INSTITUTO FEDERAL
Santa Catarina
Câmpus Gaspar

Teste de Software

Prof. Marcos Rodrigo momo, M.Sc.
marcos.momo@ifsc.edu.br

Gaspar, abril 2021.

Formação e Docência

- 1994 - FURB
- 2000 – BCC
- 2005 – Especialização TI
- 2015 – Mestrado em Engenharia Ambiental
- 2018 – Doutorando na UFSC

- Professor desde 2014 (CEDUP)
  - Linguagem de programação
  - Sistemas operacionais
  - Programação orientadas a objetos
  - Sistemas distribuídos
  - SOA

Área de pesquisa

- Sistema de monitoramento e alerta de cheias
  - 2009 Membro do grupo CEOPS/FURB
  - Sistemas distribuídos
  - Redes neurais (modelagem hidrológica)
  - Mapeamento de áreas de inundação

- TCCs na graduação
  - Sistemas especialistas aplicado ao monitoramento do sistema de alerta
  - Previsão hidrológicas em tempo atual

Roteiro

- Apresentação do professor
- Apresentação do plano de ensino
- Introdução ao teste de software
- Atividades

Plano de ensino
Tópicos

- Introdução ao teste de software
- Fundamentos do Teste de Software
- Fases de Teste de software e o ciclo de desenvolvimento (EaD)
- Introdução às técnicas de testes
- Técnicas para modelagem de testes
  - Teste de caixa-preta e caixa-branca
- Técnicas Estáticas de teste (EaD)

Plano de ensino
Objetivos

- Implementar sistemas computacionais seguindo as especificações e paradigmas da lógica e das linguagens de programação.
- Avaliar e testar sistemas computacionais de modo a garantir que foi desenvolvido de maneira apropriada e consistente, correspondendo aos requisitos estabelecidos e que apresente comportamento esperado.

- **Conhecer e aplicar técnicas de teste de software**

Plano de ensino
Metodologia ANP

- Sempre em laboratório virtual (Google Meet)
- Aulas expositiva e dialogada
- Conceituação teórica
- Aplicação prática
- Atividades em sala de aula virtual

Plano de ensino
Metodologia - EaD

- ~~Até 15% da carga horária da UC, na modalidade EaD~~
- Será disponibilizado pelo **SIGAA** o material didático
- Realização de atividade para fixação do conhecimento e para avaliação da aprendizagem com entrega no AVA adotado
- Canal de comunicação para esclarecimento de dúvidas: ferramentas institucionais disponíveis no SIGAA (chat e fórum de discussão)
- O docente dará feedback da atividade entregue pelo aluno na ferramenta SIGAA

Plano de ensino
Recuperação paralela

- Atividades em laboratório e extra classe virtual
  - Correção de trabalhos e provas em laboratório virtual
  - Disponibilização através do ambiente de aprendizagem, lista de exercícios e atividades práticas complementares sobre o conteúdo a ser recuperado
- Avaliação baseada na aplicação de prova e/ou trabalho complementar (final do período de aulas)

Plano de ensino
Provas e Trabalhos

- Prova conceitual (individual)
- Atividades em laboratório virtual (individual/grupo)
- Trabalhos intermediários com defesa (individual/grupo)
- Trabalho final com defesa (individual/grupo)

Plano de ensino
Avaliação

- A avaliação será composto por:

- 1) Trabalhos práticos (TP) – exercícios de aula;
- 2) Uma Prova individual (Prova);
- 3) Um Trabalho Final (TF).

- A média final será assim calculada:

MÉDIA-FINAL =

Trabalhos práticos (TP) [10%], Prova (PR) [40%] e Trabalho Final (TF) [50%]:

Plano de ensino

- Em caso de verificação de cópia, a nota da atividade em questão será **ZERADA**, tanto para o aluno que copiou, quanto para aquele que deixou copiar.

