1
Introdução

O crescimento do uso de software e o aumento de complexidade, tamanho, heterogeneidade, autonomia, distribuição física e dinamismo de sistemas computacionais afetam diretamente a qualidade destes sistemas (A-MOST 2007). Neste cenário, garantir que produtos e serviços oferecidos pela indústria de software apresentem menos defeitos é um desafio cada vez maior.

A prática de teste de software é apontada como parte da solução para o problema de qualidade caracterizado. Os principais objetivos de um teste são encontrar falhas e promover a conseqüente remoção completa e correta do defeito causador. É objetivo também demonstrar a correta execução do sistema testado segundo a ótica do usuário (Jorgensen, 1995). Portanto, se testes forem aplicados desde a concepção até a implantação do sistema, então se espera que seja viável identificar e corrigir os defeitos tão logo eles sejam percebidos. Evita-se o acúmulo de defeitos e o provável encadeamento de defeitos. Logo, os desenvolvedores terão maior controle sobre a qualidade do que é produzido e oferecido ao mercado, além de reduzirem o volume de retrabalho inútil.

Esta abordagem de antecipação da realização de testes (Beck, 2003) apresenta forte aceitação da comunidade de desenvolvedores de software. É conhecida pelo nome de desenvolvimento dirigido por testes (*test-driven development*). A idéia principal é realizar testes dos artefatos gerados ao longo de todas as fases de produção de um sistema. Não importa se é um requisito, um modelo, um código ou um executável, tudo será testado. O que varia são as ferramentas e as técnicas aplicadas de acordo com o tipo de artefato e o relacionamento entre artefatos.

No desenvolvimento dirigido por teste, o próprio teste muitas vezes funciona como uma especificação do sistema testado. A redação dos testes sob a forma de casos de teste obriga a especificação ser plenamente verificável, contribuindo assim para remover uma boa parte dos defeitos contidos em especificações. Como as estatísticas (Boehm & Basili, 2001) mostram que 20%

do total de defeitos num sistema é a fonte de 80% de retrabalho inútil decorrentes de especificações incorretas, essa é uma boa expectativa de redução de custos.

Dentre as técnicas utilizadas para realização de testes, a que tem o propósito de verificar a conformidade de um sistema com os requisitos definidos em sua especificação é o teste funcional. O teste funcional é classificado como um teste do tipo caixa-preta (*Black-box testing*) em que não é necessário ter o conhecimento do projeto do código ou de sua lógica para formular os casos de teste. Num teste funcional, o sistema em teste é considerado uma função que mapeia valores do domínio de entrada em valores de saída (Jorgensen, 1995). O teste independe da implementação e pode ser desenvolvido em paralelo com a programação.

Embora testes funcionais sejam importantes para validação e certificação de sistemas e para aceitação e teste de integração de componentes, alguns estudos indicam que não são suportados adequadamente pelas ferramentas existentes (Andrea 2007). Este fato sugere uma oportunidade a ser explorada em prol de usuários, desenvolvedores e testadores de sistemas.

1.1. Definição do Problema

O desenvolvimento de software dirigido por testes pode focalizar testes funcionais (*functional test-driven development*). Segundo esta proposta (Andrea, 2007), a qualidade dos sistemas de software está diretamente relacionada à identificação de falhas no atendimento de seus requisitos funcionais. Tais requisitos podem ser diretamente relacionados pelos participantes do projeto de um sistema ou podem ser obtidos a partir da análise de sua especificação. Esta especificação é constituída de documentos como a descrição de casos de uso, os modelos de projeto e a definição da interface pública de unidades de programação. Outra fonte de requisitos são os dados armazenados em ferramentas de gerência de problema e gerência de defeito (*issue tracking and defect tracking*).

Nesse contexto, garantir a qualidade do sistema significa garantir a conformidade e a consistência de seu projeto e de sua implementação com seus requisitos funcionais, mesmo quando estes sofram mudanças. Para assegurar a qualidade do sistema, é necessário avaliá-lo constantemente ao longo de seu ciclo

de vida, que vai desde o seu desenvolvimento até a sua implantação e execução num ambiente de software. Isto implica o teste funcional do sistema em diferentes níveis de concretude de sua representação. Na concepção do sistema, os conceitos e os comportamentos são mais genéricos e abstratos. Durante a sua elaboração e construção, estes se tornam cada vez mais especializados e concretos. Na transição para o usuário, eles deveriam ser plenamente concretos.

Para garantir a qualidade ao longo do ciclo de vida de um sistema é preciso estabelecer uma forma de teste funcional que seja capaz de verificar este sistema segundo a sua especificação e considerar os diferentes níveis de concretude de sua representação nesta verificação. Ao adotar a especificação como referência, estabelece-se um critério de análise de completeza do teste. Ao tratar da representação do sistema, torna-se a estrutura de teste apta a evoluir e a acompanhar o amadurecimento deste sistema.

