



Andrew Diniz da Costa

Automação do Processo de Gerência do Teste de Software

Tese de Doutorado

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio.

Orientador: Prof. Carlos José Pereira de Lucena

Rio de Janeiro
Agosto de 2012

Andrew Diniz da Costa

Automação do Processo de Gerência do Teste de Software

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Carlos José Pereira de Lucena

Orientador

Departamento de Informática – PUC-Rio

Prof. Alessandro Garcia

Departamento de Informática – PUC-Rio

Prof. Arndt Von Staa

Departamento de Informática – PUC-Rio

Prof. Viviane Torres da Silva

Universidade Federal Fluminense – UFF

Prof. Roberta de Souza Coelho

Universidade Federal do Rio Grande do Norte – UFRN

Prof. José Eugenio Leal

Coordenador Setorial do Centro

Técnico Científico – PUC-Rio

Rio de Janeiro, 10 de agosto de 2012

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

Andrew Diniz da Costa

Possui formação técnica no Instituto Brasileiro de Pesquisa em Informática (IBPI). Graduou-se em Bacharelado em Informática na PUC-Rio em 2006. Recebeu o título de Mestre em Informática na PUC-Rio em 2008. É IBM Certified Solution Designer -Rational Functional Tester for Java e possui o título de "Certified Tester Foundation Level - CTFL" pela "Brazilian Software Testing Qualifications Board - BSTQB". Desde 2003 trabalha no Laboratório de Engenharia de Software (LES) tanto em projetos de pesquisa como em projetos industriais. Atualmente é líder de três equipes de teste no LES compostas por doze pessoas. Possui mais de 10 anos de experiência em desenvolvimento de software a partir de diferentes linguagens de programação.

Ficha Catalográfica

Costa, Andrew Diniz da

Automação do processo de Gerência do Teste de Software / Andrew Diniz da Costa ; orientador: Carlos José Pereira de Lucena. - 2012.

171 f. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2012.

Inclui bibliografia.

1. Informática – Tese. 2. Coordenação de Testes de Software.. 3. Agentes de Software Autoadaptativos . 4. Autoteste. 5. Modelagem de Teste. I. Lucena, Carlos José Pereira de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática.

CDD: 004

A Deus, a minha mãe e ao meu pai que me ensinaram a acreditar que tudo é possível.

Agradecimentos

A minha mãe Nazaré e ao meu pai Wagner que são meus ídolos. Obrigado pelo exemplo de pais, por todo esforço, carinho, confiança, investimento, paciência e amor concedido durante toda minha vida. Obrigado pelas conversas e apoio para fazer o doutorado. Vocês tiveram uma grande parcela de importância nessas conquistas obtidas ao passar da minha vida. Serei eternamente grato. Amo muito vocês!! =)

A toda minha família do Pará e do Amapá que mesmo afastados sempre torceram por mim.

Ao professor Carlos José Pereira de Lucena pelas orientações, exemplo de profissional, oportunidades e confiança depositada durante minha graduação, mestrado e doutorado.

A professora Viviane Torres da Silva pelas enormes contribuições durante todos esses anos de trabalho em conjunto.

Aos professores Alessandro Garcia, Arndt Von Staa, e Roberta Coelho que tiveram contribuições importantes para a tese de doutorado.

A Paola, pelas conversas, apoio, momentos de descontração e amizade oferecidos durante esses quatro anos de doutorado.

A todos os amigos do LES, em especial ao grupo de testes de software que trabalhou dia a dia comigo em diferentes projetos, transformando os dias mais divertidos e descontraídos.

Ao CNPq pelo apoio financeiro da bolsa de doutorado.

E principalmente a Deus por tudo.

Muito obrigado!!

