UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DISSERTAÇÃO DE MESTRADO

ABORDAGENS PARA AVALIAÇÃO EXPERIMENTAL DE TESTE BASEADO EM MODELOS DE APLICAÇÕES REATIVAS

LAÍSA HELENA OLIVEIRA DO NASCIMENTO

CAMPINA GRANDE – PB
FEVEREIRO DE 2008

Abordagens para Avaliação Experimental de Teste Baseado em Modelos de Aplicações Reativas

Laísa Helena Oliveira do Nascimento

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Engenharia de Software

Patrícia Duarte Lima Machado

(Orientadora)

Campina Grande, Paraíba, Brasil

N244a

2008 Nascimento, Laísa Helena Oliveira do.

Abordagens para avaliação experimental de teste baseado em modelos de aplicações reativas / Laísa Helena Oliveira do Nascimento. - Campina Grande, 2008.

71 f.: il.

Dissertação (Mestrado em Ciência da Computação), Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

Referências.

Orientadora: Profª. Drª. Patrícia Duarte de Lima Machado.

1. Engenharia de Software. 2. Modelos de Medição. 3. Teste Baseado em Modelos. I. Título.

CDU – 004.41(043)

"ABORDAGENS PARA AVALIAÇÃO EXPERIMENTAL DE TESTE BASEADO EM MODELOS DE APLICAÇÕES REATIVAS"

LAÍSA HELENA OLIVEIRA DO NASCIMENTO

DISSERTAÇÃO APROVADA EM 28.02.2008

PROFª PATRÍCIA DUARTE DE LIMA MACHADO, Ph.D
Orientadora

PROF. JOSÉ ANTÃO BELTRÃO MOURA, Ph.D
Examinador

PROF. PAULO HENRIQUE MONTEIRO BORBA, Ph.D
Examinador

CAMPINA GRANDE – PB

Resumo

Processos de teste de *software* vêm ganhando cada vez mais espaço na indústria. Empresas têm investindo na definição e formalização dos seus processos e em meio a essa mudança de comportamento, *Model-Based Testing* (MBT) apresenta-se como uma técnica promissora de teste. No entanto, a utilização de MBT ainda é baixa e pesquisadores têm focado em maneiras de superar as barreiras para que se obtenha uma adoção maior por parte da indústria. O mundo empresarial é movido a processos e resultados. Dessa forma, o uso de MBT precisa se adaptar aos processos existentes, e estudos de caso que evidenciem as vantagens de sua utilização precisam ser conduzidos. Neste trabalho, o paradigma *Goal Question Metric* é utilizado na definição de modelos de medição que têm como foco principal a avaliação e o acompanhamento do desempenho de MBT sem causar impacto ao processo de teste já existente. Os modelos de medição consideram métricas como esforço, percentual de requisitos testáveis cobertos, percentual de casos de teste modificados, percentual de falhas, dentre outros. Os modelos não estão atrelados ao processo de MBT apresentado, podendo ser aplicados em qualquer processo que permita a coleta dos dados necessários para o cálculo das métricas. Para validar os modelos, estudos de caso foram conduzidos dentro do ambiente de testes da Motorola.

Abstract

Software testing processes have become more common in industry. Companies are investing on the definition and the formalization of their test processes and, in this context, Model-Based Testing (MBT) appears as an interesting testing technique. However, industrial adoption of MBT remains low and researchers are also focusing on how to beat the barriers to wide adoption. Processes and results move the business world so, MBT processes must be adaptable to actual testing processes. For this, experiments to evaluate the results achieved with its use must be conduct. In this work, measurement models based on the Goal Question Metric methodology are proposed. The purpose is evaluating the use of MBT without increasing actual testing process costs. The models focus on aspects as effort, testable requirements coverage, modified test cases, failures, among others. The models are not associated with the MBT process presented. They can be applied with any process that allows metrics collection. In order to validate the measurement models, case studies were conducted into Motorola testing environment.

Agradecimentos

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização e Motivação

Processos de teste de *software* vêm ganhando cada vez mais espaço na indústria de desenvolvimento de *software*. As empresas têm investido na formação de profissionais especializados em teste e buscado definir e formalizar seus processos de teste. Em meio a essa mudança de comportamento, uma técnica promissora é *Model-Based Testing* (MBT), uma técnica *black-box*[1] de teste, onde atividades comuns do processo de teste, como geração de casos de teste e análise de resultados, se baseiam na especificação da aplicação sob teste [El-Far and Whittaker, 2001]. O foco do processo de aplicação de MBT concentra-se basicamente em duas etapas: criação de um modelo formal da aplicação sob teste e derivação dos casos de teste a partir desse modelo. Em outras palavras, uma das chaves do sucesso de MBT está no modelo utilizado, que precisa conter informação suficiente para criação dos casos de teste.

Dentre as vantagens prometidas por MBT, destacam-se a possibilidade de automação da geração dos casos de teste e o compartilhamento de um modelo único entre os vários times e pessoas envolvidas no processo de desenvolvimento. No entanto, a adoção de MBT pela indústria ainda é baixa e por isso pesquisadores têm também focado em maneiras de superar as barreiras para que se obtenha uma adoção maior pela indústria [Bertolino, 2007]. Nesse sentido, dois temas vêm sendo abordados: i) como combinar diferentes estilos de

[1] Processo de derivação e/ou criação de casos de teste com base na análise da especificação, funcional ou não-funcional, de um componente ou sistema sem referenciar sua estrutura interna [Graham et al., 2007].

modelagem (por exemplo, *transition-based*, *pre/post condition-based* e *scenario-based*); ii) como integrar MBT aos processos de desenvolvimento [Bertolino, 2007].

O consórcio AGEDIS [Hartman and Nagin, 2004] é um exemplo de iniciativa para o desenvolvimento de metodologias e ferramentas para automação de testes em geral. As ferramentas suportam uma metodologia baseada em MBT e apresentam um grau considerável de automação. Existem várias ferramentas de MBT na indústria e na academia, Hartman [Hartman, 2002] apresenta uma visão geral de algumas.

Embora a existência de um modelo único seja apontada como uma vantagem de MBT, a construção desse modelo acaba sendo um dos fatores que dificulta a adoção da técnica. O nível correto de abstração do modelo é crucial para o sucesso de MBT [Prenninger and Pretschner, 2005], uma vez que modelos muito abstratos podem não conter informação suficiente para derivar casos de teste concretos. Esse nível de abstração reflete a transformação que ocorre na prática: modelos são enriquecidos com informações detalhadas do comportamento da aplicação, de forma a facilitar a geração automática de casos de testes, ficando a idéia do modelo único reservada à teoria.

Outro ponto que conta negativamente para adoção de MBT é a resistência de arquitetos e projetistas em construir modelos utilizando alguma notação formal. TaRGeT [Nogueira et al., 2007] é um exemplo de ferramenta que utiliza a abordagem apresentada em Cartaxo [Cartaxo, 2006] para geração de casos de teste a partir de um modelo formal, mas abstrai do usuário final a escrita de modelos em notação formal, uma vez que adota a estratégia apresentada em Cabral [Cabral and Sampaio, 2006] para mapear cenários de uso escritos em linguagem natural para uma notação formal. O surgimento de ferramentas que tentam solucionar o problema de criação dos modelos pode impulsionar a adoção de MBT na indústria, porém pode não ser suficiente.

O mundo empresarial é movido a processos e resultados. Dessa forma, o uso de MBT precisa se adaptar aos processos existentes, e experimentos que evidenciem as vantagens de sua utilização precisam ser conduzidos. A experimentação na engenharia de *software* é importante para que saibamos quando e como determinadas técnicas funcionam, perceber suas limitações e entender como melhorá-las [Basili, 1996].

Segundo Travassos [Guilherme Horta Travassos, 2002], um experimento deve ser tratado como um processo da formulação ou verificação de uma teoria, e para que o processo ofe-

reça resultados válidos, ele deve ser propriamente organizado e controlado, ou pelo menos, acompanhado. Na prática, não é trivial conduzir um experimento dentro de um ambiente de produção. Uma das principais barreiras é a relutância dos gerentes de projeto em adaptar o processo corrente para que o experimento seja possível, visto que essa mudança pode afetar a produtividade do time.

Além da preparação do ambiente, a condução de um experimento envolve a definição dos dados que precisam ser coletados e em quais momentos, e a definição de significados para esses dados, ou seja, é preciso definir métricas, coletá-las e analisá-las. Sinha et. al [Sinha et al., 2006] apresenta um *framework* de medição para avaliar ferramentas de geração de casos de teste baseados em modelo, mas são necessários modelos de medição específicos para o processo de MBT.

Aplicações para celular são, em sua maioria, aplicações reativas. Segundo Acetato [Aceto et al., 2007], o termo sistema reativo foi criado por David Harel e Amir Pnueli para descrever sistemas que computam a partir da reação a estímulos provenientes do seu ambiente, como por exemplo, sistemas operacionais e sofwares embarcados em dispositivos móveis. Esse tipo de sistema possui alta interatividade, tendo boa parte dos seus fluxos de execução guiados por entradas externas. Frequentemente, aplicações reativas são testadas de forma manual. Nesse domínio de aplicação, o *time-to-market* dita o ritmo da produção, prejudicando o processo de testes, já que o tempo de desenvolvimento está cada vez mais curto. MBT torna-se interessante uma vez que antecipa o ínicio do processo, visto que o modelo pode ser gerado logo após a definição dos requisitos ou em paralelo [Dalal et al., 1999], e possibilita a redução do tempo e custo do processo de teste devido a automação da geração de casos de teste.

Mas, se MBT é uma técnica interessante, porque sua adoção por parte das empresas ainda é pequena? Ao nosso ver, um dos grandes problemas está concentrado na construção do modelo: que notação utilizar? Qual o nível de abstração que se deve adotar? O modelo deve ser baseado apenas nos requisitos? Muitas são as questões em aberto, e para respondê-las estudos experimentais são essenciais para medir e avaliar o desempenho de MBT, mas principalmente para entender o processo e seus problemas.

1.2 Objetivo do Trabalho

O objetivo principal do trabalho é a investigação e a proposta de abordagens para avaliação experimental de MBT no contexto de aplicações reativas, particularmente aplicações para celular. Nesse contexto, um tipo de teste usual é o teste de *feature*, que é aplicado às unidades de composição (*features*) de uma aplicação para celular. Nosso propósito é investigar o desempenho de MBT como técnica auxiliar aplicada no teste de *feature* e para isso, as seguintes metas foram traçadas:

- **Definição de um modelo de medição para avaliar os desempenhos de MBT e de teste exploratório no contexto de teste de *feature* para aplicações para celular**. MBT e teste exploratório são duas técnicas que apresentam características interessantes para serem utilizadas em sistemas reativos e cujos ciclos de teste se repetem ao longo do desenvolvimento. Sendo assim, avaliar seus desempenhos é um primeiro passo para uma percepção inicial acerca da adequação ou não dessas técnicas no contexto considerado.

- **Definição de um modelo de medição para acompanhar o desempenho de MBT ao longo dos ciclos de teste**. O ganho que MBT pode propiciar está atrelado à forma como o mesmo está sendo aplicado. Acompanhar esse desempenho permite detectar pontos falhos. Esses pontos falhos podem então ser melhorados, refletindo na melhoria do processo.

- **Realização de estudos de caso para refinar os modelos definidos**. Os estudos de caso são realizados considerando o domínio de aplicações para celular, dentro do ambiente de testes da Motorola. A escolha do domínio foi consequência do projeto no qual este trabalho está inserido, o CInBTCRD. Como resultado, espera-se modelos facilmente integráveis ao processo de testes.

O trabalho não almeja apresentar respostas definitivas sobre a eficácia ou não de MBT no domínio de aplicações para celular, uma vez que isto envolveria a aplicação dos modelos propostos em vários estudos de caso em escala industrial. O que se deseja é fornecer um embasamento inicial para que tais estudos possam ser realizados na prática. Os modelos propostos foram definidos com base em estudos teóricos e na observação de problemas práticos.

Ao longo do trabalho, estudos de caso foram conduzidos no intuito de ajustá-los à realidade, viabilizando sua aplicação no dia-a-dia.

1.3 Metodologia

Esse trabalho envolveu muita observação e experimentação. Ao longo dele, o ciclo adotado foi: observar, estudar, definir e aprimorar. A Figura 1.1 exemplifica esse ciclo. A partir das observações foi possível entender e estudar conceitos e processos. Com os conceitos definidos, conseguiu-se definir métricas e dados a serem coletados. Esses foram aprimorados a partir dos estudos de caso realizados.

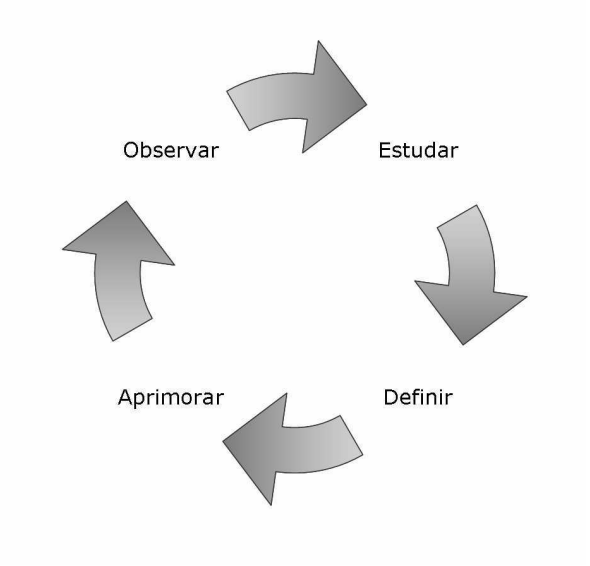


Figura 1.1: Metodologia

A definição de métricas e dados foi realizada seguindo o paradigma *Goal Question Metric*, apresentado em [Basili, 1992]. Ao final, dois modelos de medição foram definidos: o primeiro, para avaliar o desempenho de técnicas de teste quando aplicadas no teste de *feature* para aplicações para celular, particularmente MBT e teste exploratório; o segundo, para acompanhar o desempenho de MBT ao longo dos ciclos de teste.

O aprimoramento dos modelos foi feito a partir de estudos de caso conduzidos dentro do ambiente de testes da Motorola. O maior ganho com a realização desses estudos foi a adaptação dos modelos à realidade, uma vez que eles contêm apenas métricas possíveis e

facilmente coletáveis. Por fim, os modelos e estudos de caso foram relatados, resultando na escrita deste trabalho.

1.4 Relevância e Contribuições

Na área de testes de *software* há uma carência de trabalhos que avaliem o desempenho de técnicas de teste em contextos específicos. MBT é uma técnica aparentemente promissora para ser aplicada em testes de aplicações para celular, que são aplicações reativas onde os ciclos de teste se repetem ao longo do desenvolvimento do *software*. No entanto, na prática, é preciso mais resultados experimentais para que o uso de MBT nesse contexto específico seja impulsionado. Dentre as contribuições do trabalho, destacam-se:

- **Modelos de Medição** Definir modelos de medição envolve atividades referentes à compreensão do domínio e à definição e interpretação de métricas. Com modelos já definidos, é possível dedicar mais tempo ao preparo do estudo de caso, bem como vislumbrar mais facilmente diferentes métricas a serem calculadas, modificando o modelo já existente.

- **Estudos de Caso** A realização de estudos de caso dentro de contextos reais é uma grande contribuição. Geralmente não consegue-se aplicar a teoria pesquisada em estudos de caso reais e isso acaba afastando a pesquisa da realidade. Esses estudos foram importantes porque permitiram o ajuste dos modelos, proporcionando sua aplicabilidade nos processos já existentes, além de possibilitar um bom entendimento do domínio e dos problemas existentes.

1.5 Contexto de Desenvolvimento da Pesquisa

Este trabalho faz parte do *CIn Brazil Test Center Research and Development* (CInBTCRD), uma cooperação entre a Motorola Inc., o Centro de Informática da Universidade Federal de Pernambuco (CIn/UFPE), contando ainda com a colaboração do Departamento de Sistemas e Computação da Universidade Federal de Campina Grande (DSC/UFCG).

O CInBTCRD tem como foco principal a definição de um processo integrado para a geração, seleção e avaliação de casos de teste para aplicações para celular. A Figura 1.2

fornece uma visão das principais linhas de pesquisa do projeto. A partir de requisitos descritos em linguagem natural ou diagramas UML, casos de teste podem ser gerados de forma automática. Além disso, a estimativa de tempo e esforço de execução pode ser calculada e a cobertura de código na execução dos casos de teste pode ser analisada.

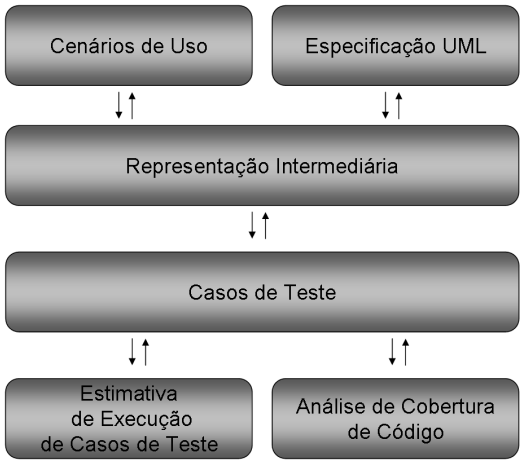


Figura 1.2: Visão geral do CInBTCRD

Como se vê na Figura 1.2, os casos de teste são gerados a partir de uma notação intermediária, que representa o modelo formal da aplicação. Esse modelo é gerado de forma automática com base nos requisitos, que são descritos em linguagem natural ou diagramas de sequência UML. A geração de casos de teste a partir de um modelo formal do comportamento da aplicação caracteriza o uso da técnica de MBT. O trabalho desenvolvido nesta dissertação busca definir modelos de medição para acompanhar o desempenho de MBT ao longo dos ciclos de teste, considerando métricas como esforço, por exemplo.

O CInBTCRD faz parte do *Motorola Brazil Test Center* (BTC), centro mundial de integração e verificação de *software* para telefonia celular da Motorola. Desde 2005, todos os celulares produzidos pela Motorola passam pelas mãos da equipe brasileira antes de chegar ao mercado global. A busca por soluções que otimizem os processos de teste e melhorem a qualidade dos celulares é uma importante atividade dentro do BTC, desempenhada por projetos como o CInBTCRD.

1.6 Estrutura da Dissertação

O restante deste documento foi estruturado da forma que segue.

Capítulo 2: Fundamentação Teórica Este capítulo apresenta os principais conceitos abordados ao longo do trabalho desenvolvido. Os conceitos de *Feature* e Teste de *Feature* são apresentados, mostrando suas características dentro do contexto de aplicações para celular. Apresentamos ainda os conceitos de Teste Exploratório, *Model-Based Testing* e o Paradigma *Goal Question Metric*.

