Requisitos para Automação dos Testes de Sistemas de Informação

Pedro Santos Neto, Rodolfo Resende, Clarindo Pádua

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627, 31.270-010, Pampulha
Belo Horizonte - MG

{pasn, rodolfo, clarindo}@dcc.ufmg.br

Abstract. *This work presents two contributions. The first contribution is a set of requirements for testing automation methods, in the context of information systems. This set can be used as a guide for the development of new methods and tools. The second contribution is the evaluation of three test automation methods using the proposed requirements. This evaluation allows the identification of improvements in the analyzed methods.*

Resumo. *Este trabalho apresenta duas contribuições. A primeira contribuição é uma proposta de requisitos, no contexto de sistemas de informação, para métodos de automação de testes. Esses requisitos podem ser utilizados como guia para o desenvolvimento de novos métodos e ferramentas. A segunda contribuição é a avaliação de três métodos de automação de testes utilizando os requisitos propostos, permitindo a identificação de melhorias nos métodos analisados.*

1. Introdução

Os Sistemas de Informação (SIs) possuem um papel cada vez mais importante no dia-a-dia das pessoas. Tais sistemas são usualmente compostos por muitos elementos que operam juntos para recuperar, processar, armazenar e distribuir informação. Esta informação é utilizada para auxiliar a análise, controle e tomada de decisão em uma organização.

Considerando a atual relevância dos SIs, podemos notar que uma falha durante sua operação pode causar grandes danos. Uma das razões para isso é o elevado custo para testar softwares. Esse custo continua alto, sendo proibitivo em algumas organizações desenvolvedoras. Segundo o Ministério da Ciência e Tecnologia [1], somente 57% das empresas pesquisadas no *Programa Brasileiro de Qualidade e Produtividade de Software - PBQP* usam testes de aceitação como método de detecção/remoção (sic) de defeitos. Essa pesquisa também mostra que apenas 52% das empresas pesquisadas utilizam testes do sistema integrado. Nossa interpretação desses dados é que o alto custo do teste de software impede que as organizações o utilize de forma sistemática.

A baixa presença das atividades de teste no desenvolvimento de sistemas, conforme dados do PBQP, e os altos custos associados a falhas em sistemas [2] motivaram nossa investigação sobre como melhorar a eficiência das atividades de teste, em particular com relação às atividades diretamente relacionadas ao desenvolvimento de SIs.

Os SIs podem ser classificados em quatro categorias segundo o tipo de utilização: operacional, conhecimento, gerencial e estratégico. Cada uma destas categorias está associada a um nível da organização e possui um perfil de usuários bem definido [3]. Os

sistemas do nível operacional, amplamente conhecidos por causa dos Sistemas de Processamento de Transações - SPTs, dão suporte às transações existentes em uma organização. Esses sistemas suportam as operações do dia-a-dia de uma organização, mantendo os dados resultantes dessas operações. Eles servem de base para a construção dos outros tipos de SIs. Neste trabalho focalizamos os SPTs, uma vez que eles são peças chaves na construção de qualquer tipo de SI. Assim, quando utilizamos o termo SI estamos nos referindo a SIs no nível operacional.

Neste trabalho apresentamos duas contribuições. A primeira contribuição é uma proposta de requisitos para automação dos testes de SIs. Esses requisitos podem ser utilizados como guia para o desenvolvimento de novos métodos e ferramentas. A segunda contribuição é a avaliação de três Métodos de Automação de Testes - MATs, utilizando os requisitos propostos. Um desses métodos foi escolhido por ter sido desenvolvido pelos autores e os outros dois por serem bastante discutidos na literatura. Essa avaliação permite a identificação de possíveis melhorias nos métodos analisados.

A apresentação deste trabalho segue a seguinte organização: a Seção 2 apresenta nossa proposta de requisitos para MATs; a Seção 3 apresenta de forma breve alguns MATs aplicáveis a sistemas de informação, juntamente com uma breve avaliação desses métodos com relação aos requisitos de automação propostos; a Seção 4 conclui o trabalho, apresentando nossos planos para futuras investigações.