- Requisitos para a aprovação:
  - Média >= 6.0
  - Frequência >= 75%

Contato e Atendimento

- Contato:
  - marcos.momo@ifsc.edu.br
- Atendimento ao aluno:
  - Quinta-feira: das 13:30 às 15:30 horas
  - Local: remotamente, agendar por e-mail para a criação de uma sala no google meet
- Material e atividades pelo SIGAA

Cronograma das aulas
Carga horária 40h
Módulo 1

	MÓDULO 1: 26/04 – 21/05 (4 semanas)					
	2ª feira	3ª feira	4ª feira	5ª feira	6ª feira	sábado
07:20:00						prog.internet II
08:15:00						prog.internet II
09:10:00						teste software
10:25:00						teste software
11:20:00						
13:30:00						
18:30:00		prog.internet II		prog.internet II	teste software	
19:25:00		prog.internet II		prog.internet II	teste software	
20:40:00		prog.internet II		prog.internet II	teste software	
21:35:00		prog.internet II		prog.internet II	teste software	

Disponível em: https://www.ifsc.edu.br/web/campus-gaspar/horarios-ensalamento

Cronograma das aulas
Carga horária 40h
Módulo 2

MÓDULO 2: 24/05 – 25/06 (5 semanas)						
2ª feira	3ª feira	4ª feira	5ª feira	6ª feira	sábado	
						07:20:00
						08:15:00
					prog.internet II	09:10:00
					prog.internet II	10:25:00
						11:20:00
						13:30:00
teste de software	prog.internet II	prog.internet II				18:30:00
teste de software	prog.internet II	prog.internet II				19:25:00
teste de software	prog.internet II	prog.internet II				20:40:00
teste de software	prog.internet II	prog.internet II				21:35:00

Disponível em: https://www.ifsc.edu.br/web/campus-gaspar/horarios-ensalamento

bloco 1: C/H por semana - > 4 * 6 aula = 24
bloco 2: C/H por semana - > 4 * 4 aula = 16
8 semanas - > Total 40 horas

ADS 4				26/4 a 30/4	3/5 a 7/5	10/5 a 15/5	17/5 a 21/5	24/5 a 29/5	31/5 a 4/6	7/6 a 12/6	14/6 a 18/6	21/6 a 25/6	28/6 a 3/7	5/7 a 9/7	26/7 a 31/7	2/8 a 6/8	9/8 a 13/8	16/8 a 21/8	23/8 a 27/8	30/8 a 3/9	6/9 a 11/9	13/9 a 18/9
	UCs	CH	Docente	Módulo 1				Módulo 2					Módulo 3				Módulo 4					finalização e conselhos de classe
1	Teste de Software	40	MARCOS RODRIGO MOMO/	6	6	6	6	4		4	4	4										
2	Análise de Sistemas II	80	ROGÉRIO ANTONIO SCHMITT/	8	8	8	8	10	8	10	10	10										
3	Práticas em Desenvolvimento de Sistemas I	80	TAMER CAVALCANTE/										10	10	10	10	10	10	10	10		
4	Sistemas Operacionais	40	ANDREU CARMINATI/														10	10	10	10		
5	Metodologia de Pesquisa	40	LEONIDAS de MELLO Jr./	10				0	0	0	0	0		10	10	10	0	0	0	0	0	
6	Programação para Internet II	80	MARCOS RODRIGO MOMO/		10	10	10	10	10	10	10	10										
7	Gerência de Projetos	40	THIAGO PAES/										8	4	4	4	4	4	4	4	4	
	TOTAL	400	TOTAL	24	24	24	24	24	18	24	24	24	18	24	24	24	24	24	24	24	4	