Nas etapas iniciais do desenvolvimento de um projeto de software, o custo associado à resolução de defeitos é reconhecidamente menor, visto que o impacto sobre o que já foi efetivamente produzido é pequeno. Corrigir a especificação de um projeto na sua fase embrionária causa muito menos transtornos do que alterar código produzido ou programas já implantados. "Encontrar e consertar um problema após a entrega é 100 vezes mais caro que encontrar e consertar este problema durante as fases de requisitos e projeto" (Boehm & Basili, 2001). Há um consenso entre a academia e a indústria de que é necessário definir uma solução que ataque o problema o mais cedo possível no ciclo de desenvolvimento (Hartman, 2002).

1.2. Solução Proposta

Para avaliar funcionalmente sistemas de software, a solução proposta nesta dissertação recorre à prática de teste baseado em modelo (*model-based testing*) descrita em alguns trabalhos da literatura (El-Far, 2001). Nesta prática, a representação de conceitos e comportamentos do sistema em teste, assim como do próprio ambiente, é feita por meio de modelos.

Modelos são descrições ou analogias usadas para ajudar a visualizar algo que não pode ser diretamente observado (*Merriam-Webster Dictionary*). Modelos

podem apresentar diferentes níveis de concretude. Quando empregados como base para geração e execução de testes de um sistema, modelos precisam ser concretos o suficiente para avaliar os aspectos relacionados aos objetivos de teste. Um processo de teste baseado em modelos considera os conceitos envolvidos, e é comum que omissões no modelo signifiquem que partes omitidas não possam ser testadas (Utting et al., 2006).

Dentre os tipos de modelos existentes, os modelos construídos a partir de gramáticas serão os modelos de teste adotados pela solução discutida neste trabalho. A escolha desta representação foi influenciada por alguns aspectos como a facilidade de validação do próprio modelo de teste e a capacidade de estruturar os testes, que compensam algumas limitações conhecidas existentes.

A necessidade de validar o modelo de teste implica no fato de que o modelo deve ser mais simples que a programação do sistema em teste, ou pelo menos mais fácil de verificar, modificar e manter (Utting et al., 2006). Do contrário, os esforços de validação do modelo serão iguais aos esforços de validação do sistema em teste. Como é fácil escrever (gerar) e interpretar (legível para seres humanos) uma gramática, ela atende a estes requisitos (Duncan & Hutchinson, 1981).

A estrutura dos testes tem impacto direto na sistematização da solução, pois é com base nesta estrutura que são gerados e executados os testes tanto positivos quanto negativos. Um teste positivo tem a conotação de um "teste correto", onde o resultado observado deve ser válido. Um teste negativo tem a conotação de um "teste errado". Normalmente o objetivo, neste caso, é forçar um resultado inválido como resultado observado. Gramáticas podem ser utilizadas para descrever os conceitos e comportamentos de um sistema, as configurações do ambiente de software e descrever um conjunto de testes.

A sistematização da solução define um processo de teste funcional onde as especificações do sistema e os testes propriamente ditos são modelados usando gramáticas. Uma arquitetura de software de apoio utiliza os modelos gramaticais para geração de dados de entrada empregados no teste de funções do sistema ou de suas partes. Essas funções são executadas e o seu resultado é contrastado com o resultado estimado pela gramática de teste. De posse da identificação ou não de falhas, reveladas pelas diferenças entre resultados estimados e obtidos, é tomada uma decisão quanto à execução de novos testes, modificação dos testes e parada dos testes. A avaliação pode ser meramente semântica (independente de

plataforma) ou pode ser uma avaliação de uma implementação específica (dependente de plataforma).

A solução apresenta limitações decorrentes do uso de gramáticas e da técnica de teste funcional. A geração de dados de teste pode sofrer o problema de explosão combinatorial para se estabelecer um determinado grau de cobertura de funcionalidade. A falta de documentação dificulta a elaboração dos testes sem auxílio do usuário ou desenvolvedor. As interfaces públicas do sistema usadas pela estrutura de teste para interagir com mesmo a fim de validar os requisitos da especificação podem não permitir acessar o estado do sistema, tornam difícil ou até impossível a verificação de conformidade das funções disponíveis.

1.3. Objetivos

Este trabalho propõe o uso de gramáticas na condução de testes funcionais de sistemas com alguns objetivos bem definidos: sistematizar a geração e a execução de testes; permitir a adoção de diferentes estratégias de teste; facilitar a adaptação dos testes a mudanças de requisitos e reaproveitar testes gerados.

A sistematização da geração e da execução de testes prepara as bases para a automação do processo de teste. Com a automação se obtém uma redução significativa do custo de identificação-remoção de faltas e uma redução do impacto das atividades de teste no desenvolvimento e na operação de um sistema (Hartman, 2002).