2. Requisitos para Automação do Teste de Sistemas de Informação

Para definição da proposta de requisitos para automação dos testes de SIs analisamos alguns métodos de desenvolvimento aplicáveis a tais sistemas. Particularmente, analisamos os trabalhos de Zachman e Sowa [4] e o PRAXIS [5], uma vez que esses dois trabalhos estão intimamente ligados ao desenvolvimento de sistemas de informação. Tais métodos possuem um núcleo de atividades compatíveis com relação ao desenvolvimento de um SI. A partir da descoberta dessas atividades, aliada à nossa experiência na área e a literatura disponível, foi possível notar que certos elementos são essenciais para qualquer iniciativa relacionada à automação de testes de SIs. Cada um desses elementos identificados se tornou um dos requisitos propostos neste trabalho.

Os requisitos propostos neste trabalho estão associados ao teste de sistema, cuja finalidade é exercitar um sistema por completo, incluindo a verificação de requisitos funcionais e não-funcionais [6]. A automação dos testes de requisitos não-funcionais é uma tarefa complexa. Por esse motivo decidimos não perseguir um espectro completo da automação desses testes.

2.1. Uso da Arquitetura do Sistema

Um MAT deve especificar a geração de testes que levem em consideração a arquitetura do sistema.

Os requisitos de um sistema de informação subsidiam a definição da sua arquitetura. No desenvolvimento, os sistemas são particionados em subsistemas que podem ser mais facilmente gerenciados e detalhados. A definição da arquitetura do sistema é feita estabelecendo-se interfaces e conexões entre os subsistemas.

A arquitetura de um sistema é um aspecto chave do desenvolvimento [7], tornando importante sua verificação. Diferentes projetos podem ter diferentes arquiteturas e, em função disto, são necessárias diferentes formas de verificação dessas arquiteturas. A verificação do sistema através da execução dos testes garante não só a qualidade do

sistema, mas também da sua arquitetura. Em particular, a arquitetura do sistema pode ser usada para a geração de testes de desempenho e estresse.

2.2. Uso da Descrição dos Dados Persistentes

Um MAT deve especificar a geração de testes que utilizem as descrições dos dados persistentes do SI.

Os SIs tipicamente utilizam dados persistentes. A maioria dos métodos de desenvolvimento de SIs prescreve a identificação e determinação dos relacionamentos entre esses dados, incluindo também a determinação dos domínios de entrada válidos. Essa informação é valiosa para a especificação de testes. A geração de testes deve fazer uso de todos os aspectos da descrição dos dados persistentes buscando garantir a corretude dos dados mantidos pelo sistema.

2.3. Uso da Descrição das Interfaces para Entrada de Dados

Um MAT deve especificar a geração de testes que levem em consideração a especificação das entradas de dados do sistema.

Os SPTs geralmente estão associados a um grande volume de entrada de dados. A entrada de dados pode ser feita de forma automática ou manual, através do uso de interfaces de usuário ou interfaces de software. Na maioria dos casos, os dados de entrada possuem relação com os dados persistentes, sendo de alguma forma manipulados antes de serem armazenados.

As entradas manuais são geralmente realizadas utilizando-se as interfaces de usuário. Essas interfaces podem conter diferentes formas de exibição, alteradas por vários mecanismos diferentes, como por exemplo, por causa do nível de acesso de um usuário ou por causa dos comandos já executados em tal interface. Um MAT deve especificar a geração de testes que verifiquem as diferentes formas de exibição de uma interface de usuário, verificando se todas as funções existentes estão disponíveis e funcionando adequadamente.

Além disso, o acesso a uma determinada interface de usuário pode exigir a navegação no sistema. Tal navegação pode exigir a geração de diversas entradas que forcem a exibição de novas interfaces de usuário. Um MAT deve especificar a geração de testes que exercite as possíveis transferências de controle entre interfaces de usuário, garantindo assim que todas as funções disponíveis no sistema possam ser executadas.

