

# DISSERTAÇÃO DE MESTRADO

UNIVERSIDADE FEDERAL DE SÃO CARLOS  
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA  
PROGRAMA DE PÓS-GRADUAÇÃO EM  
CIÊNCIA DA COMPUTAÇÃO

## **“Caracterização de Desafios e Estratégias de Teste para Sistemas Adaptativos”**

**ALUNO:** Bento Rafael Siqueira  
**ORIENTADOR:** Prof. Dr. Fabiano Cutigi Ferrari

São Carlos  
Junho/2016

CAIXA POSTAL 676  
FONE/FAX: (16) 3351-8233  
13565-905 - SÃO CARLOS - SP  
BRASIL

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**CARACTERIZAÇÃO DE DESAFIOS E  
ESTRATÉGIAS DE TESTE PARA SISTEMAS  
ADAPTATIVOS**

**BENTO RAFAEL SIQUEIRA**

**ORIENTADOR: PROF. DR. FABIANO CUTIGI FERRARI**

São Carlos – SP

Junho/2016

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**CARACTERIZAÇÃO DE DESAFIOS E  
ESTRATÉGIAS DE TESTE PARA SISTEMAS  
ADAPTATIVOS**

**BENTO RAFAEL SIQUEIRA**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Engenharia de Software

Orientador: Prof. Dr. Fabiano Cutigi Ferrari

São Carlos – SP

Junho/2016

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária UFSCar  
Processamento Técnico  
com os dados fornecidos pelo(a) autor(a)

S618c Siqueira, Bento Rafael  
Caracterização de desafios e estratégias de teste para sistemas adaptativos / Bento Rafael Siqueira. - São Carlos : UFSCar, 2016.  
164 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2016.

1. Sistemas Adaptativos. 2. Teste de Software. 3. Desafios de Teste. 4. Sistemas Sensíveis ao Contexto.  
I. Título.




UNIVERSIDADE FEDERAL DE SÃO CARLOS  
Centro de Ciências Exatas e de Tecnologia  
Programa de Pós-Graduação em Ciência da Computação

---

Folha de Aprovação


---

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de Dissertação de Mestrado do candidato Bento Rafael Siqueira, realizada em 30/06/2016.



---

Prof. Dr. Fabiano Cutigi Ferrari  
(UFSCar)




---

Prof. Dr. Valter Vieira de Camargo  
(UFSCar)

---

Prof. Dr. Arilo Claudio Dias Neto  
(UFAM)

Certifico que a sessão de defesa foi realizada com a participação à distância do membro Prof. Dr. Arilo Claudio Dias Neto. Depois das arguições e deliberações realizadas, o participante à distância está de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa do aluno Bento Rafael Siqueira.



---

Prof. Dr. Fabiano Cutigi Ferrari  
Coordenador da Comissão Examinadora  
(UFSCar)

# Agradecimentos

---

Primeiramente, desculpem-me se me esqueci de alguém, nesses agradecimentos, foram tantas as pessoas que me ajudaram e me incentivaram que não consigo as sumarizar.

Gostaria de agradecer a Deus, por ter provido tudo o que conquistei e por dar forças e sonhos para trabalhar todos os dias em busca da realização pessoal e profissional.

Agradeço ao meu Pai (seu “Nico”) por ter apoiado em todas as etapas da minha vida e, também, por ser meu grande incentivo de trabalhar todos os dias (assim como ele sempre fez).

Agradeço a minha Mãe (Dona “Sirlene”) por desde pequeno sempre ter zelado por mim e, também, pela compreensão nos momentos difíceis dessa empreitada (que não foram poucos).

Agradeço a minha querida Kathiani, por ser uma pessoa maravilhosa, estar sempre ao meu lado e, também, por me aturar e me entender nos momentos difíceis.

Agradeço aos meus irmãos (Osmar, Jean e Daniel) por todo apoio e incentivo, não teria conseguido nada sem vocês.

Agradeço ao meu orientador (O Fabiano) por sempre ter sido um excelente orientador e amigo, o qual podemos ter sempre como uma referência no modo de agir e de conduzir as coisas.

Agradeço à professora Sandra, ao professor Auri e aos meus amigos do Laboratório LaPES (Odair, Abade, Alessandro, Jésus, Gaspar, Ana, Thiago, Cleitin, Gasltadi, Ivan, Elis, Guido, Jaum, Pacini, Lucas, Cavamura, Fábio...), muito obrigado pelo apoio (principalmente nas parcerias e revisões de textos rsrs, chega um momento (após várias revisões) que não conseguimos mais ver nossos próprios erros).

Agradeço aos Professores Edes Costa, Walison Barberá, Cássio Adorni e Rodrigo Faccioli, por me ajudarem e incentivarem desde lá no início (os quais tive sempre como referência para modelar o meus próprios sonhos).

Agradeço à Travel Technology Interactive (principalmente ao Juliano Gomes) por sempre ter apostado em mim e ter ajudado-me nos momentos difíceis iniciais ao ingresso do Mestrado.

Agradeço à CNPq pelo apoio financeiro, assim que me tornei aluno regular no PPGCC/UFSCar.

Agradeço à FAPESP ter estado apoiando-me financeiramente para capacitação e treinamento técnico à docência.

Agradeço à UNIVESP – principalmente à Mônica Garbin e ao Lucas Nóbilo (juntamente com o professor Waldomiro) pela oportunidade em poder atuar em disciplinas que muito me contribuíram à formação.

Por fim, agradeço a todos do PPGCC/UFSCar pelos excelentes professores, pesquisadores e amigos, os quais tive o prazer de conhecer e de trabalhar.

**Contexto:** Abordagens de teste tradicionais são ainda incipientes e superficialmente avaliadas quanto à sua efetividade em revelar defeitos em Sistemas Adaptativos (SAs). É nítido o desafio de garantir a corretude de SAs levando-se em consideração as características de adaptação desses sistemas, culminando na seguinte pergunta: como caracterizar adequadamente, de forma abrangente, as dificuldades para se testar SAs? Na literatura não foi encontrada qualquer estratégia de teste que seja guiada por desafios de teste caracterizados. **Objetivos:** Neste trabalho buscou-se compreender e caracterizar os desafios impostos à atividade de teste de SAs. Uma vez caracterizados, os desafios, propôs-se investigar estratégias de teste baseadas nestes desafios. **Metodologia:** Para atingir o objetivo seguiram-se as etapas: (i) caracterização do estado da arte por meio de uma Revisão Sistemática (RS); (ii) investigação da existência de desafios de teste em SAs presentes em repositórios de código-fonte; (iii) definição de uma estratégia de teste, denominada T, com base nos desafios caracterizados da RS; (iv) definição de uma estratégia de teste combinada, denominada T\*, composta por três abordagens de teste; (v) avaliação da efetividade das estratégias T e T\*; e, por fim, (vi) investigação da presença dos desafios nos SAs analisados, por meio das estratégias T e T\*. **Resultados:** Os principais resultados foram: (i) uma caracterização de desafios para o teste de SAs; e (ii) e uma definição de estratégias de teste baseadas nos desafios caracterizados. A avaliação foi realizada com a aplicação de um estudo exploratório, utilizando um SA, de uma estratégia de teste somente baseada em desafios (T) e de uma estratégia de teste baseada em desafios combinada com outras abordagens (T\*), a fim de comparar os resultados de ambas as estratégias. **Conclusão:** A caracterização dos desafios auxiliou na definição e execução de estratégias de teste, as quais mitigaram os desafios presentes nos SAs e puderam identificar defeitos.

**Palavras-chave:** Sistemas Adaptativos; Teste de Software; Desafios de Teste; Sistemas Sensíveis ao Contexto.

# Abstract

---

**Context:** Traditional testing approaches, considering the context of Adaptive Systems (ASs), have been evaluated incipiently and superficially with respect to their effectiveness in identifying faults. It is very clear the challenge of guaranteeing the correctness of ASs, due to the adaptive properties of these systems. Thus, the following question raises: how to characterise adequately and broadly the difficulties for testing ASs? There is not an approach that is driven by challenges found in the literature. **Goals:** In this work, we analysed and characterised the challenges faced by the testing activity when applied to ASSs. By characterising the challenges, we investigated testing strategies based on them. **Methodology:** In order to achieve the goals, we performed the following steps: (i) characterising the state-of-art of testing of ASs, using a Systematic Literature Review (SLR); (ii) investigating the challenges in source codes of real ASs that are available online in source code repositories; (iii) defining the testing strategy named T, based on the characterised challenges; (iv) defining the testing strategy T\* based on the challenges and comprised by three testing approaches; (v) evaluating the effectiveness of the strategies T and T\*; and (vi) investigating the challenges along the application of T and T\*. **Results:** the main results were: (i) a characterisation of fine-grained and coarse-grained challenges for testing ASs; and (ii) two testing strategies based on such challenges. We evaluated the strategies by running an exploratory study which encompassed one AS and both testing strategies. **Conclusion:** The characterisation of challenges supported the definition and execution of the testing strategies. These strategies were able to reduce the impact of the challenges and to identify faults in the AS.

**Keywords:** adaptive systems; software testing; testing challenges; context-aware systems.



---

# Listagens de Códigos

---

5.1	Defeito que foi corrigido para que o SA pudesse funcionar . . . . .	104
5.2	Defeito de Predicado Morto . . . . .	105
5.3	Defeitos de Ativação não Determinística . . . . .	106
5.4	Um fragmento de código-fonte referente a um dos casos de teste criados. . .	110

---

# Lista de Figuras

---

2.1	Loop de controle fechado MAPE-K – adaptada do trabalho de Hurtado et al. (2011). . . . .	9
2.2	Interação entre componentes de um sistema por meio de evento e ciclo de tempo – adaptada do trabalho de Iglesia (2014). . . . .	10
2.3	Exemplo ilustrativo de um <i>Smartphone</i> Equipado com Sensores em dois contextos diferentes. . . . .	11
2.4	Exemplo de um programa ( $p$ ) e suas respectivas instâncias de contexto $ins(C)$ 's. . . . .	13
2.5	Exemplo ilustrativo do cálculo da HD. . . . .	15
2.6	Exemplo ilustrativo do <i>PhoneAdapter</i> (Sama et al., 2010a). . . . .	18
2.7	Mapeamento do código-fonte do <i>PhoneAdapter</i> . . . . .	19
2.8	O comportamento do <i>PhoneAdapter</i> para obtenção de dados de sensores. . . . .	20
2.9	O comportamento do <i>PhoneAdapter</i> para o armazenamento de dados no tempo de execução. . . . .	21
2.10	O comportamento do <i>PhoneAdapter</i> para a análise de dados no tempo de execução. . . . .	21
2.11	O comportamento do <i>PhoneAdapter</i> para a adaptação no tempo de execução. . . . .	22
2.12	Processo de Teste de Software adaptado do trabalho de Höhn (2011). . . . .	24
3.1	Processo de pesquisa adaptado de Fabbri et al. (2013). . . . .	28
3.2	Associações entre grafos utilizando a abordagem S05 (Lu et al., 2006). . . . .	46
3.3	Um Fluxo de Contexto com um terremoto artificial (Munoz e Baudry, 2009). . . . .	48
3.4	Máquina de estados finita de adaptação ( <i>Adaptation Finite-State Machine – A-FSM</i> ) adaptada de Sama et al. (2010a). . . . .	49
4.1	Desafios genéricos. . . . .	54
5.1	Modelo GQM. . . . .	78
5.2	O uso de uma A-FSM com notação de Máquina de <i>Mealy</i> . . . . .	86
5.3	Representação para iniciar uma análise de Relação Metamórfica. . . . .	94
5.4	Grafos sgps e sbt das variáveis de contexto do <i>PhoneAdapter</i> (SA5). . . . .	96

5.5	Grafos volume, vibracall e modo avião das variáveis de contexto do <i>Phone-Adapter</i> (SA5). . . . .	97
5.6	Associações entre grafos utilizando a abordagem S05 (Lu et al., 2006). . . . .	100
5.7	Automatização das Estratégias de teste Executada. . . . .	111
B.1	Arquitetura do sistema SA1-Novi. . . . .	143
B.2	Arquitetura do sistema SA2-STMobile. . . . .	144
B.3	Arquitetura do sistema SA3-Context-Aware Music Player. . . . .	145
B.4	Arquitetura do sistema SA4-Proxilence. . . . .	147

---

# Lista de Tabelas

---

2.1	Propriedades de sistemas adaptativos . . . . .	16
3.1	Máquinas de busca utilizadas na RS . . . . .	31
3.2	Evolução da <i>string</i> de busca. . . . .	32
3.3	Número de estudos retornados por base de dados. . . . .	34
3.4	Número de estudos selecionados e descartados na seleção inicial. . . . .	35
3.5	Número de estudos selecionados e descartados na seleção final. . . . .	36
3.6	Conjunto de estudos da seleção final. . . . .	38
3.7	Conjunto de estudos da seleção final. . . . .	39
4.1	Desafios específicos – Parte 1. . . . .	65
4.2	Desafios específicos – Parte 2. . . . .	66
4.3	As propriedades nos SAs analisados. . . . .	67
4.4	Propriedades dos SAs analisados e Desafios Específicos . . . . .	68
4.5	Desafios Específicos relacionados aos SAs analisados . . . . .	69
4.6	Relacionamento entre abordagens de teste e desafios específicos - Parte 1. . . . .	70
4.7	Relacionamento entre abordagens de teste e desafios específicos - Parte 2. . . . .	71
4.8	Relacionamento entre abordagens e fases de teste . . . . .	72
4.9	Relacionamento entre abordagens, desafios específicos e SAs - Parte 1. . . . .	74
4.10	Relacionamento entre abordagens, desafios específicos e SAs - Parte 2. . . . .	75
5.1	Métricas do GQM . . . . .	80
5.2	Ranking das abordagens com base nos desafios específicos nos SAs analisados . . . . .	83
5.3	Tabela de Interpretação do Modelo GQM. . . . .	84
5.4	Algumas possíveis transições entre estados realizadas pelo <i>PhoneAdapter</i> (SA5), adaptadas do trabalho de Sama et al. (2010a). . . . .	88
5.5	A agenda de atividades de um usuário . . . . .	89
5.6	Casos de Teste criados com a Abordagem S03 (Tse et al., 2004) . . . . .	93
5.7	Restrições de domínio baseadas na agenda do usuário . . . . .	95
5.8	Valores das Variáveis de Contexto nas suas respectivas transições/situações. . . . .	102
5.9	Fluxos de contextos onde Terremotos Artificiais foram encontradas com a Abordagem S14 (Munoz e Baudry, 2009). . . . .	103

5.10	Falhas externalizadas na aplicação das Estratégias T e T* . . . . .	107
A.1	Sistemas recuperados ao aplicar as <i>strings</i> de busca nos repositórios . . . . .	136
A.2	Sistemas selecionados ao aplicar o critério de seleção . . . . .	136
A.3	Sistemas caracterizados por linguagem de programação . . . . .	137
B.1	Questões e diretrizes para identificar propriedades de SAs . . . . .	140
B.2	Sistemas com as propriedades obrigatórias. . . . .	141
C.1	Autores que estão em mais de um dos estudos selecionados. . . . .	152
C.2	Sínteses dos estudos primários - Parte 1. . . . .	153
C.3	Sínteses dos estudos primários - Parte 2. . . . .	154
C.4	Sínteses dos estudos primários - Parte 3. . . . .	155
C.5	Sínteses dos estudos primários - Parte 4. . . . .	156
C.6	Desafios para testar Sistemas Adaptativos - Parte 1. . . . .	157
C.7	Desafios para testar Sistemas Adaptativos - Parte 2. . . . .	158
C.8	Defeitos de Sistemas Adaptativos - Parte 1. . . . .	159
C.9	Defeitos de Sistemas Adaptativos - Parte 2. . . . .	160
D.1	Mapeamento de casos de teste para o critério Todas Situações . . . . .	162
D.2	Mapeamento de casos de teste para o critério Todas Associações def-use . . . . .	163
D.3	Mapeamento de casos de teste para o critério Todas Situações fora do Padrão . . . . .	164

---

# Lista de Abreviaturas e Siglas

---

<b>A-FSM</b>	Máquina de Estados Finita de Adaptação ( <i>Adaptation Finite-State Machine</i> )
<b>ASTT</b>	Teste de Tabela de Abalos Artificiais ( <i>Artificial Shaking Table Testing</i> )
<b>BPMN</b>	Modelo e Notação de Processo de Negócio ( <i>Business Process Model and Notation</i> )
<b>BT</b>	<i>Bluetooth</i>
<b>CaFG</b>	Grafos de Fluxo de Controle Sensíveis ao Contexto ( <i>Control Context-aware Flow Graph</i> )
<b>CE</b>	Critério de Exclusão
<b>CFG</b>	Grafos de Fluxo de Controle ( <i>Control Flow Graph</i> )
<b>CI</b>	Critério de Inclusão
<b>CM</b>	Gerenciador de Contexto ( <i>Context Manager</i> )
<b>GPS</b>	Sistema de Posicionamento Global ( <i>Global Positioning System</i> )
<b>GQM</b>	Objetivos, Questões e Métricas (Goals, Questions and Metrics)
<b>HD</b>	Distância de Hamming ( <i>Hamming Distance</i> )
<b>ID</b>	Identificador
<b>MAPE-K</b>	Monitor, Analisador, Planejador, Executor e Conhecimento
<b>MEF</b>	Máquina de Estados Finita
<b>QP</b>	Questão Principal
<b>QS</b>	Questão Secundária
<b>RM</b>	Relação Metamórfica ( <i>Metamorphic Relation</i> )
<b>RS</b>	Revisão Sistemática ( <i>Systematic Review</i> )
<b>SA</b>	Sistema Adaptativo ( <i>Adaptive System</i> )
<b>SAA</b>	Sistema Autoadaptativo ( <i>Self-adaptive System</i> )
<b>SBES</b>	Simpósio Brasileiro de Engenharia de Software
<b>SC</b>	Desafio Específico ( <i>Specific Challenge</i> )

# Sumário

---

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Contexto . . . . .	1
1.2	Objetivo . . . . .	3
1.3	Metodologia . . . . .	4
1.4	Organização do Trabalho . . . . .	6
<b>2</b>	<b>Fundamentação teórica</b>	<b>7</b>
2.1	Considerações Iniciais . . . . .	7
2.2	Sistemas Adaptativos . . . . .	7
2.2.1	Conceitos de Sistemas Adaptativos e o MAPE-K . . . . .	8
2.2.2	Definições Utilizadas para Contexto . . . . .	11
2.2.3	Propriedades de Sistemas Adaptativos . . . . .	14
2.2.4	Um Exemplo Ilustrativo de um Sistema Adaptativo . . . . .	17
2.3	Teste de Software . . . . .	22
2.3.1	Conceitos e Terminologias de Teste . . . . .	22
2.3.2	Processo de Teste de Software . . . . .	23
2.4	Considerações Finais . . . . .	25
<b>3</b>	<b>Uma Revisão Sistemática sobre Teste de Sistemas Adaptativos</b>	<b>26</b>
3.1	Considerações Iniciais . . . . .	26
3.2	Processo de Pesquisa . . . . .	27
3.3	Planejamento da Pesquisa . . . . .	28
3.3.1	Objetivo de Pesquisa . . . . .	29
3.3.2	Questões de Pesquisa . . . . .	29
3.3.3	Estratégia de Busca Utilizada no Trabalho Original . . . . .	31
3.3.4	Estratégia de Busca Utilizada Neste Trabalho . . . . .	32
3.3.5	Critérios de Seleção Para Os Estudos . . . . .	33
3.4	Condução da Pesquisa . . . . .	34
3.4.1	Resultados da Aplicação de Strings . . . . .	34
3.4.2	Resultados da Seleção Inicial . . . . .	35
3.4.3	Resultados da Seleção Final . . . . .	35

3.4.4	Estudos Publicados Pelos Autores Citados . . . . .	36
3.5	Os Estudos Seleccionados . . . . .	37
3.6	Dados Extraídos e Empacotamento . . . . .	40
3.6.1	Abordagens de Teste de SAs e Técnicas Associadas . . . . .	40
3.6.2	Desafios para o Teste de SAs . . . . .	42
3.6.3	Tipos de Defeitos de Software no Contexto de SAs . . . . .	43
3.7	Detalhamento de Algumas Abordagens de Teste . . . . .	44
3.7.1	Abordagem S03 (Tse et al., 2004) . . . . .	44
3.7.2	Abordagem S05 (Lu et al., 2006) . . . . .	45
3.7.3	Abordagem S14 (Munoz e Baudry, 2009) . . . . .	47
3.7.4	Abordagem S17 (Sama et al., 2010a) . . . . .	49
3.8	Considerações Finais . . . . .	50
<b>4</b>	<b>Caracterização de Desafios e Abordagens de Teste para Sistemas Adap-</b>	
	<b>tativos</b>	<b>51</b>
4.1	Considerações Iniciais . . . . .	51
4.2	Os Desafios Específicos . . . . .	52
4.3	Os Desafios Genéricos . . . . .	53
4.3.1	Discussão . . . . .	59
4.4	A Presença dos Desafios em Sistemas . . . . .	59
4.4.1	Desafios Específicos Relacionados às Propriedades dos Sistemas . . . . .	61
4.5	Abordagens de Teste . . . . .	62
4.5.1	Abordagens de Teste que Estão Relacionadas aos Desafios Específicos . . . . .	62
4.5.2	Abordagens de Teste Relacionadas às Fases de Teste . . . . .	63
4.5.3	Abordagens de Teste Relacionadas aos Desafios Específicos Presentes nos SAs . . . . .	73
4.6	Considerações Finais . . . . .	76
<b>5</b>	<b>Um Estudo Exploratório de Abordagens de Teste para Sistemas Adap-</b>	
	<b>tativos</b>	<b>77</b>
5.1	Considerações Iniciais . . . . .	77
5.2	Definição do Estudo . . . . .	77
5.2.1	Objetivos . . . . .	78
5.2.2	Questões . . . . .	79
5.2.3	Métricas . . . . .	80
5.2.4	Tabela de Intepretação do GQM . . . . .	80
5.3	Procedimento de Seleção de Abordagens de Teste e SAs para Serem Testados . . . . .	81
5.3.1	Definição da Estratégia T . . . . .	81
5.3.2	Definição da Estratégia T* . . . . .	81
5.3.3	Seleção e Revisão do Modelo de Especificação de Teste . . . . .	85
5.4	Respostas às Questões [Q1]–[Q4] do GQM . . . . .	90
5.5	Execução dos Testes . . . . .	91
5.5.1	Execução da Estratégias T . . . . .	91
5.5.2	Execução da Estratégia T* . . . . .	95
5.6	Avaliação das Estratégias T e T* . . . . .	104



5.6.1	Defeitos Preliminares Encontrados . . . . .	104
5.6.2	Defeitos e Falhas Encontradas pelas Abordagens . . . . .	104
5.7	Respostas às Questões [Q5] e [Q6] do GQM . . . . .	108
5.8	Discussões Adicionais . . . . .	109
5.8.1	Automatização da aplicação das Estratégias T e T*: . . . . .	109
5.8.2	Desafios Observados com a Aplicação das Abordagens . . . . .	111
5.8.3	Desafios Não Observados com a Aplicação das Abordagens . . . . .	114
5.9	Ameaças à Validade . . . . .	115
5.10	Considerações Finais . . . . .	116
<b>6</b>	<b>Conclusões</b>	<b>118</b>
6.1	Revisitando as Contribuições do Trabalho . . . . .	119
6.2	Trabalhos Relacionados . . . . .	120
6.3	Limitações . . . . .	121
6.4	Trabalhos Futuros . . . . .	121
	<b><i>Referências</i></b>	<b>123</b>
	<b>Apêndice</b>	<b>135</b>
<b>A</b>	<b>Um Processo de Busca por Sistemas Adaptativos</b>	<b>135</b>
A.1	Planejamento e Execução da Seleção . . . . .	135
<b>B</b>	<b>Uma Análise de Propriedades de Sistemas Adaptativos</b>	<b>138</b>
B.1	O Processo de Identificação de Propriedades nos Sistemas Adaptativos . . . . .	139
B.2	As Propriedades nos Sistemas Adaptativos . . . . .	140
B.3	Características dos Sistemas Adaptativos Selecionados . . . . .	141
B.3.1	As Características do SA1-Noví . . . . .	142
B.3.2	As Características do SA2-STMobile . . . . .	143
B.3.3	As Características do SA3-Context-Aware Music Player . . . . .	145
B.3.4	As Características do SA4-Proxilence . . . . .	146
B.3.5	As Características do SA5-PhoneAdapter . . . . .	146
B.4	Análise e Discussão dos Sistemas . . . . .	148
<b>C</b>	<b>Demais Tabelas da Revisão Sistemática</b>	<b>150</b>
<b>D</b>	<b>Demais Tabelas do Estudo Exploratório</b>	<b>161</b>

---

# Introdução

---

---

## 1.1 Contexto

Por volta da década de 60, quando surgiu o termo *Engenharia de Software*, com a então chamada *crise do software*, engenheiros identificaram grandes problemas de complexidade nos sistemas de software (Lalanda et al., 2013). Com o aumento do tamanho e complexidade dos sistemas, o problema de se projetar um software caminhou para além dos algoritmos e estrutura de dados (Garlan e Shaw, 1994). Por exemplo, um software que envia mensagens entre dois usuários precisa lidar com a infraestrutura de cada usuário. O usuário que envia pode estar utilizando um sistema operacional diferente do usuário que recebe a mensagem. Isso implicaria em, por exemplo, lidar com diferentes dispositivos de software/hardware. Apesar da complexidade, os sistemas de software têm estado cada vez mais presentes na vida social e profissional das pessoas (Cheng et al., 2009). Entretanto, os riscos da evolução e a necessidade de se adaptar a novas demandas de mercado são características inevitáveis (Oreizy et al., 1999). Um exemplo é a tecnologia móvel que levou ao aumento do desenvolvimento de infraestruturas pervasivas<sup>1</sup> que apoiam a “computação em qualquer hora e lugar” (Liu, 2013).

---

<sup>1</sup>Uma estrutura pervasiva é aquela que tende a se propagar por toda parte (Stevenson, 2011).

Para atender a essa demanda, os Sistemas Adaptativos (SAs)<sup>2</sup> podem ser desenvolvidos com o intuito de se comportarem de acordo com as alterações do ambiente. Caso ocorra alguma mudança no seu contexto de execução, um SA se adapta para operar de acordo com o novo cenário (Yu e Gao, 2014). O termo *contexto* remete a um conjunto de variáveis – denominadas Variáveis de Contexto – em que cada variável contém um conjunto de atributos de ambiente (Wang et al., 2014). Um atributo de ambiente, por exemplo, pode representar a latitude e longitude de um dispositivo GPS utilizado por um usuário. Neste sentido, uma aplicação dirigida por regras pode se adaptar automaticamente, de acordo com mudanças nas variáveis de contexto, para auxiliar o usuário em seu dia-a-dia. Caso as regras não sejam pré-definidas, o usuário, neste caso, precisaria apenas configurar as regras e alguns possíveis contextos.

Embora haja uma crescente demanda por SAs, ressalta-se que planejar, projetar e manter tais sistemas utilizando abordagens tradicionais são tarefas difíceis de se realizar (Lalanda et al., 2013). Assim, necessitou-se de um conjunto de práticas personalizadas para desenvolver SAs. Como resultado, nasceu o termo *Computação Autônoma* (do inglês, *Autonomic Computing*) (Horn, 2001), que descreve capacidades e componentes que podem ser desenvolvidos para apoiar o autogerenciamento de SAs.

A Engenharia de Software introduziu uma gama de técnicas que têm ajudado na construção de software e na condução de processos de software com qualidade (Lalanda et al., 2013). No contexto de SAs, a garantia de qualidade é um esforço fundamental, pois se houver falhas nestes sistemas, os usuários que os utilizam podem ser impactados (Wang et al., 2014). Obviamente, falhas impactam usuários de qualquer tipo de sistema. Entretanto, um SA pode ser autogerenciável. Assim, diferentemente de um sistema comum, que é monitorado por um usuário a todo momento, o SA pode mudar seu comportamento conforme o seu contexto de execução.

Em relação à qualidade do software desenvolvido, ela pode ser categorizada em dois grupos de fatores: (i) fatores diretamente mensuráveis – por exemplo, defeitos que podem ser revelados durante o teste de software – e (ii) fatores que podem ser mensurados indiretamente – por exemplo, características de usabilidade e manutenibilidade (Pressman, 2010). No contexto da qualidade de software e de seus fatores diretamente mensuráveis, o teste de software tem a finalidade de revelar defeitos. Em contrapartida, é impossível provar a inexistência de defeitos, pois para isso seria necessário executar o teste de forma exaustiva, ou seja, com todas as entradas possíveis (Delamaro et al., 2007). Apesar disso, o teste, quando realizado de maneira sistemática e criteriosa, pode contribuir para o

---

<sup>2</sup>Alguns autores também os denominam como *Sistemas Sensíveis ao Contexto* (Lalanda et al., 2013; Lu et al., 2006; Matalonga et al., 2015b; Sama et al., 2010a)

aumento da confiança do software de maneira geral (Harrold, 2000), mesmo que todas as possíveis entradas para sua execução não sejam fornecidas.

Wang et al. (2014) mencionam que abordagens de teste tradicionais são ineficientes no contexto de SAs, devido às características inerentes a esses sistemas. Como um exemplo, uma técnica de teste que leva em consideração apenas um domínio de entrada e resultados para tal domínio é ineficaz em um SA, durante sua adaptação, que pode mudar o resultado para determinado domínio. Assim, é nítido o desafio de garantir a corretude de SAs levando-se em consideração a adaptação desses sistemas. Portanto, detectar defeitos ou exercitar implementações de maneira efetiva não é uma tarefa trivial (King et al., 2011a).

O principal problema abordado neste trabalho é: como caracterizar adequadamente, de forma abrangente, as dificuldades para se testar SAs? Neste sentido, quais maneiras seriam eficazes de se testar SAs? Na literatura não foi encontrada qualquer estratégia de teste que seja guiada por desafios de teste caracterizados.

No trabalho de Ferrari et al. (2011) caracterizaram-se desafios impostos à atividade de teste em SAs (por meio de uma Revisão Sistemática (RS) da Literatura). Ademais, levantou-se a hipótese: *Saber quais os possíveis desafios que podem ser enfrentados, durante a atividade do teste em SAs, pode ajudar na definição de uma estratégia de teste para esse tipo de sistema.* Com base na hipótese levantada define-se, na sequência o objetivo deste trabalho.

## 1.2 Objetivo

Neste trabalho buscou-se compreender e caracterizar os desafios impostos à atividade de teste de SAs. Uma vez caracterizados, os desafios, propôs-se investigar estratégias de teste baseadas nestes desafios. Para tal objetivo, as seguintes questões de pesquisa foram definidas:

- **QP1:** Quais são os desafios impostos pelos SAs à atividade de teste?
- **QP2:** Quais são os desafios caracterizados encontrados durante a atividade de teste em SAs reais?
- **QP3:** Quais são os defeitos encontrados com a aplicação das estratégias de teste baseadas em desafios caracterizados neste trabalho?
- **QP4:** Quais são os desafios mitigados com a aplicação das estratégias de teste baseadas em desafios?

A metodologia seguida para alcançar o objetivo e responder as questões de pesquisa é apresentada a seguir.

## 1.3 Metodologia

O objetivo do trabalho foi alcançado por meio da realização de algumas etapas, brevemente descritas a seguir:

- 1 - *Atualização e extensão de uma RS previamente realizada*: nesta etapa reproduziu-se a RS realizada por Ferrari et al. (2011), sem filtro por data, avaliando-se novos trabalhos encontrados e estendeu-se a RS, atualizando a estratégia de identificação de estudos, com a adição de novas palavras-chaves e com o uso da técnica *Snowballing*;
- 2 - *Investigação de SAs disponíveis em repositórios de código-fonte na Web*: nesta etapa aplicaram-se palavras-chaves referentes a SAs em repositórios de código-fonte, a fim de encontrar códigos-fonte candidatos a serem analisados. Além disso, definiu-se uma diretriz de propriedades para auxiliar na seleção de SAs;
- 3 - *Caracterização de desafios*: nesta etapa agruparam-se os desafios impostos à atividade de teste presentes na literatura, relacionando os desafios mencionados por diferentes autores em diferentes contextos de aplicação. Em tal etapa obteve-se uma lista de desafios específicos e uma de genéricos<sup>3</sup>;
- 4 - *Caracterização de abordagens e seu relacionamento com as fases de teste*: nesta etapa avaliaram-se as abordagens de teste identificadas na RS realizada, categorizando tais abordagens de acordo com um processo de teste genérico. Assim, identificaram-se quais as atividades de teste poderiam ser realizadas com uso das abordagens;
- 5 - *Análise de desafios e SAs*: nesta etapa relacionaram-se os SAs selecionados na etapa 2 com os desafios caracterizados na etapa 3. Neste relacionamento levaram-se em consideração as propriedades definidas para seleção de SAs, definidas na etapa 2, e os desafios específicos caracterizados durante a etapa 3;
- 6 - *Análise de desafios e abordagens de teste*: nesta etapa relacionaram-se os desafios específicos, caracterizados na etapa 3, com abordagens de teste recuperadas na etapa 1. Neste relacionamento, levaram-se em consideração as características das abordagens de teste e suas possíveis contribuições em mitigar cada um dos desafios

---

<sup>3</sup>Um desafio específico é um único desafio de teste para SAs; já um desafio genérico é um agrupamento de desafios específicos de teste para SAs

específicos caracterizados. Além disso, ressalta-se que os desafios genéricos não foram utilizados durante este relacionamento. Utilizaram-se os desafios específicos para se obter um relacionamento com uma granularidade fina (ou seja, não se analisou um grupo de desafios, mas sim cada um dos desafios pertencentes a cada um dos grupos, que compõem os desafios genéricos);

- 7 - *Análise de desafios, abordagens e SAs*: nesta etapa identificaram-se as abordagens de teste, recuperadas na etapa 1, que mais mitigariam desafios presentes nos SAs analisados. Neste relacionamento, levaram-se em consideração os desafios específicos que mais estavam presentes nos SAs, por meio da etapa 5, e quais abordagens de teste que se relacionam a estes desafios, por meio da etapa 6;
- 8 - *Seleção de abordagens e SAs*: nesta etapa selecionou-se (i) a abordagem de teste que mais se relaciona aos desafios específicos (etapa 6); (ii) a abordagem de teste que mais se relaciona à fase projeto e à fase finalização de teste (etapa 4); (iii) a abordagem de teste que mais se relaciona a desafios presentes nos SAs analisados (etapa 5); e por fim, (iv) o SA que mais relaciona a diretriz de propriedades (etapa 2).
- 9 - *Seleção de um modelo subjacente de teste*: nesta etapa utilizou-se uma máquina de estados finita de adaptação, denominada A-FSM, com uso da notação de uma máquina de *mealy*. Com isso, modelou-se o comportamento do SA selecionado na etapa 2, representando-se a execução de casos de teste;
- 10 - *Definição das estratégias de teste*: nesta etapa definiram-se as estratégias de teste  $T$  e  $T^*$ , com as abordagens de teste selecionadas na etapa 8. Durante esta etapa, as estratégias de teste definidas foram aplicadas utilizando a máquina de *mealy*, modelada na etapa 9. Ressalta-se que  $T$  e  $T^*$  são, respectivamente, os nomes dados às estratégias de teste utilizadas neste trabalho. A primeira se refere a uma abordagem de teste isolada e a segunda a uma combinação de abordagens de teste;
- 11 - *Automatização das estratégias  $T$  e  $T^*$* : nesta etapa automatizou-se a aplicação das estratégias de teste  $T$  e  $T^*$ , utilizando-se como base a máquina de *mealy* da etapa 9. Assim, utilizou-se o modelo subjacente de teste, a fim de representar execução de teste por meio de formalismos como a transição entre estados.
- 12 - *Análise dos resultados*: nesta etapa avaliaram-se os defeitos encontrados antes e durante a aplicação das estratégias  $T$  e  $T^*$ , caracterizando-se tais defeitos de acordo com a RS realizada na etapa 1. Além disso, investigaram-se os desafios que, de fato,

estavam presentes durante a aplicação das estratégias T e T\* e como as estratégias mitigariam os desafios relacionados.

## 1.4 Organização do Trabalho

Neste capítulo foram apresentados o contexto no qual este trabalho foi inserido, a motivação para o trabalho, os objetivos alcançados e a metodologia utilizada para alcançar os objetivos. O restante do trabalho está organizado da seguinte forma:

- No *Capítulo 2* foram apresentados os principais conceitos relacionados ao Teste Software e aos SAs.
- No *Capítulo 3* foi apresentada a RS realizada, incluindo detalhes sobre o planejamento, condução e a análise dos resultados.
- No *Capítulo 4* foi descrita a caracterização de desafios e abordagens de teste em SAs.
- No *Capítulo 5* foi descrito o estudo exploratório realizado neste trabalho.
- No *Capítulo 6* foram descritas as conclusões, envolvendo as questões de pesquisa elencadas neste Capítulo.

O trabalho também contém quatro apêndices que estão organizados da seguinte forma: no *Apêndice A* foi descrito os passos realizados durante o processo de busca por SAs nos repositórios de código-fonte; no *Apêndice B* foram descritas a análise realizada dos SAs e a diretriz de propriedades definida; no *Apêndice C* estão as tabelas utilizadas durante a RS e o estudo exploratório deste trabalho; e, por fim, no *Apêndice D* estão as tabelas auxiliares utilizadas durante a condução do estudo exploratório.

---

# Fundamentação teórica

---

---

## 2.1 Considerações Iniciais

Os dois grandes temas deste trabalho são Sistemas Adaptativos (SAs) e o Teste de Software. Neste capítulo apresentam-se alguns dos principais conceitos a respeito desses dois temas. Assim, inicialmente, conceitos de SAs são introduzidos por meio de algumas definições de SAs e por uma introdução ao principal modelo teórico utilizado na literatura, o MAPE-K, para representar loops de controles de um SA. Além disso, são introduzidas as definições de contextos utilizadas, seguidas das propriedades de SAs e de um exemplo ilustrativo do SA utilizado no estudo exploratório deste trabalho. O teste de software é introduzido neste capítulo, por meio de conceitos e terminologias, seguidas das principais técnicas de teste presentes na literatura. Além disso, apresenta-se um processo genérico de teste, a fim de fundamentar as atividades presentes nas fases de teste de software.

## 2.2 Sistemas Adaptativos

Na sequência são descritos conceitos de Sistemas Adaptativos (SAs), com uma breve introdução ao modelo teórico MAPE-K. Além disso, apresentam-se algumas definições a



respeito de contextos utilizadas neste trabalho, seguidas de um exemplo ilustrativo que utiliza como base dispositivos móveis cientes de contexto.

### 2.2.1 Conceitos de Sistemas Adaptativos e o MAPE-K

Um Sistema Adaptativo (SA) é um sistema cujo comportamento se altera em decorrência de mudanças nas necessidades do usuário ou de contexto. Já um sistema autoadaptativo (SAA) é um tipo especial de SA que pode também reagir a mudanças em seu próprio ambiente operacional (Evers et al., 2014; Oreizy et al., 1999). Diversas áreas estão interessadas no desenvolvimento e pesquisas de SAs, dentre essas destacam-se computação autônoma, redes de computadores, sistemas multiagentes, computação ubíqua, dentre outras (Cheng et al., 2009).

Há diferentes tipos de SAs e também diferentes níveis de adaptabilidade. Há aqueles que são inteiramente adaptativos e também há aqueles que possuem uma pequena parte que pode ser considerada adaptativa. Em geral, as características de adaptação são classificadas de acordo com seu objetivo e são conhecidas como auto-\*. As principais são descritas brevemente a seguir (Horn, 2001; IBM, 2006):

**Autoconfiguração:** capacidade do sistema mudar sua configuração dinamicamente em resposta a alterações no contexto, utilizando-se de políticas programadas previamente. Essas alterações podem ser a inserção ou remoção de componentes no sistema, ou até mesmo a mudança da localização em que o sistema é executado.

**Autocura:** capacidade do sistema descobrir, diagnosticar e reagir a interrupções. Assim, o sistema pode detectar um mau funcionamento e iniciar uma correção baseada em políticas previamente definidas sem a requerer a interrupção da sua execução.

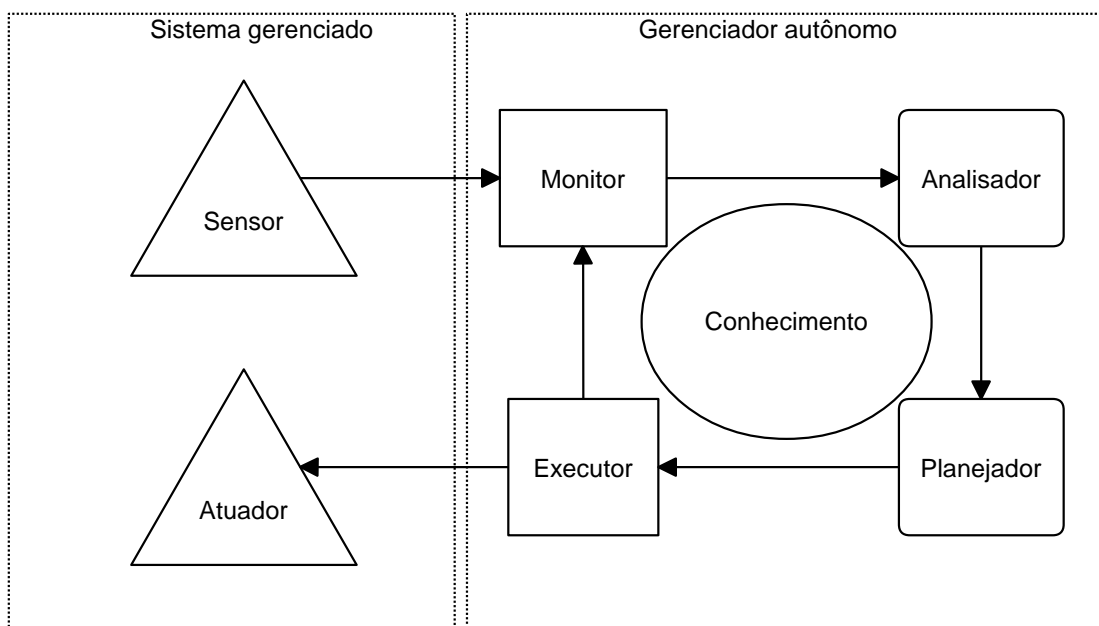
**Auto-otimização:** capacidade do sistema em se automonitorar e autoajustar dinamicamente para melhorar a qualidade do serviço oferecido utilizando os recursos da forma mais equilibrada possível. Portanto, a aplicação pode ajustar-se conforme a necessidade do usuário final, fornecendo assim um serviço de melhor qualidade sem que haja desperdício de recursos.

**Autoproteção:** capacidade do sistema antecipar, detectar, identificar e proteger-se contra possíveis ameaças. Ela pode detectar um comportamento hostil que quebre as políticas de segurança e tomar as devidas ações, tornando-se menos vulnerável.

A maioria dos autores que atuam na área de SAs reconhece que tais sistemas possuem duas partes conceitualmente distintas: o subsistema controlado e o subsistema controlador.

O controlado é o sistema base, responsável por fornecer os requisitos esperados pelo usuário. O controlador é o que faz o sistema ser considerado adaptativo, pois ele tem a responsabilidade de monitorar o sistema base e disparar as adaptações. Também há consenso de que o subsistema controlador é composto por loops de controle, que é um termo oriundo da teoria do controle (Garlan et al., 2004; Shaw, 1995; Weyns et al., 2013).

Um modelo teórico de loop de controle bastante utilizado é MAPE-K (Hurtado et al., 2011; IBM, 2006). A ideia principal é servir como modelo conceitual para a estruturação de SAs, prescrevendo os elementos conceituais que esses sistemas devem ter. Isso significa que um SA deve refletir esses conceitos de alguma forma em sua implementação. Na Figura 2.1 tem-se a representação desse modelo, em que a primeira parte — o *sistema gerenciado* — possui os sensores responsáveis por informar dados do sistema/ambiente e os atuadores responsáveis por interagir com o sistema. A segunda parte, denominada de *gerenciador autônomo*, representa o modelo de loop de controle MAPE-K. Como pode ser observado, existem quatro elementos básicos monitor, analisador, planejador e executor (MAPE) e um componente de conhecimento (K), que diz respeito à coleta dos dados que possibilita decisões baseadas nesse conhecimento.

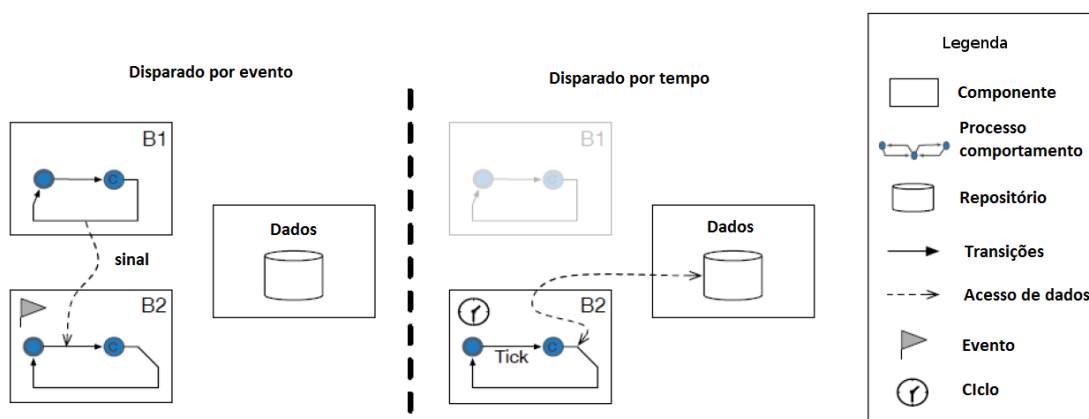


**Figura 2.1:** Loop de controle fechado MAPE-K – adaptada do trabalho de Hurtado et al. (2011).

Os quatro elementos básicos (MAPE) são detalhados a seguir: i) Monitor: responsável por prover o mecanismo que coleta, agrega, filtra e reporta os detalhes coletados dos recursos gerenciados; ii) Analisador: responsável por correlacionar e modelar as situações

complexas. Esse mecanismo permite que o gerenciador autônomo aprenda sobre o ambiente e ajude a prever situações futuras; iii) Planejador: responsável por construir as ações necessárias para atingir as metas e os objetivos determinados. O planejador utiliza-se de mecanismos e políticas para guiar o seu trabalho; e iv) Executor: responsável por controlar a execução do que foi planejado considerando as alterações dinâmicas.

Sistemas que utilizam o tempo ou eventos como maneiras de efetuar adaptações podem utilizar o loop de controle MAPE-K (Iglesia, 2014). Na Figura 2.2 ilustra-se como é realizada a comunicação entre componentes por meio de tempo e eventos.



**Figura 2.2:** Interação entre componentes de um sistema por meio de evento e ciclo de tempo – adaptada do trabalho de Iglesia (2014).

Na Figura 2.2 são ilustradas duas abordagens. Na primeira, em que se utiliza disparo por evento (lado esquerdo da figura), B1 envia um sinal para B2 que, por sua vez, procede de acordo com a chamada de B1, podendo iniciar o loop de controle MAPE-K. Na segunda abordagem, representada no lado direito da figura, utiliza-se disparo por tempo. Nessa abordagem, existe um temporizador responsável por realizar chamadas após um determinado tempo ter decorrido.

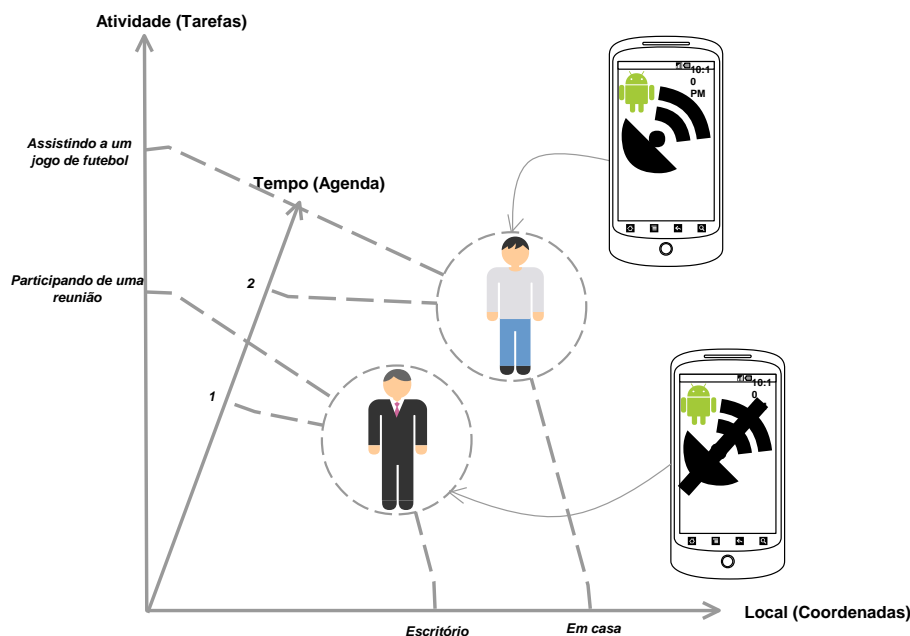
Observa-se que, então, ao lidar com um sistema que é composto por um sistema gerenciado e um Gerenciador autônomo outras propriedades podem ser encontradas. A presença de um Gerenciador autônomo, por exemplo, implica na procura pelos elementos do MAPE e, ainda, como mencionado por Iglesia (2014), uma interação baseada em tempo/eventos pode ser encontrada.

Na sequência apresenta-se um exemplo ilustrativo de um *Smartphone* equipado com Sensores, a fim de apresentar o conceito de Contextos em SAs.

## 2.2.2 Definições Utilizadas para Contexto

### Um Smartphone equipado com Sensores<sup>1</sup>

O sistema pode capturar de modo contínuo informações dos contextos do usuário, tais como *locais* e *atividades*, e usá-las como novas entradas para alternar entre os modos, por exemplo: (i) quando o usuário estiver assistindo a um jogo de futebol em casa, o sistema irá autoadaptar-se colocando o volume do *smartphone* no máximo; ou então (ii) quando o usuário estiver em uma reunião, o sistema irá autoadaptar-se colocando o *smartphone* no modo *vibrador*. Esses dois contextos mencionados são ilustrados na Figura 2.3.



**Figura 2.3:** Exemplo ilustrativo de um *Smartphone* Equipado com Sensores em dois contextos diferentes.

Assim como é ilustrado na Figura 2.3, como exemplo, define-se no sistema as *tarefas* que serão realizadas, a *agenda* de horários referentes às tarefas e as *coordenadas* referentes aos locais onde o usuário está. Assim que o sistema do *smartphone* reconhece o contexto, ele autoadapta-se para as respectivas configurações (por exemplo, *volume* ou *vibrador*).

Segundo Bartel et al. (2012), SAs decodificam o ambiente para dentro de uma abstração, chamada *contexto*. Uma aplicação *p* *ciente de contexto* (do inglês, *context-aware*) pode

<sup>1</sup>Para mais detalhes desse tipo de aplicação, no trabalho de Sama et al. (2010a) é desenvolvida uma aplicação, intitulada *PhoneAdapter*, que permite ao usuário definir as adaptações do sistema por meio da definição de regras de predicados (Ex: se o usuário estiver em reunião, o sistema irá autoadaptar-se e colocar o *smartphone* no modo *silencioso*).

