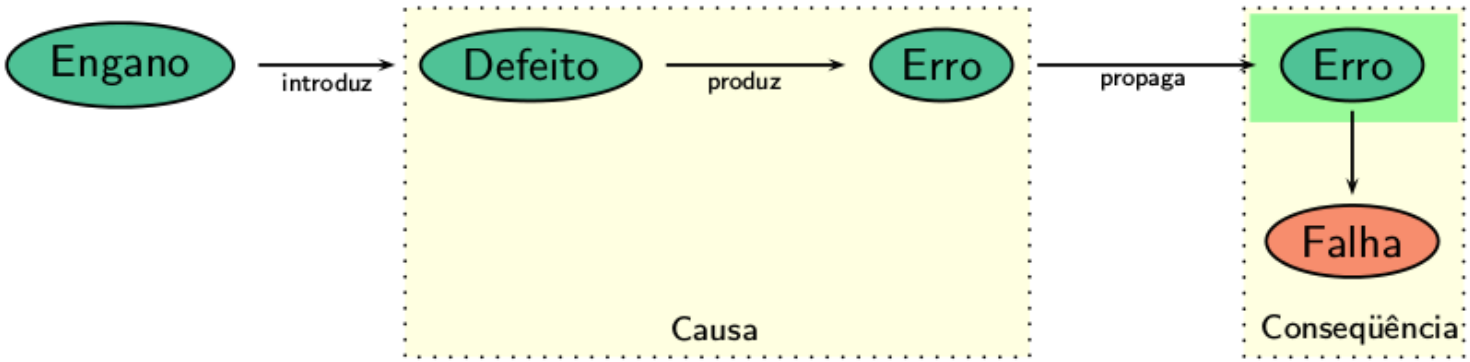

Figure: 12

- [Google, 2019]