INSTITUTO FEDERAL
Santa Catarina
Câmpus Gaspar

Motivação para fazer Teste de Software

Clientes mais exigentes

Empresas de SW se reestruturando

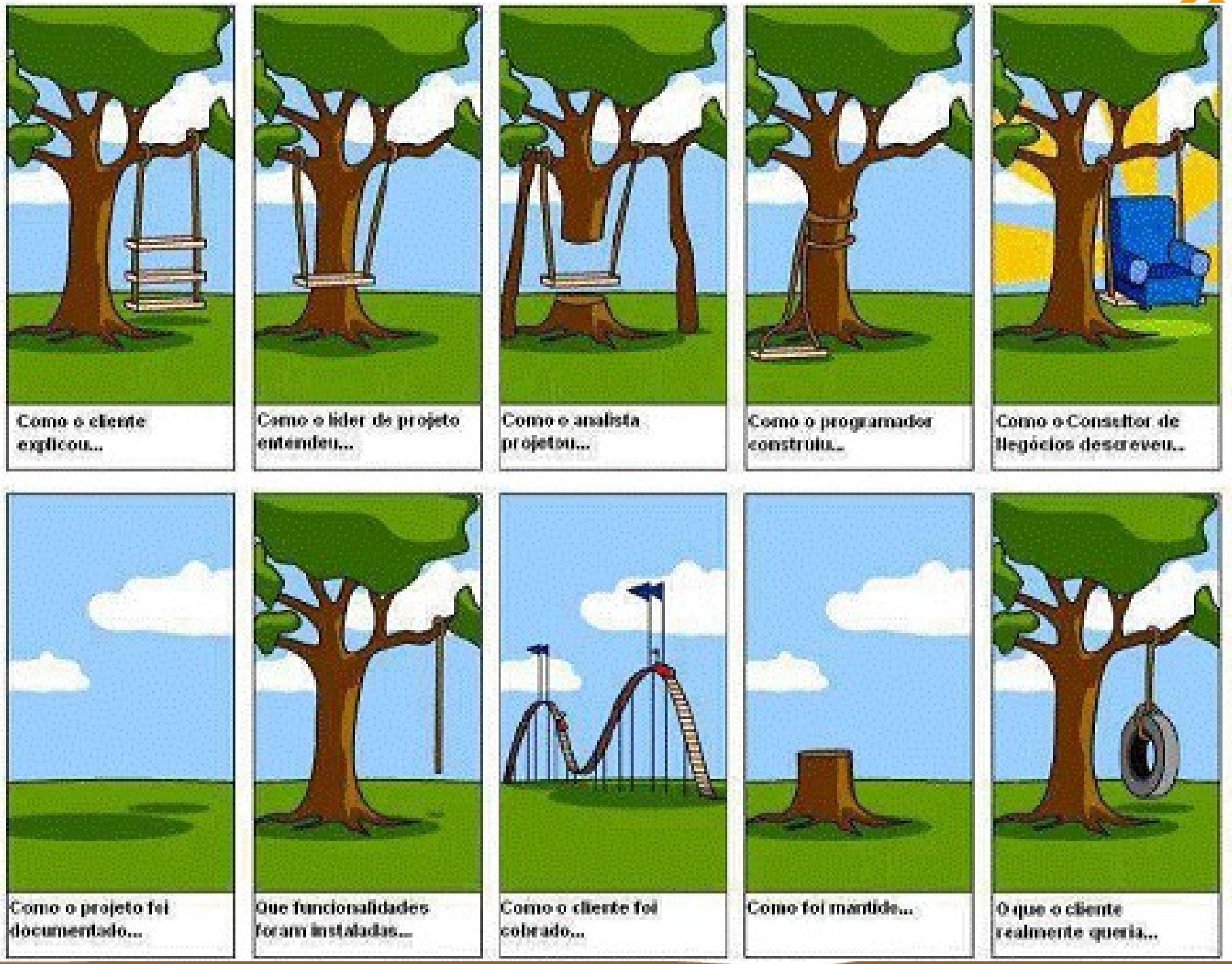
- Equipes/processos de desenvolvimento
- Equipe especializada em teste

Produto de SW com MELHOR qualidade

Mercado mais competitivo

Validação, verificação e teste

- Construção de *software* não é tarefa simples.
- Pode ser bastante complexa dependendo das características e dimensões do sistema a ser criado.
- Consequência
  - Está sujeito a diversos tipos de problemas que acabam resultando na obtenção de um produto diferente daquele que se esperava.