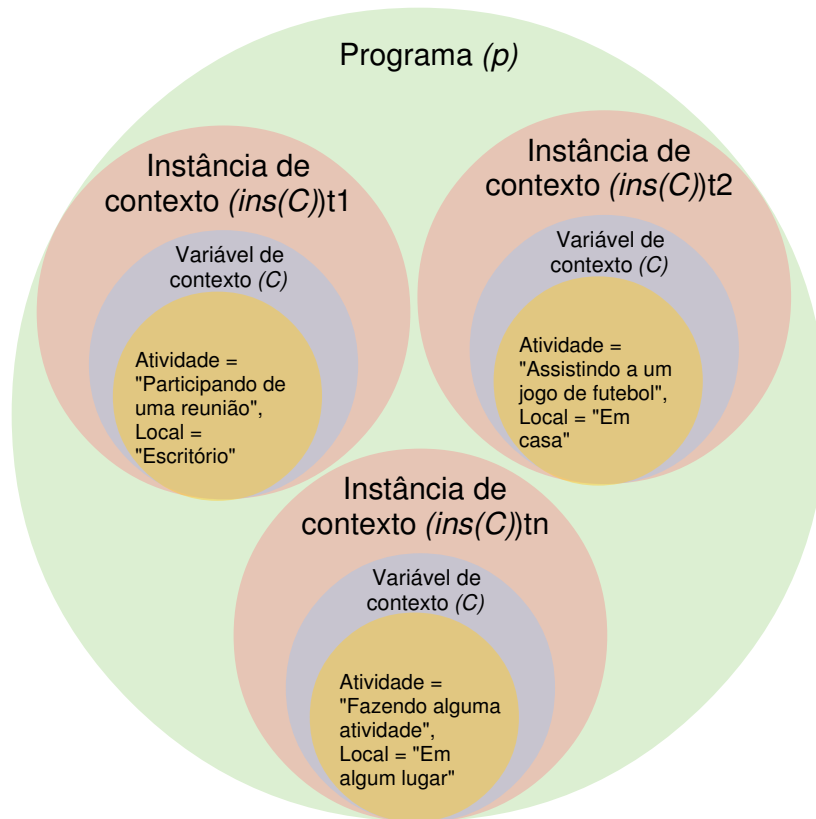
associar-se com um conjunto de atributos de ambiente, que podem ser classificados por *variáveis de contexto* (Lu et al., 2008). A seguir, são descritos os conceitos relacionados a contexto presentes em SAs, cada um seguido de um exemplo ilustrativo extraído do trabalho de Wang et al. (2014):

- **Variáveis de Contexto:** uma variável de contexto  $C$  é uma *tupla*  $(campo_1, campo_2, \dots, campo_n)$ , em que cada  $campo_i$  ( $i = 1, 2, \dots, n$ ) representa um atributo de ambiente subscripto por  $p$ . Variáveis de contexto são de natureza simbólica, quantificando os padrões ambientais aos quais  $p$  reage. É preciso que elas sejam inicializadas para que os contextos – execuções de  $p$  – utilizem-nas.
- **Instâncias de Contextos:** uma instância de contexto  $ins(C)$  de uma variável de contexto  $C$  é uma *tupla*  $(i_1, i_2, \dots, i_n)$ , tal que cada  $i_t$  ( $t = 1, 2, \dots, n$ ) leva a forma de  $(campo_t = valor : tipo, tempo)$ , tal que *valor*, *tipo* e *tempo* são: o valor da instância; o tipo; e o tempo de amostragem para  $campo_t$ , respectivamente. Por exemplo, na Figura 2.3, quando o usuário está *Assistindo a um jogo de futebol*, a variável de contexto é inicializada como uma instância de contexto com “Em casa” e “Assistindo a um jogo de futebol” para os campos Local e Atividade, respectivamente. Nesse exemplo, tem-se ainda o campo Tempo que funciona como uma agenda composta por horários, que representam os momentos em que cada contexto está.
- **Fragmento de Fluxo de Contexto:** uma sequência de  $ins(C)$ 's – Fragmento de Fluxo de Contexto ou  $cstream(C)$  – pode ser descrita como  $\langle ins(C)_{t_1}, ins(C)_{t_2}, \dots, ins(C)_{t_n} \rangle$ . Cada  $ins(C)_{t_i}$ , ( $i = 1, 2, \dots, n$  e  $t_i < t_{i+1}$ ) em  $cstream(C)$  é uma amostragem de instância de contexto no tempo  $t_i$  e todas instâncias de contexto em  $cstream(C)$  são ordenadas por  $t_i$ . Na Figura 2.4 é apresentado um exemplo de programa ( $p$ ) e suas respectivas instâncias.

Por exemplo, no *smartphone* ilustrativo da Figura 2.3, conforme o tempo passa, o fragmento do fluxo de contexto captura uma série de atividades:

1. (Participando de uma reunião, Escritório, 1);
2. (Assistindo a um jogo de futebol, Em casa, 2);
3. (Fazendo alguma atividade, Em algum lugar, n).

O primeiro campo é a Atividade, o segundo é o Local e o terceiro é o Tempo, respectivamente.



**Figura 2.4:** Exemplo de um programa ( $p$ ) e suas respectivas instâncias de contexto  $ins(C)$ 's.

- Predicado de Adaptação: Lu et al. (2006) definem uma situação como  $\langle ins(C), p, Act \rangle$ , na qual  $ins(C)$  é um conjunto de variáveis de contexto que a situação contém;  $p$  é uma condição de disparo de variáveis em  $ins(C)$ ; e  $Act$  é uma ação adaptativa para ser invocada pelo *middleware*<sup>2</sup> se  $p$  for igual a “verdadeiro”.

Ou seja, um Predicado de Adaptação – conhecido também como política de adaptação (Munoz e Baudry, 2009) – é dito satisfeito se  $p(ins(C) = true)$ ; caso contrário é dito como excepcional. Um outro exemplo, no sistema do *smartphone*, podem-se definir vários outros predicados de adaptação para o sistema autoadaptar-se como mostrado a seguir:

1. **SE** Local **IGUAL** *Universidade* **E** Atividade **IGUAL** *Participando de uma avaliação* **ENTÃO** Desligar o *smartphone*.

<sup>2</sup>Uma aplicação que se objetiva a ser uma ponte entre um sistema operacional, banco de dados ou outras aplicações (Stevenson, 2011). No trabalho de Lu et al. (2006), por exemplo, o *middleware* é uma parte responsável pela avaliação do contexto e disparo de adaptações.

2. **SE** Local **IGUAL** *Empresa* **E** Atividade **IGUAL** *Trabalhando* **ENTÃO** Colocar *smartphone* no silencioso.
3. **SE** Local **IGUAL** *Casa* **E** Atividade **IGUAL** *Cozinhando* **ENTÃO** Aumentar o volume do *smartphone* no máximo.

Ou seja, a adaptação dependerá das validações dos predicados de adaptação pré-definidos como políticas que são, de acordo com os paradigmas da computação autônoma (Klein et al., 2008), especificados por administradores do sistema.

- Diversidade de contexto: Segundo Wang et al. (2014), a diversidade do contexto pode ser mensurada por meio do número de alterações de contexto em um fragmento de fluxo de contexto (ou  $cstream(C)$ ). Computa-se, para cada  $cstream(C)$ , o total da distância de hamming (do inglês, HD - *Hamming Distance*)<sup>3</sup> entre todos os pares de instâncias de contextos, que se define pela equação 2.1:

$$\sum_{i=1}^{L-1} HD(ins(C)_i, ins(C)_{i+1}). \quad (2.1)$$

O  $HD(ins(C)_i, ins(C)_{i+1})$ , definido na equação 2.1, é a HD entre os pares das instâncias de contexto  $ins(C)_i$  e  $ins(C)_{i+1}$  para  $i = 1, 2, \dots, L-1$ , no qual  $L$  é o tamanho do fluxo do contexto. Por exemplo no *smartphone* de Wang et al. (2014), o fragmento de fluxo de contexto com a sequência é:

1. (sala de reunião, apresentação do relatório) <sub>$t=1$</sub> ;
2. (sala de reunião, discussão) <sub>$t=2$</sub> ;
3. (casa, assistindo ao futebol) <sub>$t=3$</sub> .

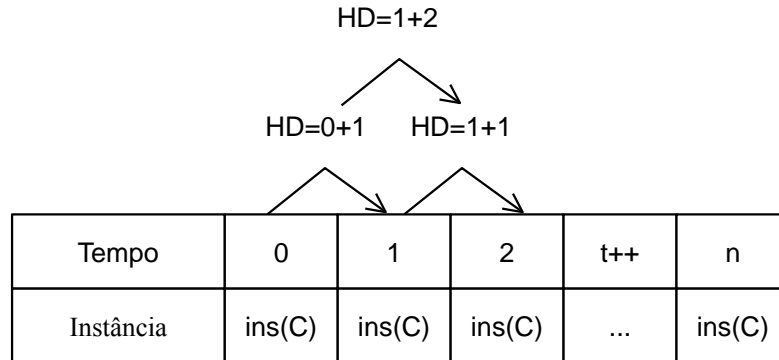
O cálculo da HD, ilustrado na Figura 2.5, apresenta que: de  $t=1$  para  $t=2$  a HD é “0 + 1 = 1”; e de  $t=2$  para  $t=3$  a HD é “1 + 1 = 2”. Logo a diversidade do contexto da sequência é dada pelo total da HD de **3** (Wang et al., 2014).

### 2.2.3 Propriedades de Sistemas Adaptativos

Neste trabalho, exploraram-se duas categorias de propriedades de SAs: as propriedades obrigatórias e as não-obrigatórias. Dentro da categoria das propriedades obrigatórias,

---

<sup>3</sup>A distância de Hamming foi originalmente proposta por Hamming (1950) para tuplas binárias, mas foi generalizada para cobrir tuplas da forma  $(campo_1, campo_2, \dots, campo_n)$  tais que os números dos valores possíveis em cada  $campo_i$  são finitos. O trabalho de Forney (1966) apresenta maiores detalhes da aplicação.



**Figura 2.5:** Exemplo ilustrativo do cálculo da HD.

tem-se o *gerenciador autônomo* e o *sistema gerenciado*. Estas duas propriedades são consideradas como fundamentais em um SA (Garlan et al., 2004; Shaw, 1995; Weyns et al., 2013). Dentro da categoria das propriedades não-obrigatórias, têm-se propriedades complementares às obrigatórias, de modo que, fazem o SA trabalhar dinamicamente (por exemplo, adicionar ou remover recursos em tempo de execução).

### As Propriedades Obrigatórias

Para o gerenciador autônomo, por exemplo, desempenhar sua função ele inclui loops de controle baseados no MAPE-K (Hurtado et al., 2011). Entretanto, assim como avaliado no trabalho de Ramirez e Cheng (2010), o componente Planejador na prática normalmente não é implementado (por se tratar de um componente que envolve características de aprendizagem e evolução dinâmicas). Por este motivo, a partir do MAPE-K, definiram-se somente como obrigatórias as propriedades *monitor* (M), *analizador* (A), *executor* (E) e *conhecimento* (K). Classificou-se a propriedade conhecimento (K) como estática ou dinâmica (ou como ambos), no qual o estático não permite que o usuário personalize o sistema e o dinâmico permite que o faça (Iglesia, 2014; Sama et al., 2010a). Além disso, pode haver a junção de ambos conhecimentos estático e dinâmico, de modo que o sistema permite ao usuário personalizar algumas funcionalidades e a outras não.

Para que o sistema gerenciado seja controlado pelo gerenciador autônomo, também são necessárias as propriedades *sensores* e *atuadores*. Sensores que disponibilizam, ao gerenciador autônomo, dados do sistema/ambiente e Atuadores que realizam as adaptações, no sistema, propriamente ditas (Iglesia, 2014).



E por fim, é fundamental que essas propriedades mencionadas necessitem interagir entre si. Para isso, mais duas propriedades obrigatórias foram definidas: *As interação baseada em tempo e/ou evento* (Iglesia, 2014), de modo que as outras propriedades podem utilizar uma, outra ou ambas propriedades (tempo e evento) para interagirem entre si.

Portanto, neste trabalho, os sistemas analisados que contemplaram todas as propriedades obrigatórias foram classificados como SAs, os demais foram descartados.

**Tabela 2.1:** Propriedades de sistemas adaptativos

ID	Propriedade	Obrigatória
1	<i>Multithreading</i> (gerenciador autônomo e sistema gerenciado)	x
2	Interação baseada em tempo	x
3	Interação baseada em eventos	x
4	Sensores	x
5	Atuadores	x
6	Conhecimento	x
7	Monitor	x
8	Analisador	x
9	Planejador	
10	Executor	x
11	Adiciona/remove recursos no tempo de execução	

### As Propriedades Não-obrigatórias

Dentro da categoria das propriedades não obrigatórias, a primeira delas é o *planejador*. Esta propriedade na prática, assim como destacado por Ramirez e Cheng (2010), normalmente não é implementada em SAs. Esta propriedade além de não ser trivial de se implementar, ela é bem restrita às funcionalidades de um domínio em particular (por exemplo pode envolver a previsões futuras e a um conhecimento amplo de configurações que um sistema deve estar apto a realizar).

Uma outra propriedade, mencionada por autores (Lalanda et al., 2013), é a possibilidade de um SA poder acoplar e desacoplar recursos (por exemplo componentes e sensores) no tempo de execução. Esta propriedade também foi definida a fim de identificar quais sistemas a fazem, entretanto, como uma propriedade não obrigatória.

Essas propriedades não obrigatórias foram definidas para auxiliar as obrigatórias no entendimento dos SAs. Ressaltando-se que são optativas, os sistemas que as tiverem também deverão ter as obrigatórias.

## 2.2.4 Um Exemplo Ilustrativo de um Sistema Adaptativo

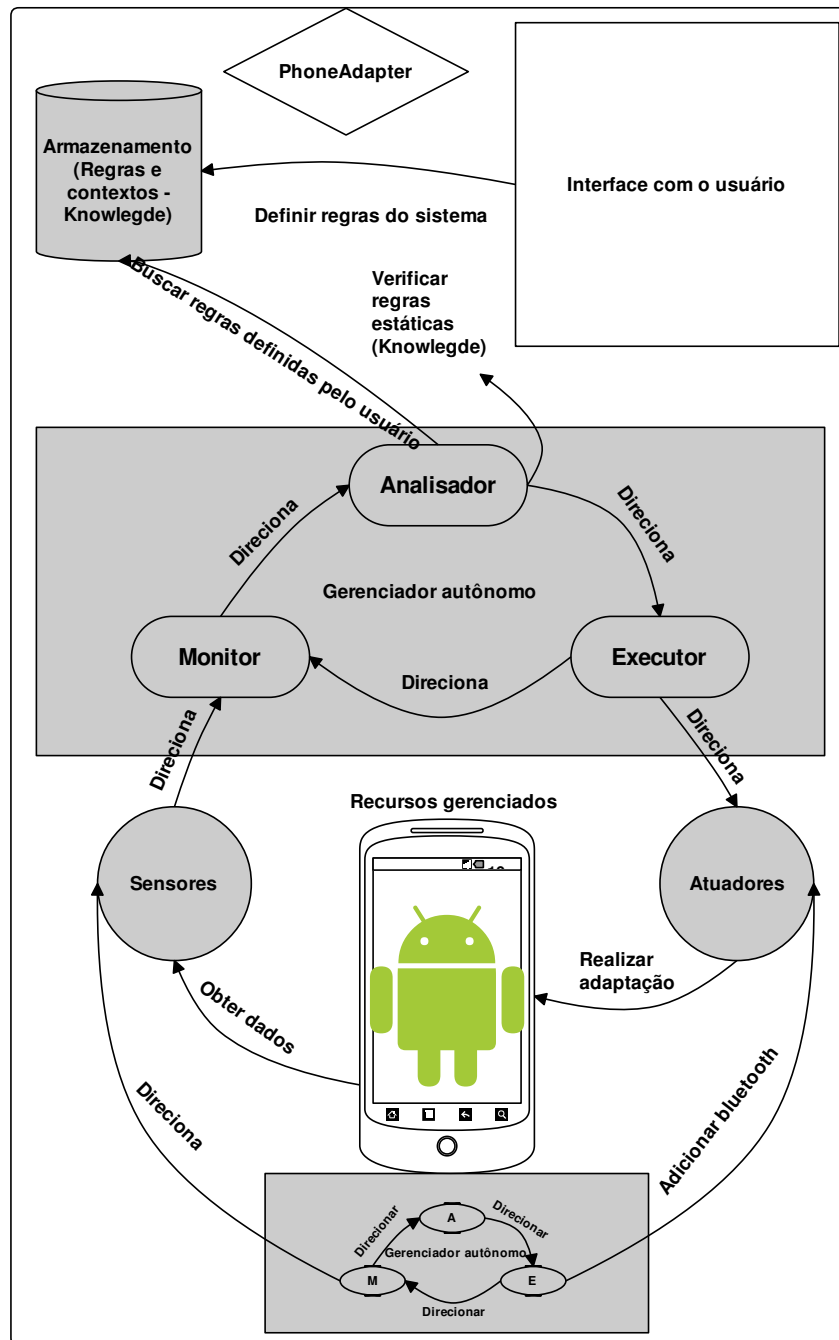
Aplicações móveis tornaram-se parte do dia-a-dia das pessoas não apenas para entretenimento, como jogos ou redes sociais, mas também para tarefas críticas, tais como transações bancárias (Van der Meulen e Rivera, 2014). A tecnologia móvel levou ao aumento do desenvolvimento de infraestruturas pervasivas que apoiam a “computação em qualquer hora e lugar” (Liu, 2013). O *Smartphone* ilustrado na seção anterior é um exemplo do que pode ser dito de um passo em direção a aplicações adaptativas cientes de contextos (ou sensíveis ao contexto) (Sama et al., 2008).

O *PhoneAdapter* (Liu, 2013) é um sistema dirigido por regras que pode se adaptar automaticamente, de acordo com as mudanças ambientais, para auxiliar o usuário em seu dia-a-dia. Assim, o usuário apenas precisará configurar as regras e seus possíveis perfis.

Durante a condução do estudo exploratório deste trabalho, apresentado no Capítulo 5, o *PhoneAdapter* foi um dos sistemas analisados a fim de mapear propriedades de SAs. Observa-se na Figura 2.6 uma possível representação da arquitetura abstraída do *PhoneAdapter*. Na parte superior da Figura apresenta-se a *interface com o usuário* que possibilita a personalização das regras e perfis (contextos) do sistema. No *PhoneAdapter* uma possível variável de contexto seria a “Casa” formada pelos atributos *localização* (latitude e longitude) ou determinado dispositivo *Bluetooth*. Uma possível regra seria “o sistema deve aumentar o áudio do *Smartphone* quando o local “CASA DO USUÁRIO” for detectado.“. Tais contextos e regras são armazenados em um banco de dados no *PhoneAdapter*, a fim de serem analisadas quando o SA estiver em execução.

Apesar de ser apenas um sistema prova de conceito, que pode ser instalado em qualquer dispositivo *android* a partir da versão 2.3, o *PhoneAdapter* ilustra bem a definição de um SA. Este sistema, assim como no *smartphone* ilustrado na seção anterior, obtém as variáveis de contexto do ambiente (por exemplo dados de GPS e de *bluetooth*), cria instâncias de contexto para tais variáveis, analisa as instâncias de contexto de acordo com as regras e executa as adaptações no *smartphone* em que o *PhoneAdapter* está instalado.

No meio da Figura 2.6 pode ser visualizado um retângulo cinza que se remete a um gerenciador autônomo. O modelo conceitual MAPE foi mapeado e, como pode ser observado, alguns dos elementos do MAPE estão presentes (monitor, analisador e executor). O gerenciador autônomo, por meio do monitor, obtém dados dos sensores do dispositivo móvel e encaminha para o analisador que analisa o contexto atual do sistema e as regras possíveis registradas. Ao identificar a regra atual, por meio dos predicados de adaptação, a adaptação é encaminhada pelo executor que agendará a mesma para os atuadores realizarem-na.

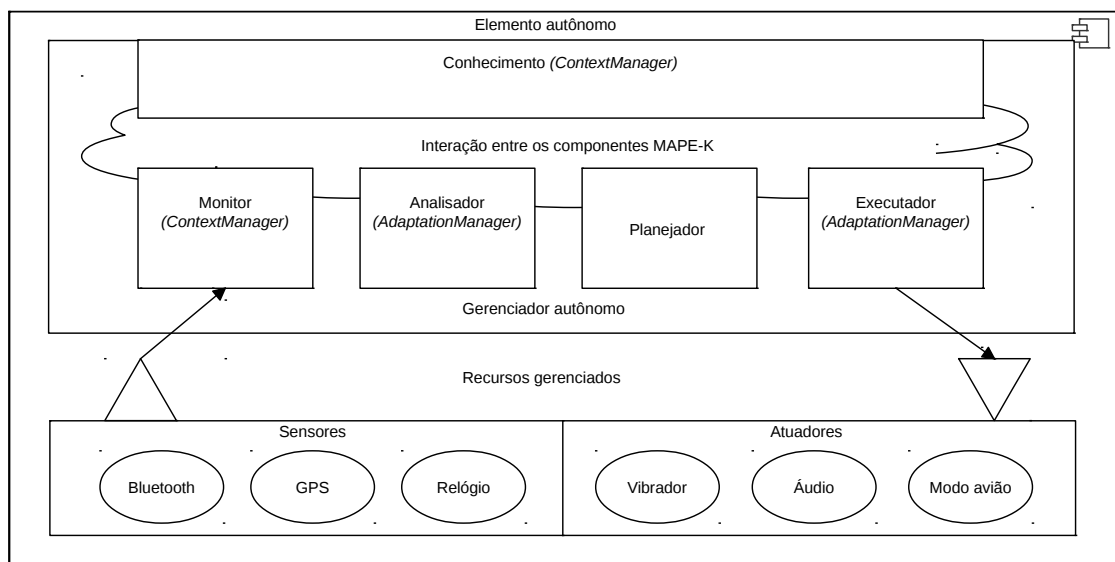


**Figura 2.6:** Exemplo ilustrativo do *PhoneAdapter* (Sama et al., 2010a).

Como pode ser observado na Figura 2.6 o sistema gerenciado, neste caso, são os próprios recursos do *smartphone*, denominados recursos gerenciados (Lalanda et al., 2013), a saber,

GPS, DataHora, *Bluetooth* (sensores) e Tipo de Toque, Volume, Vibrador e Modo Avião (atuadores). Os sensores são responsáveis por obter dados do ambiente e os atuadores são responsáveis por disparar adaptações. Além disso, um outro gerenciador autônomo pode ser visualizado na parte inferior da Figura 2.6. Neste caso, tal gerenciador autônomo explora os dispositivos *Bluetooth* que são pareados ao *smartphone*, a fim de adicionar novos e/ou excluir dispositivos *bluetooth*.

Com base no modelo conceitual MAPE-K e nas propriedades obrigatórias, mencionadas nas seções anteriores, pode-se mapear o *PhoneAdapter* com a Figura 2.7. Alguns autores introduzem o elemento autônomo, como aquele que contém um gerenciador autônomo e um sistema gerenciado (neste caso Recursos gerenciados).

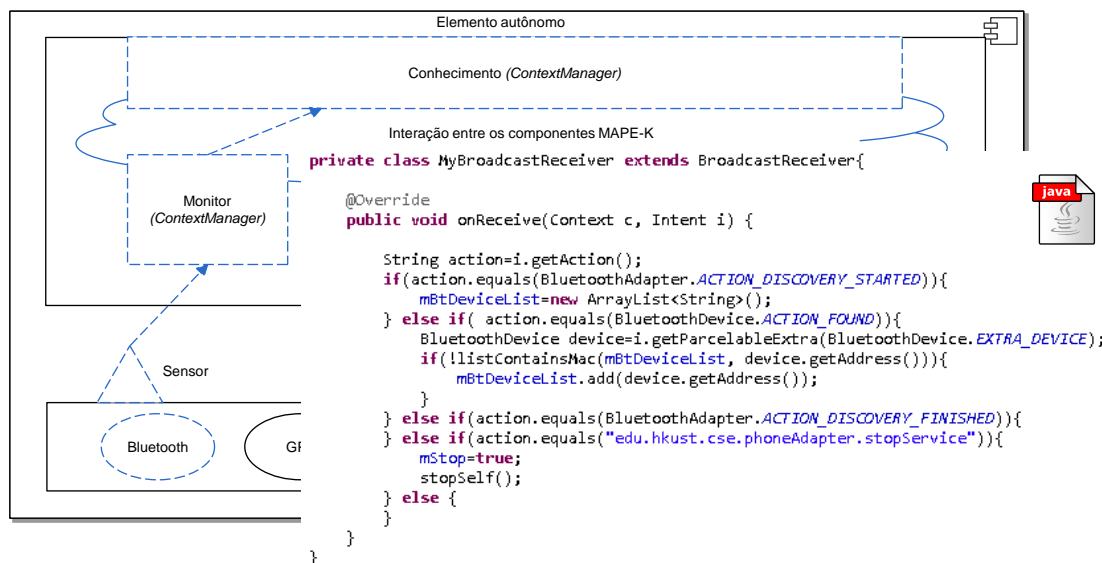


**Figura 2.7:** Mapeamento do código-fonte do *PhoneAdapter*.

Observa-se que o *PhoneAdapter* contém dois principais componentes: o ContextManager e o AdaptionManager.

Na Figura 2.8 ilustra-se um trecho de código-fonte do *PhoneAdapter* e qual a relação de tal trecho com o modelo mapeado. A relação é apresentada por meio das linhas tracejadas em azul. A classe do trecho herda de uma *BroadcastReceiver*<sup>4</sup> que fornece funcionalidades de recebimento de contextos e mensagens de execução de uma aplicação Android. No caso do trecho de código-fonte da Figura 2.8, manipulam-se dados vindos do sensor de

<sup>4</sup>ou apenas *Receiver*: é um componente *android* que permite registrar eventos que acontecem no *android* em tempo de execução <https://developer.android.com/reference/android/content/BroadcastReceiver.html>.



**Figura 2.8:** O comportamento do *PhoneAdapter* para obtenção de dados de sensores.

*Bluetooth*. Na Figura 2.9, apresenta-se o local onde é caracterizado o contexto corrente do *PhoneAdapter*, o qual podemos denominar componente conhecimento. Neste sentido, observa-se que os dados do contexto são centralizados no componente conhecimento que é utilizado por todos os outros componentes do MAPE. Na Figura 2.10 apresenta-se um trecho de código que ilustra um comportamento de um analisador. Neste trecho, por exemplo, realizam-se várias verificações a fim de elencar a “melhor” regra para o contexto corrente, ou seja, aquela que satisfaz o contexto corrente do usuário. Além disso, por fim, na Figura 2.11 apresenta-se um trecho de código ilustrando a adaptação realizada nos recursos gerenciados, funcionalidade denominada de um componente executor. Observa-se que, neste momento, as linhas do código-fonte encaminham (atuadores) mudanças aos recursos gerenciados respectivos. Ressaltando-se também que as linhas tracejadas em azul, na Figura 2.11, remetem-se aos componentes relacionados a tal adaptação.

Garantir a corretude desse tipo de sistema da “computação em qualquer hora e lugar” implica em uma série de desafios que são impostos à atividade de Teste (Ferrari et al., 2011), como caracterizados no Capítulo 4 deste trabalho. Assim, na próxima seção é realizada uma introdução ao teste de software.

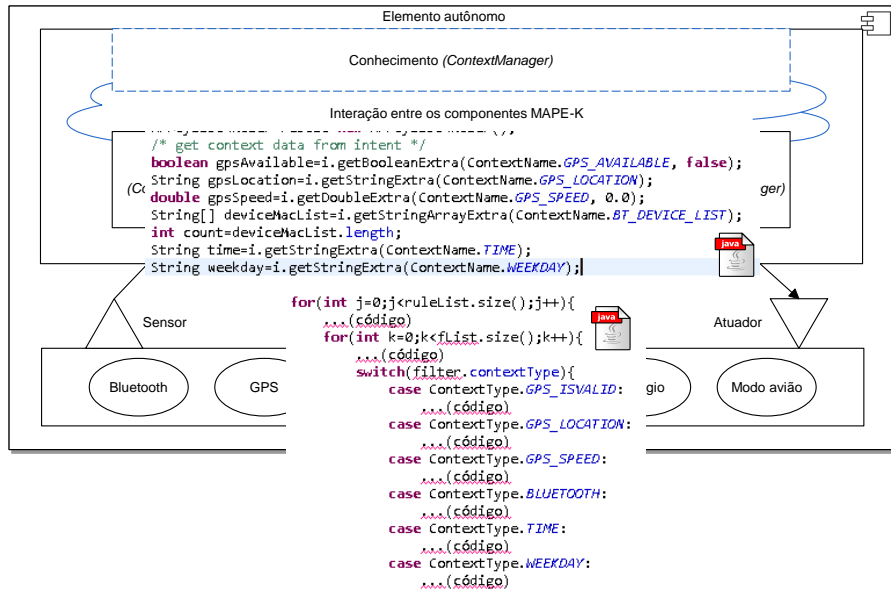


Figura 2.9: O comportamento do *PhoneAdapter* para o armazenamento de dados no tempo de execução.

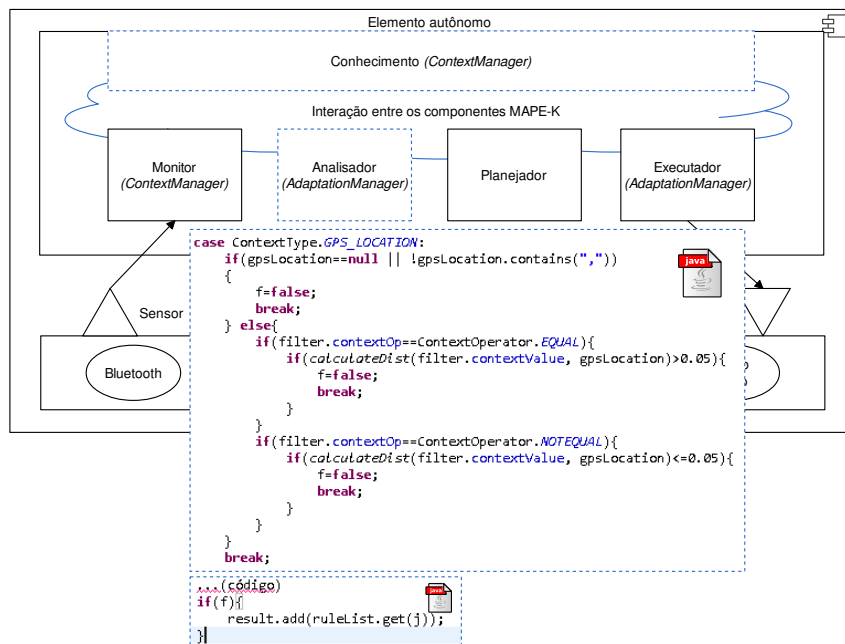


Figura 2.10: O comportamento do *PhoneAdapter* para a análise de dados no tempo de execução.

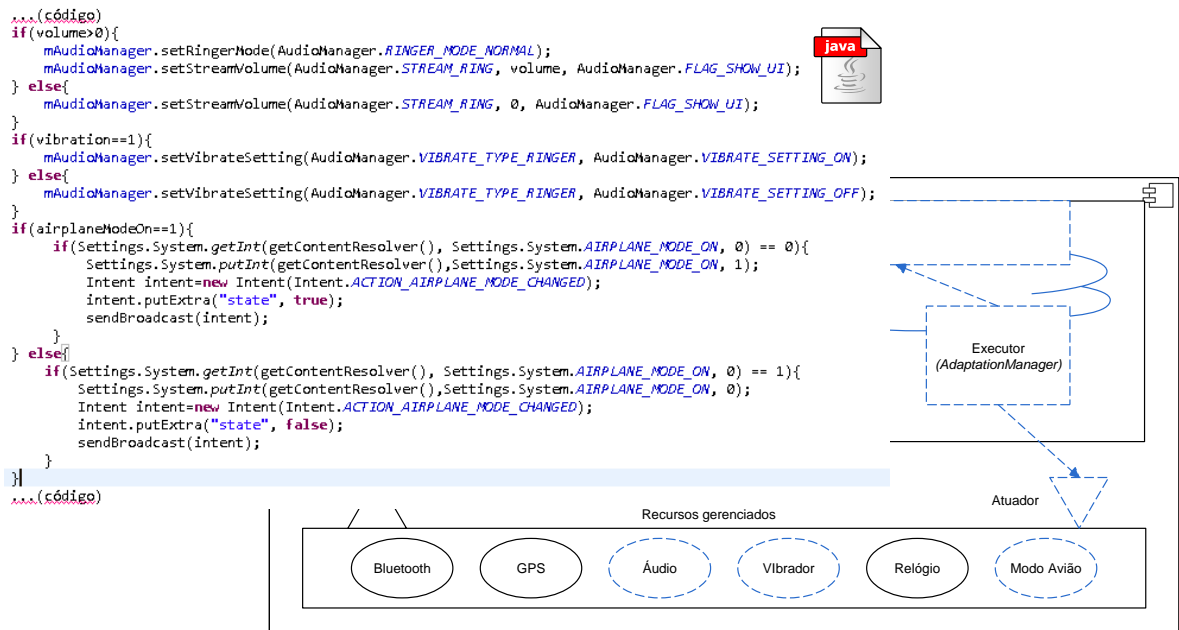


Figura 2.11: O comportamento do *PhoneAdapter* para a adaptação no tempo de execução.

## 2.3 Teste de Software

Na sequência são apresentados conceitos e terminologias sobre o teste; as principais técnicas e critérios de teste; e um processo genérico de teste de software.

### 2.3.1 Conceitos e Terminologias de Teste

A atividade de teste tem como objetivo primário de revelar defeitos no software (Myers et al., 2011). Em geral, executa-se o software em qualquer nível de abstração que assim o permita, com o uso de entradas específicas, e se verifica se o comportamento observado está de acordo com o esperado (Myers et al., 2011). Qualquer desacordo entre o comportamento esperado e o observado constitui uma falha. Toda falha é decorrente da execução de um defeito no software. A execução do defeito produz um estado inconsistente (ou seja, um erro), que pode ser externalizado e constituindo, assim, uma falha (IEEE, 1990). Ressalta-se que, embora o teste possa revelar defeitos, é impossível provar a inexistência de defeitos (Myers et al., 2011), pois para isso seria necessário executar o teste de forma exaustiva, ou seja, com todas as entradas possíveis (Delamaro et al., 2007). Apesar disso, o teste, quando realizado de maneira sistemática e criteriosa, pode contribuir para o

aumento da confiança das funcionalidades que o software desempenha (Harrold, 2000), mesmo que todas as possíveis entradas para sua execução não sejam fornecidas.

A atividade de teste é tradicionalmente dividida em três fases (Delamaro et al., 2007): (i) Teste de unidade, na qual o foco está nas menores unidades de um programa, que podem ser funções, procedimentos, métodos ou classes; (ii) Teste de integração, em que a ênfase é dada na construção da estrutura do sistema, à medida que as diversas partes do software são integradas; e (iii) Teste de sistema, que considera o software totalmente integrado e executando com a infraestrutura completa de hardware, software e pessoas. Além dessas fases tradicionais, o Teste de Regressão também é fundamental para identificar novos defeitos que porventura tenham sido introduzidos com as alterações realizadas do software. Ou seja, o teste de regressão visa a garantir que os novos requisitos são satisfeitos e que os requisitos anteriores continuam válidos. Embora esse tipo de teste possa ser utilizado durante o processo de desenvolvimento, o teste de regressão é tipicamente realizado durante a manutenção do código (Delamaro et al., 2007).

Um *caso de teste* pode ser representado por um par ordenado  $(d, S(d))$  tal que  $d$  é elemento de um domínio de entrada  $D$  e  $S(d)$  é a saída esperada para a entrada  $d$ . Entretanto, um dos principais problemas na atividade de teste diz respeito ao tamanho de  $D$ , que pode ser muito grande (ou até mesmo infinito) para algumas funcionalidades do software. Sendo assim, *critérios de teste* ajudam o testador a definir subconjuntos do domínio, o que reduz a quantidade de casos de teste que devem ser criados (Frankl e Weyuker, 2000).

Selecionar o domínio de entrada  $d$  de dados de teste, para “cobrir” as funcionalidades de um sistema, é uma das principais atividades do teste (Richardson et al., 1992) e, neste momento, um oráculo de teste ( $S(d)$ ) determina se o sistema está ou não se comportando corretamente. Neste sentido, cada critério de teste está relacionado a uma *técnica de teste* específica. As principais técnicas investigadas e empregadas são: funcional; estrutural; baseada em modelos e baseada em defeitos. A diferença entre as técnicas é o tipo de informação utilizada para estabelecer os subdomínios (Delamaro et al., 2007). Por exemplo, o teste funcional baseia-se na especificação do software em teste. O teste baseado em defeitos, por outro lado, baseia-se em defeitos comumente inseridos no software durante sua construção.

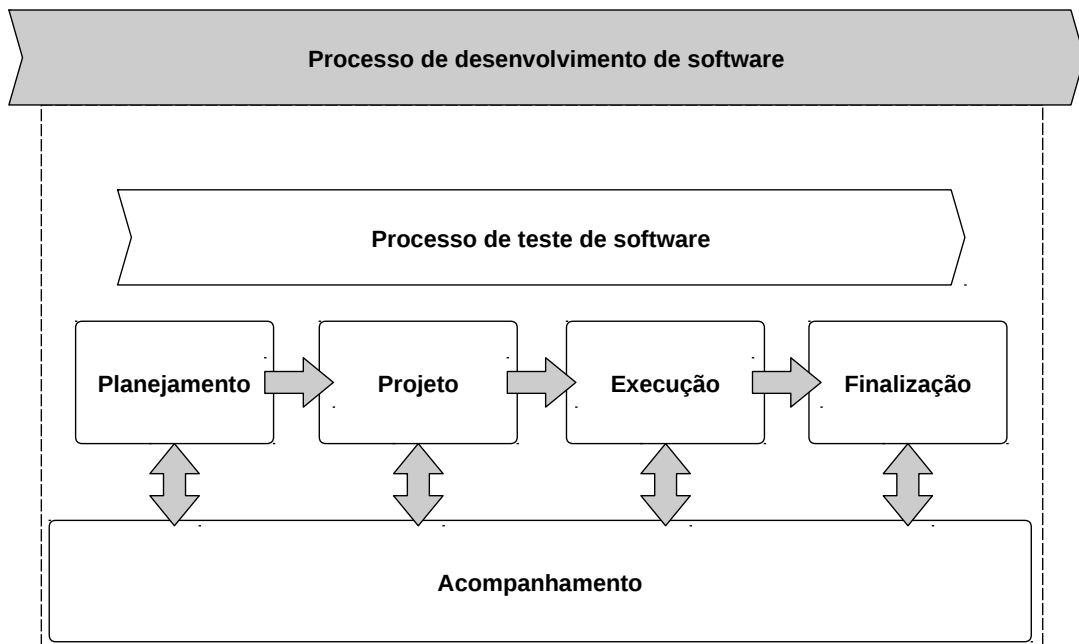
### 2.3.2 Processo de Teste de Software

Na seção anterior ressaltou-se a necessidade de se conduzir o teste durante todo o desenvolvimento do software, de modo que existem diferentes perspectivas durante o seu ciclo de



vida. Portanto, as fases e as técnicas de teste devem ser aplicadas sistematicamente, sendo acompanhadas, medidas e melhoradas (Mette e Hass, 2008), então, um processo de teste de software deve ser definido. Deste modo, questões como escopo, esforço, prazo e alocação devem ser avaliadas no processo, pessoas envolvidas devem ser orientadas corretamente quanto as suas funções e devem possuir pleno conhecimento das etapas do processo.

Um processo genérico de teste de software inclui as seguintes fases: *planejamento*, *projeto*, *execução*, *acompanhamento* e *finalização* (Camargo et al., 2013). Essas fases são ilustradas na Figura 2.12 e são descritas na sequência. Observa-se que cada uma das fases é composta por um conjunto de atividades.



**Figura 2.12:** Processo de Teste de Software adaptado do trabalho de Höhn (2011).

1. *Planejamento*: Nesta fase é definida a estratégia de teste, a qual inclui atividades que definem: 1) o que deve ser testado; 2) o que deve ficar fora do teste; 3) qual abordagem de teste deve ser seguida; 4) quais riscos devem ser considerados; 5) qual equipe deve participar; 6) quanto tempo deve ser dedicado; e 7) quais as tarefas a serem realizadas.
2. *Projeto*: Nesta fase são criados os casos de teste, incluindo atividades que definem: 1) quais técnicas de teste e como as funcionalidades serão cobertas; 2) refinamento das abordagens de teste; 3) especificação dos casos de teste; 4) revisão dos requisitos de ambiente; e 5) especificação do procedimento de teste.

3. *Execução*: Nesta fase os casos de teste são executados. Ela inclui atividades para definir: 1) a aplicação dos casos de teste, seguindo os procedimentos de teste; e 2) o registro da manifestação de defeitos no software.
4. *Acompanhamento*: Nesta fase, sendo também relacionada a todas as outras fases, é realizado todo o acompanhamento da execução do processo de teste, contendo atividades que relatam: 1) casos de teste executados; 2) defeitos identificados; 3) defeitos em aberto; 4) defeitos em aberto, pertencentes à área de risco; 5) funcionalidades que foram cobertas pelo teste; 6) quanto o software já foi testado; e 7) informação se novos casos de teste são necessários.
5. *Finalização*: Nesta fase, informações e resultados são consolidados, analisados e compilados. Ela inclui atividades que relatam e/ou mensuram: 1) andamento do teste; 2) resultados obtidos; 3) desempenho da equipe; 4) métricas definidas no processo de teste; 5) recursos gastos; 6) tempo gasto na execução das atividades em relação ao planejado; e 7) lições aprendidas.

A evolução do processo de teste deve ser uma etapa constante a fim de alcançar melhores resultados em sua aplicação. Ressalta-se que além das cinco fases mencionadas levam-se em consideração três restrições 1) o tipo de software que será testado; 2) requisitos referentes à confiabilidade e segurança; e 3) disponibilidade de recursos (financeiros e humanos) (Höhn, 2011).

Neste trabalho, as atividades e as fases de teste foram utilizadas a fim de mapear quais as fases de teste que estão sendo encontradas na literatura, no contexto do teste de SAs.

## 2.4 Considerações Finais

Neste capítulo foram apresentados os conceitos de SAs e o modelo teórico MAPE-K. Além disso, foram introduzidas as definições de contextos utilizadas, seguidas das propriedades de SAs e de um exemplo ilustrativo do SA utilizado no estudo exploratório deste trabalho. O teste de software também foi abordado, por meio da apresentação dos principais conceitos e terminologias, e de um processo genérico de teste. Todos os conceitos e exemplos apresentados embasam as investigações e definições conduzidas neste trabalho de mestrado, que são apresentadas a partir do próximo capítulo.

---

# Uma Revisão Sistemática sobre Teste de Sistemas Adaptativos

---

---

## 3.1 Considerações Iniciais

No Capítulo 1 elencou-se a dificuldade de se testar um Sistema Adaptativo (SA), mencionando que existem desafios impostos à atividade de teste de tais sistemas. Além disso, naquele capítulo, destacou-se uma hipótese levantada no trabalho de Ferrari et al. (2011) a respeito de que *saber quais os possíveis desafios que podem ser enfrentados, durante a atividade do teste em SAs, pode ajudar na definição de uma estratégia de teste para esse tipo de sistema*. A fim de sustentar tal hipótese, uma das questões de pesquisa definida foi “QP1: Quais são os desafios impostos pelos SAs à atividade de teste?”.

No Capítulo 2 os dois principais temas deste trabalho, Sistemas Adaptativos (SAs) e Teste de Software, foram introduzidos contextualizando-se os principais conceitos e terminologias presentes na literatura para ambos os temas. Além disso, exemplos ilustrativos foram apresentados a fim de exemplificar o funcionamento de SAs, bem como algumas técnicas e critérios de teste de software.

Portanto, este capítulo visa a apresentar a etapa 1 da metodologia utilizada neste trabalho, qual seja: “*Atualização e extensão da Revisão Sistemática realizada por Ferrari et al. (2011)*”. Nessa etapa, reproduziu-se a RS realizada por Ferrari et al. (2011), sem

filtro por data, avaliando-se novos trabalhos encontrados e estendeu-se a RS, atualizando a estratégia de identificação de estudos com uso da técnica *Snowballing*.

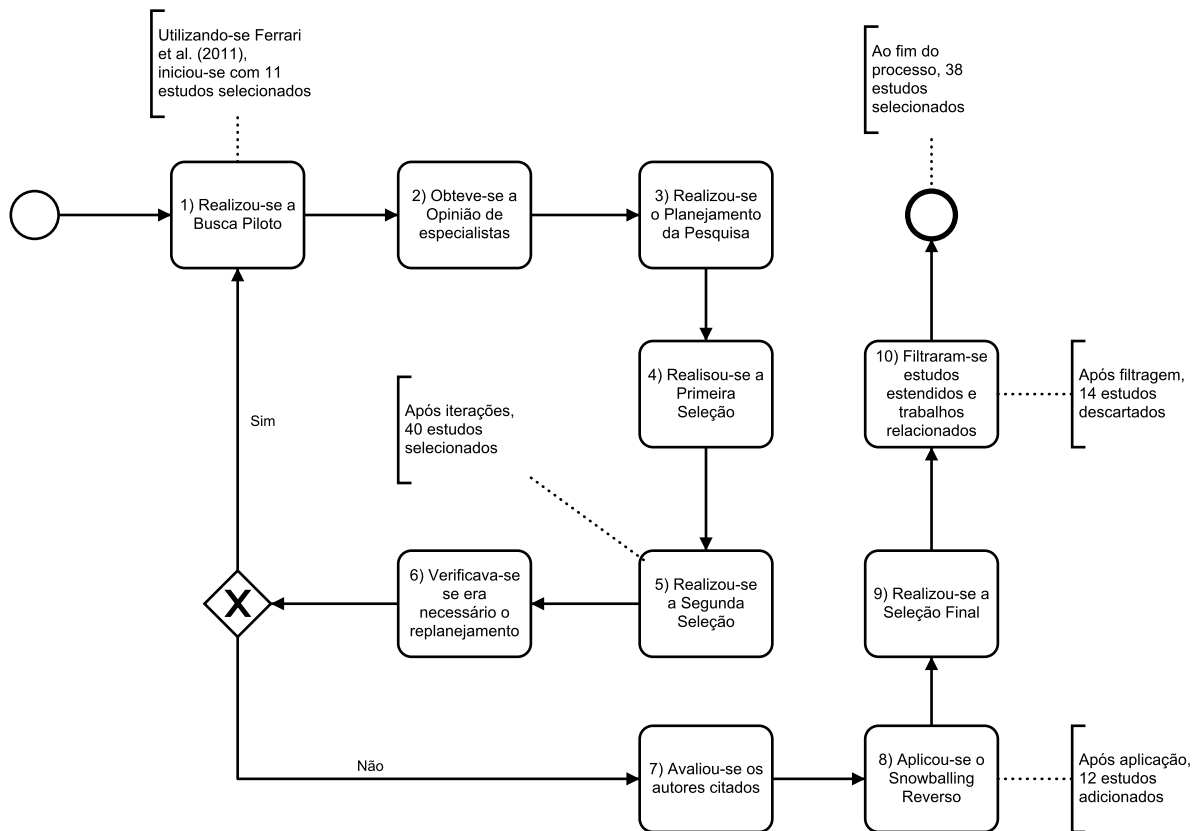
## 3.2 Processo de Pesquisa

Uma RS visa a identificar, a quantificar de maneira generalizada e, posteriormente, a aprofundar-se em uma área de pesquisa, de acordo com os termos utilizados e os resultados obtidos no processo da RS (Biolchini et al., 2007). Este processo é composto por quatro fases, denominadas: (i) Planejamento, (ii) Seleção Inicial, (iii) Seleção Final e Extração e (iv) Síntese (Fabbri et al., 2013). Na sequência a fase (i) Planejamento é apresentada na Seção 3.3, de modo que são apresentados o objetivo, as questões de pesquisa da RS, a estratégia de busca e os critérios para seleção de estudos. As fases (ii) Seleção Inicial e (iii) Seleção Final e Extração são apresentadas na Seção 3.4. Finalmente, a fase (iv) Síntese é apresentada nas seções 3.5 e 3.6.

A fim de refinar os estudos selecionados, filtraram-se os estudos que eram estendidos e propostos por trabalhos equivalentes, como sugerido no trabalho de Kitchenham (2004). Nesta etapa, identificaram-se aqueles estudos que tinham relação com outros estudos, verificando se algum estudo estendia a outro ou se algum estudo propunha a mesma abordagem que outro. Ao fim da etapa, refinou-se o conjunto de estudos de 52 estudos para 41 estudos. Além disso, descartaram-se três estudos da seleção final por não se tratarem de estudos primários (Ferrari et al., 2011; Matalonga et al., 2015a,b). Tais estudos foram classificados como trabalhos relacionados, os quais são abordados ao fim deste trabalho, de modo que, o conjunto final da seleção final compreendeu-se por 38 estudos primários.

O processo da RS envolveu algumas tarefas, desde o planejamento até a condução da pesquisa, que podem ser visualizadas na Figura 3.1, por meio de uma diagrama BPMN (*Business Process Model and Notation*) (Yamasathien e Vatanawood, 2014):

- Tarefa 1: Com uma busca piloto aplicaram-se as mesmas *strings* de busca utilizadas no trabalho de Ferrari et al. (2011);
- Tarefa 2: Com a opinião de especialistas adicionaram-se dois termos na *string* de busca (“autonomic” e “adaptive software”) e a máquina de busca *Web of Science*;
- Tarefas 3, 4, 5 e 6: Com o mesmo protocolo do trabalho de Ferrari et al. (2011) aplicaram-se as fases de seleção inicial e final, destacando-se o fluxo exclusivo para a abordagem iterativa de Fabbri et al. (2013).
- Tarefa 7: Com as inicial e seleção Final realizadas elencaram-se os autores citados;



**Figura 3.1:** Processo de pesquisa adaptado de Fabbri et al. (2013).

- Tarefa 8: Com os autores citados aplicou-se a técnica *snowballing* reverso com critério de seleção por autor, também com a inclusão da máquina de busca *Google Scholar*;
- Tarefas 9 e 10: Com o conjunto de estudos obtidos da seleção final e do *snowballing*, aplicaram-se um filtro por estudos estendidos e por estudos que propunham as mesmas abordagens.

Das tarefas de 1 até 6 utilizou-se uma abordagem iterativa (Fabbri et al., 2013) e das tarefas de 7 até 10 aplicou-se a técnica *snowballing* reverso (Wohlin, 2014). Além disso, visualiza-se a evolução do conjunto de estudos recuperados com a anotação de texto, ao lado das suas respectivas tarefas na Figura 3.1.

### 3.3 Planejamento da Pesquisa

No planejamento da pesquisa define-se um protocolo que tem por objetivo guiar todo o processo de pesquisa. Neste protocolo definem-se: objetivos, questões de pesquisa,

estratégia de busca, critérios de inclusão e exclusão, formulário para extração de dados e forma de análise de resultados (Biolchini et al., 2007).

O trabalho de Ferrari et al. (2011) subsidiou todo o planejamento da pesquisa deste trabalho. Assim, utilizaram-se os mesmos objetivos, questões de pesquisa e critérios de inclusão e exclusão do trabalho original. Por contraste, foram atualizados a estratégia de busca e o formulário de extração de dados. Todos os detalhes serão apresentados a seguir.

### 3.3.1 Objetivo de Pesquisa

Assim como no trabalho de Ferrari et al. (2011), neste trabalho objetivou-se a identificação de abordagens de teste de software para SAs (se necessário, com as devidas personalizações) e desafios correntes para testar adequadamente esse tipo de sistemas. Além disso, objetivou-se a identificar os tipos de defeitos de SAs descritos na literatura pesquisada.

### 3.3.2 Questões de Pesquisa

A fim de formular questões bem construídas, que podem ser respondidas mais facilmente, deve-se estruturá-las em partes (Nobre et al., 2003). Assim como no trabalho de Ferrari et al. (2011), as questões de pesquisa da RS, que permearam todo o trabalho, foram:

- **Questão Principal 01 (QP1):** Quais são as abordagens de teste de software existentes para apoiar o desenvolvimento de SAs?
- **Questão Secundária 01 (QS1):** Quais são os desafios para o teste de software no contexto de SAs?
- **Questão Secundária 02 (QS2):** Quais os tipos de defeitos de software no contexto de SAs?

Essas 3 questões de pesquisa compartilham as mesmas população, controle e aplicação.

- **População:** No contexto deste trabalho, observaram-se os estudos primários<sup>1</sup> que, no caso deste trabalho, descrevem o teste de software aplicado em SAs.
- **Controle:** No contexto deste trabalho, identificaram-se coleções de estudos primários por meio do estudo feito por Ferrari et al. (2011), todos relacionados a abordagens de teste de SAs. No entanto, destaca-se a não utilização de filtros ao aplicar este

---

<sup>1</sup>Um estudo primário relata uma pesquisa realizada em primeira mão (Greenhalgh, 2014, p.222), tal que o autor do estudo teve o contato direto com os resultados e contribuições da pesquisa em questão (ou seja, o respectivo autor foi o responsável pela coleta e análise dos resultados da pesquisa).

controle, isto é, todos os trabalhos encontrados – relevantes para a pesquisa – foram avaliados.

- **Aplicação:** No contexto deste trabalho, a aplicação incluiu desenvolvedores e pesquisadores que trabalham com SAs.

Essas três questões de pesquisa diferenciaram-se, umas das outras, em intervenção e resultados, isso porque se tratavam de focos diferentes; por exemplo, para a QP1 esperava-se responder “quais foram as abordagens encontradas”; já para a QS1 esperava-se responder “quais foram os desafios encontrados”.

### QP1

- **Intervenção:** No contexto deste trabalho, observaram-se as abordagens de teste de software propostas para SAs.
- **Resultados:** Para a QP1, no contexto deste trabalho, esperavam-se maneiras de se aplicar o teste de software em SAs. Isso incluiu: diretrizes, metodologias, técnicas, modelos, métricas, ferramentas e heurísticas.

### QS1

- **Intervenção:** Para a QS1, no contexto deste trabalho, observaram-se os desafios em se aplicar o teste de software em SAs.
- **Resultados:** Para a QS1, no contexto deste trabalho, esperavam-se os desafios em se aplicar o teste de software em SAs. Isso incluiu: desafios elencados, questões definidas e dificuldades encontradas pelos autores na condução de seus estudos, ao aplicar o teste de software em SAs.

### QS2

- **Intervenção:** Para a QS2, no contexto deste trabalho, observaram-se possíveis defeitos, erros e falhas presentes em SAs.
- **Resultados:** Para a QS2, no contexto deste trabalho, esperavam-se caracterizações mencionadas pelos autores. Isso incluiu: taxonomias, modelos, tipos e classificação de defeito, erro e falha.

### 3.3.3 Estratégia de Busca Utilizada no Trabalho Original

Uma das etapas da definição do protocolo do trabalho original, isto é, de Ferrari et al. (2011), foi a definição das máquinas de busca de bibliotecas digitais e indexadores que foram utilizadas na condução da RS, presentes na tabela 3.1 cuja rodada é 2.

**Tabela 3.1:** Máquinas de busca utilizadas na RS

Engine	Url	Round
<i>IEEE Xplore</i>	<i>ieee.org/ieeexplore</i>	2
<i>ACM Digital library</i>	<i>dl.acm.org</i>	2
<i>SpringerLink</i>	<i>ink.springer.com</i>	2
<i>ScienceDirect – Elsevier</i>	<i>www.sciencedirect.com</i>	2
<i>ISI Web of Science</i>	<i>www.webofknowledge.com</i>	1
<i>Google scholar</i>	<i>scholar.google.com</i>	1

**Critério de seleção das fontes:** utilizaram-se máquinas de busca de bibliotecas digitais e indexadores que continham e/ou recuperavam estudos relevantes na área de pesquisa.

