Instituto de Ciências Matemáticas e de Computação

ISSN -

Um Mapeamento Sistemático sobre a Geração de Casos de Teste para Sistemas de Transição com Entrada e Saída

Sofia Larissa da Costa Paiva

Adenilso da Silva Simão

RELATÓRIOS TÉCNICOS DO ICMC

Nº

São Carlos, SP, Brasil
Janeiro/2014

Resumo

O Teste Baseado em Modelos (TBM) envolve a geração automatizada de casos de teste a partir de modelos produzidos durante o ciclo de desenvolvimento do software. A comunidade de teste tem investigado métodos para geração de casos de teste a partir de Sistemas de Transição com Entrada e Saída (do inglês, *Input/Output Transition Systems*, IOTSs). IOTSs têm sido utilizados em TBM por serem modelos mais genéricos que Máquinas de Estado Finito (MEFs). Este trabalho descreve a condução de um mapeamento sistemático com o objetivo de identificar os métodos de geração de casos de testes a partir de IOTSs. A partir dos estudos selecionados, foram analisadas as definições do modelo utilizado para o teste, a forma de geração e aplicação dos testes, os estudos empíricos realizados e o suporte computacional oferecido.

Palavras-chaves: teste de conformidade, teste formal, sistemas de transição com entrada e saída, mapeamento sistemático.

Sumário

CAPÍTULO

1

Introdução

A atividade de teste contribui para aumentar a confiança de que o software apresenta as funções especificadas. Como essa atividade encontra limitações de tempo e recursos, o Teste Baseado em Modelos (TBM) surgiu como uma estratégia promissora já que apoia as fases de projeto e geração de testes automatizados [Utting, 2010]. A automatização é benéfica, pois o software e o modelo podem ser alterados com frequência, evitando o trabalho manual e reduzindo os custos dos testes. Desse modo, essa abordagem oferece maior confiança nos resultados e menor custo em sua aplicação [Simão, 2007; Tretmans, 2008].

Para tornar viável o emprego de TBM, é necessário utilizar métodos rigorosos para especificar o comportamento do software em desenvolvimento, uma vez que a utilização de métodos formais podem ser utilizados como a base para a automatização de partes do processo de teste e pode conduzir a um teste mais eficiente e eficaz [Hierons et al., 2009]. Assim, ambiguidades e inconsistências podem ser facilmente descobertas e corrigidas, proporcionando um software com menos defeitos. Existem diversas técnicas para modelar de maneira rigorosa uma aplicação, e uma das que vem recebendo atenção da comunidade são os Sistemas de Transição com Entrada/Saída (do inglês, *Input/Output Transition Systems*, IOTSs) [Tretmans, 2008; Weiglhofer and Wotawa, 2009].

O uso de IOTSs possibilita a modelagem de sistemas nos quais as entradas e saídas são desacopladas, e a próxima entrada pode chegar antes mesmo da saída que é produzida em resposta à entrada anterior [Petrenko and Yevtushenko, 2002]. A capacidade de aceitar entradas em qualquer estado não é adequada para modelar alguns tipos de sistemas, como sistemas interativos, pois esse tipo de sistema recusa a entrada em algumas situações. Porém, pode ser usado para modelar comportamento de processos, tais como implementações, especificações e testes, além de servir como um modelo semântico para várias linguagens formais [Tretmans, 2008; Gaudel, 2010].

IOTSs têm recebido atenção especial da comunidade de teste. Testes baseado no modelo IOTS são explorados por Tretmans [1996, 2008], o qual estabelece a teoria de teste *ioco* (do inglês, *Input/Output Conformance*). A relação *ioco* expressa uma relação formal entre modelos de implementação e de especificação, e é definida sobre um conjunto de observações feitas quando realizam-se testes em determinada implementação. Uma formalização do processo de teste de conformidade requer modelos formais para a especificação e para a implementação, além da definição formal dos experimentos de teste e observações [Heerink and Tretmans, 1998]. Muitos trabalhos sobre *ioco* podem ser encontrados na literatura [Tretmans, 1996; Heerink and Tretmans, 1998; van der Bijl et al., 2004; Weiglhofer and Aichernig, 2010; Tretmans, 2008; Weiglhofer and Wotawa, 2009; Krichen, 2010].

Um dos algoritmos mais utilizados para geração de casos de teste a partir de IOTSs é o de Tretmans [2008], que é um algoritmo recursivo e não determinístico para geração de casos de teste para a relação *ioco*. A primeira transição do caso de teste é derivada a partir dos estados que a especificação pode estar inicialmente. As partes restantes do caso de teste são recursivamente derivadas a partir dos estados alcancáveis na especificação, dado o estado inicial pela primeira transição do caso de teste. O algoritmo é não-determinístico no sentido que cada passo recursivo pode continuar de diferentes maneiras: o caso de teste pode terminar com o veredito *passar*; o caso de teste pode continuar com qualquer entrada permitida pela especificação e pode ser interrompido por uma saída chegando; ou o caso de teste pode esperar por uma saída e verificá-la, ou concluir que a implementação está em quietude. Infinitos casos de teste podem ser gerados a partir da especificação por meio desse algoritmo [Tretmans, 2008]. Porém, é necessário um grande número de iterações para gerar um caso de teste, e o algoritmo não garante que os casos de teste gerados cobrem determinado critério.

Apesar dos algoritmos existentes, a questão de seleção de testes é uma das importantes questões abertas no teste baseado em IOTSs. Uma vez que teste exaustivo para qualquer sistema realístico não é uma opção, uma importante questão é que casos de teste devem ser gerados automaticamente. Relacionada à questão de seleção de testes, está a questão de como a completude, cobertura ou qualidade de um conjunto de testes gerados automaticamente podem ser expressos, medidos e controlados [Tretmans, 2008].

Muitos trabalhos tem surgido com o intuito de resolver questões relacionadas à geração de testes em IOTSs [Huo and Petrenko, 2009; Weiglhofer and Wotawa, 2009; Simao and Petrenko, 2011; Hierons, 2012b,c]. Realizar um mapeamento sistemático é um passo importante para identificar os principais estudos nesse tópico bem como as lacunas existentes. Dentre os processos existentes para mapeamento sistemático, o processo sugerido por Kitchenham [2004] foi utilizado.

Este trabalho identificou os métodos para geração de casos de teste com IOTSs através de um mapeamento sistemático. Dentre os estudos identificados, 85 estudos foram selecionados, analisados quantitativamente em relação às características do modelo empregado no teste, formas de aplicação do teste, estudos empíricos, autoria e referências. A seguir é apresentado o planejamento realizado para o mapeamento sistemático.

1.1 Planejamento

O mapeamento sistemático é um tipo de revisão sistemática com o objetivo de fornecer uma visão geral de certa área por meio de um processo rigoroso de coleta e avaliação de evidências de um tópico específico [Petersen et al., 2008]. Como resultado, pode-se compreender as tendências de pesquisa bem como as lacunas da área. As evidências são coletadas de estudos primários, que são estudos individuais que apresentam resultados relevantes para determinada área de pesquisa [Kitchenham, 2004].

O planejamento do mapeamento sistemático foi realizado de acordo com o modelo de protocolo apresentado por Kitchenham [2004]. A definição do objetivo de pesquisa e das questões de pesquisa são dadas a seguir.

- **Objetivo:** Identificar os métodos existentes para a geração de casos de teste em IOTSs.

- **Questões de Pesquisa:**

  1. Quais são os métodos existentes para geração de casos de teste em IOTS?

  2. Podemos distribuir as publicações seguindo uma taxonomia? Se sim, como está essa distribuição?

  3. Que métodos de pesquisa estão sendo utilizados para validar as propostas?

  4. Quais são os principais grupos de pesquisa nesse tópico?

  5. Quais são os estudos mais referenciados?

A partir do objetivo e questões de pesquisa, foram definidas estratégias para busca e seleção dos estudos. Tais estratégias são descritas a seguir.

1.1.1 Estratégia de Busca para Seleção de Estudos Primários

A estratégia de busca e seleção dos estudos primários foi definida de acordo com as fontes de estudos, palavras-chaves e de acordo com a lista de controle:

- *Critérios para seleção das fontes:* Foram analisadas as fontes que indexam as principais conferências e periódicos onde os avanços referentes à área de Engenharia de Software são publicados. Também foram selecionados para a busca manual o *Brazilian Workshop on Systematic and Automated Software Testing* (SAST), por ser um evento nacional em que podem ser identificados estudos não indexados pelas fontes.

- *Métodos de Pesquisa:* Foi utilizada a busca automática, através da criação de uma *string* de busca que será executada nas fontes selecionadas, e também a busca manual nos últimos seis anos de SAST.

- *Palavras-chave:* test case generation, Input/Output Transition Systems.

- *String de busca:* (test case generation OR test generation) AND (iots OR input/output transition systems OR labeled transition systems OR labelled transition systems OR transition system OR input/output labeled transition systems OR input/output labelled transition systems)

- *Listagem das Fontes Selecionadas:* IEEE[1], Springer[2], Scopus[3], Web of Science-ISI[4], ACM[5], Elsevier-Science Direct[6], Compendex-Engineering Village[7] e Oxford Journals[8] (incluída por causa de um artigo da lista de controle).

A lista de controle foi definida juntamente com especialista, relatando alguns dos principais trabalhos já conhecidos que devem constar na lista de estudos selecionados (Tabela 1.1).

Tabela 1.1: Lista de Controle

Tretmans, J. **Model Based Testing with Labelled Transition Systems**. Formal Methods and Testing, 2008, 1-38 [Tretmans, 2008]
Hierons, R. M. **Implementation Relations for Testing through Asynchronous Channels**, The Computer Journal Advance Access, August 2012, 1-15. [Hierons, 2012c]
Hierons, R. M. **The complexity of asynchronous model based testing**. Theoretical Computer Science, v. 451, 2012,70-82. [Hierons, 2012b]
Simão, A. S.; Petrenko, A. **Generating asynchronous test cases from test purposes**. Information and Software Technology, v. 53, 2011, 1252-1262. [Simao and Petrenko, 2011]
Fraser, G.; Weiglhofer, M.; Wotawa, F. **Coverage Based Testing with Test Purposes**. Quality Software, 2008. QSIC '08. The Eighth International Conference on, 2008, 199 -208. [Fraser et al., 2008]

1.1.2 Critérios para Seleção dos Estudos

Os seguintes critérios de inclusão/exclusão de estudos foram definidos:

- *Critério 1:* Artigos que tratem de teste

- *Critério 2:* Artigos que tratem de IOTSs

- *Critério 3:* Artigos que tratem de métodos de geração de casos de teste em IOTSs

- *Critério 4:* Artigos completos em português ou inglês

- *Critério 5:* Artigos com texto completo disponível na *Web*

[1] http://ieeexplore.ieee.org
[2] http://www.springerlink.com/
[3] http://www.scopus.com
[4] http://www.webofknowledge.com/
[5] http://portal.acm.org/dl.cfm
[6] http://www.sciencedirect.com/
[7] http://www.engineeringvillage2.org
[8] http://www.oxfordjournals.org/

- *Critério 6:* Em caso de estudos repetidos e duplicados, deve-se selecionar o estudo mais recente e mais completo.

1.2 Processo de Seleção dos Estudos Primários

O processo para selecionar os estudos relevantes é dividido em duas etapas: seleção preliminar e seleção final. Para a seleção preliminar, os seguintes passos foram seguidos:

1. **Processo de seleção preliminar**

   - Identificação dos estudos relevantes;

   - Exclusão de estudos com base na leitura do título, resumo e palavras-chave;

   - Validação dos resultados com especialista.

Após a seleção preliminar, foram realizados os seguintes passos para a obtenção da lista final de artigos selecionados:

2. **Processo de seleção final**

   - Exclusão de estudos com base na leitura dinâmica do texto completo;

   - Validação do resultado do mapeamento com especialista;

   - Documentação e extração de dados dos estudos incluídos.

1.2.1 Estratégias de Extração e Sumarização dos Resultados

Como estratégia para extração de informação foi adotada a ferramenta Zotero[9] para extração dos dados das fontes e a ferramenta JabRef[10] foi utilizada para armazenamento e organização dos estudos e das informações extraídas. Para cada estudo selecionado, foram extraídos em uma planilha os dados apresentados na tabela 1.2.

Quanto à sumarização dos resultados, foram gerados gráficos para mostrar características representativas para a continuidade do estudo no tópico de geração de teste a partir de IOTSs. Além disso, os estudos foram agrupados conforme o tipo de definição do modelo estabelecido, as formas de avaliação das abordagens propostas e as ferramentas utilizadas, além de outras características que foram identificadas para classificação dos estudos.

1.3 Organização

Este relatório está organizado da seguinte forma. O Capítulo 2 apresenta como foi conduzido o mapeamento sistemático por meio de buscas nas fontes selecionadas e aplicação do

[9] http://www.zotero.org/
[10] http://jabref.sourceforge.net/

Tabela 1.2: Dados extraídos dos estudos selecionados

Extração de Dados
Identificador do artigo
Título
Autores
Ano de publicação
Tipo de publicação
Definição do modelo
Método de geração
Critério de teste
Suporte computacional
Validação

processo de seleção. No Capítulo 3 são apresentados os estudos selecionados após o processo de seleção e um breve resumo de cada um. O Capítulo 4 apresenta os dados extraídos dos estudos e discute os resultados encontrados, sumarizando algumas evidências. Finalmente, no Capítulo 5 são apresentadas as conclusões e futuras direções deste trabalho.

CAPÍTULO

2

Condução do Mapeamento Sistemático

O mapeamento sistemático foi conduzido por um período de 5 meses (Fevereiro/2013 a Junho/2013), de acordo com o planejamento apresentado no Capítulo 1. Nas próximas seções são detalhadas a estratégia adotada para construção das *strings* de busca e os resultados para cada uma das fontes selecionadas.

2.1 Seleção Preliminar

A seleção preliminar foi conduzida em 3 etapas:

- **Identificação dos estudos relevantes:** Foram aplicadas as *strings* de busca nas 8 fontes selecionadas. Utilizou-se a ferramenta Zotero[1] para extração dos dados encontrados nas fontes. Também foi realizada a busca manual nos Anais do SAST e nenhum trabalho foi selecionado. Foram eliminados os estudos repetidos com o uso da ferramenta JabRef.
- **Exclusão de estudos com base no título, resumo e palavras-chave:** A partir da lista de estudos obtida, foi realizada a exclusão de estudos com o uso a ferramenta JabRef para a organização dos estudos selecionados.
- **Validação com especialista:** Um especialista acompanhou os resultados da seleção preliminar.

Essas etapas são detalhadas nas próximas seções.

[1] http://www.zotero.org

2.1.1 Construção das *Strings* de Busca

Para a construção das *strings* de busca, foram utilizadas as palavras-chaves definidas no Capítulo 1, e considerados os conhecimentos obtidos na leitura dos artigos da lista de controle. Assim, a *string* definida para cada fonte foi:

IEEE:

(“test case generation” OR “test generation”) AND (“IOTS” OR “input/output transition systems” OR “labeled transition systems” OR “labelled transition systems” OR “input/output labeled transition systems” OR “input/output labelled transition systems”)

Springer:

(“test case generation” or “test generation”) and (“IOTS” or “input/output transition systems” or “labeled transition systems” or “labelled transition systems” or “input/output labeled transition systems” or “input/output labelled transition systems”)

Scopus:

({test case generation} OR {test generation}) AND ({IOTS} OR {input/output transition systems} OR {labeled transition systems} OR {labelled transition systems} OR {input/output labeled transition systems} OR {input/output labelled transition systems})

ACM:

(“test case generation” OR “test generation”) AND (“iots” OR “input/output transition systems” OR “labeled transition systems” OR “labelled transition systems” OR “input/output labeled transition systems” OR “input/output labelled transition systems”)

Elsevier-ScienceDirect:

({test case generation} or {test generation}) and (IOTS or {input/output transition systems} or {labeled transition systems} or {labelled transition systems} or {input/output labeled transition systems} or {input/output labelled transition systems})

Compendex-Engineering Village:

({test case generation} or {test generation}) and (IOTS or {input/output transition systems} or {labeled transition systems} or {labelled transition systems} or {input/output labeled transition systems} or {input/output labelled transition systems})

Web of Science:

((“test case generation” or “test generation”) and (IOTS or “input/output transition systems” or “labeled transition systems” or “labelled transition systems” or “input/output labeled transition systems” or “input/output labelled transition systems”))

Oxford Journals:

(“test case generation” or “test generation”) and (IOTS or “input/output transition systems” or “labeled transition systems” or “labelled transition systems” or “input/output labeled transition systems” or “input/output labelled transition systems”)

2.1.2 Resultados das Buscas

Para a busca automática, foi executada em cada base sua respectiva *string* de busca no dia 05/02/2013. Os resultados para cada base são apresentados na Tabela 2.1. Na busca manual nos Anais do SAST não foi recuperado nenhum estudo.

Tabela 2.1: Estudos retornados

Fontes	**Retornados**
IEEE	132
ACM	52
Scopus	179
Springer Link	237
ScienceDirect	86
Web of Science	25
Engineering Village	25
Oxford Journals	3
TOTAL	**741**

Do total de 741 estudos retornados, 192 eram duplicados. Foram excluídos 116 estudos da Scopus, 2 da IEEE, 25 da Engineering Village, 21 da Web of Science e 13 da ACM. A maioria dos estudos duplicados excluídos eram da Scopus, pois ela indexa eventos de outras bases. O total de estudos relevantes obtidos para a seleção preliminar foi reduzido para 549. A Tabela 2.2 exibe o número de estudos obtido em cada base.

2.1.3 Seleção Preliminar de Trabalhos

Inicialmente, a revisora leu os títulos, resumos e palavras-chaves dos 549 estudos e excluiu 294 deles, restando 255 para a próxima etapa. 6 estudos foram excluídos pelo critério 6 (estudos repetidos), 1 estudo foi excluído pelo critério 5 (o texto completo não estava disponível na Web), 186 foram excluídos pelo critério 2 (não tratavam de LTS/IOTS), e 99 foram excluídos pelo critério 1 (não tratavam de Teste de Software).

Vale ressaltar que muitos estudos foram incluídos para a seleção final por não conterem informações suficientes no resumo, ou seja, não foi possível identificar qual modelo era utilizado

Tabela 2.2: Estudos retornados após retirar os duplicados

Fontes	Retornados
IEEE	127
ACM	38
Scopus	62
Springer Link	237
ScienceDirect	77
Web of Science	4
Engineering Village	1
Oxford Journals	3
TOTAL	**549**

para a geração de teste. Além disso, muitos estudos excluídos tratavam apenas de Especificação Formal com o uso de IOTSs ou LTSs.

2.2 Seleção Final e Extração dos Resultados

Nessa fase, foi realizada uma leitura dinâmica do texto completo, de modo que foi possível estabelecer os estudos incluídos para a fase de análise. A lista final de estudos selecionados foi validada com um especialista.

Devido a indisponibilidade do texto completo na *Web* (critério de exclusão 5), 9 estudos foram eliminados, restando 246 estudos. Foram excluídos 171 estudos com base na leitura do texto completo, sendo que 9 deles foram excluídos pelo critério 6 (estudos repetidos), 9 foram excluídos pelo critério 4 (não estava em português ou inglês), 35 foram excluídos pelo critério 3 (não tratavam de métodos de geração de teste para IOTSs/LTSs), 111 foram excluídos pelo critério 2 (não tratavam de IOTS/LTS), e 3 foram excluídos pelo critério 1 (não tratava de teste de software). Portanto, foram selecionados 84 estudos para análise e extração de dados. O especialista recomendou a inclusão de mais um estudo [Rusu et al., 2005], totalizando 85 estudos para extração de dados e análise.