A aplicação de diferentes estratégias confere flexibilidade à solução de teste na medida em que diversos algoritmos e heurísticas podem ser usados de acordo com o objetivo de teste. Como o teste de software é uma atividade usualmente experimental e baseada em heurísticas (Duncan & Hutchinson, 1981), permitir a construção de heurísticas sob a forma de modelos baseados em especificações amplia a capacidade de avaliação da solução.

A adaptação a mudanças de requisitos e o reaproveitamento de testes gerados garantem a evolução e a manutenção da solução de teste.

1.4. Resultados

Como conseqüência do estudo sobre o teste funcional e o uso de modelos para orientar a condução dos testes de um sistema computacional, a presente pesquisa acabou propondo um processo de teste funcional (Figura 1) divido em fases, cada uma composta por atividades (capítulo 4).

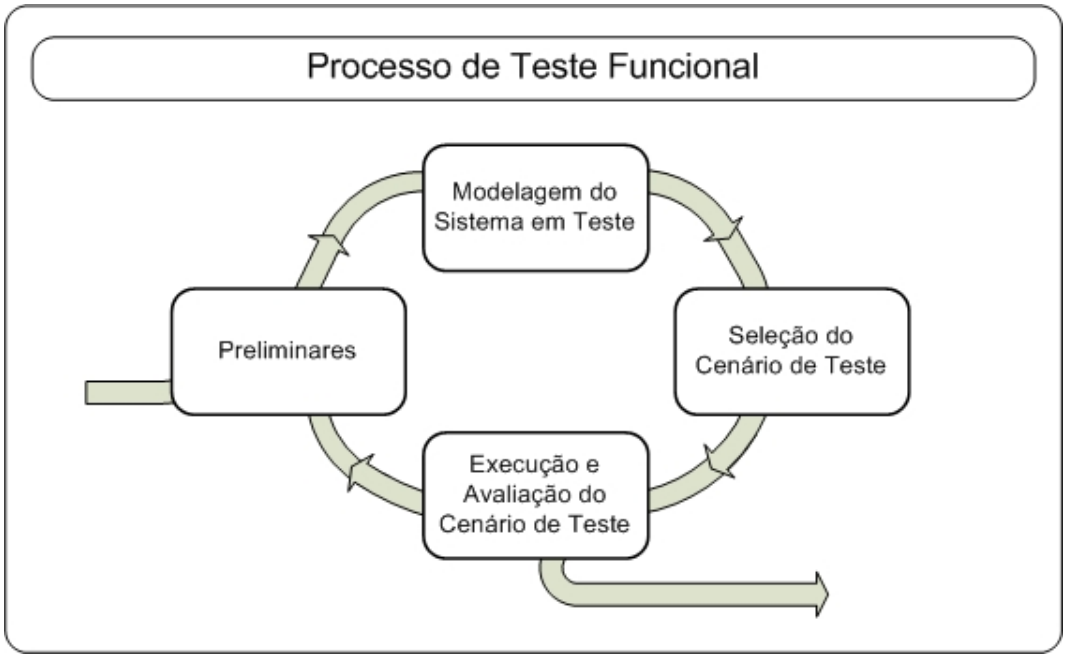

Figura 1 – Processo de Teste Funcional

Este processo de teste é apoiado por uma arquitetura de software baseada em modelos construídos a partir de gramáticas (capítulo 3). As gramáticas definem como os modelos capturam informações sobre a estrutura e o comportamento dos sistemas. Também definem como são gerados os testes do sistema e como é o seu ambiente de execução. Os modelos gramaticais obtidos em conjunto com uma estratégia de teste funcional (capítulo 2) compõem um cenário de teste do sistema. Este cenário apresenta uma representação computacional que é resultante do projeto dos modelos gramaticais segundo o paradigma orientado a objetos.