**Métodos de busca nas fontes:** definiram-se e refinaram-se uma *string* base de acordo com a padronização e sintaxe de cada máquina de busca, para que fosse possível a utilização de todas as bases de dados selecionadas, presentes na Tabela 3.1.

**Tipo dos estudos primários:** Levaram-se em consideração estudos publicados em periódicos e anais de eventos científicos, relatórios técnicos e trabalhos em andamento.

**Idioma dos Estudos:** O idioma alvo para pesquisa dos estudos foi o inglês, por ser o idioma mais utilizado na literatura científica.

**Palavras-chaves:** Utilizaram-se as seguintes palavras-chaves, com suas respectivas variações: “*testing*”; “*adaptive system*”; “*adaptive systems*”; “*context-aware*” e “*context aware*”.

**Filtro por data:** no trabalho de Ferrari et al. (2011), assim como neste trabalho, não se utilizaram filtro por data, isto é, aplicaram-se as *strings* com o objetivo de avaliar todos os estudos que foram recuperados, independentemente do período de publicação. Tal medida foi tomada a fim de não descartar possíveis estudos primários que são indexados tardiamente, pelos indexadores de máquinas de busca, após o seu período de publicação (Kitchenham e Brereton, 2013).

Para a definição da *string* de busca, ressalta-se que foram realizadas adaptações das *strings* para cada máquina de busca, considerando trechos específicos dos estudos primários, isto é título, palavras-chaves e resumo.



### 3.3.4 Estratégia de Busca Utilizada Neste Trabalho

No trabalho de Fabbri et al. (2013) propôs-se um processo iterativo para se conduzir uma pesquisa, no contexto de mapeamentos e revisões sistemáticas. Neste processo a pesquisa é conduzida de modo que, sempre que necessário, retorna-se a uma atividade anterior. O retorno pode acontecer quando, durante a pesquisa, o pesquisador se depara com situações em que se deve planejar novamente alguma tarefa em particular. Como exemplo, durante as fases de seleção inicial e final dos estudos primários, o autor pode encontrar termos não conhecidos no início da pesquisa. Isso pode, por exemplo, gerar um aprimoramento na *string* base do protocolo.

Durante a validação do protocolo, a opinião de outros pesquisadores pode contribuir para complementar o processo da pesquisa e, conseqüentemente, contribuir nos resultados da RS conduzida (Fabbri et al., 2013).

Com uso de um processo iterativo de pesquisa (Fabbri et al., 2013), utilizou-se o trabalho de Ferrari et al. (2011) como entrada a uma busca piloto. Realizou-se a aplicação das *strings* em máquinas de busca, de modo a observar os estudos recuperados, a fim de refinar a *string* de busca base (Kitchenham e Charters, 2007). Neste passo, a opinião de outros pesquisadores foi levada em consideração, de modo que, incluíram-se novos termos para a *string* e duas novas Máquinas de busca. Na Tabela 3.2 apresentam-se a *string* base utilizada no trabalho de Ferrari et al. (2011) e a *string* base utilizada neste trabalho.

**Tabela 3.2:** Evolução da *string* de busca.

Ferrari et al. (2011)	This work
<i>("testing") and ("adaptive system" or "adaptive systems" or "context-aware" or "context aware")</i>	<i>("testing") and ("adaptive system" or "adaptive systems" or "context-aware" or "context aware" or "adaptive software" or "autonomic")</i>

A inclusão do termo “Adaptive Software” originou-se da presença do termo em pesquisas realizadas na literatura cinzenta<sup>2</sup>. Além disso, o termo “autonomic” originou-se do elevado número de trabalhos recuperados na busca que tratavam do tema “Computação autônoma” (Horn, 2001).

<sup>2</sup> A Literatura Cinzenta é definida como “aquela que é produzida sobre todos os níveis governamentais, acadêmicos e industriais”, que pode ser obtida nos formatos impresso (por exemplo livros de bibliotecas físicas) e eletrônico (documentos da internet). Além disso, esta literatura não é controlada por publicadores comerciais (por exemplo ACM Digital library, IEEEExplore, ScienceDirect) – <http://greyliit.org/about> – Acessado em: Outubro de 2015.

Devido à inclusão de máquinas de busca, a Tabela 3.1 sumariza as máquinas de busca utilizadas no trabalho original (Ferrari et al., 2011) e neste trabalho. A coluna Rodada informa as rodadas em que determinada máquina de busca foi utilizada. A Rodada 2 remete-se à máquina de busca que foi utilizada em ambos os trabalhos de Ferrari et al. (2011) e este. A Rodada 1 remete-se à máquina de busca que foi utilizada apenas neste trabalho. Portanto, neste trabalho, adicionaram-se as máquinas *Web of Science* e *Google Scholar*

A inclusão da máquina de busca “Web of Science” originou-se de sugestões de outros pesquisadores, os quais participaram como revisores de trabalhos submetidos ao Simpósio Brasileiro de Engenharia de Software (SBES)<sup>3</sup>. Além disso, a inclusão da Máquina de Busca “Google Scholar” originou-se da aplicação da técnica complementar de busca *Snowballing*, mencionada na mais à frente nesta seção.

### 3.3.5 Critérios de Seleção Para Os Estudos

Definiram-se três critérios de inclusão (CIs) e cinco critérios de exclusão (CEs), de modo que, cada CI referiu-se a uma respectiva questão de pesquisa e de cada CI originou-se um respectivo CE. Além disso, criaram-se dois CEs extras que ajudaram na seleção de estudos. Tais CIs e CEs – que foram os mesmos utilizados no trabalho de Ferrari et al. (2011) – são apresentados a seguir:

- Critérios de Inclusão (CIs):
  - **CI-1 (PQ1)**: Tratar de técnicas de teste de software propostas para ou aplicadas em SAs.
  - **CI-2 (SQ1)**: Descrever e/ou discutir desafios para o teste de software no contexto de SAs.
  - **CI-3 (SQ2)**: Caracterizar tipos de defeitos de software no contexto de SAs.
- Critérios de Exclusão (CEs):
  - **CE-1 (PQ1)**: Não tratar de técnicas de teste de software propostas para ou aplicadas em SA.

---

<sup>3</sup><http://cbsoft.org/cbsoft2015/> – O Congresso Brasileiro de Software: Teoria e Prática (CBSoft) é um dos principais eventos realizado anualmente pela Sociedade Brasileira de Computação (SBC), com a intenção de promover e incentivar a troca de experiências entre as comunidades científica, acadêmica e profissional sobre as mais recentes pesquisas, tendências e inovações – práticas e teóricas – na área de software.

- **CE-2 (SQ1)**: Não descrever e/ou discutir desafios para o teste de software no contexto de SAs.
- **CE-3 (SQ2)**: Não caracterizar tipos de defeitos de software no contexto de SAs.
- **CE-4**: Estudos primários escritos em línguas diferentes do inglês;
- **CE-5**: Textos que sejam apenas prefácios ou introdução.

Os estudos que passaram em apenas um critério de inclusão foram incluídos no conjunto final, devido ao caráter complementar dos objetivos da pesquisa realizada.

## 3.4 Condução da Pesquisa

Ao decorrer de todo o processo de RS efetuam-se o empacotamento dos artefatos e a documentação dos passos realizados, o que possibilita uma futura auditoria e reprodução do processo (Fabbri et al., 2013). Assim, com base no processo iterativo definido no trabalho de Fabbri et al. (2013), na Condução da Pesquisa utilizada realizaram-se: 1) a Fase Seleção Inicial; 2) a Fase Seleção Final e Extração. Além disso, a fim de complementar a estratégia de busca, utilizaram-se a técnica de *Snowballing* Reverso (Wohlin, 2014), seguida de um filtro de estudos estendidos (Kitchenham, 2004).

Realizaram-se o planejamento e a condução da pesquisa entre o período de *Abril de 2014 e Setembro de 2015* (isto é 18 meses). Durante este período, evoluíram-se a estratégia de busca e o conjunto de estudos, subsidiados por opinião de especialistas e a técnica *Snowballing* como dito anteriormente.

### 3.4.1 Resultados da Aplicação de Strings

Ao aplicar as *strings* nas máquinas de busca obteve-se 2177 estudos, apresentados na Tabela 3.3:

**Tabela 3.3:** Número de estudos retornados por base de dados.

Base de dados	IEEE	ACM	Science Direct	Web of Science	Springer Link	Total
Nº de estudos	162	195	149	933	738	2177
Porcentagem	7,63%	8,96%	6,84%	42,86%	33,90%	100%

Desses 2177 estudos, 162 (7,63%) pertencem à IEEE, 195 (8,96%) à ACM, 149 (6,84%) à ScienceDirect, 933 (42,86%) à Web Of Science e 738 (33,90%) à Springer Link.

### 3.4.2 Resultados da Seleção Inicial

Na seleção inicial recuperaram-se estudos primários dos repositórios para que pudessem passar pela seleção inicial, na qual se analisaram certas partes dos estudos (Fabbri et al., 2013).

Ao avaliar o título e o resumo dos 2177 estudos obtidos na aplicação de *strings*, obteve-se, conforme mostrado na Tabela 3.4, os resultados da Seleção Inicial. Dos 127 estudos selecionados, 36 (22,22%) pertencem à IEEE, 39 (20,00%) à ACM, 8 (5,37%) à ScienceDirect, 18 (2,43%) à Springer Link e 26 (7,79%) à Web Of Science. Assim, os estudos selecionados representam 5,83 % dos estudos recuperados. Nota-se que na Tabela 3.4 apresenta-se também a quantidade de estudos que foram descartados na seleção inicial.

**Tabela 3.4:** Número de estudos selecionados e descartados na seleção inicial.

Bases de dados	Selecionados	Descartados
<b>IEEE</b>	36 (22,22 %)	126 (77,78 %)
<b>ACM</b>	39 (20,00 %)	156 (80,00 %)
<b>Science Direct</b>	8 (5,37 %)	141 (94,63 %)
<b>Springer Link</b>	18 (2,43 %)	720 (97,57 %)
<b>Web of Science</b>	26 (2,79 %)	907 (97,21 %)
<b>Total</b>	127 (5,83 %)	2050 (94,17 %)

### 3.4.3 Resultados da Seleção Final

Na seleção final efetuou-se a leitura completa dos estudos primários, com nova aplicação dos critérios de inclusão e exclusão. Nesta fase, efetuou-se também a extração de dados, que teve como objetivo armazenar as informações dos estudos que são relevantes para a pesquisa realizada (Fabbri et al., 2013).

Ao efetuar a leitura por completo dos 127 estudos obtidos na Seleção Inicial, obteve-se os resultados sumarizados na Tabela 3.5. Dos 40 estudos selecionados, 18 (50,00 %) pertencem à IEEE, 15 (38,46 %) à ACM, 1 (2,50 %) à ScienceDirect, 3 (16,67 %) à Springer Link e 3 (11,54 %) à Web Of Science. Assim, os estudos selecionados representam 31,50 % dos estudos da seleção inicial. Além disso, apresenta-se na Tabela 3.5 a quantidade de estudos que foram descartados na seleção final.

Observa-se pelas tabelas 3.4 e 3.5 que houve uma grande quantidade de estudos descartados. Na sequência apresentam-se os principais motivos a respeito da quantidade de estudos descartados.

**Tabela 3.5:** Número de estudos selecionados e descartados na seleção final.

Bases de dados	Selecionados	Descartados
<b>IEEE</b>	18 (50,00 %)	18 (50,00 %)
<b>ACM</b>	15 (38,46 %)	24 (61,54 %)
<b>Science Direct</b>	1 (12,50 %)	7 (87,50 %)
<b>Springer Link</b>	3 (16,67 %)	15 (83,33 %)
<b>Web of Science</b>	3 (11,54 %)	23 (88,46 %)
<b>Total</b>	40 (31,50 %)	87 (68,50 %)

Dos 40 estudos selecionados na seleção final, 39 (97,50%) atenderam ao critério **CI-1**, 21 (52,50%) ao critério **CI-2** e 10 (25,00%) ao critério **CI-3**. Cada estudo atende a um ou mais critérios de inclusão. Sendo assim, quase todos os estudos (97,50%) propuseram ou decorreram sobre abordagens de teste para SAs e metade dos estudos (52,50%) elencaram desafios no teste desses sistemas. Além disso, 1/4 (25,00%) dos estudos mencionaram defeitos desses sistemas.

Descartaram-se os estudos que não contemplaram nenhum dos critérios CI-1; CI-2; e CI-3, ou ainda quando foram escritos em línguas diferentes do inglês ou textos que eram apenas prefácios ou introdução.

### 3.4.4 Estudos Publicados Pelos Autores Citados

Neste trabalho, classificaram-se os autores presentes em mais de um dos estudos selecionados na seleção final. Observa-se na Tabela C.1 – no Apêndice C – que 35 autores se enquadraram como citados.

Uma vez classificados quais autores eram citados, apresentados na Tabela C.1 no Apêndice C, utilizaram-se esses autores como critérios de seleção para a técnica *Snowballing* Reverso. Assim, os estudos selecionados na seleção final foram novamente retomados, a fim de avaliar suas listas de referências procurando por estudos em que esses 35 autores estavam presentes.

Ao aplicar o *snowballing* reverso por autor nos 40 estudos da seleção final, obteve-se um conjunto de 16 novos estudos. Nesta etapa, optou-se pela inclusão da máquina de busca *Google Scholar*, como mencionado anteriormente.

Avaliaram-se títulos e resumos desses 16 estudos, assim como realizado na fase seleção inicial. Desta avaliação, 12 estudos foram selecionados para a seleção final e, assim, realizou-se a leitura por completo de tais estudos. Desses 12 estudos, 10 (83,33%) atenderam ao critério **CI-1**, 7 (58,33%) ao critério **CI-2** e 1 (8,33%) ao critério **CI-3**. Um estudo pode atender a mais de um CI.

## 3.5 Os Estudos Seleccionados

Após realizado o processo ilustrado na Figura 3.1, obtiveram-se os resultados apresentados das tabelas 3.6 e 3.7. Na Tabela 3.6 apresentam-se os primeiros 20 estudos seleccionados e na Tabela 3.7 apresentam-se os últimos 18 estudos restantes. Nessas colunas das tabelas, apresentam-se: 1) um identificador (ID) para cada estudo seleccionados, de 1 até 38; 2) os autores de cada estudo; 3) o ano de publicação dos estudos; 4) a Rodada, sendo 2 para os estudos seleccionados no trabalho de Ferrari et al. (2011) e 1 para os estudos seleccionados neste trabalho; 5) a fonte onde o estudo estava disponível, mencionando-se se obteve o estudo por meio do *Snowballing*; 6) as técnicas de teste abordadas, sendo elas Funcional, Estrutural, Baseada em defeitos, Baseada em modelos; 7) os estudos em que seus autores mencionam algum desafio no teste de SAs; 8) os estudos que caracterizavam algum tipo de defeito em SAs; 9) o tipo de estudo conduzido para avaliar abordagem; e, por fim, 10) estudos que foram incluídos (ou seja, sobrepostos) pelos estudos listados nas tabelas.

**Tabela 3.6:** Conjunto de estudos da seleção final.

ID	Autor	Ano	Rodada	Fonte	Técnica	Desafios	Defeitos	Avaliação	Sobreposição
S01	Kephart e Chess	2003	2	IEEE (Snowballing)		x		Exemplo ilustrativo	
S02	Flores et al.	2004	1	IEEE	Funcional e baseada em Modelos			Exemplo ilustrativo	
S03	Tse et al.	2004	2	IEEE (Snowballing)	Funcional	x		Exemplo ilustrativo	
S04	Chan et al.	2006	2	World Scientific (Snowballing)	Outras			Exemplo ilustrativo	Chan et al. (2005)
S05	Lu et al.	2006	1	ACM	Estrutural	x		Estudo exploratório	
S06	Merdes et al.	2006	2	ACM (Snowballing)	Funcional			Estudo exploratório	
S07	King et al.	2007b	2	IEEE	Funcional; Estrutural e Baseado em defeitos	x		Estudo exploratório	
S08	Lu	2007	2	IEEE (Snowballing)		x		Exemplo ilustrativo	
S09	Niebuhr e Rausch	2007	1	ACM	Funcional	x		Exemplo ilustrativo	
S10	Wang et al.	2007	1	IEEE	Estrutural	x		Exemplo ilustrativo	
S11	Jaw et al.	2008	2	IEEE	Baseada em Modelos	x		Estudo de caso	
S12	Lu et al.	2008	1	IEEE	Estrutural	x		Estudo exploratório	
S13	Sama et al.	2008	1	ACM	Baseada em Modelos		x	Estudo exploratório	
S14	Munoz e Baudry	2009	2	arXiv (Snowballing)	Baseada em defeitos	x		Quasi-Experimento	
S15	Niebuhr et al.	2009	1	IEEE	Funcional	x		Exemplo ilustrativo	Niebuhr e Rausch (2009a) Niebuhr e Rausch (2009b)
S16	Ye et al.	2009	2	ACM	Estrutural			Discussão	
S17	Sama et al.	2010a	1	IEEE	Baseada em Modelos	x	x	Estudo exploratório	
S18	Sama et al.	2010b	1	ScienceDirect			x	Meta-análise	
S19	Vassev et al.	2010	2	IEEE	Outras	x		Exemplo ilustrativo	
S20	Wang et al.	2010b	1	IEEE	Baseada em defeitos		x	Estudo exploratório	Wang et al. (2010a)
<b>Totais</b>					<i>Snowballing</i> = 6	17	13	4	4

Tabela 3.7: Conjunto de estudos da seleção final.

ID	Autor	Ano	Rodada	Fonte	Técnica	Desafios	Defeitos	Avaliação	Sobreposição
S21	Welsh e Sawyer	2010	1	IEEE	Baseada em Modelos	x		Estudo de caso	
S22	Akour et al.	2011	2	IEEE	Funcional e Estrutural			Estudo exploratório	
S23	Bartel et al.	2012	2	arXiv (Snowballing)	Baseada em defeitos		x	Estudo exploratório	Bartel et al. (2011)
S24	Garvin et al.	2011	2	ACM	Funcional			Estudo exploratório	
S25	King et al.	2011b	2	IEEE	Baseada em defeitos	x		Estudo exploratório	King et al. (2011a, 2007a, 2008); Ramirez et al. (2008); Stevens et al. (2007)
S26	Silva e Lemos	2011	2	ACM	Outras			Exemplo ilustrativo	
S27	Wotawa	2012	2	Springer Link	Baseada em Modelos	x		Discussão	
S28	Micskei et al.	2012	2	Springer Link	Baseada em Modelos	x		Exemplo ilustrativo	
S29	Püschel et al.	2012	2	Google Scholar (Snowballing)	Baseada em Modelos	x		Exemplo ilustrativo	
S30	Weyns	2012	2	ACM	Baseada em Modelos	x		Exemplo ilustrativo	
S31	Fredericks et al.	2013	2	IEEE	Outras	x		Exemplo ilustrativo	
S32	Püschel et al.	2013	2	ThinkMind (Snowballing)		x	x	Exemplo ilustrativo	
S33	Eberhardinger et al.	2014	2	Springer Link	Baseada em Modelos	x		Exemplo ilustrativo	
S34	Fredericks et al.	2014	2	IEEE	Outras			Quasi-Experimento	
S35	Griebe e Gruhn	2014	2	ACM	Baseada em Modelos	x		Estudo exploratório	
S36	Wang et al.	2014	2	ACM	Estrutural e Baseada em defeitos			Estudo exploratório	Wang e Chan (2009)
S37	Yu e Gao	2014	2	IEEE	Estrutural e Baseada em modelos	x	x	Estudo exploratório	
S38	Cámara et al.	2015	2	IEEE	Baseada em Defeitos	x	x	Estudo exploratório	Cámara et al. (2013, 2014)
<b>Totais</b>					<i>Snowballing</i> = 3	17	12	4	8



Ao todo, selecionaram-se 38 estudos para caracterização do estado da arte. Dos 38 estudos, 34 (87,90 %) contemplam o critério CI-1 (coluna Técnica); 25 (62,50 %) o CI-2 (coluna Desafios); e 8 (20,00 %) o CI-3 (coluna Defeitos). Desses 38, 10 foram recuperados por *snowballing* e 12 foram descartados por serem estendidos por outros estudos. Além disso, três estudos (Ferrari et al., 2011; Matalonga et al., 2015a,b) foram classificados como trabalhos relacionados, os quais são abordados ao fim deste trabalho.

Os detalhes extraídos de cada um dos estudos, bem como suas restrições, podem ser visualizados – no Apêndice C – nas Tabelas C.2, C.3, C.4 e C.5.

## 3.6 Dados Extraídos e Empacotamento

Na fase *síntese*, realiza-se a sumarização dos dados extraídos dos estudos (Fabbri et al., 2013). Assim, durante a etapa de extração de dados, extraem-se os dados a fim de subsidiar as respostas para responder as questões de pesquisas realizadas. Os resultados da sumarização são apresentados a seguir.

### 3.6.1 Abordagens de Teste de SAs e Técnicas Associadas

A primeira questão de pesquisa foi “Quais são as abordagens de teste de software existentes para apoiar o desenvolvimento de SAs?”.

Nas Tabelas 3.6 e 3.7 apresenta-se o conjunto de estudos resultantes da seleção final. Nestas Tabelas observam-se, por meio da coluna rotulada com “Técnica”, que dos 30 estudos que utilizaram alguma técnica de teste, em oito estava presente a técnica Funcional; em oito a Estrutural; em 13 a baseada em modelos; em sete a baseada em defeitos; e em cinco foi considerada mais de uma técnica na abordagem.

A fim de sumarizar o estado da arte sobre as técnicas de teste para SAs, na sequência apresenta-se uma visão geral das técnicas identificadas com a realização da RS.

#### Teste Funcional

Abordagens de teste que empregaram a técnica funcional envolveram a composição dinâmica de componentes por meio de testes embutidos (Merdes et al., 2006; Niebuhr e Rausch, 2007; Niebuhr et al., 2009) e a definição de relações que se devem aplicar aos casos de teste (Tse et al., 2004). Tais relações, chamadas de *relações metamórficas*, são utilizadas para estabelecer um relacionamento entre casos de testes a partir de uma restrição específica de domínio. Além disso, uma outra abordagem (Garvin et al., 2011)

envolve o gerenciamento de um histórico de falhas a fim de realizar previsões de futuras falhas que possam ocorrer em um sistema sob teste.

## **Teste Estrutural**

Abordagens de teste que empregaram a técnica estrutural envolveram o uso de situações referentes ao contexto do SA sob teste (Lu, 2007; Lu et al., 2006, 2008) e a geração automática de conjuntos de casos de teste (Wang et al., 2007). Em todas as abordagens citadas, critérios de teste baseados em fluxo de dados foram aplicados. Além disso, uma outra abordagem envolveu a definição de grafos capazes de representar cenários de um SA, ao analisar o código-fonte de um dado *middleware* (Ye et al., 2009). Nesta abordagem, trataram-se diferentes estratégias de se modelar um *middleware*, bem como a geração de casos de teste com base no respectivo *middleware*.

## **Teste Baseado em Modelos**

Abordagens de teste que empregaram a técnica baseada em modelos envolveram a definição de máquina de estados para a detecção de defeitos de SAs (Sama et al., 2010a, 2008), a geração de casos de teste com base em modelos propostos (Griebe e Gruhn, 2014; Jaw et al., 2008; Püschel et al., 2012; Wotawa, 2012) e a definição de oráculos a fim de diminuir a quantidade de casos de testes a serem gerados (Eberhardinger et al., 2014; Jaw et al., 2008; Micskei et al., 2012; Welsh e Sawyer, 2010). Além disso, uma outra abordagem empregou o teste caixa-preta baseado em um modelo comportamental do SA, derivado a partir da implementação (Weyns, 2012).

## **Teste Baseado em Defeitos**

Abordagens de teste que empregaram a técnica baseada em defeitos envolveram a estratégia de variações ambientais que podem revelar defeitos do SA sob teste (Cámara et al., 2015; King et al., 2011a,b; Munoz e Baudry, 2009; Wang et al., 2010a), a utilização de metamodelos para obter *operadores de mutação*, gerados a partir de políticas de adaptação do SA sob teste (Bartel et al., 2012), e a definição padrão de operadores de mutação que podem ser aplicados a componentes do SA. Estas abordagens tinham em comum a utilização do critério Semeadura de Defeitos (King et al., 2011a; Munoz e Baudry, 2009) e do critério Análise de Mutantes (Bartel et al., 2012; Cámara et al., 2015; Munoz e Baudry,

2009; Wang et al., 2010a).

## Técnicas Híbridas

Abordagens de teste que empregaram mais de uma técnica de teste envolveram a geração de casos de teste (Akour et al., 2011; Yu e Gao, 2014), o desenvolvimento de estratégias de testes embutidos (denominadas *autoteste*) para simulação de versões de um SA com e sem defeitos (King et al., 2007a), e uma abordagem para otimizar a seleção de testes, baseada em cobertura de código-fonte com informações sobre variáveis de contexto (Wang et al., 2014). Nessas abordagens, há a utilização em comum de critérios de teste estrutural (baseados em fluxos de dados) (Akour et al., 2011; Wang et al., 2014; Yu e Gao, 2014), critérios de teste funcional (análise de valor limite, aleatório e particionamento de equivalência) (Akour et al., 2011; King et al., 2007a), critérios de teste baseado em defeitos (semeadura de defeitos e análise de mutantes) (King et al., 2007a; Wang et al., 2014), e ainda a utilização do teste baseado em modelos de estados, com uso de bi grafos<sup>4</sup> (Yu e Gao, 2014). Além disso, uma outra abordagem empregou a técnica funcional com a definição de modelos abstratos para representar o contexto do SA sob teste (Flores et al., 2004).

## Outras técnicas

Outros trabalhos relacionados envolveram a criação de um *framework* de teste denominado MAPE-T<sup>5</sup> – baseado no MAPE-K. (Fredericks et al., 2013), a definição de uma estratégia para geração e a adaptação de casos de teste em tempo de execução (Chan et al., 2006; Fredericks et al., 2014; Vassev et al., 2010) e a definição de uma estratégia para desenvolver teste de integração para SAs (Silva e Lemos, 2011).

Ressalta-se que, mais à frente neste trabalho, essas abordagens são caracterizadas junto aos desafios e fases de teste.

### 3.6.2 Desafios para o Teste de SAs

A segunda questão de pesquisa foi “Quais são os desafios para o teste de software no contexto de SAs?”. A fim de responder a esta questão, destaca-se que dos estudos presentes

---

<sup>4</sup>Um bi grafo consiste de dois grafos, um grafo de local e um grafo de link. Esses são utilizados para simular uma locação e interconexão de relacionamento de contextos, respectivamente (Xu et al., 2011).

<sup>5</sup>do inglês, *monitoring, analyzing, planning, executing and testing* – monitor, analisador, planejador, executor e teste

nas tabelas 3.6 e 3.7, nos 25 estudos marcados com “x” na coluna rotulada com “Desafios”, seus respectivos autores mencionaram algum desafio imposto no teste de software aplicado em SAs.

Dentre as principais características que se remeteram aos desafios, destacaram-se: (i) “*Como antecipar todas as mudanças de contexto relevantes e quando podem impactar o comportamento de SAs*”; (ii) “*Como lidar com interferência de usuário na configuração do sistema*”; e (iii) “*Como mapear e/ou modelar todas as possíveis variações de contexto e gerar testes para as diferentes combinações*”. Respectivamente, as características (i) e (ii) enfatizaram a dificuldade de se testar um sistema em que não se sabe exatamente qual será o seu comportamento, uma vez que este comportamento pode ser não determinístico (Sama et al., 2010a) devido à presença inerente aos dados do ambiente que o SA monitora. Um sistema que permite ao usuário a “liberdade” de definir regras personalizadas, por exemplo, elencará um desafio em se criar uma técnica que teste comportamentos que podem ser os mais diversos possíveis. Além disso, a característica (iii) enfatiza que mesmo criando-se uma técnica que teste o sistema, como fazer para manter os testes atualizados e precisos se o comportamento que o sistema possa vir a desempenhar é variável.

Nota-se que alguns autores ressaltaram desafios após avaliarem a aplicação de determinada técnica de teste (Fredericks et al., 2013) ou, até mesmo, elencaram dificuldades ao avaliar a literatura junto a um domínio de aplicação de SA em particular (Lu et al., 2006, 2008). Todas essas dificuldades (desafios) levantadas, nas palavras dos próprios autores, podem ser visualizadas – no Apêndice C – nas Tabelas C.6 e C.7. Mais à frente neste trabalho, esses desafios são caracterizados junto às técnicas de teste encontradas durante a aplicação da RS. Detalhes dessa caracterização são apresentados no Capítulo 4 na Seção 4.5.

### **3.6.3 Tipos de Defeitos de Software no Contexto de SAs**

A terceira questão de pesquisa foi “Quais os tipos de defeitos de software no contexto de SAs?”. A fim de responder a esta questão, destaca-se que em cada um dos estudos marcados com “x” na coluna rotulada com “Defeitos” (Tabelas 3.6 e 3.7), seus respectivos autores utilizaram ou caracterizaram algum tipo de defeito de software específico de SAs. Identificaram-se tais defeitos nos estudos ao avaliar trabalhos com caracterizações, modelos, taxonomias ou avaliações de defeitos, erros e falhas no contexto de SAs.

Ressalta-se que alguns trabalhos avaliaram defeitos relacionados a máquinas de estados finitas (MEFs) (Sama et al., 2010a, 2008), analisando-se o comportamento do SA por meio da transição entre estados em uma MEF. Alguns autores trabalharam com defeitos

relacionados a SAs que usaram o MAPE-K como base (Püschel et al., 2013). Outros autores, por sua vez, trabalharam com defeitos relacionados a SAs baseados em *middlewares* e/ou componentes que podem ser acoplados e desacoplados (Bartel et al., 2012; Cámara et al., 2015; Wang et al., 2010a; Yu e Gao, 2014). Além disso, alguns autores fundamentaram que defeitos em SAs ocorrem devido a características baseadas em várias camadas (Sama et al., 2010b), de modo que, quando um sistema é desenvolvido em camadas estas podem ser interdependentes e uma pode não ter detalhes da outra. Mais detalhes sobre os defeitos encontrados podem ser visualizados – no Apêndice C – nas Tabelas C.8 e C.9.

## 3.7 Detalhamento de Algumas Abordagens de Teste

Nos próximos capítulos desta dissertação, a RS é retomada e são analisados os trabalhos dela recuperados. No Capítulo 5 são definidas duas estratégias de teste, denominadas T e T\*. Estas estratégias foram compostas pelas abordagens S03 (Tse et al., 2004), S05 (Lu et al., 2006) e S14 (Munoz e Baudry, 2009). Sendo assim, na sequência são descritas tais abordagens. Além disso, a abordagem S17 (Sama et al., 2010a) proveu uma máquina de estados para representar o comportamento de um SA. Tal máquina foi utilizada neste trabalho, com formalismos de máquina de *mealy*.

### 3.7.1 Abordagem S03 (Tse et al., 2004)

A abordagem S03 de Tse et al. (2004) consiste em aplicar o teste caixa preta, de modo que explosões de configurações internas são encapsuladas. Nela utilizam-se *relações metamórficas* (RMs), que se configuram como relações esperadas ou existentes sobre conjuntos de distintos dados de entrada e correspondentes valores de saída para múltiplas execuções do programa alvo.

Utilizam-se as RMs para a aplicação do teste metamórfico, que é uma estratégia de teste baseada em propriedades de relações metamórficas. Se um grupo de casos de teste e saídas correspondentes não satisfazem uma respectiva RM, então o programa sobre teste contém um defeito. Além disso, aplicam-se várias vezes os casos de teste permeando as RMs, a fim de encontrar defeitos.

Um *caso de teste* pode ser representado por um par ordenado  $(d, S(d))$  tal que  $d$  é elemento de um domínio de entrada  $D$  e  $S(d)$  é a saída esperada para a entrada  $d$  (Frankl e Weyuker, 2000). No caso de uma RM, deve-se definir um conjunto de  $N$  pares ordenados  $(d, S(d))$  relacionados, sendo  $N$  a quantidade de casos de teste relacionados com um domínio em particular. Neste sentido, o uso do teste metamórfico se dá pela execução de  $N$  casos

de teste (isto é uma relação metamórfica), em um sistema, com entradas  $d$  de  $D$  e saídas  $S(d)$ . Se uma das saídas  $S(d)$  não é a esperada, então o sistema contém um defeito.

As principais restrições para a abordagem são a não trivialidade da identificação das relações metamórficas e a utilização de um exemplo ilustrativo para avaliar a abordagem. Apesar disso, com a constante “evolução de casos de testes” que compõem relações metamórficas, tem-se a possibilidade de mitigar alguns desafios referentes a, por exemplo, “como realizar o teste em um SA que não tem fronteira clara”. Além disso, ao classificar o SA como caixa-preta permite-se testar os componentes do SA sem levar em consideração a estrutura interna dos componentes. Como um exemplo, um sistema composto por vários componentes poderia ser testado utilizando as interfaces entre tais componentes.

### 3.7.2 Abordagem S05 (Lu et al., 2006)

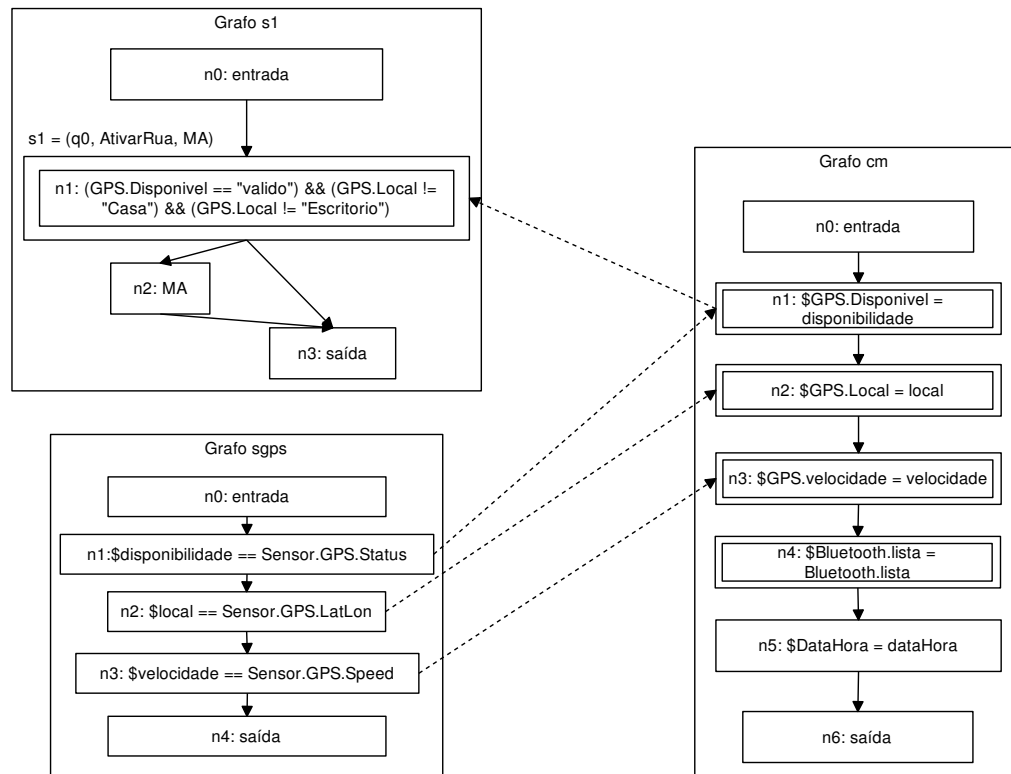
Na abordagem S05 (Lu et al., 2006), antes da aplicação de critérios para criação dos casos de teste, criam-se grafos de fluxo sensíveis ao contexto (*Context-aware Flow Graph* - CaFG), como os ilustrados na Figura 3.2. Com tais grafos é possível a aplicação de critérios de teste baseados em fluxo de controle e de dados, com informação adicional a respeito das associações existentes entre os fluxos de dados. Como exemplo, tem-se os fluxos de dados  $a$  e  $b$  sendo executados paralelamente. O fluxo  $a$  altera o valor da variável  $v$  também utilizada no fluxo  $b$ , desta forma, quando o fluxo  $b$  utilizar o valor de  $v$ , este estará inconsistente. O uso de CaFG permite identificar relações que um fluxo de dados pode ter com outro fluxo de dados. No exemplo mencionado dos fluxos  $a$  e  $b$ , pode-se utilizar um CaFG que destacaria qual nó de  $a$  está “associado” a qual nó em  $b$ .

Assim, tem-se como análise a verificação de definições de variáveis e seus usos nos demais CaFGs, levando em consideração que a variável definida em um determinado nó pode ser atualizada em outros nós.

Como um exemplo, um SA composto por *middlewares* com processos paralelos e/ou distribuídos pode conter valores nos fluxos de dados (de variáveis de contexto) desatualizados (Lu et al., 2006). Portanto, com uso de CaFGs podem-se mapear definições de variáveis que possam ter valores inconsistentes.

Na abordagem S05, de Lu et al. (2006), propuseram-se quatro critérios de teste estruturais, que são:

1. *All Situations*: criam-se casos de teste que permeiam as regras de transição entre contextos de um SA, assim todas as regras do SA devem ser abrangidas pela execução do conjunto de testes criado a partir desse critério.



**Figura 3.2:** Associações entre grafos utilizando a abordagem S05 (Lu et al., 2006).

2. *All Def-use Associations*: criam-se casos de teste que permeiam, além das regras de transição entre contextos de um SA, as associações entre os fluxos de dados. Essas associações podem ser visualizadas na Figura 3.2, na qual cada retângulo vertical significa um grafo de fluxo de dados, cada grafo é composto por retângulos horizontais que significam nós do fluxo de dados entre os nós, a seta direcional significa o fluxo, as setas tracejadas entre os grafos significam as associações entre os fluxos de dados que um SA pode ter e, por fim, os nós com retângulos duplos são aqueles que podem conter inconsistência de dados.

No caso da Figura 3.2, a execução de um nó *n1* presente no grafo *s1* pode conter inconsistência, devido ele poder ter sido alterado ao mesmo tempo no nó *n1* do grafo *cm* e sucessivamente do *n1* do grafo *sgps*. Este cenário é comum em SAs cujo desafio C-5 caracterizado está presente.

3. *All Pairwise DU Associations*: criam-se casos de teste a fim de abrangir aqueles nós que têm execuções cíclicas, isto é, se um nó *n1* é chamado por um nó *n2* e vice versa,

tem-se uma execução cíclica. Portanto, com este critério criam-se casos de teste que permeiam tais execuções.

4. *All Outstanding Situations*: criam-se casos de teste a partir da negação dos casos de teste criados pelo critério *All Situations*. Diferentemente do critério *All Situations* que testa como o SA deve se comportar em determinadas situações, o critério *All Outstanding Situations* deve testar como o SA *não* deve se comportar em determinadas situações. Como um exemplo, para a regra “AtivarRua” ser testada verifica-se se o usuário está “Na rua”. Por contraste, com o critério *All Outstanding Situations*, um caso de teste criado testaria a negação da regra “AtivarRua”, de modo a simular que se o usuário não estiver “Na rua” então a regra “AtivarRua” deve ser falsa, e se for falsa, então o caso de teste irá passar.

### 3.7.3 Abordagem S14 (Munoz e Baudry, 2009)

A abordagem S14 de Munoz e Baudry (2009) propõe uma estratégia para a seleção de variações ambientais que podem revelar defeitos nas regras do SA. Denominada teste de tabela de abalos artificiais (ASTT, *Artificial Shaking Table Testing*), a estratégia é uma técnica, do mesmo nome e largamente utilizada, da *engenharia civil* para avaliar resistências estruturais de construção, causados por eventos externos (Munoz e Baudry, 2009).

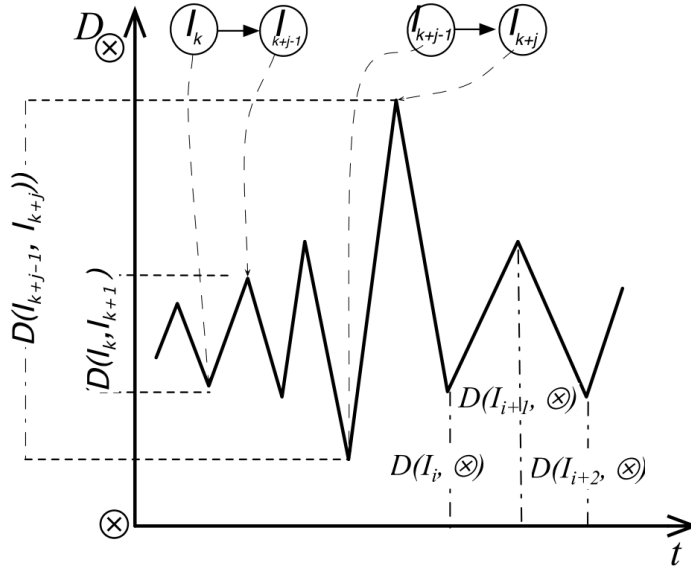
A ASTT faz uso de terremotos artificiais que simulam mudanças violentas nas condições ambientais, que estressam a capacidade de adaptação do sistema. Com o uso da abordagem, para se testar, identificam-se no SA as *variáveis de contexto*, *instâncias de contexto*, *fluxo de contexto* e *diversidade de contexto*, fundamentados no Capítulo 2.

Na Figura 3.3 ilustra-se uma função não linear que representa a evolução de uma instância de contexto no tempo (isto é o fluxo de contexto). Dada a função  $f(x) = z$ , na figura, o eixo das ordenadas ( $t$ ) representa o domínio tempo da função  $f$  e o eixo das abcissas ( $D$ ) a imagem distância da função  $f$ . Na figura a função  $D(I_k, I_{k+1})$  representa a distância entre a instância de contexto  $I_k$  e a outra  $I_{k+1}$ . Assim, para encontrar os terremotos artificiais, utilizam-se as equações 3.1 e 3.2:

$$\begin{aligned} & \exists ( I_k, I_{k+1}, \dots, I_{k+j} ) \subseteq f / \\ & ( D(I_k, I_{k+1}) \ll D(I_{k+j-1}, I_{k+j}) ) \vee \\ & ( D(I_k, I_{k+1}) \gg D(I_{k+j-1}, I_{k+j}) ), \quad k = 1..n \end{aligned} \tag{3.1}$$



$$\begin{aligned}
& \forall I_i, I_{i+1}, I_{i+2} \in f \\
& (D(I_i, \otimes) \geq D(I_{i+1}, \otimes) \wedge D(I_{i+1}, \otimes) \leq D(I_{i+2}, \otimes)) \vee \\
& (D(I_i, \otimes) \leq D(I_{i+1}, \otimes) \wedge D(I_{i+1}, \otimes) \geq D(I_{i+2}, \otimes)), \quad i = 1..n
\end{aligned} \tag{3.2}$$



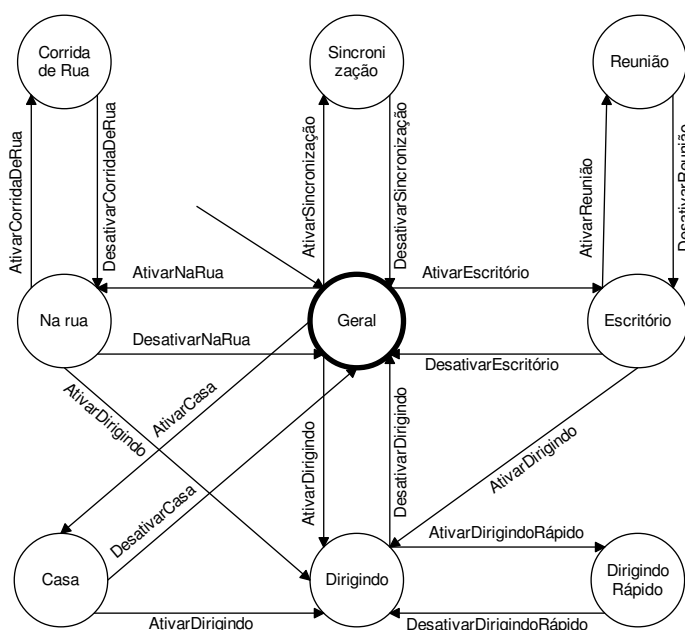
**Figura 3.3:** Um Fluxo de Contexto com um terremoto artificial (Munoz e Baudry, 2009).

Assim, um fluxo de contexto, como o presente na Figura 3.3, tem um terremoto artificial se ele tem as propriedades das equações 3.1 e 3.2. Para a equação 3.1, ele deve conter ao menos uma sequência de instâncias de contexto  $I_k, I_{k+1}, \dots, I_{k+j}$ , tal que a distância entre o primeiro par de elementos é diferente da distância entre o último par. Esta propriedade força um fluxo de contexto a conter variações violentas em transições de contexto. Além disso, para a equação 3.2, força-se um fluxo de contexto a conter instâncias de contexto cuja distância oscila, com relação à origem  $\otimes$ . Portanto, os fluxos de contextos que contém ambas as equações 3.1 e 3.2 aplicadas devem ser levados em consideração na criação de casos de teste, a fim de testar fluxos de contextos mais propícios a conterem defeitos.

O conceito de Diversidade de Contexto (Wang et al., 2014), fundamentado no Capítulo 2, pode ser visualizado claramente com a representação desses terremotos, de modo que dada uma função não linear  $f(x) = z$ , conforme o domínio  $x$  (tempo  $t$ ) evolui a imagem  $z$  (distância  $D$ ) se altera.

### 3.7.4 Abordagem S17 (Sama et al., 2010a)

Sama et al. (2010a) propuseram o uso de máquinas de estados finitas de adaptação (*Adaptation Finite-State Machine – A-FSM*) para se representar os possíveis estados de um SA, bem como as possíveis transições entre eles. Na abordagem, os estados são denominados *perfis* e as transições são denominadas *regras de adaptação*. Neste trabalho, assim como fundamentado no Capítulo 2, perfis foram denominados *instâncias de contexto* e *regras de adaptação* foram denominadas *predicados de adaptação*. Na Figura 3.4 ilustra-se a A-FSM adaptada de Sama et al. (2010a), ressaltando-se que cada perfil remete a uma determinada adaptação a ser realizada no SA modelado.



**Figura 3.4:** Máquina de estados finita de adaptação (*Adaptation Finite-State Machine – A-FSM*) adaptada de Sama et al. (2010a).

Observa-se, por meio da Figura 3.4, que tal exemplo ilustrativo contém nove estados (*instâncias de contexto*) e 16 transições (*predicados de adaptação*). Tendo-se o estado “Geral” como inicial e final, a A-FSM leva em consideração que, a todo instante, sensores (por exemplo, GPS e *Bluetooth*) obtêm dados de variáveis de contexto. A todo instante, também, verificam-se tais dados, de modo que, caso se satisfaça um *predicado de adaptação*, realiza-se uma transição entre os estados e, conseqüentemente, uma adaptação no SA. Como um exemplo, um usuário está no estado “Geral” e o sensor de GPS capturou a

localização do usuário. Durante a verificação do Predicado de Adaptação, verificou-se que a localização atual do usuário é a “Casa” do usuário. Assim, acontecerá uma transição do estado “Geral” para o estado “Casa”. O mesmo se repete para cada estado, ao se verificar os *predicados de adaptação*, levando-se em consideração as variáveis de contexto vindas dos sensores.

### 3.8 Considerações Finais

Neste capítulo foi relatada a etapa 1 da metodologia utilizada neste trabalho, sendo ela a “*atualização e extensão da RS realizada por Ferrari et al. (2011)*”. Nesta etapa reproduziu-se a RS realizada por Ferrari et al. (2011), sem filtro por data, avaliando-se novos trabalhos encontrados e estendeu-se a RS, atualizando a estratégia de identificação de estudos, com uso da técnica *Snowballing*. Assim, contribuiu-se com um primeiro passo em direção a responder à questão de pesquisa “QP1: Quais são os desafios impostos pelos SAs à atividade de teste?”.

Com a condução da RS obtiveram-se: 25 estudos que mencionam desafios para se testar SAs; 30 estudos em que se aplicam alguma abordagem de teste em SAs; e, por fim, oito estudos que caracterizaram defeitos de SAs.

No próximo capítulo apresenta-se o processo seguido em busca da resposta das questões de pesquisa, mais especificamente em responder a questão de pesquisa QP1 e dar um passo em direção à resposta para a questão “QP2: Quais são os desafios caracterizados encontrados em SAs reais?”.

---

# Caracterização de Desafios e Abordagens de Teste para Sistemas Adaptativos

---

---

## 4.1 Considerações Iniciais

No Capítulo 1 elencou-se a dificuldade de se testar um Sistema Adaptativo (SA), mencionando que existem desafios impostos à atividade de teste de tais sistemas. Além disso, naquele Capítulo, destacou-se uma hipótese levantada no trabalho de Ferrari et al. (2011) a respeito de que *Saber quais os possíveis desafios que podem ser enfrentados, durante a atividade do teste em SAs, pode ajudar na definição de uma estratégia de teste para esse tipo de sistema*. A fim de sustentar tal hipótese, três das questões de pesquisa definidas foram “QP1: Quais são os desafios impostos pelos SAs à atividade de teste?”; “QP2: Quais são os desafios caracterizados encontrados durante a atividade de teste em SAs reais?”; e “QP3: Quais são os defeitos encontrados com a aplicação das estratégias de teste baseadas em desafios?”.

Para a questão de pesquisa QP1, no Capítulo 3 apresentou-se a etapa 1 da metodologia utilizada neste trabalho – “*Atualização e extensão da RS realizada por Ferrari et al. (2011)*”.

Assim, visando a responder a QP2, neste capítulo apresenta-se a caracterização dos desafios que foram recuperados na RS, utilizando propriedades de SAs para avaliar sistemas. Além disso, em direção a responder a QP3, apresentam-se as análises das abordagens de teste recuperadas da RS, utilizando o critério de inclusão CI-1.

Portanto, este capítulo apresenta algumas das etapas da metodologia utilizada neste trabalho, sendo elas: (2) *Investigação de SAs disponíveis em repositórios de código-fonte na Web*; (3) *Caracterização de desafios*; (4) *Caracterização de abordagens e seu relacionamento com as fases de teste*; (5) *Análise de desafios e SAs*; (6) *Análise de desafios e abordagens de teste*; e (7) *Análise de desafios, abordagens e SAs*.

## 4.2 Os Desafios Específicos

Após a análise dos 25 estudos primários que abordaram desafios impostos ao teste das tabelas 3.6 e 3.7, extraiu-se uma lista com 47 desafios de teste de SAs – presentes no Apêndice C nas Tabelas C.6 e C.7. Ao realizar a leitura completa dos estudos, possibilitou-se o entendimento do problema, do contexto e da terminologia utilizada pelos autores originais dos estudos. Com isso, observou-se uma sobreposição entre os desafios, de modo que, um mesmo desafio era mencionado por vários autores em outras palavras ou contextos diferentes. Tal sobreposição resultou em uma lista com 34 desafios específicos presentes nas Tabelas 4.1 e 4.2.

Definiram-se esses desafios como desafios específicos (*Specific Challenges* - SCs), enumerando-os de *SC-1* até *SC-34*, uma vez que a maioria deles foi caracterizada de SAs que foram construídos de acordo com abordagens específicas de desenvolvimento (Fredericks et al., 2013; Kephart e Chess, 2003; Weyns, 2012), abordagens de teste (King et al., 2011b; Munoz e Baudry, 2009; Wang et al., 2007), ou tecnologias (Cámara et al., 2015; Niebuhr e Rausch, 2009b; Sama et al., 2010b). Por exemplo, Fredericks et al. (2013) apresentaram um *framework* de teste, denominado MAPE-T, que é baseado no MAPE-K. Para o uso da abordagem, implicaria-se que o SA fosse desenvolvido com base no MAPE-K. Como um outro exemplo, Sama et al. (2010b) apresentaram testes realizados em um SA que utilizou um dispositivo móvel.

Levando em consideração as similaridades das descrições de desafios encontrados, realizou-se um processo de análise de possíveis combinações. Tal análise considerou a descrição, o contexto e as similaridades entre os desafios identificados. Como um exemplo, considerando os desafios específicos *SC-32* e *SC-4*, ambos referem-se ao crescimento no número de configurações de SAs. Outro exemplo refere-se aos desafios *SC-2* e *SC-32*. Neste caso, referem-se à imprevisibilidade em SAs. Esses e os demais desafios podem ser

visualizados nas Tabelas 4.1 e 4.2. Na coluna Desafios estão os desafios agrupados que foram caracterizados para o seu respectivo SC. Na coluna SC está a identificação para o desafio específico que representa seu respectivo grupo de desafios. Na coluna Descrição apresenta-se uma breve descrição do desafio. Na coluna IDC apresenta-se a análise de similaridades entre os desafios específicos, resultando nos desafios genéricos que são descritos na próxima seção.

### 4.3 Os Desafios Genéricos

A Figura 4.1 sumariza os resultados a respeito da generalização entre os desafios específicos, presentes nas Tabelas 4.1 e 4.2. Definiu-se tal generalização como desafios genéricos, enumerados de C-1 a C-12.

Na Figura 4.1, observa-se que os círculos representam os desafios específicos, os retângulos representam os desafios genéricos (junto com suas pequenas descrições) e as linhas mostram as associações entre desafios específicos e genéricos.