Dentre os estudos excluídos, é importante enfatizar que 56 estudos tratavam de Máquinas de Estado Finito (MEFs). Isso pode ser explicado pelo fato de que esse modelo possui uma teoria de teste bem formada e estabelecida, e atualmente alguns estudos relacionam os métodos de geração de teste baseado a partir de MEFs com os métodos de geração a partir de IOTSs [Hierons, 2012b,c].

2.2.1 Extração de dados

A partir dos 85 estudos selecionados, algumas informações relevantes foram extraídas. A Figura 2.1 apresenta o tipo de publicação dos estudos selecionados. Nota-se que a maioria foi publicado em periódico seguido de publicações em conferências. No entanto, os estudos publicados em coleções LNCS (*Lecture Notes in Computer Science*, da Springer) correspondem

a anais de conferências indexadas pela Springer. Portanto, cerca de 55% dos estudos dos estudos identificados foi publicado em conferência. A Tabela 2.3 apresenta os principais fóruns de publicação dos estudos e o número de estudos de cada um. É importante observar que as duas primeiras conferências da Tabela 2.3, nas quais há mais estudos selecionados, foram recentemente unificadas em uma única conferência (*International Conference on Testing Software and Systems*). O periódico *Electronic Notes in Theoretical Computer Science* se destaca por ser o que contém mais estudos dentre os selecionados. Apesar disso, observa-se que os estudos estão bem distribuídos entre os fóruns.

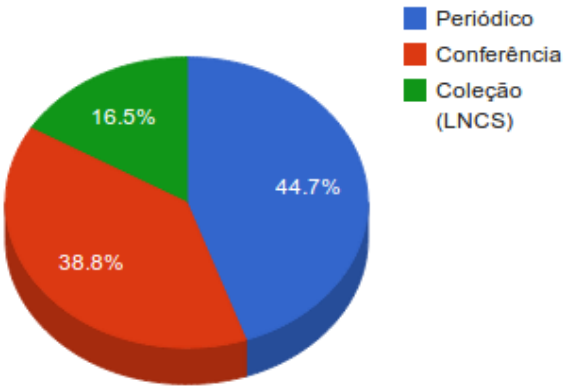

Figura 2.1: Tipo de Publicação dos estudos selecionados

Tabela 2.3: Principais Fóruns dos Estudos Selecionados

Fórum	**#**
Conferências	
International Conference on Formal Approaches to Software Testing	4
International Conference on Testing of Communicating Systems	4
International Conference on Quality Software	3
International Conference on Software Engineering and Formal Methods	3
International Conference on Software Engineering Research, Management Applications	2
International Symposium on Automated Technology for Verification and Analysis	2
IEEE International Symposium on Software Reliability Engineering	2
Periódicos	
Electronic Notes in Theoretical Computer Science	6
International Journal on Software Tools for Technology Transfer	3
Information and Software Technology	2
Theoretical Computer Science	2
Formal Aspects of Computing	2
Distributed Computing	2

A Figura 2.2 apresenta a distribuição dos estudos pelo modelo aplicado. A maioria dos estudos é sobre IOTSs (*Input/Output Transition Systems*), seguido do modelo IOLTS. Apesar

de ser mais genérico, o modelo LTS (*Labeled Transition Systems*) também foi citado. A seção 4.1 irá detalhar os modelos utilizados nos estudos.

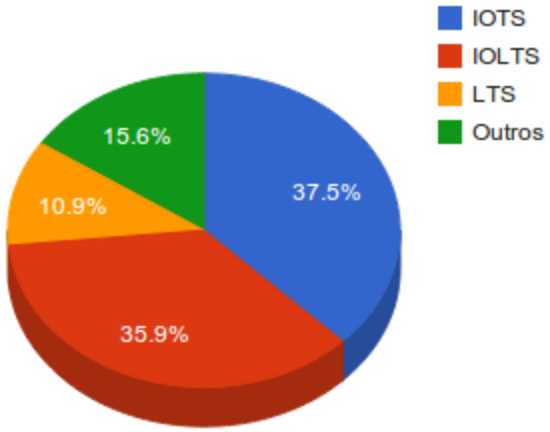

Figura 2.2: Modelos utilizados nos estudos selecionados

Na Figura 2.3, é exibida a distribuição dos estudos no decorrer dos anos. Existem poucos estudos no final da década de 1990, e houve um notável crescimento de publicações a partir do ano de 2004, sendo que o maior número de publicações concentra-se em 2012. Notou-se também uma diminuição no número de publicações de 2009 até 2011.

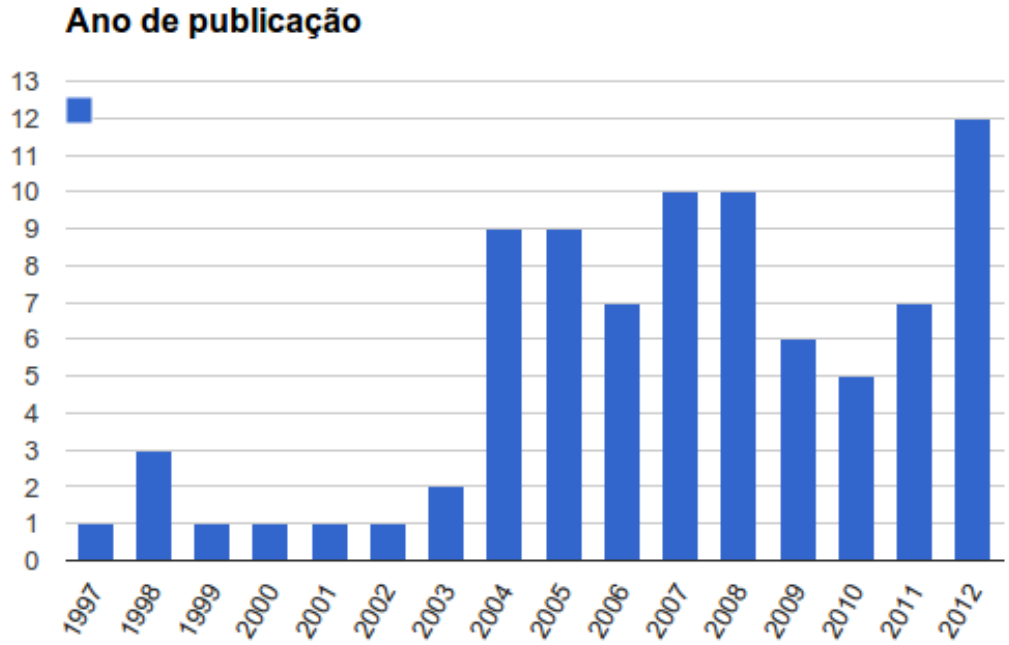

Figura 2.3: Ano de Publicação dos estudos selecionados

A Tabela 2.4 apresenta os estudos obtidos e os selecionados para cada fonte. Mais da metade dos estudos selecionados é da *Springer*. Isso ocorre porque essa fonte indexa muitos eventos e periódicos na área de Teste formal, de forma similar à área de Especificação Formal [Costa and Pereira, 2013], já que esses dois tópicos são relacionados.

Uma representação de resultados bastante comum em mapeamentos sistemáticos é um *bubble plot* [Petersen et al., 2008], no qual os eixos x e y representam categorias, os pontos do

Tabela 2.4: Estudos retornados e incluídos para cada fonte

Fontes	Retornados	Incluídos
IEEE	127	14
ACM	38	2
Scopus	62	10
Springer Link	237	46
ScienceDirect	77	11
Web of Science	4	1
Engineering Village	1	0
Oxford Journals	3	1

gráfico representam intersecções entre as categorias e são indicados por bolhas, com raio proporcional à quantidade de estudos que está classificando sob aquelas duas categorias específicas. A Figura 2.4 apresenta o mapa sistemático obtido para os estudos selecionados trabalho. O eixo *x* apresenta os modelos utilizados e o eixo *y* apresenta a forma de aplicação do teste. A maioria dos trabalhos utiliza os modelos IOLTS e IOTS. Considerando o modelo IOLTS, a maioria dos trabalhos preocupa-se com detalhes específicos do tipo de sistema na qual o teste será aplicado, como sistemas reativos e sistemas concorrentes. Já o modelo IOTS tem os trabalhos concentrados no teste distribuído e teste assíncrono, sendo estes os estudos mais recentes dessa área.

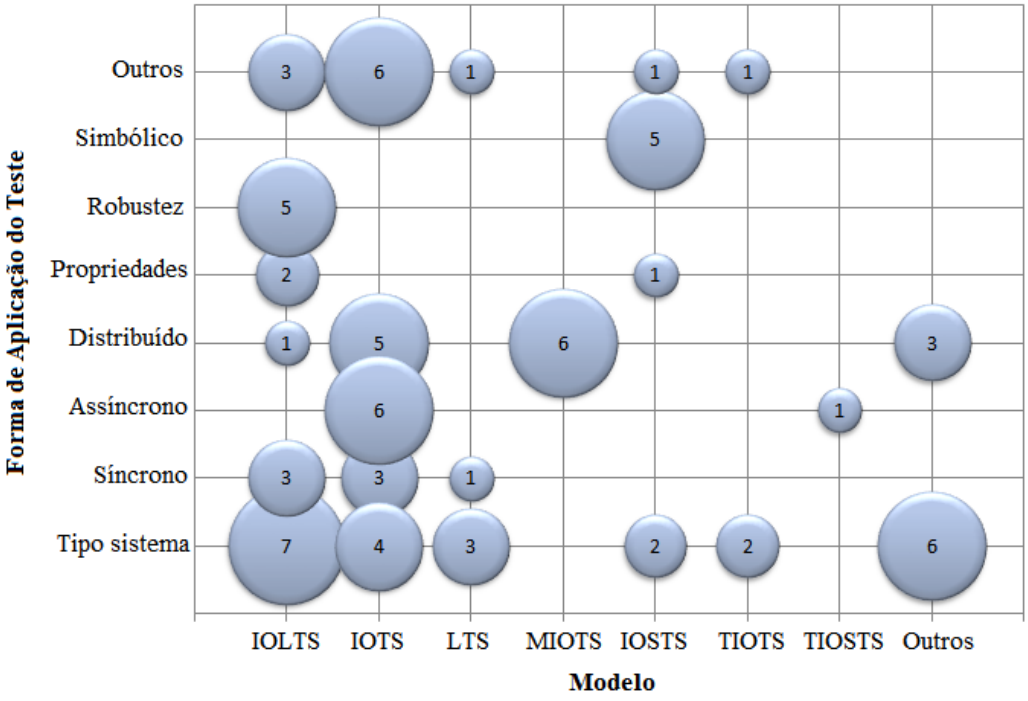

Figura 2.4: Mapa dos estudos selecionados

2.3 Ameaças à validade

Foram identificadas algumas ameaças à validade deste trabalho de mapeamento sistemático. Um estudo não foi retornado pela busca, mas foi encontrado pelo especialista, mesmo o trabalho

sendo indexado por uma das fontes utilizadas. Não foi possível identificar outros sinônimos que melhorassem a *string* de busca. Alguns estudos foram excluídos por não possuir o texto completo disponível na Web. É possível que algum estudo relevante tenha sido excluído. Porém, tais trabalhos podem ter sido publicados com uma versão estendida ou com mais resultados consolidados.

Além disso, a busca automática foi realizada em apenas 8 engenhos de busca (quantidade aceitável por [Kitchenham, 2004]). Apesar de ter abrangido engenhos e indexadores relevantes, é possível que nem todas as evidências tenham sido alcançadas. Os critérios de inclusão e exclusão foram aplicados apenas por um revisor, o que é previsto por [Kitchenham, 2004] para alunos de PhD, desde que o orientador participe da revisão. Outra limitação diz respeito ao idioma dos estudos, já que alguns estudos foram eliminados por não estarem em inglês. Apesar das ameaças identificadas, futuras replicações deste mapeamento podem amenizá-las.

2.4 Considerações Finais

A partir do planejamento apresentado no Capítulo 1, o mapeamento sistemático foi conduzido nos engenhos de busca selecionados. Após os procedimentos de seleção, 85 estudos foram selecionados para a etapa de extração e sumarização de dados. A maioria dos trabalhos selecionados são estudos recentes, mostrando que há interesse em pesquisa nessa área. Apesar das limitações desse estudo, o capítulo a seguir apresenta uma visão geral dos estudos selecionados.

CAPÍTULO

3

Métodos para Geração de Teste a partir de IOTSs

Esta seção apresenta uma breve descrição dos 85 estudos selecionados após a condução do mapeamento sistemático. A Tabela 3.1 apresenta a identificação, título, autores e ano de publicação para cada estudo. A seção 3.1 exibe uma breve descrição de cada estudo, que foi identificado pelo ID.

Tabela 3.1: Informações sobre os 85 estudos selecionados

ID	Título	Autores	Ano
1	Protocol Conformance Testing a SIP Registrar: an Industrial Application of Formal Methods [Aichernig et al., 2007]	Aichernig, B.; Peischl, B.; Weiglhofer, M.; Wotawa, F.	2007
2	Improving Fault-based Conformance Testing [Aichernig et al., 2008]	Aichernig, B. K.; Weiglhofer, M.; Wotawa, F.	2008
3	A Model-Based Approach for Robustness Testing [Fernandez et al., 2005]	Fernandez, J.-C.; Mounier, L.; Pachon, C.	2005
4	Testing Transition Systems with Input and Output Testers [Petrenko et al., 2003]	Petrenko, A.; Yevtushenko, N.; Huo, J.	2003
5	From Faults Via Test Purposes to Test Cases: On the Fault-Based Testing of Concurrent Systems [Aichernig and Delgado, 2006]	Aichernig, B.; Delgado, C.	2006
6	Testing and Model-Checking Techniques for Diagnosis [Gromov and Willemse, 2007]	Gromov, M.; Willemse, T.	2007

7	Test Generation Derived from Model-Checking [Jéron and Morel, 1999]	Jéron, T.; Morel, P.	1999
8	A formal abstract framework for modelling and testing complex software systems [Aiguier et al., 2012]	Aiguier, M.; Boulanger, F.; Kanso, B.	2012
9	Off-line Test Case Generation for Timed Symbolic Model-based Conformance Testing [Bannour et al., 2012]	Bannour, B.; Escobedo, J.; Gaston, C.; Le Gall, P.	2012
10	A concurrent TTCN based approach to conformance testing of distributed routing protocol OSPF v2 [Bi et al., 1998]	Bi, J.; Wu, J.; Chen, X.	1998
11	Symbolic Execution Techniques for Test Purpose Definition [Gaston et al., 2006]	Gaston, C.; Le Gall, P.; Rapin, N.; Touil, A.	2006
12	Testing interruptions in reactive systems [Andrade and Machado, 2012]	Andrade, W.; Machado, P.	2012
13	Test generation for interworking systems [Koné and Castanet, 2000]	Koné, O.; Castanet, R.	2000
14	A Fresh Look at Testing for Asynchronous Communication [Bhateja et al., 2006]	Bhateja, P.; Gastin, P.; Mukund, M.	2006
15	Action Refinement in Conformance Testing [van der Bijl et al., 2005]	van der Bijl, M.; Rensink, A.; Tretmans, J.	2005
16	Compositional Testing with ioco [van der Bijl et al., 2004]	van der Bijl, M.; Rensink, A.; Tretmans, J.	2004
17	Formal Conformance Testing of Systems with Refused Inputs and Forbbiden Actions [Bourdonov et al., 2006]	Bourdonov, I. B.; Kossatchev, A. S.; Kuliamin, V. V.	2006
18	A Semantic Framework for Test Coverage [Briones et al., 2006]	Briones, L.; Brinksma, E.; Stoelinga, M.	2006
19	Automatic test cases generation for statechart specifications from semantics to algorithm [Chen, 2011]	Chen, L.	2011
20	Model-Based Testing of Environmental Conformance of Components [Frantzen and Tretmans, 2007]	Frantzen, L.; Tretmans, J.	2007
21	Software Testing Based on Formal Specification [Gaudel, 2010]	Gaudel, M.-C.	2010
22	Formal test-case generation for UML statecharts [Gnesi et al., 2004]	Gnesi, S.; Latella, D.; Massink, M.	2004

23	Implementation Relations for Testing Through Asynchronous Channels [Hierons, 2012c]	Hierons, R. M.	2012
24	TGV: theory, principles and algorithms [Jard and Jéron, 2005]	Jard, C.; Jéron, T.	2005
25	Testing processes from formal specifications with inputs, outputs and data types [Lestiennes and Gaudel, 2002]	Lestiennes, G.; Gaudel, M.-C.	2002
26	Compositional Verification of Input-Output Conformance via CSP Refinement Checking [Sampaio et al., 2009]	Sampaio, A.; Nogueira, S.; Mota, A.	2009
27	Model-Based Testing and Some Steps towards Test-Based Modelling [Tretmans, 2011]	Tretmans, J.	2011
28	Model based testing with labelled transition systems [Tretmans, 2008]	Tretmans, J.	2008
29	Unifying Input Output Conformance [Weiglhofer and Aichernig, 2010]	Weiglhofer, M.; Aichernig, B.	2010
30	A Test Generation Framework for quiescent Real-Time Systems [Briones and Brinksma, 2005]	Briones, L.; Brinksma, E.	2005
31	Testing Real-Time Systems Using UPPAAL [Hessel et al., 2008]	Hessel, A.; Larsen, K.; Mikucionis, M.; Nielsen, B.; Pettersson, P.; Skou, A.	2008
32	On Conformance Testing for Timed Systems [Schmaltz and Tretmans, 2008]	Schmaltz, J.; Tretmans, J.	2008
33	More testable properties [Falcone et al., 2012]	Falcone, Y.; Fernandez, J.-C.; Jéron, T.; Marchand, H.; Mounier, L.	2012
34	Property Oriented Test Case Generation [Fernandez et al., 2004]	Fernandez, J.-C.; Mounier, L.; Pachon, C.	2004
35	Towards Property Oriented Testing [Machado et al., 2007]	Machado, P. D.; Silva, D. A.; Mota, A. C.	2007
36	Automated Test and Oracle Generation for Smart-Card Applications [Clarke et al., 2001]	Clarke, D.; Jéron, T.; Rusu, V.; Zinovieva, E.	2001
37	Symbolic Model Based Testing for Component Oriented Systems [Faivre et al., 2007]	Faivre, A.; Gaston, C.; Le Gall, P.	2007
38	Test Generation Based on Symbolic Specifications [Frantzen et al., 2005]	Frantzen, L.; Tretmans, J.; Willemse, T.	2005
39	A Symbolic Framework for Model-Based Testing [Frantzen et al., 2006]	Frantzen, L.; Tretmans, J.; Willemse, T.	2006