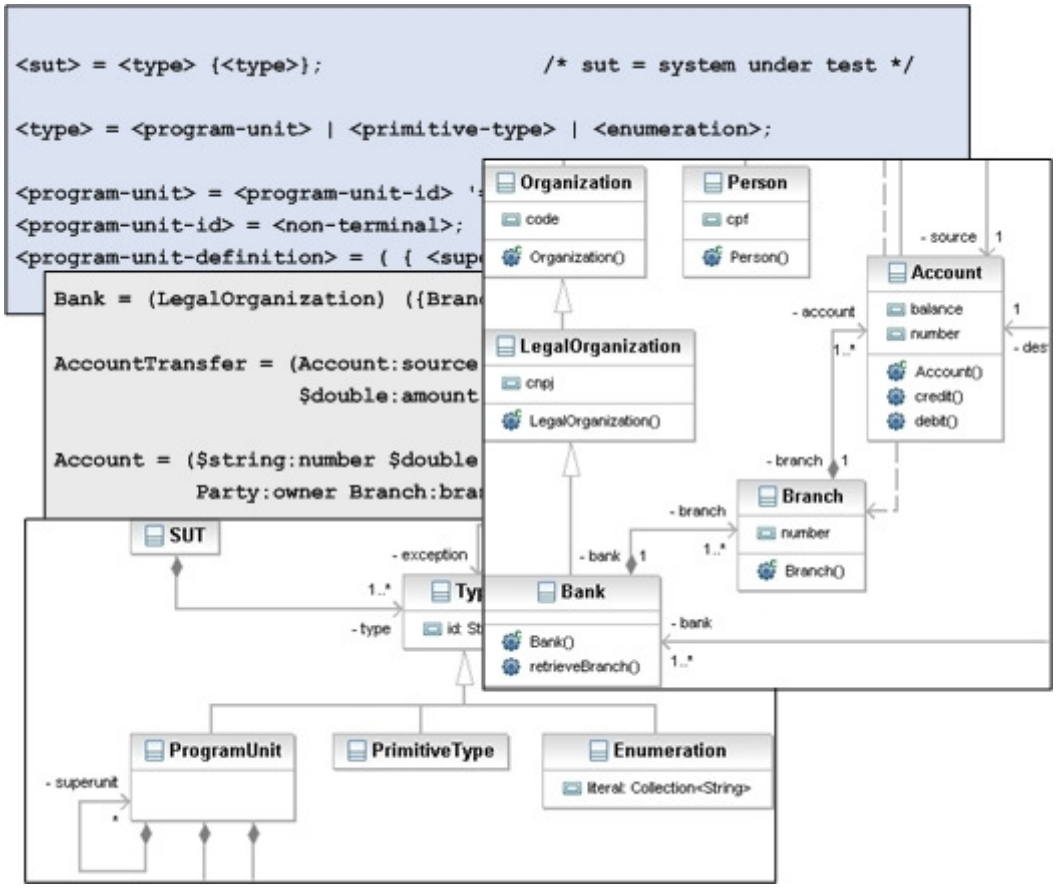

Figura 2 – Gramática de Teste; Modelo Gramatical; Modelo Computacional; Sistema

A partir da interpretação destes modelos é possível gerar testes conceituais e executáveis do sistema. Tais testes são sensíveis às mudanças de requisitos e apresentam fácil adaptação dado que basta alterar apenas os modelos gramaticais na sua forma textual, visto que a arquitetura de software garante a evolução da parte computacional.

A pesquisa apresenta uma abordagem nova que ainda precisa amadurecer, mas que propicia um campo para a realização de novas pesquisas. Tais pesquisas estão relacionadas à validação do processo proposto em domínios de teste diferente, à evolução da arquitetura de software de apoio, ao refinamento dos algoritmos correspondentes às estratégias de testes, para citar algumas possibilidades.

1.5. Organização da Dissertação

Este documento foi dividido em capítulos que descrevem o processo de teste funcional baseado em modelos gramaticais, exemplos de sua aplicação e relacionamentos com outros trabalhos da comunidade acadêmica.

O ***capítulo 1*** introduziu a pesquisa. Definiu o problema evidenciado. Propôs uma solução que será delineada e detalhada ao longo desta dissertação. Apresentou brevemente os resultados obtidos.

O ***capítulo 2*** aborda a prática de teste de software. Descreve a terminologia utilizada. Caracteriza o teste funcional. Retrata o teste baseado em modelo e suas diferenças em relação ao teste dirigido por modelo. Aponta alguns dos desafios identificados como relevantes aos testes funcionais.

O ***capítulo 3*** trata dos modelos gramaticais que serão usados na geração e execução dos testes. Define a gramática de teste usada na composição dos modelos de teste. Cita os trabalhos relacionados que serviram de fonte de inspiração para modelagem. Descreve os modelos de teste usados no processo de teste funcional.

O ***capítulo 4*** exibe o processo de teste funcional. Identifica suas fases e atividades. Descreve a arquitetura projetada e desenvolvida para apoiar o seu emprego. Utiliza exemplos para consolidar a técnica e os conceitos usados.

O ***capítulo 5*** apresenta um estudo de caso elaborado em concordância com o processo de teste funcional proposto. Destaca o procedimento de modelagem e a realização das principais atividades do processo de teste.

O ***capítulo 6*** discute brevemente a conformidade entre o teste baseado em gramática e uma especificação de referência. Contrapõe este teste e outras abordagens de teste. Indica algumas contribuições obtidas por esta pesquisa.

O ***capítulo 7*** conclui este trabalho, analisando os resultados obtidos e destacando algumas futuras linhas de pesquisas derivadas destes resultados.