De acordo com a Figura 4.1, dos oito desafios específicos (*SC-1*, *SC-4*, *SC-6*, *SC-9*, *SC-12*, *SC-18*, *SC-21* e *SC-22*) de Ferrari et al. (2011) definiram-se sete desafios genéricos (*C-1* – *C-7*). Estes são representados em fundo branco na Figura 4.1. Os outros, que são representados em fundo cinza, foram identificados e caracterizados neste trabalho.

Assim como mencionado na seção anterior, alguns desafios específicos são descrições em diferentes palavras, com diferentes contextos e em variados níveis de detalhes. Portanto, analisou-se não somente pequenas descrições de cada desafio, mas também foram considerados os contextos dos respectivos estudos primários após a leitura completa e extração de dados (descritas no Capítulo 3).

Em resumo, elencou-se uma lista de 12 desafios genéricos para o teste de SAs. Na sequência, com base na lista de desafios, apresentam-se detalhes de cada desafio específico, a fim de justificar suas associações uns com os outros. Ressalta-se que os SCs dos desafios específicos são mostrados dentro dos parênteses.

#### **C-1 – Como lidar com o crescimento exponencial de configurações de sistema que devem ser testadas**

Os autores Welsh e Sawyer (2010), Munoz e Baudry (2009), Tse et al. (2004) e Micskei et al. (2012) mencionam a dificuldade de lidar com todas possíveis combinações de “variações de contexto”, ao projetar casos de teste (*SC-1*). Fredericks et al. (2013) apresentam a questão de limitar (ou não) a habilidade do sistema adaptar-se (*SC-2*). Em ambos desafios está presente a dificuldade em se testar um SA que possa ter um

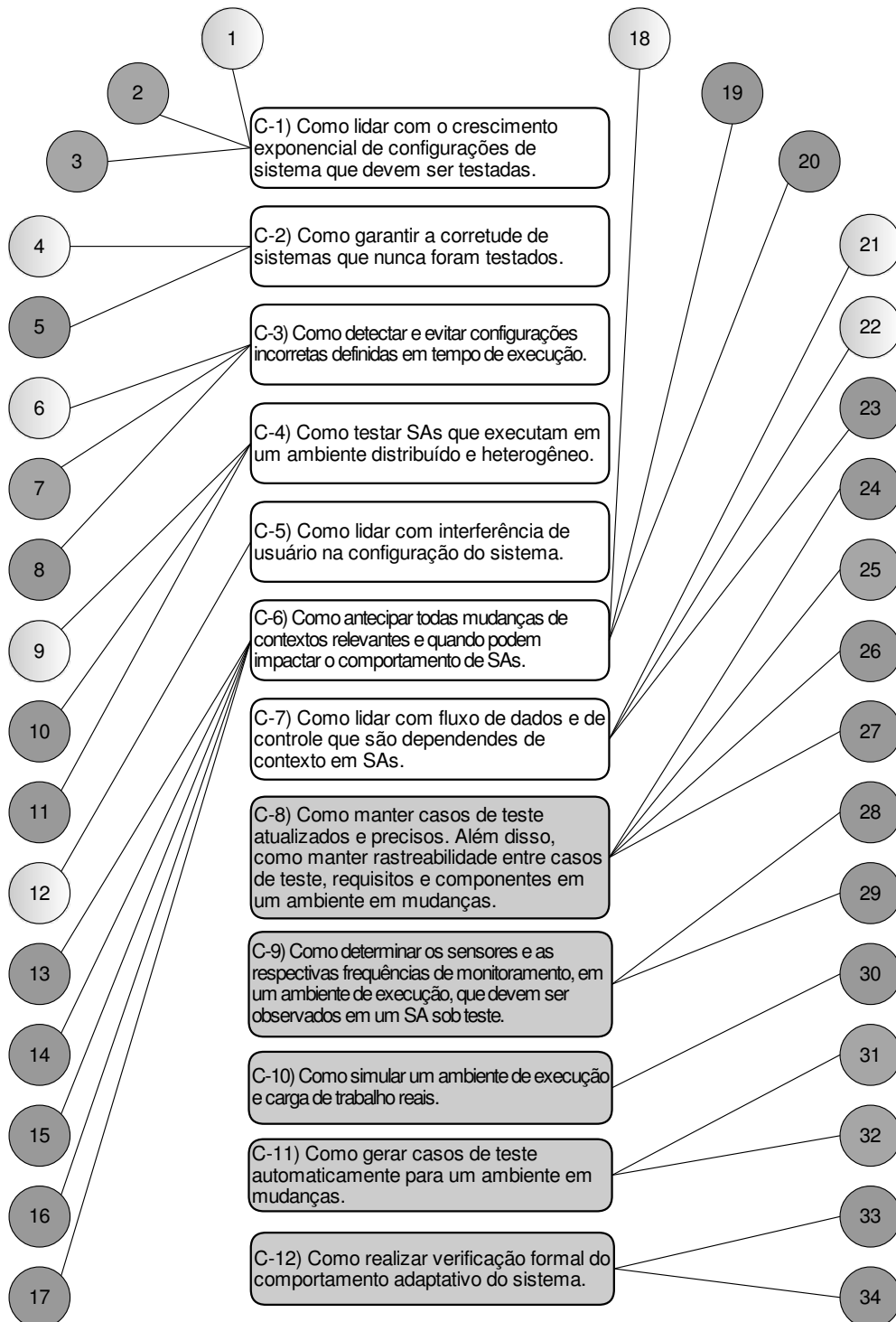


Figura 4.1: Desafios genéricos.

comportamento baseado em combinações de configurações. Além disso, um outro desafio (*SC-3*) identificado refere-se à questão de quando parar o teste, ao utilizar uma abordagem dirigida por modelos (Wotawa, 2012). Neste sentido, um SA que remova e/ou adicione dinamicamente componentes em tempo de execução acarretará em dificuldades referentes ao possível crescimento exponencial de configurações, durante a atividade de teste.

Observa-se que uma grande dificuldade encontra-se na tarefa de projetar um conjunto de teste, a fim de executar um número de possibilidades que pode mudar, devido à constante mudança de configuração e/ou estrutura do SA sob teste.

### **C-2 – Como garantir a corretude de sistemas que nunca foram testados**

Os autores Cámara et al. (2015), Niebuhr et al. (2009), Niebuhr e Rausch (2007) e Welsh e Sawyer (2010) mencionam que a dinamicidade de SAs, os quais não têm fronteira clara, permite que as configurações variantes não sejam previsíveis (*SC-4*). Além disso, no mesmo contexto, Lu (2007) destaca que uma grande dificuldade consiste na definição de oráculos de teste (*SC-5*).

Assim como no C-1, tais desafios lidam com configurações que não são previsíveis, portanto, é um grande desafio garantir a corretude de configurações não testadas anteriormente.

### **C-3 – Como detectar e evitar configurações incorretas definidas em tempo de execução**

Os autores Niebuhr e Rausch (2009b), Niebuhr e Rausch (2007) e Micskei et al. (2012) mencionam que a dinamicidade de SAs, os quais não têm fronteira clara, permite que as configurações variantes não sejam previsíveis (*SC-6*). Por outro lado, Niebuhr e Rausch (2007) mencionam a impossibilidade de provar a corretude de conexões entre interfaces de componentes em tempo de execução. Tais desafios também pertencem à questão de detectar configurações incorretas que são definidas no tempo de execução (*SC-8*). Além disso, Fredericks et al. (2013) mencionam a dificuldade (*SC-7*) de definir propriedades de primeira classe não funcionais e adaptáveis em propriedades testáveis do sistema.

Assim como no C-2, é um grande desafio definir dinamicamente casos de teste para evitar configurações incorretas, de modo a garantir a contínua operação do sistema.

### **C-4 – Como testar SAs que executam em um ambiente distribuído e heterogêneo**

Os autores Sama et al. (2010a), King et al. (2011b) e Kephart e Chess (2003) mencionam a dificuldade de testar recursos que são compartilhados entre processos paralelos e/ou distribuídos (*SC-9*). Já os autores Tse et al. (2004) mencionam a dificuldade de projetar



casos de teste que levem em conta um comportamento assíncrono (*SC-10*). Nos referidos trabalhos destaca-se a dificuldade em antecipar mudanças de ambiente quando o SA envolve a múltiplos domínios de aplicação e vários fornecedores. Além disso, Lu (2007) menciona a dificuldade de lidar com uma arquitetura desenvolvida em camadas, que encapsula os contextos gerados (*SC-11*).

Neste cenário, por exemplo, é desafiador definir casos de testes para SAs que possuem propriedades como dinamismo, heterogeneidade e sincronização.

### **C-5 – Como lidar com interferência de usuário na configuração do sistema**

Os autores Sama et al. (2010a) e Griebel e Gruhn (2014) mencionam a dificuldade de projetar casos de teste que simulem a interferência do usuário (*SC-12*). Tal desafio remete-se a SAs em que se dá liberdade ao usuário para personalizar as configurações do sistema.

Assim como nos desafios genéricos C-1 e C-2, é desafiante antecipar configurações do SA. Neste caso, em particular, tais configurações vêm das possíveis personalizações do usuário.

### **C-6 – Como antecipar todas mudanças de contextos relevantes e quando podem impactar o comportamento de SAs**

Os autores Wang et al. (2007) destacam que possíveis mudanças no contexto podem afetar o comportamento do sistema a qualquer momento, após o sistema estar em execução (*SC-18*). Da mesma forma, Fredericks et al. (2013) destacam a necessidade de identificar mudanças adaptativas importantes, dentro do ambiente do sistema em execução, de modo confiável (*SC-15*). Além disso, Griebel e Gruhn (2014) mencionam a questão de antecipar mudanças nos parâmetros de contexto (*SC-18*). Todos esses desafios específicos são relacionados à necessidade de antecipar mudanças de contexto e seus impactos no comportamento do sistema. Com relação a ambientes incertos, ou sistemas com características de aprendizagem, Jaw et al. (2008) destacam a dificuldade de validar modelos de projeto (*SC-17*) e Püschel et al. (2012) mencionam a dificuldade de testar um sistema cujos contextos mudam o tempo todo (*SC-13*). Tal desafio também é destacado por outros autores com foco na aprendizagem do SA (*SC-19*) (Micskei et al., 2012) ou com foco na estrutura do SA (*SC-20*) (King et al., 2007b) de acordo com a evolução do sistema.

Na questão de lidar com espaço de estado que SAs podem ter, Eberhardinger et al. (2014) mencionam a dificuldade de desenvolver um sistema que seja evolutivo, de modo que a evolução não seja totalmente especificada na etapa de projeto (*SC-14*). Püschel et al. (2013), entretanto, mencionam a necessidade de manipular a decisão comportamental

(isto é a caracterização de contexto) que impacta a estrutura do sistema e processos em execução (*SC-14*). Além disso, Fredericks et al. (2013) mencionam a dificuldade de selecionar variáveis de contexto que devem ser observadas (*SC-16*).

Com base neste conjunto inter-relacionado de desafios específicos, a principal questão consiste na construção de um conjunto de casos de teste que contemplem todas as variáveis relevantes de contextos.

### **C-7 – Como lidar com fluxo de dados e de controle que são dependentes de contexto em SAs**

Os pesquisadores Munoz e Baudry (2009) e Lu et al. (2006) destacam a dificuldade de aplicar o critério de teste baseado em fluxo de dados, devido aos defeitos serem sensíveis ao contexto, conterem interferência do ambiente e conterem fluxo de controle sensíveis ao contexto (*SC-21*). Por outro lado, Lu et al. (2008) e Püschel et al. (2013) relatam a questão de evitar estados de programa inconsistentes causados por “contextos ruidosos” (*SC-22*). Tais desafios relacionaram-se à propagação de erros através de componentes de um SA. Além disso, Lu (2007) destacam suas preocupações quando o teste é realizado sem levar em consideração a adaptação dinâmica, que um SA pode ter em tempo de execução (*SC-23*). Tal característica impõe um grande desafio no teste, assim, apenas aplicar um teste caixa-preta, por exemplo, não testaria adequadamente um SA.

Observa-se, na sobreposição de desafios específicos, a dificuldade relacionada à complexidade de se projetar modelos de testes (por exemplo grafos de fluxo de dados e de controle) e de se definir conjuntos de testes para cobrir propriedades de modelos (por exemplo caminhos de grafos), especificamente devido à natureza de mudanças de um SA em termos de estrutura e dados manipulados.

### **C-8 – Como manter casos de teste atualizados e precisos. Além disso, como manter rastreabilidade entre casos de teste, requisitos e componentes em um ambiente em mudanças**

Os autores Fredericks et al. (2013) mencionam a questão de manter a rastreabilidade entre requisitos, casos de teste e componentes de SAs (*SC-27*). Destacam, também, a dificuldade de definir propriedades de primeira classe não funcionais e adaptáveis em propriedades testáveis do sistema (*SC-24*). Além disso, os autores destacam a dificuldade em se avaliar precisamente o impacto nos casos de teste causados por adaptações do sistema (*SC-25*) (Micskei et al., 2012) e questões relacionadas a métricas de cobertura de teste (*SC-26*) (Fredericks et al., 2013).

Com base neste conjunto de desafios específicos, a principal questão consiste na questão de avaliar e atualizar relações de rastreabilidade entre os artefatos de teste e de software do SA sob teste.

### **C-9 – Como determinar os sensores e as respectivas frequências de monitoramento, em um ambiente de execução, que devem ser observados em um SA sob teste**

Os autores Fredericks et al. (2013) mencionam o desafio de determinar quais sensores, ou agregações de valores de sensores, podem mensurar propriedades desejáveis (*SC-29*). Esta questão está relacionada à determinação de dados relevantes de entrada. Além disso, destaca-se a necessidade de determinar com qual frequência tais dados devem ser coletados e monitorados (*SC-28*).

No cenário de preparar o ambiente e o conjunto de casos de teste, por exemplo, destacou-se uma dificuldade para determinar quais sensores devem estar habilitados durante o teste. Além disso, é difícil determinar a frequência que os dados serão coletados da execução do SA.

### **C-10 – Como simular um ambiente de execução e carga de trabalho reais**

Os autores Kephart e Chess (2003) e Micskei et al. (2012) mencionam a dificuldade de construir testes que capturem o tamanho e a complexidade de sistemas reais e/ou de grande carga de trabalho (*SC-30*). Esta questão permeia um desafio para simular, de modo real, ambientes de execução e cargas de trabalho que um sistema possa ter.

Particularmente para SAs, simular sistemas reais e de carga de trabalho é uma tarefa que está diretamente relacionada à imprevisibilidade e a limites do sistema.

### **C-11 – Como gerar casos de teste automaticamente para um ambiente em mudanças**

Os autores Yu e Gao (2014) mencionam a dificuldade de gerar automaticamente casos de teste para SAs (*SC-32*) que têm uma estrutura reconfigurável e/ou que o ambiente de execução é inerente a propriedades reconfiguráveis. Neste contexto, Vassev et al. (2010) elencam a dificuldade de reduzir o número de teste que são automaticamente gerados para um SA (*SC-31*).

### **C-12 – Como realizar verificação formal do comportamento adaptativo do sistema**

Os autores Weyns (2012) elencam o desafio específico de realizar verificação de modelos de SAs, particularmente na relação com propriedades adaptativas (*SC-33*). Diferentemente dos demais desafios, que estão no contexto de validação, o desafio *SC-33* está relacionado a verificação. Outros autores (Micskei et al., 2012) destacam a dificuldade de utilizar linguagens formais para representar o comportamento de um SA sensível ao contexto (*SC-34*). Para modelar o teste com tais linguagens, dever-se-à levar em consideração o comportamento do SA que é sensível ao contexto. Por tal motivo, representar formalmente todo o comportamento do SA em tempo de execução não é uma tarefa trivial.

### 4.3.1 Discussão

Os desafios específicos e genéricos identificados e caracterizados, como um resultado deste trabalho, revelaram que prover confiabilidade de SAs é uma tarefa muito complexa. Existem várias propriedades de SAs que aumentam a dificuldade na validação desses sistemas. Apesar disso, notou-se que alguns desafios de teste foram mais recorrentes do que outros.

Observa-se que o desafio genérico *C-6* (Como antecipar todas mudanças de contextos relevantes e quando podem impactar o comportamento de SAs) está relacionado com oito desafios específicos. Outras questões que repetidamente chamam a atenção dos pesquisadores são *C-2*, *C-3* and *C-4*, seguidas por *C-1* e *C-8*.

Observou-se também que os desafios genéricos estão estritamente relacionados a: (i) decisões em tempo de execução de SAs (*C-1*, *C-2*, *C-3*, *C-6*, *C-7*, *C-10*, *C-12*), requerendo que o teste lide com imprevisibilidade de configurações que o sistema possa ter; (ii) ambientes distribuídos e heterogêneos (*C-4*), consequentemente levando o teste para além da fronteira de um único sistema; (iii) interferência de usuários (*C-5*), que requer testes que lidem com adaptações que são personalizadas por usuários do sistema; (iv) atualização e rastreabilidade entre artefatos de software variados (*C-8*), que requerem a manutenção de artefatos que incluem incerteza que é inerente de SAs; (v) definição de sensores e frequência de monitoramento de dados (*C-9*), que tem um impacto direto nas configurações do ambiente de teste; e (vi) geração de casos de teste (*C-11*), que pode ser não efetiva devido à incerteza presente em SAs.

## 4.4 A Presença dos Desafios em Sistemas

Uma das atividades desenvolvidas neste trabalho foi a busca por sistemas candidatos à análise de suas propriedades, que estão disponíveis em repositórios de código-fonte na

*Web*<sup>1</sup>. Esta busca teve como objetivo elencar sistemas com comportamento adaptativo, segundo os próprios autores de tais sistemas. Neste processo, identificaram-se algumas peculiaridades a respeito dos sistemas como, por exemplo, a predominância da linguagem de programação Java e sistemas para dispositivos móveis.

Com uso dos repositórios mencionados com palavras-chaves relacionadas a SAs, obtiveram-se inicialmente 3846 códigos-fonte de sistemas. Como uma seleção inicial, selecionaram-se os códigos-fontes cujos autores descreveram algum tipo de comportamento adaptativo. Após avaliar tais descrições foram selecionados 122 códigos-fonte. Desses 122 códigos-fonte, selecionaram-se 66 que foram desenvolvidos utilizando-se a linguagem Java. Desses 66 códigos-fonte, selecionaram-se 39 que foram desenvolvidos utilizando-se a plataforma *android*. Portanto, ao identificar tal tendência (Java e dispositivos móveis), caracterizaram-se quais dos estudos primários selecionados na RS realizada neste trabalho, e apresentado no Capítulo 3, estavam no contexto de dispositivos móveis (ou avaliavam suas abordagens utilizando estes dispositivos). Dos 6 trabalhos encontrados (Griebe e Gruhn, 2014; Merdes et al., 2006; Sama et al., 2010a, 2008, 2010b; Ye et al., 2009), 3 deles (Sama et al., 2010a, 2008, 2010b) utilizam o sistema chamado *PhoneAdapter* que, também, estava disponível na web. Com isso, então, obteve-se um conjunto de 40 códigos-fonte cujas descrições i) continham comportamento adaptativo; e ii) foram desenvolvidos com uso de Java para dispositivos móveis (*android*). Tal processo é apresentado no apêndice A.

Um conjunto de 11 propriedades de SAs foi definido a fim de auxiliar na análise dos códigos-fonte. Tais propriedades foram classificadas em dois grupos: obrigatórias e não-obrigatórias. Assim, aqueles códigos-fontes que possuíam as propriedades obrigatórias foram classificados como SAs.

Com os 40 códigos-fonte selecionados, avaliou-se o código-fonte desses sistemas a fim de identificar as propriedades de SAs presentes. Como um exemplo, aqueles sistemas cujo códigos-fontes apresentavam a sincronização de dados e mensagens entre várias *threads* foram classificados com a propriedade *Multithreading*.

Utilizando-se as propriedades de SAs para selecionar os códigos-fonte, obteve-se um conjunto de cinco códigos-fonte que continham as propriedades obrigatórias. Tal análise de códigos-fonte, presente na Tabela 4.3, classificou esses cinco códigos-fonte como “adaptativos”. Detalhes a respeito da análise dessas propriedades nos códigos-fontes podem ser visualizados no apêndice B. Sendo assim, na Tabela 4.3, apresenta-se uma visão geral das propriedades presentes nos cinco SAs selecionados.

---

<sup>1</sup>Os repositórios foram o *GitHub* (<http://github.com/>), o *SourceForge* (<http://sourceforge.net/>) e o *Bitbucket* (<http://bitbucket.org/>), acessados em Maio de 2015.

Apesar da diferença entre os sistemas ser pequena em termos dos totais de propriedades presentes em cada um deles, destaca-se o SA5 como o mais aderente à “filosofia” dos SAs. Tal sistema contém dez propriedades elencadas. Um outro dado a ressaltar é com relação a propriedade “Adiciona/remove recursos em tempo de execução”; tal propriedade está presente em apenas dois sistemas (SA1 e SA5), a saber o SA1 foi denominado pelos respectivos autores como *Novi* e o SA5 como *PhoneAdapter*.

No exemplo ilustrativo da seção 2.2.4 utilizou-se o *PhoneAdapter* (SA5) como base. Ao analisar o código-fonte deste SA foi possível uma abstração teórica do funcionamento do mesmo, a qual é ilustrada na Figura 2.6. Além disso, na seção 2.2.4 exemplificou-se como o mapeamento das propriedades foi realizado no *PhoneAdapter* (SA5). O mesmo mapeamento foi reproduzido nos demais SAs analisados.

#### 4.4.1 Desafios Específicos Relacionados às Propriedades dos Sistemas

Nesta etapa do trabalho, foram estabelecidos os relacionamentos entre os 34 desafios específicos e as 11 propriedades de SAs, sem levar em consideração os SAs analisados. A propriedade *multithreading*, por exemplo, foi relacionada aos desafios específicos *SC-4*, *SC-6* e *SC-14* que enfatizavam as dificuldades de se “testar SAs que não têm fronteira clara”. A fronteira, neste caso, remete-se ao comportamento paralelo e/ou distribuído que SAs podem ter, por exemplo, um SA composto com vários sistemas distribuídos que se comunicam entre si. Neste sentido, a propriedade *multithreading* foi relacionada, também, aos desafios específicos *SC-9*, *SC-10* e *SC-11* que enfatizavam as dificuldades de se testar um SA composto por recursos que são compartilhados entre processos paralelos e/ou distribuídos, de um ou mais sistemas. Além disso, a propriedade *multithreading* foi relacionada aos desafios específicos *SC-21* e *SC-23* que enfatizaram dificuldades de se testar fluxos de dados de SA com “interferência do ambiente”, devido às características de processos paralelos e/ou distribuídos. O mesmo relacionamento foi realizado com as demais propriedades e desafios, que se apresentam na Tabela 4.4 com um “x” indicando os respectivos relacionamentos. Ressalta-se que na Tabela 4.3 as propriedades são enumeradas de acordo com as descrições das mesmas presentes na Tabela 4.3.

Na Tabela 4.4, destaca-se que algumas propriedades foram mais relacionadas a desafios específicos que outras. A propriedade *add/rem*, por exemplo, relacionou-se a 30 dos desafios específicos. Isso ocorreu porque tal propriedade contém a característica de adicionar e/ou remover algum recurso do SA em tempo de execução, assim, adicionar e/ou remover recursos

em tempo de execução leva o sistema a ter que mudar sua estrutura e/ou comportamento (que nem sempre podem ser previstos)(Micskei et al., 2012; Munoz e Baudry, 2009; Tse et al., 2004; Welsh e Sawyer, 2010). Neste sentido, os desafios específicos pertencentes à caracterização do desafio genérico *C-1*, por exemplo, foram relacionados a esta propriedade.

Ao relacionar os desafios específicos às propriedades de SAs, uma vez que cada SA continha um grupo de propriedades, pôde-se identificar quais desafios específicos poderiam estar relacionados aos SAs analisados neste trabalho. Neste sentido, na Tabela 4.4, sete desafios específicos (*SC-4*, *SC-9*, *SC-10*, *SC-11*, *SC-14*, *SC-21* e *SC-23*) foram relacionados à propriedade *multithreading* e, portanto, tais desafios foram relacionados aos SAs que continham tal propriedade. A saber, todos os SAs elencados continham a propriedade *multithreading*, de modo que os desafios específicos relacionados a tal propriedade foram também relacionados a todos os SAs analisados. Assim, o relacionamento entre os SAs e os desafios específicos, durante a etapa de teste, pode ser resumido na Tabela 4.5, a qual se destacam os sistemas SA2, SA3 e SA5 como os que mais se relacionaram aos desafios específicos por meio das propriedades de SAs. Ressalta-se que o relacionamento entre SAs e desafios foi baseado no conjunto dos cinco SAs analisados neste trabalho.

## 4.5 Abordagens de Teste

Os estudos originados do critério de inclusão CI-1 da RS forneceram as abordagens de teste que estavam no contexto de SAs. Assim, efetuaram-se a análise do relacionamento entre abordagens de teste e desafios específicos, a análise do relacionamento entre fases de teste e abordagens de teste da RS e, por fim, a análise do relacionamento entre as abordagens de teste e SAs elencados com base no desafios específicos.

### 4.5.1 Abordagens de Teste que Estão Relacionadas aos Desafios Específicos

A fim de identificar as abordagens de teste para lidarem com os desafios específicos, relacionaram-se as abordagens de teste encontradas na RS – originadas do critério de inclusão CI-1 da tabela 3.6 – aos desafios específicos caracterizados. Como um exemplo, a abordagem S03 (Tse et al., 2004) aplica o teste caixa preta para gerar casos de teste, por exemplo, o desafio específico *SC-4* “SAs que não têm fronteira clara” e o *SC-10* “SAs comportamento assíncrono” poderiam ser mitigados, uma vez que lidar com um sistema como uma caixa preta preocupa-se apenas com os dados de entrada e de saída, assim, independentemente da “fronteira” do sistema (que pode ter um ou vários sub sistemas)

ele seria testado como apenas um sistema. Um outro exemplo, a abordagem S17 (Sama et al., 2010a) propôs a representação de SAs por meio de uma MEF que pode simular o comportamento e possíveis transições entre estados do SA, por exemplo, o desafio *SC-12* “lidar com interferência de usuários” poderia ser mitigado, uma vez que se desenvolve a MEF com base no usuário do SA. O mesmo relacionamento, dessas abordagens e desafios acima, foi realizado com as demais abordagens e os desafios específicos. Como resultado obteve-se os relacionamentos presentes nas tabelas 4.6 e 4.7.

Observa-se que algumas abordagens se destacaram mais que outras. Como exemplo a abordagem S03 (Tse et al., 2004) e a S15 (Niebuhr et al., 2009) relacionaram-se com, respectivamente, 19 e 16 desafios específicos. Ressalta-se que apenas a quantidade de desafios específicos relacionados às abordagens de teste não implica na inclusão ou exclusão do uso de uma determinada abordagem. Outros fatores predominantes são a avaliação e o domínio de aplicação utilizados durante a abordagem. Algumas abordagens, como é o caso da S31 (Fredericks et al., 2013), necessitam que o SA sob teste tenha sido desenvolvido para se autotestar em tempo de execução. Portanto, para definir se uma abordagem é ou não superior a outra, deve-se avaliar também as propriedades e características do SA desenvolvido.

#### **4.5.2 Abordagens de Teste Relacionadas às Fases de Teste**

A fim de identificar as abordagens de teste que mais contemplam as fases de teste, relacionaram-se as abordagens de teste encontradas na RS – originadas do critério de inclusão CI-1 da tabela 3.6 – às fases de testes, fundamentadas na Seção 2.3.2. Como um exemplo, a abordagem S05 (Lu et al., 2006) forneceu as atividades: 1) uma técnica de teste estrutural; 2) quatro critérios de teste; 3) um procedimento sobre como gerar casos de teste; e 4) um conceito para geração de casos de teste com base nas situações do SA. Assim, como fundamentado na Seção 2.3.2 sobre fases de teste, a fase de *Projeto* de teste é composta por cinco atividades. Dessas cinco atividades, quatro foram contemplados na abordagem S05 (Lu et al., 2006). Além disso, a mesma abordagem forneceu métricas que podem ser utilizadas com base nos critérios da abordagem, sendo “métricas” a quarta atividade da fase *Finalização*. O mesmo procedimento para estabelecer esse relacionamento entre abordagens de teste de SAs e fases do processo de teste foi realizado para todas as abordagens identificadas na RS. Como resultado obtiveram-se os relacionamentos presentes na Tabela 4.8. Na Tabela 4.8 a primeira linha superior de título refere-se às abordagens de teste analisadas. A primeira coluna à esquerda refere-se às fases de teste fundamentadas na Seção 2.3.2, de modo que, A é *Planejamento*, B é *Projeto*, C é *Execução*,



D é *Acompanhamento* e E é *Finalização*. Além disso, as fases são enumeradas de acordo com suas respectivas atividades também apresentadas na Seção 4.8.

Observam-se, na Tabela 4.8, que algumas abordagens se destacaram mais que outras e, ainda, que a fase *Projeto* é a mais predominante em todas as abordagens. Ressalta-se que apenas a quantidade de fases de teste relacionadas às abordagens não implica na inclusão ou exclusão do uso de uma determinada abordagem. Um outro fator predominante é o se determinada abordagem será utilizada em conjunto com outra abordagem. Como um exemplo a abordagem S22 (Akour et al., 2011) está focada na aplicação do teste de regressão, o que implicaria na aplicação do teste após o SA ter sido testado anteriormente. Portanto, deve-se definir qual a necessidade de utilizar ou não uma determinada abordagem.

**Tabela 4.1:** Desafios específicos – Parte 1.

Desafios	SC	Descrição	IDC
32; 34; 35; 38 e 40	SC-1	A dificuldade de lidar com todas possíveis combinações de “variações de contexto” quando projetar os casos de teste (Micskei et al., 2012; Munoz e Baudry, 2009; Tse et al., 2004; Welsh e Sawyer, 2010).	<b>C-1</b>
33	SC-2	A questão de limitar (ou não) a habilidade do sistema adaptar-se (Fredericks et al., 2013).	
43	SC-3	A questão de quando parar o teste, ao utilizar uma abordagem dirigida por modelos (Wotawa, 2012).	
1; 2; 31; e 32	SC-4	A dinamicidade de SAs, os quais não têm fronteira clara, permite que as configurações variantes não sejam previsíveis (Cámara et al., 2015; Niebuhr e Rausch, 2007; Niebuhr et al., 2009; Welsh e Sawyer, 2010).	<b>C-2</b>
47	SC-5	A dificuldade de definir os oráculos de teste (Lu, 2007).	
2; 4; e 37	SC-6	A dinamicidade de SAs, os quais não têm fronteira clara, permite que as configurações variantes não sejam previsíveis (Micskei et al., 2012; Niebuhr e Rausch, 2007; Niebuhr et al., 2009).	<b>C-3</b>
3	SC-7	A dificuldade de definir propriedades de primeira classe não funcionais e adaptáveis em propriedades testáveis do sistema (Fredericks et al., 2013).	
31	SC-8	A impossibilidade de provar a corretude de conexões entre interfaces de componentes no tempo de execução (Niebuhr e Rausch, 2007).	
5; 6; 29; e 30	SC-9	A dificuldade de testar recursos que são compartilhados entre processos paralelos e/ou distribuídos (Kephart e Chess, 2003; King et al., 2011b,b; Sama et al., 2010a).	<b>C-4</b>
36	SC-10	A dificuldade de projetar casos de teste que levem em conta um comportamento assíncrono (Tse et al., 2004).	
45	SC-11	A dificuldade de lidar com uma arquitetura desenvolvida em camadas, que encapsula os contexto gerados (Lu, 2007).	
7; e 28	SC-12	A dificuldade de projetar casos de teste que simulem a interferência do usuário (Griebe e Gruhn, 2014; Sama et al., 2010a).	<b>C-5</b>
8	SC-13	A questão de testar um sistema cujos contextos mudam o tempo todo (Püschel et al., 2012).	<b>C-6</b>
9; e 11	SC-14	A necessidade de manipular o espaço de decisão comportamental que impacta na estrutura do sistema e nos processos em execução (Eberhardinger et al., 2014; Püschel et al., 2013).	
10	SC-15	A questão de identificar confiavelmente mudanças que são importantes dentro do sistema e em seu ambiente de execução (Fredericks et al., 2013).	
25	SC-16	A questão de determinar que propriedades do sistema devem ser observadas (Fredericks et al., 2013).	
26	SC-17	O desafio de validar modelos de projeto que contém características de incerteza e aprendizagem (Jaw et al., 2008).	
27; e 28	SC-18	As mudanças possíveis em contexto que podem afetar o comportamento da aplicação a qualquer momento durante a execução (Griebe e Gruhn, 2014; Wang et al., 2007).	
39	SC-19	A dificuldade de lidar com características de aprendizagem e raciocínio, que seus comportamentos podem mudar dependendo da evolução do ambiente (Micskei et al., 2012).	
44	SC-20	A questão de ambos comportamento e estrutura do sistema poderem mudar durante a execução (King et al., 2007b).	

**Tabela 4.2:** Desafios específicos – Parte 2.

12; 22	<i>SC-21</i>	A dificuldade para aplicar o critério de teste de fluxo de dados, devido aos defeitos serem sensíveis ao contexto, conterem interferência do ambiente e conterem fluxo de controle sensível ao contexto (Lu et al., 2006; Munoz e Baudry, 2009).	<b>C-7</b>
23; e 24	<i>SC-22</i>	A questão de evitar estados de programa inconsistentes causados por “contextos ruidosos” (Lu et al., 2008; Püschel et al., 2013).	
46	<i>SC-23</i>	A dificuldade de re-executar a execução se casos de teste são apenas valores de entrada (Lu, 2007).	
3	<i>SC-24</i>	A dificuldade de definir propriedades de primeira classe não funcionais e adaptáveis em propriedades testáveis do sistema (Fredericks et al., 2013).	<b>C-8</b>
13	<i>SC-25</i>	A questão de avaliar precisamente o impacto nos casos de teste causados pelas adaptações do sistema (Fredericks et al., 2013).	
14; e 42	<i>SC-26</i>	A questão de definir precisamente métricas de cobertura de teste (Fredericks et al., 2013; Micskei et al., 2012).	
21	<i>SC-27</i>	A questão de resolver e atualizar as relações de rastreabilidade entre requisitos individuais, componentes e casos de teste (Fredericks et al., 2013).	
15	<i>SC-28</i>	A questão de determinar qual a frequência que o monitoramento de dados deve ser realizado (Fredericks et al., 2013).	<b>C-9</b>
20	<i>SC-29</i>	A questão de determinar quais sensores, ou agregações de valores de sensores, podem mensurar propriedades desejáveis (Fredericks et al., 2013).	
16; e 41	<i>SC-30</i>	A dificuldade de construir conjuntos de teste que capturem o tamanho e complexidade de sistemas e cargas de trabalho reais (Kephart e Chess, 2003; Micskei et al., 2012).	<b>C-10</b>
17	<i>SC-31</i>	A necessidade de reduzir o número de teste que são gerados automaticamente (Vassev et al., 2010).	<b>C-11</b>
19	<i>SC-32</i>	A dificuldade para gerar casos de teste automaticamente para um ambiente em mudanças (Yu e Gao, 2014).	
18	<i>SC-33</i>	A questão para realizar checagem de modelos de propriedades do sistema relacionadas ao comportamento adaptativo (Weyns, 2012).	<b>C-12</b>
40	<i>SC-34</i>	A dificuldade de lidar com mecanismos para expressar e formalizar comportamento sensível ao contexto (Micskei et al., 2012).	

**Tabela 4.3:** As propriedades nos SAs analisados.

ID	Descrição	SA1 <sup>a</sup>	SA2 <sup>b</sup>	SA3 <sup>c</sup>	SA4 <sup>d</sup>	SA5 <sup>e</sup>
1	Multithreading (Gerenciador autônomo e Sistema gerenciado)	x	x	x	x	x
2	Interação baseada em tempo		x	x		x
3	Interação baseada em eventos	x	x	x	x	x
4	Sensores	x	x	x	x	x
5	Atuadores	x	x	x	x	x
6	Conhecimento	x	x	x	x	x
7	Monitor	x	x	x	x	x
8	Analizador	x	x	x	x	x
9	Planejador					
10	Executor	x	x	x	x	x
11	Adiciona/remove recursos no tempo de execução	x				x
<b>Total</b>		9	9	9	8	10

<sup>a</sup>Novi – Um sistema de gerenciamento pessoal sensível ao contexto – <http://sourceforge.net/projects/novi/?source=directory>

<sup>b</sup>STMobile – Um sistema que reconhece atividades do usuário e realiza notificações – <https://bitbucket.org/sbobek/stmobile/wiki/Home>

<sup>c</sup>Context-Aware Music Player – Um tocador de músicas sensível ao contexto – <https://github.com/Whippler/Context-aware-musicplayer>

<sup>d</sup>Proxilence – Um sistema que altera o áudio de acordo com a localização do usuário – <https://github.com/clamped/Proxilence>

<sup>e</sup>PhoneAdapter – Um sistema sensível ao contexto baseado em regras definidas pelo usuário – <http://sccpu2.cse.ust.hk/afchecker/phoneadapter.html>

**Tabela 4.4:** Propriedades dos SAs analisados e Desafios Específicos

Desafios	1	2	3	4	5	6	7	8	9	10	11	Total
SC-1						x					x	2
SC-2						x					x	2
SC-3						x					x	2
SC-4	x			x	x	x	x	x	x	x	x	9
SC-5						x					x	2
SC-6				x	x	x		x			x	5
SC-7										x	x	2
SC-8						x					x	2
SC-9	x	x	x	x	x		x	x	x	x	x	10
SC-10	x	x	x	x	x		x	x	x	x	x	10
SC-11	x	x	x	x	x		x	x	x	x	x	10
SC-12						x						1
SC-13						x		x			x	3
SC-14	x			x	x	x	x	x	x	x	x	9
SC-15						x		x			x	3
SC-16				x			x					2
SC-17						x					x	2
SC-18						x					x	2
SC-19						x						1
SC-20						x					x	2
SC-21	x	x	x			x		x			x	6
SC-22						x					x	2
SC-23	x	x	x			x		x			x	6
SC-24										x	x	2
SC-25						x		x			x	3
SC-26						x		x			x	3
SC-27						x		x			x	3
SC-28				x			x				x	3
SC-29				x			x					2
SC-30						x					x	2
SC-31						x					x	2
SC-32						x					x	2
SC-33						x					x	2
SC-34						x					x	2
Total	7	5	5	9	6	26	8	13	5	7	30	

**Tabela 4.5:** Desafios Específicos relacionados aos SAs analisados

Desafios	SA1	SA2	SA3	SA4	SA5	Totais
SC-1	x	x	x		x	4
SC-2	x	x	x		x	4
SC-3	x	x	x		x	4
SC-4	x	x	x	x	x	5
SC-5	x	x	x		x	4
SC-6	x	x	x	x	x	5
SC-7	x	x	x	x	x	5
SC-8	x	x	x		x	4
SC-9	x	x	x	x	x	5
SC-10	x	x	x	x	x	5
SC-11	x	x	x	x	x	5
SC-12		x	x		x	3
SC-13	x	x	x	x	x	5
SC-14	x	x	x	x	x	5
SC-15	x	x	x	x	x	5
SC-16	x	x	x	x	x	5
SC-17	x	x	x		x	4
SC-18	x	x	x		x	4
SC-19		x	x		x	3
SC-20	x	x	x		x	4
SC-21	x	x	x	x	x	5
SC-22	x	x	x		x	4
SC-23	x	x	x	x	x	5
SC-24	x	x	x	x	x	5
SC-25	x	x	x	x	x	5
SC-26	x	x	x	x	x	5
SC-27	x	x	x	x	x	5
SC-28	x	x	x	x	x	5
SC-29	x	x	x	x	x	5
SC-30	x	x	x		x	4
SC-31	x	x	x		x	4
SC-32	x	x	x		x	4
SC-33	x	x	x		x	4
SC-34	x	x	x		x	4
<b>Totais</b>	32	34	34	18	34	

**Tabela 4.6:** Relacionamento entre abordagens de teste e desafios específicos - Parte 1.

Desafios	S02	S03	S04	S05	S06	S07	S09	S10	S11	S12	S13	S14	S15	S16	S17	S19	S20
SC-1	x	x	x	x	x	x	x	x				x	x				
SC-2								x				x					
SC-3													x			x	
SC-4	x	x	x	x	x	x	x	x				x	x				
SC-5		x						x				x	x				
SC-6	x	x	x	x	x	x	x	x				x	x				
SC-7																	
SC-8	x	x	x			x	x	x				x	x				
SC-9	x	x	x	x		x	x		x				x	x	x		
SC-10	x	x	x	x		x	x		x				x	x	x		
SC-11	x	x	x	x		x	x		x				x	x	x		
SC-12	x	x				x	x	x				x	x		x		
SC-13	x	x				x	x	x				x	x				
SC-14	x	x		x	x	x						x	x				
SC-15	x	x				x	x	x				x	x				
SC-16		x															
SC-17																	
SC-18	x	x				x	x	x				x	x		x		
SC-19	x	x				x	x	x				x	x				
SC-20																	
SC-21				x						x		x					
SC-22				x						x		x					
SC-23		x		x						x		x					
SC-24																	
SC-25		x														x	
SC-26													x			x	
SC-27		x														x	
SC-28																	
SC-29		x															
SC-30																	
SC-31																x	
SC-32																x	
SC-33																	
SC-34																	
Totais	13	19	7	10	4	13	12	11	3	3	0	15	16	3	5	6	0

**Tabela 4.7:** Relacionamento entre abordagens de teste e desafios específicos - Parte 2.

Desafios	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	S31	S33	S34	S35	S36	S37	S38
SC-1	x	x		x			x	x	x			x					
SC-2	x			x			x					x					
SC-3	x						x	x				x	x		x		
SC-4	x	x		x	x		x	x	x			x		x			
SC-5	x			x								x					
SC-6	x	x		x	x		x	x	x			x		x			
SC-7																	
SC-8				x			x					x					
SC-9			x			x								x	x	x	x
SC-10	x		x			x								x	x	x	x
SC-11			x			x								x	x	x	x
SC-12	x	x		x	x			x				x		x			
SC-13		x		x	x			x				x		x			
SC-14	x	x		x					x		x						
SC-15	x	x		x	x			x				x		x			
SC-16		x															
SC-17								x									
SC-18	x	x		x	x			x				x		x			
SC-19		x		x	x			x				x		x			
SC-20																	
SC-21		x						x	x								
SC-22		x						x	x								
SC-23		x							x								
SC-24																	
SC-25													x				
SC-26	x						x	x				x	x		x		
SC-27													x				
SC-28		x															
SC-29		x															
SC-30																	
SC-31														x		x	
SC-32														x		x	
SC-33								x									
SC-34								x									
Totais	12	15	3	12	7	3	7	15	7	0	0	14	4	12	5	5	3



Tabela 4.8: Relacionamento entre abordagens e fases de teste

Fase/S	22	28	24	02	05	07	14	20	21	25	23	33	12	29	35	36	37	06	03	38	27	10	11	17	26	30	34	16	09	13	15	19	04	31		
A1		x		x		x			x	x								x	x						x								x			
A2																																				
A3																																				
A4																																				
A5																																				
A6																																				
A7		x																								x										
B1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x		x	x	x	x					
B2	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x			x										
B3	x	x	x	x	x		x	x	x		x	x	x	x	x	x	x	x	x	x	x						x		x		x	x	x			
B4	x			x	x		x		x			x																								
B5																																				
C1	x	x	x			x	x			x	x		x		x				x		x	x	x			x	x								x	
C2	x		x																																	
D1																																				
D2			x					x			x						x			x					x							x				
D3																																				
D4		x			x	x		x		x		x		x		x																				
D5																																				
D6																																				
D7																																				
E1																																				
E2																																				
E3																																				
E4		x			x	x		x		x		x		x		x									x	x		x								
E5																																				
E6																																				
E7																																				
Totais	6	6	6	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	1	2	2	2	2	1	1	

### **4.5.3 Abordagens de Teste Relacionadas aos Desafios Específicos Presentes nos SAs**

Na Seção 4.4.1 relacionaram-se as propriedades de SAs com os desafios específicos, assim, na etapa obteve-se a presença dos desafios específicos nos SAs elencados. Além disso, na Seção 4.5.1 relacionaram-se as abordagens de teste com os desafios específicos, assim, na etapa obtiveram-se as abordagens de teste que poderiam lidar com os desafios específicos. Portanto, com base nas seções 4.4.1 e 4.5.1, nesta seção, analisa-se o relacionamento entre as abordagens de teste e os SAs elencados. Este relacionamento, que pode ser visualizado nas Tabelas 4.9 e 4.10, teve como objetivo elencar as abordagens de teste que mais poderiam lidar com os desafios específicos caracterizados. O campo Desafios representa o desafio específico de teste. O campo Qtd. representa o relacionamento entre desafio específico e SAs, de modo que, 5 significa: dos cinco SAs elencados o desafio específico estava presente em todos os SAs; 4 significa: dos cinco SAs elencados o desafio específico estava presente em quatro dos SAs; e 3 significa: dos cinco SAs elencados o desafio específico estava presente em três dos SAs.

As Tabelas 4.9 e 4.10 apresentam três diferentes sub totais a fim de ilustrar quais os desafios mais presentes nos SAs elencados. Como pode ser observado, algumas abordagens se destacaram mais que outras no relacionamento. Dentre essas abordagens destacou-se a S03 (Tse et al., 2004) com 13 desafios específicos presentes nos cinco SAs analisados. Nota-se que a mesma abordagem S03 se destacou quando se estabeleceu o relacionamento entre abordagens de teste e desafios específicos. Além disso, outras abordagens S22 (Akour et al., 2011), S15 (Niebuhr et al., 2009) e S02 (Flores et al., 2004) também se destacaram com dez, nove e oito desafios específicos relacionados, respectivamente, quando os SAs elencados foram levados em consideração. Entretanto, o uso das abordagens também deve ser analisado, como um exemplo, a abordagem S22 (Akour et al., 2011) está focada na aplicação do teste de regressão, o que implicaria na aplicação do teste após o SA ter sido testado anteriormente. Deve-se definir qual a necessidade de utilizar ou não uma determinada abordagem. Se uma abordagem for selecionada como carácter complementar a outra, deve-se avaliar tal complementariedade.

**Tabela 4.9:** Relacionamento entre abordagens, desafios específicos e SAs - Parte 1.

Desafios	Qtd.	S02	S03	S04	S05	S06	S07	S09	S10	S11	S12	S13	S14	S15	S16	S17	S19	S20
SC-4	5	x	x	x	x	x	x	x	x				x	x				
SC-6	5	x	x	x	x	x	x	x	x				x	x				
SC-7	5																	
SC-9	5	x	x	x	x		x	x		x				x	x	x		
SC-10	5	x	x	x	x		x	x		x				x	x	x		
SC-11	5	x	x	x	x		x	x		x				x	x	x		
SC-13	5	x	x				x	x	x				x	x				
SC-14	5	x	x		x	x	x						x	x				
SC-15	5	x	x				x	x	x				x	x				
SC-16	5		x															
SC-21	5				x						x		x					
SC-23	5		x		x						x		x					
SC-24	5																	
SC-25	5		x														x	
SC-26	5													x			x	
SC-27	5		x														x	
SC-28	5																	
SC-29	5		x															
<b>Sub totais</b>		8	13	5	8	3	8	7	4	3	2	0	7	9	3	3	3	0
SC-1	4	x	x	x	x	x	x	x	x				x	x				
SC-2	4								x				x					
SC-3	4													x			x	
SC-5	4		x						x				x	x				
SC-8	4	x	x	x			x	x	x				x	x				
SC-17	4																	
SC-18	4	x	x				x	x	x				x	x		x		
SC-20	4																	
SC-22	4				x						x		x					
SC-30	4																	
SC-31	4																	x
SC-32	4																	x
SC-33	4																	
SC-34	4																	
<b>Sub totais</b>		3	4	2	2	1	3	3	5	0	1	0	6	5	0	1	3	0
SC-12	3	x	x				x	x	x				x	x		x		
SC-19	3	x	x				x	x	x				x	x				
<b>Sub totais</b>		2	2	0	0	0	2	2	2	0	0	0	2	2	0	1	0	0
<b>Totoais Gerais</b>		13	19	7	10	4	13	12	11	3	3	0	15	16	3	5	6	0

**Tabela 4.10:** Relacionamento entre abordagens, desafios específicos e SAs - Parte 2.

Desafios	Qtd.	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	S31	S33	S34	S35	S36	S37	S38
SC-4	5	x	x		x	x		x	x	x			x		x			
SC-6	5	x	x		x	x		x	x	x			x		x			
SC-7	5																	
SC-9	5			x			x								x	x	x	x
SC-10	5	x		x			x								x	x	x	x
SC-11	5			x			x								x	x	x	x
SC-13	5		x		x	x			x				x		x			
SC-14	5	x	x		x					x			x					
SC-15	5	x	x		x	x			x				x		x			
SC-16	5		x															
SC-21	5		x						x	x								
SC-23	5		x							x								
SC-24	5																	
SC-25	5													x				
SC-26	5	x						x	x				x	x		x		
SC-27	5													x				
SC-28	5		x															
SC-29	5		x															
<b>Sub totais</b>		<b>6</b>	<b>10</b>	<b>3</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>3</b>	<b>6</b>	<b>5</b>	<b>0</b>	<b>0</b>	<b>6</b>	<b>3</b>	<b>7</b>	<b>4</b>	<b>3</b>	<b>3</b>
SC-1	4	x	x		x			x	x	x			x					
SC-2	4	x			x			x					x					
SC-3	4	x						x	x				x	x		x		
SC-5	4	x			x								x					
SC-8	4				x			x					x					
SC-17	4								x									
SC-18	4	x	x		x	x			x				x		x			
SC-20	4																	
SC-22	4		x						x	x								
SC-30	4																	
SC-31	4														x		x	
SC-32	4														x		x	
SC-33	4								x									
SC-34	4								x									
<b>Sub totais</b>		<b>5</b>	<b>3</b>	<b>0</b>	<b>5</b>	<b>1</b>	<b>0</b>	<b>4</b>	<b>7</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>6</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>0</b>
SC-12	3	x	x		x	x			x				x		x			
SC-19	3		x		x	x			x				x		x			
<b>Sub totais</b>		<b>1</b>	<b>2</b>	<b>0</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>Totoais Gerais</b>		<b>12</b>	<b>15</b>	<b>3</b>	<b>12</b>	<b>7</b>	<b>3</b>	<b>7</b>	<b>15</b>	<b>7</b>	<b>0</b>	<b>0</b>	<b>14</b>	<b>4</b>	<b>12</b>	<b>5</b>	<b>5</b>	<b>3</b>

## 4.6 Considerações Finais

Neste capítulo, visando a responder a QP2, apresentou-se a caracterização dos desafios que foram recuperados na RS, utilizando propriedades de SAs para avaliar sistemas. Além disso, em direção a responder a QP3, apresentaram-se as análises das abordagens de teste recuperadas da RS, utilizando o critério de inclusão CI-1.

Portanto, este capítulo foram apresentadas algumas das etapas da metodologia utilizada neste trabalho, sendo elas: (2) *Investigação de SAs disponíveis em repositórios de código-fonte na Web*; (3) *Caracterização de desafios*; (4) *Caracterização de abordagens e seu relacionamento com as fases de teste*; (5) *Análise de desafios e SAs*; (6) *Análise de desafios e abordagens de teste*; e (7) *Análise de desafios, abordagens e SAs*.

Neste capítulo, com as caracterizações de desafios de teste, abordagens de teste e SAs, apresentaram-se: (i) um conjunto de desafios específicos; (ii) um conjunto de desafios genéricos; (iii) um relacionamento entre desafios específicos e abordagens de teste; (iv) um relacionamento entre desafios específicos e propriedades de SAs; (v) um relacionamento entre abordagens e fases de teste; e, por fim, (vi) um relacionamento entre abordagens de teste e desafios presentes em SAs. Assim, observaram-se que (i) alguns desafios são mais recorrentes que outros; (ii) algumas abordagens de teste relacionam-se aos desafios mais que outras abordagens; (iii) alguns sistemas contém mais propriedades de SAs que outros; e (iv) algumas abordagens de teste são mais indicadas de se utilizarem que outras, no contexto de mitigar desafios.

Assim, o próximo capítulo apresenta o processo seguido em busca da resposta para as questões de pesquisa: “QP3: Quais são os defeitos encontrados com a aplicação das estratégias de teste baseadas em desafios?” e “QP4: Quais são os desafios mitigados com a aplicação das estratégias de teste baseadas em desafios?”.

---

# Um Estudo Exploratório de Abordagens de Teste para Sistemas Adaptativos

---

---

## 5.1 Considerações Iniciais

Duas das questões de pesquisa definidas foram “QP3: Quais são os defeitos encontrados com a aplicação das estratégias de teste baseadas em desafios?” e “QP4: Quais são os desafios mitigados com a aplicação das estratégias de teste baseadas em desafios?”. Este capítulo visa a responder tais questões.