Resumo

Costa, Andrew Diniz da; Lucena, Carlos José Pereira de. **Automação do Processo de Gerência do Teste de Software**. Rio de Janeiro, 2012. 171p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Teste de software é uma atividade crítica no processo de desenvolvimento de sistemas, principalmente devido ao aumento da complexidade das aplicações atuais e pelo impacto que tais aplicações podem gerar. Relacionado a essa complexidade, o paradigma de sistemas multiagentes tem sido utilizado especialmente quando entidades pró-ativas, autônomas, autoadaptáveis e distribuídas precisam ser desenvolvidas. Para acompanhar a criação, manutenção e execução dos testes nesses sistemas, torna-se necessária a adoção de um processo de gerência, pois prevê a realização das atividades de planejamento, projeto, execução e acompanhamento dos testes. Visando ajudar nessa gerência, a tese apresenta o Java self-Adaptive Agent Framework for Self-Test (JAAF+T), framework que permite a criação de agentes autoadaptativos capazes de realizar autoteste, isto é, coordenar a execução dos testes necessários para validar suas autoadaptações. Como diversas informações são usadas para ajudar na gerência desses testes, documentá-las ajuda a entender como evoluir e executá-los. Baseada nessa preocupação, a tese oferece uma nova linguagem de modelagem chamada de UML Testing Profile for Coordination (UTP-C), perfil (*profile*) da UML que permite a modelagem dessas informações. Por fim, para automatizar o processo de gerência dos testes executados por agentes autoadaptativos, a tese apresenta ferramentas capazes de gerar de forma automática artefatos usados pelo JAAF+T baseados em modelos UTP-C.

Palavras-chave

Coordenação de testes de software; agentes de software autoadaptativos; autoteste; modelagem de teste.

Abstract

Costa, Andrew Diniz da; Lucena, Carlos José Pereira de. **Automation of the Management Process of the Test of Software**. Rio de Janeiro, 2012. 171p. DSc Thesis – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Testing software systems has become a critical activity of software development over time. Especially when the development of complex systems, which are able to self-adapt their behaviors when necessary, is becoming extremely common. In this context, the multi-agent system (MAS) paradigm has been used especially when distributed, autonomous and pro-active entities are represented. Aiming to control the creation, maintenance and execution of tests on these systems, it is necessary to adopt a management process that considers the activities of planning, design, execution and monitoring of the tests. In order to help this management, the thesis presents the Java self-Adaptive Agent Framework for Self-Test (JAAF+T), that aims to allow the creation of self-adaptive agents that perform self-tests. Since several data are used to help the management of the tests, documenting them helps to understand how to evolve and execute them. Based on this concern, the thesis offers a new modeling language called UML Testing Profile for Coordination (UTP-C), profile of the UML that allows the modeling of these data. At last, but not least important, tools, which were created to automate the management of the tests executed for self-adaptive agents, are presented to generate useful artifacts used for instances of the JAAF+T based on UTP-C models.

Keywords

Coordination of software testing; self-adaptive agents of software; self-testing; test modeling.

Sumário

1 Introdução	16
1.1. Motivação	17
1.2. O Problema	19
1.3. Principais Contribuições	19
1.4. Organização do Documento	20
2 Fundamentação Teórica	21
2.1. Conceitos Gerais da Área de Testes de Software	21
2.2. UML Testing Profile (UTP)	24
2.3. Sistemas Multiagentes	26
2.4. Sistemas Autoadaptativos que Realizam Autoteste	28
2.5. Framework JADE	29
3 Informações para Coordenação da Execução de Testes	32
3.1. Procedimentos Empíricos	33
3.2. Informações Identificadas	34
3.3. Considerações Finais	37
4 JAAF+T: Framework de Autoteste para Agentes de Software	38
4.1. Visão Geral do JAAF+T	39
4.2. Arquivos XML	43
4.3. Pontos Fixos e Flexíveis	51
4.4. Como Usar o Framework JAAF+T	53
4.5. Considerações Finais	54
5 Modelo Conceitual de Teste	56
5.1. Caso de Teste	57
5.2. Teste	58
5.3. Artefato em Teste	59
5.4. Suite de Teste	59
5.5. Classificação de Teste	60
5.6. Pacote de Desenvolvimento	61
5.7. Sistema em Teste	61
5.8. Critério de Seleção de Testes	61