2.4. Uso de Critérios e Técnicas na Geração de Testes

Um MAT deve especificar o uso de um ou mais critérios de teste.

Os métodos para automação de teste devem utilizar as mais variadas informações relativas ao funcionamento do Sistema Sob Teste - SST. Independente das informações utilizadas, um ou mais critérios de teste devem ser seguidos. Um critério define quais propriedades de um programa devem ser exercitadas para constituir um teste completo [8]. Diferentes critérios podem ser utilizados, sejam critérios estruturais, baseados no código do sistema desenvolvido, sejam critérios funcionais, baseados na especificação do SST. As técnicas de teste também podem ser usadas para garantir melhores níveis de cobertura, tais como o particionamento em classes de equivalência e análise de valor-limite [6]. Métodos de automação devem definir de forma explicita os critérios e técnicas utilizados na geração dos testes para tentar garantir resultados consistentes.

2.5. Execução Automática dos Testes

Um MAT deve especificar os requisitos que permitem a execução automática de teste.

A execução automática de teste é um dos passos fundamentais na automação. Esse requisito é um dos mais atendido pelas ferramentas de teste comerciais. A execução de um teste geralmente é composta por diversos passos, como por exemplo:

- povoamento do mecanismo de armazenamento, com dados necessários ao teste;
- carga e instanciação dos elementos para entrada de dados;
- atribuição dos dados especificados nos testes nos respectivos elementos das interfaces de usuário;
- execução das ações associadas ao teste.

Um MAT deve especificar tudo o que for necessário para a execução automática dos testes. São exemplos disso a determinação de convenções que devem ser seguidas durante a implementação do sistema, desenvolvimento utilizando certos mecanismos (*patterns*) e uso de tecnologias específicas.

2.6. Geração de Oráculos para Análise dos Resultados dos Testes

Um MAT deve especificar os requisitos que permitem a existência de um oráculo para verificação dos resultados do teste.

O teste de software é geralmente composto por cinco atividades: planejamento, desenho, implementação, execução e análise dos resultados [5]. No mercado existem muitas ferramentas com suporte para implementação e execução de testes, e praticamente não existem ferramentas comerciais com suporte para o análise dos resultados. Um MAT deve ser capaz de interpretar os resultados de um teste, gerando um veredicto associado ao testes executado. Esse requisito é o mais difícil e caro de ser implementado por um MAT com um custo computacional viável [9].

2.7. Geração de Artefatos de Testes

Um MAT deve especificar a geração automática dos principais artefatos associados às atividades de teste.

Independentemente do processo de software utilizado durante o desenvolvimento, existem artefatos diretamente relacionados à atividade de teste [10]. Dentre esses artefatos destacamos:

- Plano de Teste. Documento que detalha o escopo, abordagem, recursos e agenda das atividades de teste. Além disso, identifica os ítens de teste, construções a serem testadas, tarefas a serem realizadas, pessoal para realização e riscos associados.
- Especificação de Casos de Teste. Documento especificando entradas, resultados esperados e um conjunto de condições de execução para um item de teste.
- Especificação dos Procedimentos de Teste. Documento contendo os passos para execução de um conjunto de casos de teste.
- Relatório de Incidentes. Documento relatando qualquer evento ocorrido durante o processo de teste que requer investigação.

Um MAT deve especificar a geração dos artefatos relacionados à atividade de teste, para facilitar a análise e interpretação dos testes gerados. Esses artefatos facilitam a análise da equipe de testes sobre a necessidade de criação de testes adicionais. Tais artefatos deveriam ser automaticamente extraídos dos diversos modelos que compõe o SST e os testes gerados para o SST.

3. Métodos para Automação de Testes de Sistemas de Informação

Existem diversos trabalhos relacionados à automação de testes. Nesta seção descrevemos três métodos para automação de testes, comparando-os sob a ótica dos requisitos propostos neste trabalho. Esses trabalhos foram selecionados por serem métodos para automação por completo, e não apenas uma visão particular sobre uma oportunidade para automação específica. Um desses métodos foi escolhido por ter sido desenvolvido pelos autores deste trabalho, e os outros dois por serem bastante discutidos na literatura.

