7 Conclusão e Trabalhos Futuros

O teste é uma etapa importante no desenvolvimento de software. Quando realizado de forma apropriada pode identificar uma grande parcela dos defeitos contidos no software, independentemente deles terem por origem as especificações, a arquitetura, o projeto, ou o código. Ou seja, um teste somente diz o seguinte "encontrei as seguintes falhas", ou então "não encontrei falhas". Neste último caso, o software pode conter defeitos não explorados. Isso enfatiza a necessidade do teste ser bastante minucioso para que se torne um instrumento confiável de controle da qualidade. A automação da geração e da execução dos testes contribui para uma melhoria da confiabilidade. A remoção dos defeitos causadores das falhas encontradas contribui então para uma sensível melhoria da qualidade do software entregue. Mas, como mencionado, um teste mal feito pode não reportar falhas – baixa eficácia –, embora o software contenha um grande número de defeitos. Também é mencionado que as ferramentas ainda tendem a ser pouco eficazes. Em virtude disso, e possivelmente devido a falhas de formação dos desenvolvedores e testadores, o processo de teste acaba sendo manual na maior parte das vezes. Isso faz com que haja um retrabalho grande a cada versão do software, pois os testes deveriam ser refeitos, o que, devido ao custo, acaba sendo postergado e depois esquecido. Além disso, é comum não existir uma documentação apropriada para verificar o que foi testado e, se necessário, repetir os testes.

7.1.Contribuições

Neste trabalho foi apresentado um novo processo de teste que utiliza como ferramenta auxiliar uma tabela de decisão para a geração automática de casos de teste. As contribuições do trabalho foram as seguintes:

- Desenvolvimento de um editor e verificador de tabelas de decisão
- Criação e validação do processo de teste baseado na geração automática e na execução automática dos casos de teste.

- Desenvolvimento de um gerador de dados aleatórios a partir de tabelas de decisão,
- Desenvolvimento de um gerador de scripts de teste com base nos dados gerados
- Realização de experimentos para a avaliação do processo e das ferramentas desenvolvidas.

A primeira contribuição desse trabalho foi o desenvolvimento de uma ferramenta de edição de tabela de decisão (ETD) que valide a sua completeza e não ambigüidade. Para a área de teste, essa ferramenta permite garantir que todos os casos de teste sejam descritos na tabela de acordo com as condições preenchidas. Como foi visto, no capítulo do estado da arte, a tabela de decisão pode ser utilizada para outras áreas como gerenciamento de regras de negócio e essas áreas também poderiam utilizar o ETD.

Com o gerador de dados aleatórios foi possível gerar dados de acordo com as restrições apresentadas através da tabela de decisão. É importante frisar que o gerador de dados aleatório é independente da tabela de decisão, sendo apenas necessário gerar um arquivo XML com um formato específico. Os dados gerados podem ser ações, como selecionar um botão, uma restrição numérica, como por exemplo x > 10 ou gerar uma *string* a partir de uma expressão regular. Essa última geração ainda não é amplamente desenvolvida e é uma funcionalidade particularmente importante, para tal foi construída uma função que interpreta uma expressão regular e gera *strings* que satisfaçam essa expressão.

Ao gerar o script foi observado que é possível gerar um script JUnit a partir de um arquivo XML descrevendo os casos de teste. Essa contribuição é importante, pois, como JUnit é uma ferramenta automatizada, isso faz com que diminua bastante o esforço para a construção de scripts de teste e a posterior execução dos testes.

Essas três contribuições permitiram a criação de um processo de teste semi-automatizado. Um aspecto muito importante de possuir um processo semi-automatizado é possibilitar a geração da quantidade desejada de scripts e que cada um dos scripts vai ter um conjunto de dados diferentes de modo que a execução percorra caminhos diferentes no código. Isso facilita a encontrar falhas que dependem do preenchimento de um campo com um valor específico, além de não

fixar os dados do teste (diferentes gerações podem gerar diferentes testes) e não viciar os testes com análises subjetivas.

É importante ressaltar que esse processo apresentado pertence a um processo maior que pretende gerar testes a partir de uma especificação. O processo completo está descrito figura 62.