40	Model-based Test Selection for Infinite-State Reactive Systems [Jeannet et al., 2007]	Jeannet, B.; Jéron, T.; Rusu, V.	2007
41	Symbolic execution techniques for Refinement Testing [Le Gall et al., 2007]	Le Gall, P.; Rapin, N.; Touil, A.	2007
42	Automatic Verification and Conformance Testing for Validating Safety Properties of Reactive Systems [Rusu et al., 2005]	Rusu, V.; Marchand, H.; Jéron, T.	2005
43	A robustness testing method for network security [Fu and Kone, 2012]	Fu, Y.; Kone, O.	2012
44	A Formal Approach to Test the Robustness of Embedded Systems using Behaviour Analysis [Rollet and Saad-Khorchef, 2007]	Rollet, A.; Saad-Khorchef, F.	2007
45	Testing robustness of communicating systems using ioco-based approach [Rollet and Salva, 2009]	Rollet, A.; Salva, S.	2009
46	Two complementary approaches to test robustness of reactive systems [Rollet and Salva, 2008]	Rollet, A.; Salva, S.	2008
47	Factorized test-generation for multi-input/output transition systems [Brinksma et al., 1998]	Brinksma, E.; Heerink, L.; Tretmans, J.	1998
48	Conformance Relations for Distributed Testing Based on CSP [Cavalcanti et al., 2011]	Cavalcanti, A.; Gaudel, M.-C.; Hierons, R.	2011
49	Overcoming controllability problems in distributed testing from an input output transition system [Hierons, 2012a]	Hierons, R.	2012
50	Testing in the Distributed Test Architecture: An Extended Abstract [Hierons, 2008]	Hierons, R. M.	2008
51	Controllable Test Cases for the Distributed Test Architecture [Hierons et al., 2008]	Hierons, R.; Merayo, M.; Núñez, M.	2008
52	Using Time to Add Order to Distributed Testing [Hierons et al., 2012b]	Hierons, R.; Merayo, M.; Núñez, M.	2012
53	Implementation relations and test generation for systems with distributed interfaces [Hierons et al., 2012a]	Hierons, R.; Merayo, M.; Núñez, M.	2012
54	Scenarios-Based testing of systems with distributed ports [Hierons et al., 2011]	Hierons, R.; Merayo, M.; Núñez, M.	2011
55	Using schedulers to test probabilistic distributed systems [Hierons and Núñez, 2012]	Hierons, R.; Núñez, M.	2012
56	Testing probabilistic distributed systems [Hierons and Núñez, 2010]	Hierons, R.; Núñez, M.	2010

57	Testing Multi Input/Output Transition System with All-Observer [Li et al., 2004a]	Li, Z.; Wu, J.; Yin, X.	2004
58	Refusal testing for MIOTS with nonlockable output channel [Li et al., 2003]	Li, Z.; Wu, J.; Yin, X.	2003
59	Distributed testing of multi input/output transition system [Li et al., 2004b]	Li, Z.; Yin, X.; Wu, J.	2004
60	The complexity of asynchronous model based testing [Hierons, 2012b]	Hierons, R. M.	2012
61	Covering transitions of concurrent systems through queues [Huo and Petrenko, 2005]	Huo, J.; Petrenko, A.	2005
62	On Testing Partially Specified IOTS through Lossless Queues [Huo and Petrenko, 2004]	Huo, J. L.; Petrenko, A.	2004
63	Synchronizing Asynchronous Conformance Testing [Noroozi et al., 2011]	Noroozi, N.; Khosravi, R.; Mousavi, M.; Willemse, T.	2011
64	Generating asynchronous test cases from test purposes [Simao and Petrenko, 2011]	Simao, A.; Petrenko, A.	2011
65	Asynchronous Input-Output Conformance Testing [Weiglhofer and Wotawa, 2009]	Weiglhofer, M.; Wotawa, F.	2009
66	Automatic Conformance Testing of Internet Applications [van Beek and Mauw, 2004]	van Beek, H.; Mauw, S.	2004
67	State identification problems for input/output transition systems [Bensalem et al., 2008]	Bensalem, S.; Krichen, M.; Tripakis, S.	2008
68	Test case generation using PDA [Bhateja, 2011]	Bhateja, P.	2011
69	Grammar based asynchronous testing [Bhateja, 2009]	Bhateja, P.	2009
70	Test suite consistency verification [Boroday et al., 2008]	Boroday, S.; Petrenko, A.; Ulrich, A.	2008
71	Automated Conformance Verification of Hybrid Systems [Brandl et al., 2010]	Brandl, H.; Weiglhofer, M.; Aichernig, B.	2010
72	Automatic Model-Based Generation of Parameterized Test Cases Using Data Abstraction [Calamé et al., 2007]	Calamé, J. R.; Ioustinova, N. van der Pol, J.	2007
73	Test Generation from Recursive Tiles Systems [Chédor et al., 2012]	Chédor, S.; Jéron, T.; Morvan, C.	2012
74	Formalizing interoperability for test case generation purpose [Desmoulin and Viho, 2009]	Desmoulin, A.; Viho, C.	2009
75	Coverage based testing with test purposes [Fraser et al., 2008]	Fraser, G.; Weiglhofer, M.; Wotawa, F.	2008

76	Toward formal TTCN-based test execution [Hao and Wu, 1997]	Hao, R.; Wu, J.	1997
77	Heuristics for Faster Error Detection With Automated Black Box Testing [Kervinen and Virolainen, 2005]	Kervinen, A.; Virolainen, P.	2005
78	Using software architecture for code testing [Muccini et al., 2004]	Muccini, H.; Bertolino, A.; Inverardi, P.	2004
79	Optimal strategies for testing nondeterministic systems [Nachmanson et al., 2004]	Nachmanson, L.; Veanes, M.; Schulte, W.; Tillmann, N.; Grieskamp, W.	2004
80	Specification and Testing of E-Commerce Agents Described by Using UIOLTSs [Pardo et al., 2010]	Pardo, J.; Núñez, M.; Ruiz, M.	2010
81	A formal approach to protocol interoperability testing [Ruibing and Jianping, 1998]	Ruibing, H.; Jianping, W.	1998
82	Architectural Unit Testing [Scollo and Zecchini, 2005]	Scollo, G.; Zecchini, S.	2005
83	Using coverage to automate and improve test purpose based testing [Weiglhofer et al., 2009]	Weiglhofer, M.; Fraser, G.; Wotawa, F.	2009
84	Heuristics for ioco-Based Test-Based Modelling [Willemse, 2007]	Willemse, T.	2007
85	Testing Systems of Concurrent Black-Boxes—An Automata-Theoretic and Decompositional Approach [Xie and Dang, 2006]	Xie, G.; Dang, Z.	2006

3.1 Descrição dos Estudos

Nesta seção, são descritos os estudos identificados, agrupados em função do foco principal desse trabalho.

3.1.1 Métodos que empregam modelo de falhas

Estudo 1: Esse trabalho [Aichernig et al., 2007] investiga o teste de conformidade no registro de um protocolo iniciação de sessão (SIP Registrar) para sistemas reativos. Foi utilizado o modelo IOLTS com as ferramentas TGV e CADP *toolbox*. Foram discutidas questões críticas em relação à abstração e, ao contrário da maioria dos estudos de caso realizados, são fornecidas justificativas para essas abstrações. Propósitos de teste foram projetados com a finalidade de cobertura estrutural e aplicou-se uma abordagem baseada em falhas, sendo registradas questões de escalabilidade. Os casos de testes gerados detectaram falhas severas nas implementações. Utilizar duas estratégias diferentes para projetar propósitos de teste obteve bom resultado nos

estudos empíricos apresentados, uma vez que métodos baseados em falhas revelam erros adicionais que não são revelados com propósitos de teste projetados estruturalmente.

Estudo 2: Esse estudo [Aichernig et al., 2008] trata da geração de propósitos de teste a partir de defeitos, visando dois problemas: casos de teste para falhas que não podem ser detectadas usando a relação *ioco*, e a construção de espaço de estados para a especificação original e para a especificação mutada. A técnica proposta para geração de teste baseada em *ioco* visando falhas mostrou-se um avanço por gerar casos de teste relevantes para *ioco* e reduzir o número de casos de teste. Desse modo, a técnica proposta apresenta-se mais razoável para teste em escala industrial, como mostrado em um estudo de caso industrial.

Estudo 3: Esse estudo [Fernandez et al., 2005] estende o teste de conformidade para o teste de robustez utilizando o formalismo IOLTS. O modelo de falhas é expresso por mutações sintáticas realizadas na especificação. Os requisitos de robustez são expressos por uma fórmula de lógica temporal linear descrevendo o comportamento esperado do sistema em termos de interações com o testador. A abordagem consiste em verificar a especificação diante dos requisitos de robustez para produzir sequências de diagnóstico. Tais sequências são retornadas como casos de teste abstratos para serem executados na implementação. Os requisitos de robustez são propriedades limitadas expressas por um autômato parametrizado em infinitas palavras. Sequências correspondentes são instanciadas no tempo do testes para manter a execução do teste finita. A proposta foi implementada em uma ferramenta e experimentada em pequenos programas Java. Essa abordagem é muito melhor quando destina-se a casos de teste com respeito a requisitos de robustez esperados.

Estudo 4: Nesse estudo [Petrenko et al., 2003] é apresentado um framework para teste de filas com quietude. O testador é decomposto em dois processos: aplicar entradas para a implementação através de uma fila, e ler as saídas que a implementação coloca na fila finita até que nenhuma saída seja detectada, ou seja, o testador detecta a quietude. Dois tipos de testadores de fila, modelados como IOTSs, são propostos: testador de filas com quietude e testador de filas suspensas. O procedimento de derivação de teste proposto foi elaborado pensando em um modelo de falhas. Novas relações de implementação foram definidas para verificar tanto os tipos de testadores quanto os procedimentos de derivação de teste. Diferente da abordagem de Tretmans [2008], o conjunto de teste resultante é finito e relacionado aos pressupostos sobre as potenciais falhas. Por ser um conjunto finito, é possível verificar relações de equivalência nos casos de teste e não apenas relações de pré-ordem.

3.1.2 Métodos que aplicam verificação de modelos

Estudo 5: Esse estudo [Aichernig and Delgado, 2006] apresenta um método que utiliza teste de mutação e verificação de modelos para gerar propósitos de teste para sistemas concorrentes. Os problemas discutidos são: a falta de estratégias de seleção de teste; identificação de propósitos de teste; mutantes equivalentes; e automação da geração de testes. Esse método de teste de mutação para geração de propósitos de teste é similar à teoria para cálculo de refinamento e OCL apresentada em trabalhos anteriores. Assim, essa teoria de IOLTS baseada

em falhas é uma instanciação de propriedades de refinamento, e sua generalidade é mostrada através de um estudo de caso.

Estudo 6: Esse estudo [Gromov and Willemse, 2007] apresenta um método para conduzir ao diagnóstico de defeitos baseado em modelos de defeitos. Inspirando-se em abordagens clássicas de diagnóstico para Máquinas de estado finito (MEFs), foram apresentados conceitos e técnicas para fazer o processo de seleção do modelo de defeitos mais efetivo em modelos LTS. Foi adotada a noção de distinguibilidade de MEFs para LTS, e também a noção de ortogonalidade que auxilia a direcionar os esforços em aspectos isolados do modelo de defeitos. Além disso, os conceitos de diagnóstico definidos foram integrados a problemas de verificação de modelos. Um algoritmo e um procedimento de semi-decisão que implementam essas técnicas e conceitos foram propostos. Um exemplo ilustra a técnica apresentada.

Estudo 7: Esse estudo [Jéron and Morel, 1999] aplica verificadores de modelos na geração de casos de teste. Foi proposto um algoritmo para geração de casos de teste em tempo de execução baseado no clássico algoritmo de Tarjan para grafos, também utilizado em verificadores de modelos em tempo de execução. Foi empregado o modelo IOLTS. Foi apresentado um framework para derivação de outros algoritmos, na qual três instanciações de tal framework foram apresentadas para geração de testes. Tais algoritmos foram integrados na ferramenta TGV, e comparações com outros trabalhos foram realizadas. Os casos de teste produzidos pela abordagem proposta tem alta qualidade e são muito similares àqueles escritos à mão, com escolhas e *loops*.

3.1.3 Métodos baseados em regras de inferência

Estudo 8: A definição de componentes de Barbosa [2003] permite unificar em um mesmo framework uma ampla família de formalismos baseados em estados, como Mealy Automata e LTS. Outra definição, de Rutten [Hansen et al., 2006], define o comportamento de componentes com o uso de funções de transferência causal. Esse estudo [Aiguier et al., 2012] mostra como definir funções causais que fundamentam qualquer sistema, define também dois operadores básicos de integração de componentes para formar amplos sistemas. Foi proposto nesse trabalho uma teoria de teste de conformidade para componentes, definindo propósitos de teste e um algoritmo que gera casos de teste a partir dos propósitos de teste. O algoritmo é dado por uma série de regras de inferência, e cada regra manipula uma observação ou um estímulo enviado ao sistema. Por último, foi mostrado que sob certas condições, a relação de conformidade é preservada ao longo de cada operador de integração.

Estudo 9: Esse estudo [Bannour et al., 2012] propõe uma abordagem de teste *off-line* em um framework TBM com características temporais, utilizando o modelo Timed Input/Output Symbolic Transition Systems. A abordagem possui os seguintes passos: seleção baseada na cobertura considerando os atrasos; execução da sequência gerada no sistema, gerando as saídas com atraso; e computação do veredito final do teste. O algoritmo para geração *off-line* das sequências de teste utilizou regras de inferência, como apresentado em trabalhos anteriores.

A abordagem foi implementada na ferramenta Diversity, e um exemplo de aplicação mostrou vantagens, como a emissão de vereditos melhores, já que todo o *trace* pode ser analisado.

Estudo 10: O protocolo OSPF (*Open Shortest Path First*) foi um protocolo amplamente utilizado na *Web*. Esse estudo [Bi et al., 1998] propõe uma técnica de teste de conformidade para o protocolo OSPF. Tal técnica é baseada na notação Concurrent TTCN (C-TTCN), utilizada para descrever conjuntos de testes abstratos. Foi apresentada a arquitetura abstrata para teste baseado em C-TTCN utilizando o modelo IOLTS que tem a mesma semântica operacional. Dois passos foram definidos para a geração de sequências de teste: identificação de todos os caminhos de interação e geração dos sub-caminhos verificando suas dependências. Para a geração do conjunto de teste com C-TTCN, é gerada uma árvore de teste não-determinística a partir de regras de inferência, sendo depois mapeados para um conjunto de teste completo. O método foi implementado na ferramenta TUGEN, que automatiza o processo de geração.

Estudo 11: Esse estudo [Gaston et al., 2006] apresenta uma contribuição ao teste de conformidade simbólico. Propósitos de teste são definidos como alguma sub-árvores particular contida em uma árvore de execução simbólica. Foram introduzidos dois novos critérios para computar propósitos de teste automaticamente: todos os comportamentos simbólicos de tamanho n e restrição por inclusão (a sub-árvore extraída satisfaz um critério de cobertura que é baseado em um procedimento de detecção redundante). Um algoritmo para geração de casos de teste foi proposto como um conjunto de regras de inferência. Cada regra manipula uma observação do sistema ou um estímulo enviado pelo caso de teste.

3.1.4 Métodos baseados na busca em profundidade

Estudo 12: Esse estudo [Andrade and Machado, 2012] apresenta uma estratégia para o teste de conformidade de sistemas reativos com interrupções, cobrindo a modelagem, geração e seleção de casos de teste sólidos. O modelo adotado é o *Annotated LTS* (ALTS), que possui descrições especiais, tornando o processo de geração viável. Esse modelo é implementado na ferramenta LTS-BT, que traduz especificações em alto nível para o modelo ALTS. Foi introduzido um algoritmo para geração de casos de teste com interrupções, utilizando a estratégia de busca em profundidade e gravando o código de interrupção. São apresentadas propriedades dos casos de teste com interrupção geradas por esse algoritmo. Um caso de estudo apresenta a avaliação da abordagem proposta, mostrando que ela contribui para um processo de teste mais efetivo e produtivo.

Estudo 13: Esse estudo [Koné and Castanet, 2000] introduz uma técnica para seleção de testes em tempo de execução, evitando a explosão combinatorial de estados. Foi apresentado um método que computa automaticamente padrões de teste sem a construção de todo o grafo de comportamento. Os testes são computados diretamente a partir da especificação dos subsistemas. Essa técnica tem sido experimentada com sucesso em outros trabalhos, mas para o teste de sistemas interoperativos é uma novidade. O formalismo utilizado é IOSM, que é um tipo de LTS. Um algoritmo para geração de teste foi apresentado, utilizando a busca em profundidade. Um estudo de caso apresenta o uso da técnica com sucesso.

3.1.5 Métodos baseados na teoria *ioco*

Estudo 14: Nesse estudo [Bhateja et al., 2006] foi considerada a noção natural de observabilidade para teste de sistemas baseados em pares de entrada-saída. Utilizando essa noção, foram propostas duas noções de teste de equivalência: uma corresponde a apresentar todas as entradas do teste antes, e a outra corresponde a alimentar entradas interativamente. A primeira é estritamente fraca comparando-se com a de Tretmans [2008], enquanto a segunda noção é incomparável. Foram estabelecidos resultados de decidibilidade, na qual a equivalência fraca é indecidível para sistemas de estado finito, do mesmo modo que a equivalência de Tretmans [2008]. Entretanto, a equivalência forte é decidível para sistemas de transição bem estruturado. Também foi mostrado que a noção fraca de equivalência é decidível se for gravado o comportamento de entrada-saída de um sistema como um gráfico de sequência de mensagens não rotulado.

Estudo 15: Esse trabalho [van der Bijl et al., 2005] investiga o uso de refinamento de ações para obter casos de teste automaticamente no nível de detalhe requerido, adicionando informações ao relacionar uma ação do modelo com um comportamento mais detalhado. Foi mostrado o refinamento de ações com o uso do caso não-trivial: refinamento de entrada-entradas atômico linear. Essa abordagem de refinamento foi estendida para tipos de refinamentos mais genéricos. Em relação à corretude, foi introduzida a relação de implementação *uioco*. Foi mostrado sob quais condições o refinamento de um conjunto abstrato de testes completos resulta em um conjunto de testes completamente refinado. Uma consequência é que a equivalência da especificação não é preservada pelo refinamento de ações.

Estudo 16: Esse estudo [van der Bijl et al., 2004] aborda as propriedades de composicionalidade da relação *ioco* para operações em LTS: composição paralela e ocultamento. Se *ioco* tem as propriedades de composicionalidade para essas operações, então a corretude das partes implica na corretude do todo (sistema integrado), ou que uma falha do sistema todo implica em uma falha em um componente (chamado de pré-congruência). Foi mostrado que *ioco* possui essa propriedade para a composição paralela e ocultamento na ausência de especificações parciais de ações de entrada. Porém, como completar a especificação é mais caro, foi mostrado que pode-se utilizar especificações incompletas e ainda satisfazer essa propriedade. Isso conduziu à uma nova relação de implementação que é um pouco mais fraca do que *ioco*. Os resultados mostrados implicam que *ioco* pode ser usada para teste composicional se as especificações forem modeladas como IOTSs.

Estudo 17: Considerando o teste assíncrono com IOLTS, onde a relação *ioco* não é válida, entradas não especificadas devem ser melhor compreendidas. Esse estudo [Bourdonov et al., 2006] define uma extensão da relação *ioco* que pode ter ações recusadas e proibidas. Também são mostradas duas operações para completar a especificação, que transformam qualquer LTS em um LTS habilitado para entrada e possuindo o mesmo conjunto de implementações em *ioco*-conformidade. Isso torna possível definir propriedades de composição de LTS preservando a *ioco*-conformidade.

