**Curso: Sistemas de Informação – Unidade São Gabriel**
**Disciplina: Fundamentos de Testes de Software – Período: 5º - Turno: Noite**
**Professor: Claudiney Vander Ramos**
**Trabalho Final**
**Data: 28/04/2020**

**Entregas:**

**01 - Plano de Testes, Testes de Unidade e Integração – DATA: 12/05/2020**
**02 - Testes de Sistema (desempenho: carga, estresse, volume) e Aceitação – DATA: 02/06/2020**
**03 - Gerência de Configuração (ponto extra) – DATA: 02/06/2020**

**Pontuação:** Esta atividade prática vale 20 pontos na disciplina *Fundamentos de Testes de Software.*

**Observações importantes:**

- O trabalho deve ser desenvolvido em grupos com até 5 alunos (máximo).

- Cada grupo deve preparar uma apresentação mostrando os resultados dos testes realizados.

- **A nota é individual, de acordo com a participação do aluno na execução do trabalho e na apresentação (todos os alunos devem participar da apresentação).**

**Objetivo:** Planejar, especificar, e executar rotinas de testes em um sistema desenvolvido pelo seu grupo na disciplina Técnicas Avançadas de Programação, ou outro sistema para os grupos cujos alunos não estejam cursando a disciplina Técnicas Avançadas de Programação.

## Parte 1 - Testes de Unidade e Integração

Aplicar técnicas de testes de unidade e integração, visando a cobertura de código e a integração entre os módulos do sistema de informação desenvolvido pelo grupo.

O grupo deve utilizar um *framework*/ferramenta para gerar um conjunto de casos de testes de unidade para esse sistema (por exemplo, JUnit, JsUnit, NUnit, CsUnit, PHPUnit, SimpleTest).

## Parte 2 - Testes de Sistema - Testes de Desempenho/Stress/Carga, Testes de Aceitação, Desenvolvimento guiado por testes (*Test Driven Development* – TDD)

Aplicar técnicas de testes de integração, sistema, aceitação, desempenho/carga/*stress* em um sistema de informação definido pelo grupo. Nesta etapa, o foco deve ser nos testes das funcionalidades e de aspectos não funcionais (desempenho, segurança, etc), considerando o sistema integrado.

O grupo deve utilizar *frameworks*/ferramentas para gerar uma bateria de testes para esse sistema (ex: Selenium, HTTPerf, WAPT, WebLoad, NeoLoad, New Relic, JMeter).

Usar os princípios do desenvolvimento guiado por testes para desenvolver um dos módulos do sistema (escolher um módulo ainda não implementado, escrever os casos de teste, e implementar o módulo com base neste conjunto de casos de teste, usando o ciclo de TDD).

**Tarefas que devem ser cumpridas:**

- Escrever um plano de testes para o sistema – seguir modelo do processo RUP
  - http://www.mimuw.edu.pl/~trybik/edu/0607/io/templates/rup-tp-template-master.pdf
  - http://sce.uhcl.edu/helm/rup_school_example/wcsoftwareprocessweb/templates/test/pt_tplan.htm
  - http://sce.uhcl.edu/helm/rationalunifiedprocess/process/artifact/ar_tstpl.htm
- Escrever os casos de testes para o sistema.
- Gerar relatórios com os resultados dos testes executados.

**O que deve ser entregue:**

- O projeto e a implementação, com a especificação das classes e dos casos de uso.
- A implementação das classes de testes necessárias.
- O projeto e a implementação dos casos de testes.
- A análise de cobertura dos testes (entregar os relatórios ou os gráficos que mostrem a cobertura atingida).
- A análise dos resultados dos testes executados.

**PONTO EXTRA 1**

Aplicar os conceitos de gerência de configuração e controle de versões para esse sistema. Utilizar uma ferramenta de gerência de configuração / controle de versão (ex: SVN, CVS, Git, Mercurial, Bazaar).

- Apresentação da ferramenta de Gerência de Configuração e Controle de Versão.
- Identificar os itens de configuração de software (ICS) e utilizar uma ferramenta para a gerência de configuração e controle de versão/alteração destes itens
- Utilizar uma ferramenta de controle de versão para gerenciar as diferentes versões dos artefatos do projeto
- Utilizar a ferramenta de controle de versão para auditoria e relatório de status da configuração

**PONTO EXTRA 2**

Escolher um tema e validar com o professor (para evitar temas repetidos)

- Teste baseado em Riscos
- Teste de Segurança
- Teste de Instalação/Desinstalação
- Teste de Regressão
- Teste da Fumaça (*Smoke Testing*)
- Teste de Mutação (*Mutation Test*)
- Teste de compatibilidade (*browsers*)
- Teste Baseado em Modelos (*Model Based Testing*)
- MPT.BR (Melhoria de Processo de Teste)
- Gestão de Defeitos de Software
- Homologação de Software

| **Data** | **Grupo** | **Linguagem/Framework de Testes / Gerência de Configuração** |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## Links: Ferramentas de testes - Testes de unidade/integração

1. **JavaScript**
    1. http://abundantcode.com/javascript-unit-testing-tools/
    2. http://qunitjs.com/
    3. http://unitjs.com/
    4. http://jsunit.net/
2. **Java**
    5. http://junit.org

6. https://netbeans.org/kb/docs/java/junit-intro_pt_BR.html

3. **C#**

7. http://www.nunit.org/
8. http://www.csunit.org/

4. **PHP**

9. https://phpunit.de/
10. http://www.simpletest.org/

## Links: Ferramentas de testes - Testes de Desempenho

1. Apache JMeter
2. LoadRunner
3. WebLOAD
4. Appvance
5. NeoLoad
6. LoadUI
7. WAPT
8. Loadster
9. LoadImpact
10. Rational Performance Tester
11. Testing Anywhere
12. OpenSTA
13. QEngine (ManageEngine)
14. Loadstorm
15. CloudTest
16. Httperf

Link:

- http://www.softwaretestinghelp.com/performance-testing-tools-load-testing-tools/

## Links: Ferramentas de Gerência de Configuração

1. VSS – Visual source safe
2. CVS - Concurrent version system
3. Rational Clear Case

4. SVN - Subversion
5. Perforce
6. Mercurial
7. Bazaar
8. Git

Link:

- http://www.softwaretestinghelp.com/top-5-software-configuration-management-tools/