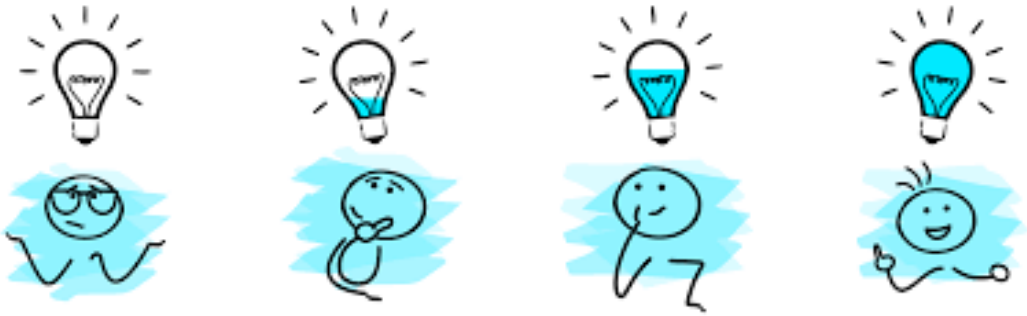

Como o cliente explicou...
Como o líder de projeto entendeu...
Como o analista projetou...
Como o programador construiu...
Como o Consultor de Negócios descreveu...
Como o projeto foi documentado...
Que funcionalidades foram instaladas...
Como o cliente foi cobrado...
Como foi mantido...
O que o cliente realmente queria...

Validação, verificação e teste

- Muitos fatores podem ser identificados como causas de tais problemas, mas a maioria deles tem uma única origem: **erro humano**!

Validação, verificação e teste

- Atividades de engenharia normalmente dependem da **habilidade**, da **interpretação** e da **execução** das pessoas que o constroem.
- Consequência: erros acabam surgindo, mesmo com a utilização de **métodos** e **ferramentas** de Engenharia de Software.



Validação, verificação e teste

- Para que os erros não perdurem, ou seja, **para serem descobertos antes de o *software* ser liberado para utilização (MUNDO IDEAL)**
- Existe uma série de atividades, coletivamente chamadas de “Validação, Verificação e Teste” ou “VV&T” com a finalidade de garantir que:
  - A **maneira** pela qual o ***software*** está sendo construído está correta.
  - O **produto** que está sendo construído está conforme foi especificado.

Validação, verificação e teste

VERIFICAÇÃO: refere-se ao conjunto de tarefas que garantem que o *software* ***implementa corretamente*** uma função específica.

"Estamos criando o produto corretamente?"

VALIDAÇÃO: refere-se ao conjunto de tarefas que asseguram que o *software* foi criado e pode ser ***rastreado segundos os requisitos do cliente***.

"Estamos criando o produto certo?"

Validação, verificação e teste

- MAS…. As atividades de VV&T **não se restringem ao produto final**.
- Podem e **devem** ser conduzidas durante **TODO** o processo de desenvolvimento do *software*
- Desde a sua concepção
  - Levantamento de requisitos
- Deve englobar diferentes técnicas.

Fases de desenvolvimento de software

- Levantamento dos requisitos (análise)
- Modelagem dos diagramas (projeto)
  - Caso de uso, diagrama de classes, diagrama de sequência entre outros diagramas
- Implementação dos sistema (implementação)
- Fases de teste

Fase de testes

- Testes feitos pelo próprio programador durante a programação
  - Unit test: teste de classes individuais (ou de grupos de classes relacionadas)
  - Functional test: teste de funções inteiras (item de menu, p. ex.)
  - Component test: teste de componentes inteiros (exe, dll, ...) sem (ou com pouco) scaffolding
  - Testes feitos por equipes independentes de teste
  - System test: testa a integração entre todos os componentes do produto
  - Alpha test: teste de produto inteiro dentro de casa
  - Beta test: teste de produto inteiro fora de casa
  - **Testes devem ser automatizados**

Validação, verificação e teste

- As atividades de VV&T podem ser divididas em **estáticas** e **dinâmicas**.
  - **Estáticas** – não requerem a execução ou mesmo a existência de um programa ou modelo executável para serem conduzidas.
  - **Dinâmicas** – se baseiam na execução de um programa ou modelo.

Teste de Software

- É uma **atividade contínua e dinâmica**.
- O intuito é executar o programa ou o modelo utilizando ***algumas*** entradas em particular e verificar se seu comportamento está de acordo com o esperado.
- Caso a execução apresente resultados não especificados, dizemos que **um erro ou defeito** foi identificado.
- Os dados de tal execução podem servir como fonte de informação para a localização e a correção de tais defeitos (*Debugging* ou Depuração).