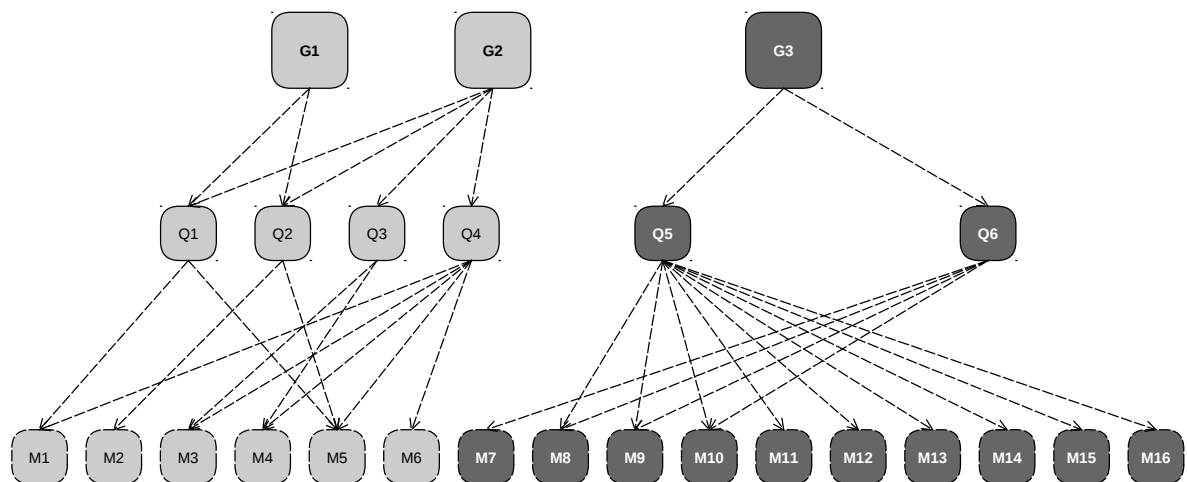
Portanto, este capítulo apresenta algumas das etapas da metodologia utilizada neste trabalho, sendo elas: (8) *Seleção de abordagens e SAs*; (9) *Seleção de um modelo subjacente de teste*; (10) *Definição das estratégias de teste*; (11) *Automatização das estratégias T e T\**; e (12) *Análise dos resultados*.

## 5.2 Definição do Estudo

Para a condução e avaliação de estudos, Basili (1992) propôs um paradigma dirigido por objetivos, questões e métricas denominado GQM (*Goal/Question/Metric*). O uso de tal modelo auxilia o planejamento e a condução do estudo, de modo a definir os principais

objetivos do estudo; a responder as questões necessárias para alcançar aos objetivos; e, por fim, a utilizar métricas bem definidas que ajudarão a responder às questões.

Neste trabalho, o GQM é composto por duas partes. A primeira parte, sendo apresentada por formas na cor cinza claro na Figura 5.1, contém os objetivos para a escolha das abordagens de teste e definição das estratégias T e T\*, selecionando o SA a ser utilizado no estudo. A segunda parte, sendo apresentada por formas na cor cinza escuro na Figura 5.1, contém o objetivo para a realização dos testes e a avaliação das estratégias T e T\*. Ambos os objetivos são apresentados na sequência, por meio do modelo sugerido por Basili (1992). Além disso, detalhes sobre a interpretação do GQM podem ser visualizados na Tabela 5.3.



**Figura 5.1:** Modelo GQM.

### 5.2.1 Objetivos

Na definição dos objetivos, Basili (1992) sugere que se devem especificar o objeto de estudo a se *analisar*, o *propósito* de tal análise, o que diz *respeito* à análise (ou seja, “o que” analisar no objeto de estudo), o *ponto de vista* da análise e o *contexto* em que a análise está inserida. Três são os objetivos do estudo:

- **[G1] - Definir uma estratégia T**
  - **Analisar:** as abordagens de teste da RS

- **Com o propósito de:** identificar uma abordagem de teste
- **Com respeito a:** desafios específicos com os quais as abordagens de teste se relacionam
- **Do ponto de vista de:** pesquisadores em desenvolvimento e teste de SAs
- **No contexto:** de um projeto de mestrado na área de Engenharia de Software
- [G2] - Definir uma estratégia T\*
- **Analisar:** as abordagens de teste da RS
- **Com o propósito de:** elencar uma estratégia de teste complementar à estratégia T
- **Com respeito a :** as fases de teste e os desafios específicos presentes nos SAs analisados
- **Do ponto de vista de:** pesquisadores em desenvolvimento e teste de SAs
- **No contexto:** de um projeto de mestrado na área de Engenharia de Software
- [G3] - Avaliar as estratégias T e T\*
- **Analisar:** a aplicação das estratégias T e T\*
- **Com o propósito de:** avaliar a efetividade e o custo das estratégias T e T\*
- **Com respeito a :** os casos de teste criados e executados
- **Do ponto de vista de:** pesquisadores em desenvolvimento e teste de SAs
- **No contexto:** de um projeto de mestrado na área de Engenharia de Software

### 5.2.2 Questões

Para alcançar aos objetivos definiu-se um conjunto de questões, que são:

- [Q1] - Qual abordagem de teste mais se relaciona aos desafios específicos?
- [Q2] - Qual abordagem mais se relaciona às fases de teste?
- [Q3] - Quais desafios específicos mais se relacionam aos SAs analisados?
- [Q4] - Qual abordagem de teste mais se relaciona aos desafios específicos nos SAs?
- [Q5] - Qual a efetividade da aplicação das estratégias T e T\*?
- [Q6] - Qual o custo da aplicação das estratégias T e T\*?

Tais questões são revisitadas no decorrer do capítulo.



### 5.2.3 Métricas

Os dados coletados neste trabalho remetem-se ao uso dos resultados da RS apresentada no capítulo anterior e ao uso de dados dos testes realizados. Para avaliação dos dados, a fim de responder as questões, um conjunto de métricas foi definido e apresentado na Tabela 5.1. Mais detalhes sobre as métricas são apresentados neste capítulo, ao passo que as análises são descritas.

**Tabela 5.1:** Métricas do GQM

ID	Descrição da Métrica
[M1]	Número de desafios específicos com os quais as abordagens de teste se relacionam
[M2]	Número de atividades de teste, relacionadas às fases de projeto e finalização, contempladas pelas abordagens de teste
[M3]	Número de propriedades de SAs presentes nos sistemas analisados
[M4]	Número de desafios específicos relacionados às propriedades de SAs
[M5]	Número de critérios de teste empregados nas abordagens de teste
[M6]	Grau de complementariedade de abordagens de teste, em relação às fases do processo de teste, tomando a estratégia T como referência
[M7]	Número de cenários e restrições a serem testados
[M8]	Número de casos de teste criados na primeira abordagem que compõe a estratégia T*
[M9]	Número de casos de teste criados na segunda abordagem que compõe a estratégia T*
[M10]	Número de casos de teste criados na terceira abordagem que compõe a estratégia T*
[M11]	Número de defeitos encontrados pela primeira abordagem que compõe a estratégia T*
[M12]	Número de defeitos encontrados pela segunda abordagem que compõe a estratégia T*
[M13]	Número de defeitos encontrados pela terceira abordagem que compõe a estratégia T*
[M14]	Número de falhas externalizadas pela primeira abordagem que compõe a estratégia T*
[M15]	Número de falhas externalizadas pela segunda abordagem que compõe a estratégia T*
[M16]	Número de falhas externalizadas pela terceira abordagem que compõe a estratégia T*

### 5.2.4 Tabela de Interpretação do GQM

A interpretação do GQM é apresentada na Tabela 5.3, de modo a sumarizar as métricas que foram utilizadas no estudo. O campo ID refere-se à identificação da interpretação; o campo Relação  $Q_j \times M_i$  ao relacionamento de uma respectiva questão a um conjunto de métricas do GQM; o campo Regra à expressão para a interpretação; o campo Interpretação à respectiva descrição; e, por fim, o campo Resultado ao resultado obtido na respectiva interpretação. Ressalta-se que a Tabela 5.3 apresenta resultados que serão abordados mais a frente neste capítulo. Como exemplo, na Tabela 5.3, com a expressão “MAIOR(M1) && M5>0” obtém-se a abordagem de teste com maior número de desafios específicos

relacionados (“MAIOR(M1)”), utilizando um filtro por abordagens de teste que fornecem algum tipo de critério de teste para se criar casos de teste (“M5>0”). As cinco primeiras interpretações (IDs 1-5) remetem-se às definições para a execução das estratégias de teste. As demais interpretações (IDs 6-14) remetem-se aos resultados das execuções das estratégias de teste.

## 5.3 Procedimento de Seleção de Abordagens de Teste e SAs para Serem Testados

O procedimento para a seleção de abordagens de teste e SAs para serem testados é sumarizado pelas formas na cor cinza claro na Figura 5.1, contendo os objetivos, as questões e as métricas.

### 5.3.1 Definição da Estratégia T

A definição da Estratégia T, relacionada ao objetivo [G1], utilizou as abordagens de teste da RS como objeto de estudo. Tal objeto de estudo foi analisado com o propósito de identificar uma abordagem de teste que faria parte da Estratégia de teste T. Neste sentido, analisaram-se os desafios específicos com os quais as abordagens de teste se relacionam, descritos na Seção 4.5. O dado analisado foi a quantidade de desafios específicos com os quais as abordagens de teste se relacionam.

Visando a responder a questão [Q1], elencou-se a abordagem de teste que mais se relaciona aos desafios específicos das tabelas 4.6 e 4.7. Observa-se que a abordagem S03 (Tse et al., 2004) relaciona-se com 19 dos 34 desafios específicos caracterizados. Por essa razão, a abordagem S03 (Tse et al., 2004) foi selecionada e denominada *Estratégia T*. Neste sentido, o ID 1 da Tabela 5.3 apresenta a interpretação respectiva. Como pode ser observado, a regra é “MAIOR(M1) && M5 > 0”. Tal regra representa a “abordagem de teste com mais desafios específicos relacionados, contendo critérios/subsídios para geração de casos de teste”.

### 5.3.2 Definição da Estratégia T\*

A definição da Estratégia T\*, relacionada ao objetivo [G2], utilizou as abordagens de teste da RS como objeto de estudo. Tal objeto de estudo foi analisado com o propósito de elencar uma estratégia de teste complementar à Estratégia de teste T. Neste sentido, analisaram-se

as fases de teste e os desafios específicos presentes nos SAs analisados, descrito na Seção 4.5. O dado analisado foram as abordagens que mais se relacionavam às fases de teste e as abordagens que mais poderiam mitigar os desafios específicos presentes nos SAs analisados.

Os seguintes passos foram realizados:

**Passo 1:** visando a responder a questão [Q2], elencou-se a abordagem que mais contempla as fases de *Projeto* e *Finalização*, pois em tais fases estão presentes subsídios para se criar casos de teste e se avaliar os resultados de teste, respectivamente. Assim, procurou-se elencar a abordagem que apresenta detalhes com relação ao projeto de casos de teste e a presença de métricas para se avaliar o teste realizado. Sendo assim, a abordagem que se destacou foi a abordagem S05 com quatro das cinco atividades da fase *Projeto* e uma das cinco atividades da fase *Finalização* da Tabela 4.8. Neste sentido, o ID 2 da Tabela 5.3 apresenta a interpretação respectiva. Como pode ser observado, a regra é “MAIOR(M2) && M5 > 0”. Tal regra representa a “abordagem de teste com mais fases de teste (projeto e finalização relacionadas), contendo critérios/subsídios para geração de casos de teste”.

**Passo 2:** visando a responder à questão [Q3], obteve-se o conjunto de desafios específicos que mais se relacionam aos SAs, de modo o ID 3 da Tabela 5.3 apresenta a interpretação respectiva, como pode ser observado, a regra é “MAIORES(RELAÇÃO(M3,M4))”. Tal regra representa a “relação 1 para N das propriedades de SAs com os desafios específicos”.

**Passo 3:** visando a responder à questão [Q4], analisaram-se as abordagens de teste que mais poderiam mitigar desafios específicos. Na Tabela 5.2 apresentam-se as primeiras 12 abordagens em relação a esse quesito. Neste sentido, o ID 4 da Tabela 5.3 apresenta a interpretação respectiva. Como pode ser observado, a regra é “MAIORES(RELAÇÃO(M1,M3,M4,M6)) && M5 > 0”. Tal regra representa a “abordagem de teste que mais se relaciona aos desafios específicos nos SAs analisados, contendo critérios/subsídios para geração de casos de teste”.

Observa-se que a abordagem que mais se destacou foi a já selecionada para a estratégia T (S03). Por tal motivo, essa abordagem não foi novamente elencada.

A segunda abordagem, S22 com dez dos dezoito desafios, classificou-se como uma abordagem para teste de regressão e por tal motivo esta foi descartada. As abordagens S15, S07 e S09 foram descartadas por focarem no autoteste, o que implicaria no desenvolvimento de um componente autotestável, que ficou fora do escopo deste trabalho. As demais abordagens (S02, S14, S35, S21, S28 e S33) foram avaliadas, a fim de identificar entre elas uma abordagem complementar às duas primeiras abordagens selecionadas (S03 e S05). Neste sentido, o ID 4 da Tabela 5.3 apresenta a interpretação respectiva. Assim, as abordagens S02, S35, S21 e S33 foram descartadas por utilizarem seus próprios modelos

para modelar um SA para realizar o teste. Além disso, entre as abordagens S14 e S28, a abordagem 14 contém mais desafios específicos relacionados, portanto, a abordagem S28 foi descartada. Ressalta-se o uso da métrica [M6], de modo que, as abordagens devem ser complementares às já selecionadas (S03 e S05). Além disso, abordagem S05 já foi selecionada no **Passo 1**, por tal motivo foi descartada.

Assim, ao avaliar a abordagem S14, identificou-se a mesma como uma abordagem complementar às demais selecionadas (S03 e S05). Tal complementariedade é apresentada como a Estratégia T\* mais a frente na Seção 5.5.2.

**Tabela 5.2:** Ranking das abordagens com base nos desafios específicos nos SAs analisados

ID	Autor	SCs nos 5 SAs analisados
S03	Tse et al. (2004)	13
S22	Akour et al. (2011)	10
S15	Niebuhr et al. (2009)	9
S02	Flores et al. (2004)	8
S05	Lu et al. (2006)	8
S07	King et al. (2007b)	8
S09	Niebuhr e Rausch (2007)	7
S14	Munoz e Baudry (2009)	7
S35	Griebe e Gruhn (2014)	7
S21	Welsh e Sawyer (2010)	6
S28	Micskei et al. (2012)	6
S33	Eberhardinger et al. (2014)	6

**Tabela 5.3:** Tabela de Interpretação do Modelo GQM.

ID	Relação Qj x Mi	Regra	Interpretação	Resultado
1	Q1 x M1 x M5	MAIOR(M1) && M5 >0	Abordagem de teste com mais desafios específicos relacionados e contém, critérios/-subsídios para geração de casos de teste	Abordagem S03 (Tse et al., 2004)
2	Q2 x M2 x M5	MAIOR(M2) && M5 >0	Abordagem de teste com mais fases de teste (projeto e finalização) relacionadas e contém, critérios/subsídios para geração de casos de teste	Abordagem S05 (Lu et al., 2006)
3	Q3 x M3 x M4	MAIORES(RELAÇÃO(M3,M4))	Relação 1 para N das propriedades de SAs com os desafios específicos	Lista de desafios que mais se relacionam às propriedades de SAs e, conseqüentemente, aos SAs
4	Q4 x M1 x M3 x M4 x M5 x M6	MAIORES(RELAÇÃO(M1,M3,M4,M6)) && M5 >0	Abordagem de teste que mais se relaciona aos desafios específicos nos SAs analisados	Abordagem S14 (Munoz e Baudry, 2009)
5	Q5 x M8 x M14	M8 / M14	Primeira aplicação (Estratégia T): Média de casos de teste por falha externalizada (49 / 4)	49 / 4 = 12,25
6	Q5 x M8 x M9 x M14 x M15	(M8 + M9) / (M14 + M15)	Segunda aplicação (primeira inclusão de casos de teste): Média de casos de teste por falha externalizada (89 / 14)	89 / 14 = 6,36
7	Q5 x M8 x M9 x M10 x M14 x M15 x M16	(M8 + M9 + M10) / (M14 + M15 + M16)	Terceira aplicação (Estratégia T*) (segunda inclusão de casos de teste): Média de casos de teste por falha externalizada (102 / 16)	102 / 16 = 6,38
8	Q5 x M14 x M11	M14 / M11	Primeira aplicação (Estratégia T): Média de falhas externalizadas por defeitos encontrados (4 / 1)	4 / 1 = 4
9	Q5 x M15 x M12	M15 / M12	Segunda aplicação (primeira inclusão de casos de teste): Média de falhas externalizadas por defeitos encontrados (14 / 4)	14 / 4 = 3,5
10	Q5 x M16 x M13	M16 / M13	Terceira aplicação (Estratégia T*) (segunda inclusão de casos de teste): Média de falhas externalizadas por defeitos encontrados (16 / 4)	16 / 4 = 4
11	Q6 x M7 x M8	M8 / M7	Primeira aplicação (Estratégia T): Média de casos de teste da abordagem S03 por cenário (49 / 28) especificado	49 / 28 = 1,75
12	Q6 x M7 x M8 x M9	(M8 + M9) / M7	Segunda aplicação (primeira inclusão de casos de teste): Média de casos de teste adicionando a abordagem S05 por cenário (89 / 28) especificado	89 / 28 = 3,18
13	Q6 x M7 x M8 x M9 x M10	(M8 + M9 + M10) / M7	Terceira aplicação (Estratégia T*) (segunda inclusão de casos de teste): Média de casos de teste adicionando a abordagem S14 por cenário (102 / 28) especificado	102 / 28 = 3,64

### 5.3.3 Seleção e Revisão do Modelo de Especificação de Teste

Utilizaram-se os estudos da RS que fornecem modelo de especificação de teste como objeto de estudo. Tal objeto de estudo foi analisado com o propósito de elencar um modelo que pudesse ser utilizado para testar os SAs analisados. Neste sentido, analisou-se o SA que mais se relacionou com propriedades de SAs, descrito na Seção 4.4, a fim de especificar cenários e restrições de teste. O dado analisado foi o modelo que poderia ser utilizado no SA selecionado. O passo a passo para a seleção do modelo é apresentado a seguir.

**Passo 1:** a seleção do SA apto a ser utilizado neste trabalho considerou o SA que continha o maior número de propriedades de SAs. Assim, o SA selecionado foi o *PhoneAdapter* (SA5) com dez das onze propriedades mencionadas na Seção 4.5.

**Passo 2:** identificou-se que o SA selecionado foi também utilizado em um dos trabalhos recuperados da RS (Sama et al., 2010a), os quais propuseram a máquina de estado finita de adaptação (*Adaptation Finite-State Machine*, A-FSM) fundamentada na Seção 3.7.4. Portanto, *por conveniência*, adotou-se a A-FSM para derivar os testes. Uma vez que a A-FSM foi apresentada junto ao sistema *PhoneAdapter* (SA5), o qual acabou sendo selecionado para ser testado neste estudo.

**Passo 3:** a A-FSM, de acordo com a notação adotada pelos autores originais, não representa as saídas oriundas de eventos que levam à transição entre estados. Assim, a fim de representar, além do comportamento do SA, a execução dos casos de teste foi subsidiada com uso de uma notação de máquinas de *mealy* (Bensalem et al., 2008) juntamente com a máquina de estado finita de adaptação. Neste sentido, ressalta-se o uso da métrica [M7] que utiliza o número de cenários e restrições a serem testados.

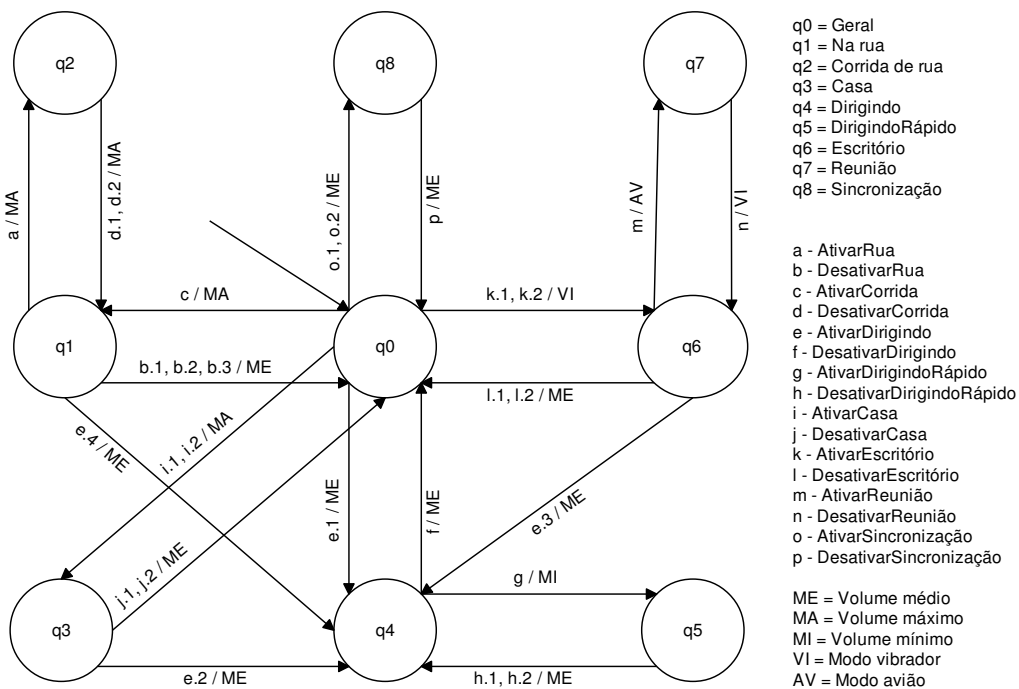
Com o uso da Máquina de Estados Finita de Adaptação, pôde-se representar o comportamento do *PhoneAdapter* (SA5) por meio dos estados (representando contextos), transições entre estados (representando situações) e casos de teste com uso da notação de Máquina de *Mealy*. Com isso, a maneira como modelar o SA bem e os casos de teste foi estabelecida.

O sistema *PhoneAdapter* (SA5) permite que seu comportamento seja personalizado pelo usuário. O usuário pode, assim, definir contextos possíveis que o SA pode operar e regras para realizar transições entre contextos. Para representar contextos e regras para transições entre contextos, Sama et al. (2010a) propuseram uma Máquina de Estados Finita de Adaptação (A-FSM – *Adaptation-Finite State Machine*). Entretanto, uma A-FSM, de acordo com a notação adotada pelos autores originais, não representa as saídas oriundas de eventos que levam à transição entre estados. Uma alternativa para esta representação seria o uso de uma máquina de *mealy* (Bensalem et al., 2008), que além de se representarem

transições entre estados (isto é, entradas de casos de teste) podem-se representar saídas de transições (isto é, saídas esperadas de casos de teste). Sendo assim, utilizou-se na A-FSM a notação de máquinas de *mealy* para representar a execução de casos de teste em SAs.

Uma máquina de *mealy* é definida pela sêxtupla  $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ , tal que  $Q$  é o conjunto de estados;  $\Sigma$  é o alfabeto finito de entradas;  $\Delta$  é o alfabeto finito de saídas;  $\delta$  é a função que mapeia  $Q \times \Sigma$  em  $Q$ , isto é, para cada par  $(q, a)$ , onde  $q \in Q$  e  $a \in \Sigma$  e  $\delta(q, a) = p$  é a transição de um estado  $q$  (com entrada  $a$ ) para um estado  $p$ .  $\lambda$  é a função que mapeia  $Q \times \Sigma \rightarrow \Delta$ , fornecendo uma saída associada a uma transição de estado;  $q_0 \in Q$  e é o estado inicial. Assim,  $\lambda(q, a) = s$  é a função de saída, tal que  $s$  é a saída do estado  $q$  quando a entrada for  $a$ .

A fim de representar estes contextos e regras, a máquina de *mealy* na Figura 5.2 contém nove estados que representam o dia-a-dia de um usuário. O estado “Geral” ( $q_0$ ) é o estado inicial – neste o usuário estará ao executar o sistema e ao término das demais transições. A partir do estado “Geral” ( $q_0$ ) pode-se trocar para os estados “Na rua” ( $q_1$ ), “Casa” ( $q_3$ ), “Dirigindo” ( $q_4$ ), “Escritório” ( $q_6$ ) e “Sincronização” ( $q_8$ ).



**Figura 5.2:** O uso de uma A-FSM com notação de Máquina de *Mealy*.

A cada momento que uma transição é realizada uma adaptação é executada no áudio. “MA” significa que o áudio passará a estar no volume máximo; “ME” significa que o áudio passará a estar com 50% do volume; “MI” significa que o áudio estará em, por exemplo, 10% do volume; “VI” significa que o áudio estará desligado e o *vibrador* ligado; e por fim, “AV” significa que “Modo avião” do *smartphone* estará ligado.

Detalhes da máquina de *mealy* da Figura 5.2 podem ser encontrados na Tabela 5.4, onde os estados, transições e saídas podem ser visualizados.

Como um exemplo, a transição de ID  $g$  da Tabela 5.4, quando o usuário estiver no estado “Dirigindo” seu carro e o GPS do SA identifica que a velocidade está superior a 70 km/h, ocorre a transição para o estado “Dirigindo rápido”. Assim, dependendo do estado do usuário e se uma determinada regra for satisfeita, acontece a transição de um estado para outro.

Formalmente a máquina de *mealy* da Figura 5.2 pode ser descrita pela sêxtupla  $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ , onde:

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$
- $\Sigma = \{a, b.1, b.2, b.3, c, d.1, d.2, e.1, e.2, e.3, e.4, f, g, h.1, h.2, i.1, i.2, \dots, o.1, o.2, p\}$
- $\Delta = \{ME, MA, MI, VI, AV\}$
- $\delta = \delta(q_i, w') = q_j$ , tal que  $q_i$  e  $q_j \in Q$ , sendo  $i, j \geq 0$  e  $\leq 8$ . Sabendo que  $w' \in \Sigma$ .
- $\lambda = \lambda(q_i, w') = \Delta'$ , tal que  $q_i \in Q$ , sendo  $i \geq 0$  e  $\leq 8$ . Sabendo que  $w' \in \Sigma$  e  $\Delta' \in \Delta$ .
- $q_0$  é o estado inicial

Assim como introduzido a respeito de predicados de adaptação na Seção 2.2.2, esses predicados (na Tabela 5.4) representam algumas situações do SA, de modo que, se um predicado de adaptação for verdadeiro então o sistema deve adaptar-se para satisfazer tal predicado e mudar seu estado corrente. No adaptar no caso do *PhoneAdapter* (SA5), realizam-se mudanças acerca do volume do áudio e ligar/desligar do modo avião de um *smartphone*. A máquina de *mealy* especificada na Figura 5.2, neste caso, atua com os sensores: *bluetooth*, *data/Hora* e *GPS* e com os atuadores: *audio*, *modo avião* e *vibracall*.

Um outro ponto a se destacar é o domínio de aplicação que é baseado no usuário, de modo que caso se mude o perfil do usuário, muda-se também o modelo de contextos. Neste trabalho, o modelo de contextos foi utilizado com base no usuário que cria a sua agenda e nesta agenda ele define suas atividades como na Tabela 5.5. A agenda seria um guia para a definição do domínio baseado no perfil do usuário.



**Tabela 5.4:** Algumas possíveis transições entre estados realizadas pelo *PhoneAdapter* (SA5), adaptadas do trabalho de Sama et al. (2010a).

ID	Transição	Estado atual	Novo Estado	Predicado de Adaptação	Saída	Pri.
c	AtivarRua	Geral	Na rua	(GPS == “valido”) && (GPS.Local != “Casa” && GPS.Local != “Escritorio”)	MA	5
b.1	DesativarRua_GPSNaoValido	Na rua	Geral	(GPS != “valido”)	ME	5
b.2	DesativarRua_LocalCasa	Na rua	Geral	(GPS.Local == “Casa”)	ME	5
b.3	DesativarRua_LocalEscritorio	Na rua	Geral	(GPS.Local == “Escritorio”)	ME	5
a	AtivarCorrida	Na rua	Corrida de rua	(GPS == “valido”) && (GPS.Velocidade >5 km/h)	MA	5
d.1	DesativarCorrida_GPSNaoValido	Corrida de rua	Na rua	(GPS != “valido”)	MA	5
d.2	DesativarCorrida_VelocidadeMenorIgual5	Corrida de rua	Na rua	(GPS.Velocidade ≤ 5 km/h)	MA	5
e.1	AtivarDirigindo_Geral	Geral	Dirigindo	(Bluetooth == bt_carro)	ME	2
e.2	AtivarDirigindo_Casa	Casa	Dirigindo	(Bluetooth == bt_carro)	ME	2
e.3	AtivarDirigindo_Escritorio	Escritório	Dirigindo	(Bluetooth == bt_carro)	ME	2
e.4	AtivarDirigindo_NaRua	Na rua	Dirigindo	(Bluetooth == bt_carro)	ME	2
f	DesativarDirigindo	Dirigindo	Geral	(Bluetooth != bt_carro)	ME	2
g	AtivarDirigindoRapido	Dirigindo	DirigindoRapido	(GPS == “valido”) && (GPS.Velocidade >70 km/h)	MI	1
h.1	DesativarDirigindoRapido_GPSNaoValido	DirigindoRapido	Dirigindo	(GPS != “valido”)	ME	1
h.2	DesativarDirigindoRapido_VelocidadeMenorIgual70	DirigindoRapido	Dirigindo	(GPS.Velocidade ≤ 70 km/h)	ME	1
i.1	AtivarCasa_Bluetooth	Geral	Casa	(Bluetooth == bt_casa)	MA	5
i.2	AtivarCasa_GPS	Geral	Casa	(GPS == “valido”) && GPS.Local == “Casa”)	MA	5
j.1	DesativarCasa_Bluetooth	Casa	Geral	(Bluetooth != bt_casa)	ME	5
j.2	DesativarCasa_GPS	Casa	Geral	(GPS == “valido”) && GPS.Local != “Casa”)	ME	5
k.1	AtivarEscritorio_Bluetooth	Geral	Escritorio	(Bluetooth == bt_escritorio)	VI	5
k.2	AtivarEscritorio_GPS	Geral	Escritorio	(GPS == “valido”) && GPS.Local == “Escritorio”)	VI	5
l.1	DesativarEscritorio_Bluetooth	Escritorio	Geral	(Bluetooth != bt_escritorio)	ME	5
l.2	DesativarEscritorio_GPS	Escritorio	Geral	(GPS == “valido”) && GPS.Local != “Escritorio”)	ME	5
m	AtivarReuniao	Escritorio	Reuniao	(Time >= reuniao_inicio) && (Bluetooth.lista.count ≥ 3	AV	4
n	DesativarReuniao	Reuniao	Escritorio	(Time >= reuniao_fim)	VI	4
o.1	AtivarSincronizacao_BTCasa	Geral	Sincronização	(Bluetooth == bt_casa)	ME	9
o.2	AtivarSincronizacao_BTEscrito	Geral	Sincronização	(Bluetooth == bt_escritorio)	ME	9
p	DesativarSincronizacao	Sincronização	Geral	(Bluetooth != bt_casa && Bluetooth != bt_escritorio)	ME	9

**Tabela 5.5:** A agenda de atividades de um usuário

Períodos	Domingo	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
08 às 12	Estudar						
12 às 17		Trabalhar	Trabalhar	Trabalhar	Trabalhar	Trabalhar	Trabalhar
14 às 18	Trabalhar na igreja						Estudar
18 às 19		Correr	Correr	Correr	Correr	Correr	
19 às 22	Ir à igreja	Estudar	Estudar	Estudar	Estudar	Estudar	Ir à igreja

No contexto de máquinas de estados para o teste, um dos critérios de teste proposto por Offutt et al. (2003) foi o Sequência completa (*Complete sequence*). Neste critério as sequências e transições são escolhidas pelo engenheiro de teste com base em sua experiência e conhecimento do domínio. No caso do *PhoneAdapter* (SA5) selecionado, a experiência do engenheiro de teste deve levar em consideração, além das transições entre os estados, a agenda e as propriedades presentes no SA.

Formalmente, sequências podem ser representadas como funções delta estendidas, ou seja  $\hat{\delta}$  (Hopcroft et al., 2000). Uma função delta  $\delta$  têm dois parâmetros, o primeiro é o estado (por exemplo  $q$ ) e o segundo é a entrada (por exemplo  $x$  e  $y \in w$ ). Assim, ao estender a função  $\delta$ , podem-se calcular sequências de transições entre estados, por exemplo na expressão 5.1:

$$\hat{\delta}(q, w) \vdash \delta(\hat{\delta}(q, x), y) \quad (5.1)$$

Uma sequência – baseada na restrição “De segunda a sexta-feira das 08 às 17 horas: O usuário trabalha” – que pode ser identificada na Máquina de *Mealy* da Figura 5.3 é a sequência que envolve o usuário aos estados “Geral” ( $q_0$ ), “Escritório” ( $q_6$ ) e “Reunião”. Formalmente essa sequência pode ser representada como:

Uma sequência – baseada na restrição “De segunda a sexta-feira das 08 às 17 horas: O usuário trabalha” – que pode ser identificada na máquina de *mealy* da Figura 5.3 é a sequência que envolve o usuário aos estados “Geral” ( $q_0$ ), “Escritório” ( $q_6$ ) e “Reunião” ( $q_7$ ). Formalmente, essa sequência pode ser representada como  $w = kmnl$ , tal que  $k$ ;  $m$ ;  $n$  e  $l$  são respectivamente as transições AtivarEscritorio, AtivarReunião, DesativarReunião e DesativarEscritório. Formalmente pode-se representar a sequência como:

- Decomposição da função delta estendida:

$$\hat{\delta}(q_0, kmnl)$$

$$\vdash \delta(\hat{\delta}(q_0, kmn), l)$$

$$\vdash \delta(\delta(\hat{\delta}(q_0, km), n), l)$$

$\vdash \delta(\delta(\delta(\hat{\delta}(q_0, k), m), n), l)$

- Execução da função delta estendida:

$\delta(\delta(\delta(\hat{\delta}(q_0, k), m), n), l)$

$\vdash \delta(\delta(\hat{\delta}(q_6, k), m), n)$

$\vdash \delta(\hat{\delta}(q_7, k), m)$

$\vdash \hat{\delta}(q_6, k)$

$\vdash q_0$

- Execução da função Lambda:

$\lambda(q_0, k)\lambda(q_6, m)\lambda(q_7, n)\lambda(q_6, l)$

Resultado: *VI AV VI ME*

## 5.4 Respostas às Questões [Q1]–[Q4] do GQM

Com a definição das estratégias de teste T e T\* e seleção de um modelo de especificação de teste, puderam-se responder às questões [Q1]–[Q4] do GQM:

**[Q1] - Qual abordagem de teste mais se relaciona aos desafios específicos?:**

a abordagem de teste S03 (Tse et al., 2004) foi elencada ao avaliar os desafios específicos das tabelas 4.6 e 4.7, com apoio da regra do ID 1 da Tabela 5.3. Tal abordagem destacou-se por se relacionar com 19 dos 34 desafios específicos caracterizados.

**[Q2] - Qual abordagem mais se relaciona às fases de teste?:**

a abordagem de teste S05 (Lu et al., 2006) foi elencada ao avaliar sua predominância nas fases *Projeto* e *Finalização* da Tabela 4.8, com apoio da regra do ID 2 da Tabela 5.3. Tal abordagem destacou-se por estar em quatro das cinco atividades da fase *projeto* e em uma das cinco atividades da fase *finalização*.

**[Q3] - Quais desafios específicos mais se relacionam aos SAs analisados?:**

elencou-se o conjunto de desafios específicos ao avaliar as tabelas 4.9 e 4.10, com apoio da regra do ID 3 da Tabela 5.3. Tal conjunto destacou-se por conter cinco ocorrências de desafios (Qtd. 5) nas tabelas 4.9 e 4.10, ou seja, os desafios específicos que estavam presentes nos cinco SAs analisados.

**[Q4] - Qual abordagem de teste mais se relaciona aos desafios específicos nos SAs?:**

a abordagem de teste S14 (Munoz e Baudry, 2009) foi elencada ao avaliar as abordagens de teste da Tabela 5.2, com apoio da regra do ID 4 da Tabela 5.3. Tal

abordagem de teste destacou-se por mais se relacionar aos desafios específicos presentes nos SAs analisados, contendo critérios/subsídios para geração de casos de teste.

Deste modo, respondendo às questões **Q1–Q4**, os objetivos **[G1]** e **[G2]** foram alcançados.

## 5.5 Execução dos Testes

A execução dos testes, que visa a responder as questões Q5 e Q6 do GQM apresentado na Figura 5.1, passou pela aplicação das abordagens S03, S05 e S14, cujos resultados obtidos foram medidos com a aplicação das métricas M7–M16, interpretadas de acordo com as regras definidas na Tabela 5.3.

### 5.5.1 Execução da Estratégias T

A aplicação das três abordagens de teste permitiu avaliar a efetividade e o custo das estratégias T e T\*. Ressalta-se que a máquina de *mealy* apresentada na Figura 5.2 embasou a aplicação das abordagens S03, S05 e S14. Além disso, a automatização da aplicação foi desenvolvida para auxiliar o processo de condução das estratégias T e T\*. Todas as abordagens de teste foram aplicadas no sistema *PhoneAdapter* (SA5). As descrições dos passos e resultados vêm a seguir.

A aplicação da Estratégia T envolveu a aplicação da abordagem S03 (Tse et al., 2004) no *PhoneAdapter* (SA5), utilizando a Máquina de *Mealy* como base.

#### Abordagem S03 (Tse et al., 2004)

**Passo 1:** Para aplicar a abordagem S03, considerou-se a restrição de domínio da agenda de um usuário, detalhada na Tabela 5.5, destacando-se:

1. Para cada relação metamórfica (RM), definiu-se um caminho que vai de um estado a outro por meio de transições.
2. Para cada RM, definiram-se predicados e oráculos: predicados cujas variáveis são obtidas de sensores de um sistema gerenciado e oráculos cujas assertivas são obtidas de um conjunto de saídas da máquina de *mealy*.

Na Figura 5.3 (b) apresentam-se dois loops de controle. O do retângulo cinza superior, responsável pela adaptação do sistema gerenciado com base nos contextos e regras, e o

retângulo cinza inferior, responsável pela identificação de novos dispositivos *bluetooth*. Assim, em cada transição da máquina de *mealy* da Figura 5.3 (a) tem-se a execução destes dois loops de controles da Figura 5.3 (b). Dado um estado (contexto) da máquina, o componente Monitor é responsável por obter possíveis valores dos sensores. O Analisador é responsável por identificar a melhor transição (regra) dos valores obtidos dos sensores, realizando a transição entre tais estados. O Executor é responsável por disparar a saída do estado, que na prática encaminha a adaptação para um atuador realizar a adaptação.

Assim, como introduzido na Seção 3.7.1, para uma RM definem-se  $n$  casos de teste relacionados. Sendo um caso de teste dado por um par ordenado  $(d, S(d))$  tal que  $d$  é elemento de um domínio de entrada  $D$  e  $S(d)$  é a saída esperada para a entrada  $d$ . Utiliza-se, assim, sequências formais como um ponto de partida para a criação de casos de teste. Como um exemplo, ilustra-se na sequência a criação de casos de teste para a RM Reunião.

Na Figura 5.3 (a) ilustra-se uma máquina de *mealy*, onde se pode classificar que uma transição entre estados da máquina utiliza  $d$  do domínio  $D$  e se têm para cada estado uma saída  $S(d)$ . Inicialmente, a partir das funções delta estendidas ( $\hat{\delta}$ ) e lambda ( $\lambda$ ), tem-se:  $d$  como elementos de entrada de possíveis transições e  $S(d)$  como os resultados de  $\hat{\delta}(q, a)$  e  $\lambda(q, a)$ , tal que  $q$  é o estado atual;  $a$  é a entrada do domínio (transição); a função  $\hat{\delta}$  retorna o próximo estado; e a função  $\lambda$  retorna a saída. Assim a RM “Reunião” poderia ser dada pelos casos de teste com base nas transições:

1.  $\lambda(q_0, k)$
2.  $\lambda(q_0, k)\lambda(q_6, m)$
3.  $\lambda(q_0, k)\lambda(q_6, m)\lambda(q_7, n)$
4.  $\lambda(q_0, k)\lambda(q_6, m)\lambda(q_7, n)\lambda(q_6, l)$

Assim, tem-se quatro (1, 2, 3 e 4) casos de teste que se remetem à RM Reunião. Observa-se que uma sequência de transições, juntas, formam um caso de teste. Os estados  $q_0$ ,  $q_6$  e  $q_7$ , representados em cor verde na Figura 5.3 (a), foram os utilizados na RM “Reunião”. Assim, quando ocorrer a transição  $k$  (ou seja, AtivarEscritorio) da MEF na Figura 5.3 (a) significa que os predicados  $k.1$  ou  $k.2$  na Tabela 5.4 (ou seja, “(Bluetooth == *bt\_escritorio*) || (GPS == valido && GPS.Local == Escritorio)”) eram verdadeiros. Além disso, ao ocorrer a transição  $m$  (ou seja, AtivarReunião) na Figura 5.3 (a) significa que o predicado  $m$  na Tabela 5.4 (ou seja, “(Time >= reuniaoinicio)”) era verdadeiro.

**Passo 2:** a partir das definições das RMs, puderam-se criar conjuntos de casos de teste para cada RM. A RM em que um de seus casos de teste não passar conterá um

**Tabela 5.6:** Casos de Teste criados com a Abordagem S03 (Tse et al., 2004)

ID RM	Descrição RM	Restrição de domínio/Sensor	ID	Caso de teste (Lambda)
1	Escritório	1,4 / usa bt, gps e time	1	q0_k1
			2	q0_k2
			3	q0_k1/q6_l1
			4	q0_k1/q6_l2
			5	q0_k2/q6_l1
			6	q0_k2/q6_l2
2	Reunião	1,4 / usa bt, gps e time	7	q0_k1/q6_m
			8	q0_k2/q6_m
			9	q0_k1/q6_m/q7_n
			10	q0_k2/q6_m/q7_n
			11	q0_k1/q6_m/q7_n/q6_l1
			12	q0_k1/q6_m/q7_n/q6_l2
			13	q0_k2/q6_m/q7_n/q6_l1
			14	q0_k2/q6_m/q7_n/q6_l2
3	Sincronizacao do Escritorio	1,4 / usa bt, gps e time	15	q0_o2
			16	q0_o2/q8_p
4	Na rua	2 / usa gps	17	q0_c
			18	q0_c/q1_b1
			19	q0_c/q1_b2
			20	q0_c/q1_b3
5	Corrida de Rua	2 / usa gps	21	q0_c/q1_a
			22	q0_c/q1_a/q2_d1
			23	q0_c/q1_a/q2_d2
			24	q0_c/q1_a/q2_d1/q1_b1
			25	q0_c/q1_a/q2_d1/q1_b2
			26	q0_c/q1_a/q2_d1/q1_b3
			27	q0_c/q1_a/q2_d2/q1_b1
			28	q0_c/q1_a/q2_d2/q1_b2
			29	q0_c/q1_a/q2_d2/q1_b3
			6	Casa
31	q0_i2			
7	Sincronização da Casa	3,5,7 / usa bt	32	q0_o1
			33	q0_o1/q8_p
8	Dirigindo	6,8,9 / usa gps	34	q0_e1
			35	q0_i1/q3_e2
			36	q0_i2/q3_e2
			37	q0_e1/q4_f
			38	q0_i1/q3_e2/q4_f
			39	q0_i2/q3_e2/q4_f
9	Dirigindo Rápido	6,8,9 / usa gps	40	q0_e1/q4_g
			41	q0_e1/q4_g/q5_h1
			42	q0_e1/q4_g/q5_h2
			43	q0_e1/q4_g/q5_h1/q4_f
			44	q0_e1/q4_g/q5_h2/q4_f
			45	q0_i1/q3_e2/q4_g
			46	q0_i1/q3_e2/q4_g/q5_h1
			47	q0_i1/q3_e2/q4_g/q5_h2
			48	q0_i1/q3_e2/q4_g/q5_h1/q4_f
			49	q0_i1/q3_e2/q4_g/q5_h2/q4_f

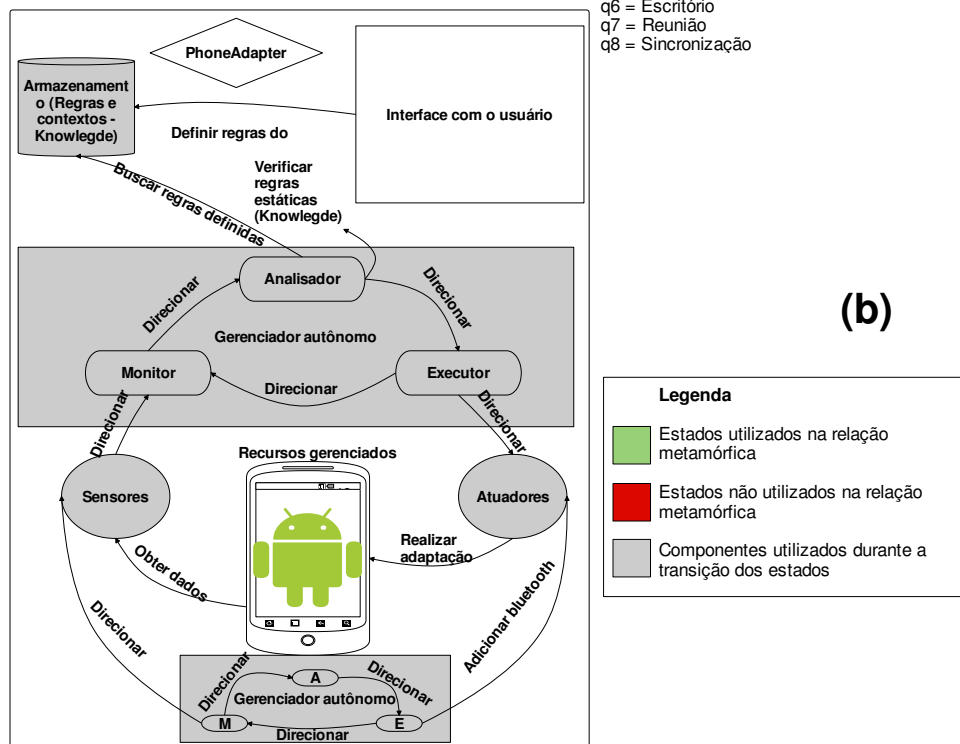
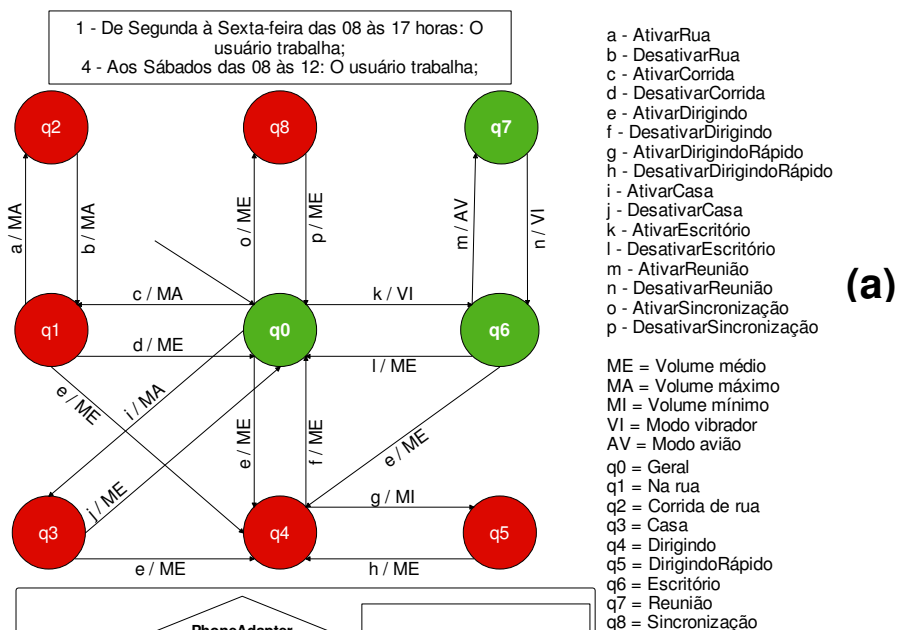


Figura 5.3: Representação para iniciar uma análise de Relação Metamórfica.

defeito. Neste sentido, o oráculo do teste pode ser evoluído junto com a execução de casos

de teste. Na sequência, na Tabela 5.6, são apresentadas as 9 RMs definidas: **1)** Escritório; **2)** Reunião; **3)** Sincronização do Escritório; **4)** Na rua; **5)** Corrida de Rua; **6)** Casa; **7)** Sincronização da Casa; **8)** Dirigindo; e, por fim, **9)** Dirigindo Rápido.

A partir das definições das RMs, puderam-se criar conjuntos de casos de teste para cada RM. A RM em que um de seus casos de teste não passar conterá um defeito. Assim, durante a definição e execução dos casos de teste, pode-se evoluir o conjunto de teste a fim de testar funcionalidades relacionadas à RM não previstas antes do teste. Por exemplo, se uma RM atribuir falha a um conjunto de teste A que tem relação de ordem com outro conjunto de teste B, o conjunto de teste B pode estar relacionado a outras funcionalidades não previstas na definição da RM. Portanto, novos casos de teste teriam que ser incluídos como relação de ordem, de modo a cobrir tais funcionalidades não previstas.

**Passo 3:** a partir da Agenda do usuário presente na Tabela 5.5, puderam-se gerar as restrições de domínio presentes na Tabela 5.7.

**Tabela 5.7:** Restrições de domínio baseadas na agenda do usuário

ID	Restrição de Domínio
1	De Segunda a Sexta-feira das 08 as 17 horas: O usuário trabalha
2	De Segunda a Sexta-feira das 18 as 19 horas: O usuário corre
3	De Segunda a Sexta-feira das 19 as 22 horas: O usuário estuda
4	Aos Sábados das 08 as 12: O usuário trabalha
5	Aos Sábados das 14 as 18: O usuário estuda
6	Aos Sábados das 19 as 22: O usuário vai à igreja
7	Aos Domingos das 08 as 12: O usuário estuda
8	Aos Domingos das 14 as 18: O usuário trabalha na igreja
9	Aos Domingos das 19 as 22: O usuário vai à igreja

Assim, com a máquina de *mealy* e as transições definidas, as RMs definidas e as restrições de domínios elencadas, aplicou-se a mesma estratégia utilizada para cobrir a RM “Reunião” apresentada na Figura 5.3 para as demais RMs.

Ressalta-se que, uma vez que para cada RM definiu-se um conjunto de casos de teste, se ao menos um caso de teste do conjunto falhou a falha foi atribuída aos demais casos de teste da RM.

### 5.5.2 Execução da Estratégia T\*

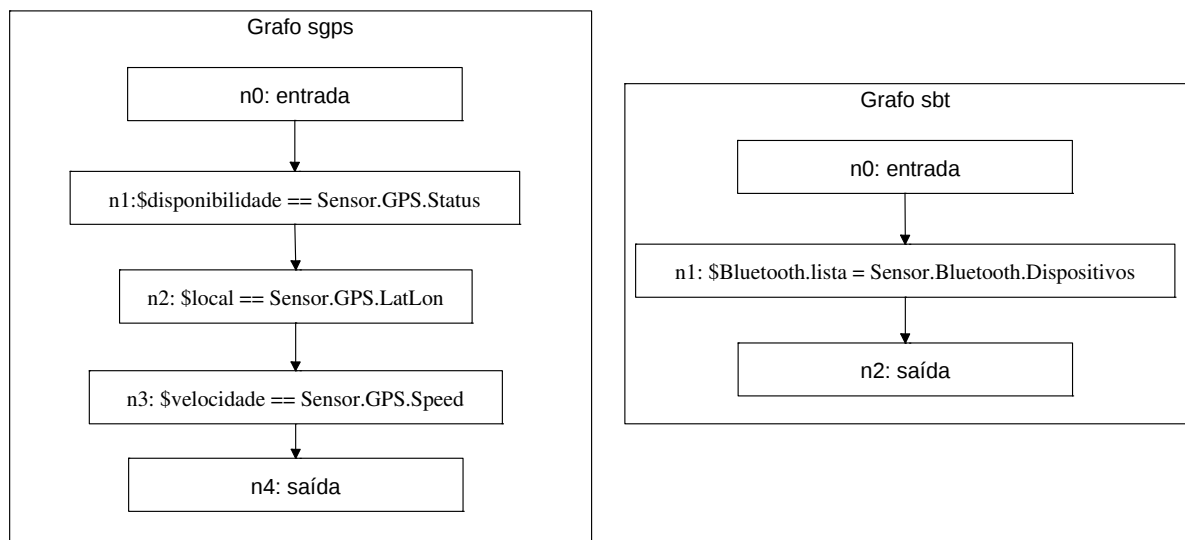
A aplicação da Estratégia T\* envolveu a aplicação da abordagem S03 (Tse et al., 2004), mesma aplicada na Estratégia T, da abordagem S05 (Lu et al., 2006) e da abordagem S14 (Munoz e Baudry, 2009). Estas abordagens foram aplicadas no mesmo SA utilizado na



aplicação da Estratégia T, o *PhoneAdapter* (SA5).