5.9. Semântica de Dependências de Testes	62
5.10. Possíveis Perguntas para o Modelo Conceitual	62
5.11. Considerações Finais	63
6 Linguagem de Modelagem UTP-C	64
6.1. Os Mecanismos de Extensão da UML e UTP	64
6.2. Metamodelo da UML Testing Profile for Coordination (UTP-C)	65
6.2.1. Suite de Teste	68
6.2.2. Contexto de Teste (TestContext)	69
6.2.3. Caso de Teste	71
6.2.4. Critério de Seleção de Teste	75
6.2.5. Classificação de Teste	77
6.2.6. Pacote de Desenvolvimento	79
6.2.7. Artefato em Teste	80
6.2.8. Estereótipos para Relacionamentos de Dependência	82
6.3. Mapeamento do Modelo Conceitual para a Linguagem UTP-C	86
6.4. Avaliações Empíricas para a UTP-C	88
6.4.1. Avaliação para Manutenção de Modelos	89
6.4.2. Avaliação para Criação de Modelos	94
6.4.3. Resultados das Avaliações Aplicadas	96
6.5. Considerações Finais	100
7 Estudo de Caso Usando UTP-C com JAAF+T	101
7.1. Abordagem Adotada	101
7.1.1. Diagramas Estáticos UTP-C	101
7.1.2. Diagramas Dinâmicos UTP-C	102
7.1.3. Processo de Adaptação no JAAF+T	103
7.1.4. Testes no JAAF+T	103
7.1.5. Dados de Entrada e Saída para Testes no JAAF+T	103
7.1.6. Fluxos de Execução dos Testes no JAAF+T	104
7.1.7. Critérios de Seleção dos Testes no JAAF+T	104
7.2. O Exemplo do Mercado Virtual	105
7.2.1. Modelagem a partir da UTP-C	107
7.2.2. Autoadaptação Realizada pelo Agente Comprador	112
7.3. Outros Exemplos Usando UTP-C e JAAF+T	119
7.4. Discussão	121
7.5. Considerações Finais	124

8 Ferramentas para Aplicar Testes Baseados em Modelos	125
8.1. Mapeamento entre Arquivos XML do JAAF+T e Modelos UTP-C	125
8.2. Componentes LUA	128
8.3. Plug-in para o RSA	130
8.3.1. Visão Geral da Arquitetura do Plug-in	131
8.3.2. Geração de Relatórios e Comentários Javadoc	133
8.3.3. Geração dos arquivos TF.xml e CFF.xml	138
8.3.4. Avaliação da Ferramenta de Teste	139
8.4. Discussão	144
8.5. Considerações Finais	145
9 Trabalhos Relacionados	146
9.1. Trabalhos Voltados a Autoadaptação e Autoteste	146
9.2. Linguagens e Ferramentas de Gerenciamento de Teste	147
10 Conclusões e Trabalhos Futuros	151
10.1. Principais Vantagens das Abordagens Propostas	151
10.2. Principais Limitações das Abordagens Propostas	152
10.3. Trabalhos Futuros	153
Referências	155
Apêndice A. Exemplo de Questionário Aplicado para Avaliação de Manutenção de Modelos baseados na UTP-C	164
Apêndice B. Modelos UTP-C do Estudo de Caso Mercado Virtual Criados no RSA	168
Apêndice C. XMLs gerados pelo Plug-in para o RSA Referente ao Domínio Mercado Virtual	170

Lista de Figuras

Figura 1. Processo de autoadaptação proposto pela IBM	29
Figura 2: Plataformas de agentes do JADE	30
Figura 3. Diagrama de classes do JADE	31
Figura 4. Procedimentos empíricos para identificar informações para coordenação de teste a partir da execução de agentes de software.	33
Figura 5. Control-loop provido pelo JAAF+T.	41
Figura 6. Diagrama de classe do JAAF+T.	42
Figura 7. Esquema do arquivo TF.xml.	45
Figura 8. Esquema do arquivo DF.xml.	46
Figura 9. Esquema do arquivo CFF.xml.	48
Figura 10. Esquema do arquivo CEF.xml.	49
Figura 11. Modelo detalhando relacionamentos entre informações de teste.	57
Figura 12. Metamodelo da UML com estereótipos da UTP-C.	66
Figura 13. Estereótipos que expressam semântica de dependências.	67
Figura 14. OrderedSuite na UTP-C.	68
Figura 15. Exemplo de modelagem usando <<OrderedSuite>>.	69
Figura 16. TestContext na UTP-C.	69
Figura 17. Exemplo de Test Context.	70
Figura 18. Exemplo de diagrama de atividades com test contexts.	71
Figura 19. Caso de Teste na UTP-C.	71
Figura 20. Exemplo de um caso de teste de unidade modelado.	73
Figura 21. Exemplo de casos de teste de aceitação modelados.	73
Figura 22. Exemplo de diagrama de atividades com casos de teste.	74
Figura 23. Critério de seleção na UTP-C.	75
Figura 24. Exemplo de modelagem de um critério de seleção da UTP-C.	76
Figura 25. Classificação de Teste na UTP-C.	77
Figura 26. Modelagem de classificações de teste.	78
Figura 27. Pacote de Desenvolvimento na UTP-C.	79
Figura 28. Modelagem usando o estereótipo <<Development>>.	79
Figura 29. Estereótipo relacionado a artefatos de um sistema.	80
Figura 30. Modelando artefatos usados em sistemas.	82
Figura 31. Semântica de dependências incluídas.	83