Estudo 18: Esse estudo [Briones et al., 2006] introduz uma abordagem semântica para cobertura de testes que visa superar algumas deficiências, como considerar a gravidade dos defeitos. O ponto de partida são modelos de falhas com peso (*Weighted Fault Model*) que dão um peso para cada erro potencial em uma implementação, e é baseado na teoria *ioco*. Foi introduzido também o Autômato de falhas, que é um LTS ampliado com uma função de peso do estado. Esse autômato representa sintaticamente os modelos de falhas com peso. Algoritmos para computar e otimizar a cobertura de testes foram propostos. Os experimentos apresentados com protocolos de *chat* mostraram que a abordagem é viável para pequenos protocolos.

Estudo 19: Esse estudo [Chen, 2011] estabelece a base teórica para a geração automatizada de casos de teste para especificações com Statecharts, com base no trabalho de Tretmans [2008] para teste de conformidade com LTS. Foi definido um modelo semântico adequado, que pode apenas especificar testadores com comportamento observável e esconder o comportamento não-observável, propondo uma definição de conjunto de teste razoável e um algoritmo de geração de teste prático. Para essa geração, foi proposta uma relação de conformidade baseada na semântica observável e hipóteses de teste.

Estudo 20: Esse estudo [Frantzen and Tretmans, 2007] apresenta uma abordagem para teste baseado em modelos de componentes. Nessa abordagem, assume-se que uma especificação do serviço fornecido está disponível tanto para o componente a ser testado quando para os componentes invocados. É discutido como a teoria *ioco* para IOTSs é aplicável para teste de componentes e onde não é. Uma nova relação de implementação foi introduzida, chamada conformidade de ambiente (*eco*). Essa relação expressa que um componente invoca corretamente outro componente de acordo com a especificação do serviço fornecida no primeiro componente. Um algoritmo para geração de teste sólido e exaustivo para a relação *eco* é apresentado.

Estudo 21: Esse estudo [Gaudel, 2010] apresenta um framework genérico para desenvolvimento de métodos de teste baseados em especificações formais, e sua especialização para vários formalismos: Máquinas de estado finito, Sistemas de transição rotulados e Sistemas de transição com prioridades. Essa abordagem assume algumas hipóteses de testabilidade na implementação. Uma noção de conjunto de teste exaustivo é derivada a partir da semântica da notação formal e a partir da definição de implementação correta. A partir disso, um conjunto de teste finito pode ser selecionado a partir da exaustividade via algumas hipóteses de teste.

Estudo 22: Esse estudo [Gnesi et al., 2004] propôs uma relação de teste de conformidade formal para Statecharts e um algoritmo para geração de casos de teste que é completo (consistente e exaustivo). Foi apresentada a base teórica, utilizando IOLTS habilitados para entrada como modelo semântico para um subconjunto de comportamentos de Statecharts. Nessa abordagem, obtém-se o respectivo IOLTS de um Statechart. A relação de conformidade utilizada no algoritmo de geração é semelhante à de Tretmans [2008], com adaptações que tratam do framework semântico para Statecharts. Também foi proposta uma linguagem específica para casos de teste. Um exemplo com a utilização da proposta ilustra sua aplicabilidade.

Estudo 23: Esse trabalho [Hierons, 2012c] trata da comunicação assíncrona entre o sistema e seu ambiente por um canal FIFO. Essa situação é comum em muitos sistemas que in-

teragem por redes, onde pode haver atraso na comunicação. São definidas novas relações de implementação e equivalências que não requerem a observação da quietude para o contexto de canais de comunicação assíncronos. As novas relações de implementação e equivalências são baseadas em semi-comutação. Assim, essas relações foram definidas em termos de IOTSs sem compô-las com filas, e assumiu-se que sequências de entradas e saídas são observadas. Como resultado, foi demostrado que utilizar relação de implementação para a comunicação assíncrona é indecidível. Isso significa que não podemos esperar algoritmos de geração de teste baseado em domínio de defeitos para teste assíncrono. Logo, quando há comunicação assíncrona com FIFO é indecidível se um elemento de um domínio de defeitos está conforme a especificação. Porém, existem domínios específicos em que a relação de comunicação assíncrona é decidível, como provado com os *Alternating* IOTSs, onde a conformidade e a equivalência são decidíveis em tempo polinomial de baixa ordem.

Estudo 24: Esse estudo [Jard and Jéron, 2005] é voltado para o teste de conformidade em sistemas reativos não-determinísticos. O trabalho apresenta a abordagem que foi implementada na ferramenta TGV, com o uso de modelos IOTSs. A teoria subjacente de TGV é apresentada juntamente com a noção de conformidade e veredito. Um algoritmo para geração de casos de teste foi proposto com algumas propriedades previamente estabelecidas e utilizando a exploração de grafos para geração de teste em tempo de execução. Lições aprendidas a partir de estudos de caso com a ferramenta são apresentadas. Também foi realizada uma comparação entre a ferramenta TGV e outras técnicas e ferramentas.

Estudo 25: Esse trabalho [Lestiennes and Gaudel, 2002] apresenta um método de seleção e derivação de testes baseado em um modelo de comunicação de processos com entradas, saídas e tipos de dados que está mais próximo da atual implementação de protocolos de comunicação. Foi considerada a relação de implementação *ioco* e foram definidos o conjunto de teste exaustivo-*ioco* e as hipóteses de teste a fim de garantir validade e imparcialidade. O método utiliza de maneira integrada a descrição do comportamento do processo e as propriedades dos tipos de dados. Foram propostas algumas orientações para escolha das hipóteses de teste, e as orientações atuais conduzem a conjuntos de teste onde casos nominais e no limite são cobertos.

Estudo 26: Esse estudo [Sampaio et al., 2009] mostra que quando a especificação e o modelo de implementação são anotados com eventos de saída que representam quietude, a relação *cspio* (baseada em CSP) é equivalente à *ioco*, com respeito ao mapeamento entre processos LTS e CSP. Também foi constrastada a estratégia de verificação de conformidade de *cspio* com um algoritmo de verificação em tempo de execução para *ioco*. Uma das vantagens dessa formalização é que benefícios podem ser obtidos a partir da semântica dos modelos e leis de CSP para realizar as provas formais. Tais provas podem ser mecânicas utilizando ferramentas como CSP-Power. A geração de testes pode ser guiada por propósitos de teste. Uma das principais contribuições é a verificação de conformidade automatizada, realizada por uma ferramenta que mecaniza todo o processo de geração de teste.

Estudo 27: Esse estudo [Tretmans, 2011] apresenta uma visão geral de TBM e da teoria *ioco*, utilizando LTSs e uma relação de implementação para verificar a corretude da implementação

com relação à especificação. Em seguida é apresentado o contexto desse novo tópico de pesquisa: modelagem baseada em teste, que consiste em realizar testes de modo que as observações realizadas sejam utilizadas na construção do modelo. É um tipo de engenharia reversa para o teste caixa-preta. Foram apresentadas de forma sistemática as adaptações necessárias para que o framework de TBM seja aplicado em conjunto com o aprendizado de modelos, considerando as consequências da perda de requisitos equivalentes entre os modelos aprendidos e o sistema ensinado.

Estudo 28: Esse estudo [Tretmans, 2008] tem o objetivo de ser um tutorial para uma teoria particular de TBM: a teoria de teste *ioco* para LTSs. Primeiramente, um framework para a teoria de teste formal de conformidade é apresentado. Nessa teoria, LTSs são utilizados como modelos para a especificação, implementação e testes. Além disso, a relação de conformidade formal *ioco* define a noção de conformidade entre a implementação e a especificação. Também é mostrado um algoritmo para a geração de casos de teste que produz casos de teste completos (consistentes e exaustivos). É considerada a execução de testes e a corretude do algoritmo de geração de testes.

Estudo 29: Alguns dos pressupostos básicos da relação de implementação *ioco* têm sido dados apenas informalmente. Esse estudo [Weiglhofer and Aichernig, 2010] contribui redefinindo a relação *ioco* na semântica predicativa denotacional de UTP (*Unifying Theories of Programming*), que traz os benefícios de uma representação formal. Além disso, a geração de teste baseada nessa teoria pode ser vista como um problema de satisfabilidade, facilitando o uso de técnicas modernas da teoria *sat* para a geração de teste. A versão UTP de *ioco* amplia seu escopo para linguagens de especificação com semântica similar à UTP.

Sistemas de Tempo Real

Estudo 30: Esse estudo [Briones and Brinksma, 2005] propõe uma extensão da teoria *ioco* para sistemas de tempo real, e foi baseado na interpretação operacional da noção de quietude. O modelo utilizado foi o Timed-IOTS e a relação de conformidade é a *tioco*. Essa relação detecta diferenças no comportamento depois da execução de *suspension traces*, mas pode não detectar diferenças no comportamento negado que requer a observação de quietude mais curta. O algoritmo de Tretmans [2008] para geração de teste foi estendido, considerando a questão temporal. Um exemplo mostrou como a teoria proposta pode ser usada para testar sistemas de tempo real, considerando a quietude.

Estudo 31: Esse estudo [Hessel et al., 2008] descreve uma abordagem para teste de conformidade em sistemas de tempo real, utilizando o modelo *Timed* IOTS. Essa abordagem foi implementada na ferramenta UPPAAL, que utiliza o modelo Uppaal *Timed Automata*, cuja semântica formal é em termos de *Timed* LTS. A abordagem propôs dois algoritmos para geração de testes: um para geração on-line e outro para geração off-line. O problema de geração de teste foi expresso como um problema de segurança, que foi resolvido com análise de alcançabilidade. O algoritmo para geração off-line foi produzido para atingir o máximo de cobertura e foi implementado na UPPAAL-Cover. Já o algoritmo para geração on-line tem o objetivo de fornecer

entradas, observar saídas ou reiniciar o teste, e foi implementado na versão UPPAAL-Tron. A abordagem mostrou-se aplicável na prática por exemplos de aplicação da abordagem.

Estudo 32: Extensões de *ioco* para sistemas de tempo real têm sido definidas, e esse estudo [Schmaltz and Tretmans, 2008] investiga essas extensões e relaciona-as entre si. Essas relações diferem-se sutilmente e foi mostrado sob quais condições elas são equivalentes. Nessa comparação, foram definidas outras duas relações de implementação: uma considera que todos os atrasos são observáveis e a segunda considera que somente atrasos até determinado limite são observáveis. As relações de implementação temporais propostas e suas relações formam a base para a teoria de TBM temporal e podem servir para comparar os algoritmos de geração e ferramentas correspondentes.

Baseado em propriedades

Estudo 33: Esse estudo [Falcone et al., 2012] apresenta classes de propriedades testáveis. Foi usada a noção de testabilidade que depende de uma relação entre o conjunto de sequências de execução que podem ser produzidas pela implementação e as consideradas propriedades-r (propriedades de tempo de execução). Foi dada uma caracterização precisa de propriedades testáveis e proposto um framework que permite obter oráculos de teste para produzir vereditos de acordo com as execuções de teste possíveis. O framework de propriedades-r fornece um testador canônico capaz de produzir um veredito dependendo de alguma interação finita entre o testador e a implementação. Foram propostas condições sob as quais faz sentido para um testador declarar um veredito fraco. Exemplos ilustraram como a proposta pode ser aplicada para os frameworks de teste existentes e também foi apresentada uma ferramenta que permite o uso dessas propriedades.

Estudo 34: Esse trabalho [Fernandez et al., 2004] propõe uma extensão do framework para a geração de casos de teste baseados em propriedades. A ideia é permitir a geração automática de testes a partir de especificações parciais e com respeito a uma propriedade particular, através dos seguintes passos: a especificação orienta a síntese de casos de teste; uma propriedade de segurança ou de vivacidade limitada é dada através de um observador que reconhece sequências que negam a propriedade; e casos de teste são gerados automaticamente examinando a especificação para encontrar a sequência de execução mais promissora capaz de mostrar a não satisfabilidade da propriedade anterior. Acredita-se que essa abordagem seja mais flexível, permitindo especificações parciais e a validação de propriedades explícitas.

Estudo 35: Esse estudo [Machado et al., 2007] apresenta uma visão geral de abordagens de teste orientada a propriedades para sistemas reativos, na qual propriedades são declaradas como propósitos de teste, utilizando modelos LTS e STS. Em ambos os casos, o embasamento teórico e ferramental tem sido desenvolvido com estudos de caso. O teste orientado a propriedades é um tipo de teste de conformidade onde os critérios de seleção são associados com uma dada propriedade que necessita ser verificada. São comparadas as abordagens para LTS e STS. Nos dois formalismos, a seleção de casos de teste ainda é um desafio.

Execução Simbólica

Estudo 36: Esse trabalho [Clarke et al., 2001] fortalece as ideias fundamentais de teste de conformidade e a alta eficiência na geração de teste incorporado na ferramenta TGV. Essa ferramenta realiza sua análise enumerando o espaço de estados da especificação, de modo que surgem problemas como explosão no espaço de estados e testes que não são compreensíveis por humanos quando representados por um grande sistema de transição. Para evitar tais problemas, foram aplicadas técnicas simbólicas para realização da análise. Foi proposto um método para a geração de casos de teste simbólicos, expressos com o formalismo IOSTS, que é considerado um formalismo em formato intermediário e empregado em ferramentas de automatização de teste. O método permite a geração automática de casos de teste e determina se os resultados da execução do teste estão corretos com relação à especificação. A ferramenta STG foi implementada para exemplificar a aplicação da técnica. Uma diferença em relação à ferramenta TGV é que a técnica proposta permite que os parâmetros do sistema sejam limitados tão tarde quanto o tempo de execução do teste.

Estudo 37: Esse estudo [Faivre et al., 2007] estende o framework de IOSTS introduzido em trabalho anterior para lidar com especificações de sistemas baseados em componentes. São utilizados mecanismos simbólicos para computar o comportamento dos subsistemas restringidos pelos sistemas na qual eles estão envolvidos. O modelo IOSTS é utilizado para denotar componentes básicos e dois operadores de estruturação: de composição e de ocultamento. Tais mecanismos permitem recuperar facilmente todas as informações relevantes relativas aos subsistemas. Também foi mostrado como usar esses comportamentos como propósitos de teste no framework de teste de conformidade baseado em *ioco*. Um exemplo ilustra que os comportamentos não podem ser extraídos pensando unicamente a nível de componente.

Estudo 38: O estudo de Frantzen et al. [2005] investigou formas de combater o problema de explosão de estados que é encontrado em ferramentas de teste com modelos baseados em estados. Foi observada a teoria de teste *ioco* para o contexto de STS, que tratam dados simbolicamente e produzem modelos com tamanho infinito. Esse trabalho considerou as relações entre os modelos STS, LTS e IOTS e a geração e execução de casos de teste. Desse modo, notou-se que a semântica de LTS e de *ioco* podem ser reutilizadas para o cenário simbólico, incluindo o clássido algoritmo para geração de casos de teste para *ioco*.

Estudo 39: Apesar de LTS ser um modelo semântico poderoso para descrever sistemas, ele possui limitações que o tornam incômodo: não é possível usar valores de dados e variáveis. Para superar tais problemas, foi proposto em um trabalho anterior o modelo Symbolic Transition Systems (STS), que incorpora a noção de dados e fluxo de controle dependente de dados. Esse estudo [Frantzen et al., 2006] propõe uma relação de implementação para tais modelos, sendo uma versão completamente simbólica de *ioco*: *sioco*. Desse modo, é possível ter um framework completo para o teste simbólico considerando a quietude. Para demonstrar a aplicabilidade, foi mostrada uma medida de cobertura de teste que é baseada em estados simbólicos. Tal medida possui vantagens sobre medidas de cobertura baseadas em estados semânticos ou de localização.

Estudo 40: Esse estudo [Jeannet et al., 2007] considera modelos IOSTS para sistemas reativos. Foi proposta uma extensão à teoria de teste *ioco*, lidando com modelos que contém tanto dados e controle quanto enumeração de valores de dados. O trabalho visa a seleção de teste off-line, na qual um caso de teste é construído a partir da especificação e um propósito de teste, e posteriormente executada na implementação. A técnica proposta evita o problema de explosão de estados devido à enumeração de valores de dados.

Estudo 41: Esse estudo [Le Gall et al., 2007] apresenta uma abordagem baseada em teste para verificar se uma especificação concreta é um legítimo refinamento de uma especificação abstrata. Foi utilizada uma combinação de execução concreta e simbólica da especificação, utilizando o modelo IOSTS. O método proposto foi inspirado em um framework de teste de conformidade bem estabelecido, baseado em propósitos de teste para seleção de casos de teste. Primeiramente, foi introduzida uma relação de refinamento. Foi realizada uma exploração cega da especificação concreta com respeito à um *trace* observável extraído da especificação abstrata. Um exemplo ilustrou a abordagem proposta mostrando alguns detalhes do algoritmo e implementações.

Estudo 42: Esse estudo [Rusu et al., 2005] apresenta uma definição e um estudo da cadeia formal de validação. É combinada a verificação e o teste de conformidade, verificando propriedades, gerando e executando casos de teste. A principal contribuição está no segundo passo: geração de casos de teste, com a proposta de um algoritmo que considera a natureza de infinitos estados da especificação e a incompletude dos passos de verificação. As entradas do algoritmo são a especificação e propriedades de segurança, e como resultado são produzidos casos de testes que satisfazem tais propriedades. Para lidar com os infinitos estados, considerou-se um modelo simbólico: IOSTSs, e o algoritmo também é simbólico. A execução dos casos de teste pode detectar violações nas propriedades da especificação, da implementação, ou ainda de conformidade entre implementação e especificação. Um exemplo simples ilustrou a abordagem.

Robustez

Estudo 43: Esse estudo [Fu and Kone, 2012] tratou do problema de teste de robustez para componentes concorrentes e em rede, estendendo a teoria de teste a partir de IOLTS. Um algoritmo para geração de casos de teste de robustez foi apresentado. Esse método pode ser utilizado para projetar ou examinar protocolos de rede, incluindo múltiplos componentes e diferentes camadas de protocolo de rede. Um estudo de caso com componentes concorrentes usando o protocolo RADIUS apresentou a viabilidade da proposta.

Estudo 44: Esse estudo [Rollet and Saad-Khorchef, 2007] trata do teste de robustez para sistemas embutidos temporais e não temporais. A abordagem proposta possui os seguintes passos: duas especificações são produzidas - uma nominal e uma reduzida - modeladas como IOLTSs e, se for temporal, Timed Input/Output Automata; sequências de teste são derivadas das duas especificações; defeitos são injetados nessas sequências de teste a fim de testar a robustez de forma controlada; as sequências são aplicadas na implementação e as respostas são registradas; e finalmente, são verificadas se a experimentação das sequências de teste é compatível

com a especificação reduzida, ou seja, se de fato as principais funcionalidades da especificação reduzida são reconhecidas pela implementação. Por último, uma relação de robustez entre a implementação e a especificação reduzida foi proposta. Como vantagem, essa abordagem evita a explosão de estados implicados na adição de defeitos na especificação.