### Abordagem S05 (Lu et al., 2006)

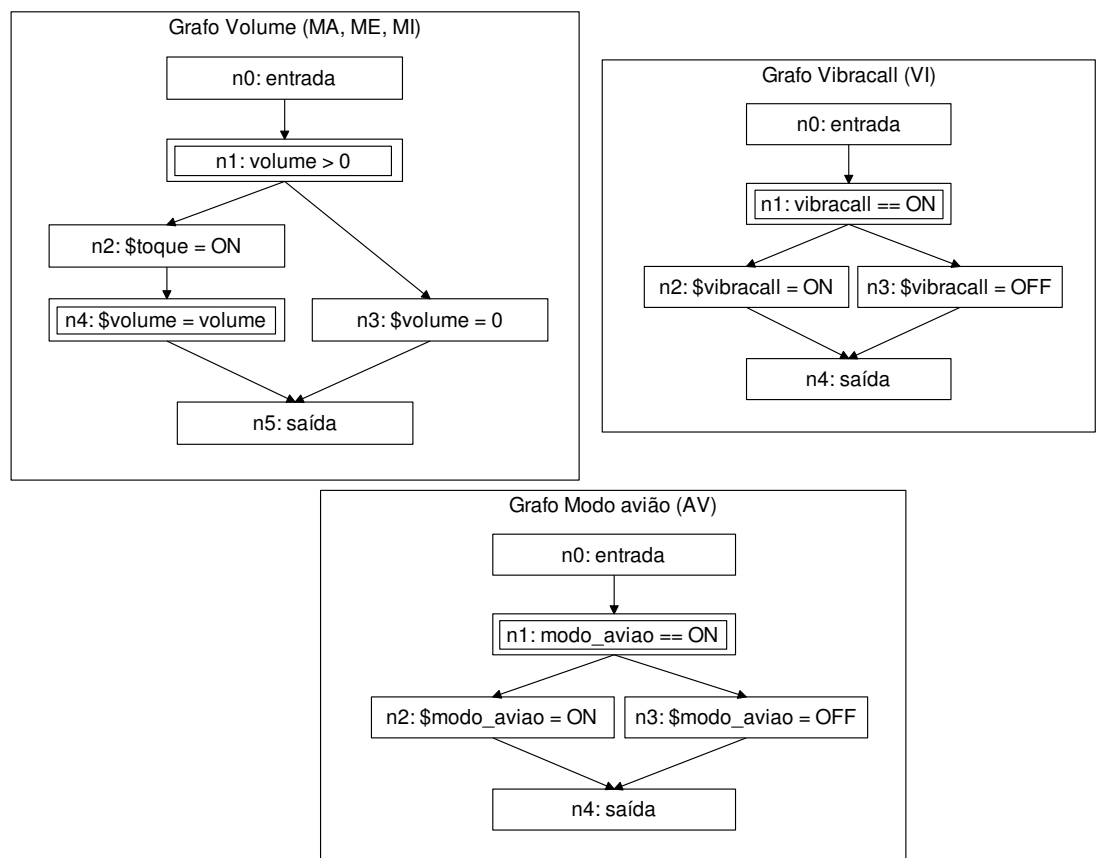
**Passo 1:** a fim de aplicar a abordagem S05 (Lu et al., 2006), introduzido anteriormente na Seção 3.7.2, identificaram-se as variáveis de contexto presentes no *PhoneAdapter* (SA5), especificamente, aquelas responsáveis por armazenar valores vindos dos sensores do SA. Com tais variáveis de contextos criaram-se dois CaFGs. O primeiro, intitulado Grafo *sgps*, contém as definições das seguintes variáveis de contexto: \$disponibilidade, responsável por armazenar a situação do sensor GPS; \$local, responsável por armazenar a latitude e a longitude do sensor GPS; \$velocidade, responsável por armazenar a velocidade do movimento do sensor GPS. O segundo, intitulado grafo *sbt*, contém a definição da variável de contexto \$Bluetooth.lista, responsável por armazenar a lista de dispositivos *Bluetooth* disponíveis para uso pelo dispositivo do SA. Ambos os grafos, *sgps* e *sbt*, podem ser visualizados na Figura 5.4. O primeiro foi composto por cinco nós e o segundo por três.



**Figura 5.4:** Grafos *sgps* e *sbt* das variáveis de contexto do *PhoneAdapter* (SA5).

**Passo 2:** identificaram-se as variáveis de contextos responsáveis por armazenar valores dos atuadores do SA. Com tais variáveis de contexto criaram-se mais três CaFGs. O primeiro, intitulado grafo *Volume*, contendo as definições das variáveis de contexto: \$volume, responsável por armazenar o volume do áudio do dispositivo; e a \$toque, responsável

por armazenar o tipo de toque do dispositivo. O segundo, intitulado grafo *Vibracall*, contento a definição da variável de contexto: `$vibracall`, responsável por armazenar se a funcionalidade *vibracall* está ligada ou desligada. O terceiro, por fim, intitulado grafo *Modo Avião*, contento a definição da variável de contexto: `$modo_avião`, responsável por armazenar se a funcionalidade *modo\_avião* está ligada ou desligada. Estes três grafos podem ser visualizados na Figura 5.5. O primeiro grafo, *volume*, foi composto por seis nós e os segundo e terceiro grafos foram compostos por cinco nós cada.



**Figura 5.5:** Grafos volume, vibracall e modo avião das variáveis de contexto do *PhoneAdapter* (SA5).

**Passo 3:** criou-se mais um CaFG que representou o *middleware* responsável por centralizar as variáveis de contextos dos sensores. Tal grafo, denominado *cm*, pode ser visualizado na Figura 5.6, na qual se fundamenta a abordagem S05. Observam-se nos grafos que alguns nós são compostos por retângulos duplos. Isso significa que as definições

das variáveis destes nós são utilizadas por outros grafos, que podem alterar os respectivos valores a qualquer momento e é a partir deste cenário que as associações entre os fluxos de dados foram identificadas. Um exemplo de associação é apresentado na Seção 3.7.2 na Figura 5.6, dadas pelas setas tracejadas, de modo que, uma variável utilizada no grafo *cm* pode estar sendo alterada no grafo *sgps*, por exemplo. No sistema que está sendo testado, a associação entre os grafos *sgps* e *cm* é dada pelas variáveis de contexto definidas no grafo *sgps* e utilizadas no grafo *cm*. Além disso, tais variáveis de contexto são utilizadas posteriormente nos grafos de situações.

**Passo 4:** a partir do trabalho de Sama et al. (2010a), identificaram-se as possíveis situações com as quais o SA poderia lidar a partir de cada um dos nove estados presentes na máquina de *mealy* da Figura 5.2. As situações puderam ser encontradas a partir das regras das transições entre os contextos presentes na Tabela 5.4, totalizando-se 28 situações. Para cada uma das situações criou-se um CaFG com o objetivo de identificar as associações entre os grafos. Na Figura 5.6, por exemplo, apresenta-se o grafo *s1* que se remete à situação 1. Cada um dos 28 grafos é similar a este, contendo 4 nós: um nó de entrada (n0), um nó de saída (n3), um nó da adaptação (n2) e um nó do predicado da regra, que utiliza as variáveis de contexto.

**Passo 5:** o critério de teste foi aplicado com base nos dois grafos de sensores, três grafos de atuadores, um grafo do *middleware* e 28 das situações, iniciou-se o processo da aplicação dos critérios de teste. Para a aplicação do critério Todas Situações, em um primeiro momento mapearam-se quais casos de teste da Tabela 5.6, criados na abordagem T, cobriam as 28 situações dos CaFGs. Tal mapeamento pode ser visualizado na Tabela D.1 do Apêndice D.

Observaram-se que das 28 situações, quatro situações não eram cobertas pela Estratégia T. Para a situação *s10* (AtivarDirigindo\_Escritorio) criaram-se dois casos de teste (50 e 51), para a situação *s11* (AtivarDirigindo\_NaRua) criou-se um caso de teste (52), para a situação *s18* (DesativarCasa\_Bluetooth) criaram-se dois casos de teste (53 e 54) e para a situação *s19* (DesativarCasa\_GPS) criaram-se dois casos de teste (55 e 56). Assim, chegou-se a um total de 56 casos de teste, somando-se as aplicações das abordagens S03 e S05.

**Passo 6:** de forma similar, para a aplicação do critério Todas Associações def-use, em um primeiro momento mapearam-se quais casos de teste das tabelas 5.6 e D.1 cobriam o critério Todas Associações def-use. Entretanto, verificou-se que todas já eram cobertas por pelo menos um dos casos de teste apresentados nas tabelas 5.6 e D.1. Portanto, não se criaram mais casos de teste para o critério Todas Associações def-use. Uma tabela de

mapeamento similar à Tabela D.1 foi criada, a qual pode ser visualizada no Apêndice C na Tabela D.2.

**Passo 7:** para a aplicação do critério Todas Associações de Pares, verificaram-se os dois grafos de sensores, três grafos de atuadores, um grafo do *middleware* e 28 grafos das situações, a fim de identificar chamadas cíclicas. Entretanto, não foi encontrada nenhuma ocorrência com tal característica. Por tal motivo, nenhum caso de teste foi criado para esse critério.

**Passo 8:** para a aplicação do critério Todas Situações Fora do Padrão, verificaram-se as 28 situações utilizadas no critério Todas Situações a fim de criar casos de teste que cobriam a negação de cada uma das situações. Como um exemplo, se uma determinada situação era testada como verdadeira, o oposto desta situação deveria ser validada como falsa. Este mesmo cenário foi aplicado às demais situações, tendo como resultado a Tabela D.3 presente no Apêndice D.

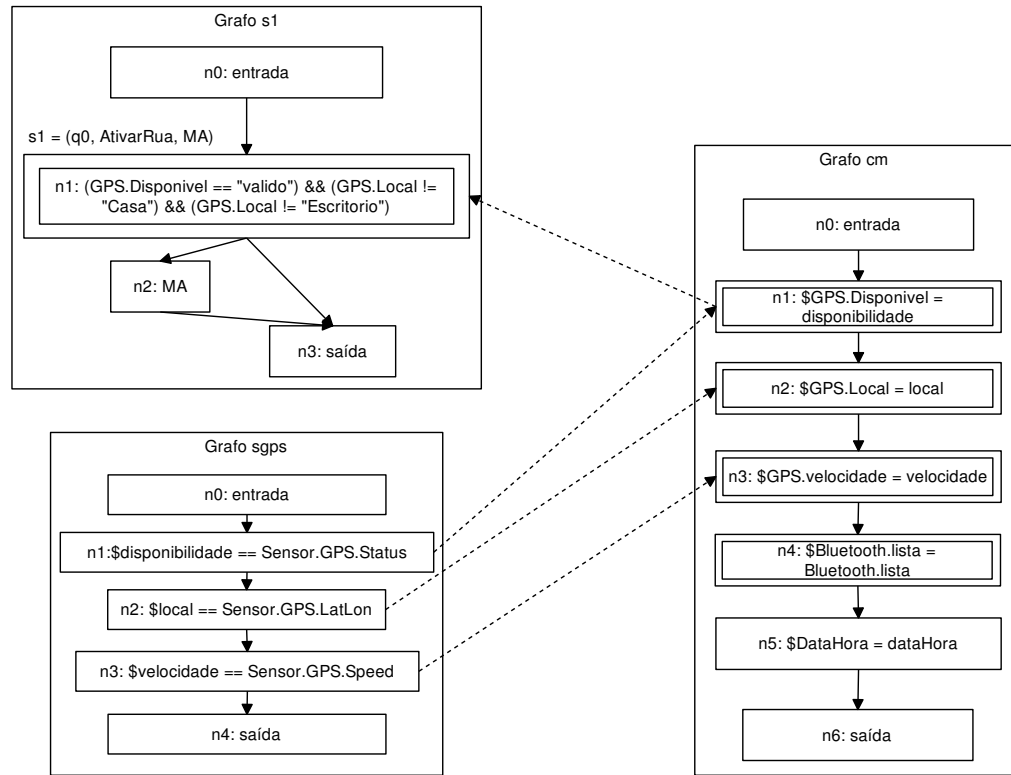
Ao final da aplicação da abordagem S05, somada à aplicação da abordagem S03, chegou-se a um total de 89 casos de teste

### **Abordagem S14 (Munoz e Baudry, 2009)**

A terceira abordagem da Estratégia  $T^*$  é a S14 (Munoz e Baudry, 2009). Conforme introduzido na Seção 3.7.3, tal abordagem visa a identificar mudanças abruptas em fluxos de contexto. Os autores partem do princípio de selecionar os fluxos de contexto em que mais os contextos sofrem oscilações nas mudanças. Tais oscilações remetem-se a mudanças constantes em variáveis de contexto. Assim, os autores propuseram testar tais oscilações a fim de encontrar falhas no sistema sob teste, sendo os seguintes passos realizados neste trabalho:

**Passo 1:** aplicação da técnica de teste *Pairwise* (Cohen et al., 1997) para gerar todos possíveis pares de instâncias de contexto, utilizando-a como um indicador de cobertura de teste. Neste trabalho, utilizou-se essa técnica nos estados da A-FSM especificada. Assim, verificaram-se as possíveis combinações de pares entre os nove estados da máquina de *mealy* da Figura 5.2. Para cada estado  $q_i$  verificaram-se seus caminhos para  $q_j$ , tal que  $i$  e  $j$  estão entre o intervalo fechado  $[0...8]$  cada um. Cada estado foi combinado, em par, com todos os demais. Como um exemplo, o estado  $q_0$  foi combinado com os estados  $q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7$  e  $q_8$ .

Observa-se que, ao realizar as combinações de pares entre os nove estados, obtém-se 81 combinações entre os estados (contextos). Assim, uma vez que não foram elencadas



**Figura 5.6:** Associações entre grafos utilizando a abordagem S05 (Lu et al., 2006).

combinações  $q_i$  para  $q_i$  (ele mesmo), por exemplo de  $q_0$  para  $q_0$ , para nove estados foram geradas 72 combinações.

**Passo 2:** as 28 transições elencadas na Tabela 5.4 foram mapeadas nas 72 combinações entre estados. Tal mapeamento levou em consideração o menor caminho entre um estado e outro, de modo que, por exemplo, do estado  $q_1$  ao estado  $q_3$  o menor caminho foi  $q_1 \rightarrow q_0 \rightarrow q_3$ . Neste menor caminho, por exemplo, puderam-se mapear 6 possibilidades de transições:  $b_1 \rightarrow i_1$ ,  $b_2 \rightarrow i_1$ ,  $b_3 \rightarrow i_1$ ,  $b_1 \rightarrow i_2$ ,  $b_2 \rightarrow i_2$  e  $b_3 \rightarrow i_2$ , sendo entre os estados  $q_1$  e  $q_0$  as transições  $b_1$ ,  $b_2$  e  $b_3$  e entre os estados  $q_0$  e  $q_3$  as transições  $i_1$  e  $i_2$ . Portanto, ao fim do mapeamento, teve-se um total de 269 possibilidades de transições.

**Passo 3:** calculou-se a diversidade de contexto para cada uma das 269 transições entre estados. Cada situação remete-se a um predicado que contém valores específicos para as variáveis de contexto do predicado, presentes na Tabela 5.8. A diversidade de contexto foi calculada com base nas mudanças das variáveis de contexto de uma situação para a outra. Como um exemplo, percorrendo os estados  $q_1 \rightarrow q_0 \rightarrow q_3$  nas transições  $b_3 \rightarrow i_2$ ,

originando-se da situação  $s4$  para a  $s17$  ( $s4 \rightarrow s17$ ) obteve-se a diversidade de contexto  $1 + 1 + 0 + 0 + 0$ , representando respectivamente as variáveis de contexto GPS.Disponivel, GPS.Local, GPS.Velocidade, Bluetooth.lista e Hora. Esse valor foi obtido pois apenas as duas primeiras variáveis de contexto sofreram alterações, de modo que a diversidade de contexto  $s4 \rightarrow s17$  é igual a 2.

**Passo 4:** aplicaram-se as duas equações (3.1 e 3.2) de definição de “terremotos artificiais” apresentadas na Seção 3.7.3 nas 269 transições entre estados, denominadas fluxos de contextos. Após a aplicação das equações, identificaram-se 13 possíveis “terromotos artificiais”, apresentados na Tabela 5.9.

Observa-se, na Tabela 5.9, que os 13 fluxos de contexto foram classificados como terremotos artificiais por satisfazerem as Equações 3.1 e 3.2 apresentadas na Seção 3.7.3. No campo Equação 3.1, analisou-se se o primeiro par da soma das diversidades de contexto era maior que o último par. Além disso, analisou-se se o primeiro par da soma das diversidades de contexto era menor que o último par. Como um exemplo, o fluxo de contexto  $s16 \rightarrow s19 \rightarrow s21 \rightarrow s24$  contém as diversidades de contexto 2, 1 e 4. O primeiro par ( $2 > 1$ ) é menor que o segundo par ( $1 < 4$ ), pois “ $2 - 1 = 1$ ” é menor que “ $4 - 1 = 3$ ”. No campo Equação 3.2, analisou-se a presença de oscilações dos valores, por exemplo, “ $2 > 1 < 4$ ” significa que oscilou de cima “2” pra baixo “1” depois para cima “4” novamente. Os 13 fluxos de contexto presente na Tabela 5.9 continham tais oscilações. Portanto, foram criados mais 13 casos de teste a fim de satisfazer os respectivos fluxos de contexto, chegando a um total de 102 casos de teste na Estratégia T\*.

**Tabela 5.8:** Valores das Variáveis de Contexto nas suas respectivas transições/situações.

ID	Descrição	GPS.Disponível	GPS.Local	GPS.Velocidade	Bluetooth.lista	Hora
s0	Inicial	não	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00
s1	AtivarRua (q0_c)	sim	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00
s2	DesativarRua_GPSNaoValido (q1_b1)	não	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00
s3	DesativarRua_LocalCasa (q1_b2)	não	Casa	0	bt_NaoCasaNemEscritorio	00:00:00
s4	DesativarRua_LocalEscritorio (q1_b3)	não	Escritório	0	bt_NaoCasaNemEscritorio	00:00:00
s5	AtivarCorrida (q1_a)	sim	NaoCasaNemEscritorio	6	bt_NaoCasaNemEscritorio	00:00:00
s6	DesativarCorrida_GPSNaoValido (q2_d1)	não	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00
s7	DesativarCorrida_VelocidadeMenorIgual5 (q2_d2)	não	NaoCasaNemEscritorio	5	bt_NaoCasaNemEscritorio	00:00:00
s8	AtivarDirigindo_Geral (q0_e)	não	NaoCasaNemEscritorio	0	bt_carro	00:00:00
s9	AtivarDirigindo_Casa (q3_e)	não	NaoCasaNemEscritorio	0	bt_carro	00:00:00
s10	AtivarDirigindo_Escritorio (q6_e)	não	NaoCasaNemEscritorio	0	bt_carro	00:00:00
s11	AtivarDirigindo_NaRua (q1_e)	não	NaoCasaNemEscritorio	0	bt_carro	00:00:00
s12	DesativarDirigindo (q4_f)	não	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00
s13	AtivarDirigindoRapido (q4_g)	sim	NaoCasaNemEscritorio	71	bt_NaoCasaNemEscritorio	00:00:00
s14	DesativarDirigindoRapido_GPSNaoValido (q5_h1)	não	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00
s15	DesativarDirigindoRapido_VelocidadeMenorIgual70 (q5_h2)	não	NaoCasaNemEscritorio	70	bt_NaoCasaNemEscritorio	00:00:00
s16	AtivarCasa_Bluetooth (q0_i1)	não	NaoCasaNemEscritorio	0	bt_casa	00:00:00
s17	AtivarCasa_GPS (q0_i2)	sim	Casa	0	bt_NaoCasaNemEscritorio	00:00:00
s18	DesativarCasa_Bluetooth (q3_j1)	não	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00
s19	DesativarCasa_GPS (q3_j2)	sim	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00
s20	AtivarEscritorio_Bluetooth (q0_k1)	não	NaoCasaNemEscritorio	0	bt_escritorio	00:00:00
s21	AtivarEscritorio_GPS (q0_k2)	sim	Escritório	0	bt_NaoCasaNemEscritorio	00:00:00
s22	DesativarEscritorio_Bluetooth (q6_l1)	não	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00
s23	DesativarEscritorio_GPS (q6_l2)	sim	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00
s24	AtivarReuniao (q6_m)	não	NaoCasaNemEscritorio	0	Count = 3	14:00:00
s25	DesativarReuniao (q7_n)	não	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	16:00:00
s26	AtivarSincronizacao_BTCasa (q0_o1)	não	NaoCasaNemEscritorio	0	bt_casa	00:00:00
s27	AtivarSincronizacao_BT_Escritorio (q0_o2)	não	NaoCasaNemEscritorio	0	bt_escritorio	00:00:00
s28	DesativarSincronizacao (q8_p)	não	NaoCasaNemEscritorio	0	bt_NaoCasaNemEscritorio	00:00:00

**Tabela 5.9:** Fluxos de contextos onde Terremotos Artificiais foram encontradas com a Abordagem S14 (Munoz e Baudry, 2009).

Estados	Transições	Situações	Diversidade de contexto	Soma	Diversidade de contexto	Soma	Diversidade de contexto	Soma	Diversidade de contexto	Soma	Total	Equação 3.1	Equação 3.2
$q3 \rightarrow q0 \rightarrow q6 \rightarrow q7$	$i1 \rightarrow j2 \rightarrow k2 \rightarrow m$	$s16 \rightarrow s19 \rightarrow s21 \rightarrow s24$	1 + 0 + 0 + 1 + 0	2	0 + 1 + 0 + 0 + 0	1	1 + 1 + 0 + 1 + 1	4			7	2 > 1; 1 < 4	1
$q5 \rightarrow q4 \rightarrow q0 \rightarrow q6 \rightarrow q7$	$h1 \rightarrow f \rightarrow k2 \rightarrow m$	$s13 \rightarrow s14 \rightarrow s12 \rightarrow s21 \rightarrow s24$	1 + 0 + 1 + 0 + 0	2	0 + 0 + 0 + 0 + 0	0	1 + 1 + 0 + 0 + 0	2	0 + 1 + 0 + 1 + 1	3	7	2 > 0; 2 < 3	1
$q3 \rightarrow q0 \rightarrow q4 \rightarrow q5$	$i2 \rightarrow j1 \rightarrow e1 \rightarrow g$	$s17 \rightarrow s18 \rightarrow s8 \rightarrow s13$	1 + 1 + 0 + 0 + 0	2	0 + 0 + 0 + 1 + 0	1	1 + 0 + 1 + 1 + 0	3			6	2 > 1; 1 < 3	1
$q6 \rightarrow q0 \rightarrow q4 \rightarrow q5$	$k2 \rightarrow l1 \rightarrow e1 \rightarrow g$	$s21 \rightarrow s22 \rightarrow s8 \rightarrow s13$	1 + 1 + 0 + 0 + 0	2	0 + 0 + 0 + 1 + 0	1	1 + 0 + 1 + 1 + 0	3			6	2 > 1; 1 < 3	1
$q5 \rightarrow q4 \rightarrow q0 \rightarrow q6 \rightarrow q7$	$h1 \rightarrow f \rightarrow k1 \rightarrow m$	$s13 \rightarrow s14 \rightarrow s12 \rightarrow s20 \rightarrow s24$	1 + 0 + 1 + 0 + 0	2	0 + 0 + 0 + 0 + 0	0	0 + 0 + 0 + 1 + 0	1	0 + 0 + 0 + 1 + 1	2	5	2 > 0; 1 < 2	1
$q3 \rightarrow q0 \rightarrow q1 \rightarrow q2$	$i1 \rightarrow j2 \rightarrow c \rightarrow a$	$s16 \rightarrow s19 \rightarrow s1 \rightarrow s5$	1 + 0 + 0 + 1 + 0	2	0 + 0 + 0 + 0 + 0	0	0 + 0 + 1 + 0 + 0	1			3	2 > 0; 0 < 1	1
$q5 \rightarrow q4 \rightarrow q0 \rightarrow q1$	$h1 \rightarrow f \rightarrow c$	$s13 \rightarrow s14 \rightarrow s12 \rightarrow s1$	1 + 0 + 1 + 0 + 0	2	0 + 0 + 0 + 0 + 0	0	1 + 0 + 0 + 0 + 0	1			3	2 > 0; 0 < 1	1
$q5 \rightarrow q4 \rightarrow q0 \rightarrow q3$	$h1 \rightarrow f \rightarrow i1$	$s13 \rightarrow s14 \rightarrow s12 \rightarrow s16$	1 + 0 + 1 + 0 + 0	2	0 + 0 + 0 + 0 + 0	0	0 + 0 + 0 + 1 + 0	1			3	2 > 0; 0 < 1	1
$q5 \rightarrow q4 \rightarrow q0 \rightarrow q6$	$h1 \rightarrow f \rightarrow k1$	$s13 \rightarrow s14 \rightarrow s12 \rightarrow s20$	1 + 0 + 1 + 0 + 0	2	0 + 0 + 0 + 0 + 0	0	0 + 0 + 0 + 1 + 0	1			3	2 > 0; 0 < 1	1
$q5 \rightarrow q4 \rightarrow q0 \rightarrow q8$	$h1 \rightarrow f \rightarrow o1$	$s13 \rightarrow s14 \rightarrow s12 \rightarrow s26$	1 + 0 + 1 + 0 + 0	2	0 + 0 + 0 + 0 + 0	0	0 + 0 + 0 + 1 + 0	1			3	2 > 0; 0 < 1	1
$q5 \rightarrow q4 \rightarrow q0 \rightarrow q8$	$h1 \rightarrow f \rightarrow o2$	$s13 \rightarrow s14 \rightarrow s12 \rightarrow s26$	1 + 0 + 1 + 0 + 0	2	0 + 0 + 0 + 0 + 0	0	0 + 0 + 0 + 1 + 0	1			3	2 > 0; 0 < 1	1
$q6 \rightarrow q0 \rightarrow q1 \rightarrow q2$	$k2 \rightarrow l2 \rightarrow c \rightarrow a$	$s20 \rightarrow s23 \rightarrow s1 \rightarrow s5$	1 + 0 + 0 + 1 + 0	2	0 + 0 + 0 + 0 + 0	0	0 + 0 + 1 + 0 + 0	1			3	2 > 0; 0 < 1	1



## 5.6 Avaliação das Estratégias T e T\*

Alguns detalhes a respeito dos defeitos encontrados e do desempenho obtido das estratégias T e T\* são apresentados na sequência, bem como uma sumarização do GQM utilizado no estudo exploratório. Ao fim da seção apresentam-se considerações a respeito dos desafios genéricos caracterizados e suas relações com as aplicações das estratégias T e T\*.

### 5.6.1 Defeitos Preliminares Encontrados

Antes de iniciar com a aplicação das estratégias de teste, encontraram-se três defeitos que não deixavam o *PhoneAdapter* (SA5) funcionar. Um destes defeitos pode ser visualizado no fragmento de código-fonte 5.1.

```
1 ...
2 case ContextType.GPS_ISVALID:
3     int value=Integer.parseInt(filter.contextValue);
4     boolean bool=(value==1)?true:false;
5     if(filter.contextOp==ContextOperator.EQUAL){
6         if(gpsAvailable!=bool){
7             f=false;
8             break;
9         }
10    }
11 ...
```

**Listagem de Código 5.1:** Defeito que foi corrigido para que o SA pudesse funcionar

Na linha 3 tenta-se converter o atributo *String* “contextValue” em um inteiro. Entretanto, tal atributo não continha valor que poderia ser convertido. A saber, ao invés de armazenar “0” ou “1”, armazenava-se “false” ou “true”, respectivamente. O mesmo defeito foi encontrado em dois outros fragmentos de código-fonte. Assim, para se testar, inicialmente modificaram-se tais fragmentos de código-fonte.

### 5.6.2 Defeitos e Falhas Encontradas pelas Abordagens

Inicialmente, executou-se a Estratégia T composta pela S03. Nesta execução, de um conjunto de 49 casos de teste, obtiveram-se quatro falhas externalizadas, as quais são apresentadas na Tabela 5.10 (IDs 1-4).

A abordagem S03 da estratégia T requer análise dos casos de teste juntamente com as respectivas RMs. Assim, os casos de teste que falharam (IDs 1-4) – da Tabela 5.10 – pertencem a duas respectivas RMs, presentes na Tabela 5.6. A RM de ID 3 é composta pelos casos de teste de IDs 15 e 16, ambos falharam. A RM de ID 7 é composta pelos

casos de teste de IDs 32 e 33, e, também, ambos falharam. Neste sentido, nenhuma outra falha foi adicionada à relação de falhas da Tabela 5.10.

Na aplicação da Estratégia T\*, de um conjunto de 102 casos de teste, obtiveram-se 16 falhas, sendo que falhas com ID's 5-16 foram somadas às falhas oriundas da aplicação da Estratégia T.

Das 16 falhas externalizadas, oito delas (IDs 1-4 e 13-16) originaram-se do mesmo defeito, presente no Algoritmo 5.2. Este fragmento de código-fonte tem por finalidade verificar, a partir da linha 4, se mais de uma regra foi satisfeita. Se sim, verificar-se-á qual das regras tem maior prioridade e, ainda, se as prioridades empatam. Assim, apenas uma regra deve ser selecionada. As falhas de IDs 1 e 2 foram externalizadas a partir dos casos de teste “test15\_T” e “test16\_T”. Esses casos de teste tiveram por objetivo testar, respectivamente, a ativação e desativação do estado Sincronização (do estado  $q_0$  para o  $q_8$  e depois para o  $q_0$ , novamente) que utilizam o *Bluetooth* do escritório do usuário. Entretanto, como existe outro predicado de adaptação (AtivarEscritorio) que também utiliza o *Bluetooth* do escritório do usuário e, ainda, tal predicado tem maior prioridade, então o predicado de adaptação “AtivarSincronizacao” não pôde ser ativado. O mesmo acontece para as falhas de IDs 3, 4, 13, 14, 15 e 16, entretanto, ao invés de se utilizar o *bluetooth* do escritório, utiliza-se o da casa do usuário.

```
1 // Se apenas uma regra for satisfeita
2 if(satisfiedRuleList.size() == 1){
3   ...
4 }else{ // Se mais que uma regra for satisfeita
5
6   // pegar a regra com maior prioridade
7   ArrayList<Rule> candidate=new ArrayList<Rule>();
8   int min=Integer.MAX_VALUE;
9   for(int j=0;j<satisfiedRuleList.size();j++){
10      if(satisfiedRuleList.get(j).priority<min){
11         min=satisfiedRuleList.get(j).priority;
12      }
13   }
14   for(int j=0;j<satisfiedRuleList.size();j++){
15      if(satisfiedRuleList.get(j).priority==min){
16         candidate.add(satisfiedRuleList.get(j));
17      }
18   }
19   // tamanho nao pode ser igual a zero, pelo menos um
20   if(candidate.size()==1){
21     /**
22     * realizar a adaptacao,
23     * atualizar o perfil atual e
24     * a lista de regra atual.
25     */
26     ...
27   }else{
```

```

28     // escolher aleatoriamente
29     Random rand=new Random();
30     int choice=rand.nextInt(candidate.size());
31     ...
32 }
33 }

```

### Listagem de Código 5.2: Defeito de Predicado Morto

O defeito do fragmento de código 5.2 foi caracterizado como “Defeito de predicado morto” (Sama et al., 2010a), uma vez que o predicado de adaptação “AtivarSincronizacao” nunca será ativado, pois sua prioridade é menor que os predicados “AtivarEscritorio” e “AtivarCasa”.

As demais falhas externalizadas (5-12) originaram-se dos defeitos presentes entre as linhas 3 e 13, presentes no Algoritmo 5.3. Este fragmento de código tem por objetivo obter os valores das variáveis de contexto, encontrar o predicado de adaptação respectivo e, na sequência, realizar a adaptação do SA.

```

1 ...
2 // pega os dados de contexto da intent
3 boolean gpsAvailable =
4     i.getBooleanExtra(ContextName.GPS_AVAILABLE, false);
5 String gpsLocation =
6     i.getStringExtra(ContextName.GPS_LOCATION);
7 double gpsSpeed =
8     i.getDoubleExtra(ContextName.GPS_SPEED, 0.0);
9 String[] deviceMacList =
10     i.getStringArrayExtra(ContextName.BT_DEVICE_LIST);
11 int count = deviceMacList.length;
12 String time = i.getStringExtra(ContextName.TIME);
13 String weekday = i.getStringExtra(ContextName.WEEKDAY);
14 ...
15 case ContextType.BLUETOOTH:
16     ...
17     if(filter.contextOp == ContextOperator.NOTEQUAL){
18         /* se a lista mac contem o valor de contexto, f = false */
19         if(macListContainsMac(deviceMacList, filter.contextValue)){
20             f = false;
21             break;
22         }
23     }
24     break;
25 ...

```

### Listagem de Código 5.3: Defeitos de Ativação não Determinística

As falhas das IDs 5-12 foram externalizadas a partir dos casos de teste test59-HT, test61-HT, test62-HT, test71-HT, test72-HT, test77-HT, test78-HT e test82-HT. Estes casos de teste originaram-se da abordagem S05 (Lu et al., 2006), executada na Estratégia T\*. Em tal etapa, geraram-se aqueles casos de teste para testar as negações das situações

**Tabela 5.10:** Falhas externalizadas na aplicação das Estratégias T e T\*.

ID	Caso de teste	Descrição	Esperado	Motivo
1	test15_T	Ativação de uma regra diferente	AtivarSincronizacao	Transição para um estado diferente do esperado.
2	test16_T	Ativação de uma regra diferente	AtivarSincronizacao	Transição para um estado diferente do esperado.
3	test32_T	Ativação de uma regra diferente	AtivarSincronizacao	Transição para um estado diferente do esperado.
4	test33_T	Ativação de uma regra diferente	AtivarSincronizacao	Transição para um estado diferente do esperado.
5	test59_HT	Ativação de uma regra diferente	Não desativar Rua	Transição para dois estados ao mesmo tempo.
6	test61_HT	Ativação de uma regra diferente	Não desativar Corrida	Transição para dois estados ao mesmo tempo.
7	test62_HT	Ativação de uma regra diferente	Não desativar Corrida	Transição para dois estados ao mesmo tempo.
8	test71_HT	Ativação de uma regra diferente	Não desativar Dirigindo rápido	Transição para dois estados ao mesmo tempo.
9	test72_HT	Ativação de uma regra diferente	Não desativar Dirigindo rápido	Transição para dois estados ao mesmo tempo.
10	test77_HT	Ativação de uma regra diferente	Não desativar casa	Transição para dois estados ao mesmo tempo.
11	test78_HT	Ativação de uma regra diferente	Não desativar casa	Transição para dois estados ao mesmo tempo.
12	test82_HT	Ativação de uma regra diferente	Não desativar escritório	Transição para dois estados ao mesmo tempo.
13	test88_HT	Ativação de uma regra diferente	AtivarSincronizacao	Transição para um estado diferente do esperado.
14	test89_HT	Ativação de uma regra diferente	AtivarSincronizacao	Transição para um estado diferente do esperado.
15	test99_HT2	Ativação de uma regra diferente	AtivarSincronizacao	Transição para um estado diferente do esperado.
16	test100_HT2	Ativação de uma regra diferente	AtivarSincronizacao	Transição para um estado diferente do esperado.

do SA, de modo que, por exemplo, se o predicado de adaptação “Bluetooth diferente de casa e escritório” ativava o estado  $q_1$  (Na rua), deveria-se gerar um caso de teste que verificava se “Bluetooth igual a casa ou a escritório” não ativava o estado  $q_1$  (Na rua). Com isso, entre as linhas 3 e 13 do fragmento de código do Algoritmo 5.3, identificaram-se defeitos caracterizados como “Defeitos de Ativações não determinística”(Sama et al., 2010a).

Uma vez que para cada RM, definida com a abordagem S03, definiu-se um conjunto de casos de teste (na Tabela 5.6 do ID 1 ao 49), se ao menos um caso de teste do conjunto falhou a falha foi atribuída aos demais casos de teste da RM. Ao todo foram nove RMs (Escritório, Reunião, Sincronização do Escritório, Na rua, Corrida de Rua, Casa, Sincronização

da Casa, Dirigindo e Dirigindo Rápido). Como um exemplo, a RM 3-Reunião teve 8 casos de teste executados (do ID 7 ao 14) e, desses, todos passaram, então a RM Reunião passou. Entretanto, as RMs 3-Sincronização do Escritório e 7-Sincronização da Casa não passaram pois em seus conjuntos de casos de teste pelo menos um de cada conjunto não passou.

Na sequência são apresentadas as análises realizadas da aplicação e execução das estratégias de teste, visando a responder as questões [Q5] e [Q6] do GQM definido. Ressalta-se que as interpretações de cada resultado, o qual é destacado entre “parênteses”, podem ser visualizadas na Tabela 5.3.

**Execução da Estratégia T:** com esta execução obtiveram-se (i) uma média de  $\frac{49}{28}$  (1,75) casos de teste por situação testada (ID 11); (ii) uma média de  $\frac{49}{4}$  (12,25) casos de teste por falha externalizada (ID 5); e, por fim, (iii) uma média de  $\frac{4}{1}$  (12,25) falhas externalizadas por defeitos encontrados (ID 8).

**Execução da segunda abordagem da Estratégia T\*:** com estas inclusão e execução obtiveram-se (i) uma média de  $\frac{89}{28}$  (3,18) casos de teste por situação testada (ID 12); (ii) uma média de  $\frac{89}{14}$  (6,36) casos de teste por falha externalizada (ID 6); e, por fim, (ii) uma média de  $\frac{14}{4}$  (3,5) falhas externalizadas por defeitos encontrados (ID 9).

**Execução da terceira abordagem da Estratégia T\*:** com estas inclusão e execução obtiveram-se (i) uma média de  $\frac{102}{28}$  (3,64) casos de teste por situação testada (ID 13); (ii) uma média de  $\frac{102}{16}$  (6,38) casos de teste por falha externalizada (ID 7); e, por fim, (iii) uma média de  $\frac{16}{4}$  (4) falhas externalizadas por defeitos encontrados (ID 10).

## 5.7 Respostas às Questões [Q5] e [Q6] do GQM

Com a avaliação das estratégias de teste T e T\*, puderam-se responder às questões [Q5] e [Q6] do GQM. Assim, o objetivo [G3] foi alcançado.

**Resposta à questão [Q5] - Qual a efetividade da aplicação das estratégias T e T\*?** a estratégia de teste T\* mostrou-se mais efetiva que a estratégia de teste T, pois: enquanto que T\* externalizou 16 falhas a T externalizou quatro falhas. As 16 falhas externalizadas pela estratégia de teste T\* originaram-se de quatro defeitos, enquanto que as quatro falhas externalizadas pela estratégia de teste T originaram-se de um defeito. Porém, durante a execução da estratégia T\*, ressalta-se que a inclusão da abordagem S14 (Munoz e Baudry, 2009) (após ter aplicadas as abordagens S03 e S05) não aumentou a efetividade da estratégia de teste T\*. As falhas externalizadas com a inclusão da abordagem S14 já haviam sido externalizadas durante a execução da estratégia de teste T.

**Resposta à questão [Q6] - Qual o custo da aplicação das estratégias T e T\*?**

a estratégia de teste T\* mostrou-se mais custosa que a estratégia de teste T, pois para os mesmos 28 cenários: enquanto que com a T\* criaram-se 102 casos de teste, com a T criaram-se 49 casos de teste. O custo de aplicação aumentou significativamente (de 49 para 89) com a primeira inclusão de casos e teste, utilizando-se a abordagem de teste S05 (Lu et al., 2006). Após a inclusão das duas abordagens (S03 e S05), o custo de se incluir os casos de teste com a abordagem S14 foi baixo (de 89 para 102). Apesar do custo, a estratégia T\* mostrou-se superior a estratégia T: enquanto que com a T\* revelaram-se quatro defeitos, com a T revelou-se somente um defeito.

## 5.8 Discussões Adicionais

### 5.8.1 Automatização da aplicação das Estratégias T e T\*:

A decomposição e a execução da função delta estendida ( $\hat{\delta}$ ) e a execução de funções lambda ( $\lambda$ ), apresentadas durante a seleção e revisão do modelo de especificação de teste, subsidiaram a automatização da aplicação das abordagens de teste neste trabalho. Neste sentido, um caso de teste foi composto por  $n$  funções lambda, de modo que cada uma delas executava um passo do caso de teste. As sequências de lambdas foram geradas a partir dos 102 casos de teste resultantes das Estratégias T e T\*.

Utilizaram-se recursos sobre o teste para dispositivos Android disponíveis no portal “*Developers Android / Best Practices for Testing*”<sup>1</sup> para embasar o desenvolvimento da automatização.

Após a definição do conjunto de teste a ser implementado, bem como o conjunto de funções lambdas que cada caso de teste iria conter, adaptou-se o código fonte do *PhoneAdapter* (SA5) selecionado a fim do mesmo ser “testável” (objetos que capturavam a execução dos testes). Assim, criaram-se métodos que, a partir da estrutura das funções lambdas, realizavam as chamadas de métodos presentes no *PhoneAdapter* (SA5). Ressalta-se que o componente “*AdaptationManager*” do *PhoneAdapter* (SA5) foi o selecionado para ser testado, uma vez que nele estava presente toda regra de negócio, contendo características de analisadores e executores. O componente “*AdaptationManager*” é um serviço do *PhoneAdapter* (SA5) responsável por obter dados de variáveis de contexto, encaminhadas pelo componente “*ContextManager*” que contém características de Monitor, e realizar as

---

<sup>1</sup><http://developer.android.com/intl/pt-br/training/testing/index.html> – Disponível em Abril/2016

adaptações do SA que no caso do *PhoneAdapter* (SA5) são mudanças no volume e no tipo de toque do dispositivo *android*.

O fragmento de código-fonte referente a um dos casos de teste criado pode ser visualizado no Algoritmo 5.4. O atributo “@Test” elenca o método para ser executado via *jUnit*, como caso de teste. O caso de teste, neste caso, é formado por quatro passos de teste (funções lambda  $\lambda$ ). Cada execução de um passo de teste gera um objeto *SaidaDoCasoDeTeste*, verificando-se se o objeto contém os resultados esperados para o passo de teste. Como um exemplo, o método “*lambda\_q0\_k1()*” testa sair do estado  $q_0$  para o estado  $q_6$  com o predicado de adaptação “k1” (ativar escritório por *bluetooth*). Assim, o método “AtivarEscritorio” verifica se o passo do teste passou. Os demais passos de teste são realizados do mesmo modo. Portanto o caso de teste “*testCase11*” só passará se todos os passos de teste não falharem.

```
1
2  /*
3  * lambda(q0,k)lambda(q6,m)lambda(q7,n)lambda(q6,l)
4  * q0 = Geral, q6 = Escritorio, q7 = Reuniao
5  * k = AtivarEscritorio, m = AtivarReuniao,
6  * n = DesativarReuniao, l = DesativarEscritorio
7  * */
8  @Test
9  public void testCase11(){
10     SaidaDoCasoDeTeste saida1 = lambda_q0_k1();
11     boolean resultado1 = AtivarEscritorio(saida1);
12     SaidaDoCasoDeTeste saida2 = lambda_q6_m();
13     boolean resultado2 = AtivarReuniao(saida2);
14     SaidaDoCasoDeTeste saida3 = lambda_q7_n();
15     boolean resultado3 = DesativarReuniao(saida3);
16     SaidaDoCasoDeTeste saida4 = lambda_q6_l1();
17     boolean resultado4 = DesativarEscritorio(saida4);
18
19     return (resultado1 && resultado2 &&
20     resultado3 && resultado4);
21 }
```

**Listagem de Código 5.4:** Um fragmento de código-fonte referente a um dos casos de teste criados.

A execução dos casos de teste pode ser visualizada na Figura 5.7. A ilustração apresenta uma das janelas disponíveis no IDE Android Studio<sup>2</sup>.

Observa-se, no canto superior direito da figura, que se apresentam o conjunto de casos de teste executados (102), o conjunto de casos de teste que falharam (16) e o tempo que levou a execução dos testes (59.27 segundos). Na janela apresentam-se detalhes da execução dos testes, indicando-se o dispositivo *android* utilizado

---

<sup>2</sup><http://developer.android.com/intl/pt-br/tools/studio/index.html> – Disponível em Abril/2016

```

Done: 102 of 102 Failed: 16 (59.27 s)
Testing started at 10:17 ...
Target device: samsung-gt_p5200-3200118b11f87000
Installing APK: /home/bento/Dropbox/Mestrado/Testing/PhoneAdater_ServiceTestRule/MainActivity/a
Uploading file to: /data/local/tmp/edu.hkust.cse.phoneAdapter
Installing edu.hkust.cse.phoneAdapter
DEVICE SHELL COMMAND: pm install -r "/data/local/tmp/edu.hkust.cse.phoneAdapter"
  pkg: /data/local/tmp/edu.hkust.cse.phoneAdapter
Success

Installing APK: /home/bento/Dropbox/Mestrado/Testing/PhoneAdater_ServiceTestRule/MainActivity/a
Uploading file to: /data/local/tmp/edu.hkust.cse.phoneAdapter.test
Installing edu.hkust.cse.phoneAdapter.test
DEVICE SHELL COMMAND: pm install -r "/data/local/tmp/edu.hkust.cse.phoneAdapter.test"
  pkg: /data/local/tmp/edu.hkust.cse.phoneAdapter.test
Success

Running tests
Test running started

```

Test	Time elapsed	Results
test01_T	0.656 s	Passed
test02_T	0.527 s	Passed
test03_T	0.579 s	Passed
test04_T	0.382 s	Passed
test05_T	0.504 s	Passed
test06_T	0.552 s	Passed
test07_T	0.326 s	Passed
test08_T	0.376 s	Passed
test09_T	0.376 s	Passed
test100_H	0.527 s	Error
test101_H	0.351 s	Passed
test102_H	0.377 s	Passed
test10_T	0.326 s	Passed
test11_T	0.351 s	Passed
test12_T	0.753 s	Passed
test13_T	0.552 s	Passed

Figura 5.7: Automatização das Estratégias de teste Executada.

(*samsung-gt\_p5200-3200118b11f87000*), a instalação do *PhoneAdapter* (SA5) no dispositivo (“*Installing edu.hkust.cse.phoneAdapter*”) e a instalação da automatização implementada (“*Installing edu.hkust.cse.phoneAdapter.test*”). Por fim, os testes foram executados no dispositivo e os resultados foram coletados.

Ressalta-se que o código-fonte da automatização apresentada foi disponibilizado no url <https://github.com/californi/dissertation> para reprodução dos resultados. Criou-se o projeto de teste utilizando o IDE Android Studio, portanto, aconselha-se o uso de tal IDE. Ao abrir o projeto, tem-se no pacote “*edu.hkust.cse.phoneAdapter.test*” a classe “*CriacaoDoAmbienteDeTeste*”. A instanciação e a execução desta classe irão gerar o ambiente de teste (que inclui a definição da máquina de estados e dos predicados de adaptação). As estratégias T e T\* podem ser reproduzidas com a instanciação e execução da classe “*ExecucaoAbordagemTeste*”. Tal execução irá aplicar os 102 casos de teste criados.

### 5.8.2 Desafios Observados com a Aplicação das Abordagens

Com a aplicação das estratégias de teste, obtiveram-se da informações mais concretas a respeito das abordagens e desafios. Na Seção 4.5 apresentaram-se as abordagens de teste relacionadas aos desafios, a fim de mitigar a aplicação do teste. Observou-se, também, uma relação das abordagens utilizadas a alguns dos desafios genéricos (C-1 a C-9) e, assim, tem-se que:

**C-1 – Como lidar com o crescimento exponencial de configurações de sistema que devem ser testadas**



Na aplicação da abordagem S03 definiram-se os possíveis estados que o *PhoneAdapter* (SA5) poderia operar e, ainda, uma restrição de domínio baseada numa agenda do usuário. Apesar de tal medida não impedir o “crescimento exponencial” de possíveis configurações, com uma máquina de estados finita e restrições de domínio pôde-se prover um conjunto reduzido de configurações (por exemplo, a agenda do usuário promoveu uma limitação por dia da semana e horário). Além disso, com a abordagem S03 definem-se RMs que contém conjuntos de casos de teste relacionados, de modo que, se um falhar os casos de teste relacionados a respectiva RM também falham.

Na aplicação da abordagem S05, inicialmente, definiram-se as possíveis situações que o *PhoneAdapter* (SA5) poderia estar. A partir deste momento, a geração dos casos de teste bem como a aplicação dos critérios propostos tiveram tais situações como ponto de partida. Assim, pôde-se prover um conjunto não exponencial de configurações.

Na aplicação da abordagem S14, utilizaram-se duas equações a fim de selecionar apenas os caminhos que contém oscilações entre as diversidades de contexto. Com isso, definiu-se um número reduzido de casos de teste.

## **C-2 – Como garantir a corretude de sistemas que nunca foram testados**

O mesmo cenário do C-1, entretanto, garantir a corretude de software como um todo não é trivial (Myers et al., 2011). Durante a abordagem S03, proveu-se uma diminuição do escopo do teste com foco nas restrições de domínio e, com a abordagem S05, aplicaram-se quatro critérios de teste estrutural. Com isso apresentou-se um nível de qualidade do sistema. O mesmo vale para a aplicação da abordagem S14, entretanto, apenas são testadas as oscilações que o SA pode conter em sua execução.

## **C-3 – Como detectar e evitar configurações incorretas definidas no tempo de execução**

O teste em tempo de execução não foi o foco deste trabalho, entretanto, limitando o escopo do teste (abordagens S03 e S05), na etapa de projeto, pode-se ainda obter um nível de qualidade do sistema (Myers et al., 2011). Para a aplicação da abordagem S14, encontrar todas configurações incorretas não foi o foco, entretanto, aplicaram-se oscilações no SA para tentar encontrar possíveis caminhos com oscilações que poderiam externalizar falhas.

#### **C-4 – Como testar SAs que executam em um ambiente distribuído e heterogêneo**

A abordagem S03 permitiu a visualização do *PhoneAdapter* (SA5) como uma caixa preta, então com o uso da máquina de estados, levou-se em consideração apenas as transições entre os estados, independentemente se estivessem várias *threads* sendo executadas. Além disso, a abordagem S05 permitiu uma análise que utilizou associações entre os fluxos de dados, com o objetivo de testar o relacionamento entre os fluxos bem como mitigar possíveis variáveis de contexto com dados inconsistentes.

#### **C-5 – Como lidar com interferência de usuário na configuração do sistema**

Com o uso da máquina de estados como modelo e a restrição da agenda do usuário, levou-se em consideração um determinado perfil de usuário. Com relação à abordagem S14, as oscilações simulam várias mudanças que o contexto do usuário pode gerar para o SA.

#### **C-6 – Como antecipar todas mudanças de contextos relevantes e quando podem impactar no comportamento de SAs**

Este desafio genérico compartilha do mesmo cenário dos desafios genéricos C-5, C-2 e C-1, para a abordagem S03. Além disso, com a definição das possíveis situações do SA e aplicação dos quatro critérios de teste ao menos as principais situações do SA foram antecipadas (abordagem S05). O mesmo vale para a abordagem S14, entretanto, ao invés de testar todas mudanças possíveis, testam-se todas oscilações de acordo com as transições entre os estados do SA.

#### **C-7 – Como lidar com fluxo de dados e de controle que são dependentes de contexto em SAs**

A abordagem S03, de fato, não utilizou informação estrutural do código-fonte. Entretanto, utilizou-se o conceito de múltiplas entradas e múltiplas saídas, relacionando-se entradas e saídas, com base na relação metamórfica vinda do contexto do SA. Já a abordagem S05, além de promover o conceito de associações entre fluxos de dados, promoveu quatro critérios de teste estrutural que utilizam informações de contexto em suas aplicações.

Com relação à abordagem S14, de fato, não se utilizou informação estrutural do código-fonte. Entretanto, utilizou-se a diversidade de contexto que implicou na avaliação

das combinações entre estados do SA e diversidade de contexto dos caminhos que permearam os estados do SA.

### **C-8 – Como manter casos de teste atualizados e precisos. Além disso, como manter rastreabilidade entre casos de teste, requisitos e componentes em um ambiente em mudanças**

Com o uso da abordagem S03, a geração dos casos de teste foi realizada a partir das relações metamórficas. Assim, ao atualizar os casos de teste podem-se levar em consideração a relação metamórfica que os originaram.

### **C-9 – Como determinar quais sensores e frequência do monitoramento de dados de SAs, em um ambiente de execução que deve ser observado**

Com o uso da abordagem S03, durante a definição dos casos de teste, levaram-se em consideração que em cada estado continham sensores e atuadores. Um exemplo da modelagem pode ser visualizado na Figura 5.3, onde a cada transição obtêm-se os dados dos sensores e encaminha-os aos atuadores.

## **5.8.3 Desafios Não Observados com a Aplicação das Abordagens**

O relacionamento entre abordagens e desafios, apresentado anteriormente na Seção 4.5, destacou que os desafios C-10, C-11 e C-12 não estavam relacionados às abordagens da RS, entretanto, algumas considerações podem ser visualizadas na sequência.

### **C-10 – Como simular um ambiente de execução real e com carga de trabalho**

O uso de um ambiente de execução real e com carga de trabalho, de fato, não se relacionou a este trabalho. O principal motivo foi com relação ao *PhoneAdapter* (SA5) utilizado na avaliação, trata-se de um código-fonte para prova e conceito não comercial. Entretanto, com a automatização desenvolvida neste trabalho, utilizando-se como base uma máquina de *mealy*, 102 casos de teste levaram 59.27 segundos para serem executados.

## C-11 – Como gerar casos de teste automaticamente para um ambiente em mudanças

Apesar de não fazer parte do escopo deste trabalho (e também das abordagens S03, S05 e S14 não se relacionarem ao C-11), observou-se um padrão com relação a uma possível geração de casos de teste. A criação dos casos de teste levou em consideração as transições entre estados em uma máquina de *mealy*. O *PhoneAdapter* (SA5), a todo momento, sempre gera os próximos possíveis estados que ele pode operar. Com isso, identificou-se uma possibilidade de implementar uma abordagem para geração de casos de teste a partir da execução do *PhoneAdapter* (SA5), coletando informações das variáveis de contexto e execução do código-fonte.

## C-12 – Como realizar verificação formal do comportamento adaptativo do sistema

Apesar das abordagens S03, S05 e S14 não se relacionarem ao C-12, o desenvolvimento da automatização foi todo baseado no uso de uma máquina de *mealy*. Com isso, pôde-se, formalmente, representar a execução dos casos de teste no SA.