Capítulo 3: Processo de MBT O capítulo apresenta o processo de MBT utilizado ao longo do trabalho. Uma visão geral da ferramenta TaRGeT também é apresentada, além de um exemplo prático de uso do processo de MBT adotado.

Capítulo 4: Avaliação Experimental de Estratégias para o Teste de *Feature* Neste capítulo apresentamos os resultados obtidos com a definição de um modelo de medição para avaliar o desempenho de teste exploratório e MBT quando utilizados no teste de *feature*. O modelo foi definido seguindo o paradigma *Goal Question Metric*.

Capítulo 5: Avaliação Experimental de *Model-Based Testing* O capítulo aborda a avaliação de MBT ao longo do processo de teste e apresenta um modelo de medição para avaliar o desempenho de MBT. Um estudo de caso é apresentado para validar o modelo.

Capítulo 6: Considerações Finais O capítulo sumariza o trabalho desenvolvido e apresenta algumas considerações sobre o mesmo. Propostas de trabalhos futuros são apresentadas e discutidas.

Capítulo 2

Fundamentação Teórica

Este capítulo tem como principal objetivo fornecer a fundamentação teórica mínima necessária acerca dos principais conceitos abordados ao longo deste trabalho. São apresentados e discutidos conceitos sobre *Feature*, Teste de *Feature*, Teste Exploratório, *Model-Based Testing* e paradigma *Goal Question Metric*. Ao final, são apresentadas característcas que tornam as abordagens exploratória e baseada em modelo candidatas promissoras para o teste de *feature*.

2.1 *Feature* e Teste de *Feature*

O termo *feature* é bastante utilizado na área de desenvolvimento de aplicações para celular. Segundo Turner [Turner et al., 1998], uma *feature* representa um conjunto de requisitos individuais que descrevem uma unidade identificável de funcionalidade. Se individualidade estiver atrelada à exclusividade, na prática a definição de *feature* torna-se um pouco diferente. A Figura 2.1 apresenta essa idéia. As *features* podem ter requisitos que pertencem a outras features, não sendo seus requisitos exclusivos dentro do contexto da aplicação, mas únicos apenas no contexto da *feature*.

Embora represente uma unidade identificável de funcionalidade, a *feature* não é um componente, com interfaces bem definidas, mas também não necessariamente é uma funcionalidade única, muitas vezes é um conjunto de funcionalidades cujo comportamento pode variar conforme a situação. Para exemplificar essa idéia de comportamento variável, vamos imaginar três *features* fictícias:

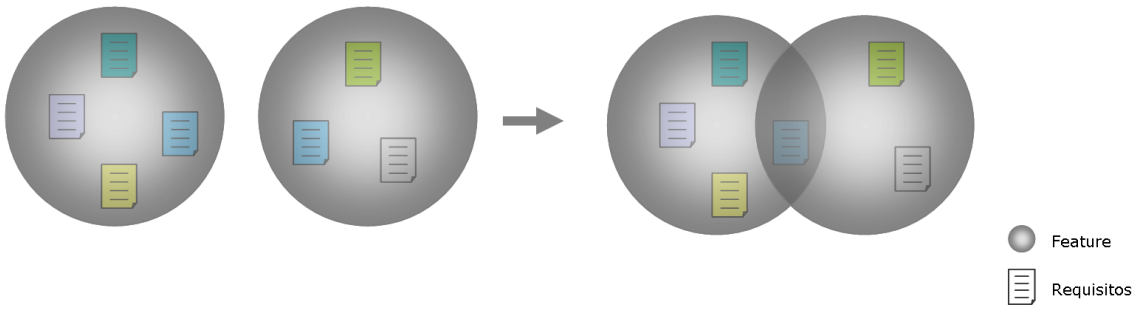


Figura 2.1: *Feature* e Requisitos

- **Envio de Mensagens** - a principal funcionalidade é o envio de mensagens de texto ou multimídia [1].
- **Armazenamento de Contatos** - A principal funcionalidade é gerenciar o armazenamento de contatos na agenda e disponibilizar esses contatos posteriormente.
- **Ligações** - A principal funcionalidade é receber ligações, além de gerenciar o status de cada uma: atendidas, não atendidas ou em espera.

Considere um cenário inicial, no qual o usuário está escrevendo uma mensagem de texto. No momento que ele escreve o corpo da mensagem, recebe uma ligação. O comportamento da *feature* "Ligações" vai sobrepor o comportamento da *feature* "Envio de Mensagens", ficando a segunda em *background*. Ao finalizar a chamada, a *feature* "Envio de Mensagens" retoma a prioridade e o celular volta para a tela de composição de mensagens.

Um segundo cenário, seria o usuário escrever o corpo da mensagem e acessar os contatos para efetuar o envio da mesma. No momento que ele acessa a lista de contatos, recebe uma ligação. Diferentemente do cenário anterior, a *feature* "Ligações" não vai sobrepor o comportamento da *feature* "Armazenamento de Contatos".

A partir dos cenários apresentados, percebe-se que as *features* podem interagir entre si e que o comportamento durante essa interação pode ser variável. Uma interação é caracterizada por cenários nos quais as funcionalidades de uma *feature* dependem das funcionalidades de uma outra *feature* [de Lucena Andrade, 2007]. O comportamento variável é comumente

[1] Uma mensagem multimídia é caracterizada por seu conteúdo, que precisa conter algum arquivo de imagem, vídeo ou áudio.

testado durante o teste de interação de *feature*. Por interagirem entre si, as *features* precisam ser bem testadas individualmente. O teste de *feature* torna-se bastante importante para reduzir o número dos chamados defeitos escapados, que são defeitos que escapam de uma fase para outra ao longo do processo de teste.

Features são desenvolvidas de maneira evolutiva. Essa evolução abrange tanto a evolução em termos de requisitos, onde uma *feature* pode evoluir para uma outra *feature* através do acréscimo de requisitos (Figura 2.2), quanto a evolução em termos de implementação, onde o código que representa a *feature* é melhorado a medida que os os ciclos de teste vão sendo executados. O processo simplificado do teste de *feature* que vamos considerar ao longo deste trabalho é apresentado na Figura 2.3: uma versão executável do *software*, contendo a implementação da *feature* a ser testada, é submetida ao ciclo de teste. Após a execução dos testes, se a porcentagem de aceitação requerida é atingida, o teste da *feature* é finalizado. Se a porcentagem não é atingida, os defeitos detectados ao longo do ciclo de teste são corrigidos e uma nova versão executável com defeitos retirados é submetida novamente ao ciclo de teste.

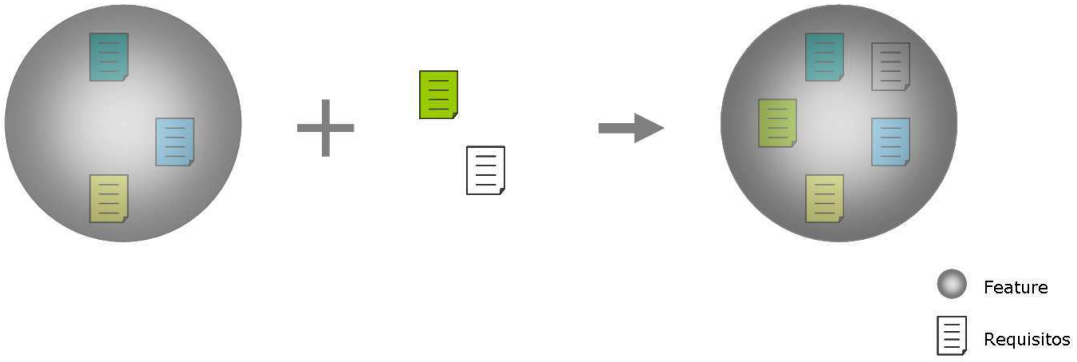


Figura 2.2: Evolução da *feature* através do acréscimo de requisitos

O teste de *feature* é um tipo de teste funcional que apresenta algumas características adicionais:

- Execução intensiva - Devido a forma como uma *feature* é desenvolvida e por ser integrada com diferentes *features*, é fundamental que seu comportamento esteja de acordo com o descrito nos documentos de requisitos. Por isso, uma *feature* precisa ser testada de forma exaustiva, maximizando a detecção de defeitos.

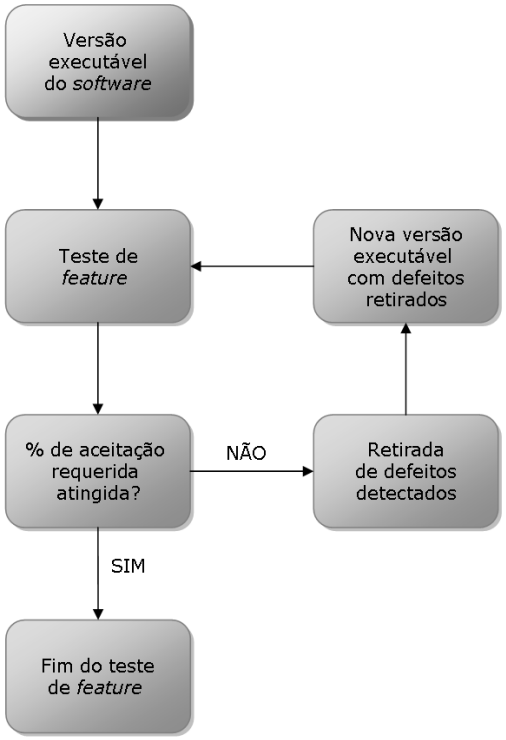


Figura 2.3: Processo simplificado do teste de *Feature*

- Rapidez - A demanda de produção devido ao *time-to-market* reduz o tempo de desenvolvimento e, conseqüentemente, o teste de *feature* precisa ser executado em um período mínimo de tempo.

- Conhecimento dos requisitos - Por ser um tipo de teste funcional, o teste de *feature* requer um nível adequado de conhecimento dos requisitos. Além disso, requer conhecimento sobre o comportamento da aplicação para planejar e executar os casos de teste, uma vez que uma *feature* pode interagir com outras.

- Execução repetitiva - Os casos de teste de uma *feature* podem ser re-executados diversas vezes durante o seu ciclo de desenvolvimento. Normalmente, existe mais de um ciclo de teste com variantes da mesma suite de testes.

Um caso de teste de *feature* é um caso de teste tradicional, com condições iniciais, passos e resultados esperados (Figura 2.4). Devido a interatividade das aplicações para celular, os casos de teste são em sua maioria executados de forma manual, requerendo uma maior tempo

para execução. Como o tempo para testes é limitado pelo *time-to-market*, a suite de testes precisa ser bem escolhida, para que seja testado o máximo possível no menor tempo.

TC 01

Condições Iniciais

Mensagem é movida para a pasta "Mensagens Favoritas". Não há memória suficiente

Passos	**Resultados Esperados**
Acesse a pasta "Entrada"	A pasta "Entrada" é exibida
Selecione alguma mensagem	A mensagem é destacada
Selecione o menu de contexto	A opção "Mover para Mensagens Favoritas" é exibida
Selecione a opção "Mover para Mensagens Favoritas"	Uma caixa de diálogo com a mensagem "Memória está cheia" é exibida
Confirme a caixa de diálogo	O conteúdo da mensagem é exibido e a mensagem não é movida.

Figura 2.4: Exemplo de um caso de teste de *feature*

Classicamente dois casos de teste são ditos redundantes sintaticamente se apresentam uma mesma seqüência de passos. Por exemplo, na Figura 2.5, percebe-se que o caso de teste 2 está inserido no caso de teste 1, caracterizando uma redundância sintática.

Passos	**Resultados Esperados**
Acesse "Mensagens"	A lista de opções é exibida. Dentre as opções, existe a opção "Criar Mensagem"
Acesse "Criar Mensagem"	***O compositor de mensagens é exibido***
Insira o conteúdo da mensagem	***O conteúdo da mensagem é exibido***
Insira o número do destinatário	***O número do destinatário é exibido***
Selecione enviar	***Uma mensagem informando que a mensagem foi enviada é exibida. A mensagem é salva na pasta "Mensagens Enviadas"***

(a) Caso de teste 1

Passos	**Resultados Esperados**
Acesse "Criar Mensagem"	***O compositor de mensagens é exibido***
Insira o conteúdo da mensagem	***O conteúdo da mensagem é exibido***
Insira o número do destinatário	***O número do destinatário é exibido***
Selecione enviar	***Uma mensagem informando que a mensagem foi enviada é exibida. A mensagem é salva na pasta "Mensagens Enviadas"***

(b) Caso de teste 2

Figura 2.5: Exemplo de casos de teste de *feature* com redundância sintática

No teste de *feature*, esse conceito de redundância vai um pouco além. Dois casos de teste são ditos redundantes se apresentam uma mesma seqüência de passos ou uma seqüência de

passos equivalentes, cuja ordem de alguns passos não caracteriza como diferentes os casos de teste. Na Figura 2.6, são apresentados dois exemplos de casos de teste que testam o envio de mensagens. Apesar de apresentarem uma seqüência de passos diferentes, os casos de teste são considerados redundantes, pois assume-se que o formulário foi devidamente validado e que não há validação prévia dos campos, portanto a ordem de preenchimento do corpo da mensagem ou do destinatário não altera o resultado do teste, por isso são considerados redundantes.

Passos	**Resultados Esperados**
Acesse "Mensagens"	A lista de opções é exibida. Dentre as opções, existe a opção "Criar Mensagem"
Acesse "Criar Mensagem"	O compositor de mensagens é exibido
Insira o conteúdo da mensagem	**O conteúdo da mensagem é exibido**
Insira o número do destinatário	*O número do destinatário é exibido*
Selecione enviar	Uma mensagem informando que a mensagem foi enviada é exibida. A mensagem é salva na pasta "Mensagens Enviadas"

(a) Caso de teste 1

Passos	**Resultados Esperados**
Acesse "Mensagens"	A lista de opções é exibida. Dentre as opções, existe a opção "Criar Mensagem"
Acesse "Criar Mensagem"	O compositor de mensagens é exibido
Insira o número do destinatário	*O número do destinatário é exibido*
Insira o conteúdo da mensagem	**O conteúdo da mensagem é exibido**
Selecione enviar	Uma mensagem informando que a mensagem foi enviada é exibida. A mensagem é salva na pasta "Mensagens Enviadas"

(b) Caso de teste 2

Figura 2.6: Exemplo de casos de teste de *feature* com redundância semântica

Essa extensão do conceito de redundância advém de outra particularidade do teste de *feature*: a influência da experiência do testador. O testador experiente sabe que uma alternância de passos em determinados momentos não mudará o resultado do caso de teste, culminando em uma redundância semântica, ou seja, os casos de teste são equivalentes pois produzem o mesmo resultado.

Determinar quando a ordem de determinados passos é relevante ou não para o caso de teste é ainda um problema para a geração automática de casos de teste de *feature*, pois na maioria das vezes esta informação encontra-se no modelo do sistema que o testador experiente guarda em mente. Essa experiência do testador, bem como a confiança de que determinados passos de um caso de teste não detectarão falhas, provêm do fato do desenvolvimento da *feature* ser evolutivo, pois dessa forma o testador sabe quais funcionalidades são vulneráveis e quais já são estáveis.

Na literatura existem poucos trabalhos que abordam o teste de *feature*. Cartaxo [Cartaxo, 2006] e Nogueira [de Carvalho Nogueira, 2006] focam no teste de *feature* para aplicações para celular, Baker et. al [Baker et al., 2002] focam no teste de *feature* para protocolos de comunicação. Andrade [de Lucena Andrade, 2007] aborda o problema do teste de interação de *features*. O escopo do nosso trabalho abrange apenas o teste de *feature* no contexto de aplicações reativas, particularmente aplicações para celular, não considerando os problemas provindos do teste de interação de *features*.

2.2 Teste Exploratório

O termo Teste Exploratório (TE) foi inicialmente publicado por Kaner em 1988, no seu livro *Testing Computer Software* no intuito de diferenciar o teste exploratório do teste *ad hoc*. Pode-se considerar o teste *ad hoc* como sendo um caso especial do teste exploratório, onde não há nenhum tipo de anotação que deixe rastros para uma posterior re-execução do teste [Agruss and Johnson, 2000].

O teste exploratório pode ser definido como uma abordagem de teste na qual o *design* e a execução dos casos de teste ocorrem simultaneamente. Em outras palavras, teste exploratório é qualquer teste no qual o testador controla o *design* à medida que os testes vão sendo executados e usa os conhecimentos adquiridos durante a execução para projetar novos e melhores testes [Bach, 2003].

O teste exploratório não é um substituto de outras abordagens de teste, mas um complemento a elas [Itkonen and Rautiainen, 2005]. Saber em que momento do processo de teste deve-se utilizá-lo bem como o quanto, pode representar a diferença entre o sucesso e o fracasso do seu uso. Alguns trabalhos destacam benefícios do teste exploratório, tais como efetividade, eficiência, rápido *feedback* e o melhor aproveitamento da criatividade do testador [Bach, 2000; Bach, 2003; Cem Kaner and Pettichord, 2002]. No entanto, ainda não há evidências científicas desses benefícios [Itkonen and Rautiainen, 2005].

A experiência e o conhecimento do testador são essenciais para que se tenha um bom resultado utilizando TE. Itkonen [Itkonen and Rautiainen, 2005] apresenta um estudo do uso de teste exploratório em três empresas. Em uma das empresas concluiu-se que TE auxiliou na detecção de importantes defeitos em um curto espaço de tempo, no entanto se um testa-

dor menos experiente e com menos conhecimento tivesse realizado os testes, os resultados obtidos talvez não tivessem sido tão bons.

Bach [Bach, 2000] apresenta uma metodologia para aplicação do teste exploratório usando sessões de teste. Uma sessão pode ser definida como um período pré-determinado de tempo, no qual o testador, de forma ininterrupta, testa a aplicação de acordo com um objetivo, também pré-definido, e ao final gera um relatório informal contendo informações acerca dos defeitos, duração da sessão de teste, testador, dentre outros detalhes.

A Fig. 2.7 apresenta um exemplo de relatório resumido de uma sessão de teste exploratório. São encontradas informações como: objetivo da sessão, nome do testador, início e fim da sessão, relato das falhas encontradas (*Bugs*) e anotações.

Testador

Laísa Helena

Duração

Início: 20/06/2007 09:00 AM
Fim: 20/06/2007 09:50 AM

Objetivo

Explorar a transferência de mensagens da pasta de entrada e da pasta de saída para a pasta de mensagens favoritas quando a memória estiver cheia.