Taxonomia

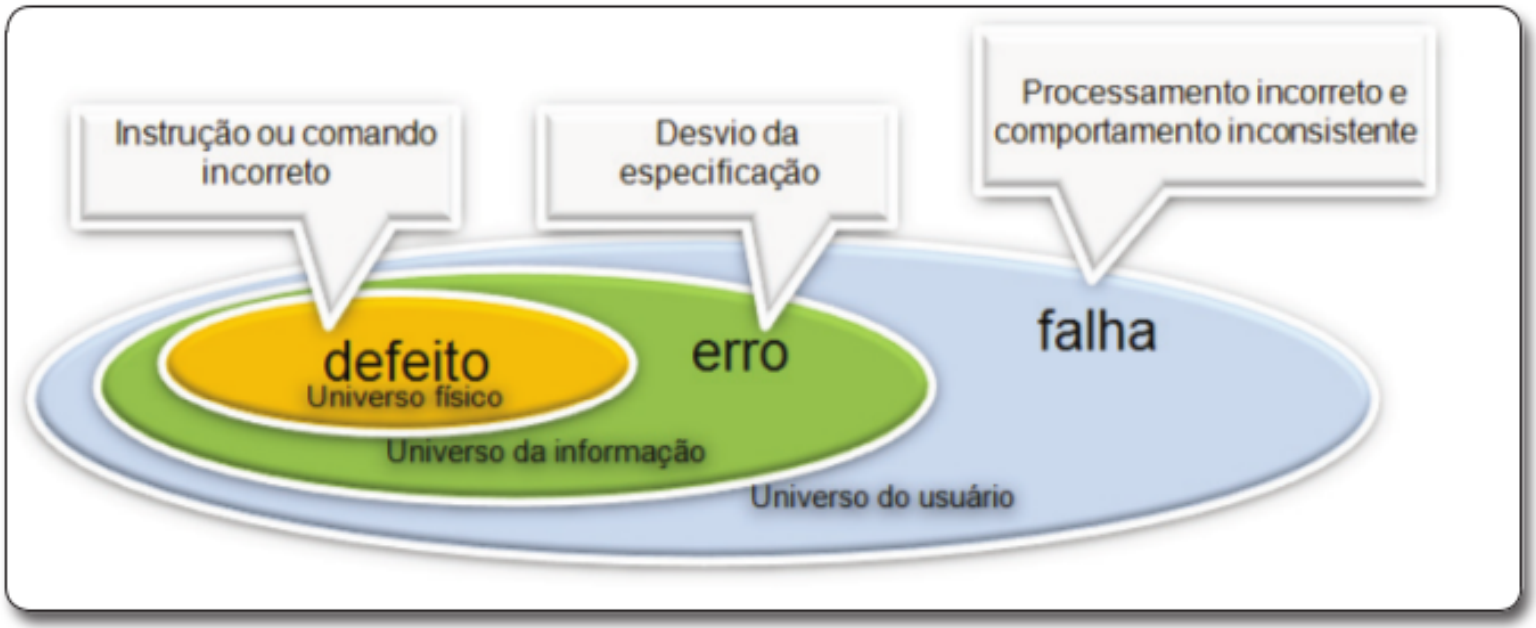

Figure: 13

- [NETO, 2009]

Dado de Teste

- Qualquer valor definido como entrada ou saída para um caso de teste qualquer.

Caso de Teste

- Especificação de um teste. Pode ser textual, script em liguagem de programção, etc.
- Deve conter:
  1. Identificador;
  2. Valor de Entrada;
  3. Pré-condição;
  4. Roteiro ou procedimento de execução;
  5. Resultado esperado.

Critério de Teste

- serve para selecionar e avaliar casos de teste de forma a aumentar as possibilidades de provocar falhas ou, quando isso não ocorre, estabelecer um nível elevado de confiança na correção do produto.

Técnica de Teste

- Forma de se definir e conduzir os testes. São definidas a partir da fonte da informação obtida para se elaborar o teste.

Por que Testar?

Figure: 14

- ???

Por que Testar?

PORQUE NÃO EXISTE SOFTWARE LIVRE DE DEFEITO.

Figure: 15

- [Google, 2019]

Qual a Origem dos Defeitos?

Figure: 16

- ???

Origem dos Defeitos

- **Erros de Construção**: (falha para satisfazer a especificação através de erros na implementação);
- **Erros de Especificação**: (falha para escrever uma especificação que corretamente represente o projeto);
- **Erros de Projeto**: (falha para satisfazer o entendimento de um Requisito);
- **Erros de Requisitos**: (falha para satisfazer um requisito real).

Por que Testar?

OS DEFEITOS ESTÃO POR TODA PARTE.

Figure: 17

- [Google, 2019]

Eficácia do Teste

Independentemente do tipo ou da origem dos defeitos, nossos testes devem ser capazes de revelar a presenạ dos mesmos.

Figure: 18

- [Google, 2019]

Paradoxo do Pesticida [Beizer, 1990]

Figure: 19

- [Google, 2019]

Objetivo do Teste

Figure: 20

- ???

Objetivo do Teste

Revelar a presença de defeitos nos nossos softwares.

Figure: 21

- [Google, 2019]

Quando Testar?