## 5.9 Ameaças à Validade

A validade interna define se o relacionamento observado entre o tratamento e o resultado é causal, de modo que não é resultado de influência de outro fator – não controlado ou medido (Travassos et al., 2002). Sendo assim, neste estudo exploratório, ressalta-se as seguintes ameaças à validade:

- **Instrumentação:** tal ameaça está no contexto da medição incorreta ou instrumento inadequado (Wainer, 2007). No estudo exploratório, o uso da abordagem S03 envolveu o seu estudo e aplicação no contexto deste trabalho. Entretanto, originalmente a abordagem foi apresentada com um exemplo ilustrativo, no contexto de formalismos matemáticos. Portanto, a maneira como a abordagem foi aplicada neste trabalho poderia não ser a esperada pelos autores originais.
- **Testagem:** tal ameaça está no contexto da sequência da coleta dos resultados versus o momento de intervenção (Wainer, 2007). No estudo exploratório, a execução das abordagens de teste seguiu uma ordem predeterminada. Primeiro aplicou-se a abordagem S03 (Tse et al., 2004), depois aplicou-se a abordagem S05 (Lu et al.,

2006) e, por fim, aplicou-se a abordagem S14 (Munoz e Baudry, 2009). Portanto, a alteração na ordem da execução poderia ter obtido, por exemplo, uma quantidade diferente de casos de teste criados.

- **Seleção:** tal ameaça está no contexto da amostragem que não foi selecionada de maneira aleatória ou não foi caracterizada de maneira igualitária, tanto no aspecto quantitativo quanto qualitativo (Wainer, 2007). No estudo exploratório, os cenários utilizados foram obtidos por meio do trabalho de Sama et al. (2010a), os quais definem 28 predicados de adaptação para o sistema *PhoneAdapter* (SA5). Além disso, definiu-se um conjunto de restrições de domínio de um possível usuário, cujo perfil pode variar entre: Trabalhar; Estudar; e Ir à Igreja. Portanto, os cenários e as restrições não foram definidas de forma aleatória ou com técnicas de Estatística.
- **Regressão à Média:** tal ameaça está no contexto quando se trabalha apenas com um grupo (somente o grupo experimental), ou seja, não há um grupo de controle (Bandeira, 2012). No estudo exploratório, executaram-se as estratégias de teste em apenas um grupo de cenários, originados de um modelo de contexto definido no sistema *PhoneAdapter* (SA5). Além disso, não há na literatura outro trabalho que executou estratégias de teste baseadas em desafios caracterizados. Portanto, além de não se ter utilizado outro trabalho como controle, não se executaram as estratégias de teste em outros cenários e/ou modelos de contexto.

## 5.10 Considerações Finais

Neste capítulo, a fim de responder às questões QP3 e QP4, apresentaram-se a definição das estratégias de teste; a avaliação da efetividade dessas estratégias de teste; e a investigação da presença dos desafios nos SAs analisados, por meio das estratégias de teste.

Portanto, este capítulo compreendeu-se em apresentar algumas das etapas da metodologia utilizada neste trabalho, sendo elas:

(8) *Seleção de abordagens e SAs*; (9) *Seleção de um modelo subjacente de teste*; (10) *Definição das estratégias de teste*; (11) *Automatização das estratégias T e T\**; e (12) *Análise dos resultados*.

Como resultados, obtiveram-se : (i) a definição de duas estratégias de teste guiadas pelos desafios caracterizados; (ii) a execução das estratégias em um SA, de modo a relevar defeitos; (iii) a avaliação que apresenta efetividade e custo das estratégias de teste; e, por fim, (iv) discussões sobre os desafios enfrentados durante a execução dos testes. Assim, observaram-se: (i) a estratégia de teste T\* mostrou-se mais efetiva que a T, a nível de

falhas externalizadas e de defeitos revelados; e (ii) nove dos desafios genéricos puderam-se ser enfrentados com as estratégias de teste. Ressalta-se que o estudo exploratório realizado baseou-se em apenas um dos cinco SAs analisados neste trabalho.

No Capítulo 6 apresentam-se as conclusões do trabalho, enfatizando-se as contribuições, limitações, trabalhos relacionados e trabalhos futuros.

---

## Conclusões

---

Neste trabalho foram investigados os desafios para o teste de Sistemas Adaptativos (SAs) e, a partir desta investigação, características das abordagens de teste foram analisadas. O principal problema abordado neste trabalho foi: como caracterizar adequadamente, de forma abrangente, as dificuldades para se testar SAs? Neste sentido, quais maneiras foram eficazes de se testar SAs? Na literatura não foi encontrada qualquer estratégia de teste que seja guiada por desafios de teste caracterizados.

No trabalho de Ferrari et al. (2011) caracterizaram-se desafios impostos à atividade de teste em SAs (por meio de uma Revisão Sistemática (RS) da Literatura) e, também, levantou-se a hipótese: *Saber quais os possíveis desafios que podem ser enfrentados, durante a atividade do teste em SAs, pode ajudar na definição de uma estratégia de teste para esse tipo de sistema.* Com base na hipótese levantada, objetivou-se compreender e caracterizar os desafios impostos à atividade de teste de SAs. Para tal objetivo definiram-se as seguintes questões de pesquisa, que são revisitadas juntamente com as conclusões obtidas:

**QP1: Quais são os desafios impostos pelos SAs à atividade de teste?** Caracterizaram-se 34 desafios específicos e, desses 34, 12 desafios genéricos. Observou-se também que os desafios genéricos estão estritamente relacionados a: (i) decisões em tempo de execução de SAs (*C-1, C-2, C-3, C-6, C-7, C-10, C-12*), requerendo que o teste lide com imprevisibilidade de configurações que o sistema possa ter; (ii) ambientes distribuídos e heterogêneos (*C-4*), conseqüentemente levando o teste para além da fronteira de um único

sistema; (iii) interferência de usuários (*C-5*), que requer testes que lidem com adaptações que são personalizadas por usuários do sistema; (iv) atualização e rastreabilidade entre artefatos de software variados (*C-8*), que requerem a manutenção de artefatos que incluem incerteza que é inerente de SAs; (v) definição de sensores e frequência de monitoramento de dados (*C-9*), que tem um impacto direto nas configurações do ambiente de teste; e (vi) geração de casos de teste (*C-11*), que pode ser não efetiva devido à incerteza presente em SAs.

**QP2: Quais são os desafios caracterizados encontrados durante a atividade de teste em SAs reais?** De acordo com o relacionamento entre propriedades de SAs e desafios específicos: (i) 18 dos 34 desafios específicos estavam relacionados aos cinco SAs analisados; (ii) 14 dos 34 desafios específicos estavam relacionados a quatro dos cinco SAs analisados; e, por fim, (iii) dois dos 34 desafios específicos estavam relacionados a três dos cinco SAs analisados. Portanto, cada um dos desafios específicos estava em pelo menos três dos cinco SAs analisados.

**QP3: Quais são os defeitos encontrados com a aplicação das estratégias de teste baseadas em desafios?** Com a aplicação da Estratégia T externalizaram-se quatro falhas, originadas de um defeito de predicado morto. Além disso, com a aplicação da Estratégia T\* externalizaram-se 16 falhas, originadas de quatro defeitos, contendo ambos os tipos de defeitos predicado morto e ativação não determinística. A estratégia de teste T\* mostrou-se mais efetiva, porém mais custosa, do que a estratégia de teste T.

**QP4: Quais são os desafios mitigados com a aplicação das estratégias de teste baseadas em desafios?** Os nove primeiros desafios genéricos [C-1]–[C-9] puderam ser enfrentados durante condução do estudo exploratório. Apesar dos desafios estarem presentes nos sistemas, o uso das estratégias de teste ajudou na condução do teste no sistema com as propriedades de SAs.

## 6.1 Revisitando as Contribuições do Trabalho

Nesta seção revisitam-se os resultados do trabalho em termos de três tipos de contribuições: (i) definições teóricas; (ii) implementação de apoio automatizado; e (iii) estudos de avaliação. Ressalta-se que as contribuições ajudam a superar as limitações da pesquisa, destacadas mais à frente neste capítulo.

- Caracterização de desafios de teste para SAs
- Diretriz para localizar propriedades de SAs



- Caracterização de abordagens e desafios de teste de SAs
- Caracterização de abordagens e fases de teste de SAs
- Caracterização dos desafios de teste que são encontrados em SAs reais
- Definição de estratégias de teste baseadas em desafios
- Automatização de teste baseada de SAs baseada em uma representação formal
- Identificação de defeitos encontrados por abordagens previamente propostas
- Identificação de defeitos não encontrados por abordagens previamente propostas
- Avaliação de efetividade da aplicação de estratégias de teste baseadas em desafios

## 6.2 Trabalhos Relacionados

O principal trabalho relacionado é o de Ferrari et al. (2011), o qual foi utilizado como base do presente trabalho por meio do fornecimento da hipótese principal e da caracterização dos desafios impostos à atividade de teste pelos SAs. Além disso, o trabalho de Ferrari et al. foi atualizado e estendido durante a condução do presente trabalho. A principal semelhança entre ambos é a caracterização de desafios genéricos e a principal diferença é a extensão da RS e a aplicação das estratégias de teste permeadas pelos desafios impostos à atividade do teste.

Outro trabalho relacionado é o de Matalonga et al. (2015a). Este trabalho não entrou na RS por se tratar de um estudo secundário. Entretanto, tal estudo auxiliou na inclusão e avaliação de um conjunto extra de estudos (adicionados como opinião de especialistas). Ressalta-se, porém, que o trabalho de Matalonga et al. utilizou diferentes critérios de inclusão e exclusão, a saber, os trabalhos selecionados por eles foram avaliados um-a-um neste trabalho. A principal semelhança é a caracterização de desafios genéricos. A principal diferença é a aplicação das estratégias de teste. Além disso, o mesmo grupo de pesquisa propôs uma caracterização de abordagens de teste (Matalonga et al., 2015b). Tal caracterização teve como base o padrão ISO/IEC/IEEE 29119/Técnicas de Teste (IEEE, 2014). Entretanto, assim como no primeiro trabalho desses autores, este trabalho não foi selecionado por ser um estudo secundário.

## 6.3 Limitações

Uma validade externa define condições que limitam a habilidade de generalizar os resultados de um experimento para a prática industrial (Travassos et al., 2002). Neste trabalho pode-se elencar as limitações:

- *Análises teóricas realizadas:* As caracterizações realizadas no Capítulo 4 foram conduzidas com base no estudo e entendimento dos trabalhos da RS do Capítulo 3. Os trabalhos continham variados níveis de detalhes e de diferentes contextos. Então, a aplicação prática do trabalho foi, de fato, no Capítulo 5 com o estudo exploratório.
- *Limitação da amostra de avaliação:* O estudo exploratório foi composto por apenas um sistema, então, os resultados não podem ser generalizados para todos os contextos em todas as aplicações. A análise dos SAs foi baseada em propriedades previamente elencadas, assim, tais propriedades podem ser avaliadas em outros sistemas também previamente avaliados neste trabalho.
- *Automatização desenvolvida:* A automatização do teste, desenvolvida neste trabalho, é focada na linguagem de programação Java na plataforma *android*. O formalismo proposto pode ser implementado outras linguagens e plataformas.
- *Número de abordagens de teste:* As abordagens de teste aplicadas, por meio das estratégias T e T\*, foram apenas três – S03 (Tse et al., 2004), S05 (Lu et al., 2006) e S14 (Munoz e Baudry, 2009). Assim, a aplicação de outras abordagens poderia obter resultados diferentes.

## 6.4 Trabalhos Futuros

Os trabalhos futuros propostos são descritos na sequência.

1. *Aplicação das estratégias de teste em diferentes SAs:* a análise de SAs gerou uma lista de possíveis SAs para serem candidatos à aplicação das estratégias T e T\*. Como um trabalho futuro, pretende-se aplicação as estratégias em outros SAs, a fim de comparar os resultados.
2. *Utilização de diferentes abordagens de teste nas estratégias T e T\*:* a caracterização das abordagens de teste gerou uma lista de abordagens candidatas a serem personalizadas e aplicadas a SAs. Como um trabalho futuro, pretende-se utilizar outras abordagens, desta caracterização, nas estratégias T e T\*.

3. *Teste de Mutação dirigido pelos desafios caracterizados*: a condução do estudo exploratório gerou uma discussão em cada um dos desafios impostos à atividade de teste, vindas da análise teórica e da aplicação prática. Como um trabalho futuro, pretende-se explorar cada um dos desafios, a fim de gerar operadores de mutação baseados em análise de desafios.
4. *Geração de casos de teste*: durante a aplicação das abordagens geraram-se duas discussões a respeito de dois desafios, o C-10 e C-11, com relação à geração de casos de teste e à aplicação do teste com alta carga de trabalho. Como um trabalho futuro, pretende-se explorar maneiras de realizar a geração de casos de teste com base nos formalismos da máquina da *mealy*.

## Referências

---

---

- AKOUR, M.; JAIDEV, A.; KING, T. M. Towards change propagating test models in autonomic and adaptive systems. In: *Proceedings of the 18<sup>th</sup> International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS)*, Las Vegas/NV - USA: IEEE Computer Society, p. 89–96, 2011.
- BANDEIRA, M. *Validade interna e externa de uma pesquisa: vieses*. Technical Report, Departamento de Psicologia/UFSJ, São João del-Rei – Brazil, 2012.
- BARTEL, A.; BAUDRY, B.; F., M.; J., K.; T., M.; LE TRAON, Y. Model driven mutation applied to adaptative systems testing. *Computing Research Repository (CoRR)*, v. abs/1205.5783, p. 408–413, 2012.
- BARTEL, A.; BAUDRY, B.; MUNOZ, F.; KLEIN, J.; MOUELHI, T.; LE TRAON, Y. Model driven mutation applied to adaptative systems testing. In: *Proceedings of the 4<sup>th</sup> International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Berlin - Germany: IEEE Computer Society, p. 408–413, 2011.
- BASIL, V. R. *Software modeling and measurement: The goal/question/metric paradigm*. Technical Report, University of Maryland at College Park, College Park, MD, USA, 1992.
- BENSALEM, S.; MOEZ, K.; STAVROS, T. State identification problems for input/output transition systems. In: *Proceedings of the 9<sup>th</sup> International Workshop on Discrete Event Systems (WODES)*, Goteborg - Sweden: IEEE Computer Society, p. 225–230, 2008.

- BIOLCHINI, J.; MIAN, P.; NATALI, A.; CONTE, T.; TRAVASSOS, G. Scientific research ontology to support systematic review in software engineering. *Advanced Engineering Informatics*, v. 21, n. 2, p. 133–151, 2007.
- CÁMARA, J.; LEMOS, R.; LARANJEIRO, N.; VENTURA, R.; VIEIRA, M. Robustness evaluation of controllers in self-adaptive software systems. In: *Proceedings of the 6<sup>th</sup> Latin-American Symposium on Dependable Computing (LADC)*, Rio de Janeiro/RJ: IEEE Computer Society, p. 1–10, 2013.
- CÁMARA, J.; LEMOS, R.; LARANJEIRO, N.; VENTURA, R.; VIEIRA, M. Testing the robustness of controllers for self-adaptive systems. *Journal of the Brazilian Computer Society*, v. 20, n. 1, p. 1–14, 2014.
- CÁMARA, J.; LEMOS, R.; LARANJEIRO, N.; VENTURA, R.; VIEIRA, M. Robustness-Driven Resilience Evaluation of Self-Adaptive Software Systems. *IEEE Transactions on Dependable and Secure Computing*, v. 6, n. 1, p. 1–1, 2015.
- CAMARGO, K.; FERRARI, F.; FABBRI, S. Identifying a subset of tmimi practices to establish a streamlined software testing process. In: *Proceedings of the 27<sup>th</sup> Brazilian Symposium on Software Engineering (SBES)*, p. 137–146, 2013.
- CHAN, W.; CHEN, T.; LU, H. A Metamorphic Approach to Integration Testing of Context-Sensitive Middleware-Based Applications. In: *Proceedings of the 5<sup>th</sup> International Conference on Quality Software (QSIC)*, Melbourne - Australia: IEEE Computer Society, p. 241–249, 2005.
- CHAN, W. K.; YUEH CHEN, T.; LU, H.; TSE, T. H.; YAU, S. Integration testing of context-sensitive middleware-based applications: a metamorphic approach. *International Journal of Software Engineering and Knowledge Engineering*, v. 16, n. 5, p. 677–704, 2006.
- CHENG, B.; LEMOS, R.; GIESE, H.; INVERARDI, P.; MAGEE, J.; ANDERSSON, J.; BECKER, B.; BENCOMO, N.; BRUN, Y.; CUKIC, B.; MARZO SERUGENDO, G.; DUSTDAR, S.; FINKELSTEIN, A.; GACEK, C.; GEIHS, K.; GRASSI, V.; KARSAI, G.; KIENLE, H. M.; KRAMER, J.; LITOU, M.; MALEK, S.; MIRANDOLA, R.; MÜLLER, H.; PARK, S.; SHAW, M.; TICHY, M.; TIVOLI, M.; WEYNS, D.; WHITTLE, J. *Software engineering for self-adaptive systems: A research roadmap*, chapter Software Engineering for Self-Adaptive Systems Berlin, Heidelberg: Springer Berlin Heidelberg, p. 1–26, 2009.

- COHEN, D.; DALAL, S. R.; FREDMAN, M. L.; PATTON, G. C. The aetg system: an approach to testing based on combinatorial design. *IEEE Transactions on Software Engineering*, v. 23, n. 7, p. 437–444, 1997.
- DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. *Introdução ao teste de software*. Elsevier, 2007.
- EBERHARDINGER, B.; SEEBACH, H.; KNAPP, A.; REIF, W. Towards Testing Self-organizing, Adaptive Systems. In: *Proceedings of the 26<sup>th</sup> International Conference Testing Software and Systems (ICTSS)*, Madrid - Spain: Springer Link, p. 180–185, 2014.
- EVERS ET AL., C. The user in the loop: Enabling user participation for self-adaptive applications. *Future Generation Computer Systems*, v. 34, n. 0, p. 110–123, 2014.
- FABBRI, S. C. P. F.; FELIZARDO, K. R.; FERRARI, F. C.; HERNANDES, E. C. M.; OCTAVIANO, F. R.; NAKAGAWA, E. Y.; MALDONADO, J. C. Externalising tacit knowledge of the systematic review process. *IET Software*, v. 7, n. 6, p. 298–307, 2013.
- FERRARI, F. C.; CAPEO, B. B. P.; NOPPEN, J.; CHITCHYAN, R.; RASHID, A. Investigating testing approaches for dynamically adaptive systems. In: *Lightning Talk in the 2<sup>nd</sup> International Workshop on Variability & Composition (VariComp) - in conjunction with AOSD*, Porto de Galinhas/PE - Brazil, 2011.
- FLORES, A.; AUGUSTO, J. C.; POLO, M.; VAREA, M. Towards context-aware testing for semantic interoperability on PvC environments. In: *Proceedings of the 17<sup>th</sup> International Conference on Systems, Man and Cybernetics*, The Hague - The Netherlands: IEEE Computer Society, p. 1136–1141, 2004.
- FORNEY, G. D. J. Generalized minimum distance decoding. *IEEE Transactions on Information Theory*, v. 12, n. 2, p. 125–131, 1966.
- FRANKL, P. G.; WEYUKER, E. J. Testing software to detect and reduce risk. *Journal of Systems and Software*, v. 53, n. 3, p. 275–286, 2000.
- FREDERICKS, E.; DEVRIES, B.; CHENG, B. Towards run-time adaptation of test cases for self-adaptive systems in the face of uncertainty. In: *Proceedings of the 9<sup>th</sup> International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, Hyderabad - India: IEEE Computer Society, p. 17–26, 2014.

- FREDERICKS, E.; RAMIREZ, A.; CHENG, B. Towards run-time testing of dynamic adaptive systems. In: *Proceedings of the 8<sup>th</sup> International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, San Francisco/CA, USA: IEEE Computer Society, p. 169–174, 2013.
- GARLAN, D.; CHENG, S.; HUANG, A.; SCHMERL, B.; STEENKISTE, P. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *IEEE Computer*, v. 37, n. 10, p. 45–54, 2004.
- GARLAN, D.; SHAW, M. An introduction to software architecture. 1994.
- GARVIN, B.; COHEN, M.; DWYER, M. Using feature locality: Can we leverage history to avoid failures during reconfiguration? In: *Proceedings of the 8<sup>th</sup> Workshop on Assurances for Self-adaptive Systems (ASAS)*, New York/NY, USA: ACM, p. 24–33, 2011.
- GREENHALGH, T. *How to read a paper: The basics of evidence-based medicine*. HOW - How To. Wiley, 2014.
- GRIEBE, T.; GRUHN, V. A model-based approach to test automation for context-aware mobile applications. In: *Proceedings of the 29<sup>th</sup> Annual ACM Symposium on Applied Computing (SAC)*, New York/NY - USA: ACM Press, p. 420–427, 2014.
- HAMMING, R. W. Error detecting and error correcting codes. *Bell System Technical Journal*, v. 29, n. 2, p. 147–160, 1950.
- HARROLD, M. Testing: A roadmap. In: *Proceedings of the 22<sup>th</sup> International Conference on the Future of Software Engineering (ICSE)*, Limerick - Ireland: ACM Press, p. 61–72, 2000.
- HOPCROFT, J. E.; MOTWANI, R.; ROTWANI; ULLMAN, J. D. *Introduction to automata theory, languages and computability*. 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.
- HORN, P. *Ibm perspective on the state of information technology - autonomic computing*. Technical Report, IBM, 2001.
- HURTADO, S.; SEN, S.; CASALLAS, R. Reusing legacy software in a self-adaptive middleware framework. In: *Proceedings of the 12<sup>th</sup> International Workshop of Adaptive and Reflective Middleware (ARM)*, New York/NY - USA: ACM Press, p. 29–35, 2011.

- HÖHN, E. *Kitest: Um arcabouço de conhecimento e melhoria de processo de teste*. PhD Thesis, Ph. D. Thesis. ICMC/USP, São Carlos, SP, Brasil, 2011.
- IBM. *Autonomic computing white paper: An architectural blueprint for autonomic computing*. *IBM White Paper*, 2006.
- IEEE. *IEEE 829 standard for software and system test documentation*. Standard 610.12, Institute of Electric and Electronic Engineers, New York/NY - USA, 1990.
- IEEE. *IEEE international standard for software and systems engineering – software testing – part 4: Test techniques*. Standard 29119-4, Institute of Electric and Electronic Engineers, 2014.
- IGLESIA, D. G. D. L. *Mape-k formal templates for self-adaptive systems: Specifications and descriptions*. 2014.
- JAW, L. C.; HOMAN, D.; CRUM, V.; CHOU, W.; KELLER, K.; SWEARINGEN, K.; SMITH, T. *Model-based Approach to Validation and Verification of Flight Critical Software*. In: *Proceedings of the 29<sup>th</sup> International IEEE Aerospace Conference (AeroConf)*, Big Sky/Montana - USA: IEEE Computer Society, p. 1–8, 2008.
- KEPHART, J.; CHESS, D. *The vision of autonomic computing*. *IEEE Computer*, v. 36, n. 1, p. 41–50, 2003.
- KING, T.; ALLEN, A.; CRUZ, R.; CLARKE, P. *Safe runtime validation of behavioral adaptations in autonomic software*. In: *Proceedings of the 8<sup>th</sup> International Conference Autonomic and Trusted Computing (ATC)*, Banff - Canada: Springer Link, p. 31–46, 2011a (*Lecture Notes in Computer Science*, v.6906 LNCS).
- KING, T.; ALLEN, A.; WU, Y.; CLARKE, P.; RAMIREZ, A. *A comparative case study on the engineering of self-testable autonomic software*. In: *Proceedings of the 8<sup>th</sup> International Conference and Workshops on Engineering of Autonomic and Autonomous Systems (EASE)*, Los Alamitos/CA - USA: IEEE Computer Society, p. 59–68, 2011b.
- KING, T.; BABICH, D.; ALAVA, J.; CLARKE, P.; STEVENS, R. *Towards Self-Testing in Autonomic Computing Systems*. In: *Proceedings of the 8<sup>th</sup> International Symposium on Autonomous Decentralized Systems (ISADS)*, Sedona/Arizona - USA: IEEE Computer Society, p. 51–58, 2007a.



- KING, T.; RAMIREZ, A.; CLARKE, P.; QUINONES-MORALES, B. A Reusable Object-Oriented Design to Support Self-Testable Autonomic Software. In: *Proceedings of the 23<sup>th</sup> ACM Symposium on Applied Computing (SAC)*, Fortaleza/CE - Brazil: ACM Press, p. 1664–1669, 2008.
- KING, T.; RAMIREZ, A.; CRUZ, R.; CLARKE, P. An integrated self-testing framework for autonomic computing systems. *Journal of Computers*, v. 2, n. 9, p. 37–49, 2007b.
- KITCHENHAM, B. *Procedures for performing systematic reviews*. Technical Report, Keele University and NICTA, 2004.
- KITCHENHAM, B.; BRERETON, P. A systematic review of systematic review process research in software engineering. *Information and Software Technology*, v. 55, n. 12, p. 2049–2075, 2013.
- KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing systematic literature reviews in software engineering*. Technical Report (EBSE) 2007-001, Keele University and Durham University Joint Report, UK, 2007.
- KLEIN, C.; SCHMID, R.; LEUXNER, C.; SITOU, W.; SPANFELNER, B. A survey of context adaptation in autonomic computing. In: *Proceedings of the 4<sup>th</sup> International Conference on Autonomic and Autonomous Systems (ICAS)*, IEEE Computer Society, p. 106–111, 2008.
- LALANDA, P.; MCCANN, J. A.; DIACONESCU, A. *Autonomic computing*. Springer, 2013.
- LIU, Y. Phoneadapter. Online, <http://sccpu2.cse.ust.hk/afchecker/phoneadapter.html> - último acesso em 18/04/2014, 2013.
- LU, H. A context-oriented framework for software testing in pervasive environment. In: *Proceeding of the 29<sup>th</sup> International Conference on Software Engineering (ICSE)*, Minneapolis/MN - USA: IEEE Computer Society, p. 77–78, 2007.
- LU, H.; CHAN, W.; TSE, T. Testing context-aware middleware-centric programs: A data flow approach and an rfid-based experimentation. In: *Proceedings of the 14<sup>th</sup> International Symposium on Foundations of Software Engineering (SIGSOFT)*, Portland/OR - USA: ACM Press, p. 242–252, 2006.
- LU, H. L. H.; CHAN, W.; TSE, T. Testing pervasive software in the presence of context inconsistency resolution services. In: SCHÄFER, W.; DWYER, M. B.; GRUHN, V.,

- eds. *Proceedings of the 30<sup>th</sup> International Conference on Software Engineering (ICSE)0*, Leipzig - Germany: IEEE Computer Society, p. 61–70, 2008.
- MATALONGA, S.; RODRIGUES, F.; TRAVASSOS, G. Challenges in testing context aware software systems. In: *Proceedings of the 9<sup>th</sup> Brazilian Workshop on Systematic and Automated Software Testing (SAST)*, Belo Horizonte/MG - Brazil: Brazilian Computer Society, p. 51–60, 2015a.
- MATALONGA, S.; RODRIGUES, F.; TRAVASSOS, G. Matching context aware software testing design techniques to iso/iec/ieee 29119. In: ROUT, T.; O’CONNOR, R.; DORLING, A., eds. *Proceedings of the 15<sup>th</sup> International Conference Software Process Improvement and Capability Determination (SPICE)*, Gothenburg - Sweden: Springer International Publishing, p. 33–44, 2015b.
- MERDES, M.; MALAKA, R.; SULIMAN, D.; PAECH, B.; BRENNER, D.; ATKINSON, C. Ubiquitous rats: How resource-aware run-time tests can improve ubiquitous software system. In: *Proceedings of the 6<sup>th</sup> International Workshop on Software Engineering and Middleware (SEM)*, New York/NY - USA: ACM Press, p. 55–62, 2006.
- METTE, A.; HASS, J. Testing processes. In: *Proceedings of the 7<sup>th</sup> International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Los Alamitos, CA, USA: IEEE Computer Society, p. 322–327, 2008.
- VAN DER MEULEN, R.; RIVERA, J. Gartner says by 2017, mobile users will provide personalized data streams to more than 100 apps and services every day. Online, <http://www.gartner.com/newsroom/id/2654115> - último acesso em 18/04/2014, 2014.
- MICSKEI, Z.; SZATMÁRI, Z.; OLÁH, J.; MAJZIK, I. A concept for testing robustness and safety of the context-aware behaviour of autonomous systems. In: *Proceedings of the 6<sup>th</sup> International Conference of Agent and Multi-Agent Systems (KES-AMSTA)*, Dubrovnik - Croatia: Springer Link, p. 504–513, 2012 (*Lecture Notes in Computer Science*, v.7327 LNAI).
- MÜLLER, H.; PEZZÈ, M.; SHAW, M. Visibility of control in adaptive systems. In: *Proceedings of the 2<sup>nd</sup> International Workshop on Ultra-Large-Scale Software-Intensive Systems (ULSSIS)*, CMU Press, 2008.
- MUNOZ, F.; BAUDRY, B. *Artificial table testing dynamically adaptive systems*. Technical Report inria-00365874, INRIA Bretagne Atlantique, Beaulieu - France, 2009.

- MYERS, G. J.; SANDLER, C.; BADGETT, T. *The Art of Software Testing*. ITPro collection. Wiley, 2011.
- NIEBUHR, D.; RAUSCH, A. A concept for dynamic wiring of components: Correctness in dynamic adaptive systems. In: *Proceedings of the 6<sup>th</sup> Workshop on Specification and Verification of Component-Based Systems (SAVCBS) - held in conjunction with ESEC/FSE*, Dubrovnik - Croatia: ACM Press, (short paper), p. 101–102, 2007.
- NIEBUHR, D.; RAUSCH, A. Guaranteeing correctness of component bindings in dynamic adaptive systems. In: *Proceedings of the 35<sup>th</sup> Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Patras - Greece: IEEE Computer Society, p. 454–457, 2009a.
- NIEBUHR, D.; RAUSCH, A. Guaranteeing correctness of component bindings in dynamic adaptive systems based on runtime testing. In: *Proceedings of the 4<sup>th</sup> International Workshop on Services Integration in Pervasive Environments (SIPE)*, New York/NY - USA: ACM Press, p. 7–12, 2009b.
- NIEBUHR, D.; RAUSCH, A.; KLEIN, C.; REICHMANN, J.; SCHMID, R. Achieving dependable component bindings in dynamic adaptive systems - a runtime testing approach. In: *Proceedings of the 3<sup>rd</sup> International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, San Francisco/CA - USA: IEEE Computer Society, p. 186–197, 2009.
- NOBRE, M.; BERNARDO, W.; JATENE, F. A prática clínica baseada em evidências. Parte I: questões clínicas bem construídas. *Revista da Associação Médica Brasileira*, v. 49, n. 4, p. 445–449, 2003.
- OFFUTT, J.; LIU, S.; ABDURAZIK, A.; AMMANN, P. Generating test data from state-based specifications. *The Journal of Software Testing, Verification and Reliability*, v. 13, n. 1, p. 25–53, 2003.
- OREIZY ET AL., P. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems*, v. 14, n. 3, p. 54–62, 1999.
- PRESSMAN, R. S. *Software engineering: A practitioner's approach*. 7th. ed. New York/NY - USA: McGraw-Hill, 2010.
- PÜSCHEL, G.; GÖTZ, S.; WILKE, C.; ASSMANN, U. Towards systematic model-based testing of self-adaptive software. In: *Proceedings of the 5<sup>th</sup> International Conference on*

- Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE)*, Valencia - Spain: IARIA Computer Society, p. 65–70, 2013.
- PÜSCHEL, G.; SEIGER, R.; SCHLEGEL, T. Test modeling for context-aware ubiquitous applications with feature petri nets. In: *Proceedings of the 4<sup>th</sup> Symposium on Engineering Interactive Computing Systems (SIGCHI)*, Copenhagen - Denmark: ACM Press, p. 37–40, 2012.
- RAMIREZ, A.; CHENG, B. Design patterns for developing dynamically adaptive systems. In: *Proceedings of the 32<sup>th</sup> International Conference on Software Engineering (ICSE)*, Cape Town, South Africa: ACM Press, p. 49–58, 2010.
- RAMIREZ, A. E.; MORALES, B.; KING, T. A self-testing autonomic job scheduler. In: *Proceedings of the 46<sup>th</sup> ACM Annual Southeast Regional Conference (ACMSE)*, New York/NY - USA: ACM Press, p. 304–309, 2008.
- RICHARDSON, D.; AHA, S.; O'MALLEY, T. Specification-based test oracles for reactive systems. In: *Proceedings of the 14<sup>th</sup> International Conference on Software Engineering (ICSE)*, New York, NY, USA: ACM Press, p. 105–118, 1992.
- SAMA, M.; ELBAUM, S.; RAIMONDI, F.; ROSENBLUM, D. S.; WANG, Z. Context-aware adaptive applications: Fault patterns and their automated identification. *IEEE Transactions on Software Engineering*, v. 36, n. 5, p. 644–661, 2010a.
- SAMA, M.; ROSENBLUM, D. S.; WANG, Z.; ELBAUM, S. Model-based fault detection in context-aware adaptive applications. In: *Proceedings of the 16<sup>th</sup> International Symposium on Foundations of Software Engineering (FSE)*, Atlanta/GA - USA: ACM Press, p. 261–271, 2008.
- SAMA, M.; ROSENBLUM, D. S.; WANG, Z.; ELBAUM, S. Multi-layer faults in the architectures of mobile, context-aware adaptive applications. *Journal of Systems and Software*, v. 83, n. 6, p. 906 – 914, 2010b.
- SERIKAWA, M.; SIQUEIRA, B.; FERRARI, F.; MENOTTI, R.; CAMARGO, V. Guidelines for modularizing the monitor component when refactoring adaptive systems. In: *Proceedings of the 2<sup>nd</sup> Latin-American School on Software Engineering (ELA-ES)*, Porto Alegre - Brazil, p. 4, 2015.
- SHAW, M. Beyond objects: A software design paradigm based on process control. *SIGSOFT Software Engineering Notes*, v. 20, n. 1, p. 27–38, 1995.

- SILVA, C.; LEMOS, R. Dynamic plans for integration testing of self-adaptive software systems. In: *Proceedings of the 6<sup>th</sup> International Symposium on Software Engineering for Adaptive and Self-Managing System (ICSE)*, Honolulu/HI - USA: ACM Press, p. 148–157, 2011.
- SOH, Z. Context and vision: Studying two factors impacting program comprehension. In: *Proceedings of the 19<sup>th</sup> International Conference on Program Comprehension (ICPC)*, Kingston - ON: IEEE Computer Society, p. 258–261, 2011.
- STEVENS, R.; STEVENS, R.; PARSONS, B.; PARSONS, B.; KING, T. A self-testing autonomic container. In: *Proceedings of the 45<sup>th</sup> Annual Southeast Regional Conference (ACM SE)*, Winston-Salem/NC - USA: ACM Press, p. 1–6, 2007.
- STEVENSON, A. *Oxford dictionary of english*. Oxford reference online premium. OUP Oxford, 2011.
- TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G. *Introdução à engenharia software experimental*. Technical Report, COPPE/UFRJ, Rio de Janeiro – Brazil, 2002.
- TSE, T.; CHAN, W.; YAU, S.; LU, H.; CHEN, T. Testing context-sensitive middleware-based software applications. In: *Proceedings of the 28<sup>th</sup> Annual International Computer Software and Applications Conference (COMPSAC)*, Hong Kong - China: IEEE Computer Society, p. 1–9, 2004.
- VASSEV, E.; HINCHEY, M.; NIXON, P. Automated test case generation of self-managing policies for nasa prototype missions developed with assl. In: *Proceedings of the 4<sup>th</sup> International Symposium on Theoretical Aspects of Software Engineering (TASE)*, Taipei - Taiwan: IEEE Computer Society, p. 3–8, 2010.
- WAINER, J. Métodos de pesquisa quantitativa e qualitativa para a ciência da computação. 2007.  
Disponível em <http://www.pucrs.br/famat/viali/mestrado/mqp/material/textos/Pesquisa.pdf>
- WANG, H.; CHAN, W. Weaving context sensitivity into test suite construction. In: *Proceedings of the 24<sup>th</sup> International Conference on Automated Software Engineering (ASE)*, Auckland - New Zealand: IEEE Computer Society, p. 610–614, 2009.
- WANG, H.; CHAN, W.; TSE, T. Improving the effectiveness of testing pervasive software via context diversity. *ACM Transactions on Autonomous and Adaptive Systems*, v. 9, n. 2, p. 1–28, 2014.

- WANG, H.; KONG, H.; CHAN, W. *On the construction of context-aware test suites*. Technical Report TR-2010-01, Laboratory for Computer Science, University of Hong Kong, Hong Kong - China, 2010a.
- WANG, H.; ZHAI, K.; TSE, T. Correlating context-awareness and mutation analysis for pervasive computing systems. In: *Proceedings of the 10<sup>th</sup> International Conference on Quality Software (QSIC)*, Zhangjiajie - China: IEEE Computer Society, p. 151–160, 2010b.
- WANG, Z.; ELBAUM, S.; ROSENBLUM, D. S. Automated generation of context-aware tests. In: *Proceedings of the 29<sup>th</sup> International Conference on Software Engineering (ICSE)*, Minneapolis/MN - USA: IEEE Computer Society, p. 406–415, 2007.
- WELSH, K.; SAWYER, P. Managing testing complexity in dynamically adaptive systems: A model-driven approach. In: *Proceedings of the 3<sup>rd</sup> International Conference on Software Testing, Verification and Validation (ICST)*, Paris - France: IEEE Computer Society, p. 290–298, 2010.
- WEYNS, D. Towards an integrated approach for validating qualities of self-adaptive systems. In: *Proceedings of the 9<sup>th</sup> International Workshop on Dynamic Analysis (WODA)*, Minneapolis/MN - USA: ACM Press, p. 24–29, 2012.
- WEYNS, D.; USMAN IFTIKHAR, M.; SODERLUND, J. Do external feedback loops improve the design of self-adaptive systems? a controlled experiment. In: *Proceedings of the 8<sup>th</sup> International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, IEEE, p. 3–12, 2013.
- WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: *Proceedings of the 18<sup>th</sup> International Conference on Evaluation and Assessment in Software Engineering (EASE)*, London - England: ACM Press, p. 1–10, 2014.
- WOTAWA, F. Adaptive autonomous systems - from the system's architecture to testing. *Communications in Computer and Information Science*, v. 336 CCIS, p. 76–90, 2012.
- XU, D. Z.; XU, D.; ZHOU, L. Bigraphical model of context-aware in ubiquitous computing environments. In: PARK, J. J.; NIKOLAOU, C.; CAO, J., eds. *Proceedings of the 17<sup>th</sup> Asia-Pacific Services Computing Conference (APSCC)*, Jeju - Korea (South): IEEE Computer Society, p. 389–394, 2011.

- YAMASATHIEN, S.; VATANAWOOD, W. An approach to construct formal model of business process model from bpmn workflow patterns. In: *Proceedings of the 4<sup>th</sup> International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, Bangkok - Thailand: IEEE Computer Society, p. 211–215, 2014.
- YE, C.; CHEUNG, S.; WEI, J.; ZHONG, H.; HUANG, T. A study on the replaceability of context-aware middleware. In: *Proceedings of the 1<sup>st</sup> Asia-Pacific Symposium on Internetware (APASI)*, Beijing, China: ACM Press, p. 4:1–4:10, 2009.
- YU, L.; GAO, J. Generating test cases for context-aware applications using bigraphs. In: *Proceedings of the 8<sup>th</sup> International Conference on Software Security and Reliability (SERE)*, San Francisco/CA - USA: IEEE Computer Society, p. 137–146, 2014.

---

# Um Processo de Busca por Sistemas Adaptativos

---

---

Uma das atividades desenvolvidas neste trabalho foi a busca por sistemas candidatos a análise de suas propriedades, que estão disponíveis em repositórios de código fonte na *Web*. Esta busca teve como objetivo elencar aqueles sistemas em que seus autores mencionaram que realizam comportamento adaptativo. Neste processo, identificou-se algumas peculiaridades a respeito dos sistemas, indicando-se a predominância da linguagem de programação java e sistemas para dispositivos móveis. Este resultado direcionou a busca aos sistemas para dispositivos móveis, que recuperados na condução da Revisão Sistemática (RS).

## A.1 Planejamento e Execução da Seleção

Inicialmente, definiram-se os repositórios de código fonte para identificar SAs que estão disponíveis para *download*. Os repositórios foram o *GitHub*<sup>1</sup>, o *SourceForge*<sup>2</sup>, e o *Bitbucket*<sup>3</sup>. Nestes repositórios, aplicaram-se *strings* de busca cujas palavras-chave foram:

---

<sup>1</sup><http://github.com/> - acessado em 2 de maio de 2015

<sup>2</sup><http://sourceforge.net/> - acessado em 9 de maio de 2015

<sup>3</sup><http://bitbucket.org/> - acessado em 13 de maio de 2015



*context-aware*; *adaptive system*; *self-adaptive*; *adaptive software*; ou *autonomic*, de modo que, obtiveram-se sistemas cujas descrições continham tais palavras-chave. Na tabela A.1, apresenta-se a quantidade de sistemas recuperados de cada palavra-chave em cada um dos repositórios de código fonte.

**Tabela A.1:** Sistemas recuperados ao aplicar as *strings* de busca nos repositórios

	GitHub	SourceForge	Bitbucket	Totais
<i>context-aware</i>	216	1300	7	1523
<i>adaptive system</i>	211	1750	41	2002
<i>self-adaptive</i>	44	0	7	51
<i>adaptive software</i>	41	0	16	57
<i>autonomic</i>	117	16	80	213
				3846

Observou-se que a máquina de busca referente ao *SourceForge* não aplicou adequadamente os filtros pelas *strings*, inclusive, a utilização do termo *adaptive system* retornou resultados iguais aos termos contendo a *substring* “adaptive” (i.e. *self-adaptive* e *adaptive software*). Por este motivo as células referentes a tais termos, na tabela A.1, contém o valor zero.

A fim de filtrar os sistemas obtidos nesta aplicação de *strings*, definiu-se o critério de seleção “Sistemas cujas descrições mencionaram algum tipo de comportamento adaptativo”. Este critério foi escolhido para que fossem avaliados os sistemas que seus respectivos autores registraram-os – nos repositório de código fonte – mencionando que estes sistemas tinham comportamento adaptativo. Ao aplicar tal critério de seleção, obteve-se um conjunto de 122 SAs, como se resume na tabela A.2.

**Tabela A.2:** Sistemas selecionados ao aplicar o critério de seleção

	GitHub	SourceForge	Bitbucket	Totais
<i>context-aware</i>	44	6	2	52
<i>adaptive system</i>	36	3	2	41
<i>self-adaptive</i>	4	3	1	8
<i>adaptive software</i>	2	3	0	5
<i>autonomic</i>	4	1	11	16
				122

Ao avaliar a lista dos sistemas obtidos na tabela A.2, caracterizou-se a linguagem de programação que foi utilizada no desenvolvimento de cada SA. Esta análise é apresentada na tabela A.3.

**Tabela A.3:** Sistemas caracterizados por linguagem de programação

Linguagem	Total
Java	66
JavaScript	14
C++	10
C#	8
Python	6
Objective-C	6
C	3
Arduino	2
NetLogo	2
PHP	2
Ruby	2
Assembly	1
Groovy	1
MATLAB	1

Na tabela A.3, ao observar a predominância da linguagem de programação Java, com relação às outras, selecionou-se apenas os 66 sistemas desenvolvidos em tal linguagem. Desses 66 sistemas, o código fonte de 4 deles não estavam disponíveis, assim, 62 sistemas desenvolvidos em java foram analisados.

Um outro dado levantado é a predominância dos sistemas para dispositivos móveis, 40 de um total de 62. Desses 40, 39 foram desenvolvidos utilizando a plataforma Android<sup>4</sup>. Neste sentido, caracterizou-se quais dos estudos primários selecionados na Revisão Sistemática (RS) realizada neste trabalho, e apresentado no Capítulo 3, estavam no contexto de dispositivos móveis (ou avaliavam suas abordagens utilizando estes dispositivos). Dos 6 trabalhos encontrados (Griebe e Gruhn, 2014; Merdes et al., 2006; Sama et al., 2010a, 2008, 2010b; Ye et al., 2009), 3 deles (Sama et al., 2010a, 2008, 2010b) utilizam o sistema chamado *PhoneAdapter*.

Ressalta-se também que, durante a condução da RS (Capítulo 3), os autores dos estudos primários foram contatados a fim de solicitar a disponibilização do código fonte dos sistemas utilizados por eles nas avaliações dos estudos. Entretanto, apenas um dos autores dos estudos Fredericks et al. (2014, 2013) respondeu e este, por sua vez, não disponibilizou o código fonte do sistema. Além disso, uma busca *ad-hoc* e exaustiva foi realizada a fim de encontrar tais sistemas, mas sem sucesso de obtenção.

---

<sup>4</sup><http://developer.android.com/> – acessado em 15 de outubro de 2015.

---

## Uma Análise de Propriedades de Sistemas Adaptativos

---

---

Ao passo que a busca de sistemas foi realizada, necessitou-se da criação de critérios para avaliar o código-fonte dos sistemas recuperados. Esta medida foi tomada a fim de justificar a seleção dos sistemas, registrar o processo para uma replicabilidade do mesmo e aumentar o nível de qualidade da condução do estudo exploratório, proposto para avaliar a abordagem deste trabalho. Neste sentido, definiu-se um conjunto de propriedades de Sistemas Adaptativos (SAs) que guiaram a classificação dos sistemas avaliados. Para tal classificação, definiu-se, também, um processo para a identificação das propriedades nos sistemas.

O processo de identificação permitiu a caracterização do código-fonte dos sistemas, de acordo com o conjunto das propriedades de SAs. Esta caracterização auxiliou nas conclusões de quais propriedades estão presentes nos sistemas e qual sistema poderia ser utilizado para um estudo exploratório.

## B.1 O Processo de Identificação de Propriedades nos Sistemas Adaptativos

Apesar da definição de um conjunto de propriedades de SAs, a identificação destas propriedades nos sistemas não é uma tarefa trivial (Cheng et al., 2009). Isto pois, em geral, as propriedades são específicas às características do sistema alvo, o que leva a uma dificuldade em identificá-las em outros contextos. Esta dificuldade diz respeito às propriedades que estão entrelaçadas e espelhadas pelo o código-fonte dos sistemas (Cheng et al., 2009; Müller et al., 2008; Oreizy et al., 1999), isto implica das propriedades não estarem evidentes no código-fonte, dificultando a identificação das mesmas.

Entretanto, ressalta-se novamente que algumas propriedades dependem de outras (Iglesia, 2014), e.g.: i) para o Monitor desempenhar sua função, ele precisa de um Sensor; ii) para o Analisador desempenhar sua função, ele precisa de um Monitor; iii) para o Executor desempenhar sua função, ele precisa de um Atuador; iv) para existir um Gerenciador autônomo, as funções monitoramento, análise e execução devem existir (Monitor, Analisador e Executor, respectivamente); v) para existir um Sistema gerenciado, ele deve ser gerenciado por um Gerenciador autônomo, por meio de Sensores e Atuadores; vi) para os componentes trabalharem juntos, eles devem se comunicar entre si (e.g. por evento ou por tempo). Sendo assim, ao identificar as propriedades “Gerenciador autônomo” e “Sistema gerenciado”, as demais podem ser abstraídas ao identificar as dependências entre essas propriedades.

Com base nas dependências mencionadas acima – de *i* até *vi* – algumas questões foram utilizadas, de modo que, durante a análise de cada sistema, os questionamentos e diretrizes da tabela B.1 guiaram a análise dos sistemas encontrados.

Essas questões e diretrizes foram definidas a partir do trabalho de Iglesia (2014), que propôs diretrizes para projetar SAs. Além disso, Serikawa et al. (2015) propuseram algumas diretrizes preliminares para a identificação de Monitores em código fonte.

No processo de identificação das propriedades – como motivação inicial para análise dos códigos-fontes dos sistemas – utilizou-se o trabalho de Serikawa et al. (2015), que assim como os autores, realizou-se a leitura dos códigos-fontes a fim de abstraí-los e mapeá-los. Este processo possibilitou uma visão geral do sistemas sob análise.

**Tabela B.1:** Questões e diretrizes para identificar propriedades de SAs

Nº	Questão	Diretriz
1	Contém ao menos um <i>Gerenciador autônomo</i> e um <i>Sistema gerenciado</i> ?	<i>threads</i> que se comunicam entre si, de modo que uma controla (i.e. gerenciador autônomo) a sincronização da outra (i.e. sistema gerenciado).
2	Contém ao menos um <i>Sensor</i> e um <i>Atuador</i> ?	foi possível encontrar algum tipo de componente que obtém (sensor) e altera (atuador) dados do sistema/ambiente.
3	Contém ao menos um <i>Monitor</i> ?	alguma das <i>threads</i> (ou até mesmo trechos de código) tem os objetivos de obter (e.g. recebimento de dados por parâmetro) e/ou preparar (e.g. conversão entre tipo de dados) os dados dos sensores e/ou do sistema.
4	Contém ao menos um <i>Analisador</i> ?	alguma das <i>threads</i> (ou até mesmo trechos de código) tem o objetivo de validar os dados do monitor, classificado como tal acima.
5	Contém ao menos um <i>Planejador</i> ?	o sistema gerencia planos e metas futuras, de acordo com o Conhecimento ( <i>Knowlegde - K</i> ) do sistema).
6	Contém ao menos um <i>Executor</i> ?	alguma das <i>threads</i> (ou até mesmo trechos de código) tem o objetivo de executar adaptações ao sistema.
7	Contém ao menos uma <i>interação baseada em Eventos ou em Tempo</i> ?	as <i>threads</i> enviam mensagens umas às outras por meio de disparo de eventos e/ou um relógio contador.
8	O sistema armazena dados dos sensores?	os dados de sensores são inseridos em um banco de dados.
9	Contém Conhecimento ( <i>Knowlegde - K</i> ) estático?	o usuário não pode personalizar o comportamento do sistema.
10	Contém Conhecimento ( <i>Knowlegde - K</i> ) dinâmico?	o usuário pode personalizar o comportamento do sistema.
11	Adiciona/remove recursos no tempo de execução?	o sistema adiciona/remove sensores ou componentes no tempo de execução.

## B.2 As Propriedades nos Sistemas Adaptativos

Na tabela 4.3, apresenta-se uma visão ampla das propriedades presentes nos 5 SAs selecionados. Observa-se que a propriedade Planejador não está presente em nenhum dos sistemas, de modo que, isto já era previsto de acordo com Ramirez e Cheng (2010).

Apesar da diferença entre os sistemas – de acordo com os totais das propriedades – ser pequena, destaca-se o SA5-PhoneAdapter como “o que tem mais propriedades de SAs”, incluindo-se obrigatórias e não obrigatórias (sendo 12). Um outro dado a ressaltar é com relação a propriedade “Adiciona/remove recursos no tempo de execução”, tal propriedade está presente em apenas dois sistemas (SA1-Novis e SA5-PhoneAdapter).

## B.3 Características dos Sistemas Adaptativos Selecionados

Dos 62 sistemas desenvolvidos em java, em apenas 4 possibilitou-se identificar todas as propriedades definidas como critério de seleção. Além disso, como levantado no capítulo A, incluiu-se o sistema PhoneAdapter. A avaliação deste sistema possibilitou encontrar todas as propriedades obrigatórias de SAs, da tabela 2.1, de modo que na tabela B.2 resume-se os SAs selecionados.

**Tabela B.2:** Sistemas com as propriedades obrigatórias.

ID	Nome	Descrição	Url
SA1	Novi	Um assistente pessoal sensível ao contexto	<a href="http://sourceforge.net/projects/novi/?source=directory">http://sourceforge.net/projects/novi/?source=directory</a>
SA2	STMoblie	Um sistema que reconhece atividades do usuário e realiza notificações	<a href="https://bitbucket.org/sbobek/stmobile/wiki/Home">https://bitbucket.org/sbobek/stmobile/wiki/Home</a>
SA3	Context-Aware Music Player	Um tocador de músicas sensível ao contexto	<a href="https://github.com/Whipler/Context-aware-musicplayer">https://github.com/Whipler/Context-aware-musicplayer</a>
SA4	Proxilence	Um sistema que altera o audio de acordo com a localização do usuário	<a href="https://github.com/clamped/Proxilence">https://github.com/clamped/Proxilence</a>
SA5	PhoneAdapter	Um sistema sensível ao contexto baseado em regras definidas pelo usuário	<a href="http://sccpu2.cse.ust.hk/afchecker/phoneadapter.html">http://sccpu2.cse.ust.hk/afchecker/phoneadapter.html</a>

Para cada um dos SAs selecionados, abstraiu-se e criou-se um modelo em alto nível para representá-lo graficamente.

Com a avaliação do código-fonte dos sistemas, identificaram-se *threads* que observam e realizam mudanças no *smartphone*. Estas *threads* foram classificadas como o Gerenciador autônomo. Neste sentido, classificaram-se os recursos do *smartphone* como Sistema gerenciado.

Lalanda et al. (2013) classificam o Sistema gerenciado como “Recursos gerenciados”, a fim de especificar e dividir o Sistema gerenciado em vários módulos gerenciados. Neste caso, seriam vários os recursos a serem classificados, e.g., para um GPS deveria ter um sistema gerenciado exclusivo. Entretanto, a fim de simplificar os recursos do *smartphone* foram classificado como unicamente como um “Sistema gerenciado”. o Sistema gerenciado

fornece Sensores e Atuadores, de modo que respectivamente, um disponibiliza dados do Sistema gerenciado e outro permite a alteração do Sistema gerenciado.