Figura 32. Dependência entre testes de criação de artefato.	83
Figura 33. Dependência entre testes informando o caso de teste que depende.	83
Figura 34. Procedimentos empíricos adotados na avaliação de manutenção.	94
Figura 35. Resultados da avaliação de manutenção de modelos.	97
Figura 36. Resultados da avaliação de criação de modelos.	99
Figura 37. Visão geral da reputação de testemunho.	106
Figura 39. Modelagem de classificação de teste no mercado virtual.	109
Figura 40. Critérios do mercado virtual, modelados a partir da UTP-C.	110
Figura 41. Suite do mercado virtual modelada em um diagrama de classes.	111
Figura 42. Diagrama de atividades do suíte do sistema mercado virtual.	112
Figura 43. Transformação realizada pelo componente RSATF.lua.	129
Figura 44. Transformação realizada pelo componente RSACFF.lua.	129
Figura 45. Modelo conceitual do plug-in para o RSA.	131
Figura 46. Diagrama de classes do novo plug-in para o RSA.	132
Figura 47. Diagrama de classes e atividades da UTP-C modeladas no RSA.	133
Figura 48. Atributos adicionais de um caso de teste modelado.	134
Figura 49. Relatório gerado a partir do plug-in.	135
Figura 50. Tela para geração de artefatos de teste.	136
Figura 51. Diagrama de classe UTP-C modelado no RSA para o sistema mercado virtual	168
Figura 52. Diagrama de atividade UTP-C modelado no RSA para o sistema mercado virtual.	169

Lista de Tabelas

Tabela 1. Mapeamentos de informações úteis para coordenação da execução dos testes nos arquivos XML do JAAF+T.	51
Tabela 2. Mapeamento do modelo conceitual para UTP-C.	87
Tabela 3. Perfil dos participantes da avaliação de manutenção.	89
Tabela 4. Informação sobre o sistema de estoque e suprimento de petróleo.	90
Tabela 5. Informação sobre o sistema alocação de petróleo.	91
Tabela 6. Detalhamento dos questionários aplicados.	92
Tabela 7. Perfil das pessoas que participaram da avaliação de criação de modelos.	95
Tabela 8. Tipos de erros identificados nas respostas informadas pelos participantes nas avaliações aplicadas.	98
Tabela 9. Mapeamento para transformação de diagramas estruturais da UTP-C para o arquivo TF.xml.	127
Tabela 10. Mapeamento para transformação de diagramas dinâmicos da UTP-C para o arquivo CFF.xml.	128
Tabela 11. Análise de quais trabalhos relacionados estão considerando informações de teste manipuladas pela UTP-C.	150

Lista de Abreviaturas e Siglas

ISTQB	International Software Testing Qualifications Board
JAAF+T	Java self-Adaptive Agent Framework for self-Test
MBT	Model Based Test
MDD	Model Driven Development
OMG	Object Management Group
RQM	Rational Quality Manager
RSA	Rational Software Architect
RTM	Rational TestManager
SMA	Sistema MultiAgente
SUT	System Under Test
TMM	Test Maturity Model
TMMi	Test Maturity Model integration
UML	Unified Modeling Language
UTP	UML Testing Profile
UTP-C	UML Testing Profile for Coordination