3.1. TOTEM

Briand e Labiche desenvolveram um método para o teste funcional de sistemas denominado TOTEM (**T**esting **O**bject-orien**T**ed syst**E**ms with the unified **M**odeling language) [11]. O TOTEM foi o primeiro método de automação de teste baseado no uso das especificações do SST, sendo um dos trabalhos mais citados da área. Esse método deriva requisitos de teste a partir de artefatos criados durante a fase de análise de um processo de software baseado no Processo Unificado - PU [12]. A descrição do método não relata a existência de uma ferramenta de suporte, nem uma avaliação dos possíveis ganhos associados ao uso de tal técnica.

3.2. MODEST

O MODEST [13] é um método para automação dos testes de sistemas baseado nas especificações do SST. O MODEST prescreve o uso de modelos descrevendo sistema para a geração e execução automática de testes, podendo reduzir o esforço exigido durante a construção de software. O método possui uma ferramenta de suporte, chamada MODESTоо, que foi utilizada em um estudo experimental que mostrou benefícios em termos de esforço e de qualidade no uso do método no desenvolvimento de parte de um sistema de informação.

3.3. AGEDIS

O AGEDIS [14] é um método completo para automação de testes apoiado por ferramentas de suporte e apropriado para sistemas distribuídos. O método proposto pelo AGEDIS é baseado em um processo interativo composto por diversos passos. Durante estes passos os dados para executar os testes são criados, incluindo a construção de um modelo de funcionamento do sistema e uma descrição das características do teste a ser realizado, indicando a cobertura desejada, interfaces entre o modelo e o software e pontos de observação. Diversas ferramentas foram desenvolvidas e alguns estudos de casos foram realizados mostrando a aplicabilidade do método em casos reais.

3.4. Análise dos Métodos

Nesta seção analisamos os métodos apresentados sob a ótica de cada um dos requisitos para automação de testes descritos na seção anterior. Utilizamos as descrições dos métodos, publicadas em artigos e relatórios técnicos, para realizar tal análise.

3.4.1. Uso da Arquitetura do Sistema

- O TOTEM não explicita como deve ser a arquitetura dos sistemas aos quais ele pode ser aplicado. Isso nos leva a crer que ele possa ser genérico com relação à arquitetura de software do SST, no entanto, não existem atividades no método relacionadas à verificação da arquitetura.

- O MODEST prevê uma atividade relacionada ao uso da arquitetura do SST para a geração de testes, mas atualmente se restringe a uma arquitetura pré-estabelecida. Com base nessa arquitetura alguns testes específicos são criados, como por exemplo, teste de desempenho e estresse.
- O AGEDIS prevê a especificação de diretivas para execução dos testes (*Test Execution Directives*) que contém informação sobre a arquitetura usada para execução dos testes. Isso inclui a informação sobre a configuração dos recursos para execução em sistemas distribuídos. Embora não seja diretamente o uso da descrição da arquitetura do sistema, esse mecanismo permite configurar parâmetros a serem verificados durante a realização dos testes para um conjunto limitado de arquiteturas.

3.4.2. Uso da Descrição dos Dados Persistentes

- O TOTEM prescreve o uso das descrições dos dados persistentes para a geração de testes. Essas descrições são apresentadas em diagramas de classe que mostram os atributos das entidades envolvidas, juntamente com a definição dos relacionamentos entre as entidades e as classes que as utilizam.
- O MODEST prescreve o uso das descrições dos dados persistentes para a geração de testes de forma semelhante ao TOTEM, no entanto, o método prescreve a descrição não só das classes persistentes e seus relacionamentos, mas também prescreve o detalhamento de diversas outras informações, tais como tamanho máximo e mínimo dos dados suportados, tipo de valores permitidos e obrigatoriedade dos dados.
- O AGEDIS prescreve a criação de um modelo comportamental para descrever o SST. Esse modelo é implementado como um perfil UML [15], descrevendo o sistema através de diagramas de classes. O comportamento de cada classe é descrito em diagramas de estado integrados com uma linguagem de ações. As classes persistentes devem ser descritas nesse modelo, que é usado para a geração de testes.