Nos SAs selecionados destacam-se algumas características particularidades. Ressalta-se também que essas classificações dependem da abstração e entendimento do domínio (Iglesia, 2014; Lalanda et al., 2013).

### **B.3.1 As Características do SA1-Novi**

No sistema SA1-Novi contém uma interface com o usuário que possibilita a definição de uma agenda, contendo as atividades e data/hora do usuário.

A propriedade Monitor, presente na figura B.1, representa 4 monitores que foram encontrados. O primeiro observa eventos referentes à tela do usuário. O segundo observa eventos referentes à conexão de redes de redes sem-fio. O terceiro coleta mudanças de localização. O quarto, por fim, coleta mudanças da gravidade do usuário.

Utilizando-se a agenda definida pelo usuário para cada um dos 4 Monitores pode-se identificar um Analisador, presente na figura B.1. Estes analisadores lidam com os dados obtidos dos Monitores, de modo que depois, estes dados são disponíveis para manipulação pelo Executor. Ressalta-se que, e.g., Iglesia (2014) classifica este tipo de propriedade como um “pré-processador” do Monitor, que prepara os dados em conjunto com o Monitor.

A propriedade Executor, presente na figura B.1, representa as ações que são direcionadas aos atuadores, que realizam diretamente as mudanças no Sistema gerenciado.

Destaca-se, também, neste sistema a propriedade Conhecimento (*Knowledge* - K) como estático, i.e., o usuário não personaliza o comportamento do sistema.

No grupo de sensores deste SA, incluem-se redes sem-fio; sensores de tela; atividades do usuário; *GPS*; gravidade do usuário e relógio. Além disso, no grupo de atuadores, estão presentes funcionalidades para notificações e sugestões para o uso novas atividades com base em atividades realizadas anteriormente.

Uma outra propriedade a se destacar é a “Adiciona/remove recursos no tempo de execução”, que neste sistema classificou-se o recurso de “adicionar novas aplicações”. O loop de controle mencionado – com as propriedades Monitor, Analisador e Executor – fornece um recurso de sugestões de aplicações. Estas sugestões são utilizadas por um serviço do sistema que adiciona novas aplicações no sistema em questão. Esta adição tem por objetivo incrementar recursos ao sistema.

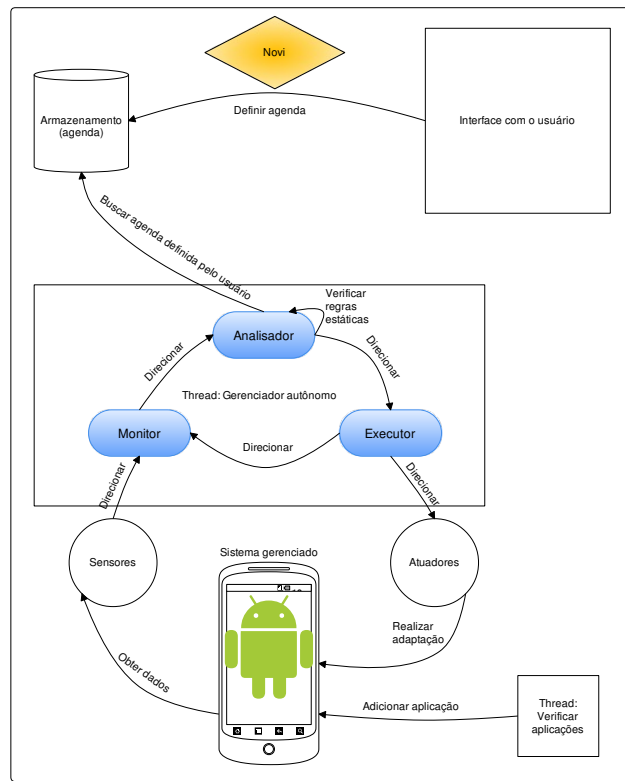


Figura B.1: Arquitetura do sistema SA1-Novi.

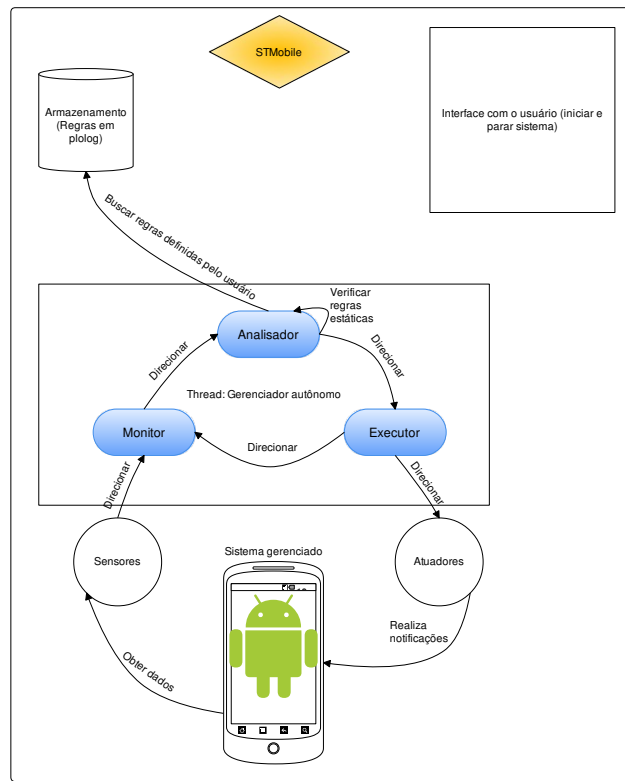
### B.3.2 As Características do SA2-STMobile

No sistema SA2-STMobile contém uma interface com o usuário que possibilita apenas o usuário iniciar ou parar o sistema.

O estado de um usuário é dado por sua localização e atividade, se o estado do usuário conter ameaça – julgada por regras prolog – uma notificação é encaminhada ao usuário. Além disso, estão presentes funcionalidades para notificações e sugestões para o uso novas atividades com base em um arquivo contendo regras em linguagem lógica prolog;

A propriedade Monitor, presente na figura B.2, representa 2 monitores que foram encontrados. O primeiro observa eventos referentes a localização do usuário. O segundo, por fim, observa eventos referentes as atividades que o usuário está realizando.





**Figura B.2:** Arquitetura do sistema SA2-STMobile.

A propriedade Analisador, presente na figura B.2, representa um método sincronizado (i.e. CheckData()) que as *threads* do sistema o utilizam para validar os dados obtido pelo monitor.

A propriedade Executor, presente na figura B.2, representa notificações que são direcionadas aos atuadores, que realizam diretamente o envio de mensagens ao Sistema gerenciado.

Destaca-se, também, neste sistema a propriedade Conhecimento (*Knowledge - K*) como dinâmica e estática, i.e., o usuário pode personalizar parte do comportamento do sistema, utilizando um arquivo *prolog*. Entretanto, ressalta-se que teria que ser um usuário que conheça a linguagem *prolog* e também entenda de regras de negócio específicas do domínio.

No grupo de sensores deste SA, incluem-se atividades do usuário e *GPS*. Além disso, no grupo de atuadores, estão presentes funcionalidades para notificações e sugestões para o uso novas atividades com base em um arquivo contendo regras em linguagem lógica *prolog*.

### B.3.3 As Características do SA3-Context-Aware Music Player

No sistema SA3-Context-Aware Music Player contém uma interface com o usuário que possibilita a interação com um tocador de músicas. Neste tocador o usuário pode realizar funções comuns a um tocador de músicas qualquer (e.g. iniciar uma música, parar e adicionar novas músicas, descrevendo-as). A principal funcionalidade a se destacar é: o sistema adapta a lista de músicas que está em execução de acordo com o contexto do usuário (e.g. se o usuário está correndo, músicas de corrida serão adicionadas na lista de músicas). Além disso, caso o local em que o usuário encontra-se é “barulhento”, o sistema aumentará o volume do áudio.

A propriedade Monitor, presente na figura B.3, representa 2 monitores que foram encontrados. O primeiro observa eventos referentes a localização do usuário e o segundo, por fim, observa eventos referentes as atividades que o usuário está realizando.

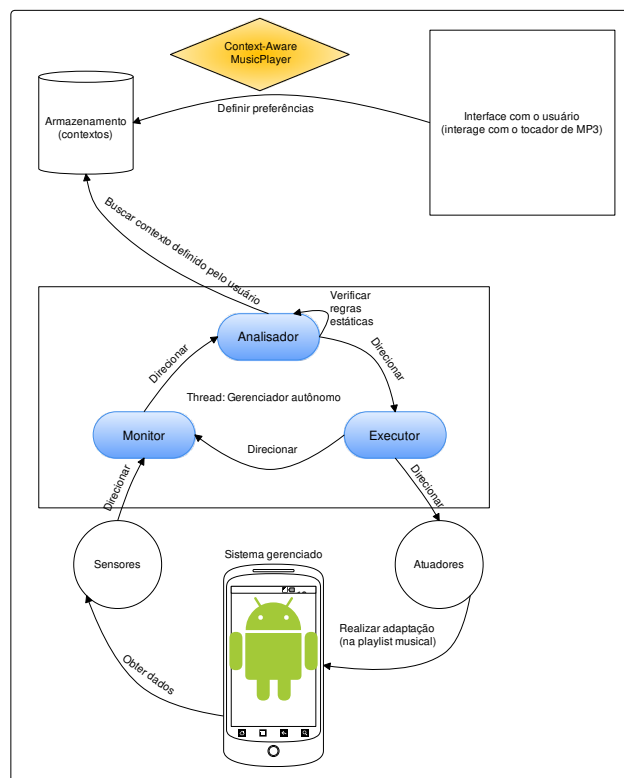


Figura B.3: Arquitetura do sistema SA3-Context-Aware Music Player.

A propriedade Analisador, presente na figura B.3, inclui 2 analisadores. O primeiro verifica a necessidade de atualizar a lista de músicas e o segundo, por fim, verifica se a atividade atual do usuário é diferente da anterior.

A propriedade Executor, presente na figura B.3, representa 2 executores. O primeiro encaminha mudanças na lista de músicas e o segundo encaminha mudanças no volume do áudio.

Destaca-se, também, neste sistema a propriedade Conhecimento (*Knowledge - K*) como estático, i.e., o usuário não personaliza o comportamento do sistema.

O grupo de sensores deste SA inclui atividades do usuário e *GPS* e o grupo de atuadores inclui métodos de mudança na lista de músicas e no volume do áudio.

### **B.3.4 As Características do SA4-Proxilence**

No sistema SA4-Proxilence contém uma interface com o usuário que possibilita a visualização de um mapa, em que o usuário pode visualizar os lugares em que já esteve, registrando novos lugares de acordo com a movimentação do usuário. Além disso, durante a movimentação, quando o sistema identifica uma proximidade com um lugar registrado, ele muda o volume do áudio.

A propriedade Monitor, presente na figura B.4, representa 1 monitor que foi encontrado. Este monitor lida com as mudanças na localização do usuário, capturada com o respectivo sensor.

A propriedade Analisador, presente na figura B.4, inclui 1 analisador. Este analisador verifica se o local informado pelo monitor já foi armazenado antes, i.e., verificando-se se a localização atual já foi percorrida pelo usuário.

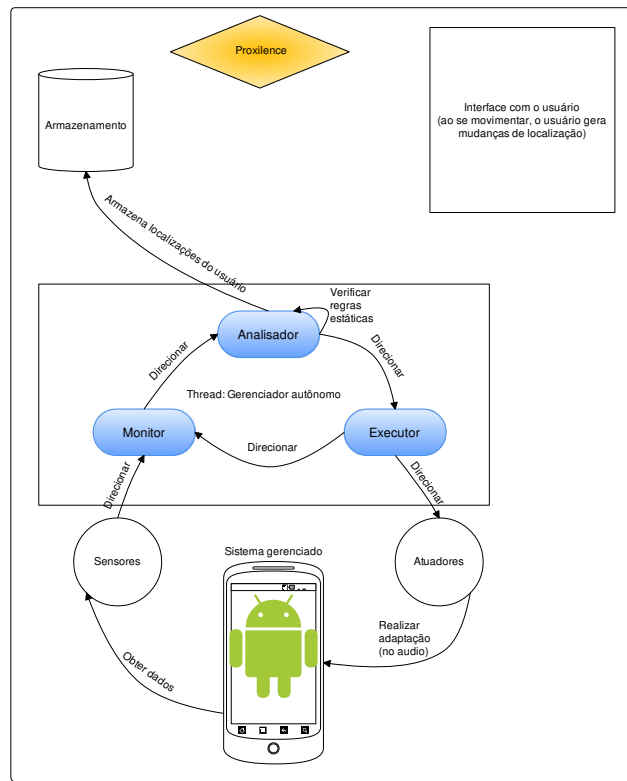
A propriedade Executor, presente na figura B.4, representa 1 executor. Este executor encaminha a um atuador mudanças a serem realizadas no volume do áudio.

Destaca-se, também, neste sistema a propriedade Conhecimento (*Knowledge - K*) como estático, i.e., o usuário não personaliza o comportamento do sistema.

O sensor classificado foi o de *GPS* e o atuador aquele que, além de inserir no banco de dados a nova localização do usuário, muda o volume do áudio.

### **B.3.5 As Características do SA5-PhoneAdapter**

No sistema SA5-PhoneAdapter contém uma interface com o usuário que possibilita a criação de contextos e regras, de modo que, o usuário pode personalizar o sistema a fim de se adaptar de acordo com as regras definidas pelo usuário. As interfaces gráficas do



**Figura B.4:** Arquitetura do sistema SA4-Proxillence.

o sistema permite o usuário definir os contextos que a aplicação terá e ainda como o sistema deve se comportar em determinadas transições entre contextos.

A propriedade Monitor, presente na figura 2.6, representa 3 monitores que foram encontrados. O primeiro observa eventos referentes a data e hora do usuário, o segundo lida com eventos disparados pelo *bluetooth* e o terceiro, por fim, observa eventos referentes à localização do usuário.

A propriedade Analisador, presente na figura 2.6, representa um método (i.e. *checkRules()*) de validação de regras. Este método lida com os dados encaminhados pelos monitores, a fim de verificar quais as atuais regras que satisfazem ao usuário.

A propriedade Executor, presente na figura 2.6, representa 1 executor. Este executor encaminha a um atuador mudanças – as quais satisfizeram as regras e contextos do usuário – a serem realizadas no volume do áudio e no tipo de toque do *smartphone* (e.g. *vibracall*).

Destaca-se, também, neste sistema a propriedade Conhecimento (*Knowledge* - K) como dinâmico e estático, i.e., o usuário pode personalizar parte do comportamento do sistema. Ressalta-se que as principais regras do sistema – aquelas que envolvem às transições entre os contextos – são dinâmicas, de modo que, o usuário decide quais e quando serão as transições entre os contextos.

Os sensores classificados foram o de *GPS* e o de *Bluetooth*. Além disso, os atuadores classificados referem-se ao que altera o volume do áudio, ao que muda o toque do *smartphone* e ao que muda o modo *Avião* do *smartphone* (i.e. ligado ou desligado).

Uma outra propriedade a se destacar é a “Adiciona/remove recursos no tempo de execução”, que neste sistema classificou-se o recurso de “adicionar novos dispositivos *bluetooth*”. O sistema contém uma *thread* que verifica de tempos-a-tempos se novos dispositivos *bluetooth* podem ser adicionados, ao encontrar esses dispositivos os mesmos são adicionados no sistema no tempo de execução.

## B.4 Análise e Discussão dos Sistemas

Apesar dos SAs selecionados serem diferentes uns dos outros, tanto com relação ao domínio quanto ao código-fonte, em tais sistemas possibilitou-se a identificação das propriedades obrigatórias de SAs. Isto possibilita uma semelhança entre os SAs com relação ao seus funcionamentos, e.g., pode-se mencionar o modelo MAPE-K. Mesmo que os sistemas não foram projetados seguindo o MAPE-K, alguns componentes do modelos puderam ser encontrados, devido a sua importância no funcionamento de um SA (e.g. um gerenciador autônomo de um SA não pode funcionar sem Coleta, Análise e Decisão (Lalanda et al., 2013)).

Um outro ponto a se destacar – no código-fonte dos sistemas – é a utilização de ambas interações baseadas em tempo e em eventos, i.e., além de existirem partes de código-fonte que utilizam relógios e disparadores de métodos, existem partes de código-fonte que utilizam ouvintes e disparadores de métodos. Ressalta-se, a predominância da interação baseada em eventos, em todos os sistemas ela está presente e em alguns deles a interação baseada em tempo.

Um primordial aspecto a se mencionar é referente a compreensão de programa, que se pode dividir em compressão teórica e empírica (Soh, 2011). A avaliação de código-fonte é dependente do avaliador e de seu conhecimento a respeito do domínio do sistema sob avaliação. Neste sentido, sendo a compreensão de programa baseada em processos cognitivos do próprio avaliador, é difícil a classificação exata das propriedades nos sistemas.

No ponto de vista do teste, um ponto a ressaltar é avaliação de código-fonte estático, a fim de identificar propriedades em “sistemas que se adaptam no tempo de execução”. Se já é impossível provar que um sistema está correto (Myers et al., 2011), é ainda mais difícil detectar e evitar resultados incorretos do sistema no tempo de execução (Niebuhr e Rausch, 2007).

A partir da análise dos sistemas evidencia-se incipiência em algumas de suas características. Esta incipiência apresenta, de fato, ameaças à validade na análise dos sistemas, pelo fato dos sistemas não estarem em perfeitas condições de funcionalidade (i.e. testes já realizados pelos autores do código-fonte), a abstração realizada a partir do código-fonte pode conter viés com relação a não execução de funcionalidades dos sistemas.

Em um dos sistemas, identificou-se problemas logo em sua compilação (e.g. SA3-Context-Aware Music Player, e.g. SA4-Proxilence), principalmente por causa do uso de bibliotecas utilizadas que se tornaram obsoletas (e.g. *LocationClient*<sup>1</sup>). Outros problemas também como travamentos de tela, devido a *threads* não sincronizadas (e.g. SA1-Novi, SA2-STMobile), e conversões entre tipos de dados, que fizeram o sistema parar sua execução (e.g. SA5-PhoneAdapter).

Na maioria sistemas, apresenta-se uma “promessa” de que no futuro as funcionalidades serão aprimoradas. No SA1-Novi, e.g., identificou-se monitores de localização que, apesar obterem dados do sensor de *GPS*, os dados não são manipulados.

Dos 5 sistemas, o SA5-PhoneAdapter – apesar de erros<sup>2</sup> de conversões entre tipos de dados – ele foi o único sistema em que foi possível simular execuções de um Gerenciador autônomo (i.e. coleta, análise e decisão) e de um Sistema gerenciado (i.e. acesso a sensores e atuadores). Além disso, o SA5-PhoneAdapter permitiu a definição de possíveis contextos e regras de transições entre contextos.

---

<sup>1</sup><https://developers.google.com/android/reference/com/google/android/gms/location/package-summary--acessadoem19/10/2015>

<sup>2</sup>Como tais ocorrências apenas foram encontradas durante um processo de depuração do sistema, classificou-se como “erros”.

---

## Demais Tabelas da Revisão Sistemática

---

Neste apêndice apresentam-se as Tabelas auxiliares referente à Revisão Sistemática (RS) da literatura, realizado neste trabalho e descrito no Capítulo 3.

- Tabela C.1: Nesta tabela apresentam-se os autores elencados como relevantes da RS. Nela estão presentes o Nome do autor, a quantidade de Estudos que o respectivo autor está em autoria e a Porcentagem relacionada à seleção de estudos como um todo.
- Tabelas C.2, C.3, C.4 e C.5: Nestas tabelas apresentam-se os dados extraídos referentes às abordagens de teste encontradas no RS realizado. Nela encontram-se uma identificação para o estudo (ID), Nome do autor e Ano de publicação, uma síntese da Proposta do respectivo estudo e as principais rescrições elencadas pelo autor e/ou analisadas ao avaliar o respectivo estudo.
- Tabelas C.6 e C.7: Nestas tabelas apresentam-se dados extraídos referentes aos desafios impostos à atividade de teste de SAs. Nela encontram-se uma identificação para o desafio (ID), a citação do respectivo Autor e uma descrição extraída dos respectivos estudos, nas palavras dos próprios autores.
- Tabelas 4.1 e 4.2: Nestas tabelas são apresentados os desafios específicos que foram caracterizados neste trabalho, destacando na coluna IDC o respectivo desafio genérico.

- Tabelas C.8 e C.9: Nestas Tabelas apresentam-se dados extraídos referentes aos defeitos caracterizados pelos autores dos respectivos estudos. Nela encontram-se uma identificação para o defeito (ID), Autor e Ano de publicação do estudo, o Nome do defeito e a Descrição do defeito.



**Tabela C.1:** Autores que estão em mais de um dos estudos selecionados.

	Nome	Estudos	Porcentagem
1	King, T.M.	7	17,50
2	Tse, T.H.	7	17,50
3	Wang, Huai	7	17,50
4	Chan, W.K.	6	15,00
5	Clarke, P.J.	4	10,00
6	Elbaum, S.	4	10,00
7	Lemos, R.	4	10,00
8	Lu, Heng.	4	10,00
9	Ramirez, A.J.	4	10,00
10	Rosenblum, D.S.	4	10,00
11	Wang, Zhimin	4	10,00
12	Camara, J.	3	7,50
13	Laranjeiro, N.	3	7,50
14	Niebuhr, D.	3	7,50
15	Rausch, A.	3	7,50
16	Sama, M.	3	7,50
17	Ventura, R.	3	7,50
18	Vieira, M.	3	7,50
19	Yau, S.	3	7,50
20	Allen, A.	2	5,00
21	Baudry, B.	2	5,00
22	Chen, T.Y.	2	5,00
23	Cheng, B.H.C.	2	5,00
24	Cruz, R.	2	5,00
25	Fredericks, E.	2	5,00
26	Klein, J.	2	5,00
27	Matalonga, S.	2	5,00
28	Micskei, Z.	2	5,00
29	Morales, B.	2	5,00
30	Munoz, F.	2	5,00
31	Puschel, G.	2	5,00
32	Rodrigues, F.	2	5,00
33	Stevens, R.	2	5,00
34	Travassos, G.H.	2	5,00
35	Wei, J.	2	5,00

**Tabela C.2:** Sínteses dos estudos primários - Parte 1.

ID	Autor	Ano	Proposta	Restrições
S1	Kephart e Chess	2003	Uma visão geral da computação autônoma, nada proposto	Utilizou-se um exemplo ilustrativo
S2	Flores et al.	2004	Um modelo abstrato para interoperabilidade semântica. Aplicam-se estratégias de teste baseadas em informações de contexto de componentes e de usuários	Utilizou-se um exemplo ilustrativo
S3	Tse et al.	2004	Propôs-se investigar relações metamórficas, de modo que o <i>middleware</i> é testado como uma caixa preta. Sendo assim, comportamento assíncrono e explosões de combinações podem ser encapsuladas	Utilizou-se um exemplo ilustrativo
S4	Chan et al.	2006	Uma abordagem de teste de integração baseada em <i>middleware</i> ciente de contexto. Propuseram-se o uso de pontos iniciais e finais de casos de teste, além disso uma checagem de relação entre múltiplos casos de teste	Requer-se grande conhecimento do domínio de aplicação
S5	Lu et al.	2006	Propuseram-se medidas para adequação de critérios de teste de fluxo de dados	A ferramenta de avaliação pode conter defeitos não revelados, principalmente, por se tratar de uma aplicação de pequeno porte
S6	Merdes et al.	2006	Uma abordagem que combina o teste no tempo de execução com recursos conscientes de contexto	Não é considerada a dependência do contexto em situações extra dispositivos e usuários; a performance não foi avaliada; e desenvolveu-se apenas testes com alguns cenários
S7	King et al.	2007b	Propôs-se um framework de autoteste que valida o sistema através de teste de regressão	Utilizou-se um protótipo e implementou-se a abordagem de Validação de Replicação, entretanto, esta abordagem é ineficaz para sistemas de grande porte, pois ela faz inúmeras cópias do sistema
S8	Lu	2007	Propôs-se a geração de casos de teste que captura contexto; propôs-se um modelo para representar um contexto dirigido por camadas; e, por fim, uma forma de definir oracles de teste	Apesar de se ter utilizado uma equação para validar o oracle de teste, a abordagem não é validada com um experimento
S9	Niebuhr e Rausch	2007	Uma abordagem baseada em componentes duplicados, a fim de testar a cópia antes de satisfazer uma configuração	Utilizou-se um exemplo ilustrativo e não se avaliou dependências cíclicas
S10	Wang et al.	2007	Uma abordagem para melhorar a consciência de contexto de um conjunto de casos de teste. Identificaram-se pontos de programa que podem afetar o comportamento de um SA e geraram-se variações potenciais	as ferramentas de análise estática são pouco escaláveis, os critérios para avaliar foram preliminares e utilizou-se um exemplo ilustrativo

**Tabela C.3:** Sínteses dos estudos primários - Parte 2.

ID	Autor	Ano	Proposta	Restrições
S11	Jaw et al.	2008	Propuseram-se a geração de casos de teste que captura contextos; um modelo para representar um contexto orientado a camadas e uma forma de definir oráculos de teste	Avaliaram-se apenas dois cenários de teste
S12	Lu et al.	2008	Propuseram-se um conjunto de equações para estudar os mecanismos de feedback e um conjunto de critérios de teste baseados em fluxo de dados	Utilizou-se apenas um sub conjunto do sistema para avaliar a proposta
S13	Sama et al.	2008	Uma abordagem de detecção de defeito em um sistema baseado em uma máquina de estados formal	A avaliação do sistema quando contém um número elevado de regras não foi realizada
S14	Munoz e Baudry	2009	Uma estratégia para selecionar somente uma porção do espaço do ambiente	A solução se mostrou eficiente em um sistema pequeno, mas não se sabe com sistemas de grande porte
S15	Niebuhr et al.	2009	Uma infraestrutura baseada em componentes que identifica quando os componentes se juntam ou se soltam	Utilizou-se um exemplo ilustrativo e não se avaliou dependências cíclicas e <i>overheads</i> .
S16	Ye et al.	2009	Propôs-se uma abordagem de teste para detectar desvio de comportamento potencial de uma aplicação sensível ao contexto sobre diferentes <i>middleware</i>	A abordagem não foi validada com experimento
S17	Sama et al.	2010a	Um modelo para desenvolvimento de SAs; um conjunto de defeitos; algoritmos para detecção de defeitos (baseados em máquinas de estados)	Para avaliação utilizou-se um sistema de pequeno porte
S18	Sama et al.	2010b	Propuseram-se uma arquitetura baseada em camadas para SAs e uma relação de defeitos e falhas de SAs	Dificuldades com a manipulação de defeitos; problemas com características assíncronas e utilizou-se uma meta-análise como avaliação
S19	Vassev et al.	2010	Uma abordagem para geração de casos de teste guiada por uma análise de impacto de mudanças	A geração foi focada na linguagem de especificação ASSL e utilizou-se um exemplo ilustrativo
S20	Wang et al.	2010b	Apresentou-se resultados que operadores de mutação tradicionais podem ser efetivos em SAs; introduziu-se uma análise de diversidade de contexto junto de <i>score</i> de mutação	Utilizaram-se como métrica: <i>score</i> de mutação e tempo de geração de casos de teste para matar mutantes.

**Tabela C.4:** Sínteses dos estudos primários - Parte 3.

ID	Autor	Ano	Proposta	Restrições
S21	Welsh e Sawyer	2010	Um método dirigido por modelos a fim de identificar os cenários mais prováveis para descobrir o comportamento do sistema	Depende da qualidade da avaliação do especialista do domínio
S22	Akour et al.	2011	Uma abordagem de teste de tempo de execução baseado em propagação de mudanças estruturais	O protótipo simula apenas uma mudança e o autor ainda destaca a necessidade de se realizar um experimento controlado
S23	Bartel et al.	2012	Uma abordagem para geração de mutantes e regras de adaptação	Sistema de pequeno porte e com baixo volume de dados
S24	Ferrari et al.	2011	O MS em que este trabalho se embasou.	Utilizou-se uma discussão como avaliação
S25	Garvin et al.	2011	Uma abordagem baseada em detecção de falhas, utilizadas para gerar casos de teste. Propôs-se também algoritmo para evitar falhas.	a abordagem validada foi em uma aplicação não adaptativa e o número de falhas categorizadas também foi pequeno.
S26	King et al.	2011b	Introduziu-se o conceito <i>Autonomic Self-testing (AST)</i> com a abordagem transacionada SAV (Validação baseada em bloqueio de estados)	a principal ameaça à validade é relacionada à performance
S27	Silva e Lemos	2011	Um framework para geração dinâmica de planos para teste de integração.	A abordagem assume que um componente que já foi testado não precisa ser retestado, considerou-se um número pré-determinado de componentes e utilizou-se um exemplo ilustrativo.
S28	Wotawa	2012	Propôs-se uma abordagem de modelos para diagnóstico que auxilia na geração de testes	Apesar de utilizar uma abordagem formal, com auxílio de equações, o autor não aplicou experimentos e o teste tem que focar no modelo, que não é trivial, segundo o autor
S29	Micskei et al.	2012	Propuseram-se uma linguagem para contexto e cenários, uma abordagem para gerar contextos de teste e métricas de cobertura	Utilizou-se apenas um exemplo ilustrativo
S30	Püschel et al.	2012	Uma ferramenta que auxilia e um processo para geração de casos de teste. Cria-se um modelo de falha junto à aplicação do teste baseado em modelos	Foca-se no domínio de aplicação para dispositivos móveis e utilizou-se um exemplo ilustrativo

**Tabela C.5:** Sínteses dos estudos primários - Parte 4.

ID	Autor	Ano	Proposta	Restrições
S31	Weyns	2012	Uma abordagem para verificação estática; teste baseado em modelos e diagnósticos no tempo de execução	Requeriu-se uma nova classe de propriedades (comportamento de adaptação; atualizações de segurança; progresso de atualizações; integridade de adaptação; compatibilidade e liberdade de interferência), o teste baseado em modelos ainda precisa de estudos para a etapa de diagnóstico do tempo de execução e utilizou-se um exemplo ilustrativo
S32	Fredericks et al.	2013	Um framework que apoia a definição de estratégias de teste utilizando componentes baseados no MAPEK	Utilizou-se um exemplo ilustrativo
S33	Püschel et al.	2013	Uma abordagem para aplicação de um modelo de Defeito, erro e falha, com uso do MAPE-K.	Não foi testado em aplicações reais e de grande porte e utilizou-se um exemplo ilustrativo
S34	Eberhardinger et al.	2014	Uma abordagem para diminuir o tamanho de espaço a ser testado, a fim de distinguir um comportamento correto de um incorreto	A abordagem foi baseada em cenários de variação ambiental que foi gerada a partir da especificação dos agentes, logo o comportamento não especificado não se tem como prever. Além disso, utilizou-se um exemplo ilustrativo
S35	Fredericks et al.	2014	Uma abordagem para adaptar casos de teste no tempo de execução. A abordagem monitora o ambiente para evidência de mudança, então adapta casos de teste individuais	Aplicaram-se 26 casos de teste em 15 diferentes contextos, apesar de ter sido rodado o mesmo experimento 50 vezes
S36	Griebe e Gruhn	2014	Uma abordagem de transformação de modelos para geração de casos de teste na etapa de projeto	A abordagem de geração de casos de teste requiriu modelagem estática de casos de teste
S37	Wang et al.	2014	Estratégias para fornecer um conjunto de teste sensível ao contexto; grandes resultados sobre eficiência e ineficiência do teste; e a noção da diversidade de contexto	Utilizaram-se estudos empíricos, inclusive, o autor menciona a necessidade de generalizar resultados e realizar <i>benchmarks</i> .
S38	Yu e Gao	2014	Uma abordagem para geração de casos de teste, definindo maneiras de diminuir o número de casos de teste gerados	Na abordagem assumiu-se que um <i>middleware</i> contém serviços atômicos, então não se avalia em serviços compostos.
S39	Matalonga et al.	2015b	Um trabalho correlato, entretanto, teve-se por objetivo classificar as técnicas de teste encontradas para o padrão ISO-29119-p4.	Avaliado com discussão.
S40	Matalonga et al.	2015a	Um trabalho correlato, entretanto, elenca-se 4 grupos de desafios (originados de 15 problemas), apresentaram-se 5 soluções (sendo 3 dessas relacionadas aos desafios)	Avaliado com discussão.
S41	Cámara et al.	2015	uma abordagem composta por duas fases: i) Identificar falhas nos controladores, inserindo entradas para causar defeitos na interface do controlador. ii) quantificar o impacto que falhas no controlador têm sobre a elasticidade do sistema.	A abordagem foi limitada em um conjunto de falhas/defeitos pre-estabelecidos, o teste também foi limitado a um conjunto de condições do ambiente, e 3) A abordagem leva em consideração apenas um ciclo de loop do controlador (i.e. a comunicação no tempo de execução entre controladores não foi levada em consideração).

**Tabela C.6:** Desafios para testar Sistemas Adaptativos - Parte 1.

<b>ID</b>	<b>Autor</b>	<b>Descrição</b>
1	Cámara et al. (2015)	<i>A necessidade de avaliar todo o sistema.</i>
2	Niebuhr et al. (2009)	<i>A dinamicidade de SAs, os quais não têm fronteira clara, permite que as configurações variantes não sejam previsíveis.</i>
3	Fredericks et al. (2013)	<i>A dificuldade de promover propriedades de primeira classe não funcionais e adaptáveis em propriedades testáveis do sistema.</i>
4	Niebuhr e Rausch (2007)	<i>A dificuldade de detectar e evitar possíveis resultados de configurações incorretas do sistema no tempo de execução.</i>
5	Sama et al. (2010a)	<i>A atualização assíncrona de variáveis de contexto compartilhadas.</i>
6	King et al. (2011b)	<i>A sincronização entre componentes, assegurando harmonia entre os loops de controles fechados.</i>
7	Sama et al. (2010a)	<i>A questão de garantir aos usuários a liberdade de personalizar configurações do sistema.</i>
8	Püschel et al. (2012)	<i>A questão de testar um sistema cujos contextos mudam o tempo todo.</i>
9	Eberhardinger et al. (2014)	<i>A necessidade de manipular o espaço do sistema que evolui no tempo de execução.</i>
10	Fredericks et al. (2013)	<i>A questão de identificar confiavelmente mudanças que são importantes dentro do sistema e em seu ambiente de execução.</i>
11	Püschel et al. (2013)	<i>A necessidade de manipular o espaço de decisão comportamental que impacta na estrutura do sistema e nos processos em execução.</i>
12	Munoz e Baudry (2009)	<i>A dificuldade de calcular o fluxo de dados esperado em SAs.</i>
13	Fredericks et al. (2013)	<i>A questão de avaliar precisamente o impacto nos casos de teste causados pelas adaptações do sistema.</i>
14	Fredericks et al. (2013)	<i>A dificuldade de definir limiares ou taxas de aceitação para os dados de teste.</i>
15	Fredericks et al. (2013)	<i>A questão de determinar qual a frequência que o monitoramento de dados deve ser realizado.</i>
16	Kephart e Chess (2003)	<i>A dificuldade de construir conjuntos de teste que capturem o tamanho e complexidade de sistemas e cargas de trabalho reais.</i>
17	Vassev et al. (2010)	<i>A necessidade de reduzir o número de teste que são gerados automaticamente.</i>
18	Weyns (2012)	<i>A questão para realizar checagem de modelos de propriedades do sistema relacionadas ao comportamento adaptativo.</i>
19	Yu e Gao (2014)	<i>A dificuldade para gerar casos de teste automaticamente para um ambiente em mudanças.</i>
20	Fredericks et al. (2013)	<i>A questão de determinar quais sensores, ou agregações de valores de sensores, podem mensurar propriedades desejáveis.</i>
21	Fredericks et al. (2013)	<i>A questão de resolver e atualizar as relações de rastreabilidade entre requisitos individuais, componentes e casos de teste.</i>
22	Lu et al. (2006)	<i>A dificuldade para aplicar o critério de teste de fluxo de dados, devido aos defeitos serem sensíveis ao contexto, conterem interferência do ambiente e conterem fluxo de controle sensível ao contexto.</i>
23	Lu et al. (2008)	<i>A questão de evitar estados de programa inconsistentes causados por “contextos ruidosos”.</i>

**Tabela C.7:** Desafios para testar Sistemas Adaptativos - Parte 2.

<b>ID</b>	<b>Autor</b>	<b>Descrição</b>
24	Püschel et al. (2013)	<i>A questão de lidar com propagação de erros através de componentes e múltiplos loops de controle.</i>
25	Fredericks et al. (2013)	<i>A questão de determinar que propriedades do sistema devem ser observadas.</i>
26	Jaw et al. (2008)	<i>O desafio de validar modelos de projeto que contém características de incerteza e aprendizagem.</i>
27	Wang et al. (2007)	<i>As mudanças possíveis em contexto que podem afetar o comportamento da aplicação a qualquer momento durante a execução.</i>
28	Griebe e Gruhn (2014)	<i>A questão de lidar direto com interações do usuário e antecipar mudanças em parâmetros de contexto.</i>
29	Kephart e Chess (2003)	<i>A dificuldade de antecipar o ambiente, especialmente quando ele estende através de múltiplos domínios administrativos ou empresas.</i>
30	King et al. (2011b)	<i>A ausência de ferramentas de apoio para testar propriedades adaptativas de sistemas, que se relacionam com sincronização.</i>
31	Niebuhr e Rausch (2007)	<i>A impossibilidade de provar a correteza de conexões entre interfaces de componentes no tempo de execução.</i>
32	Welsh e Sawyer (2010)	<i>A impossibilidade de garantir o comportamento correto de um sistema em mudança, cujo número de configurações pode ser imprevisível e crescente.</i>
33	Fredericks et al. (2013)	<i>A questão de limitar (ou não) a habilidade do sistema adaptar-se.</i>
34	Munoz e Baudry (2009)	<i>A explosão combinatória de variantes e condições do ambiente.</i>
35	Tse et al. (2004)	<i>A dificuldade de lidar com todas possíveis combinações de “variações de contexto” quando projetar os casos de teste.</i>
36	Tse et al. (2004)	<i>A dificuldade de lidar com condições concorrentes de contextos entre as camadas de middleware e aplicação.</i>
37	Micskei et al. (2012)	<i>A dificuldade de lidar não somente sobre os eventos recebidos, mas também sobre o estado percebido do ambiente.</i>
38	Micskei et al. (2012)	<i>A dificuldade de lidar com o grande número de possíveis situações (e.g. objetos, atributos e interações de contextos possíveis para serem especificadas).</i>
39	Micskei et al. (2012)	<i>A dificuldade de lidar com características de aprendizagem e raciocínio, que seus comportamentos podem mudar dependendo da evolução do ambiente.</i>
40	Micskei et al. (2012)	<i>A dificuldade de lidar com mecanismos para expressar e formalizar comportamento sensível ao contexto.</i>
41	Micskei et al. (2012)	<i>A dificuldade de testar situações extremas e de estresse.</i>
42	Micskei et al. (2012)	<i>A questão de definir precisamente métricas de cobertura de teste.</i>
43	Wotawa (2012)	<i>A questão de quando para o teste, ao utilizar uma abordagem dirigida por modelos.</i>
44	King et al. (2007b)	<i>A questão de ambos comportamento e estrutura do sistema poderem mudar durante a execução.</i>
45	Lu (2007)	<i>A dificuldade de lidar com uma arquitetura desenvolvida em camadas, que encapsula os contexto gerados.</i>
46	Lu (2007)	<i>A dificuldade de re-executar a execução se casos de teste são apenas valores de entrada.</i>
47	Lu (2007)	<i>A dificuldade de definir os oráculos de teste.</i>

**Tabela C.8:** Defeitos de Sistemas Adaptativos - Parte 1.

ID	Autores	Ano	Nome	Descrição
1	Sama et al.	2010a; 2008	Ativação não determinística	em dado SA é encontrado em uma configuração C1, e existe mais de uma regra de adaptação com mesma prioridade, podendo levar o SA a diferentes configurações (exemplo C2 ou C3)
2	Sama et al.	2010a; 2008	Predicado morto	existe uma regra de adaptação que pode nunca ser disparada, ou seja, um predicado morto)
3	Sama et al.	2010a; 2008	Estado morto	a propriedade de vivacidade de um estado Y, dado que Y seja um estado não final, garante que é Y possa ser acessado por outro estado X e que Y possa acessar outro estado Z. Portanto, o estado X consegue acessar ao estado Z por meio do estado Y. Assim, partindo do princípio que a transição entre estados é dada por verificação de predicados, se houver um conjunto de predicados mortos que desabilite a transição de um estado a outro, então isso gera um defeito de estado morto
4	Sama et al.	2010a; 2008	Competição de adaptação	este defeito pode ser gerado por um conjunto de regras de adaptação concorrendo entre si, de modo que ocorrem algumas adaptações intermediárias ao mesmo tempo que outras
5	Sama et al.	2010a; 2008	Ciclo de adaptação	um conjunto de resultados de regras de adaptação concorrendo entre reconfigurações do SA, periodicamente
6	Sama et al.	2010a; 2008	Estado inalcançável	uma configuração que pode nunca ser alcançada por meio de uma configuração inicial do SA
7	Wang et al.	2010b	Faltando construir	determinada configuração sugere a união entre um módulo A e um módulo B. Esta união deve gerar um terceiro módulo, o módulo C. Entretanto, o módulo B está indisponível devido a um defeito ocorrido anteriormente, ou seja, não é possível gerar o módulo C
8	Bartel et al.	2012	Compleitude ambiental	que são defeitos relacionados à aquisição de contextos referentes ao ambientes em si.
9	Bartel et al.	2012	Adaptação	que são defeitos relacionados à lógica de adaptação do sistema ao adquirir contextos.
10	Yu e Gao	2014	Aquisição de contexto	relaciona-se ao <i>middleware</i> que obtêm informações do contexto do sistema.
11	Yu e Gao	2014	Componente de regras de negócios	relaciona-se ao <i>middleware</i> que mantém as regras de negócio do sistema.
12	Yu e Gao	2014	Variável booleana	relaciona-se ao serviço que inicializa as variáveis booleanas das regras do sistema.
13	Yu e Gao	2014	Operador booleano	relaciona-se ao serviço que mantém o operador relacional de verificação booleana de variáveis booleanas.
14	Yu e Gao	2014	Prioridade/preferência booleana	relaciona-se ao serviço que verifica se determinados valores booleanos têm prioridades em questão a outros valores booleanos.
15	Yu e Gao	2014	Operadores relacionais	relaciona-se ao serviço que verifica determinados operadores relacionais.



**Tabela C.9:** Defeitos de Sistemas Adaptativos - Parte 2.

ID	Autores	Ano	Nome	Descrição
16	Yu e Gao	2014	Expressões aritméticas	relaciona-se ao serviço que verifica integridade de expressões aritméticas.
17	Cámara et al.	2015	Mensagem	relaciona-se a mensagens nulas, em branco, pré-definidas, estouro de tamanho.
18	Cámara et al.	2015	Data e hora	relaciona-se a datas e horas em branco, de formato inválido, acima do limite, abaixo do limite.
19	Cámara et al.	2015	Nome de variável	relaciona-se a formato inválido, mudança de tipo, mudança de identificador, removida.
20	Cámara et al.	2015	Valor de variável	relaciona-se a formato inválido, acima do limite, abaixo do limite.
21	Püschel et al.	2013	Interpretador de sensor corrompido	disparo ao componente Analisador, entretanto com um estado inconsistente (erro) de “dados de sensor não interpretados” que por sua vez pode ou não externalizar como uma falha de eventos.
22	Püschel et al.	2013	Interpretador de evento corrompido	disparo ao componente Analisador, entretanto com um estado inconsistente (erro) de “evento não interpretado” que por sua vez pode ou não externalizar como uma falha para outro componente.
23	Püschel et al.	2013	Interpretador de modelo corrompido	disparo ao componente Analisador, entretanto com um estado inconsistente (erro) de “modelo não interpretado” que por sua vez pode ou não externalizar como uma falha de modelos. Tais modelos podem envolver a lógica de adaptação que devem ser instanciada pelo Analisador, i.e., seriam as especificações ou restrições que o modelo deve manipular.
24	Püschel et al.	2013	Raciocínio corrompido	disparo ao componente Analisador, entretanto com um estado inconsistente (erro) de “adaptação derivada incorreta” que por sua vez pode ou não externalizar como uma falha para outro componente.
25	Püschel et al.	2013	Planejador corrompido	disparo ao componente Planejador, entretanto com um estado inconsistente (erro) de “inconsistência de planejamento” que por sua vez pode ou não externalizar como um plano inconsistente.
26	Püschel et al.	2013	Agendador corrompido	disparo ao componente Agendador, entretanto com um estado inconsistente (erro) de “inconsistência de agendamento” que por sua vez pode ou não externalizar ações não agendadas.
27	Püschel et al.	2013	Manipulador de modelo corrompido	disparo ao componente Executor, entretanto com um estado inconsistente (erro) de “inconsistência no modelo de execução” que por sua vez pode ou não externalizar ações não encaminhadas para serem executadas.
28	Püschel et al.	2013	Configurador corrompido	disparo ao componente Executor, entretanto com um estado inconsistente (erro) de “inconsistência no modelo re-configuração” que por sua vez pode ou não externalizar ações não encaminhadas para serem executadas.
29	Püschel et al.	2013	Produtor de eventos corrompido	disparo ao componente Monitor, entretanto com um estado inconsistente (erro) de “evento não interpretado” que por sua vez pode ou não externalizar ações não encaminhadas aos atuadores.
30	Püschel et al.	2013	Encaminhador corrompido	um disparo ao componente Executor, entretanto com um estado inconsistente (erro) de “inconsistência em operações de atuadores” que por sua vez pode ou não externalizar ações não executadas aos recursos gerenciados.

---

## Demais Tabelas do Estudo Exploratório

---

---

Neste apêndice apresentam-se as Tabelas auxiliares referente à avaliação dos estudos realizados e descritos no Capítulo 5.

- Tabela D.1: Nesta tabela apresenta-se o mapeamento de casos de teste, aplicando a abordagem S05 (Lu et al., 2006), utilizando o critério de teste *All Situations*.
- Tabela D.2: Nesta tabela apresenta-se o mapeamento de casos de teste, aplicando a abordagem S05 (Lu et al., 2006), utilizando o critério de teste *Def-Use Associations*.
- Tabela D.3: Nesta tabela apresenta-se o mapeamento de casos de teste, aplicando a abordagem S05 (Lu et al., 2006), utilizando o critério de teste *All Outstanding Situations*.

**Tabela D.1:** Mapeamento de casos de teste para o critério Todas Situações

ID	Descrição	Casos de teste
s1	AtivarRua	17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
s2	DesativarRua_GPSNaoValido	18, 24, 27
s3	DesativarRua_LocalCasa	19, 25, 28
s4	DesativarRua_LocalEscritorio	20, 26, 29
s5	AtivarCorrida	21, 22, 23, 24, 25, 26, 27, 28, 29
s6	DesativarCorrida_GPSNaoValido	22, 24, 25, 26
s7	DesativarCorrida_VelocidadeMenorIgual5	23, 27, 28, 29
s8	AtivarDirigindo_Geral	34, 37, 40, 41
s9	AtivarDirigindo_Casa	31, 35, 36, 38, 39, 45, 46, 47, 48, 49
s10	AtivarDirigindo_Escritorio	-
s11	AtivarDirigindo_NaRua	-
s12	DesativarDirigindo	37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49
s13	AtivarDirigindoRapido	40, 41, 42, 43, 44, 45, 46, 47, 48, 49
s14	DesativarDirigindoRapido_GPSNaoValido	41, 43, 46, 48
s15	DesativarDirigindoRapido_VelocidadeMenorIgual70	42, 44, 47, 49
s16	AtivarCasa_Bluetooth	30, 35, 38, 45, 46, 47, 48, 49
s17	AtivarCasa_GPS	31, 36, 39
s18	DesativarCasa_Bluetooth	-
s19	DesativarCasa_GPS	-
s20	AtivarEscritorio_Bluetooth	1, 3, 4, 7, 9, 11, 12
s21	AtivarEscritorio_GPS	2, 5, 6, 8, 10, 13, 14
s22	DesativarEscritorio_Bluetooth	3, 5, 11, 13
s23	DesativarEscritorio_GPS	4, 6, 12, 14
s24	AtivarReuniao	7, 8, 9, 10, 11, 12, 13, 14
s25	DesativarReuniao	9, 10, 11, 12, 13, 14
s26	AtivarSincronizacao_BT_Casa	32, 33
s27	AtivarSincronizacao_BT_Escritorio	15, 16
s28	DesativarSincronizacao	16, 33

**Tabela D.2:** Mapeamento de casos de teste para o critério Todas Associações def-use

ID	Variável de Contexto	def	use	Casos de teste
1	GPS.Disponivel	cm:n1	s1:n1	17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
2	GPS.Disponivel	cm:n1	s2:n1	18, 24, 27
3	GPS.Disponivel	cm:n1	s5:n1	21, 22, 23, 24, 25, 26, 27, 28, 29
4	GPS.Disponivel	cm:n1	s6:n1	22, 24, 25, 26
5	GPS.Disponivel	cm:n1	s13:n1	40, 41, 42, 43, 44, 45, 46, 47, 48, 49
6	GPS.Disponivel	cm:n1	s14:n1	41, 43, 46, 48
7	GPS.Disponivel	cm:n1	s17:n1	31, 36, 39
8	GPS.Disponivel	cm:n1	s19:n1	53
9	GPS.Disponivel	cm:n1	s21:n1	2, 5, 6, 8, 10, 13, 14
10	GPS.Disponivel	cm:n1	s23:n1	4, 6, 12, 14
11	GPS.Local	cm:n2	s1:n1	17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
12	GPS.Local	cm:n2	s3:n1	19, 25, 28
13	GPS.Local	cm:n2	s4:n1	20, 26, 29
14	GPS.Local	cm:n2	s17:n1	31, 36, 39
15	GPS.Local	cm:n2	s19:n1	53
16	GPS.Local	cm:n2	s21:n1	2, 5, 6, 8, 10, 13, 14
17	GPS.Local	cm:n2	s23:n1	4, 6, 12, 14
18	GPS.Velocidade	cm:n3	s5:n1	21, 22, 23, 24, 25, 26, 27, 28, 29
19	GPS.Velocidade	cm:n3	s7:n1	23, 27, 28, 29
20	GPS.Velocidade	cm:n3	s13:n1	40, 41, 42, 43, 44, 45, 46, 47, 48, 49
21	GPS.Velocidade	cm:n3	s15:n1	42, 44, 47, 49
22	Bluetooth.lista	cm:n4	s8:n1	34, 37, 40, 41
23	Bluetooth.lista	cm:n4	s9:n1	31, 35, 36, 38, 39, 45, 46, 47, 48, 49
24	Bluetooth.lista	cm:n4	s10:n1	50
25	Bluetooth.lista	cm:n4	s11:n1	51
26	Bluetooth.lista	cm:n4	s12:n1	37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49
27	Bluetooth.lista	cm:n4	s16:n1	30, 35, 38, 45, 46, 47, 48, 49
28	Bluetooth.lista	cm:n4	s18:n1	52
29	Bluetooth.lista	cm:n4	s20:n1	1, 3, 4, 7, 9, 11, 12
30	Bluetooth.lista	cm:n4	s22:n1	3, 5, 11, 13
31	Bluetooth.lista	cm:n4	s24:n1	7, 8, 9, 10, 11, 12, 13, 14
32	Bluetooth.lista	cm:n4	s26:n1	32, 33
33	Bluetooth.lista	cm:n4	s27:n1	15, 6
34	Bluetooth.lista	cm:n4	s28:n1	16, 33
35	DataHora	cm:n5	s24:n1	7, 8, 9, 10, 11, 12, 13, 14
36	DataHora	cm:n5	s25:n1	9, 10, 11, 12, 13, 14
37	volume	volume:n3	volume:n1	3, 4, 5, 6, 11, 12, 13, 14
38	volume	volume:n4	volume:n1	16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49
39	vibracall	vibracall:n2	vibracall:n1	3, 4, 5, 6, 10, 11, 12, 13, 14
40	modo_aviao	modo_aviao:n2	modo_aviao:n1	9, 10, 11, 12, 13, 14

**Tabela D.3:** Mapeamento de casos de teste para o critério Todas Situações fora do Padrão

ID	Situação negada	Caso de teste
1	!AtivarRua	57
2	!DesativarRua_GPSNaoValido	58
3	!DesativarRua_LocalCasa	59
4	!DesativarRua_LocalEscritorio	59
5	!AtivarCorrida	60
6	!DesativarCorrida_GPSNaoValido	61
7	!DesativarCorrida_VelocidadeMenorIgual5	62
8	!AtivarDirigindo_Geral	63
9	!AtivarDirigindo_Casa	64,65
10	!AtivarDirigindo_Escritorio	66,67
11	!AtivarDirigindo_NaRua	68
12	!DesativarDirigindo	69
13	!AtivarDirigindoRapido	70
14	!DesativarDirigindoRapido_GPSNaoValido	71
15	!DesativarDirigindoRapido_VelocidadeMenorIgual70	72
16	!AtivarCasa_Bluetooth	73
17	!AtivarCasa_GPS	74
18	!DesativarCasa_Bluetooth	75,76
19	!DesativarCasa_GPS	77,78
20	!AtivarEscritorio_Bluetooth	79
21	!AtivarEscritorio_GPS	80
22	!DesativarEscritorio_Bluetooth	81
23	!DesativarEscritorio_GPS	82
24	!AtivarReuniao	83,84
25	!DesativarReuniao	85,86
26	!AtivarSincronizacao_BT_Casa	87
27	!AtivarSincronizacao_BT_Escritorio	87
28	!DesativarSincronizacao	88,89