SERVIÇO PÚBLICO FEDERAL
MINISTERIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE INFORMÁTICA, GESTÃO E DESIGN

SimSUS		
Testes de Software		
Preparado por	Álvaro Martins Espíndola, Maria Teresa Menezes Costa, Helise de Assis Almeida	Versão: 1.0

1. Introdução

Durante o desenvolvimento de um software, os programadores e envolvidos no projeto podem permitir que falhas cheguem ao produto final. Tais problemas, às vezes muito minuciosos para serem percebidos à primeira vista, atrapalham a continuidade do programa. Visando contornar a situação, vários testes de softwares foram criados, abrangendo desde camadas globais do sistema até níveis específicos. Neste trabalho serão abordados as técnicas, os tipos e os níveis de teste de software, sendo que um dos citados será usado no projeto SimSUS e explicado mais afundo em um tópico separado.

2. Técnicas de Teste de Software

Apesar de existir diversas maneiras de se testar um software, as técnicas de software são as mesmas. Elas são capazes de garantir o funcionamento e auxiliar na avaliação do processo. As principais técnicas são: funcional e estrutural.

2.1. Teste Funcional

O teste funcional também pode ser chamado de teste da caixa preta devido ao fato de que não existe conhecimento relacionado ao conteúdo do produto a ser testado, apenas sua entrada e saída. A detecção de problemas é feita de acordo com a apresentação das entradas e saídas respectivas de um componente ou sistema. Caso elas não estejam relacionadas, o teste encontrou um erro. O objetivo dessa técnica é verificar e garantir que os requisitos do sistema tenham sido atendidos.

2.2. Teste Estrutural

Esse teste é mais conhecido como teste da caixa branca e objetiva testar todo o código fonte para verificar o fluxo do programa. Essa técnica serve como

complemento ao teste funcional. É responsável pela análise da estrutura interna do sistema e utiliza os dados a respeito do fluxo do programa para apresentar os requisitos de teste.

3. Tipos de Testes de Software

O número de tipos de testes de softwares é grande e varia de acordo com seu objetivo particular. Nos tópicos a seguir estão selecionados alguns desses tipos.

3.1. Teste de Configuração

Testa se o alvo do teste funciona no hardware a ser instalado.

3.2. Teste de Instalação

Testa se o software é capaz de instalar em diferentes hardwares e em variadas condições, como pouco espaço de memória, interrupções de rede, atualizações, dentre outras.

3.3. Teste de Integridade

Testa os métodos de acesso à base de dados para registrar comportamento inadequado ou corrupção dos dados.

3.4. Teste de Segurança

Testa se o sistema apresenta vulnerabilidades, além do comportamento do mesmo mediante simulação de acesso ilegal.

3.5. Teste de Usabilidade

Testa a aprovação do sistema de acordo com a opinião dos usuários.

4. Níveis de Testes de Software

Os estágios ou níveis de teste analisam os problemas relativos ao nível em que o teste será realizado. Pode-se dividi-los em: aceitação, sistema, integração e unitário.

4.1. Teste de Aceitação

Nos primeiros patamares, estão localizados os testes de aceitação, os quais são planejados a partir da especificação de requisitos e simulam operações de rotina do sistema, verificando se o resultado está em conformidade com o solicitado. Uma técnica usada para tal processo é a chamada de Teste de caixa-preta, na qual são colocadas várias entradas que não são escolhidas de acordo com a estrutura do programa. Desse modo, muitas saídas são verificadas e conferidas com o esperado.

4.2. Teste de Sistema

Nessa seção, os testes são feitos como um usuário final, utilizando entradas que ele usaria na utilização do software dentro do ambiente dele. O objetivo desse teste é buscar as falhas, simulando o ambiente que será encontrado pelo usuário. Assim, é verificada a satisfação do produto no que tange a respeito de seus requisitos. Nela também pode se usar o método da caixa-preta.

4.3. Teste de Integração

Nessa fase serão testadas duas, ou mais, áreas do sistema atuando em conjunto. Aqui serão verificadas as falhas na interação entre as unidades já testadas separadamente. Essa fase de teste não é responsável pelo tratamento de interfaces com outros sistemas, já que este pode ser realizado até mesmo no decorrer do desenvolvimento. As falhas mais comuns são relacionadas ao envio e recebimento de dados. Um exemplo clássico é o teste de uma classe DAO, no qual o banco de dados e o modelo do DAO serão integrados e testados. Isso pode ser feito através do JUnit, framework desenvolvido para testes automatizados, porém é mais utilizado nos Testes de Unidade.

4.4. Teste de Unidade

Conhecido também como Teste Unitário, essa fase realiza os testes nas menores unidades do software. Cada unidade é considerada um componente do sistema, de modo que este deva operar corretamente de acordo com sua função, porém não garante que a parte funcionará integrada com as demais. Os aspectos a serem analisados nesse tipo de teste são os parâmetros de entrada e saída das interfaces, a integridade dos dados, as condições de limite e o tratamento de

erros. Dessa forma, é possível encontrar falhas de pequenas partes do sistema, afim de mostrar o funcionamento correto independente do todo. No Teste Unitário é realizado o teste de uma unidade lógica com uso de dados suficientes para se testar apenas a lógica da unidade em questão.