Alguns termos do jargão

Termo	Descrição	Exemplo (sistema de avaliação climática remoto)
Erro humano (Engano)	Comportamento humano que resulta na introdução de erros no sistema	Programador decide computar a hora da próxima transmissão como sendo (hora atual + 1).
Erro de sistema (*system fault*)	Característica de um sistema que pode levar a um defeito de sistema	Adicionar 1 a hora atual sem verificar se hora atual > 23
Defeito de sistema (*system error*)	Um estado incorreto do sistema, ou seja, um estado do sistema que não é esperado por seus projetistas	Valor da hora de transmissão é entrado como 24.XX em vez de 00.XX
Falha de sistema (*system failure*)	Um evento que ocorre em algum momento, quando o sistema não fornece o serviço esperado por seus usuários	A informação não é transmitida porque a hora está incorretamente informada

Mais alguns termos do jargão

- Domínio de entrada de um programa:
  - É o conjunto de todos os possíveis valores que podem ser utilizados para executar corretamente o programa.
- Chamaremos programa de **P** e o domínio de entrada do programa de **D(P)**.

Mais alguns termos do jargão

- Um “dado de teste” para um programa é um elemento de D(P).
- Um “caso de teste” é um par formado por:
  - Um “dado de teste”
  - Mais o resultado esperado para a execução do programa com aquele dado de teste.

- Caso de teste = dado de teste + Resultado

Exemplo

- Tem-se um programa que recebe como parâmetros de entrada dois números inteiros *x* e *y:*
  - Com *y>=0*, e computa o valor de x^y, indicando um erro caso os argumentos estejam fora do intervalo especificado.

- **O D(P) é formado por todos os possíveis pares de números inteiros (x,y).**

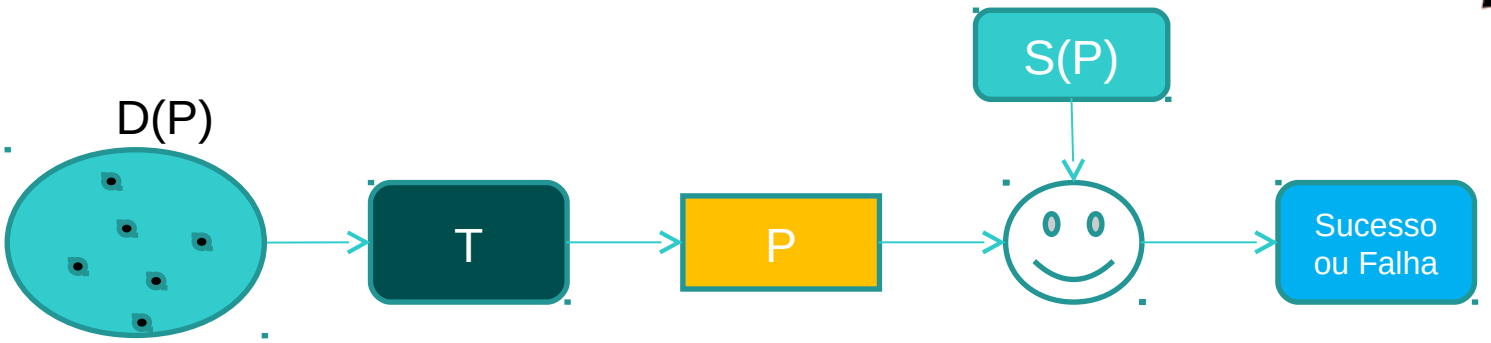
- Da mesma forma, pode-se estabelecer um **domínio de saída** do programa, que é o conjunto de todos os possíveis resultados produzidos pelo programa.
  - Neste exemplo, teríamos o conjunto de números inteiros e as mensagens de erro produzidos pelo programa como domínio de saída.

Exemplo (continuação)

- No programa que computa x^y teríamos os seguintes casos de teste:
  - Caso de teste 1: <(2,3),8>
  - Caso de teste 2: <(3,-1),”Erro”>

- Ao conjunto de todos os casos de teste usados durante uma determinada atividade de teste costuma-se chamar “**conjunto de teste**” ou “**conjunto de casos de teste**”.