Bugs

#Bug_01

O texto da mensagem informando ao usuário que a memória está cheia está incorreto, difere da mensagem descrita nos requisitos.

#Bug_02
Ao tentar mover uma mensagem da pasta de entrada para a pasta de mensagens favoritas quando esta continha 20 mensagens e a primeira 450, a mensagem de diálogo informando que a memória estava cheia foi exibida, no entanto a aplicação travou e foi preciso reiniciar o fone.

Anotações
#Anotação_01
Não tenho certeza sobre a corretude do posicionamento das pastas na aplicação mensagem, os requisitos não deixam claro esta informação.

Figura 2.7: Exemplo de relatório de uma sessão de teste

Embora o uso de TE seja interessante quando se tem um testador experiente e com largo conhecimento da aplicação, seu uso pode ser produtivo quando o time não conhece a aplicação, estando todos no mesmo nível de conhecimento. O uso de TE instiga o testador a se

aventurar na aplicação, dessa forma o time pode adquirir conhecimento do funcionamento do sistema de forma mais rápida. Nessa situação, as prováveis falhas encontradas seriam sugestivas.

O grande ganho com o uso de TE está no desenvolvimento das habilidades do testador. Diferentemente do teste pré-definido que torna o processo de teste um tanto quanto mecânico, o TE motiva o testador a explorar a aplicação de acordo com sua intuição. Portanto, o testador explorador precisa ser capacitado para desempenhar tal papel de forma bem sucedida.

2.3 *Model-Based Testing*

Model-Based Testing (MBT) é uma abordagem *black-box* de teste de *software* onde atividades comuns do processo de teste, como geração de casos de teste e análise de resultados, se baseiam na especificação da aplicação sob teste [El-Far and Whittaker, 2001]. A especificação do sistema deve ser descrita por um modelo formal que descreva com exatidão seu comportamento. Se o modelo é válido, então é possível avaliar se a aplicação se comporta conforme sua especificação, comparando o comportamento de entrada/saída do modelo com o do sistema sob teste [Prenninger and Pretschner, 2005].

As principais atividades do MBT são (Fig. 2.8):

- Construção do Modelo - A concepção do modelo inicia-se com o entendimento do sistema sob teste. Representar mentalmente as funcionalidades do sistema é um pré-requisito para se construir um modelo [El-Far and Whittaker, 2001]. Definido esse modelo mental, deve-se escolher um modelo formal que melhor represente os requisitos do software em questão para então se construir o modelo do mesmo.

- Geração dos Casos de Teste - Consiste na geração dos casos de teste a partir das especificações representadas no modelo. Cada caso de teste deve possuir passos e os respectivos resultados esperados. O grau de dificuldade na automação desta etapa está associado ao modelo escolhido para representar o sistema sob teste. Em alguns modelos, como por exemplo, máquinas de estado finito, para garantir a automação se faz necessário apenas a implementação de um algoritmo que percorra aleatoriamente o

respectivo diagrama de transição de estados [El-Far and Whittaker, 2001].

- Execução dos Casos de Teste - Após a geração, os casos de teste são executados e espera-se que a aplicação se comporte conforme o descrito nos resultados esperados. Caso isto não ocorra, é dito que o caso de teste falhou, representando a não correspondência entre implementação e especificação.

- Coleta e Análise de Resultados - A coleta de resultados consiste em guardar informações acerca da execução dos casos de teste, tais como tempo de execução e resultado do teste. A partir dos resultados é possível fazer uma análise para que melhorias no modelo sejam realizadas e o processo de geração evolua no sentido de gerar casos de teste cada vez mais relevantes.

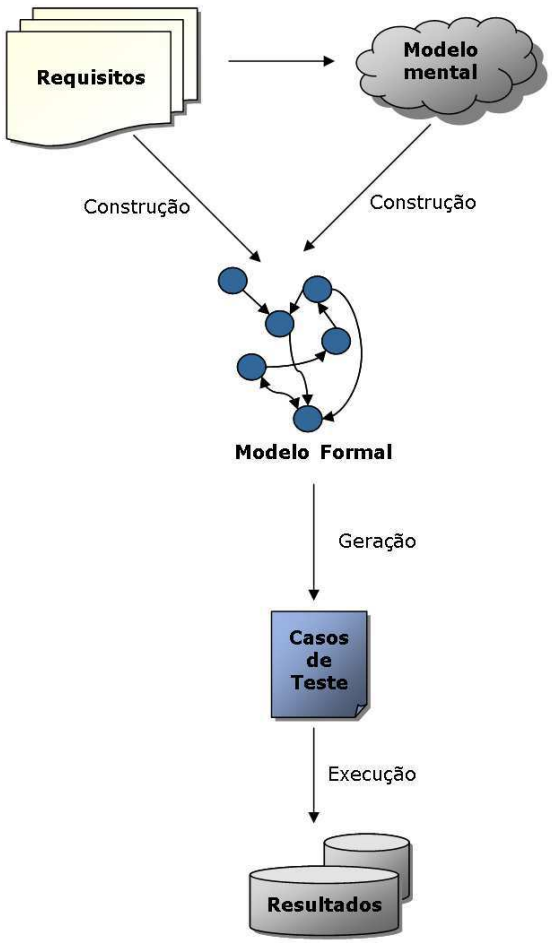


Figura 2.8: Atividades do MBT

Dentre as vantagens que o uso de MBT pode propiciar, podemos destacar:

- Compartilhamento de um modelo único entre os vários times e pessoas envolvidas no processo de desenvolvimento, o que reduz ambigüidades no entendimento do comportamento do sistema;
- Automação da geração dos casos de teste;
- Antecipação do início do processo de teste, pois o modelo pode ser construído já durante a definição dos requisitos;
- Facilidade de manutenção do conjunto de casos de teste, já que uma alteração no modelo pode ser rapidamente refletida nos casos de teste, mantendo a consistência entre eles.

A automação da geração dos casos de teste permite a manutenção, com baixo esforço, da consistência entre os testes e os requisitos. Além disso, a automação pode reduzir o custo do processo de teste uma vez que reduz o tempo de criação dos casos de teste [Hartman and Nagin, 2004]. Outra vantagem do uso de MBT é a possibilidade de reuso dos artefatos de desenvolvimento, particularmente documentos de requisitos [Barbosa et al., 2007].

Embora a utilização de MBT apresente vantagens, a geração automática de casos de teste a partir de modelos ainda apresenta alguns problemas, como o grande número de casos de teste gerados, cobertura de código, redundância sintática[2] e semântica[3], e explosão do espaço de estados. Além dessas questões, existe a necessidade de entendimento, por parte dos que utilizarão o modelo, da notação escolhida para representar o mesmo. No Capítulo 3, o processo de MBT adotado neste trabalho é apresentado e exemplificado.

2.4 Paradigma *Goal Question Metric*

O paradigma *Goal Question Metric* (GQM) é um mecanismo para definição e interpretação de medição de *software* [Basili, 1992]. A medição de processos de desenvolvimento e produtos é uma forma de criar um histórico corporativo [Basili et al., 1995]. Além disso, auxilia no

[2]Dois casos de teste são ditos redundantes sintaticamente se apresentam uma mesma sequência de passos.

[3]A redundância semântica ocorre quando dois casos de teste produzem o mesmo resultado, tornando-se equivalentes.

planejamento de projetos, permite detectar pontos fortes e fracos de um processo ou produto e avaliar a qualidade de processos específicos [Basili, 1992].

A definição de um modelo de medição seguindo a abordagem GQM ocorre de maneira *top-down*, uma vez que a abordagem baseia-se no fato de que uma organização precisa primeiramente definir seus objetivos para então defini-los de forma operacional e, finalmente, prover um *framework* para interpretar os dados conforme os objetivos traçados. O resultado da aplicação da abordagem GQM é a especificação de um modelo de medição com enfoque em um objetivo organizacional. O modelo de medição resultante possui três níveis:

- Conceitual (*Goal*) - Nesse nível, um objetivo é definido para algum objeto, considerando razões, pontos de vista, critérios de qualidade e ambiente. Um objeto de medição pode ser um artefato, um processo, um *software*, dentre outros.
- Operacional (*Question*) - No nível operacional são definidas questões que tentam caracterizar o objeto de medição com relação ao critério de qualidade desejado e determinar a qualidade a partir do ponto de vista em questão.
- Quantitativo (*Metric*) - Um conjunto de dados é definido para responder cada questão de maneira quantitativa. Os dados podem ser:
  - Objetivos - Dependem do objeto e independem do ponto de vista no qual são coletados [Basili et al., 1995]. Por exemplo: número de versões de um documento, tamanho de um programa.
  - Subjetivos - Dependem tanto do objeto quanto do ponto de vista [Basili et al., 1995]. Exemplo: nível de satisfação do usuário.

A aplicação do paradigma GQM envolve cinco macro etapas:

1. **Definição dos objetivos** - Nessa etapa define-se o objetivo da medição, o que se deseja saber ao final da mesma.
2. **Elaboração de questões que caracterizam os objetivos** - Como o objetivo definido anteriormente pode ser atingido?
3. **Especificação de métricas que respondem as questões levantadas** - Quais métricas podem responder às questões? Quais dados precisam ser coletados?

4. **Desenvolvimento de mecanismos para a coleta de dados** - Como coletar os dados? Ao longo ou ao final de quais atividades?

5. **Coleta, validação e análise dos dados** - Calcular as métricas e interpretar os resultados.

A Figura 2.9 mostra a estrutura hierárquica do modelo GQM: o objetivo é definido, questões que caracterizam tal objetivo são elaboradas e por fim, métricas que respondem as questões são definidas e os dados coletados.

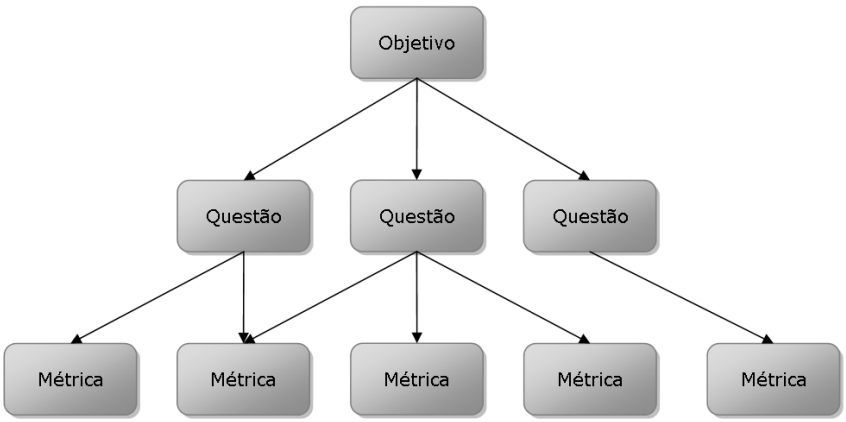


Figura 2.9: Estrutura hierárquica do modelo GQM

2.5 Conclusões

Neste capítulo foram apresentados conceitos sobre *feature*, teste de *feature*, teste exploratório, MBT e o paradigma GQM. As abordagens exploratória e baseada em modelo apresentam características interessantes, sendo aparentemente promissoras se aplicadas no processo de teste de *feature*.

Realizando teste exploratório, o testador faz uso dos seus instintos, o que aumenta a probabilidade de detectar defeitos escondidos[4] mais facilmente do que utilizando outras abor-

[4]Entenda-se por defeito escondido o defeito que não está inserido nos cenários da aplicação que são utilizados frequentemente pelo usuário. Geralmente, estes são os cenários mais cobertos pelos testes pré-definidos, por isso, diz-se que o teste exploratório pode vir a detectar uma quantidade maior desses defeitos.

dagens de teste. Essa característica pode reduzir o número de defeitos escapados e melhorar a efetividade do teste de *feature* em termos de números de defeitos encontrados. Além disso, como mencionado anteriormente, a execução dos casos de testes de *feature* ocorre geralmente de forma manual, devido a interatividade das aplicações para celular. Esse aspecto é adequado para estratégias como TE. Finalmente, uma outra característica do teste exploratório é que o testador adquire e aumenta seu conhecimento sobre o sistema durante a execução dos testes. Uma vez que o teste de *feature* demanda um bom conhecimento dos requisitos, essa característica do teste exploratório reforça nosso sentimento com relação a sua adequação ao teste de *feature*.

Aplicando MBT, o tempo do processo de teste pode ser reduzido através da automação da geração dos casos de teste. Uma vez que o teste de *feature* precisa ser rápido, essa característica torna MBT uma abordagem interessante para ser aplicada nesse contexto. Um outro fator é a facilidade de manutenção do conjunto de casos de teste, que contribui no gerenciamento da suite de teste ao longo dos ciclos de teste de uma *feature*. Como a *feature* pode evoluir em termos de requisitos, estes podem ser facilmente atualizados e a suite re-gerada sem maiores custos, devido a automação do processo de geração.

Na literatura da engenharia de *software* experimental existem alguns mecanismos para definição de objetivos mensuráveis, como o *Software Quality Metrics Approach* [Boehm et al., 1976], o *Quality Function Deployment Approach* [Kogure and Akao, 1983] e o *Goal/Question/Metric Paradigm* [Basili, 1992; Basili et al., 1995]. Basili [Basili, 1992] apresenta um quadro comparativo desses três mecanismos considerando alguns aspectos como escopo de aplicação e estrutura. A Tabela 2.1 apresenta um resumo desse quadro comparativo.

Tabela 2.1: Comparação entre as abordagens SQM, QFD e GQM.

	SQM	QFD	GQM
Objeto de Estudo	Produto Final	Produtos	Qualquer produto, processo ou modelo
Ponto-de-vista	Cliente, Usuário	Cliente, Usuário	Cliente, Usuário, desenvolvedor, gerente,...

A abordagem SQM foi desenvolvida para que o cliente avalie o produto, desenvolvido por contrato, sendo o objeto de estudo o produto final. A abordagem QFD foca nos documentos de *software* que visam desenvolver o produto conforme as necessidades do cliente. Como o paradigma *Goal/Question/Metric* possibilita que um processo seja o objeto de estudo, ele foi o escolhido para auxiliar na definição dos modelos de medição apresentados nesse trabalho. O próximo capítulo apresenta uma visão geral do proceso de MBT adotado ao longo deste trabalho.

Capítulo 3

O Processo de MBT

Neste capítulo são apresentados o processo de MBT adotado ao longo deste trabalho e uma visão geral da ferramenta TaRGeT, utilizada para geração automática dos casos de teste. O processo aqui descrito baseia-se no processo genérico de MBT apresentado no Capítulo 2.

3.1 Visão Geral do Processo

O processo genérico de MBT apresentado no Capítulo 2 é composto por quatro etapas: i) construção do modelo; ii)geração dos casos de teste; iii) execução dos casos de teste; e iv) coleta e análise de resultados. O processo adotado neste trabalho ajusta as etapas i e ii de forma a utilizar a ferramenta TaRGeT, descrita na Seção 3.2.

A Figura 3.1 apresenta as etapas do processo genérico de MBT e do processo adotando a ferramenta. Em destaque está a principal diferença entre os processos: a construção do modelo formal. Utilizando a TaRGeT, a escrita do modelo formal é substituída pela escrita de cenários de uso, ficando a construção do modelo transparente para o usuário.

A construção do modelo é uma das etapas mais importantes para que o uso de MBT seja bem sucedido. O *test designer* precisa ter em mente um modelo abstrato da aplicação e um bom conhecimento dos requisitos. Com o uso da TaRGeT, a construção do modelo remete-se à escrita de cenários de uso da aplicação.

Escritos os cenários, os mesmos devem ser inspecionados para reduzir o número de casos de teste modificados na fase de execução. TaRGeT gera os casos de teste de forma automática. Os testes gerados requerem execução manual, que deve ser realizada de acordo

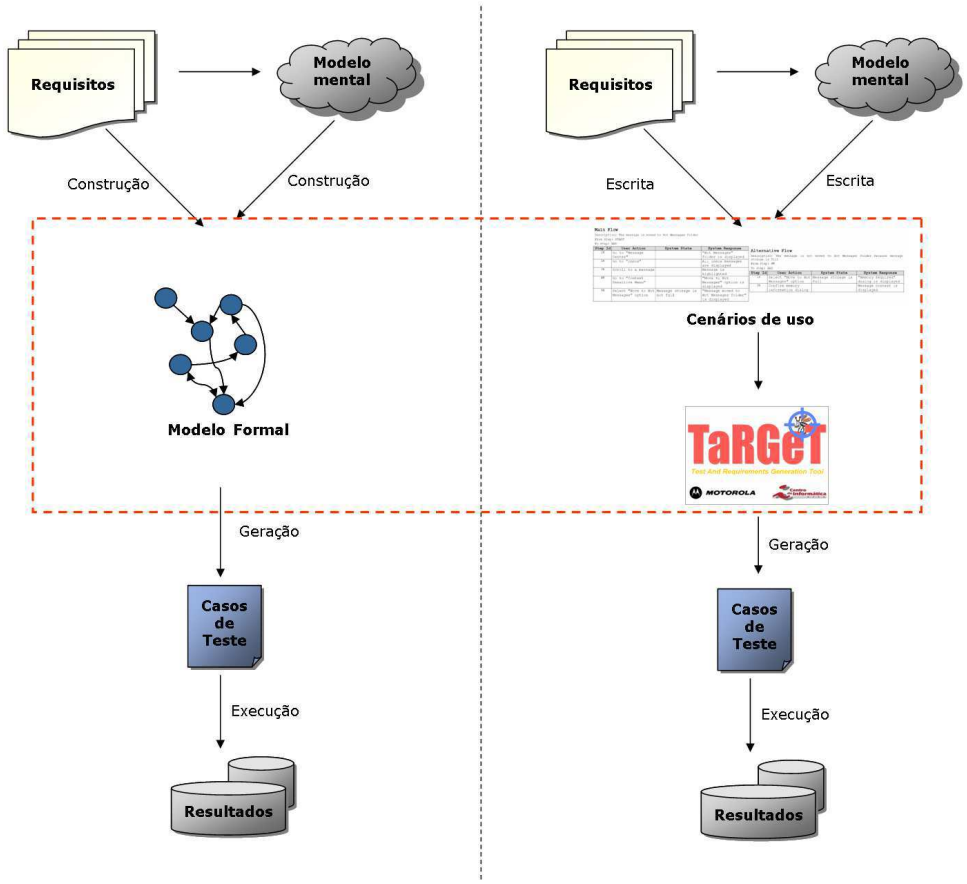


Figura 3.1: Processo genérico X Processo adotando TaRGeT

com o processo utilizado. Durante este trabalho, a execução dos casos de teste ocorreu conforme os processos de execução da Motorola. Assim como os cenários de uso, os casos de teste gerados podem passar por um processo de inspeção antes de serem executados. Isso reduz o número de casos de teste modificados ou descartados, por não condizerem com a realidade, durante a fase de execução. Os processos de inspeção são muito importantes, principalmente quando se tem um time independente de execução.

