3 Processo de Teste

Nesse capítulo será apresentado um processo de teste que foi desenvolvido para que diminua o retrabalho e o esforço gasto no processo de teste tradicional. Inicialmente é mostrada uma visão geral do processo e cada macro atividade desse processo será detalhada em uma seção específica.

3.1. Visão Geral do Processo

A figura 3 apresenta um processo de teste desenvolvido que possibilita a transformação da especificação dos requisitos em scripts de teste de maneira semi-automática.

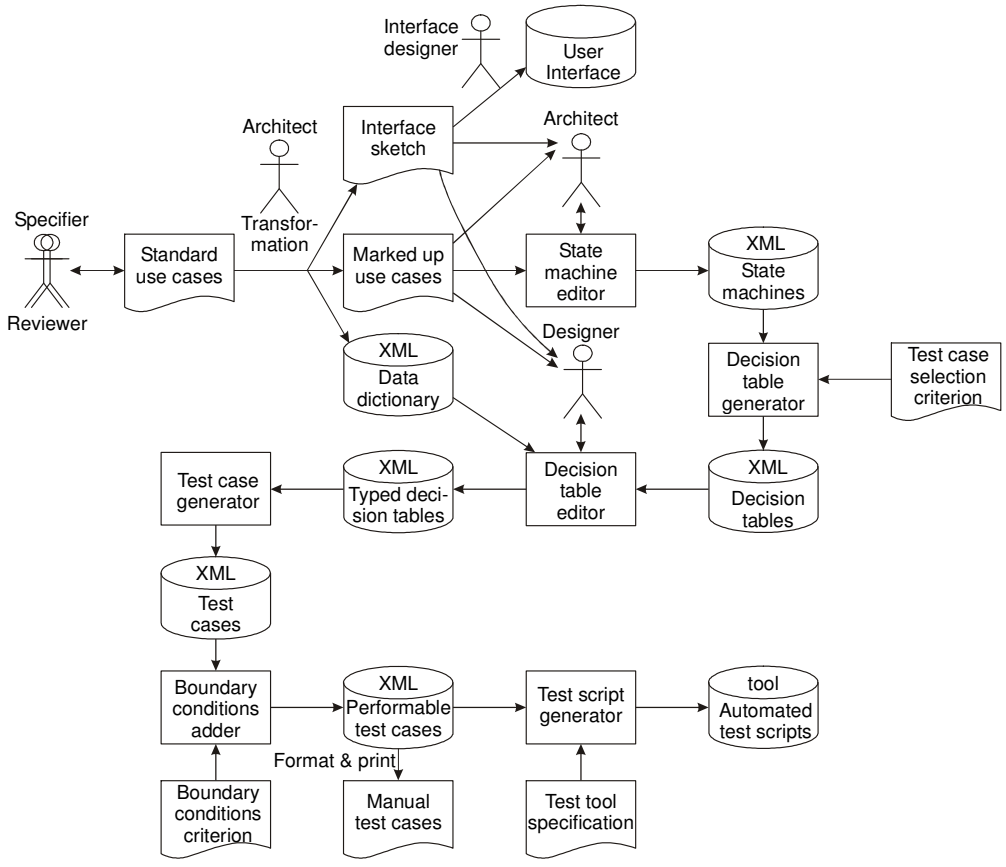

Figura 3 – Processo de Testes

Esse trabalho irá contémplar uma parte desse processo, sendo que futuramente serão realizados trabalhos para completar o processo. O processo de teste utilizado neste trabalho é o apresentado na figura 4. Cada um dos módulos apresentados são independentes um dos outros e a única comunicação existente é realizada através de arquivos XML. Essa independência permite que qualquer um dos módulos possa ser substituído, desde que a saída, ou seja, o arquivo XML obedeça ao formato estabelecido. Por exemplo, em um processo de teste que gera scripts JUnit, caso se queira substituir o formato do script utilizado para passar a utilizar Selenium, é necessário somente criar um módulo de geração de scripts para esse formato e substituir o gerador de scripts JUnit utilizado anteriormente. Isso é importante, pois garante uma maior flexibilidade ao processo.