Cenário típico da atividade de teste



- Definindo-se um conjunto de casos de teste T, executa-se o programa em teste com T e verifica-se qual é o resultado obtido.
- Se o resultado coincidir com o resultado esperado, então nenhum erro foi identificado.
- Se para algum caso de teste o resultado obtido difere do esperado, então um defeito foi revelado.
- Fica por conta do testador, baseado na especificação do programa S(P) ou qualquer forma de documento que defina seu comportamento, a análise sobre a correção de uma execução.

Fases da atividade de teste

- Atividade de teste é complexa.
- Diversos fatores podem colaborar para a ocorrência de erros.
- Exemplos distintos de enganos:
  1. Utilização de um algoritmo incorreto para computar o valor das mensalidades a serem pagas para um empréstimo.
  2. Não utilização de uma política de segurança em alguma funcionalidade do *software*.

Fases da atividade de teste

- No primeiro caso, provavelmente o erro está confinado a uma função ou rotina que implementa de **forma incorreta** uma dada funcionalidade.
- No segundo caso, mesmo que exista uma certa política de segurança implementada de maneira correta, é preciso verificar se **todos os pontos** nos quais essa política deveria ser aplicada fazem-no de maneira correta.
- Por isso, a atividade de teste é dividida em fases com objetivos distintos.

Fases da atividade de teste

- De uma forma geral, podemos estabelecer como fases:
  - Teste de unidade,
  - Teste de integração,
  - Teste de sistemas.

Fases da atividade de teste
Teste de unidade

- O **Teste de Unidade** tem como foco as menores unidades de um programa, que podem ser funções, procedimentos, métodos ou classes.
- Espera-se que sejam identificados erros relacionados a algoritmos incorretos ou mal implementados, estruturas de dados incorretas, ou simples erros de programação.
- Como **cada unidade é testada separadamente**, o teste de unidade pode ser aplicado à medida que ocorre a implementação das unidades e pelo próprio desenvolvedor.

Fases da atividade de teste
Teste de integração

- No **Teste de Integração**, que deve ser realizado **após** serem testadas as unidades individualmente, a ênfase é dada na construção da estrutura do sistema.
- À medida em que as diversas partes do *software* são colocadas para trabalhar juntas, é preciso verificar se a interação entre elas funciona de maneira adequada e não leva a erros.
- Neste caso, é necessário um grande conhecimento das estruturas internas e das interações existentes entre as partes do sistema e, por isso, o teste de integração tende a ser executado pela **própria equipe de desenvolvimento**.

Fases da atividade de teste
Teste de sistema

- Depois que se tem o sistema completo, com todas as suas partes integradas, inicia-se o **Teste de Sistema**.
- O objetivo é verificar se as funcionalidades especificadas nos documentos de requisitos estão todas corretamente implementadas.
- Aspectos de correção, completude e coerência devem ser explorados, bem como **requisitos não funcionais** como segurança, performance e robustez.
- Muitas organizações adotam a estratégia de designar uma **equipe independente** para realizar o teste de sistemas.

Fases da atividade de teste
Teste de regressão

- Além dessas três fases de teste, destaca-se, ainda o que se costuma chamar de “Teste de Regressão”.
- Esse tipo de teste não se realiza durante o processo “normal” de desenvolvimento, mas sim durante a manutenção do *software*.
- A cada modificação efetuada no sistema, após a sua liberação, corre-se o risco de que novos defeitos sejam introduzidos.

Fases da atividade de teste
Teste de regressão

- Por este motivo, é necessário, após a manutenção, realizar testes que mostrem que as modificações efetuadas estão corretas

- Preciso testar se com os novos requisitos implementados (se for o caso) funcionam como o esperado e que os requisitos anteriormente testados continuam válidos.

Bibliografia

- Delamaro, M.E.; Maldonado, J.C.; Jino, M. **Introdução ao Teste de Software**. São Paulo: Elsevier, 2007.
- Justo, Daniela Sbizera Profª Dra. Introdução a teste de software. IFSC. Gaspar.