Estudo 45: Nesse trabalho [Rollet and Salva, 2009] é proposto um método de geração de casos de testes envolvendo aspectos de robustez para sistemas de comunicação. É tratado apenas do teste funcional caixa-preta, onde apenas as entradas e saídas são conhecidas e o sistema é descrito como um IOLTS. As situações não especificadas consideradas nesse trabalho são: ocorrência de uma entrada conhecida mas não esperada naquele momento; ocorrência de uma entrada não conhecida; e ocorrência de uma saída não conhecida. A abordagem é baseada em *ioco*, possibilitando detectar estados quiescentes na implementação. É considerado um alfabeto aumentado do sistema, e foi proposta uma maneira de injetar as situações não especificadas nos casos de teste. Um algoritmo (extensão de Tretmans), foi proposto para obter um conjunto de casos de teste sólidos e exaustivos.

Estudo 46: Esse estudo [Rollet and Salva, 2008] apresenta duas abordagens complementares para o teste de robustez. A primeira identifica alguns defeitos clássicos na especificação considerando eventos inesperados, e usa-os para criar uma matriz de defeitos, ou seja, uma especificação ampliada. Já a segunda integra diretamente os defeitos nas sequências de teste, nas quais tem-se duas especificações, uma nominal e uma reduzida com o mínimo do comportamento necessário. A principal diferença entre as duas é o fato de que robustez implica ou não em conformidade.

Teste Distribuído

Estudo 47: Esse estudo [Brinksma et al., 1998] discute uma técnica de geração de teste fatorada para o critério de corretude multi-*ioco* (*mioco*). A técnica é considerada fatorada no sentido de que corretudes complicadas podem ser divididas naquelas que são fáceis de serem verificadas, sendo viável para amplas especificações. Foi utilizado Multi-IOTS para essa teoria de corretude *mioco* e o algoritmo proposto é capaz de gerar testes a partir da especificação com respeito à *mioco*. Condições são apresentadas nas quais a especificação pode reduzir o tamanho dos testes sem a habilidade de gerar testes válidos. Duas técnicas fatoradas são apresentadas: uma para gerar um conjunto de teste consistente e outra para gerar um conjunto de teste completo com respeito à *mioco*.

Estudo 48: Esse estudo [Cavalcanti et al., 2011] investiga relações de refinamento para CSP - uma álgebra de processos bem estabelecida - quando o teste ocorre em um conjunto de testadores distribuídos. Enquanto existem muitos estudos em teste distribuído para IOTSs e Máquinas de Estado Finito, o uso de CSP introduz novos desafios. Primeiro foi considerado o refinamento cooperado, onde existe a possibilidade de verificar coletivamente as observações de um mesmo ponto. Depois foi considerado o refinamento independente, onde não existe forma de sintetizar as observações. Também foi considerada a geração de teste e mostrado como mensagens coordenadas podem ser utilizadas para tornar os testes consistentes.

Estudo 49: Esse trabalho [Hierons, 2012a] discute o uso de mensagens coordenadas e como elas podem ser adicionadas aos casos de teste para uso em teste distribuído. Foi utilizado o modelo IOTS. O algoritmo apresentado para decidir se um caso de teste global é controlável caracteriza o conjunto de problemas de controlabilidade. Além disso, um segundo algoritmo adiciona mensagens coordenadas em um caso de teste global a fim de superar os problemas de controlabilidade. Ambos os algoritmos tem complexidade polinomial. Porém, o problema de encontrar um conjunto minimal de mensagens coordenadas para superar problemas de controlabilidade é NP-difícil, mesmo se for restringida atenção aos casos de teste que estão na forma de árvores.

Estudo 50: O uso de arquitetura de teste distribuído conduz a problemas adicionais de controlabilitade e observabilidade. Esse estudo [Hierons, 2008] descreve alguns trabalhos que visam tanto conduzir a um melhor entendimento do efeito da arquitetura de teste distribuído, quanto estender o trabalho de modelos e implementações não-determinísticas. Existem diferenças e desafios adicionais se testamos com IOTS, uma vez que podemos ter espaço de estados infinito e entradas e saídas não necessitam ser alternadas. Trabalhos recentes tem caracterizado o poder de testar em arquitetura de teste distribuído. A esperança é que isso conduzirá a algoritmos de geração de teste que produzam sequências de teste que objetivem a correspondente noção de equivalência fraca, visando decidir se o sistema e a especificação podem ser distinguidas nessa arquitetura.

Estudo 51: Esse estudo [Hierons et al., 2008] define o que significa um caso de teste ser controlável e mostra que é possível decidir em tempo polinomial se um caso de teste possui tal propriedade. Como formalismo, foi utilizado o modelo IOTS. Uma nova relação de implementação, *c-dioco*, foi proposta para teste controlável na arquitetura de teste distribuído. Foi proposto um algoritmo não-determinístico para gerar casos de teste controláveis, sendo uma adaptação de um algoritmo para gerar casos de teste a partir de MEFs temporais.

Estudo 52: No contexto de teste com interfaces distribuídas, esse estudo [Hierons et al., 2012b] considerou uma perspectiva alternativa para fornecer informação adicional em relação à causalidade entre ações realizadas em diferentes portas. Foi utilizada a informação de tempo, e foram investigados diferentes pressupostos em relação a como o temporizador local se relaciona com as relações de implementação correspondentes. Adicionar tempo às ações é um mecanismo comum em formalismos como álgebra de processos, para representar sistemas temporais concorrentes. Foi considerado o teste a partir do modelo IOTS, e para cada cenário de teste distribuído foram propostas relações de implementação correspondentes.

Estudo 53: Esse estudo [Hierons et al., 2012a] investiga relações de implementação e teste para IOTSs com múltiplas portas. Três relações de implementação foram definidas para especificações habilitadas para entrada. Foi mostrado como elas se relacionam com outras relações e com a relação de implementação *ioco*, que é usada tradicionalmente quando há somente uma porta. Também foi explorado o conceito de caso de teste local, mostrando como mapear um caso de teste global controlável para um caso de teste local controlável. Um algoritmo foi proposto para mostrar se um caso de teste é controlável, juntamente com um algoritmo

para gerar casos de teste que fornecem as características das relações de implementação propostas.

Estudo 54: Esse estudo [Hierons et al., 2011] trata do teste formal de sistemas com portas distribuídas. Foi introduzida a relação de implementação *sdioco*, que é a relação *dioco* baseada em Cenários, que representa uma extensão adequada para relações estabelecidas anteriormente. Foi analisado como as condições dessa nova relação afetam a noção de controlabilidade para geração de casos de teste controláveis. O modelo IOTS foi utilizado como formalismo. Desse modo, foi definido o que significa para um sistema passar um caso de teste nas novas condições da relação de implementação proposta.

Estudo 55: Esse estudo [Hierons and Núñez, 2012] considera o teste de sistemas que interagem com seu ambiente através de um número de interfaces fisicamente distribuídas e onde as observações são, portanto, distribuídas. Apenas recentemente foram considerados modelos probabilísticos, fornecendo um framework de teste formal para lidar com esse tipo de sistemas. Nesse estudo, foi considerado o uso de planejadores, também chamados adversários, em sistemas probabilísticos. Uma relação de implementação foi proposta, que depende dos *traces* observados nas diferentes portas. Essa metodologia pode ter semelhanças com noções semânticas de teste para autômatos probabilísticos. Foi considerada a noção de planejadores globais e locais. Assim, novas relações de implementação foram propostas e suas propriedades estudadas, associando-as com relações previamente definidas.

Estudo 56: Esse estudo [Hierons and Núñez, 2010] investigou o problema de teste de sistemas probabilísticos que tem interfaces distribuídas, chamadas portas. Esse é um problema significante, uma vez que sistemas distribuídos são geralmente probabilísticos por natureza. Primeiramente, foi explorado o teste com sistemas de transição probabilísticos, definindo uma relação de implementação onde o testador local pode retornar seu conjunto de observações e um veredito pode ser produzido a partir deles. Também foi explorado o teste para IOTS probabilístico que tem múltiplas portas, e uma relação de implementação para tal teste foi definida.

Estudo 57: Esse estudo [Li et al., 2004a] descreve um algoritmo para geração de teste para Multi IOTSs. O algoritmo gera um conjunto de casos de teste para uma classe especial de MIOTSs: todos-observadores, que é um tipo de sistema que pode observar todos os canais de saída simultaneamente. Tal conjunto de casos de teste é menor que um conjunto de casos de teste para um observador singular, economizando tempo tanto na geração quanto na execução de testes. Foi estudado também o problema de gerar testes todos-observadores fatorados, utilizando uma técnica para observador singular.

Estudo 58: Esse estudo [Li et al., 2003] estende o teste de rejeição com Multi IOTSs para uma teoria mais genérica que é aplicada para testar implementações com canais bloqueáveis e não-bloqueáveis. Foi proposta uma classe de observadores, chamada observadores plurais, para definir uma nova relação de teste de rejeição. Como características dessa relação estão a inclusão de *suspension traces* e conjuntos de saída são investigados. Foi desenvolvido um algoritmo para geração de conjuntos de teste completos com respeito à relação *mioco*.

Estudo 59: Esse estudo [Li et al., 2004b] apresenta uma teoria de teste de rejeição em teste distribuído. Foi estudada a distribuição dos testes de observador singular centralizados e a distribuição dos testes de todos-observadores centralizados. Observadores singulares acessam um canal após outra implementação e nunca observam mais do que um canal simultaneamente. Para distribuir o teste de observador singular, ativa-se os testadores um a um para comunicarem-se com a implementação. O processo de teste proposto acontece realmente como uma corrida de revezamento.

3.1.6 Teste assíncrono

Estudo 60: O problema de produzir estratégias (casos de teste) foi tratado nesse estudo [Hierons, 2012b] para alcançar três objetivos: alcançar estados de um IOTS, executar transições em um IOTS e distinguir dois estados em um IOTS. Foram considerados canais FIFO e não-FIFO. Foi mostrado que os problemas citados podem ser resolvidos em tempo polinomial para IOTSs observáveis. O problema geral é EXPTIME-difícil. Além de ser utilizado para geração de casos de teste, um IOTS observável pode também ser utilizado para verificar se uma sequência observável de observações contém uma falha (problema do Oráculo), e tal problema em canais FIFO pode ser resolvido em tempo polinomial. Para canais não-FIFO os problemas transpiram ser EXPTIME-difícil.

Estudo 61: Esse estudo [Huo and Petrenko, 2005] investiga o problema de projetar uma estratégia de teste em um contexto de filas, garantindo a execução de uma transição específica em IOTSs. Foi estudado como cobrir transições com ou sem pressupostos imparciais. Para a cobertura de transições sem pressupostos imparciais, o problema foi completamente resolvido pela construção de estratégias de jogo compostas apenas de *traces* livres de conflito de entrada e saída da especificação. Para a cobertura com pressupostos imparciais, o problema foi parcialmente resolvido. A abordagem é aplicável a outros tipos de sistemas, mas em IOTSs determinísticos, cujos *traces* são livres de conflitos de entrada e saída, essa solução é completa e eficiente porque é possível encontrar estratégias de vitória razoáveis em tempo polinomial.

Estudo 62: Esse estudo [Huo and Petrenko, 2004] investiga como testar IOTSs através de filas sem perdas (que armazenam as entradas para consumi-las depois). Nesse trabalho considera-se que as especificações são parciais, significando que a especificação não é necessariamente habilitada para a entrada. Ao contrário de trabalhos anteriores, pressupõe-se que a especificação do sistema pode ter tanto ações de entrada bloqueadas quanto ações não especificadas. Dois algoritmos para derivação de testes são apresentados com o critério de cobertura em mente: um para IOTSs completamente especificados e outro para IOTSs parcialmente especificados no contexto de filas ilimitadas. Com base nos conjuntos de teste derivados do teste de IOTSs através de filas ilimitadas, discute-se como usar filas limitadas na arquitetura de teste, que torna o método mais prático. O uso do critério de cobertura para derivar uma suite de teste finita evita a composição da especificação com filas ilimitadas ou a elaboração de uma implementação com um observador local.

Estudo 63: Dados alguns problemas identificados quando é utilizado um canal de comunicação no teste assíncrono, são necessárias adaptações para que os testes gerados para a comunicação assíncrona alcancem o veredito correto. Nesse contexto, esse estudo [Noroozi et al., 2011] revisitou algumas questões de teste assíncrono e formulou um conjunto de teoremas e provas que demonstram quando é possível sincronizar o teste assíncrono. Desse modo, esses teoremas estabelecem condições de suficiência quando o veredito alcançado pelo teste de sistemas assíncronos corresponde com o teste local através de interação síncrona. No caso do teste *ioco*, mostramos que tais condições também são necessárias.

Estudo 64: Esse estudo [Simao and Petrenko, 2011] investiga o problema de construir casos de teste assíncronos para um dado propósito de teste e uma especificação em IOTSs que é consistente quando o testador e a implementação comunicam-se por filas. Foi identificada uma classe de especificações de IOTSs em que os casos de testes assíncronos e síncronos coincide. Foi encontrada uma maneira de transformar o propósito de teste prévio para compor a especificação tal que a distorção das filas seja considerada na composição de filas resultante. Com isso, é possível gerar casos de teste consistentes evitando a composição da especificação com filas.

Estudo 65: Esse estudo [Weiglhofer and Wotawa, 2009] discute como restringir as escolhas do testador para ajudar a mitigar alguns dos pressupostos do teste de conformidade de entrada e saída. Foi mostrado como garantir vereditos corretos no caso assíncrono para um amplo conjunto de implementações. Nos IOTSs, foi diminuído o requisito de habilidade para entrada, requerindo essa habilidade apenas nos estados com quietude. Acredita-se que isso é mais útil na prática do que assumir a habilidade para entrada em todos os estados. Os resultados do exemplo aplicado para Torx com essa abordagem indicaram a necessidade de manipular eventos assíncronos mesmo para um simples exemplo.

3.1.7 Outros

Estudo 66: Esse estudo [van Beek and Mauw, 2004] trata do teste de conformidade para aplicações *Web*, mostrando as diferentes características desse tipo de aplicação. Tais aplicações podem ter falhas de comunicação entre cliente e servidor e ainda mensagens chegando em ordem inversa. O trabalho foi conduzido no contexto do projeto DiCons, visando o teste dinâmico das aplicações *Web*. Para modelar tais sistemas, foi utilizado o modelo *Multi Request-Response Transition Systems* (RRTS), que pode ser comparado à *Multi IOTS*, com características de aplicações *Web*. Para a geração do conjunto de testes, foi estendido o algoritmo de Tretmans [2008] mostrando que os conjuntos gerados são completos. Um exemplo mostrou a aplicabilidade da proposta, revelando erros rapidamente.

Estudo 67: Um problema conhecido da bem-estabelecida teoria de teste para MEFs, em particular máquinas de Moore e Mealy, é a identificação de estados. Porém, para modelar sistemas assíncronos, IOTS são modelos mais adequados, sendo utilizados no teste de conformidade. Esse estudo [Bensalem et al., 2008] propõe uma abordagem para identificação de estados em

IOTS, realizando a transformação de um IOTS determinístico em uma máquina de Mealy determinística, na qual o problema de identificação de estados pode ser resolvido com as técnicas existentes. Esse método permite verificar se dois estados são distinguíveis e a existência de experimentos predefinidos/adaptativos de distinção e de *homing* (com saídas diferentes). Tal método pode ser facilmente adaptado para verificação e sincronização de estados.

Estudo 68: Já foi provado que o comportamento remoto de um IOLTS pode ser capturado por gramáticas livres de contexto [Bhateja, 2009]. Como qualquer linguagem expressa em uma gramática livre de contexto permite a geração automática de Push-Down Automata (PDA), esse estudo [Bhateja, 2011] propõe uma metodologia para simular um PDA a partir de um dado IOLTS. Esse PDA pode gerar os casos de testes necessários para testar o IOLTS remotamente. O estudo descreve como simular um PDA a partir de um IOLTS, considerando dois casos: teste estático e teste dinâmico. PDA é um modelo computacional utilizado principalmente para o fluxo de controle de um programa recursivo.

Estudo 69: Esse estudo [Bhateja, 2009] propõe a modelagem de sistemas com base em uma gramática livre de contexto, que é convencionalmente realizada por Sistemas de Transição Rotulados (LTS). O estudo apresenta uma metodologia para converter LTS em uma gramática livre de contexto, e mostra como essa gramática baseada em modelos auxilia na geração e avaliação dos testes. O uso de uma gramática livre de contexto possui vantagens, tais como: seu formalismo é amplamente lido e compreendido; é muito utilizada em compiladores e interpretadores, sendo fácil colocá-la em uso; e suas sentenças podem ser descritas usando árvores, que é uma forma de representação simples.

Estudo 70: Nesse estudo [Boroday et al., 2008] é investigado um problema comum da área de teste: testar o testador. O método proposto verifica a consistência do conjunto de testes, propondo condições de consistência e boa formação de testes. Além disso, o método verifica um conjunto de testes com dependências intertestes (pré-condições) que, ao contrário de outros trabalhos, não depende da especificação. Ao mesmo tempo, foi mostrado como as inconsistências, detectáveis sem qualquer uso da especificação do sistema, relatam falhas dos testes, tais como inconsistência e negligência. Tais falhas foram previamente definidas e detectadas apenas com base em uma dada especificação. Também discute-se questões relacionadas a testes com veredito inconclusivo e refinamento da relação de consistência dos testes.

Estudo 71: Esse estudo [Brandl et al., 2010] propõe usar *Qualitative Reasoning* (QR), técnica de Inteligência Artificial, para descrever o comportamento contínuo de sistemas híbridos. Na técnica proposta, são utilizadas Equações Diferenciais Qualitativas para descrição do comportamento. Um trabalho anterior introduziu Sistemas de Ação Qualitativa, dando uma semântica de LTS a esses sistemas. Assim, foi possível aplicar uma extensão do teste *ioco* para verificar a conformidade de sistemas híbridos, já que um rótulo especial foi introduzido, denotando a evolução das quantidades de entradas e saídas. Um protótipo foi desenvolvido para verificar a conformidade em tempo de execução. O experimento realizado mostrou bons resultados da verificação de conformidade em especificações com mutação.

Estudo 72: Esse estudo [Calamé et al., 2007] propõe um framework de teste que alivia o problema de gerar um número muito grande de casos de teste pela combinação de abstrações de dados e solução de restrições com técnicas de geração de teste enumerativa. Dado um propósito de teste, é possível obter um caso de teste abstrato aplicando algoritmos de geração de teste enumerativos para o sistema abstrato. Tal sistema é derivado a partir de uma especificação transformada. Para isso, os casos de teste devem ser executáveis, isto é, o caso de teste abstrato deve ser concretizado. A transformação ocorre da especificação original para um programa de lógica de restrições e transfere o solucionador de restrições para a fase de execução do teste. A geração de testes ocorre na ferramenta TGV, que utiliza IOLTS como formalismo. A aplicação de estudos de caso em outros estudos já publicados mostram que a abordagem é escalável em sistemas de tamanho industrial.

Estudo 73: Para a geração de teste em programas reativos recursivos, existem várias técnicas para definir comportamento recursivo, tais como push-down automata, máquinas de estado recursivo, grafos regulares, e gramáticas de substituição. Todas elas possuem suas vantagens. Esse estudo [Chédor et al., 2012] buscou obter o melhor delas construindo o modelo *Recursive Tiles Systems*, que é um modelo que define IOLTS de infinitos estados baseado em grafos regulares. Foi considerada uma classe importante de RTS: *weighted* RTS. Essa classe possui a propriedade de ser determinizável, e determinizar um *weighted* RTS é decidível. Foram propostos algoritmos para produzir casos de testes consistentes, estritos e exaustivos, tanto na geração off-line quanto on-line. Esses algoritmos habilitam a implantação de propósitos de teste, e foram utilizados propósitos de teste para a seleção de testes.