3.4.3. Uso da Descrição das Interfaces para Entrada de Dados

- O TOTEM prescreve as descrições das pós-condições dos métodos das classes integrantes do sistema. Essas pós-condições são utilizadas durante a geração de testes. Isso inclui as pós-condições dos métodos das classes representando as interfaces de usuário. No entanto, ações dos usuários não são simuladas, não existe a prescrição da verificação das diferentes formas de exibição das interfaces de usuário nem a verificação das possíveis transferências de controle entre tais interfaces.
- O MODEST possui três atividades relacionadas a descrição das interfaces para entrada de dados. Em uma atividade denominada *"Detalhamento das Classes de Fronteira"* é prescrita a descrição das interfaces de usuário, detalhando os campos e comandos visíveis aos usuários finais. Na atividade denominada *"Descrição da Navegação"* é prescrita a especificação das transferências de controle entre as interfaces de usuário. Na atividade *"Descrição do Comportamento das Classes de Fronteira"* é prescrita a apresentação das diferentes formas de exibição das interfaces de usuário. Todas essas informações são utilizadas na geração dos testes, na tentativa de simular o uso do sistema por parte dos seus usuários.
- O AGEDIS prescreve a descrição das interfaces de usuários em um modelo comportamental do sistema. Além disso, nas diretivas para execução dos testes devem

ser prescritos os aspectos necessários para que os testes possam ser executados. Embora não tenham sido encontrados maiores detalhes sobre o funcionamento desse mecanismo, acreditamos que ele seja usado na geração de testes.

3.4.4. Uso de Critérios e Técnicas na Geração de Testes

- O TOTEM não explicita os critérios utilizados para geração automática dos testes. As mensagens contidas nos diagramas de seqüencia do sistema são utilizadas nessa geração, mas não são detalhadas as regras usadas durante tal tarefa.
- O MODEST usa diversos critérios para a geração de testes. Testes gerados com base nos diagramas de classes usam, basicamente, os critérios definidos por Andrews et al. [16]. O uso dos diagramas de estado que descrevem as interfaces de usuários é guiado pelo critério *Full Predicate Coverage*, definido por Offutt et al. [17]. Os testes criados a partir dos diagramas de interação seguem os critérios também definidos por Andrews et al. Existem ainda três outros critérios definidos no próprio método: critério de desempenho, estresse e navegação. Além disso, são utilizadas as técnicas de teste de particionamento de equivalência e análise de valor-limite.
- No AGEDIS as diretivas para geração de testes indicam os critérios a serem utilizados. Os critérios são aplicados aos diagramas de estado, diagramas de seqüencia e na forma de parâmetros passados ao gerador de testes. Existem cinco diretivas de geração de testes associadas a critérios: aleatória, cobertura de estados, cobertura de transições, cobertura de interface e cobertura de interface com parâmetros.

3.4.5. Execução Automática dos Testes

- Apesar do TOTEM mencionar aspectos relacionados à execução automática dos testes, não foi construída uma ferramenta que valide os aspectos mencionados, mostrando a possibilidade da execução automática de testes.
- O MODEST explicita os requisitos que permitem à MODESToo conter um componente responsável pela execução automática de testes denominado *Executor*.
- O AGEDIS explicita os requisitos que permitem à execução automática dos testes. Ele possui uma máquina de execução que usa a suite de testes abstratos e as diretivas de execução dos testes durante essa tarefa.