Para a eficácia desse teste, é necessária a utilização de outros testes de software em conjunto. Uma forma de utilizar essa técnica de teste é a integração da ferramenta JUnit para testar classes ou métodos desenvolvidos em Java. Pode-se também aderir à técnica de testes manuais ou testes efetuados com apoio de ferramentas que verificam os métodos usados para o desenvolvimento do software.

5. Teste de Unidade com JUnit

No momento de utilização dos testes de software, é necessário avaliar como ele será testado. Uma opção é o JUnit, juntamente com sua IDE e a linguagem de programação Java, que será empregado no SimSUS.

O JUnit é um framework que facilita o desenvolvimento e execução de testes unitários em código Java. Ele fornece uma completa API (conjunto de classes) para construir os testes e Aplicações gráficas e em modo console para executar os testes criados. Os principais motivos que favorecem o uso desse framework são a verificação de cada unidade de código se ela funciona da forma esperada, a facilidade de criação, execução automática de testes e a apresentação dos resultados e o fato de ser orientado a Objeto e gratuito, podendo realizar seu download de acordo com a configuração de IDE. Uma sugestão para evitar problemas de software utilizando essa ferramenta é a execução cotidiana dos testes unitários, afinal é mais fácil resolver problemas menores.

O funcionamento do JUnit consiste basicamente na criação de uma classe de testes para testar uma classe correspondente, garantindo com tais testes de unidade, que cada método testado está produzindo o esperado. Para isso é necessário herdar da classe TestCase do framework, fornecer um construtor recebendo como parâmetro o nome do teste a fazer, fornecer um ou mais métodos com o nome test...() sem parâmetros e sem valor de retorno, cada método de testes testa algo usando o método assertTrue(msg, condição) e fornecer um método estático suite() que informa quais são os métodos de testes.

O framework fornece uma interface gráfica para executar os testes. Para isso, basta informar o nome da classe de testes e quando estes forem executados, os métodos

testados podem apresentar resultados: verde para sucesso, roxo para falha e vermelho para exceção.

6. Teste Manual

Diferente de todos os outros já citados, o teste manual consiste em não utilizar um software para realizar as rotinas e operações do programa. Esse tipo é utilizado quando se necessita de análise e pensamento lógico e, fazendo um paralelo com o projeto SimSUS, o jogo disponibiliza vários caminhos de soluções que possuem diferentes resultados. Assim, o jogador pode moldar sua própria estratégia.

7. Conclusão

A utilização de testes de softwares é muito importante quando se deseja desenvolver produtos mais estáveis. A presença de níveis diferentes de informações requer o uso de variados tipos de testes, de forma que resulte em um programa com o máximo de eficácia possível. Um exemplo que será empregado no projeto SimSUS é o Teste de Unidade, com o auxílio do JUnit, que analisa cada componente separado do software. A escolha desse framework ocorreu devido ao fato de que ele exibe possíveis erros ou falhas das menores unidades do software desenvolvidas, o que previne o aparecimento de bug's, proveniente de códigos mal escritos. Sendo assim, a utilização do JUnit no projeto garantirá um nível de qualidade do produto durante o processo de desenvolvimento do software, já que tem como função verificar possíveis falhas evitando que ao final elas se tornem um grande problema.

Contudo, necessitaremos também do Teste Manual, pois a jogabilidade e a lógica de nosso jogo necessita de um comportamento diversificado, como o humano, que possa testar todos os caminhos possíveis para atingir o objetivo.

8. Referências

(1) Disponível em: <http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035> Acesso em 5 setembro 2015.

(2) Disponível em: < http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2004/Dissertacao_CidinhaCostaGouveia.pdf> Acesso em 5 setembro 2015

(3) Disponível em: <http://www.ic.unicamp.br/~ranido/mc626/FasesTestes.pdf> Acesso em 12 setembro 2015

(4) Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/apoo/html/impl/impl3.htm> Acesso em 12 setembro 2015

(5) Disponível em: <http://www.ic.unicamp.br/~eliane/Cursos/MC636-2009/Aula18-testes_unid_integracao.pdf> Acesso em 12 setembro 2015

(6) Disponível em: < http://www.matera.com/br/2013/07/19/fases-de-testes-de-software/> Acesso em 12 setembro 2015

(7) Disponível em: <http://siep.ifpe.edu.br/anderson/blog/?page_id=976> Acesso em 12 setembro 2015

(8) Disponível em: <http://www.devmedia.com.br/testes-de-unidade-com-junit/4637#ixzz3lzwVQJ8j> Acesso em 12 setembro 2015