Os resultados das execuções são coletados e armazenados em locais específicos, como ferramentas de gerenciamento de testes, por exemplo, e posteriormente analisados. A análise pode envolver, dentre outros apectos, o percentual de casos de teste que falharam e o esforço de execução.

3.1.1 O Processo de MBT dentro do Processo de Teste de *Feature*

A aplicação de MBT dentro do processo de teste de *feature* ocorre sem maiores custos ao processo de teste. No Capítulo 2 foi apresentado um processo genérico para o teste de *feature*, onde ciclos de teste se repetem até que o percentual de falhas requerido seja atingido. A Figura 3.2 mostra como o processo de MBT descrito nesta seção pode ser adotado dentro do processo de teste de *feature* sem maiores custos.

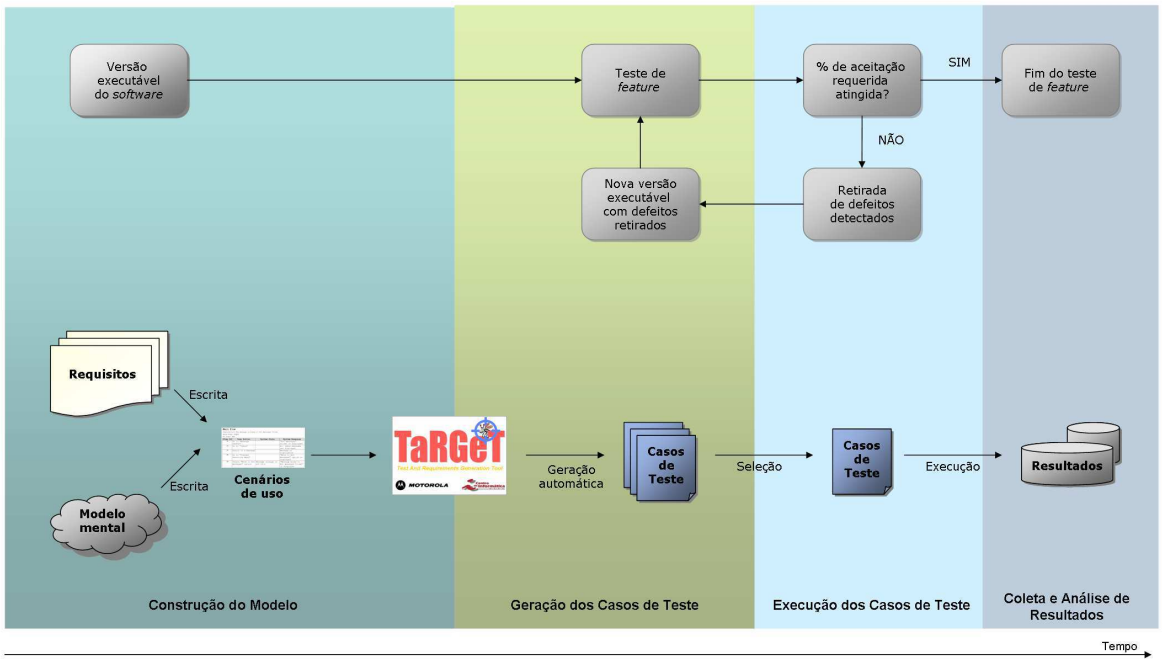


Figura 3.2: MBT e FT

Antes da primeira versão executável estar disponível para os testes, os cenários de uso podem ser escritos a partir dos requisitos, inspecionados e em seguida, os casos de teste gerados e inspecionados. Com a versão para testes liberada, os casos de teste gerados de forma automática podem ser selecionados, executados e os resultados armazenados. Caso o percentual de falhas requerido não seja atingido, uma nova seleção dentro da suite pode acontecer ou o mesmo conjunto de testes pode ser novamente executado. O armazenamento dos resultados é interessante para fazer estimativas futuras e acompanhar o desempenho do uso de MBT ao longo dos ciclos de execução.

No mundo ideal, ao se usar MBT, toda e qualquer modificação na suite de testes deve ocorrer no modelo da aplicação e os requisitos devem cobrir todos os possíveis cenários de

execução. Quando uma *feature* evoluísse devido a alterações nos requisitos, por exemplo, os cenários de uso deveríam ser atualizados e não os casos de teste. Na prática, esse mundo ideal é um pouco diferente. Os requisitos são normalmente incompletos, não contemplando todos os possíveis cenários, e modificações podem ser feitas diretamente nos casos de teste.

No processo de teste *feature* considerado neste trabalho, podem existir vários ciclos de execução de testes para uma mesma *feature*. Novos casos de teste podem ser adicionados à suite inicial e essa adição de casos de teste pode ser feita de forma manual, o que torna os cenários de uso desatualizados. A atualização do modelo formal é um requisito para que se tenha um bom resultado com a aplicação de MBT, sendo este um problema a ser considerado na utilização do processo de MBT descrito.

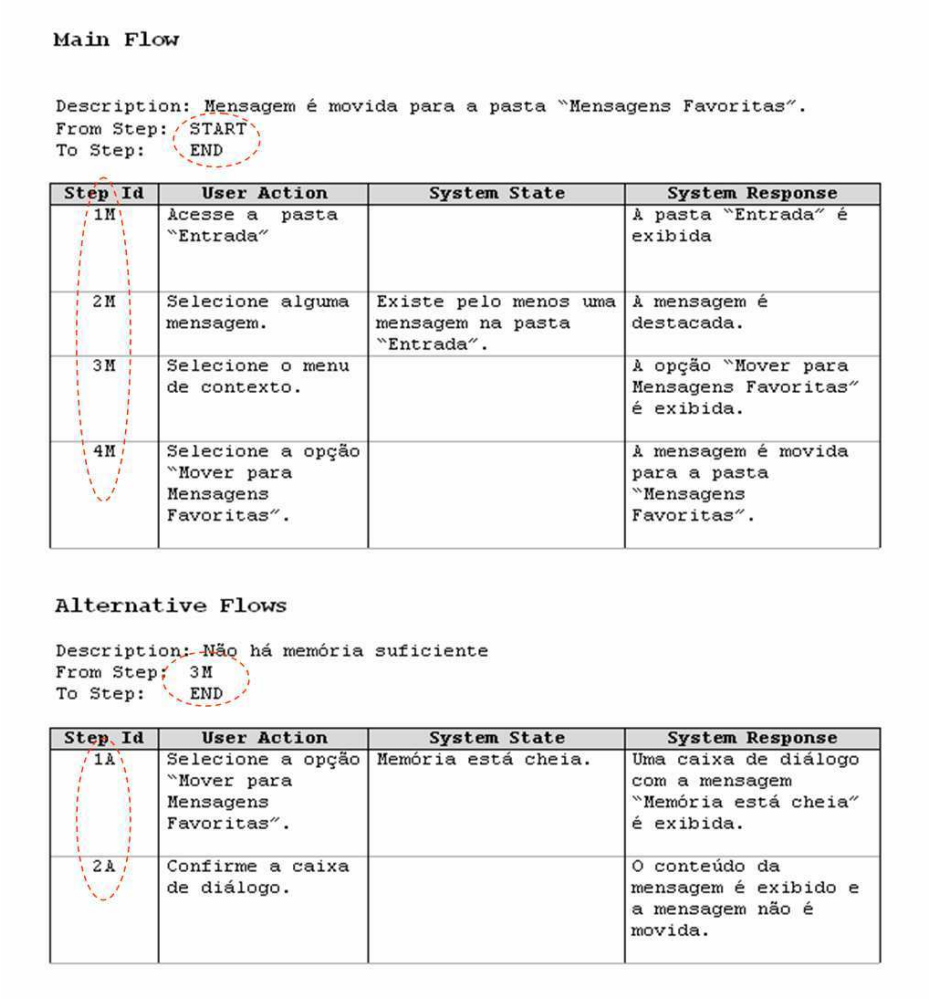
3.2 A Ferramenta TaRGeT

Uma das barreiras na utilização de MBT nos processos de teste é a necessidade de definir formalmente o modelo da aplicação sob teste. Uma alternativa é a utilização de ferramentas que tornam transparente para o usuário a construção desse modelo. No intuito de incentivar a adoção de MBT dentro da Motorola, o CInBTCRD desenvolveu uma ferramenta para geração de casos de teste a partir de requisitos, a *Test and Requirements Generation Tool* (TaRGeT).

TaRGeT [Nogueira et al., 2007] é uma ferramenta de geração automática de casos de teste de *feature* a partir de cenários de uso da aplicação sob teste. Esses cenários são escritos por *test designers* que se baseiam no seu conhecimento e nos requisitos da *feature* e da aplicação.

Os cenários de uso são escritos em linguagem natural, de acordo com o template apresentado em [Cabral and Sampaio, 2006; Nogueira et al., 2007]. O template faz uso de tabelas (Figura 3.3) onde são indicados: ação do usuário (*User Action*); estado do sistema (*System State*); e resultado esperado (*System Response*). A ação do usuário indica a ação que deve ser executada, o resultado esperado indica a resposta do sistema para aquela ação, e o estado do sistema indica uma condição necessária para que executada a ação, aquele resultado esperado aconteça.

Os cenários são compostos por fluxos que podem ou não ter alguma interseção. Os



Main Flow

Description: Mensagem é movida para a pasta "Mensagens Favoritas".
From Step: START
To Step: END

Step Id	User Action	System State	System Response
1M	Acesse a pasta "Entrada"		A pasta "Entrada" é exibida
2M	Selecione alguma mensagem.	Existe pelo menos uma mensagem na pasta "Entrada".	A mensagem é destacada.
3M	Selecione o menu de contexto.		A opção "Mover para Mensagens Favoritas" é exibida.
4M	Selecione a opção "Mover para Mensagens Favoritas".		A mensagem é movida para a pasta "Mensagens Favoritas".

Alternative Flows

Description: Não há memória suficiente
From Step: 3M
To Step: END

Step Id	User Action	System State	System Response
1A	Selecione a opção "Mover para Mensagens Favoritas".	Memória está cheia.	Uma caixa de diálogo com a mensagem "Memória está cheia" é exibida.
2A	Confirme a caixa de diálogo.		O conteúdo da mensagem é exibido e a mensagem não é movida.

Figura 3.3: Exemplo de entrada para a ferramenta TaRGeT

fluxos representam sequências de ações que podem ser executadas na aplicação. A Figura 3.3 mostra parte de um cenário de uso no qual existem dois fluxos, um principal e um alternativo. O fluxo principal inicia no estado *START*, passa pelos estados *1M*, *2M*, *3M* e *4M*, finalizando no estado *END*, onde *START* e *END* são os estados inicial e final do grafo que representa o modelo formal dos cenários da aplicação sob teste. O fluxo alternativo vem do estado *3M*, passando posteriormente pelos estados *1A* e *2A*, finalizando no estado *END*. Graficamente, essa interação dos fluxos é representada como apresentado na Figura 3.4

A partir dos cenários de uso, a TaRGeT gera de forma automática e transparente para o usuário, um modelo formal que representa os cenários da aplicação sob teste. A partir desse modelo, as suites de teste podem ser geradas de maneira automática.

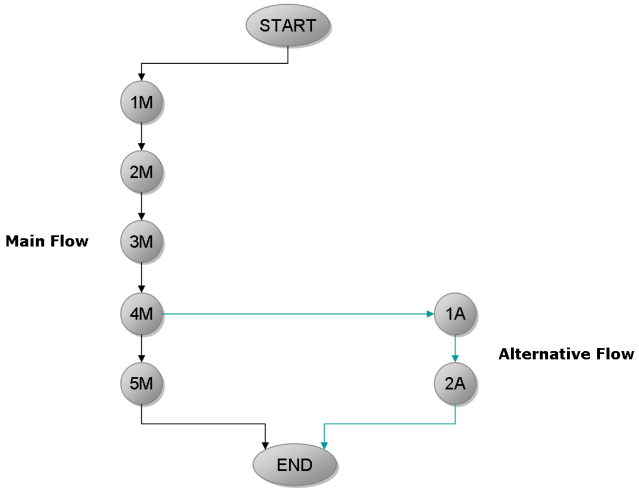


Figura 3.4: Fluxo que representa o exemplo da Figura3.3

No momento da geração, a TaRGeT oferece algumas opções para restringir a quantidade de casos de teste da suite final, como: $i)$ geração por requisitos, onde o usuário seleciona os requisitos e são gerados apenas os casos de teste que cobrem esses requisitos; $ii)$ geração por cenário de uso, onde serão gerados apenas os casos de teste cobertos nos cenários selecionados; $iii)$ geração por similaridade [Emanuela G. Cartaxo, 2007], onde a quantidade de casos de teste da suite final é reduzida conforme a similaridade informada. Por exemplo, se o modelo permite gerar uma suite com 50 casos de teste, mas uma similaridade igual a 50% é informada, a suite final será gerada com os 25 casos de teste menos similares; $iv)$ geração por propósito [Cartaxo, 2006], onde serão gerados apenas os casos de teste que atendem a determinado propósito de teste. Um propósito de teste é uma especificação de propriedades desejáveis no modelo. No caso da TaRGeT, um propósito pode ser entendido como uma sequência de um ou mais nós dentro do grafo que representa o modelo da aplicação. Por exemplo, considerando o grafo da Figura 3.4, um possível propósito seria $4M, 1A$, que significa que se deseja apenas os casos de teste nos quais se tenta mover uma mensagem para a pasta de mensagens favoritas mas não se consegue devido a falta de memória. Durante este trabalho foi considerada apenas a geração completa, que contempla todos os caminhos do grafo que representa os cenários do sistema sob teste.

Em resumo, o processo de uso da TaRGeT consiste em escrever cenários de uso e utilizar a ferramenta para geração dos casos de teste. A Figura 3.5 exemplifica essa idéia. A

partir dos requisitos, o *test designer* escreve cenários de uso que servirão de entrada para ferramenta TaRGeT gerar de forma automática os casos de teste.

Figura 3.5: Processo de utilização da TaRGeT

3.3 Aplicação Prática do Processo usando a TaRGeT

Para exemplificar o processo de MBT utilizado neste trabalho, vamos considerar sua aplicação dentro do processo de teste de *feature*. Será utilizada uma *feature* bem pequena e fictícia, composta pelos requisitos descritos no documento de requisitos apresentado na Figura 3.6.

Feature Mensagens Favoritas

Requisitos

R1. O usuário pode mover uma mensagem da pasta “Entrada” para a pasta “Mensagens Favoritas”.

R2. Se a memória estiver cheia, uma messagem de diálogo informando que não há memória suficiente deve ser exibida.

Figura 3.6: Exemplo fictício de documento de requisitos

Ao receber os requisitos da *feature*, o *test designer* inicia o processo de escrita de cenários de uso. Analisando os requisitos, ele percebe que precisa testar dois cenários: um onde o usuário consegue mover uma mensagem para a pasta de mensagens favoritas, e um outro no qual o usuário não consegue executar essa ação com sucesso devido a falta de memória.

Com os cenários definidos, ele inicia o preenchimento do template da TaRGeT. Nesse momento, ele precisa utilizar seus conhecimentos sobre a aplicação. Por exemplo, como chegar na opção de menu que permite mover uma mensagem para a pasta de mensagens favoritas? Esse tipo de informação não se encontra no documento de requisitos da *feature* e

para obtê-lo, o *test designer* precisa ter conhecimento sobre o funcionamento da aplicação ou ir em busca de outros documentos de requisitos, que descrevem outras *features*.

Utilizando o template da TaRGeT, um possível resultado é apresentado no cenário de uso da Figura 3.7. O cenário pode variar em termos da linguagem utilizada de acordo com o *test designer* que o escreve, por exemplo, usar a palavra “Entre” ao invés de “Acesse”. No entando, a sequência geral dos fluxos é mantida.

Main Flow

Description: Mensagem é movida para a pasta "Mensagens Favoritas".
From Step: START
To Step: END

Step Id	User Action	System State	System Response
1M	Acesse a pasta "Entrada"		A pasta "Entrada" é exibida
2M	Selecione alguma mensagem.	Existe pelo menos uma mensagem na pasta "Entrada".	A mensagem é destacada.
3M	Selecione o menu de contexto.		A opção "Mover para Mensagens Favoritas" é exibida.
4M	Selecione a opção "Mover para Mensagens Favoritas".		A mensagem é movida para a pasta "Mensagens Favoritas".

Alternative Flows

Description: Não há memória suficiente
From Step: 3M
To Step: END

Step Id	User Action	System State	System Response
1A	Selecione a opção "Mover para Mensagens Favoritas".	Memória está cheia.	Uma caixa de diálogo com a mensagem "Memória está cheia" é exibida.
2A	Confirme a caixa de diálogo.		O conteúdo da mensagem é exibido e a mensagem não é movida.

Figura 3.7: Cenário de uso para os requisitos da Figura 3.6

Além de um bom conhecimento dos requisitos da *feature* e do conhecimento do comportamento da aplicação, a escrita de cenários de uso requer do *test designer* um *feeling* de testador, que o leve a imaginar cenários que provavelmente encontrarão falhas.

Escritos os cenários de uso, a ferramenta TaRGeT os recebe como entrada e gera uma suite de testes. Nesse exemplo, apenas dois casos de teste são gerados, um que cobre o fluxo

principal e um outro que cobre o fluxo alternativo (Figura 3.8).

TC 01

Condições Iniciais

Mensagem é movida para a pasta "Mensagens Favoritas".

Passos	**Resultados Esperados**
Acesse a pasta "Entrada"	A pasta "Entrada" é exibida
Selecione alguma mensagem	A mensagem é destacada
Selecione o menu de contexto	A opção "Mover para Mensagens Favoritas" é exibida
Selecione a opção "Mover para Mensagens Favoritas".	A mensagem é movida para a pasta "Mensagens Favoritas".

TC 02

Condições Iniciais

Mensagem é movida para a pasta "Mensagens Favoritas". Não há memória suficiente

Passos	**Resultados Esperados**
Acesse a pasta "Entrada"	A pasta "Entrada" é exibida
Selecione alguma mensagem	A mensagem é destacada
Selecione o menu de contexto	A opção "Mover para Mensagens Favoritas" é exibida
Selecione a opção "Mover para Mensagens Favoritas"	Uma caixa de diálogo com a mensagem "Memória está cheia" é exibida
Confirme a caixa de diálogo	O conteúdo da mensagem é exibido e a mensagem não é movida.