3.4.6. Geração de Oráculos para Análise dos Resultados dos Testes

- Apesar do TOTEM mencionar aspectos relacionados à geração automática de um oráculo, não foi construída uma ferramenta que valide os aspectos mencionados mostrando a viabilidade do uso dessa informação na prática.
- Os requisitos que permitem a implementação de um oráculo estão presentes no MODEST, no entanto, esses requisitos não se encontram consolidados, tornando difícil sua compilação. A viabilidade do uso dos requisitos está demonstrada com a MODESToo.
- Os requisitos que permitem a implementação de um oráculo estão presentes no AGEDIS. A máquina de execução do AGEDIS contém um oráculo gerado a partir da composição do modelo comportamental do sistema. A cada teste realizado o estado do sistema é analisado para verificar se o modelo confere com os resultados obtidos através da execução dos testes.

Método	Arquitetura	Entidades	Entrada	Critério	Execução	Oráculo	Artefatos
TOTEM	↓	↑	↓	↓	↓	↘	↓
MODEST	↗	↑	↑	↑	↑	↑	↑
AGEDIS	↗	↑	↗	↗	↑	↑	↘

Tabela 1: Comparativo dos métodos de automação de testes para SIs.

3.4.7. Geração de Artefatos de Teste

- O TOTEM não especifica a geração automática de artefatos de teste.
- O MODEST especifica a geração de um plano de teste parcial. O resultado da execução de um teste também gera um relatório de execução semelhante ao prescrito pelo IEEE [10].
- O AGEDIS não especifica a geração de artefatos de teste seguindo as especificações IEEE. No entanto, a máquina de execução do AGEDIS armazena os resultados de uma execução para posterior análise utilizando um componente denominado "*Visualizador*"que permite a análise dos testes executados e dos resultados obtidos na execução, semelhante ao que é prescrito pelo IEEE no documento denominado Relatório de Incidentes. Não existe menção a outros elementos que gerem os artefatos com as informações mais importantes sobre os testes, conforme descrito na Subseção 2.7.

3.5. Análise Comparativa dos Métodos

A Tabela 1 apresenta um resumo da análise do atendimento dos requisitos para automação dos testes de sistemas de informação por parte dos métodos analisados. Utilizamos a seguinte convenção na tabela: requisitos não atendidos são representados com uma seta para baixo (↓); requisitos mencionados no método, porém de forma muito superficial e de difícil implementação são representados com uma seta inclinada para baixo (↘); requisitos parcialmente atendidos pelo método são representados com uma seta inclinada para cima (↗); requisitos totalmente atendidos pelo método são representados com uma seta para cima (↑).

O método TOTEM atende a poucos dos requisitos propostos neste trabalho. Muitas situação relativas à automação dos testes de sistemas de informação somente são identificadas durante a implementação de uma ferramenta, inexistente no trabalho. A geração de oráculos, por exemplo, é mencionada no TOTEM, porém, isso é feito de forma pouco aprofundada e de difícil implementação.

O MODEST atende a todos os requisitos para automação de testes, no entanto o método é restrito em termos de arquitetura. Indicamos isso na Tabela 1 através do uso de uma seta inclinada para cima, indicando que o requisito é atendido parcialmente.

O AGEDIS atende à maioria dos requisitos propostos. O uso da arquitetura está limitado à configuração de alguns parâmetros para a execução de testes envolvendo sistemas distribuídos. O método não simula fielmente a entrada de dados no SST. No AGEDIS não existem critérios associados, por exemplo, à cobertura de relacionamentos entre classes e navegação no sistema. O método não prescreve a geração de certos artefatos relacionados aos testes executados que poderiam auxiliar bastante uma equipe de teste.

4. Conclusão

Neste trabalho apresentamos uma proposta de requisitos para Métodos de Automação de Teste - MATs relacionados a sistemas de informação. Acreditamos que o atendimento a esses requisitos seja de fundamental importância, visto que eles foram obtidos através da análise de alguns métodos de desenvolvimento de SIs, aliados à experiência dos autores

na área. Essa proposta de requisitos pode ser utilizada como guia para desenvolvimento e melhoria de métodos e ferramentas de testes.