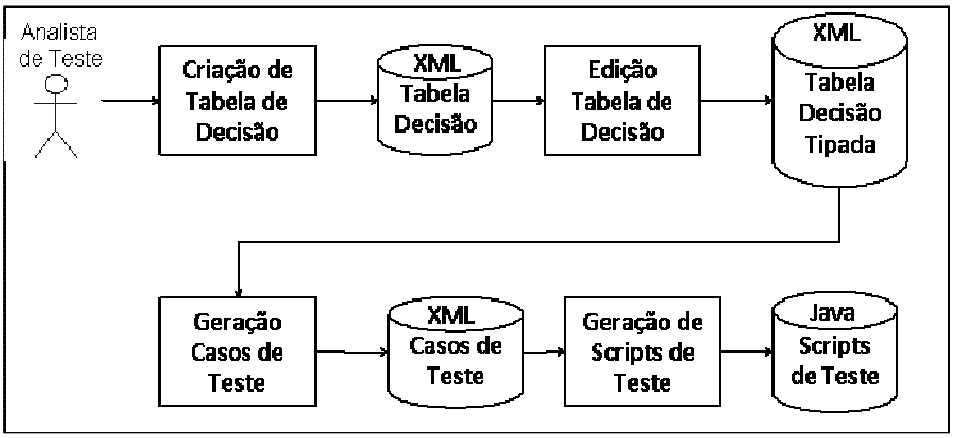

Figura 4 – Processo de Testes

O processo inicia com a criação de uma tabela de decisão. Essa tabela de decisão irá representar o comportamento de uma determinada interface do sistema a ser testado e será armazenada em um arquivo XML.

A segunda atividade tem como responsabilidade atribuir tipos para os campos da interface descrita na tabela de decisão e gera um arquivo XML que contém a tabela de decisão tipada.

Esse segundo arquivo XML é utilizado como entrada para a próxima atividade, que é a geração dos casos de teste. Esse módulo irá gerar um arquivo XML que contém todos os casos de teste e cada caso de teste possui o valor de cada campo e o oráculo do teste.

A descrição mais detalhada de cada uma das atividades está descrita nas seções subseqüentes. Para a realização de cada uma dessas etapas foi

desenvolvida uma ferramenta que auxilia o processo. Todas essas ferramentas estão descritas no capítulo 3.

3.2.Criação de Tabela de Decisão

A primeira atividade do processo é a geração da tabela de decisão. Para tal, é necessário que um integrante da equipe de teste leia os requisitos para uma determinada funcionalidade e descreva as condições e ações associadas à funcionalidade em uma tabela de decisão. Como foi dito anteriormente uma das características mais importantes da tabela de decisão é a possibilidade da verificação de completeza e não ambigüidade. O resultado desse processo será uma tabela de decisão descrita em um arquivo XML.

Para auxiliar essa atividade foi desenvolvida a ferramenta ETD em Java para fazer a criação de tabelas de decisão e a verificação da completeza e não ambigüidade da mesma.

3.3.Edição de Tabela de Decisão

Para possibilitar a geração dos casos de teste e correspondentes scripts de teste, é necessária a adição de algumas informações relativas aos campos da interface e que inicialmente não estão descritas na tabela de decisão. As informações sobre o campos são relativos aos tipos de valores assumidos e também de comportamento individuais que serão detalhadas no próximo parágrafo. Para adicionar essas informações foi desenvolvida a ferramenta ETD que permite a gerar um arquivo XML contendo informações da tabela de decisão igual estava descrito no primeiro arquivo XML e informações adicionais relativa aos campos.

Essas informações adicionais estão sempre relacionadas com os campos da interface a ser testada. Para todos os campos é informado qual seu tipo de componente gráfico, sendo que as opções possíveis são: campo texto, botão, lista, *combo box*, *radio button* ou *check box*.

Além disso, alguns componentes gráficos possuem características especiais. Quando o componente é do tipo campo texto, existem duas informações adicionais: o tipo de valor que o campo aceita e a regra de preenchimento do

campo. Essa regra só é preenchida quando o tipo do valor é string e essa regra é na verdade uma expressão regular para representar a formatação do campo. O componente do tipo lista e *combo box* tem três informações adicionais: os valores válidos, os valores não válidos e se o campo permite seleção múltipla.

3.4.Geração Automática de Casos de Teste

Para a geração automática dos casos de teste são necessários dois passos: a geração de quais serão os casos de teste e os dados para cada caso.

Os casos de teste são extraídos da tabela de decisão. Cada coluna da tabela de decisão representa um caso de teste. Para realizar a geração automática de dados de teste, serão utilizadas as condições da tabela de decisão criada anteriormente. É importante ressaltar que os dados são sempre gerados a partir de uma tabela verificada, ou seja, completa e não ambígua. Por esse motivo, os relacionamentos entre as condições não restringem a geração dos dados, já que eles são controlados quando do preenchimento e validação da tabela realizada na primeira etapa.

A geração de dados é realizada de acordo com o tipo de campo descrito no arquivo XML da tabela de decisão tipada.