Figura 3.8: Casos de teste gerados para o cenário da Figura 3.7

3.4 Conclusões

Neste capítulo foi apresentado o processo de MBT utilizado ao longo deste trabalho. O processo se adequa ao processo genérico de MBT apresentado no Capítulo 2 e encaixa suas atividades dentro do processo de teste de *feature* também apresentado no Capítulo 2.

O processo utiliza a TaRGeT como ferramenta auxiliar para geração automática dos casos de teste. Para isso, as atividades referentes à construção do modelo e geração de casos de teste do processo genérico de MBT foram adaptadas. Embora o processo se encaixe nas atividades do processo de teste de *feature*, ele não está atrelado ao mesmo, podendo ser aplicado em outros contextos.

Um dos requisitos para o sucesso de MBT é a modificação do modelo formal e não dos casos de teste gerados a partir dele, quando se deseja atualizar a suite de testes. Esse é um

problema que pode acontecer no processo de teste de *feature* apresentado, visto que podem existir vários ciclos de teste para uma mesma *feature* e a suite utilizada pode ser atualizada ao longo desses ciclos. No mundo ideal, essa atualização seria realizada nos cenários de uso e a suite de teste gerada novamente, mas no dia-a-dia isso nem sempre ocorre.

O exemplo de aplicação prática apresentado buscou dar uma idéia ao leitor de como o processo pode ser adotado no dia-a-dia e como requisitos são transformados em cenários de uso. Maiores detalhes não podem ser aqui descritos devido às restrições de sigilo, por isso uma *feature* fictícia composta por apenas dois requisitos foi criada, a fim de facilitar o entendimento por parte do leitor.

Capítulo 4

Avaliação Experimental de Estratégias para o Teste de *Feature*

Neste capítulo é apresentado um modelo de medição para avaliar o desempenho de MBT e teste exploratório quando aplicados no contexto de teste funcional de aplicações reativas, particularmente teste de *feature* no domínio de aplicações para celular. Como mencionado no Capítulo 2, essas estratégias apresentam características que as tornam candidatas interessantes ao teste de *feature*.

O teste de *feature* precisa ser rápido, é normalmente executado de forma manual, requer uma execução intensiva e repetitiva, além de requerer um bom conhecimento dos requisitos por parte dos *test designers*.

O uso de MBT pode reduzir o tempo do processo de teste através da automação da geração dos casos de teste. A facilidade de manutenção do conjunto de casos de teste é outra característica interessante para o teste de *feature*, devido à repetição da execução ao longo dos ciclos de teste e da evolução da *feature* em termos de requisitos. Com uma geração automática, pode-se atualizar os requisitos e re-gerar a suite sem maiores custos.

A abordagem exploratória pode ajudar a reduzir o número de defeitos escapados, uma vez que instiga o testador a usar seus instintos, é indicada para aplicações interativas e ajuda a aumentar o conhecimento sobre a aplicação, característica interessante para o teste de *feature* que requer um bom conhecimento dos requisitos.

O restante do capítulo está dividido da seguinte forma: a Seção 4.1 apresenta o modelo de medição, a metodologia de utilização do modelo e um estudo de caso realizado visando

a validação do mesmo. A Seção 4.2 traz uma proposta de abordagem para o teste de *feature* no domínio de aplicações para celular. Finalmente, na Seção 4.3, comentários e conclusões acerca do modelo de medição e da abordagem proposta são apresentados.

4.1 Modelo de Medição

O modelo de medição descrito nesta seção foi definido utilizando-se o paradigma GQM, apresentado no Capítulo 2.

Em linhas gerais, o paradigma é aplicado através de cinco etapas:

1. Definição dos Objetivos
2. Elaboração de Questões que Caracterizam o Objetivo
3. Especificação de Métricas que Respondem as Questões Levantadas
4. Desenvolvimento de Mecanismos para a Coleta de Dados
5. Coleta, Validação e Análise dos Dados

As três primeiras etapas concentram-se na definição do modelo propriamente dito, enquanto as demais concentram-se: i) na coleta dos dados definidos no modelo; ii) no cálculo das métricas; e iii) na análise dos resultados.

Seguindo a ordem sequencial das etapas, o primeiro passo para a construção do modelo é a definição do objetivo. O objetivo sintetiza o que se deseja avaliar, ou seja, define o propósito do estudo. De acordo com o paradigma GQM, a definição de um objetivo inicia-se com a escolha do objeto de estudo e em seguida com a resposta da questão: Por que esse objeto precisa ser estudado?

O objeto de estudo são abordagens de teste, particularmente MBT e teste exploratório. É necessário medir os desempenhos dessas abordagens, em termos de custos e detecção de defeitos, para que se obtenham históricos ao longo dos ciclos de teste. Dessa forma, é possível acompanhar o melhoramento ou não do processo de teste quando as mesmas são utilizadas e avaliar se as mesmas são adequadas ou não para serem aplicadas no processo de teste de *feature*.

A segunda etapa é a elaboração de questões, que se remete a parte operacional da elaboração do modelo. As questões devem ser definidas visando atingir o objetivo proposto na etapa conceitual. O objetivo é avaliar o desempenho da estratégia de teste quando aplicada no contexto do teste de *feature* de aplicações para celular. Como avaliar desempenho? Uma possibilidade é defini-lo em termos do esforço necessário para aplicar a estratégia e da eficiência da mesma na detecção de defeitos.

A especificação de métricas corresponde ao nível quantitativo do modelo. Métricas devem ser definidas para responder às questões elaboradas na etapa operacional. Para medir o desempenho, dois indicadores interessantes são esforço e eficiência. Nesse caso, três métricas podem ser consideradas: esforço, taxa de detecção de defeitos e relevância média dos defeitos.

Quando se pensa em esforço, se pensa inicialmente no tempo requerido para se aplicar algo. Então, o esforço E é definido como sendo o tempo requerido para se aplicar a estratégia:

$E = T_p + T_e$,

onde T_p é o tempo de preparo, T_e é o tempo de execução dos testes e validação dos resultados, particularmente no caso de teste exploratório onde algumas vezes se faz necessário investigar se a falha de fato ocorreu. O tempo de preparo T_p se modifica de acordo com a estratégia adotada:

- **Teste exploratório** - o tempo de preparo T_p compreende o tempo de leitura dos documentos de requisitos da *feature* e o tempo de definição do plano de testes.

- **Teste baseado em modelo** - o tempo de preparo T_p compreende o tempo de leitura dos documentos de requisitos da *feature*, o tempo de construção do modelo e o tempo de geração dos casos de teste. A construção do modelo pode variar de acordo com o modelo formal utilizado e com o uso de ferramentas que geram o modelo formal de forma automática a partir de uma dada entrada. O tempo de geração pode ser insignificante se utilizada uma ferramenta de geração automática.

No contexto de teste de *software*, a eficiência de uma estratégia está diretamente associada à taxa de defeitos detectados. Sendo assim, a taxa de detecção de defeitos é definida da seguinte forma:

$T_{def} = \frac{N_f}{E}$,

onde N_f é o número de defeitos detectados durante a execução dos testes, E é o esforço requerido. É importante lembrar que o número de defeitos detectados pode diferir do número de casos de teste que falharam, isso porque dois ou mais casos de teste podem detectar o mesmo defeito.

Apesar do número de defeitos escapados ser uma métrica importante, pois pode indicar problemas no processo de teste, ela não foi considerada no modelo. O cálculo de defeitos escapados só pode ser realizado em etapas seguintes do processo de teste e, como o modelo foi definido para ser aplicado durante a fase de teste de *feature*, o número de defeitos escapados não foi aqui considerado. Caso o modelo seja aplicado dentro de uma organização na qual o número de defeitos escapados não pode ser descartado durante essa fase, uma possível alternativa é a utilização de estimativa, onde o número de defeitos escapados seria estimado de acordo com alguma função que utilizasse dados históricos da organização, por exemplo.

Na prática, uma estratégia de teste A pode ter uma eficiência maior que uma estratégia de teste B, e no entanto, B ser mais interessante por detectar defeitos mais relevantes. Por esse motivo, uma outra métrica a ser considerada para avaliar desempenho é a relevância média dos defeitos detectados. Definimos então a relevância R como:

$R = \frac{\sum R_f}{N_f}$,

onde N_f é o número de defeitos detectados durante a execução dos testes; R_f é um valor atribuído ao defeito conforme sua relevância, seguindo algum critério pré-definido. Para atribuir valores aos defeitos, seguimos o seguinte critério de pontuação:

- **1** - Problema **transparente**, invisível ao usuário. Pode ser corrigido em uma *build* posterior sem justificativa formal. Exemplo: *layout* mal definido, erro gramatical no manual.

- **2** - Problema **pequeno** que não impede o usuário de completar a ação que estava executando. O usuário pode ou não perceber o problema. Exemplo: Mensagens de erro ambíguas.

- **3** - Problema **moderado** que dificulta mas não impede que o usuário realize a função desejada. Exemplo: dados precisam ser modificados pra que a função possa ser realizada.

- **4** - Problema **sério** que causa perda de funcionalidade e desempenho. Exemplo: impossibilidade de usar todas as funcionalidades do produto.
- **5** - Problema **crítico** que impossibilita o uso do produto. Exemplo: *crash* do sistema, perda de dados do usuário.

A Figura 4.1 apresenta o resultado das etapas referentes à definição do modelo. No nível conceitual (Nível 1) tem-se o objetivo que se quer atingir. No nível operacional (Nível 2), as questões que auxiliam no alcance desse objetivo. E, finalmente no nível quantitativo (Nível 3), as métricas que devem ser calculadas para responder as questões do nível dois.

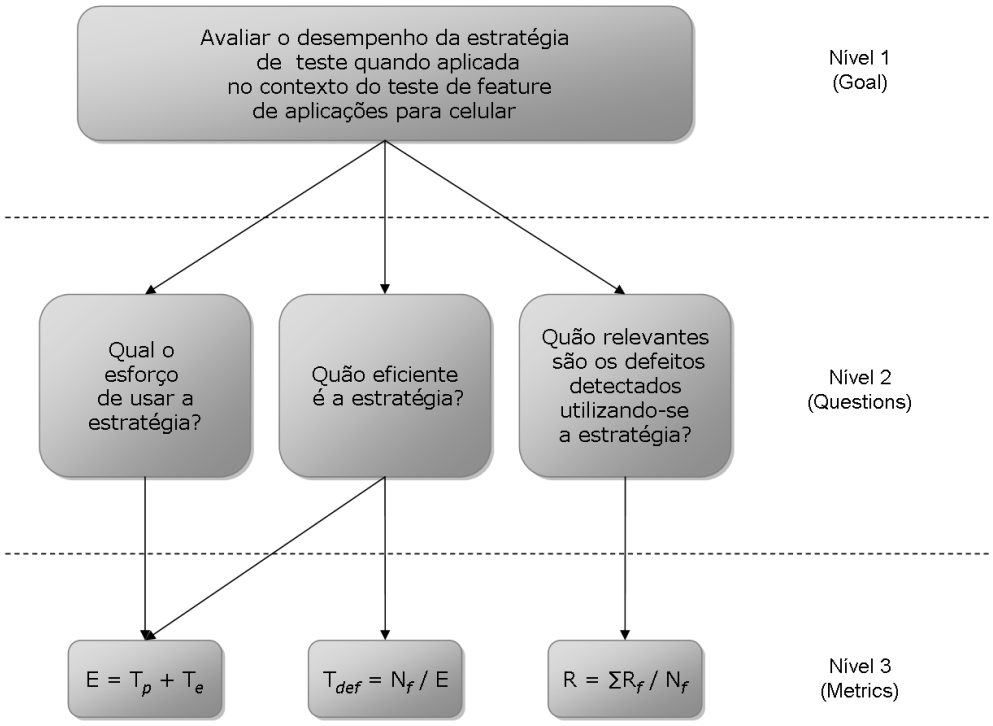


Figura 4.1: Modelo de Medição

Na próxima seção é apresentada a metodologia de aplicação, que corresponde às etapas 4 e 5 do paradigma GQM.

4.1.1 Metodologia de Aplicação

A aplicação do modelo de medição consiste em coletar os dados necessários para o cálculo das métricas, bem como analisar os resultados obtidos. Considerando MBT, as seguintes

atividades são necessárias para coletar os dados:

1. **Construção do modelo** - A geração do modelo comportamental consiste em mapear os requisitos funcionais para uma notação formal ou semi-formal que representará o comportamento funcional da *feature*. Essa atividade pode ser automatizada. Cabral [Cabral and Sampaio, 2006] apresenta um método para geração automática de uma especificação formal a partir de documentos de requisitos.

2. **Geração dos casos de teste** - A partir do modelo comportamental, os casos de teste são gerados. A geração é passível de automação. Cartaxo e Nogueira [Cartaxo, 2006; de Carvalho Nogueira, 2006] apresentam ferramentas para geração automática de casos de teste a partir de modelos LTS e CSP, respectivamente.

3. **Execução dos casos de teste** - Consiste na execução dos casos de teste gerados na atividade anterior. Ferramentas que registram o tempo de execução e o resultado do caso de teste podem ser utilizadas, assim como ferramentas de log, que permitem o registro do passo a passo da execução do teste.

Considerando teste exploratório, as seguintes atividades são necessárias:

1. **Definição do plano de testes** - Compreende a leitura dos documentos de requisitos e a escrita do plano de testes. O plano de testes define uma ou mais sessões de teste (Para maiores detalhes acerca de uma sessão de teste, ver Seção 2.2).

2. **Execução do plano de testes** - Essa atividade consiste na execução das seções de teste definidas no plano de testes. Ferramentas de log podem ser utilizadas, uma vez que permitem o registro do passo a passo da execução do teste, auxiliando em uma posterior re-execução.

A coleta dos dados deve ocorrer durante as atividades listadas. Para calcular o esforço (E), é preciso coletar dados sobre o tempo gasto nas atividades envolvidas. No caso da taxa de defeitos detectados (T_{def}) e da relevância (R), é preciso guardar os resultados da execução dos casos de teste, no caso de MBT, e o resultado da execução das sessões de teste, no caso de teste exploratório. A Tabela 4.1 apresenta a relação entre as atividades e os dados que devem ser coletados em cada uma.

Tabela 4.1: Atividades e métricas que poderão ser calculadas a partir dos dados coletados.

Atividade	**Dado**
Definição do plano de testes (TE)	T_p
Execução do plano de testes (TE)	T_e, N_f, R_f
Construção do modelo (MBT)	T_p
Geração dos casos de teste (MBT)	T_p
Execução dos casos de teste (MBT)	T_e, N_f, R_f

Em qualquer estudo de caso, a validade dos dados precisa ser considerada. É essencial que dados reais sejam coletados de forma natural para que o resultado da análise reflita o que acontece na prática.

A análise dos dados compreende o cálculo e a interpretação das métricas: os dados coletados são recuperados e as métricas calculadas. Além das métricas sugeridas para cada estratégia, uma métrica interessante para a análise é a similaridade dos defeitos detectados. Com essa métrica é possível descobrir quais dos defeitos detectados são comuns às duas estratégias, analisar se existem padrões nos defeitos detectados, dentre outras descobertas.

O conjunto de defeitos detectados usando teste exploratório e MBT $D_{TE \bigcap MBT}$ é definido como:

$D_{TE \bigcap MBT} = D_{TE} \bigcap D_{MBT}$,

onde D_{TE} é o conjunto de defeitos detectados aplicando-se teste exploratório; D_{MBT} é o conjunto de defeitos detectados aplicando-se teste baseado em modelo.

Os resultados obtidos podem ser interpretados de inúmeras maneiras. A interpretação varia conforme a situação na qual o estudo é realizado. O valor da eficiência, por exemplo, pode ser menos importante que o valor da relevância, o que significa que se uma estratégia A for menos eficiente que uma estratégia B, mas detectar defeitos mais importantes ela é mais adequada ao teste de *feature*. O esforço elevado de uma estratégia A pode ser compensado se ela apresenta uma relevância razoável. Nesse caso, se o objetivo maior é encontrar defeitos relevantes, a estratégia A torna-se adequada, embora requeira um esforço elevado. Em resumo, determinar a adequação de qualquer estratégia considerando esforço, eficiência e/ou relevância, vai depender diretamente do objetivo do teste: encontrar o maior número de

defeitos no menor tempo; encontrar o maior número de defeitos relevantes independente do esforço; encontrar defeitos relevantes mas levar em consideração esforço; etc.

A metodologia apresentada corresponde às atividades de processos genéricos de aplicação de MBT e teste exploratório. Contudo, a utilização do modelo não é dependente desta metodologia, podendo o mesmo ser utilizado em qualquer processo onde seja possível coletar os dados necessários para o cálculo das métricas.

4.1.2 Estudo de Caso

Esta seção apresenta um estudo empírico realizado para ajustar o modelo de medição definido na Seção 4.1. Inicialmente são apresentados ambiente e metodologia utilizados, e em seguida uma análise dos resultados obtidos é realizada.

Ambiente e Metodologia

O estudo foi conduzido dentro do ambiente Motorola por um pesquisador inexperiente na escrita de casos de teste. A Tabela 4.2 apresenta algumas informações acerca do pesquisador.

Tabela 4.2: Dados do participante.

	SIM	NÃO
Pesquisador	X	
Test Designer		X
Experiência		X
Conhecimento das *features*		X
Conhecimento da ferramenta	X	

Nesse estudo de caso, duas *features* foram utilizadas. Seus requisitos são descritos em documentos de requisitos que especificam o comportamento individual de uma *feature*. A Tabela 4.3 apresenta detalhes sobre as mesmas.

As atividades foram executadas considerando a *feature* A e posteriormente a *feature* B. No entanto, para uma melhor compreensão, são apresentadas as atividades relativas a MBT para as duas *features* e em seguida as atividades relativas a teste exploratório. Não foi feito

Tabela 4.3: Informações sobre as *features* utilizadas.

	$N^0 de Documentos$	$N^0 de Requisitos$
Feature A	2	44
Feature B	2	50

uso de ferramentas para coletar as métricas, apenas um relógio para cronometrar o tempo gasto em cada atividade foi utilizado.

Model-Based Testing Seguindo o processo apresentado no Capítulo 3, a atividade inicial é a construção do modelo. Devido ao uso da ferramenta TaRGeT [Nogueira et al., 2007], a construção do modelo transformou-se na especificação de cenários de uso conforme o *template* apresentado em Cabral [Cabral and Sampaio, 2006].