Estudo 74: Esse estudo [Desmoulin and Viho, 2009] fornece uma definição formal para interoperabilidade, chamada de critérios de interoperabilidade, que fornece condições para verificar se a implementação é considerada interoperável. Com base nesse critério, um método para gerar casos de teste interoperáveis automaticamente foi proposto, evitando o conhecido problema da explosão de estados. O modelo IOLTS foi o formalismo utilizado. A equivalência entre os dois critérios propostos - critério global e bilateral - conduziu a proposta do novo método de geração. O algoritmo para geração foi baseado em dependências entre eventos. Um exemplo mostrou como aplicar a proposta, comparando a geração com os dois critérios propostos.

Estudo 75: Esse estudo [Fraser et al., 2008] apresenta um conjunto de propósitos de teste genéricos baseados em critérios de cobertura para especificações em Lotos, cuja semântica é dada por IOLTSs. Tal abordagem estabelece uma estratégia de seleção de testes juntamente com a eficiência de algoritmos de derivação de teste baseado em propósitos de teste. Os critérios propostos foram: cobertura de ações, cobertura de decisões, cobertura de condições, e cobertura de decisões/condições. Também foi proposta uma técnica para minimizar o conjunto de testes gerados. Foi demonstrada a viabilidade dessa abordagem com experimentos que utilizaram diferentes versões de aplicações industriais.

Estudo 76: Esse estudo [Hao and Wu, 1997] apresenta uma abordagem de execução de teste formal baseada em TTCN (notação para teste de protocolo). A abordagem consiste em duas

fases: na primeira, um interpretador converte casos de teste em notação TTCN. Na segunda fase, um executor, definido como uma máquina TTCN, realiza a avaliação da árvore gerada na primeira fase de acordo com a semântica de TTCN e é dado o veredito. Modelos IOTSs descrevem TTCN. Foi mostrada uma aplicação prática e técnicas de implementação importantes foram discutidas. A abordagem proposta é um componente importante do teste de conformidade de protocolos, tem boa escalabilidade, é independente de protocolo e de sistema e foi o início de uma ferramenta para realizar o teste de conformidade.

Estudo 77: Esse estudo [Kervinen and Virolainen, 2005] orienta o teste de conformidade baseado em LTS, introduzindo e comparando algoritmos com heurísticas que visam revelar erros rapidamente. Foi mostrado como o problema de projetar a cobertura de uma especificação juntamente com o algoritmo de seleção de teste pode ser dividida em partes. Critérios de cobertura foram formalizados e uma arquitetura para mecanismo de teste foi apresentada, mostrando o contexto na qual o algoritmo opera. O principal algoritmo apresentado inclui separação de passos e de estados. Os algoritmos são comparados pelo número de passos de teste requeridos para detectar erros. Os experimentos realizados mostram que quando os erros são simples, a função de avaliação que mede a cobertura do nome da ação junto com a cobertura de transição detecta erros mais rapidamente.

Estudo 78: Esse estudo [Muccini et al., 2004] fornece um gerenciador de teste com um método sistemático para extrair classes de teste adequadas para teste de alto nível. Tais classes são refinadas em testes concretos em nível de código com base na especificação da dinâmica da arquitetura de software. Essa dinâmica é modelada como um LTS abstrato (ALTS). A relação entre os testes de arquitetura abstrata, os testes de arquitetura concreta e os testes executáveis precisa ser estabelecida de modo que os testes arquiteturais possam ser refinados em testes a nível de código. Assim, derivar um conjunto adequado de teste ocasiona na derivação de caminhos que cobrem o ALTS. Um estudo de caso mostrou a viabilidade da abordagem em um contexto real.

Estudo 79: Esse estudo [Nachmanson et al., 2004] apresentou três algoritmos para teste de sistemas não-determinísticos. A geração de casos de teste nessa abordagem equivale a uma estratégia de jogo, onde os dois jogadores são a ferramenta de teste e a implementação. O jogo explora alguns grafos de estados. O primeiro algoritmo determina uma estratégia ótima para jogo de alcançabilidade limitada. No segundo algoritmo é determinada uma estratégia de vitória para jogo de alcançabilidade limitada. Por fim, o terceiro algoritmo determina uma estratégia rápida de cobertura de arestas. Para garantir casos de teste pequenos foi introduzida uma função de custo de arestas. A estratégia de jogo decide qual aresta deve ser considerada pela ferramenta em cada estado para ganhar o jogo. Os algoritmos são implementados na ferramenta de teste AsmL.

Estudo 80: Esse estudo [Pardo et al., 2010] define um novo formalismo, *Utility* IOLTS, que consiste em um LTS com características para definir o comportamento de agentes. Três relações de implementação que podem ser utilizadas para estabelecer a conformidade entre a implementação e a especificação foram definidas para estabelecer a conformidade entre uma

implementação e sua especificação. Foi redefinida a noção de teste para obter mais informações a partir da implementação. Um algoritmo para derivar conjuntos de teste completos a partir da especificação foi proposto.

Estudo 81: Esse estudo [Ruibing and Jianping, 1998] propõe uma abordagem formal para a construção de teste de sistemas interoperantes baseados na semântica operacional de TTCN Concorrente, que é uma notação que manipula comportamentos para teste de sistemas concorrentes. Foi formalmente definida a semântica operacional de TTCN baseada em Input-Buffered Transition Systems. Foi discutida a execução de teste e o processo de veredito de teste de interoperabilidade com TTCN. Um protótipo foi implementado com tal proposta, e após a realização de um estudo de caso bons resultados foram alcançados.

Estudo 82: Esse estudo [Scollo and Zecchini, 2005] apresenta uma combinação de modelagem arquitetural orientada a objetos e a semântica de IOLTSs como um framework de teste formal para avaliar a testabilidade de requisitos funcionais. Primeiramente, os requisitos dinâmicos das unidades arquiteturais são modelados como diagramas de estados da UML. O segundo passo é uma tradução mecânica dos diagramas UML para diagramas de estados em LOTOS (com semântica de IOLTSs). A ferramenta TGV gerou os casos de teste com a semântica de IOLTSs. Um estudo de caso foi realizado com sucesso, realizando a análise e geração de casos de teste.

Estudo 83: Esse estudo [Weiglhofer et al., 2009] apresenta um conjunto de propósitos de teste genéricos baseados no critério de cobertura para especificações Lotos. Dois novos critérios são apresentados: cobertura de processos e cobertura de condições/decisões modificadas. Tais propósitos genéricos auxiliam a geração manual de propósitos de teste, mas garantem apenas uma cobertura fraca, enquanto apenas um caso de teste é derivado de cada propósito de teste. Os resultados experimentais da aplicação desses dois novos critérios foram apresentados. Foram derivados conjuntos de teste utilizando os critérios propostos, mostrando que a cobertura alcançada pelo projeto manual de propósitos de teste é potencialmente insuficiente, sendo recomendada então uma técnica complementar baseada em cobertura.

Estudo 84: Esse estudo [Willemse, 2007] apresenta técnicas de modelagem baseada em teste, que é inclinada ao aprendizado de máquina e utilizada para obter modelos parciais a partir de experimentos na implementação. A abordagem obtém modelos de sistema utilizando técnicas de TBM, e o objetivo é utilizar os modelos para teste de regressão. Um algoritmo básico, baseado em aproximação, juntamente com um conjunto de heurísticas são propostos para a construção de modelos do sistema baseado em contra-exemplos encontrados na teoria de teste *ioco*. Um estudo de caso demonstra o uso da técnica proposta e sua efetividade é ilustrada com teste de mutação. Como resultado, o estudo de caso detectou 85% de todos os mutantes.

Estudo 85: Esse trabalho [Xie and Dang, 2006] trata do problema de teste global caixa-preta para sistemas concorrentes, que consistem de um sistema hospedeiro e um número de caixas-pretas. O modelo LTS representa as ações desse tipo de sistema. Uma nova abordagem, denominada técnica *push-in* foi proposta para resolver o problema de teste global, que é reduzido para testes caixa-preta individuais realizados um a um em determinada ordem. Utilizando

uma abordagem baseada em autômatos, sequências de teste para cada caixa-preta são geradas a partir da descrição do sistema bem como os resultados do teste caixa-preta anterior são dados em determinada ordem. Tal técnica é automática, consistente e completa, mostrando que nenhum comportamento global ruim aparece no final do teste. Uma série de experimentos foram executados e os resultados relatam que o problema de teste global pode ser completamente solucionado com um número bem menor de testes sobre cada caixa-preta individual.

CAPÍTULO

4

Análise dos Resultados

Este capítulo apresenta os resultados obtidos após análise dos 85 estudos selecionados em relação aos seguintes tópicos: modelos, formas de aplicação dos testes, métodos de geração, tipos de validação, ferramentas, autoria e referências, de modo que as questões de pesquisa sejam respondidas.

4.1 Modelos

Nesta seção são discutidos os modelos empregados nos métodos de teste identificados nos estudos. Apesar do foco deste estudo ser IOTSs, foi possível identificar definições variadas bem como nomenclaturas diferentes para modelos com as mesmas características, conforme visto na Figura 2.2. A seguir, são apresentados os principais modelos identificados nos estudos.

- **IOTS:** 24 trabalhos aplicam o modelo IOTS para modelagem dos testes. Nesse modelo, as entradas do ambiente nunca são recusadas pelo sistema e as saídas do sistema nunca são recusadas pelo ambiente. Isso caracteriza um modelo onde todas as entradas estão habilitadas em todos os estados [Bensalem et al., 2008; van der Bijl et al., 2005, 2004; Boroday et al., 2008; Hierons et al., 2012a, 2011; Hierons, 2012b,c; Huo and Petrenko, 2004; Lestiennes and Gaudel, 2002; Noroozi et al., 2011; Petrenko et al., 2003; Simao and Petrenko, 2011; Tretmans, 2011; Weiglhofer and Aichernig, 2010; Weiglhofer and Wotawa, 2009; Willemse, 2007; Hierons et al., 2012b; Hierons, 2008; Cavalcanti et al., 2011; Ruibing and Jianping, 1998; Hao and Wu, 1997; Huo and Petrenko, 2005; van Beek and Mauw, 2004]. Em vários estudos é considerado que um IOTS pode chegar ao estado de quietude [Tretmans, 2008; Jard and Jéron, 2005; Jéron and Morel, 1999; Weiglhofer and Wotawa, 2009; Hierons, 2012b; Huo and Petrenko, 2004].

- **LTS:** Esse modelo consiste de uma estrutura com estados e transições, de modo que as transições são rotuladas por ações entre elas (sem diferenciar se são entradas ou saídas). 6 estudos utilizam esse modelo [Bhateja et al., 2006; Bourdonov et al., 2006; Frantzen et al., 2005; Kervinen and Virolainen, 2005; Gromov and Willemse, 2007; Xie and Dang, 2006].

- **IOLTS:** 23 trabalhos consideram o modelo IOLTS (*Input/Output Labeled Transition Systems*), sendo que esse modelo é um LTS em que cada transição pode ser rotulada ou com uma ação de entrada ou uma ação de saída ou ainda por uma transição interna. A característica de habilidade para entrada é desconsiderada [Aichernig and Delgado, 2006; Aichernig et al., 2007, 2008; Bhateja, 2011; Briones et al., 2006; Calamé et al., 2007; Chen, 2011; Desmoulin and Viho, 2009; Falcone et al., 2012; Fernandez et al., 2004; Fraser et al., 2008; Gnesi et al., 2004; Jéron and Morel, 1999; Rollet and Saad-Khorchef, 2007; Rollet and Salva, 2009, 2008; Scollo and Zecchini, 2005; Weiglhofer et al., 2009; Brandl et al., 2010; Fernandez et al., 2005; Chédor et al., 2012; Jard and Jéron, 2005; Bi et al., 1998]. A quietude, estado em que o sistema não produz nenhuma saída e não pode proceder autonomamente, é considerada em alguns trabalhos [Jard and Jéron, 2005; Jéron and Morel, 1999].

- **IOSTS/STS:** Um STS (*Symbolic Transition Systems*) é um sistema de transição que incorpora uma noção explícita de dados e fluxo de controle dependente de dados. Um IOSTS (*Input/Output Symbolic Transition Systems*) é um STS que divide o conjunto de ações em ações de entrada e ações de saída. 8 estudos citam esses modelos [Clarke et al., 2001; Faivre et al., 2007; Frantzen et al., 2006; Gaston et al., 2006; Jeannet et al., 2007; Le Gall et al., 2007; Rusu et al., 2005; Frantzen et al., 2005].

- **TIOSTS:** O modelo TIOSTS (*Timed IOSTS*) é um sistema de transição onde dados e temporalidade são especificados simbolicamente, e é empregado em um estudo [Bannour et al., 2012].

- **TIOTS:** 4 estudos citam o modelo TIOTS (*Timed IOTS*), que é um IOTS com a noção de temporalidade [Briones and Brinksma, 2005; Hessel et al., 2008; Schmaltz and Tretmans, 2008; Pardo et al., 2010].

- **MIOTS:** Um MIOTS (*Multi IOTS*) é um IOTS em que as entradas e saídas são particionadas em diferentes canais, refletindo a localização em que ocorrem as ações. Além disso, um MIOTS não tem necessariamente a característica de habilidade para entrada, mas cada canal de ações de entrada deve ser simultaneamente habilitado. 6 estudos utilizam tal modelo [Brinksma et al., 1998; Li et al., 2004a, 2003, 2004b; Hierons, 2012a; Hierons et al., 2008].

- **PIOTS:** O modelo PIOTS (*Probabilistic IOTS*) é citado em 2 trabalhos, e consiste em um sistema de transição com distinção entre entradas reativas (uma distribuição de probabilidade para cada uma das entradas que partem de determinado estado) e uma saída geradora

(uma distribuição de probabilidade única para todas as saídas que partem de determinado estado) [Hierons and Núñez, 2012, 2010].

Analisando a variedade de definições e diferentes nomenclaturas, notou-se que alguns modelos são definidos com as mesmas propriedades. Embora a habilidade para entrada seja uma característica peculiar de IOTSs, alguns estudos consideram essa propriedade em IOLTSs [Jard and Jéron, 2005; Aichernig and Delgado, 2006; Aichernig et al., 2007, 2008; Calamé et al., 2007; Gnesi et al., 2004]. O inverso também foi encontrado, onde IOTSs foram definidos sem requerer a habilidade para entrada [Weiglhofer and Wotawa, 2009; Weiglhofer and Aichernig, 2010]. Esse fato mostra a flexibilidade das definições. É importante enfatizar que cada propriedade do modelo pode ser um fator determinante no contexto de geração de testes, por isso deve-se observá-las.

Embora a maioria dos estudos utilize IOTSs ou IOLTSs, outras notações são utilizadas ao considerar peculiaridades específicas para certos contextos:

- *GALTS - Generic Annotated Labeled Transition Systems* [Andrade and Machado, 2012]: tipo especial de LTS que possui descrições adicionais para viabilizar a geração de teste com interrupções.
- *ALTS - Abstract LTS* [Muccini et al., 2004]: uma redução de um LTS, conservando apenas os comportamentos que interessam à uma visão de arquitetura de software pré-definida.
- *IOSM - Input/Output State Machine* [Koné and Castanet, 2000]: similar ao modelo IOLTS.
- *RRTS - Request-Response Transition Systems* [van Beek and Mauw, 2004]: comparável ao modelo IOTS para o contexto de aplicações *Web*, de modo que a entrada é considerada uma requisição e a saída é a resposta à requisição.
- *RTS - Recursive Tiles Systems* [Chédor et al., 2012]: um IOLTS de infinitos estados baseado em grafos regulares.
- *UIOTS - Utility IOTS* [Pardo et al., 2010]: um IOLTS que introduz novas características para definir comportamento de agentes.
- *WFM - Weighted Fault Model* [Briones et al., 2006]: um LTS ampliado, com a informação adicional de peso para cada erro potencial em uma implementação.

4.2 Aplicação dos Testes

Com a observação das formas de aplicação dos testes nos estudos selecionados, foram identificadas características para aplicação dos métodos de geração de casos de teste (podendo existir mais de um contexto em um mesmo método):

Tipo de sistema

Nesse sentido, 22,35% dos estudos apresentam métodos de geração de testes criados com o objetivo de suprir peculiaridades de determinados tipos de sistemas, tais como sistemas de tempo real [Bannour et al., 2012; Briones and Brinksma, 2005; Hessel et al., 2008], sistemas de protocolo de comunicação [Aichernig et al., 2007, 2008; Hao and Wu, 1997], aplicações *Web* [van Beek and Mauw, 2004], sistemas concorrentes [Bi et al., 1998; Aichernig and Delgado, 2006; Xie and Dang, 2006], sistemas reativos [Andrade and Machado, 2012; Boroday et al., 2008; Bhateja, 2009; Jard and Jéron, 2005; Jeannet et al., 2007; Machado et al., 2007], sistemas orientados a componentes [Faivre et al., 2007; Frantzen and Tretmans, 2007], comunicação de processos [Lestiennes and Gaudel, 2002], entre outros.

Tipo de teste realizado

18,82% dos estudos tratam da forma de aplicação do teste. Foram identificadas três formas de aplicação: o teste síncrono, como descrito na teoria *ioco*, que considera a comunicação entre o testador e a implementação síncrona e simétrica [Tretmans, 2008; Rollet and Salva, 2009; Weiglhofer and Aichernig, 2010]; o teste assíncrono com a utilização de canais de comunicação, podendo ocorrer atrasos [Hierons, 2012b,c; Huo and Petrenko, 2005, 2004; Noroozi et al., 2011; Petrenko et al., 2003; Weiglhofer and Wotawa, 2009]; e o teste distribuído [Cavalcanti et al., 2011; Hierons, 2012a, 2008; Hierons et al., 2012b,a, 2011].

Requisitos de teste

Uma estratégia para definir quais propriedades ou funcionalidades serão testadas é utilizar propósitos de teste, como apresentado em 10,5% dos estudos [Aichernig and Delgado, 2006; Aichernig et al., 2007; Calamé et al., 2007; Simao and Petrenko, 2011; Clarke et al., 2001; Fraser et al., 2008; Gaston et al., 2006; Sampaio et al., 2009; Weiglhofer et al., 2009]. Um propósito de teste representa um conjunto finito de *traces* que de alguma forma estão relacionados com uma dada funcionalidade. Além disso, um propósito de teste pode ser considerado um cenário que deve ser seguido durante o teste. Porém, o resultado do teste, que é o veredito do teste, não é especificado [Simao and Petrenko, 2011]. Propósitos de teste podem ser baseados em algum critério de teste, como apresentado por Weiglhofer et al. [2009] em uma abordagem para geração de propósitos de teste baseados em critérios de cobertura para especificações *Lotos*.

Outra forma de definir as propriedades a serem testadas é cobrindo transições ou estados. Somente 3 estudos recentes consideram utilizar critérios de teste para cobertura de transições [Huo and Petrenko, 2005, 2004; Hierons, 2012b].