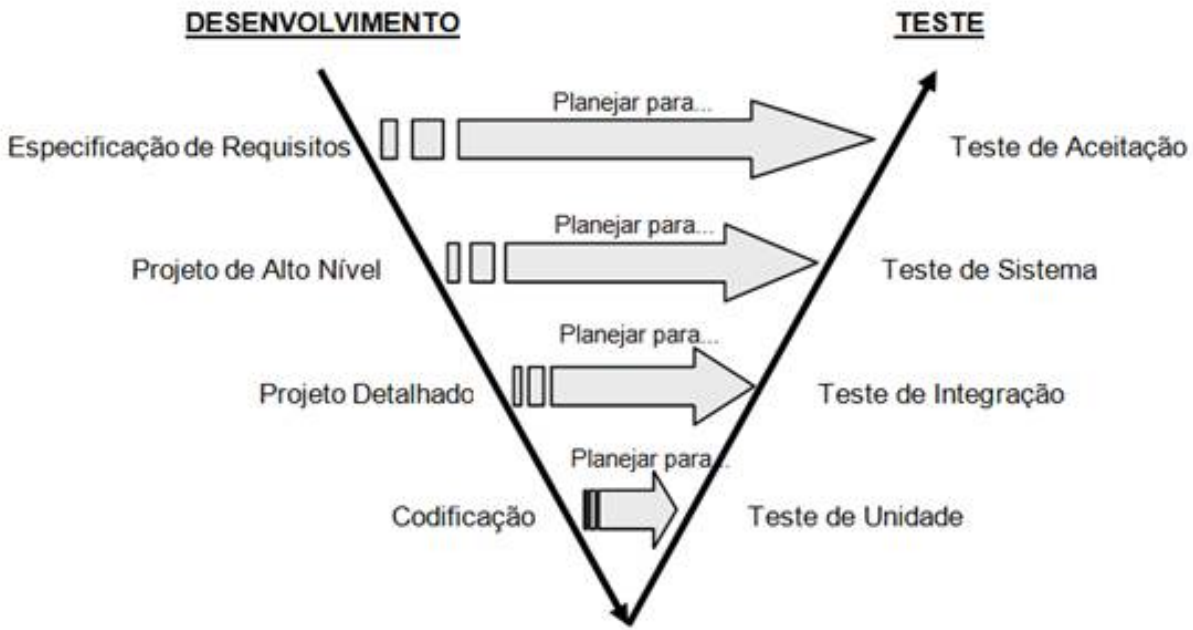

Figure: 22

- [Google, 2019]

Efetividade dos Níveis de Teste

- Os testes de unidades executados podem reduzir entre 30% e 50% dos defeitos dos softwares.
- O processo de revisão de código permite diminuir entre 20% e 30% os defeitos.
- O processo de testes de sistemas pode remover de 30% a 50% os defeitos remanescentes.

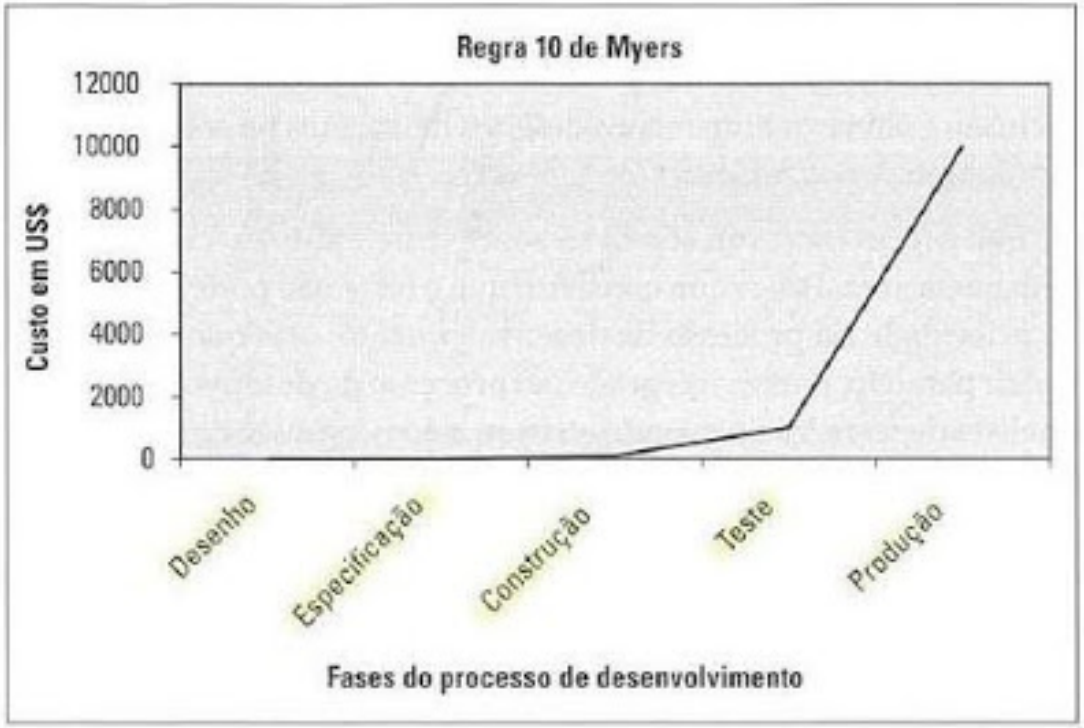

Figure: 23

- Regra 10 de Myers. [Google, 2019]