Concluída a escrita dos cenários de uso, os casos de teste foram gerados de forma automática, utilizando TaRGeT, e executados de forma manual. As Tabelas 4.4 e 4.5 mostram dados coletados durante a execução. Todos os casos de teste referentes a *feature* A foram executados. Considerando a *feature* B, os casos de teste 04 e 07 não foram executados devido a problemas existentes na versão executável do *software* utilizada e que imposibilitaram a execução. O caso de teste 03 foi bloqueado devido a alguma condição não satisfeita ao longo do teste e que impediu a conclusão da execução, como por exemplo algum passo que o testador não sabia como executar.

A Tabela 4.6 apresenta os dados coletados durante as atividades de MBT. Devido a geração automática do modelo e dos casos de teste, T_p refere-se basicamente a leitura dos documentos de requisitos e a escrita dos cenários de uso seguindo o template requerido.

Por fim, as métricas foram calculadas com base nos dados coletados. A Tabela 4.7 apresenta as métricas e seus respectivos valores. Como não foram detectados defeitos utilizando o teste baseado em modelo, a relevância não pode ser calculada.

Tabela 4.4: Dados coletados durante a execução dos casos de teste referentes a *feature* A.

	Tempo gasto na execução do caso de teste (seg)	Resultado
Caso de teste 01	54	Passou
Caso de teste 02	45	Passou
Caso de teste 03	46	Passou
Caso de teste 04	35	Passou
Caso de teste 05	58	Passou
Caso de teste 06	45	Passou
Caso de teste 07	182	Passou
Caso de teste 08	50	Passou
Caso de teste 09	42	Passou
Caso de teste 10	40	Passou
Caso de teste 11	29	Passou

Tabela 4.5: Dados coletados durante a execução dos casos de teste referentes a *feature* B.

	Tempo gasto na execução do caso de teste (seg)	Resultado
Caso de teste 01	46	Passou
Caso de teste 02	57	Passou
Caso de teste 03	1120	Bloqueado
Caso de teste 04	-	Não executado
Caso de teste 05	81	Passou
Caso de teste 06	25	Passou
Caso de teste 07	-	Não executado

Teste Exploratório Após a escolha das *features*, os planos de teste foram definidos a partir da leitura dos documentos de requisitos e da definição das sessões de teste.

Cada sessão de teste foi definida para ter uma hora de duração. Ao final da execução dos planos, um relatório informal foi escrito com um resumo das informações coletadas: tempo

Tabela 4.6: Dados coletados utilizando MBT.

	T_p (min)	T_e (min)	N_f
Feature A	127	10.44	0
Feature B	146	22.15	0

Tabela 4.7: Métricas obtidas aplicando teste baseado em modelo.

	Feature A	Feature B
Esforço (min)	137	168
Eficiência	0	0
Relevância	$\perp$	$\perp$

gasto e defeitos detectados. A Tabela 4.8 mostra os dados coletados para cada *feature*.

Tabela 4.8: Dados coletados usando TE.

	T_p (min)	T_e (min)	N_f
Feature A	78	50	0
Feature B	77	51	1

Após a execução das sessões de teste, as métricas definidas foram calculadas com base nos dados coletados. A Tabela 4.9 mostra as métricas calculadas com o uso do teste exploratório no contexto do teste de *feature*.

Tabela 4.9: Métricas obtidas aplicando teste exploratório.

	Feature A	Feature B
Esforço (min)	128	128
Eficiência	$\perp$	$0,51$
Relevância	$\perp$	1

Análise dos Resultados

Analisando os resultados apresentados nas Tabelas 4.10 e 4.11, percebe-se que usando teste exploratório um defeito com relevância igual a 1 foi encontrado durante o estudo que utilizou a *feature* B. Usando a abordagem MBT, nenhum defeito foi detectado. Percebe-se também que o esforço em aplicar teste exploratório foi menor que o esforço em aplicar MBT, tanto utilizando a *feature* A quanto a *feature* B .

Tabela 4.10: Comparativo de métricas - feature A.

	Esforço (min)	Eficiência	Relevância
Teste Exploratório	128	0	$\perp$
Teste Baseado em Modelo	137	0	$\perp$

Tabela 4.11: Comparativo de métricas - feature B.

	Esforço (min)	Eficiência	Relevância
Teste Exploratório	128	$0,46$	1
Teste Baseado em Modelo	168	0	$\perp$

O uso de MBT possibilita uma cobertura de 100% dos requisitos, o mesmo não ocorrendo com teste exploratório. No entanto, o teste exploratório possibilita uma maior cobertura de estados, visto que o testador tem a liberdade de explorar a aplicação, diferentemente de MBT, onde os casos de teste gerados seguem uma seqüência de passos pré-determinados. Essa maior cobertura de estados aumenta a probabilidade de detectar defeitos escondidos, fato que pode ajudar a explicar a detecção de um defeito utilizando teste exploratório.

Os critérios de parada foram: i) cobertura dos requisitos através dos cenários de uso, considerando MBT; e ii) execução das sessões de teste, considerando teste exploratório. Talvez um critério comum de parada como, por exemplo, *esforço aplicado ou número estimado de defeitos escapados (quando acrescido ao modelo de medição definido)*, seja mais interessante, pois equipara as condições nas quais as estratégias são aplicadas.

Os celulares utilizados na execução dos casos de teste foram preparados com versões de *software* já testadas por outros times. Talvez, por esse motivo, apenas um defeito foi detectado. Embora a probabilidade de detecção de defeitos ser baixa desde o início do estudo de caso, o estudo foi levado adiante para validar e melhorar o modelo de medição proposto. Além disso, o estudo reforçou a idéia que teste exploratório é uma abordagem interessante para detectar defeitos escondidos, diminuindo o número de defeitos escapados que são um problema real no teste de *feature*.

No mundo ideal, sabe-se que estudos de caso devem ser conduzidos de forma independente. No estudo de caso realizado, as abordagens deveriam ter sido aplicadas por pessoas distintas, para garantir que o conhecimento adquirido com uma abordagem não fosse utilizado na aplicação da outra abordagem. As dificuldades para conseguir realizar o estudo, principalmente as de preparação do ambiente, impediram que o estudo fosse realizado como o mundo ideal requer. Embora teste exploratório tenha sido aplicado antes de MBT, auxiliando na aquisição de conhecimento sobre o comportamento da aplicação, ao se escrever os cenários de uso requeridos por MBT buscou-se não fazer uso do conhecimento adquirido com TE.

4.2 Abordagem para o Teste de *Feature*

A partir dos resultados preliminares do estudo de caso e do conhecimento sobre teste exploratório e MBT, percebeu-se que uma abordagem conjunta seria bem interessante. A Figura 4.2 apresenta essa abordagem.

Na primeira fase, teste exploratório é utilizado como ferramenta para aquisição de conhecimento sobre o comportamento da aplicação. O *log* de execução pode ser capturado e utilizado posteriormente na complementação do modelo formal do comportamento da aplicação.

Em uma segunda fase, MBT é aplicado. O modelo formal pode ser gerado de forma automática a partir de requisitos descritos em uma linguagem natural controlada (LNC). A partir deste modelo, casos de teste podem ser gerados.

O uso de MBT em uma segunda fase é interessante devido à execução de mais de um ciclo de teste a partir de uma mesma suite de testes. Se a geração dos casos de teste acontece

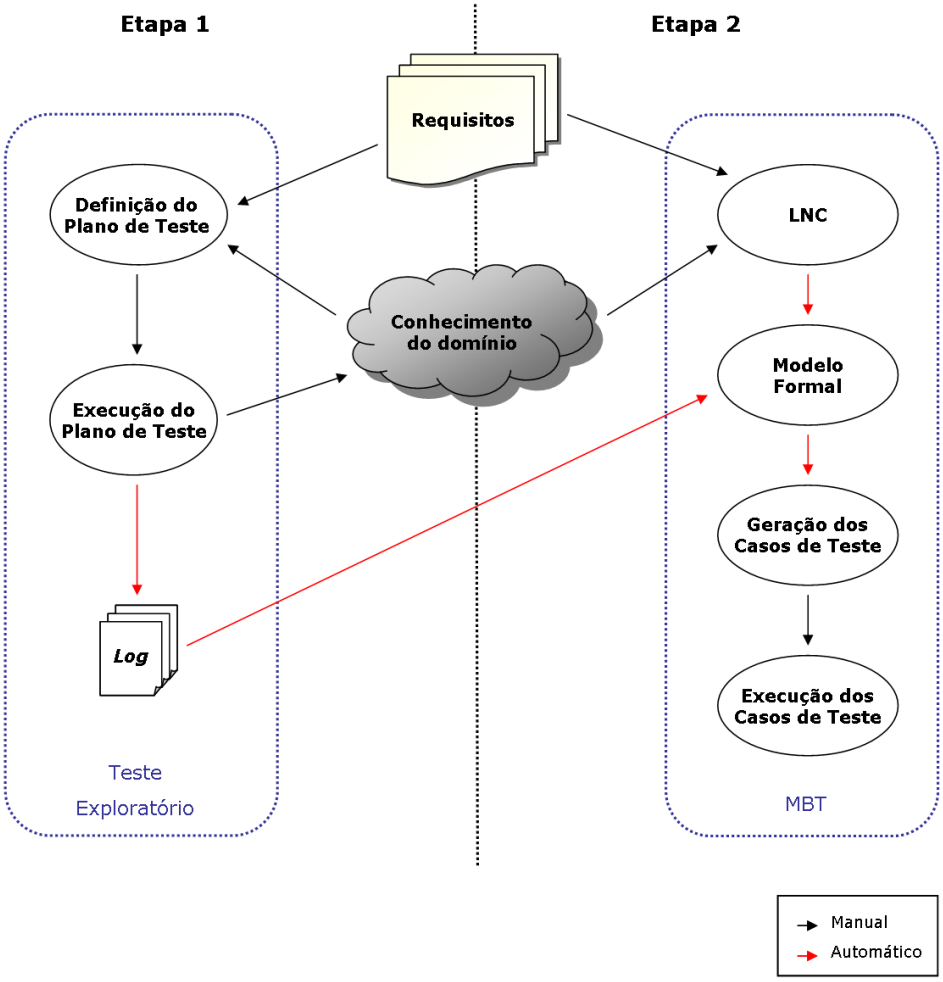


Figura 4.2: Abordagem para o teste de *feature*

de forma automática, o custo de manutenção da suite é bastante reduzido, dado que uma atualização nos requisitos pode ser refletida imediatamente nos casos de teste, desde que refletida no modelo formal.

Durante a realização deste trabalho, não foi possível validar essa estratégia conjunta aplicando-a em uma situacao real. No entanto, durante a realização do estudo de caso apresentado na Seção 4.1.2, teste exploratório foi aplicado antes de MBT. Embora o conhecimento adquirido com TE não tenha sido aplicado na escrita dos cenários de uso, para preservar a validade do estudo de caso, o sentimento com relação ao uso de teste exploratório como ferramenta para aquisição de conhecimento foi confirmado, pois a ecrita dos cenários de uso poderia ter sido facilitada com o uso do conhecimento adquirido sobre o

funcionamento da aplicação.

4.3 Conclusões

Neste capítulo foi apresentado um modelo de medição para avaliar o desempenho do teste exploratório e de MBT quando utilizados no teste de *feature* para aplicações para celular. Uma metodologia para coleta dos dados também foi apresentada.

A principal vantagem da metodologia sugerida é a equivalência entre as atividades propostas e as atividades que fazem parte do teste exploratório e do teste baseado em modelo. Uma vez que experimentar tem um alto custo, devido ao tempo que demanda e aos cuidados necessários para que os dados coletados sejam válidos e bem analisados, adaptar uma metodologia de avaliação ao processo já existente, contribui para o não aumento desse custo. Embora a metodologia apresentada corresponda às atividades de processos genéricos de aplicação de MBT e teste exploratório, a utilização do modelo de medição independente da mesma, podendo o mesmo ser utilizado em qualquer processo onde seja possível coletar os dados necessários para o cálculo das métricas.

A partir dos estudos de caso realizados, consolidou-se a percepção de que teste exploratório e MBT possuem características que podem contribuir com a melhoria do teste de *feature*. Dessa forma, uma abordagem conjunta das duas estratégias foi proposta.

Dentro do contexto deste trabalho, não foi possível validar a estratégia conjunta em uma situacao real, pois isto implicaria em mudanças bruscas no processo de desenvolvimento e testes, principalmente por teste exploratorio ser utilizado em uma etapa inicial da estratégia.

Embora a abordagem proposta não seja aplicável no ambiente Motorola, ela é interessante para ambientes onde os documentos de requisitos são escassos, e *test designers* e desenvolvedores trabalham de forma isolada. Nesse cenário, a utilização de teste exploratório como ferramenta de aquisição de conhecimento é uma boa opção, pois pode acelerar o processo de conhecimento da aplicação sob teste e instigar o *test designer* na descoberta de novos cenários bem como de defeitos escondidos, muitas vezes não detectados por testes pré-definidos.

Capítulo 5

Avaliação Experimental de MBT

Este capítulo apresenta um modelo de medição para acompanhar o desempenho de MBT ao longo dos ciclos de teste. A Seção 5.1 explica o porquê da definição do modelo de medição. A Seção 5.2 apresenta o modelo de medição definido, a metodologia de utilização do modelo e dois estudos de caso realizados aplicando o mesmo. Finalmente, na Seção 5.3 são feitas algumas considerações.

5.1 Contextualização

Como dito anteriormente, uma das maiores dificuldades em aplicar MBT encontra-se na construção do modelo. Muitos podem ser os problemas: notação utilizada na construção do modelo, nível de abstração do modelo, requisitos incompletos, requisitos não testáveis. Identificar onde estão as falhas permite a correção das mesmas e uma consequente melhoria no processo de teste.

Ao longo de seis meses o processo de teste de *feature* na Motorola foi observado e alguns estudos de caso nos quais aplicou-se MBT na criação dos casos de teste de *feature* foram conduzidos. Dentre os problemas encontrados, boa parte envolve os requisitos:

- **Requisitos não testáveis** - Requisitos do tipo: "Assume-se que *Messaging* e *Phone-book* são *features* habilitadas no aparelho." são requisitos dos quais se deriva uma condição inicial e não um caso de teste.
- **Requisitos implícitos** - No Capítulo 2 foram apresentados os conceitos de *feature*

e teste de *feature*. Foi dito que a *feature* é desenvolvida de forma evolutiva, seja através da evolução de requisitos seja através da evolução do código. Muitas vezes, a evolução por meio dos requisitos torna os requisitos implícitos ou desatualizados, pois aplicações similares são desenvolvidas e muitas *features* são parecidas, levando a não "repetição" dos requisitos, tornando-os implícitos.

- **Linguagem figurada** - Algumas vezes a linguagem utilizada nos requisitos requer interpretação do *test designer*. Por exemplo, um requisito que afirma que o usuário pode estar ou não em alguma tela, significa que determinada *feature* precisa estar habilitada ou não no aparelho.

- **Linguagens diferentes** - A linguagem dos requisitos é diferente da linguagem utilizada nos casos de teste. Por exemplo, a partir de um requisito que contém *turn off speaker* será escrito um caso de teste que contém *Press Speaker OFF softkey*. Isto dificulta a construção automática de um modelo formal da aplicação sob teste a partir dos requisitos.

A partir da observação do processo e da detecção de possíveis problemas que podem prejudicar o desempenho de MBT, definiu-se um modelo de medição para acompanhar esse desempenho ao longo dos ciclos de teste, buscando dessa forma, a melhoria do processo de teste. A seção seguinte apresenta esse modelo.

5.2 Modelo de Medição

Assim como no modelo apresentado no capítulo anterior, o paradigma GQM foi utilizado na definição do modelo apresentado nesta seção. O objeto de estudo é MBT. Seu desempenho precisa ser medido e acompanhado para detectar possíveis falhas no processo, bem como avaliar se a adoção de MBT é adequada.

O objetivo é acompanhar o desempenho de MBT ao longo dos ciclos de teste, mas como realizar esse acompanhamento? Uma possibilidade é coletar dados relativos a esforço, requisitos e resultados dos casos de teste e analisá-los ao longo dos ciclos de teste.

Ao se utilizar MBT, pode-se associar o esforço às horas dedicadas à geração dos casos de teste. Esta é uma métrica interessante para medir produtividade e fazer estimativas. Geral-

mente, ao se usar MBT, utiliza-se uma ferramenta para geração automática dos casos de teste a partir do modelo. Dessa forma, o esforço concentra-se na leitura dos documentos de requisitos, na construção do modelo formal da aplicação sob teste e na inspeção dos artefatos produzidos. Então, esforço é definido como:

$E = T_p + T_i$,

onde T_p é o tempo para ler os documentos de requisitos e construir o modelo. Quando uma ferramenta de MBT é utilizada, a construção do modelo pode ser substituída pela criação de documentos. TaRGeT [Nogueira et al., 2007] é um exemplo de ferramenta de geração de casos de teste a partir de modelo, onde o modelo é extraído de documentos de caso de uso de forma automática; T_i é o tempo de inspeção dos artefatos produzidos. Esses artefatos podem ser o próprio modelo ou documentos a partir dos quais o modelo é derivado, e os casos de teste gerados. Inspeções são importantes para reduzir a probabilidade do artefato ter sido produzido de forma incorreta. Pode-se também calcular o esforço por caso de teste, dividindo o esforço total pelo número total de casos de teste gerados.

Outra métrica importante é cobertura de requisitos. Ela é útil quando deseja-se verificar se todos os requisitos da aplicação estão sendo testados. Como existem requisitos testáveis e não-testáveis, é interessante considerar apenas os requisitos testáveis, já que estes contém informação suficiente para derivar um ou mais casos de teste. Sendo asim, o percentual de requisitos testáveis cobertos é calculado como,

$RC = (\frac{N_{rtc}}{N_{rt}}) * 100\%$,

onde N_{rtc} é o número de requisitos testáveis cobertos; N_{rt} é o número total de requisitos testáveis.

Além de saber o percentual de requisitos cobertos, é interessante saber o percentual de requisitos não testáveis, para assim, avaliar a testabilidade dos requisitos. Se este percentual estiver alto, significa que provavelmente há problemas na especificação, como, por exemplo, nível de abstração e completude dos requisitos. Esses problemas podem refletir diretamente na qualidade dos casos de teste. O percentual de requisitos não testáveis é definifido como:

$RNT = (\frac{N_r - N_{rt}}{N_r}) * 100\%$,

onde N_{rt} é o número total de requisitos testáveis; N_r é o número total de requisitos.