4.3 Métodos de geração

Dentre os métodos de geração de casos de teste a partir de IOTSs identificados, a maioria estende ou segue ideias do método proposto por Tretmans [2008], que definiu um algoritmo não-determinístico para geração de teste. Variações do algoritmo de geração de Tretmans [2008] foram propostas em 27,05% dos estudos considerando diferentes aspectos, como os tipos de dados envolvidos [Lestiennes and Gaudel, 2002], aspectos de robustez [Rollet and Salva,

2009; Fernandez et al., 2005], ou ainda consideradas subclasses de IOTS, tais como multi-IOTS [Brinksma et al., 1998; Hierons et al., 2008; Hierons, 2012a; Li et al., 2004a, 2003, 2004b], *timed* IOTS [Briones and Brinksma, 2005; Hessel et al., 2008; Schmaltz and Tretmans, 2008], e *symbolic* IOTS [Bannour et al., 2012; Clarke et al., 2001; Faivre et al., 2007; Frantzen et al., 2006; Frantzen and Tretmans, 2007; Frantzen et al., 2005; Gaston et al., 2006; Jeannet et al., 2007; Machado et al., 2007; Le Gall et al., 2007; Rusu et al., 2005].

O framework para teste de conformidade proposto por Tretmans [2008] explicita a necessidade de se ter uma relação de implementação para verificar a conformidade. Por meio do mapeamento sistemático, foi possível observar que esse framework é amplamente utilizado. Dentre os estudos selecionados, 39% utilizaram ou estenderam a relação *ioco* para diferentes contextos de teste. Dentre as relações de implementação propostas, as mais utilizadas são:

- *mioco (multi-ioco)* [Brinksma et al., 1998]: relação que estende a relação *ioco* para o contexto de múltiplos canais com o modelo MIOTS. Cada ação pertence a exatamente um canal de entrada ou um canal de saída.

- *sioco (symbolic-ioco)* [Frantzen et al., 2006]: muitas vezes, ações são parametrizadas com dados. Para evitar a explosão de estados durante a geração de teste, esses dados são tratados de forma simbólica, com o modelo IOSTS.

- *tioco (timed-ioco)* [Briones and Brinksma, 2005]: a questão da quietude é tratada com uma definição de tempo precisa, utilizando o modelo TIOTS.

- *c-dioco (controllable-distributed-ioco)* [Hierons et al., 2008]: no contexto de teste distribuído, é importante que os casos de teste gerados sejam controláveis: casos de teste onde o testador local saiba quando aplicar uma entrada. Desse modo, essa relação corresponde ao teste controlável em arquitetura de teste distribuída.

- *uioco (uncompleted-ioco)* [van der Bijl et al., 2005]: relação *ioco* adaptada para o contexto de especificações incompletas (*uncompleted specifications*).

- *cioco (component-ioco)* [Aiguier et al., 2012]: relação *ioco* no contexto de teste de componentes.

Apesar das interações entre caso de teste e testador serem síncronas no teste *ioco*, na prática muitas interações são baseadas em comunicação assíncrona ou via troca de mensagens com *buffers* que podem ser modelados como filas. Exemplos de sistemas que requerem tal tipo de comunicação são os protocolos de comunicação, sistemas de telecomunicação e *web services* [Simao and Petrenko, 2011; Hierons, 2012b,c]. Essa situação é comum em muitos sistemas que interagem por redes, onde pode haver atraso na comunicação. Isso introduz latência no teste, fazendo com que as sequências observadas não sejam necessariamente aquelas produzidas pela implementação. Uma noção alternativa da interação ser assíncrona é que a ferramenta de geração de testes pode não processar saídas tão rápido quanto elas são produzidas

pela implementação. Por isso, a maioria dos métodos de TBM não podem ser diretamente aplicados quando o testador e o sistema comunicam-se através de canais assíncronos [Hierons, 2012b; Simao and Petrenko, 2011].

Como filas tornam-se parte integrante na comunicação assíncrona com certos tipos de sistemas, elas devem ser consideradas na geração de teste. Por outro lado, a composição de filas pode trazer problemas como a explosão de estados, aumentando os custos e a complexidade dos testes. Nesse sentido, alguns estudos investigam como derivar casos de teste diretamente da especificação, evitando a composição de filas [Simao and Petrenko, 2011; Hierons, 2012c]. Para isso, foram utilizados critérios de teste amplamente conhecidos: alcançar todos os estados e executar todas as transições.

O trabalho de [Hierons, 2012b] investiga o problema de encontrar casos de teste que devem alcançar um dado objetivo. Esse trabalho trata do problema de produzir estratégias de teste, especificando qual entrada o testador deve fornecer e quando, para atingir três objetivos: alcançar estados, executar transições e distinguir dois estados em um IOTS. Deve-se observar que esses são passos básicos para a geração de teste.

Uma abordagem alternativa [Simao and Petrenko, 2011] propõe um método para geração de casos de teste evitando a composição com filas. A ideia é encontrar uma maneira de transformar um propósito de teste, levando em conta a distorção que as filas podem causar. Entretanto, não é considerado diretamente o problema de alcançar ou distinguir estados, apesar desse ser um problema relevante.

Outro trabalho [Hierons, 2012c] trata sobre a comunicação assíncrona por canais FIFO (*first-in-first-out*). Novas relações de implementação e equivalências foram definidas sem a observação da quietude. Foi mostrado que no contexto de teste assíncrono, é indecidível se existe um caso de teste que distingua dois modelos, e isso tem implicaçõs para a geração de testes baseada em domínios de defeitos. No entanto, para modelos alternados, como AIOTS (*Alternating IOTS*), esse problema é decidível e pode ser interessante estender isso para condições de testabilidade em sistemas que interagem com o ambiente via canais FIFO. Esse problema está relacionado com o trabalho de Noroozi et al. [2011], que investiga condições sob as quais um caso de teste para teste síncrono pode ser usado em teste assíncrono. Foi apresentado um conjunto de teoremas e provas que capacitam o uso do teste assíncrono da mesma forma que o teste *ioco*, a fim de testar implementações acessíveis apenas assincronamente. Desse modo, esses teoremas definem quando casos de testes síncronos são suficientes para verificar todos os aspectos de conformidade na interação assíncrona. Assim, o veredito para interação assíncrona coincide com o veredito para interação síncrona.

Além dos trabalhos que seguem a linha de Tretmans [2008], outras linhas foram identificadas:

- Métodos baseados na busca em profundidade (2,32%) [Koné and Castanet, 2000; Andrade and Machado, 2012], que geram dados de teste a partir do algoritmo de busca em profundidade.

- Métodos baseados em regras de inferência (4,75%) [Bi et al., 1998; Aiguier et al., 2012; Bannour et al., 2012; Gaston et al., 2006]: essa linha se destaca por utilizar uma generalização de formalismos, unificando em um mesmo framework os formalismos baseados em estados. A partir dessa generalização, é utilizado como algoritmo de geração um conjunto de regras de inferência, de modo que cada regra manipula uma observação ou um estímulo enviado ao sistema.

- Métodos que aplicam verificação de modelos para geração de casos de teste (3,5%) [Gromov and Willemse, 2007; Jéron and Morel, 1999; Aichernig and Delgado, 2006].

- Métodos que empregam modelos de falhas (7%) [Aichernig and Delgado, 2006; Aichernig et al., 2007, 2008; Fernandez et al., 2005; Gromov and Willemse, 2007; Petrenko et al., 2003].

4.4 Tipos de estudos experimentais

Conforme Do et al. [2005], é necessário um certo grau de julgamento subjetivo para determinar o tipo de estudo experimental realizado, já que a descrição e medidas quantitativas podem não ser claras. Entretanto, com base em diretrizes propostas previamente, Do et al. [2005] classificaram os estudos experimentais realizados na área de teste de software em: experimentos, estudos de caso e exemplos. Essa classificação foi utilizada para agrupar os estudos selecionados no mapeamento sistemático.

Os estudos foram analisados quanto à essa classificação:

- 67% dos estudos apresentaram apenas exemplos de aplicação da proposta [Bannour et al., 2012; van Beek and Mauw, 2004; Bensalem et al., 2008; Bhateja, 2011, 2009; Bi et al., 1998; Bourdonov et al., 2006; Brinksma et al., 1998; Briones and Brinksma, 2005; Cavalcanti et al., 2011; Chédor et al., 2012; Chen, 2011; Clarke et al., 2001; Desmoulin and Viho, 2009; Faivre et al., 2007; Falcone et al., 2012; Fernandez et al., 2005, 2004; Frantzen and Tretmans, 2007; Frantzen et al., 2005, 2006; Gaston et al., 2006; Gaudel, 2010; Gnesi et al., 2004; Gromov and Willemse, 2007; Hao and Wu, 1997; Hessel et al., 2008; Hierons, 2012a; Hierons et al., 2008, 2012b,a, 2011; Hierons and Núñez, 2012, 2010; Huo and Petrenko, 2005, 2004; Jeannet et al., 2007; Le Gall et al., 2007; Lestiennes and Gaudel, 2002; Li et al., 2004a, 2003, 2004b; Noroozi et al., 2011; Petrenko et al., 2003; Rollet and Saad-Khorchef, 2007; Rollet and Salva, 2009, 2008; Rusu et al., 2005; Sampaio et al., 2009; Schmaltz and Tretmans, 2008; Scollo and Zecchini, 2005; Tretmans, 2011, 2008; van der Bijl et al., 2005, 2004; Weiglhofer and Aichernig, 2010; Weiglhofer and Wotawa, 2009].

- 13% dos estudos relataram estudo de caso como forma de avaliar a viabilidade da proposta [Aichernig and Delgado, 2006; Aichernig et al., 2007, 2008; Andrade and Machado,

2012; Calamé et al., 2007; Fu and Kone, 2012; Jard and Jéron, 2005; Koné and Castanet, 2000; Muccini et al., 2004; Ruibing and Jianping, 1998; Willemse, 2007].

- 8,3% dos estudos relataram a execução de experimentos envolvendo a proposta [Brandl et al., 2010; Briones et al., 2006; Fraser et al., 2008; Kervinen and Virolainen, 2005; Simao and Petrenko, 2011; Weiglhofer et al., 2009; Xie and Dang, 2006].

- 11,7% dos estudos apenas discutiram a aplicação da abordagem, sendo alguns com contribuições teóricas [Aiguier et al., 2012; Bhateja et al., 2006; Boroday et al., 2008; Hierons, 2008, 2012b,c; Jéron and Morel, 1999; Machado et al., 2007; Nachmanson et al., 2004; Pardo et al., 2010].

Pode-se observar que um alto número de estudos foram classificados como *exemplo* (67%), e em seguida como *estudo de caso* (13%). Isso implica em um baixo número de *experimentos* (8,3%). Esse fato é um indicador de que deve ser realizada mais pesquisa empírica, principalmente experimentos, para confirmar os resultados teóricos das abordagens propostas para o teste formal envolvendo IOTSs.

4.5 Ferramentas

Várias ferramentas foram identificadas, utilizadas para a automatização do processo de geração. A seguir são descritas as ferramentas identificadas. Foi possível observar que a ferramenta TGV é bem estabelecida e amplamente utilizada não somente em pesquisa como também na indústria.

- *TGV* (*Test Generation with Verification technology*) [Jard and Jéron, 2005; Jéron and Morel, 1999; Aichernig et al., 2007; Clarke et al., 2001; Scollo and Zecchini, 2005; Calamé et al., 2007]: 6 estudos citam essa ferramenta, aplicada no contexto de protocolos de comunicação. Essa ferramenta é baseada na teoria de teste de conformidade com IOLTSs, estabelecendo importantes propriedades na geração de casos de teste. Foi provado em inúmeros estudos de casos que essa é uma ferramenta eficiente por utilizar a abordagem de teste *on-the-fly*, na qual permite a geração de casos de teste por uma exploração parcial de grafos de estados, evitando, portanto, o problema de explosão de estados [Jard and Jéron, 2005].

- *TorX* [Tretmans and Brinksma, 2003]: 1 estudo apresenta essa ferramenta, que é baseada na teoria de teste *ioco* [Tretmans, 2008] para definir corretude. Fornece geração de teste automática, implementação de teste (concretização dos casos de teste), execução de teste e análise de forma *on-the-fly*. Duas abordagens para seleção de testes foram adotadas: abordagem por propósito de teste (usuário que informa quais comportamentos devem ser testados) e abordagem heurística (baseadas em alguns pressupostos sobre o comportamento). Também foram realizados muitos estudos de caso na indústria mostrando sua eficiência no contexto de protocolos de comunicação.

- *UPPAAL* [Hessel et al., 2008]: 1 estudo cita essa ferramenta aplicada no contexto de teste de sistemas de tempo real, utilizando o modelo TIOTS. Por meio dessa ferramenta, é possível editar, simular e verificar propriedades do modelo de teste em um ambiente gráfico. A versão UPPAAL Cover implementa a geração de teste *off-line*, e a versão UPPAAL Tron implementa a geração de teste *on-line*. Na geração off-line, a geração de testes é solucionada como um problema de verificação de modelos. Já a geração *online* foi inspirada no algoritmo de Tretmans [Tretmans, 2008].

- *Diversity* [Bannour et al., 2012]: 1 estudo cita essa ferramenta, que implementa uma abordagem de geração de teste *off-line* utilizando o modelo TIOSTS, onde as propriedades temporais e de dados são expressas simbolicamente. A seleção de teste é baseada na cobertura das sequências de entrada. Dada a sequência de entrada, é obtida a saída do SUT, formando um *trace* completo.

- *LTS-BT* (*Labelled Transition System-Based Testing*) [Andrade and Machado, 2012]: ferramenta citada em 1 estudo. É seletora e geradora de casos de teste para aplicações embarcadas. Usa como entrada um modelo ALTS e gera casos de teste. A ferramenta também seleciona casos de teste de acordo com um propósito de teste e um critério de porcentagem de cobertura. Os casos de teste gerados são abstratos, e se desejado, a ferramenta mapeia o LTS para uma tabela, facilitando a execução manual.

- *TUGEN* [Bi et al., 1998]: ferramenta proposta para teste de protocolo (citada em 1 estudo), utilizando a notação C-TTCN (*Concurrent Tree Tabular Combined Notation*), que é uma extensão de LTS. O método de geração de teste tem duas fases: deriva sequências de teste a partir de um modelo de comportamento específico da abordagem e gera o conjunto de teste em C-TTCN.

- *AsmL* [Nachmanson et al., 2004]: ferramenta desenvolvida pela Microsoft e citada em 1 estudo. Tal ferramenta é aplicada para sistemas não-determinísticos, e a estratégia de geração de teste equivale a geração da estratégia de jogo [Alur et al., 1995], onde um de dois jogadores sempre deve vencer.

É interesssante notar que apenas 13,96% dos estudos citam o uso ou implementação de ferramentas. Isso mostra que muitos estudos preocupam-se apenas com contribuições teóricas para a área. Além disso, os estudos apontam a utilização de ferramentas já existentes: geradores de teste e propósitos de teste, e mesmo ferramentas que envolvem provas formais e verificação de modelos. Ainda foi notado que 8,3% dos estudos apresenta uma extensão de alguma ferramenta já existente. Isso evidencia a complexidade em construir uma ferramenta que apoia a geração de teste, levando à utilização e melhoria das ferramentas já existentes.

4.6 Autoria

Com relação aos autores dos estudos selecionados, a Tabela 4.1 apresenta os autores com maior número de estudos selecionados neste mapeamento. Os pesquisadores Robert M. Hierons e Jan Tretmans se destacam nesse grupo de autores, sendo o primeiro o autor dos trabalhos mais recentes e o segundo o autor do trabalho clássico do teste de conformidade a partir de IOTSs. Considerando o local de trabalho dos autores da Tabela 4.1, observa-se que os grupos de pesquisa na área de teste baseado em IOTSs concentram-se na Europa.

Tabela 4.1: Principais autores dos estudos selecionados

Autores	**Local**	**#**
Robert M. Hierons	Brunel University/Reino Unido	11
Jan Tretmans	Radboud University/Holanda	9
Thierry Jéron	IRISA (Institut de recherche en informatique et systèmes aléatoires)-INRIA/França	7
Manuel Núñez	Universidad Complutense de Madrid/Espanha	7
Martin Weiglhofer	Institute for Software Technology, Graz University of Technology/Áustria	7
Bernhard K. Aichernig	Institute for Software Technology, Graz University of Technology/Áustria	5
Alexandre Petrenko	Computer Research Institute of Montreal/-Canadá	5
Tim A. C. Willemse	Radboud University/Holanda	5
Franz Wotawa	Institute for Software Technology, Graz University of Technology/Áustria	5

4.7 Considerações finais

Este capítulo apresentou uma análise dos métodos identificados nos estudos selecionados por este mapeamento. Os estudos foram quantificados considerando-se os tipos de modelos utilizados, formas de aplicação dos testes, estudos empíricos e ferramentas utilizadas. Também foi discutida a autoria dos estudos e os principais grupos de pesquisa. O capítulo seguinte apresenta as conclusões e futuras direções deste estudo.

CAPÍTULO

5

Conclusões

Este trabalho apresentou os resultados de um mapeamento sistemático que identificou os métodos de geração de casos de teste para IOTSs. É importante ressaltar que não foi identificado nenhum estudo anterior que realizasse tal mapeamento. 85 estudos foram recuperados nos principais repositórios da área, evidenciando o interesse nesse tópico.

Retomando as questões de pesquisa dadas na Seção 1.1, a questão de pesquisa 1, relacionada aos métodos de geração de casos de teste para IOTSs, é respondida na Seção 4.3, nas quais são agrupados os métodos de geração dos estudos. A maioria dos métodos encontrados relata extensões da teoria de teste estabelecida por Tretmans [2008], que estabelece um framework para geração automatizada de casos de teste a partir de IOTSs.

A segunda questão de pesquisa, relacionada à uma possível taxonomia dos estudos analisados, é respondida na Seção 4.2, na qual foram identificados contextos de aplicação dos métodos. Dentre os métodos e ferramentas identificados, vários contextos de aplicação de teste são encontrados, tais como o tipo de sistema (sistemas de tempo-real, sistemas de protocolo de comunicação, aplicações *Web*), o tipo de teste realizado (teste síncrono, teste assíncrono com a utilização de canais de comunicação ou teste distribuído) e a forma de definir requisitos de teste (com propósitos de teste ou utilizando algum critério de cobertura). Além do clássico estudo de Tretmans [2008], recentemente os trabalhos de Hierons [Hierons et al., 2008, 2012b; Hierons, 2012c,b; Hierons and Núñez, 2012] têm se destacado.

Outra taxonomia é relacionada ao modelo aplicado pelos métodos de geração. Foram agrupados os estudos que aplicam os mesmos modelos na Seção 4.1. Certas classes do modelo IOTS foram identificadas. É importante compreender sob quais circunstâncias tais modelos são aplicados, uma vez que possuem diferentes características. Apesar disso, algumas definições acabam tornando-se similares por considerar ou não certas propriedades.

Quanto à terceira questão de pesquisa, sobre a avaliação das abordagens, os estudos foram agrupados de acordo com a classificação de Do et al. [2005] na Seção 4.4. Porém, observou-se que em um grande número de estudos não foi realizado nenhum experimento, e a maioria apresentou apenas exemplos de aplicação das abordagens.

A quarta questão de pesquisa foi respondida na Seção 4.6, nos quais são apresentados os principais autores e grupos de pesquisa na área de teste baseado em IOTSs. Notou-se grande presença de pesquisadores europeus. A quinta questão de pesquisa foi respondida na seção 4.3, nas quais Jan Tretmans é o mais referenciado por ser considerado o principal trabalho da área.