Níveis de Teste

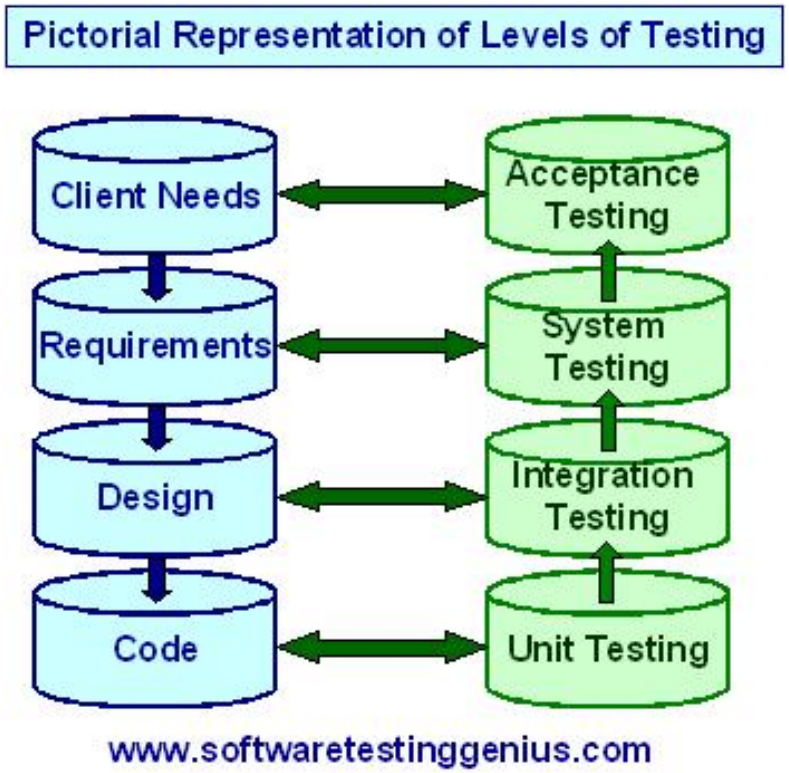

Figure: 24

- [Google, 2019]

Tipos de Teste

- Diferentes tipos de testes podem ser utilizados para verificar se um programa se comporta como o especificado.
- Basicamente, os testes podem ser classificados em:
  - Teste Funcional (**teste caixa-preta**);
  - Teste Estrutural (**teste caixa-branca**) e
  - Teste Baseado em Defeitos (**Testes de Mutantes**).

Tipos de Teste

- A técnica de teste é definida pelo tipo de informação utilizada para realizar o teste.

Técnica de Teste Funcional

- **Técnica caixa-preta** - os testes são baseados exclusivamente na especificação de requisitos do programa. Nenhum conhecimento de como o programa está implementado é requerido.

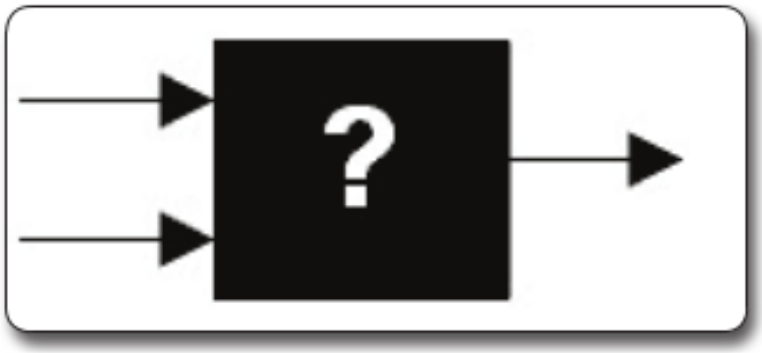

Figure: 25

- Técnica de Teste Funcional. [NETO, 2009]

Técnica de Teste Estrutural

- **Técnica caixa-branca** - os testes são baseados na estrutura interna do programa, ou seja, na implementação do mesmo.