Ao se fazer inspeções nos casos de teste gerados e executá-los, pode-se obter uma métrica interessante: o percentual de casos de teste modificados. Um caso de teste é modificado se ele

não estiver de acordo com os requisitos ou com alguma parte da aplicação que é necessária para execução mas que não está especificada nos requisitos. Por exemplo, se o *test designer* está criando testes para um aparelho novo e não conhece o comportamento da aplicação, ele pode supor que para o usuário realizar determinada ação especificada nos requisitos, o usuário provavelmente deve realizar uma outra ação. Essa outra ação pode estar errada (já que o comportamento é desconhecido) e esse problema é detectado durante a execução, precisando o caso de teste ser modificado. Um percentual alto de casos de teste modificados pode ser um indicativo de que há problemas na especificação dos requisitos ou que o time que define o modelo está com dificuldades de entender os requisitos ou definir o modelo na notação adotada. O percentual de casos de teste modificados é calculado como:

$CTM = (\frac{N_{ctm}}{N_{ct}}) * 100\%$,

onde N_{ctm} é o número de casos de teste modificados durante a inspeção ou execução dos casos de teste; N_{ct} é o número de casos de teste gerados a partir do modelo.

Um baixo percentual de casos de teste falhos pode ser um indicativo de que a suite de testes não está sendo eficiente. Um dos sete princípios gerais de teste é o paradoxo do pesticida [Graham et al., 2007], que diz que se uma mesma suite é executada ao longo de vários ciclos de teste, provavelmente chegará um momento no qual nenhum novo defeito será encontrado. Esse indicador é interessante para estimular o *test designer* a pensar em novos cenários e pode ser calculado como:

$CTF = (\frac{N_{ctf}}{N_{ct}}) * 100\%$,

onde N_{ctf} é o número de de casos de teste que falharam; N_{ct} é o número de casos de teste gerados a partir do modelo.

A Figura 5.1 apresenta os três níveis do modelo de medição definido: conceitual(Goal), operacional (Questions) e quantitativo (Metrics).

Medir e manter um histórico é importante pois possibilita responder algumas questões no futuro: MBT torna a suite de testes consistente e facilita a manutenção dos casos de teste? MBT melhora a forma como os requisitos são escritos? As métricas E, RC e CTM auxiliam na resposta da primeira questão. A métrica RNT ajuda na resposta da segunda.

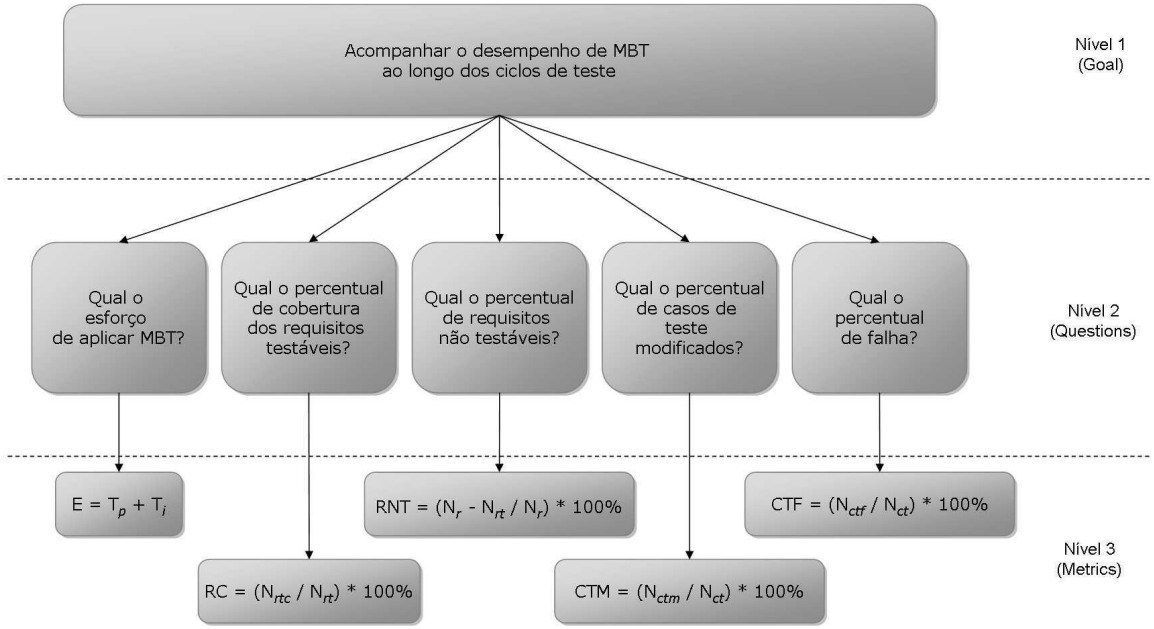


Figura 5.1: Modelo de medição para o acompanhamento do desempenho de MBT

5.2.1 Metodologia de Aplicação

Para utilizar o modelo de medição, é preciso coletar os dados necessários para o cálculo das métricas, calculá-las e analisar os resultados obtidos. Na prática, o processo de aplicação de MBT pode ser dividido em quatro etapas:

1. **Leitura de requisitos e definição do modelo formal** - Envolve a leitura dos documentos de requisitos, a compreensão dos requisitos e a construção do modelo formal. Pode-se utilizar alguma ferramenta para auxiliar ou automatizar a escrita do modelo.

2. **Inspeção** - Diz respeito a inspeção dos artefatos produzidos, como o modelo gerado ou a entrada de alguma ferramenta a partir da qual o modelo deverá ser gerado, e os casos de teste gerados.

3. **Geração** - Consiste na geração dos casos de teste a partir do modelo formal. Geralmente ferramentas que automatizam essa geração são utilizadas.

4. **Execução** - Envolve a execução dos casos de teste gerados na etapa de geração, e o armazenamento dos resultados.

Ao final ou ao longo de cada etapa, é possível coletar os dados definidos no modelo de medição. A Figura 5.2 apresenta a relação entre essas etapas e as métricas que serão calculadas a partir dos dados coletados em cada uma.

Leitura de requisitos e definição do modelo formal	T_p	N_{rt}	N_r
Inspeção	T_i		N_{ctm}
Geração dos casos de teste		N_{rtc}	N_{ct}
Execução dos casos de teste		N_{ctf}	N_{ctm}

Figura 5.2: Etapas de MBT x Métricas

Os dados para cálculo das métricas são coletados a partir do tempo gasto em determinadas atividades, como por exemplo, definição do modelo e inspeção, e a partir dos artefatos produzidos, como o número de casos de teste, por exemplo. Esses dados podem então ser coletados pelos *test designers* e testadores e reportados em cada ciclo de teste. Armazenados em alguma base de dados, eles permitem fazer estimativas e medições. As próximas seções apresentam estudos de caso nos quais utiliza-se o modelo definido na seção anterior.

5.2.2 Estudo de Caso 1

Nesta seção é apresentado um estudo de caso inicial, realizado com o objetivo de ajustar o modelo de medição definido na Seção 5.2 e de comparar a aplicação de MBT por dois perfis diferentes, com o intuito de verificar o quão importante é a experiência no momento de se aplicar MBT de forma eficiente. Inicialmente são apresentados ambiente e metodologia utilizados, e em seguida uma análise dos resultados obtidos é realizada.

Ambiente e Metodologia

O estudo aconteceu no ambiente Motorola, sendo monitorado por um pesquisador e conduzido por um *test designer* experiente e por um pesquisador inexperiente na escrita de

casos de teste. A Tabela 5.2 apresenta as informações sobre os participantes.

Tabela 5.1: Dados sobre participantes.

	Participante A	Participante B
Pesquisador	X	
Test Designer		X
Experiência na escrita de casos de teste	$Não$	Sim
Conhecimento da *feature*	$Não$	$Não$
Conhecimento da ferramenta TaRGeT	$Não$	$Não$

Assim como no estudo de caso apresentado no Capítulo 4, o processo de MBT apresentado no Capítulo 3 foi utilizado. Nenhum dos participantes conhecia a ferramenta TaRGeT ou o template por ela utilizado, dessa forma, o esforço de apresendizado foi desconsiderado.

Nesse estudo de caso, apenas uma *feature* foi utilizada. A *feature* é descrita em dois documentos de requisitos e possui 56 requisitos, dos quais 54 são testáveis. O RNT da *feature*, considerando a versão utilizada dos documentos de requisitos, é

$RNT = (\frac{56-52}{56}) * 100\% \approx 7,15\%$.

Cada participante foi responsável por ler os documentos de requisitos, escrever os cenários de uso e registrar o tempo gasto. O pesquisador condutor do estudo de caso foi responsável por gerar os casos de teste e fazer a análise dos dados. A Tabela 5.2 apresenta os dados obtidos para cada participante.

Tabela 5.2: Dados obtidos.

	Participante A	Participante B
T_p (min)	258	840
N_{rtc}	4	52
N_{ct}	8	44

O intuito do estudo não é a verificação do número de defeitos detectados, mas a comparação da cobertura de requisitos quando MBT é aplicado por pessoas com perfis profissionais

diferentes. Dessa forma, os casos de teste gerados não foram executados. O tempo de inspeção T_i não foi registrado, uma vez que não foi feita inspeção nos artefatos gerados, portanto o esforço E foi calculado considerando-se apenas o tempo de preparação T_p. A Tabela 5.3 apresenta as métricas calculadas para cada participante. Na próxima seção, é feita uma análise dos resultados obtidos.

Tabela 5.3: Métricas obtidas por Participante.

	Participante A	Participante B
E (min)	258	840
$RC(\%)$	$7,7$	100
$RNT(\%)$	$7,15$	$7,15$
N_{ct}	8	44

Análise dos resultados

Ao observar os dados apresentados na Tabela 5.3, perecebe-se uma grande diferença no número de casos de teste gerados e na cobertura de requisitos obtida pelo pesquisador inexperiente (participante A) e pelo *test designer* (participante B). O gráfico apresentado na Figura 5.3 mostra que o esforço é proporcional ao número de casos de teste gerados. O esforço do participante A foi bem inferior ao esforço do participante B, o que nos leva a questionar porque existe tamanha diferença. Será que é devido à experiência na escrita de casos de teste? Será que essa experiência tem uma grande influência no entendimento dos requisitos?

Uma outra diferença que chama a atenção é a cobertura de requisitos RC. O participante B cobriu 100% dos requisitos testáveis, enquanto o participante A cobriu apenas 7%. O gráfico apresentado na Figura 5.4 mostra esses números. O percentual de requisitos cobertos afeta diretamente o número de casos de teste gerados a partir dos cenários de uso. A partir dos cenários de uso escritos pelo participante B, foram gerados 44 casos de teste, contra apenas 8 do participante A.

O percentual de requisitos não testáveis RNT foi $7,15\%$. Este é um valor razoável ou ainda está elevado? Questões como esta precisam ser respondidas para que se possa analisar

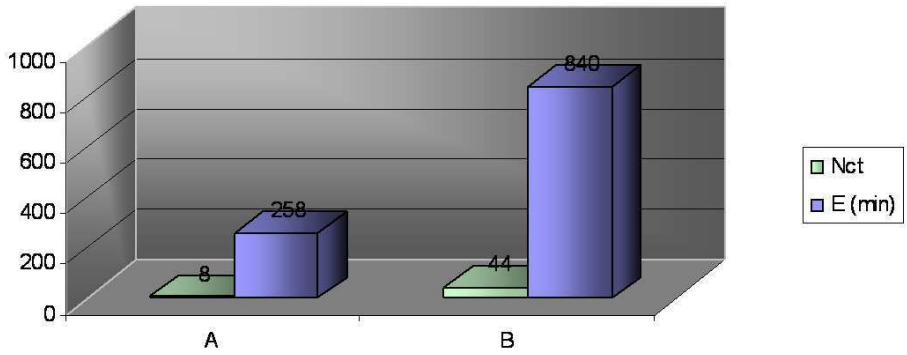


Figura 5.3: Gráfico comparativo - N_{ct} x E

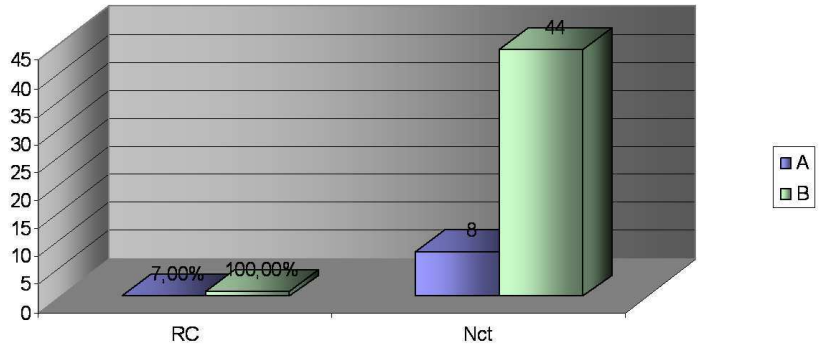


Figura 5.4: Gráfico comparativo - R_c x N_{ct}

corretamente as métricas calculadas e a partir delas saber o que precisa ser melhorado no processo.

Embora os casos de teste não tenham sido executados e nem todas as métricas definidas no modelo calculadas, o estudo aumentou o sentimento com relação à necessidade de experiência na escrita de casos de teste e de conhecimento sobre a aplicação para escrever cenários de uso que cobrem todos os requisitos testáveis. Estudos de caso precisam ser realizados para ajustar e aperfeiçoar o modelo de medição e para definir valores padrões para as métricas, como, por exemplo, o valor aceitável para o RNT.

5.2.3 Estudo de Caso 2

O estudo de caso apresentado nesta seção tem como principal objetivo o ajuste do modelo de medição definido na Seção 5.2. A seção apresenta ambiente e metodologia utilizados, e em seguida uma análise dos resultados obtidos.

Ambiente e Metodologia

Mais uma vez, o estudo foi realizado dentro do ambiente Motorola, sendo conduzido por um pesquisador com o auxílio de um *test designer* experiente na escrita de casos de teste. A Tabela 5.4 apresenta algumas características sobre o pesquisador e o *test designer*.

Tabela 5.4: Dados sobre participantes.

	Pesquisador	*Test Designer*
Experiência na escrita de casos de teste	$Não$	Sim
Conhecimento da *feature*	$Não$	$Não$
Conhecimento da ferramenta TaRGeT	Sim	$Não$

A metodologia utilizada foi a do processo de MBT apresentado no Capítulo 3. O *test designer* foi responsável pela escrita dos cenários de uso e coleta de dados, enquanto o pesquisador foi responsável por gerar os casos de teste e calcular e analisar as métricas obtidas.

No estudo, apenas uma *feature* foi utilizada. A *feature* possui 56 requisitos, descritos em dois documentos de requisitos. Dos 56 requisitos, 54 são classificados como testáveis, ou seja, é possível derivar casos de teste a partir deles. Dessa forma, o RNT da *feature*, considerando a versão utilizada dos documentos de requisitos, é

$RNT = (\frac{56-52}{56}) * 100\% \approx 7,15\%$.

O esforço E necessário para construção do modelo foi calculado apenas com o tempo de preparação T_p. O tempo de inspeção T_i não foi coletado, pois não se teve o controle total sobre o estudo de caso.

Foram escritos 27 cenários de uso, a partir dos quais foram gerados 45 casos de teste. O esforço E foi $E = 860$ minutos. Ao final, o esforço por caso de teste foi de ≈ 19 min por

caso de teste. Ao comparar com bases históricas, este número representa uma redução de cerca de 40% no esforço de escrita manual de casos de teste. Todos os requisitos testáveis foram cobertos, o que implica em:

$RC = (\frac{52}{52}) * 100\% = 100\%$.

A suite de testes gerada foi utilizada em dois ciclos de teste. No segundo ciclo, a suite foi incrementada com casos de teste escritos de forma manual. Isto deveu-se ao fato do estudo envolver geração e execução de casos de testes reais e do processo de testes utilizado na Motorola não permitir um controle total na condução do estudo de caso.

Para o primeiro ciclo de execução, foram selecionados 34 dos 45 casos de teste da suite. A Tabela 5.5 mostra alguns números obtidos. Dos 34 casos des teste selecionados para o ciclo, 22 foram executados, com 20 passando (*P*) e 2 indicando falhas (*F*), 7 foram bloqueados devido a alguma ação que o testador não soube executar. Finalmente, 5 casos de teste foram modificados e não executados. A modificação de um caso de teste pode ocorrer após uma inspeção ou durante a execução. Geralmente ocorre quando o *test designer* que escreveu o caso de teste, nesse caso o cenário de uso a partir do qual o caso de teste foi gerado, não conhece bem o comportamento da aplicação e faz algumas suposições, como por exemplo, o caminho necessário para se chegar a determinada tela do celular.

Tabela 5.5: Números do ciclo de execução 1

Número de casos de teste da suite	45
Número de casos de teste do ciclo	34
Número de casos de teste *P*	20
Número de casos de teste *F*	2
Número de casos de teste *B*	7
Número de casos de teste *M*	5

Para o primeiro ciclo de execução, o percentual de casos de teste modificados foi:

$CTM = (\frac{5}{34}) * 100\% \approx 14,7\%$,

enquanto o percentual de falha encontradas foi:

$CTF = (\frac{2}{34}) * 100\% \approx 5,8\%$.

A Tabela 5.6 apresenta as métricas obtidas para o primeiro ciclo de execução.

Tabela 5.6: Métricas calculadas para o primeiro ciclo de execução.

Métrica	Valor Obtido
E	860 minutos
RC	100%
RNT	$\approx 7,15\%$
CTM	$\approx 14,7\%$
CTF	$\approx 5,8\%$

O segundo ciclo de execução da *feature* envolveu os casos de teste que foram adicionados à suite de forma manual. Como não se teve controle sobre o percentual de requisitos cobertos e os casos de teste foram escritos ao invés de gerados de forma automática a partir dos cenários de uso da *feature*, os resultados de execução do ciclo não foram considerados neste estudo de caso.

Análise dos resultados

Embora não se tenha tido controle total sobre o estudo de caso, os resultados obtidos são de grande valia. A utilização do processo de MBT apresentado no Capítulo 3 reduziu em cerca de 40% o esforço referente a escrita de casos de teste. Esse é um ganho considerável.

Os percentuais de RNT, CTM, CTF e RC precisam de mais estudos para serem melhor analisadaos. Apenas com esse estudo não podemos afirmar que um percentual CTM de 15% é alto ou razoável, por exemplo.

A realização do estudo permitiu a validação do modelo de medição proposto, no que diz respeito à viabilidade de coleta dos dados. Verificou-se que todas as métricas definidas são facilmente calculadas a partir dos dados coletados. Criando-se uma cultura de avaliação, é possível estimar com mais precisão o esforço de execução, bem como detectar pontos falhos no processo de MBT utilizado.