Analisamos três métodos de automação de testes aplicáveis a sistemas de informação para verificar o nível de atendimento aos requisitos propostos. Foi possível notar que os métodos que possuem ferramentas de suporte, e que conseqüentemente podem ser avaliados através de estudos de casos, atendem de forma mais efetiva a tais requisitos, conforme dados apresentados na Subseção 3.5, resumidos na Tabela 1. O uso de tais requisitos para avaliação de MATs pode indicar o nível de benefício associado à adoção de um método por parte de uma organização.

A identificação de requisitos relacionados à automação de testes não é uma tarefa simples. Por esse motivo nos ativemos aos SPTs neste trabalho. Estamos atualmente fazendo uma revisão dos requisitos propostos, utilizando como insumo o conhecimento da comunidade de desenvolvedores de software e a literatura disponível. Paralelamente, estamos avaliando outros métodos de automação de testes utilizando os requisitos propostos e também estamos trabalhando na generalização desses requisitos para sistemas mais complexos que os SPTs.

Referências

[1] Ministério da Ciência e Tecnologia (MCT), *Programa Brasileiro da Qualidade e Produtividade de Software*, 3^a Edição, setembro de 2004.

[2] NIST, Planning Report 02-3, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, 2002, http://www.nist.gov/director/prog-ofc/report02-3.pdf, último acesso em novembro de 2004.

[3] Cook, M., *Building Enterprise Information Architecture: Reengineering Information Systems*, Prentice Hall, 1^a edição, 1996.

[4] Zachman, J., e Sowa, J., Extending and Formalizing the Framework for Information Systems Architecture, IBM Systems Journal, volume 31, número 3, 1992, páginas 590-616.

[5] Paula Filho, W., *Engenharia de Software: Fundamentos, Métodos e Padrões*, Editora LTC, 2^a edição, 2003.

[6] Jorgensen, P., Software Testing: A Craftsman's Approach, Second Edition, CRC Press, 2004.

[7] Kruchten, P., Architectural Blueprints - The "4+1" View Model of Software Architecture, *IEEE Software*, volume 12, número 6, páginas 42-50, novembro de 1995.

[8] Goodenough, J., e Gerhart, S., Towards a Theory of Test Data Selection, IEEE Transactions on Software Engineering, volume 1, número 2, páginas 156-173, junho de 1975.

[9] Peters, D., e Parnas, D., Generating a Test Oracle from Program Documentation: Work in Progress, Proceedings of the International Symposium on Software Testing and Analysis (ISSTA'94), páginas 58-65, Seattle, Washington, EUA, 1994.

[10] IEEE, IEEE Standard for Software Test Documentation - IEEE Std 829-1998, IEEE Computer Society, 1998.

[11] Briand, L., and Labiche, Y., A UML-Based Approach to System Testing, *Proceedings of the 4th Unified Modeling Language Conference (UML'01)*, páginas 194-208, Toronto, Canadá, outubro de 2001.

[12] Jacobson, I., Rumbaugh, J., e Booch, G., *The Unified Software Development Process*, Addison Wesley, 1999.

[13] Santos Neto, P., Resende, R., e Pádua, C., A Method For Information Systems Testing Automation, *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05)*, Porto, Portugal, junho de 2005.

[14] Hartman, A., e Nagin, K., The AGEDIS Tools for Model Based Testing, *International Symposium on Software Testing and Analysis (ISSTA 2004)*, Boston, Massachusetts, EUA, julho de 2004.

[15] Rumbaugh, J., Jacobson, I., e Booch, G., *The Unified Modeling Language Reference Manual*, Addison Wesley, 1999.

[16] Andrews, A., France, R., Ghosh, S., e Craig, G., Test Adequacy Criteria for UML Design Models, *Jornal of Software Testing, Verification, and Reliability*, volume 13, número 2, páginas 95-127, junho de 2003.

[17] Offutt, J., e Abdurazik, A., Generating Tests from UML Specifications, *Proceedings of the 2nd Unified Modeling Language Conference (UML'99)*, páginas 416-429, Fort Collins, CO, USA, outubro de 1999.