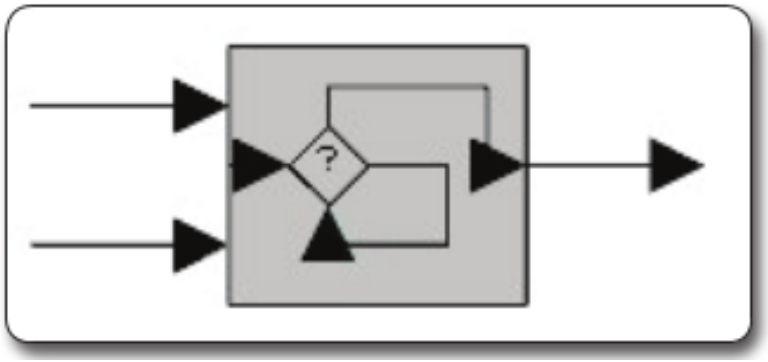

Figure: 26

- Técnica de Teste Estrutural. [NETO, 2009]

Técnica de Teste Baseada em Defeitos

- **Técnica baseada em defeitos** - os testes são baseados em informações históricas sobre defeitos cometidos frequuentemente durante o processo de desenvolvimento de software.

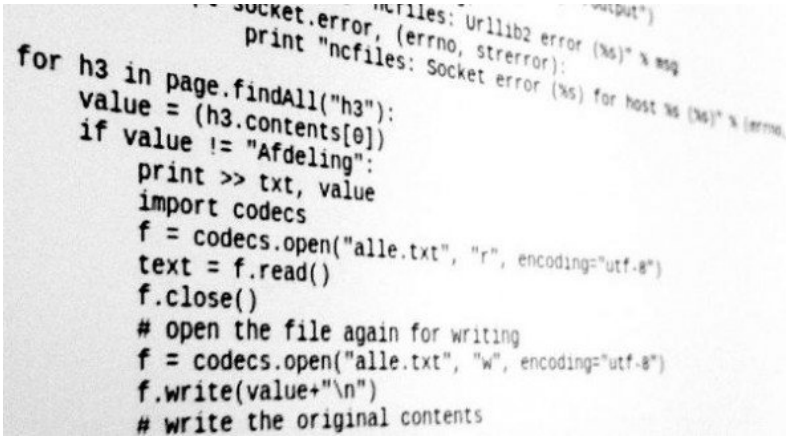

Figure: 27

- Teste de Mutantes. [NETO, 2009]

Variações de Tipos de Teste

Teste de Performance

Modelagem do uso esperado do sistema, pela simulação do acesso de vários usuários concorrentemente.

Figure: 28 - Teste de Performance. [Google, 2019]

Variações de Tipos de Teste

Teste de Carga

Testar o comportamento do sistema submetendo-o ao processamento de um grande volume de dados.

Figure: 29

Variações de Tipos de Teste

Teste de Estresse

Determinar a estabilidade de um dado sistema para além da sua capacidade normal de operação.

Figure: 30 - Teste de Estresse. [Google, 2019]

Variações de Tipos de Teste

Teste de Confiabilidade

Determinar por quanto tempo o sistema é capaz de sustentar uma performance ótima.

Figure: 31

Variações de Tipos de Teste

Teste de Regressão

1. Os testes de regressão geralmente são executados após a correção de algum defeito ou após a adição de uma nova funcionalidade.

Variações de Tipos de Teste

Teste de Regressão

1. Os testes de regressão geralmente são executados após a correção de algum defeito ou após a adição de uma nova funcionalidade.
2. Seu objetivo é garantir que nenhum defeito foi acrescentado ao sistema após sua modificação.

Variações de Tipos de Teste

Teste de Regressão

1. Os testes de regressão geralmente são executados após a correção de algum defeito ou após a adição de uma nova funcionalidade.
2. Seu objetivo é garantir que nenhum defeito foi acrescentado ao sistema após sua modificação.
3. De nada adianta testar um sistema, verificar que ele não possui defeitos para aquele conjunto de casos de teste e após modificações no sistema, aqueles casos de teste não serem novamente executados, pois as novas mudanças podem trazer defeitos para o sistema.