5.3 Conclusões

Neste capítulo foi apresentado um modelo de medição para acompanhar o desempenho de MBT ao longo dos ciclos de teste. O modelo utiliza métricas que podem ser coletadas sem aumentar o custo do processo de teste. As métricas levam em consideração um ciclo de teste e podem ser armazenadas para:

- **Estimativa na escrita da próxima suite de testes** - Com um histórico é possível fazer uma estimativa de esforço, por exemplo.
- **Seleção dos casos de teste que serão executados no próximo ciclo** - Os casos de teste que falharam no ciclo n deverão ser inseridos no ciclo seguinte.
- **Detecção de problemas** - É possível identificar se existem problemas na especificação dos requisitos, na utilização da notação de escrita do modelo formal ou na qualidade da equipe de escrita de casos de teste.

A aplicação do modelo não está associada nem se restringe ao processo de MBT apresentado no Capítulo 3. Ele é aplicável a qualquer processo que permita a coleta dos dados necessários para o cálculo das métricas.

A interpretação das métricas é algo que precisa ser investigado, para que se saiba avaliar os valores obtidos, por exemplo, afirmar se uma $CTF = 5,8\%$ é aceitável ou não. A detecção de problemas pode ser facilitada com a interpretação das métricas. Se a CTM está alta, é um indício de que os requisitos estão incompletos ou que o *test designer* não está apto para escrita dos cenários de uso, precisando aprofundar o seu conhecimento sobre o comportamento da aplicação.

Capítulo 6

Considerações Finais

Neste trabalho de dissertação dois problemas principais foram considerados: a pequena adoção de MBT dentro da indústria de desenvolvimento de *software* e a integração de MBT no processo de teste de *feature* de aplicações para celular.

Como iniciativa de solução para esses problemas, modelos de medição usados para acompanhar o desempenho de MBT foram definidos. O uso de modelos de medição permite coletar dados e calcular métricas que podem ser úteis na comprovação da eficiência de MBT como estratégia de teste, ajudando no aumento da adoção da técnica dentro da indústria. Além disso, o acompanhamento de desempenho permite focar em pontos fracos do processo, aumentando a qualidade dos testes através do melhoramento do processo.

A definição dos modelos de medição exigiu um entendimento de processos genéricos de MBT e do processo de teste de *feature*. Esse entendimento permitiu a definição do processo de MBT apresentado no Capítulo 3. O processo se adequa ao processo de teste de *feature* considerado neste trabalho.

O Capítulo 4 apresenta o primeiro modelo de medição. O modelo foi definido no intuito de comparar o desempenho de duas estratégias de teste: MBT e teste exploratório. A principal motivação para este estudo foi a intuição de que estas estratégias são complementares e que teste exploratório pode ser bastante efetivo para o entendimento de requisitos implícitos. Métricas como esforço, taxa de defeitos detectados e relevância média dos defeitos detectados foram definidas seguindo a abordagem *Goal Question Metric*. Um estudo de caso foi realizado a fim de validar o modelo de medição proposto. Ao final, uma abordagem conjunta foi proposta para o teste de *feature*. A abordagem é composta por duas fases: a primeira

faz uso de teste exploratório como instrumento para aquisição de conhecimento sobre a aplicação; a segunda fase utiliza MBT para criação e manutenção de suites de teste.

O segundo modelo, apresentado no Capítulo 5, foi definido para acompanhar o desempenho de MBT ao longo dos ciclos de teste. A existência de um modelo deste tipo é fundamental para que se possa avaliar e acompanhar os ganhos e deficiências durante a implantação de processos de MBT, buscando melhorias focadas nos pontos fracos. O paradigma *Goal Question Metric* foi utilizado na definição de cinco métricas: esforço, percentual de cobertura de requisitos testáveis, percentual de requisitos não testáveis, percentual de casos de teste modificados e percentual de falhas encontradas. Dois estudos de casos foram realizados para ajustar as métricas definidas no modelo. Ao final, obteve-se um modelo de medição com métricas facilmente coletáveis e que podem contribuir bastante na melhoria contínua do processo, se coletadas e analisadas corretamente ao longo dos ciclos de execução.

A definição de um modelo de medição envolve muita observação. É preciso entender bem o objeto de estudo. Quando esse objeto é um processo, é preciso entender também o ambiente no qual o mesmo é aplicado, os processos que interferem, as regras associadas, enfim, entender todo o contexto. Após a definição, vem a fase de validação, onde é preciso encontrar um ambiente adequado para aplicação do modelo. Encontrar esse ambiente não é algo trivial. Durante o desenvolvimento deste trabalho tivemos muitas dificuldades para validar os modelos. Os estudos de caso realizados ficaram incompletos, pois não conseguimos controlar todo o processo, uma vez que nem todas as fases do processo de teste de *feature* são realizadas por um mesmo time. Apesar das dificuldades, os estudos permitiram uma validação inicial dos modelos, já que se conseguiu calcular todas as métricas definidas.

Aplicar MBT na prática não é algo trivial. Para que se obtenha sucesso, além de adotar processos de MBT é preciso se adaptar a situações nas quais outros processos interferem. Dentro do BTC, o uso de MBT ainda é pequeno. Processos precisam ser melhor definidos e adotados. O processo apresentado no Capítulo 3 é um exemplo de iniciativa para adoção de MBT dentro do processo de teste de *feature*. Embora os estudos de caso tenham sido poucos e incompletos, eles reforçaram o sentimento com relação ao ganho que MBT pode propiciar, em termos de redução de custos e melhoramento do processo de testes.

A adoção de MBT pela indústria requer não apenas uma ferramenta de geração de casos de teste. É necessário um ambiente que suporte a criação do modelo formal, permita a

criação dos casos de teste, facilite a manutenção das suites de teste e forneça ferramentas para acompanhar o desempenho de MBT. A ferramenta TaRGeT, utilizada neste trabalho, oferece um abiente para criação do modelo formal, através da definição de cenários de uso, e para geração dos casos de teste. Para contemplar um processo completo de MBT, a TaRGeT precisa:

- **Melhorar o ambiente utilizado na definição dos cenários de uso** - Um ambiente gráfico pode facilitar bastante a escrita dos cenários de uso. É bem mais interessante para o *test designer* visualizar os cenários e suas interseções de maneira visual do que acompanhar os fluxos através das tabelas. Um ambiente gráfico pode reduzir a quantidade de inconsistências que interseções entre vários cenários de uso podem vir a causar.

- **Facilitar a evolução das suites de teste** - Como dito anteriormente, o sucesso de MBT requer disciplina na atualização do modelo formal. Qualquer mudança necessária na suite de testes deve ser feita no modelo para, só então, ser refletida nos casos de teste, que devem ser gerados novamente. Atualmente, a TaRGeT não facilita essa evolução, pois não há controle de versão das suites.

- **Permitir o acompanhamento dos ciclos de execução e da evolução das suites de teste** - Armazenar informações sobre resultados de execução, versões de documentos de requisitos, casos de teste que compoem cada suite, casos de teste selecionados para cada ciclo de execução, são fundamentais para que se tenha controle sobre as atividades de geração e execução de testes. A TaRGeT precisa incorporar ou se integrar com ferramentas que ofereçam essas funcionalidades.

- **Gerar relatórios com informações sobre as suites e os ciclos de execução** - Se a ferramenta suporta o armazenamento de informações acerca das suites e ciclos de execução, relatórios com as métricas definidas no modelo de medição apresentado no Capítulo 5 podem ser gerados de forma automática.

6.1 Trabalhos Relacionados

Durante a realização deste trabalho, não foram encontrados trabalhos que avaliam estratégias de teste no contexto de teste de *feature* para aplicações para celular.

Pretschner [Pretschner et al., 2005] avalia a qualidade de testes gerados a partir de modelos com relação a testes gerados manualmente. É apresentado um estudo de caso que faz uso de um sistema de controle de rede automotiva. O foco do trabalho está na avaliação das suites geradas, considerando aspectos como de detecção de erros, cobertura do modelo e da implementação.

Sinha et. al [Sinha et al., 2006] apresenta um *framework* de medição para avaliar ferramentas de MBT. Assim como os modelos de medição apresentados nos Capítulos 4 e 5, o framework é definido com o auxílio do paradigma *Goal Question Metric*, focando em cinco aspectos:

- **Complexidade** - Está associada à dificuldade em usar a ferramenta devido ao número de conceitos que precisam ser aprendidos. Um conceito pode ser um operador, uma função ou um mecanismo de modelagem que precisa ser aprendido para se usar a ferramenta.

- **Facilidade de aprendizado** - Está associado ao tempo que o usuário gasta para obter um nível específico de conhecimento sobre a ferramenta.

- **Efetividade** - Está relacionada à efetividade da suite de testes gerada, considerando o número de requisitos cobertos.

- **Eficiência** - Está associada à facilidade de testar um sistema após a fase de aprendizado do usuário.

- **Escalabilidade** - Está associada à capacidade da ferramenta em possibilitar o gerenciamento do esforço de teste quando o tamanho da aplicação aumenta.

Os passos envolvidos na avaliação de ferramentas de MBT são descritos em um estudo de caso que envolve quatro ferramentas: Archetest, ASML, HOTTest e TestMaster. As ferramentas diferem nas técnicas de modelagem, nas técnicas de especificação de teste e nos algortimos de geração. Os aspectos abordados e as métricas que os definem não permitem

avaliar características interessantes para o processo de MBT, como por exemplo, o percentual de falhas. Além disso, aspectos como escalabilidade são melhor aplicáveis a ferramentas e não a processos. Modelos de medição precisam ser definidos para contextos específicos, pois precisam considerar características particulares de cada contexto.

Aranha [Aranha and Borba, 2008] utiliza simulação de processos na tentativa de avaliar o uso de MBT. São definidos e comparados dois processos: um processo automatizado, que utiliza a ferramenta TaRGeT [Nogueira et al., 2007], e um processo manual, no qual casos de teste são escritos com base em requisitos. O objetivo é avaliar a redução de esforço na escrita de casos de teste que MBT pode proporcionar.

O trabalho de Pretschner [Pretschner et al., 2005] foca na avaliação da qualidade das suites de teste geradas, enquanto o nosso busca uma da avaliação da estratégia de teste como um todo. O trabalho de Sinha et. al [Sinha et al., 2006] diferencia-se do nosso uma vez que foca na avaliação de ferramentas de MBT e não na avaliação do processo. O trabalho de Aranha [Aranha and Borba, 2008] busca uma avaliação de MBT usando a ferramenta TaRGeT, aproximando-se do nosso. Porém, foca no esforço de escrita de casos de teste, enquanto o nosso trabalho considera outros aspectos.

A experimentação e a avaliação de estratégias de teste é uma área ainda pouco explorada. Muito há para ser pesquisado e discutido. Esse trabalho é apenas uma pequena contribuição que se soma aos trabalhos aqui citados, dentro desta vasta área de pesquisa que é a avaliação de processos e estratégias de testes.

6.2 Trabalhos Futuros

O trabalho apresentado é apenas um embasamento inicial para que estudos experimentais sejam realizados na área de testes para sistemas reativos, em particular teste de *feature* para aplicações para celular. Como trabalhos futuros podem ser citados:

- **Realização de novos estudos de caso** - Os estudos de caso realizados durante o trabalho precisam ser repetidos e melhorados para que um histórico possa ser criado. Um histórico de dados pode auxiliar na estimação de atividades futuras, como estimativa de esforço de criação de cenários de uso, por exemplo.

- **Experimentação de outras abordagens de teste** - As abordagens de teste consideradas ao longo deste trabalho podem não ser as melhores para o teste de *feature* de aplicações para celular. É preciso avaliar outras abordagens e comparar os resultados.

- **Proposta de novos modelos de medição** - Os modelos de medição focam em determinadas métricas. Novos modelos com focos diferentes precisam ser propostos para que se avalie as estratégias levando em consideração outros aspectos, como facilidade de aprendizado, por exemplo.

- **Análise de valores das métricas** - A análise de métricas é crucial para se fazer qualquer avaliação. Os números precisam ter significado, por exemplo, o que significa um percentual de requisitos testáveis igual a 80%? Esse valor é aceitável? Observar o que acontece na prática e estabelecer intervalos númericos com significado é fundamental para que se saiba interpretar as métricas obtidas e dessa forma tirar conclusões ao final dos estudos de caso.

Bibliografia

[Aceto et al., 2007] Aceto, L., Ingólfsdóttir, A., Larsen, K. G., and Srba, J. (2007). *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press.

[Agruss and Johnson, 2000] Agruss, C. and Johnson, B. (2000). Ad hoc software testing: A perspective on exploration and improvisation. *http://www.testingcraft.com/ad_hoc_testing.pdf*. Acessado em 15 de Dezembro de 2007.

[Aranha and Borba, 2008] Aranha, E. and Borba, P. (2008). Using process simulation to assess the test design effort reduction of a model-based testing approach. In *ICSP*, pages 282–293.

[Bach, 2000] Bach, J. (2000). Session-based test management. *Software Testing Quality Engineering magazine*, vol. 2(no. 6).

[Bach, 2003] Bach, J. (2003). Exploratory testing explained. *http://www.satisfice.com/articles/et-article.pdf*. Acessado em 15 de Dezembro de 2007.

[Baker et al., 2002] Baker, P., Bristow, P., Jervis, C., King, D. J., and Mitchell, B. (2002). Automatic generation of conformance tests from message sequence charts. In *SAM*, pages 170–198.

[Barbosa et al., 2007] Barbosa, D. L., Lima, H. S., Machado, P. D. L., Figueiredo, J. C. A., Juca, M. A., and Andrade, W. L. (2007). Automating functional testing of components from uml specifications. *Int. Journal of Software Eng. and Knowledge Engineering*. To appear.

[Basili, 1992] Basili, V. R. (1992). Software modeling and measurement: the goal/question/metric paradigm. Technical report, College Park, MD, USA.

[Basili, 1996] Basili, V. R. (1996). The role of experimentation in software engineering: past, current, and future. In *ICSE '96: Proceedings of the 18th International Conference on Software engineering*, pages 442–449, Washington, DC, USA. IEEE Computer Society.

[Basili et al., 1995] Basili, V. R., Caldiera, G., and Rombach, H. D. (1995). The goal question metric approach. *Encyclopedia of Software Engineering*, 1:528–532.

[Bertolino, 2007] Bertolino, A. (2007). Software testing research: Achievements, challenges, dreams. In *FOSE '07: 2007 Future of Software Engineering*, pages 85–103, Washington, DC, USA. IEEE Computer Society.

[Boehm et al., 1976] Boehm, B. W., Brown, J. R., and Lipow, M. (1976). Quantitative evaluation of software quality. In *ICSE '76: Proceedings of the 2nd international conference on Software engineering*, pages 592–605, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Cabral and Sampaio, 2006] Cabral, G. and Sampaio, A. (2006). Formal specification generation from requirement documents. In *SBMF 2006: Proceedings of the Brazilian Symposium on Formal Methods*, pages 217–232.

[Cartaxo, 2006] Cartaxo, E. G. (2006). Geração de casos de teste funcional para aplicações de celulares. Master's thesis, COPIN - Universidade Federal de Campina Grande.

[Cem Kaner and Pettichord, 2002] Cem Kaner, J. B. and Pettichord, B. (2002). *Lessons Learned in Software Testing: A Context-Driven Approach*. John Wiley & Sons, Inc., New York.

[Dalal et al., 1999] Dalal, S. R., Jain, A., Karunanithi, N., Leaton, J. M., Lott, C. M., Patton, G. C., and Horowitz, B. M. (1999). Model-based testing in practice. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 285–294. IEEE Computer Society Press.

[de Carvalho Nogueira, 2006] de Carvalho Nogueira, S. (2006). Geração automática de casos de teste csp orientada por propósitos. Master's thesis, CIn - Universidade Federal de Pernambuco.

[de Lucena Andrade, 2007] de Lucena Andrade, W. (2007). Geração de casos de teste de interação para aplicações de celulares. Master's thesis, COPIN - Universidade Federal de Campina Grande.

[El-Far and Whittaker, 2001] El-Far, I. K. and Whittaker, J. A. (2001). Model-based software testing. *Encyclopedia on Software Engineering*.

[Emanuela G. Cartaxo, 2007] Emanuela G. Cartaxo, Francisco G. O. Neto, P. D. L. M. (2007). Test case selection using similarity function. In *MOTES07 - Model-based Testing - Workshop in conjunction with the 37th Annual Congress of the Gesellschaft fuer Informatik*, volume vol. 110, pages 381–386.

[Graham et al., 2007] Graham, D., van Veenendaal, E., Evans, I., and Black, R. (2007). *Foundations of Software Testing - ISTQB Certification*. Thomson Learning.

[Guilherme Horta Travassos, 2002] Guilherme Horta Travassos, Dmytro Gurov, E. A. G. d. A. (2002). Introdução à engenharia de software experimental. Technical report.

[Hartman, 2002] Hartman, A. (2002). Agedis - model based test generation tools. *http://www.agedis.de/documents/ModelBasedTestGenerationTools_cs.pdf*. Acessado em 23 de Janeiro de 2008.

[Hartman and Nagin, 2004] Hartman, A. and Nagin, K. (2004). The agedis tools for model based testing. In *ISSTA '04: Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis*, pages 129–132, New York, NY, USA. ACM Press.

[Itkonen and Rautiainen, 2005] Itkonen, J. and Rautiainen, K. (2005). Exploratory testing: A multiple case study. In *ISESE 2005: Proceedings of the International Symposium on Empirical Software Engineering*. IEEE Computer Society.

[Kogure and Akao, 1983] Kogure, M. and Akao, Y. (1983). Quality function deployment and cwqc in japan. In *Quality Progress*.

[Nogueira et al., 2007] Nogueira, S. C., Cartaxo, E. G., Torres, D. G., Aranha, E. H. S., and Marques, R. (2007). Model based test generation: An industrial experience. In *1st Brazilian Workshop on Systematic and Automated Software Testing*, Joao Pessoa, Brazil. SBES.

[Prenninger and Pretschner, 2005] Prenninger, W. and Pretschner, A. (2005). Abstractions for model-based testing. *Electronic Notes in Theoretical Computer Science*, vol. 116:59–71.

[Pretschner et al., 2005] Pretschner, A., Prenninger, W., Wagner, S., Kühnel, C., Baumgartner, M., Sostawa, B., Zölch, R., and Stauner, T. (2005). One evaluation of model-based testing and its automation. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 392–401, New York, NY, USA. ACM.

[Sinha et al., 2006] Sinha, A., Williams, C. E., and Santhanam, P. (2006). A measurement framework for evaluating model-based test generation tools. *IBM Syst. J.*, 45(3):501–514.

[Turner et al., 1998] Turner, C. R., Wolf, A. L., Fuggetta, A., and Lavazza, L. (1998). Feature engineering. In *IWSSD '98: Proceedings of the 9th international workshop on Software specification and design*, page 162. IEEE Computer Society.