Quando o tipo do campo é numérico uma restrição é a faixa limite. Para gerar esse dado, é necessário apenas gerar um número aleatório que respeite o limite do caso de teste. Lembrando que como são gerados dados para cada caso de teste, os limites de cada variável irão variar. Quando o campo for do tipo String serão utilizadas gramáticas para gerar valores para esse campo. Essas gramáticas serão recuperadas do arquivo XML.

A geração de dados também depende do tipo de componente da interface que está sendo testado. Para os dois tipos apresentados anteriormente, será utilizado o componente campo de texto. Para geração de dados para componentes do tipo *radio button, check box* só é informado qual opção deve ser selecionada. Por último, quando o componente é do tipo lista, existirá no arquivo de entrada a definição de quais elementos da lista são válidos, quais não são e se é possível selecionar mais de um elemento.

No final, essa ferramenta irá gerar um arquivo XML contendo todos os casos de teste.

3.5.Geração Automática de Scripts de Teste

Para realizar testes funcionais, existem duas formas: utilizar a técnica de *capture e replay* ou programar teste para interfaces gráficas. Como os scripts são gerados automaticamente utilizamos a segunda forma. A segunda forma é normalmente utilizada para processo de desenvolvimento baseado em testes, ou seja, quando uma interface é desenvolvida ela já é testada. Isso acarretará algumas modificações do script e também do código, já que utilizamos em nosso experimento um sistema legado. Essas modificações serão detalhadas na seção de experimento.

A primeira atividade da etapa foi uma pesquisa das ferramentas existentes para que se pudesse escolher a mais adequada para o projeto. Foram encontradas diversas ferramentas que fazem teste de interfaces gráficas para Java. A tabela 3.1 apresenta as características das ferramentas. Das 8 ferramentas grátis avaliadas, apenas 3 delas possuem todas as características importantes para o projeto, são elas: FEST, Abbot e UISpect4J. Essas três ferramentas foram testadas e a escolhida foi a FEST, apenas por permitir a geração dos scripts de maneira mais fácil.

Softwares	Suporta JUnit	Suporta TestNG	Recuperação de elementos da interface	Faz *Capture Replay*	Gera Relatório	Funcionalidades Adicionais
FEST	X	X	X		X	Gera scripts XML
Abbot	X		X	X		Possui Editor próprio
Frankestein				X	X	Interligação com linguagem Ruby
Jemmy			X			
JFCUnit	X					
Marathon				X	X	Script gerados em python ou ruby
UISpect4J	X	X	X			Possibilita integração com dados escritos em CSV, XML.
Jacareto				X		

Tabela 3.1 – Comparativo entre ferramentas de teste automatizado.

Será gerado um script de teste no formato JUnit em um arquivo .java para cada tabela de decisão, e cada caso de teste está descrito em um método dentro do script. Para realizar a geração automática dos scripts de teste foi utilizado como

arquivo de entrada o arquivo XML contendo os casos de teste e dele foram extraídos todas as informações necessárias para a construção do script.

Como foi dito anteriormente, existem algumas modificações que são necessárias antes de executar o script. Uma delas é a inserção da instanciação da interface a ser testada no script gerado. Essa linha não é gerada automaticamente porque seria necessário o conhecimento dos parâmetros para instanciar a interface.

Outra modificação necessária, só que nesse caso não é no script gerado e sim no código da aplicação, é inserir para todos os elementos gráficos um nome especifico que é o nome das condições descritas na tabela de decisão. Esse nome é utilizado no script para reconhecimento dos componentes gráficos. Essas modificações são um trabalho manual que precisa ser realizado, mas que só precisa ser realizado uma vez por versão de software. Acredita-se que o tempo gasto com esse trabalho manual seja bem inferior ao trabalho manual de geração de testes, logo o processo de teste apresentado nesse trabalho ainda seria melhor do que a geração manual.

3.6. Validação do Processo

Para realizar a validação do processo foram realizados dois tipos de experimentos. O primeiro tipo consiste na construção de seis pequenos programas para testar se o processo de teste funciona. Cada um dos programas tem como objetivo testar um componente gráfico diferente. Os componentes gráficos testados foram: campo numérico, campo *string*, *checkbox*, *combobox*, lista e *radio button*. No capítulo 5 são apresentados, para cada um desse programas desenvolvidos, os *outputs* do processo de teste desenvolvido nesta dissertação que são: a tabela de decisão, o arquivo XML da tabela de decisão, o arquivo XML com os casos de teste e o arquivo contendo o script de teste gerado. O segundo tipo de experimento era fazer a validação do processo em um sistema em produção.