Variações de Tipos de Teste

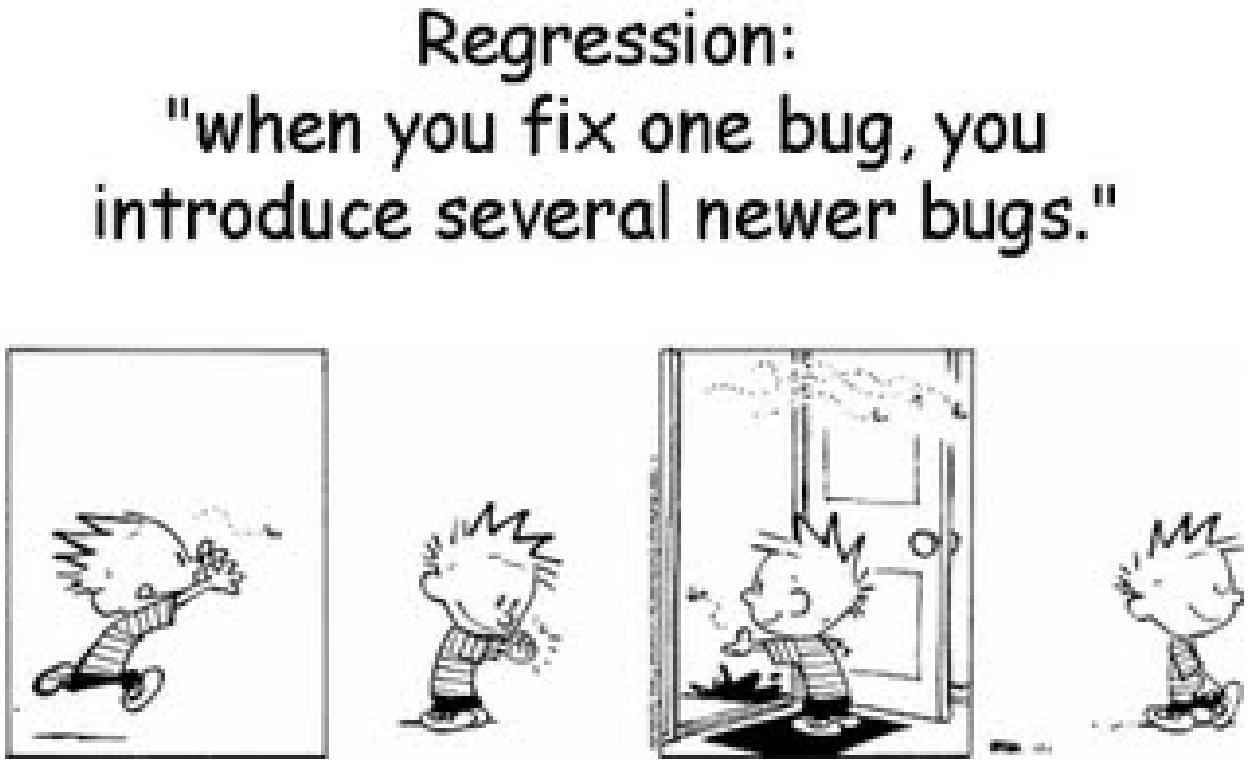

Figure: 32

- Teste de Regressão. [Google, 2019]

Referências Bibliográficas

Beizer, Boris.
Software Testing Techniques.
Van Nostrand Reinhold Company, New York, 2nd edition,1990.

B. Boehm and V. R. Basili.
Software defect reduction top 10 list.
Computer, 34(1):135-137, 2001. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/2.962984.

Lee Copeland.
A Practitioner's Guide to Software Test Design
Artech House Publishers, Boston, 2003.

R. D. Craig and S. P. Jaskiel.
Systematic Software Testing.
Artech House Publishers, 2002.

Imagens, Google
Google Imagens.
https://www.google.com.br. Acesso em 19/06/2019.

IEEE.
IEEE standard glossary of software engineering terminology.
Standard 610.12-1990 (R2002),
IEEE Computer Society Press, 2002.

Harley, Nick.
11 of the most costly software errors in history [2019 update]
https://raygun.com/blog/costly-software-errors-history/. Acesso em 19/06/2019.

Neto, Arilo Cláudio Dias Neto.
Introdução ao Teste de Software.
Engenharia de Software Magazine. Ed. 01, 2009

F. Shull, V. Basili, B. Boehm, A. W. Brown, P. Costa, M. Lindvall, D. Port, I. Rus, R.e Tesoriero, and M. Zelkowitz.
What we have learned about fighting defects.
In VIII International Symposium on Software Metrics - METRICS'02, pages 249-258, Washington, DC, USA, June 2002. IEEE Computer Society. ISBN 0-7695-1339-5.

Superintendência de Integração Tecnológica da Informação | Secretaria de Estado da Educação | Somos todos GOIÁS GOVERNO DO ESTADO