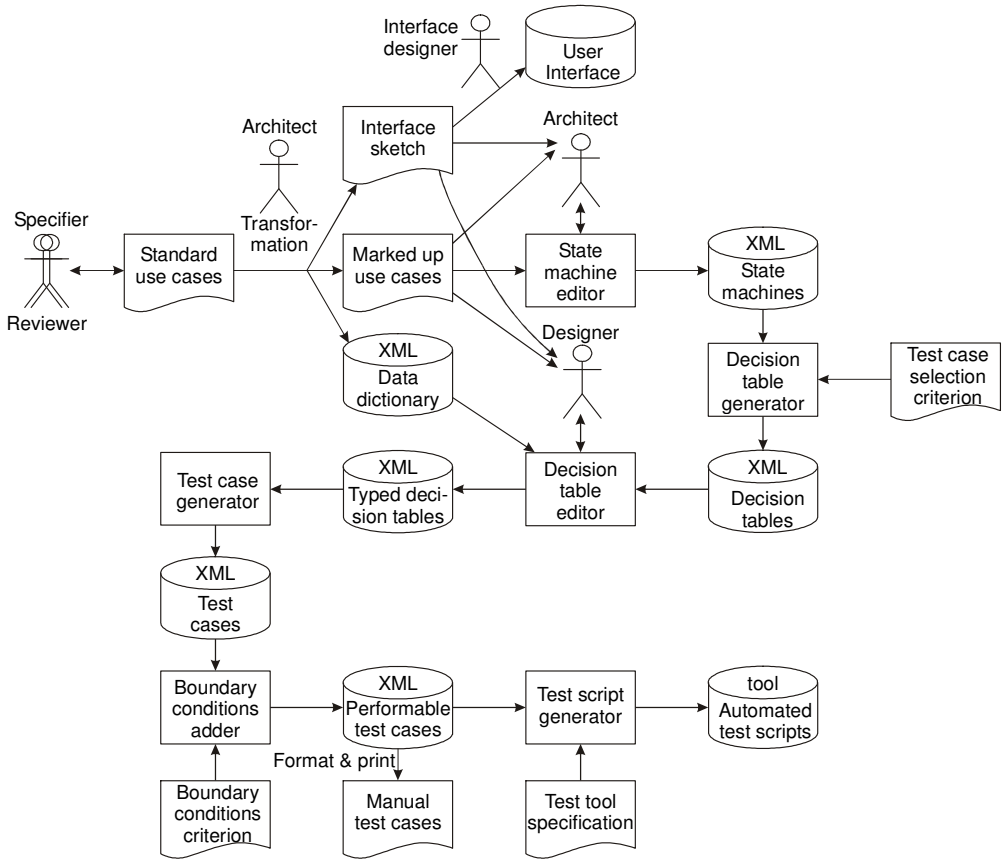

Figura 62 – Processo completo

De acordo com o processo almejado, este trabalho começou a partir do processo *"Decision Table Editor"* e realizou o processo até o fim, gerando o *scripts* de teste automatizados e os casos de teste.

O desenvolvimento dos componentes foi feita de forma modularizada, de modo que pudessem ser substituídos com facilidade, tornando o processo mais adaptável a outros ambientes de desenvolvimento.

De acordo com [BERTOLINO, 2007] 100% do processo de automação de geração e execução de testes ainda é uma aspiração. O processo apresentado está bem perto de conseguir chegar a esse estado tão desejado.

A validação desse processo foi feita inicialmente com diversos pequenos casos, descritos no capitulo 5, sendo que cada caso exercita uma interface construída especificamente para o teste contendo um tipo de específico de componente gráfico. Esses testes iniciais foram importantes para mostrar que o processo funciona independente dos componentes gráficos utilizados na interface. A segunda validação foi feita através da aplicação do processo de teste a um sistema em produção. Essa etapa foi fundamental para mostrar a aplicabilidade do processo no mundo real. Também foi verificado que a cobertura dos testes gerados a partir da tabela de decisão é maior que a realizada em testes manuais, apesar de somente algumas das características terem sido testadas. As linhas de código que não foram percorridas pelos testes foram principalmente métodos *geters* e *seters* e métodos que faziam a construção das interfaces. Também foi observado pelo experimento realizado que os testes gerados pelo processo reportaram 3 falhas de corretude, enquanto que o processo manual encontrou apenas uma falha e 4 melhorias. Um fator que pode ter influenciado a encontrar as falhas foi a diversificação dos dados gerados, não necessariamente o teste manual é realizado com o mesmo grau de abrangência considerando as possibilidades dos dados. Além disso, o segundo experimento mostrou que o esforço gasto para a construção de uma tabela de decisão ainda é compensando pelo esforço que é necessário para a especificação dos casos de teste. Outro detalhe importante ressaltado nesse segundo experimento foi que o tempo gasto na execução também foi menor, já que um é realizado automaticamente e outro manualmente.

Evidentemente o experimento é pequeno e mais experimentos seriam necessários para comprovar a real eficácia do processo. Entretanto, apesar de singelo, já mostrou ser mais eficaz do que o teste manual já que as falhas encontradas pelo novo processo também deveria ter sido encontradas pelo processo manual. Cabe observar que os defeitos causadores das falhas são defeitos originais e não defeitos inseridos através de mutantes.

7.2.Trabalhos Futuros

O processo criado foi testado com uma funcionalidade relativamente simples. É importante para melhor comprovação da melhoria que esse processo pode trazer realizar outros experimentos com sistemas em produção. Desta vez

seriam usadas funcionalidades mais complexas que podem incluir regras de negócio que não estão associadas diretamente ao preenchimento dos campos como, por exemplo, registros duplicados. Esse teste comparativo com o banco ainda não é realizado no processo.

Outro aspecto que deve ser examinado é a flexibilidade dos módulos que compõem o processo. Uma forma seria criar um gerador de scripts para outra ferramenta ou framework de teste automatizado como, por exemplo, o Rational Funcional Tester da IBM.

A principal complementação para esse trabalho é conseguir construir o processo inteiro como está descrito no capítulo 2, onde esse trabalho seria a etapa final do processo de geração de casos de teste.

Esse processo poderia ser utilizado no futuro como um processo de desenvolvimento dirigido por teste de aceitação (ATDD). A tabela de decisão pode ser gerada antes do código ser realizado. Com isso os testes de aceitação podem ser gerados e, conforme o desenvolvimento está sendo feito, os componentes seriam testados.

Um trabalho futuro que poderia ajudar na documentação dos casos de teste, é gerar um passo a passo do que está sendo feito pelo script. Isso pode também ser utilizado se fosse preciso realizar testes manuais, neste caso pelo menos já existiria um guia para o testador.