Portanto, existe um número considerável de estudos na área de teste formal baseado em IOTSs, e alguns pontos merecem atenção. A etapa de seleção de teste ainda é um desafio. Apesar do algoritmo original de Tretmans e suas variações considerarem que os conjuntos de casos de teste gerados são completos, tais algoritmos são não-determinísticos, não ficando claro como tais algoritmos podem satisfazer determinado critério de teste, por exemplo. Além disso, conceitos empregados no teste baseado em MEFs que não são utilizados por esses algoritmos, como condições de suficiência, domínios de defeitos, o uso de propriedades do modelo e geração de certas sequências podem ser fundamentais para gerar conjuntos de teste mais efetivos. Alguns estudos recentes apresentam interesse em utilizar tais conceitos para o teste baseado em IOTSs [Hierons, 2012b,c; Huo and Petrenko, 2005, 2004; Noroozi et al., 2011].

Notou-se que o teste assíncrono ainda é um desafio. Hierons [2012c] mostrou que é indecidível utilizar relações de implementação no teste assíncrono. Porém, ele mostrou que podem ser identificadas classes em que seja possível aplicar relação de implementação, como os *Alternating IOTSs*. Outra questão relacionada à geração de teste é condições de suficiência. Noroozi et al. [2011] apresentou algumas condições onde o veredito do teste assíncrono coincide com o teste síncrono.

Mesmo com os resultados encontrados, este trabalho possui limitações. O protocolo estabelecido definiu 8 engenhos de busca para recuperação dos estudos. Muitos estudos foram retornados, mas a maioria não era relevante. Como muitos resumos não continham informações sobre os modelos empregados, é possível que outros estudos não foram selecionados por esse motivo.

Como direções futuras, duplicações deste estudo podem aumentar o número de fontes para incluir mais evidências, além de melhorar o protocolo e a *string* de busca, de modo que o número de estudos retornados seja mais próximo do número de estudos relevantes. Além disso, este estudo de mapeamento pode ser utilizado como ponto de partida para evoluções na área de teste.

Agradecimentos

Esta pesquisa é apoiada financeiramente pela FAPESP (processo 2012/09650-5).

Referências Bibliográficas

Aichernig, B. and Delgado, C. (2006). From Faults via Test Purposes to Test Cases: On the Fault-Based Testing of Concurrent Systems. In Baresi, L. and Heckel, R., editors, *Fundamental Approaches to Software Engineering*, volume 3922 of *Lecture Notes in Computer Science*, pages 324–338. Springer Berlin / Heidelberg.

Aichernig, B., Peischl, B., Weiglhofer, M., and Wotawa, F. (2007). Protocol Conformance Testing a SIP Registrar: an Industrial Application of Formal Methods. In *Software Engineering and Formal Methods, 2007. SEFM 2007. Fifth IEEE International Conference on*, pages 215–226. IEEE Computer Society.

Aichernig, B. K., Weiglhofer, M., and Wotawa, F. (2008). Improving Fault-based Conformance Testing. *Electronic Notes in Theoretical Computer Science*, 220(1):63–77. Proceedings of the Fourth Workshop on Model Based Testing (MBT 2008).

Aiguier, M., Boulanger, F., and Kanso, B. (2012). A Formal Abstract Framework for Modelling and Testing Complex Software Systems. *Theoretical Computer Science*, 455(0):66–97. International Colloquium on Theoretical Aspects of Computing 2010.

Alur, R., Courcoubetis, C., and Yannakakis, M. (1995). Distinguishing Tests for Nondeterministic and Probabilistic Machines. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, volume STOC '95, pages 363–372, New York, NY, USA. ACM.

Andrade, W. and Machado, P. (2012). Testing Interruptions in Reactive Systems. *Formal Aspects of Computing*, 24(3):331–353.

Bannour, B., Escobedo, J., Gaston, C., and Le Gall, P. (2012). Off-Line Test Case Generation for Timed Symbolic Model-Based Conformance Testing. In Nielsen, B. and Weise, C., editors, *Testing Software and Systems*, volume 0 of *Lecture Notes in Computer Science*, pages 119–135. Springer Berlin / Heidelberg.

Barbosa, L. S. (2003). Towards a Calculus of State-based Software Components. *J.U.C.S - Journal of Universal Computer Science*, 9(8):891–909.

Bensalem, S., Krichen, M., and Tripakis, S. (2008). State Identification Problems for Input/Output Transition Systems. In *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on*, pages 225–230. IEEE Computer Society.

Bhateja, P. (2009). Grammar Based Asynchronous Testing. In *Proceedings of the 2nd India software engineering conference*, ISEC '09, pages 105–110, New York, NY, USA. ACM.

Bhateja, P. (2011). Test Case Generation using PDA. In *Proceedings - 5th International Conference on Theoretical Aspects of Software Engineering, TASE 2011*, pages 221–224.

Bhateja, P., Gastin, P., and Mukund, M. (2006). A Fresh Look at Testing for Asynchronous Communication. In Graf, S. and Zhang, W., editors, *Automated Technology for Verification and Analysis*, volume 4218 of *Lecture Notes in Computer Science*, pages 369–383. Springer Berlin / Heidelberg.

Bi, J., Wu, J., and Chen, X. (1998). A Concurrent TTCN based Approach to Conformance Testing of Distributed Routing Protocol OSPF v2. In *Computer Communications and Networks, 1998. Proceedings. 7th International Conference on*, pages 760–767.

Boroday, S., Petrenko, A., and Ulrich, A. (2008). Test Suite Consistency Verification. In *Design Test Symposium (EWDTS), 2008 East-West*, pages 235–239.

Bourdonov, I. B., Kossatchev, A. S., and Kuliamin, V. V. (2006). Formal Conformance Testing of Systems with Refused Inputs and Forbidden Actions. *Electronic Notes in Theoretical Computer Science*, 164(4):83–96. Proceedings of the Second Workshop on Model Based Testing (MBT 2006) – Second Workshop on Model Based Testing 2006.

Brandl, H., Weiglhofer, M., and Aichernig, B. (2010). Automated Conformance Verification of Hybrid Systems. In *Quality Software (QSIC), 2010 10th International Conference on*, pages 3–12.

Brinksma, E., Heerink, L., and Tretmans, J. (1998). Factorized Test Generation for Multi-Input/Output Transition Systems. In Petrenko, A and Yevtushenko, N, editor, *TESTING OF COMMUNICATING SYSTEMS*, pages 67–82. 11th International Workshop on Testing of Communicating Systems (IWTCS 98), TOMSK, RUSSIA, AUG 31-SEP 02, 1998.

Briones, L. and Brinksma, E. (2005). A Test Generation Framework for Quiescent Real-Time Systems. In Grabowski, J. and Nielsen, B., editors, *Formal Approaches to Software Testing*, volume 3395 of *Lecture Notes in Computer Science*, pages 64–78. Springer Berlin / Heidelberg.

Briones, L., Brinksma, E., and Stoelinga, M. (2006). A Semantic Framework for Test Coverage. In Graf, S. and Zhang, W., editors, *Automated Technology for Verification and Analysis*,

volume 4218 of *Lecture Notes in Computer Science*, pages 399–414. Springer Berlin / Heidelberg.

Calamé, J. R., Ioustinova, N., and van de Pol, J. (2007). Automatic Model-Based Generation of Parameterized Test Cases Using Data Abstraction. *Electronic Notes in Theoretical Computer Science*, 191(0):25–48. Proceedings of the Doctoral Symposium affiliated with the Fifth Integrated Formal Methods Conference (IFM 2005).

Cavalcanti, A., Gaudel, M.-C., and Hierons, R. (2011). Conformance Relations for Distributed Testing Based on CSP. In Wolff, B. and Zaïdi, F., editors, *Testing Software and Systems*, volume 7019 of *Lecture Notes in Computer Science*, pages 48–63. Springer Berlin / Heidelberg.

Chen, L. (2011). Automatic Test Cases Generation for Statechart Specifications from Semantics to Algorithm. *Journal of Computers*, 6(4):769–775.

Chédor, S., Jéron, T., and Morvan, C. (2012). Test Generation from Recursive Tiles Systems. In Brucker, A. and Julliand, J., editors, *Tests and Proofs*, volume 7305 of *Lecture Notes in Computer Science*, pages 99–114. Springer Berlin / Heidelberg.

Clarke, D., Jéron, T., Rusu, V., and Zinovieva, E. (2001). Automated Test and Oracle Generation for Smart-Card Applications. In Attali, I. and Jensen, T., editors, *Smart Card Programming and Security*, volume 2140 of *Lecture Notes in Computer Science*, pages 58–70. Springer Berlin / Heidelberg.

Costa, S. L. and Pereira, V. (2013). Revisão Sistemática: Ferramentas de Apoio do Método B e da Notação Z. In *Memorias del X Workshop Latinoamericano Ingeniería de Software Experimental*, ESELAW - 2013, pages 34–47.

Desmoulin, A. and Viho, C. (2009). Formalizing Interoperability for Test Case Generation Purpose. *International Journal on Software Tools for Technology Transfer (STTT)*, 11(3):261–267.

Do, H., Elbaum, S., and Rothermel, G. (2005). Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact. *Empirical Softw. Engg.*, 10(4):405–435.

Faivre, A., Gaston, C., and Le Gall, P. (2007). Symbolic Model Based Testing for Component Oriented Systems. In Petrenko, A., Veanes, M., Tretmans, J., and Grieskamp, W., editors, *Testing of Software and Communicating Systems*, volume 4581 of *Lecture Notes in Computer Science*, pages 90–106. Springer Berlin / Heidelberg.

Falcone, Y., Fernandez, J.-C., Jéron, T., Marchand, H., and Mounier, L. (2012). More Testable Properties. *International Journal on Software Tools for Technology Transfer (STTT)*, 14(4):407–437.

Fernandez, J.-C., Mounier, L., and Pachon, C. (2004). Property Oriented Test Case Generation. In Petrenko, A. and Ulrich, A., editors, *Formal Approaches to Software Testing*, volume 2931 of *Lecture Notes in Computer Science*, pages 1101–1102. Springer Berlin / Heidelberg.

Fernandez, J.-C., Mounier, L., and Pachon, C. (2005). A Model-Based Approach for Robustness Testing. In Khendek, F. and Dssouli, R., editors, *Testing of Communicating Systems*, volume 3502 of *Lecture Notes in Computer Science*, pages 313–313. Springer Berlin / Heidelberg.

Frantzen, L. and Tretmans, J. (2007). Model-Based Testing of Environmental Conformance of Components. In de Boer, F., Bonsangue, M., Graf, S., and de Roever, W.-P., editors, *Formal Methods for Components and Objects*, volume 4709 of *Lecture Notes in Computer Science*, pages 1–25. Springer Berlin / Heidelberg.

Frantzen, L., Tretmans, J., and Willemse, T. (2005). Test Generation Based on Symbolic Specifications. In Grabowski, J. and Nielsen, B., editors, *Formal Approaches to Software Testing*, volume 3395 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg.

Frantzen, L., Tretmans, J., and Willemse, T. (2006). A Symbolic Framework for Model-Based Testing. In Havelund, K., Núñez, M., Rosu, G., and Wolff, B., editors, *Formal Approaches to Software Testing and Runtime Verification*, volume 4262 of *Lecture Notes in Computer Science*, pages 40–54. Springer Berlin / Heidelberg.

Fraser, G., Weiglhofer, M., and Wotawa, F. (2008). Coverage Based Testing with Test Purposes. In *Proceedings of International Conference on Quality Software*, pages 199–208.

Fu, Y. and Kone, O. (2012). A Robustness Testing Method for Network Security. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, 99 LNICST:38–45.

Gaston, C., Le Gall, P., Rapin, N., and Touil, A. (2006). Symbolic Execution Techniques for Test Purpose Definition. In Uyar, M., Duale, A., and Fecko, M., editors, *Testing of Communicating Systems*, volume 3964 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin / Heidelberg.

Gaudel, M.-C. (2010). Software Testing Based on Formal Specification. In Borba, P., Cavalcanti, A., Sampaio, A., and Woodcook, J., editors, *Testing Techniques in Software Engineering*, volume 6153 of *Lecture Notes in Computer Science*, pages 215–242. Springer Berlin / Heidelberg.

Gnesi, S., Latella, D., and Massink, M. (2004). Formal Test-Case Generation for UML Statecharts. In *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, pages 75–84.

Gromov, M. and Willemse, T. (2007). Testing and Model-Checking Techniques for Diagnosis. In Petrenko, A., Veanes, M., Tretmans, J., and Grieskamp, W., editors, *Testing of Software and Communicating Systems*, volume 4581 of *Lecture Notes in Computer Science*, pages 138–154. Springer Berlin / Heidelberg.

Hansen, H. H., Costa, D., and Rutten, J. (2006). Synthesis of mealy machines using derivatives. *Electronic Notes in Theoretical Computer Science*, 164(1):27 – 45. Proceedings of the Eighth Workshop on Coalgebraic Methods in Computer Science (CMCS 2006) - Eighth Workshop on Coalgebraic Methods in Computer Science.

Hao, R. and Wu, J. (1997). Toward formal ttcn-based test execution. In *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE*, volume 1, pages 230–235.

Heerink, L. and Tretmans, J. (1998). Refusal Testing for Classes of Transition Systems with Inputs and Outputs. In *Proceedings of the IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE X) and Protocol Specification, Testing and Verification (PSTV XVII)*, FORTE X / PSTV XVII '97, pages 23–38, London, UK, UK. Chapman & Hall, Ltd.

Hessel, A., Larsen, K., Mikucionis, M., Nielsen, B., Pettersson, P., and Skou, A. (2008). Testing Real-Time Systems Using UPPAAL. In Hierons, R., Bowen, J., and Harman, M., editors, *Formal Methods and Testing*, volume 4949 of *Lecture Notes in Computer Science*, pages 77–117. Springer Berlin / Heidelberg.

Hierons, R. (2008). Testing in the Distributed Test Architecture: An Extended Abstract. In *Quality Software, 2008. QSIC '08. The Eighth International Conference on*, pages 11–14.

Hierons, R. (2012a). Overcoming Controllability Problems in Distributed Testing from an Input Output Transition System. *Distributed Computing*, 25(1):63–81.

Hierons, R., Merayo, M., and Nunez, M. (2008). Controllable Test Cases for the Distributed Test Architecture. In Cha, S., Choi, J.-Y., Kim, M., Lee, I., and Viswanathan, M., editors, *Automated Technology for Verification and Analysis*, volume 5311 of *Lecture Notes in Computer Science*, pages 201–215. Springer Berlin / Heidelberg.

Hierons, R., Merayo, M., and Núñez, M. (2011). Scenarios-Based Testing of Systems with Distributed Ports. *Software - Practice and Experience*, 41(10):999–1026.

Hierons, R., Merayo, M., and Núñez, M. (2012a). Implementation Relations and Test Generation for Systems with Distributed Interfaces. *Distributed Computing*, 25(1):35–62.

Hierons, R., Merayo, M., and Núñez, M. (2012b). Using Time to Add Order to Distributed Testing. In Giannakopoulou, D. and Méry, D., editors, *FM 2012: Formal Methods*, volume 7436 of *Lecture Notes in Computer Science*, pages 232–246. Springer Berlin / Heidelberg.

Hierons, R. and Núñez, M. (2010). Testing Probabilistic Distributed Systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6117:63–77.

Hierons, R. and Núñez, M. (2012). Using Schedulers to Test Probabilistic Distributed Systems. *Formal Aspects of Computing*, 24(4):679–699.

Hierons, R. M. (2012b). The Complexity of Asynchronous Model Based Testing. *Theoretical Computer Science*, 451(0):70–82.

Hierons, R. M. (2012c). Implementation Relations for Testing Through Asynchronous Channels. *The Computer Journal*, page bxs107.

Hierons, R. M., Bogdanov, K., Bowen, J. P., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Kapoor, K., Krause, P., Lüttgen, G., Simons, A. J. H., Vilkomir, S., Woodward, M. R., and Zedan, H. (2009). Using Formal Specifications to Support Testing. *ACM Comput. Surv.*, 41(2):9:1–9:76.

Huo, J. and Petrenko, A. (2005). Covering Transitions of Concurrent Systems through Queues. In *Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on*, pages 334–345. IEEE.

Huo, J. and Petrenko, A. (2009). Transition Covering Tests for Systems with Queues. *Softw. Test. Verif. Reliab.*, 19(1):55–83.

Huo, J. L. and Petrenko, A. (2004). On Testing Partially Specified IOTS through Lossless Queues. In Groz, R. and Hierons, R., editors, *Testing of Communicating Systems*, volume 2978 of *Lecture Notes in Computer Science*, pages 2707–2707. Springer Berlin / Heidelberg.

Jard, C. and Jéron, T. (2005). TGV: Theory, Principles and Algorithms. *International Journal on Software Tools for Technology Transfer (STTT)*, 7(4):297–315.

Jeannet, B., Jéron, T., and Rusu, V. (2007). Model-Based Test Selection for Infinite-State Reactive Systems. In de Boer, F., Bonsangue, M., Graf, S., and de Roever, W.-P., editors, *Formal Methods for Components and Objects*, volume 4709 of *Lecture Notes in Computer Science*, pages 47–69. Springer Berlin / Heidelberg.

Jéron, T. and Morel, P. (1999). Test Generation Derived from Model-Checking. In Halbwachs, N. and Peled, D., editors, *Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 108–122. Springer Berlin / Heidelberg.

Kervinen, A. and Virolainen, P. (2005). Heuristics for Faster Error Detection with Automated Black Box Testing. *Electronic Notes in Theoretical Computer Science*, 111(0):53–71. Proceedings of the Workshop on Model Based Testing (MBT 2004 – Workshop on Model Based Testing 2004.

Kitchenham, B. (2004). Procedures for Performing Systematic Reviews. Technical report tr/se-0401, Keele University and NICTA.

Koné, O. and Castanet, R. (2000). Test Generation for Interworking Systems. *Computer Communications*, 23(7):642–652.

Krichen, M. (2010). A Formal Framework for Conformance Testing of Distributed Real-Time Systems. In *Proceedings of the 14th international conference on Principles of distributed systems*, OPODIS'10, pages 139–142, Berlin, Heidelberg. Springer-Verlag.

Le Gall, P., Rapin, N., and Touil, A. (2007). Symbolic Execution Techniques for Refinement Testing. In Gurevich, Y. and Meyer, B., editors, *Tests and Proofs*, volume 4454 of *Lecture Notes in Computer Science*, pages 131–148. Springer Berlin / Heidelberg.

Lestiennes, G. and Gaudel, M.-C. (2002). Testing Processes from Formal Specifications with Inputs, Outputs and Data Types. In *Software Reliability Engineering, 2002. ISSRE 2003. Proceedings. 13th International Symposium on*, pages 3–14.

Li, Z., Wu, J., and Yin, X. (2003). Refusal Testing for MIOTS with Nonlockable Output Channels. In *Computer Networks and Mobile Computing, 2003. ICCNMC 2003. 2003 International Conference on*, pages 517–522. IEEE Computer Society.

Li, Z., Wu, J., and Yin, X. (2004a). Testing Multi Input/Output Transition System with All-Observer. In Groz, R. and Hierons, R., editors, *Testing of Communicating Systems*, volume 2978 of *Lecture Notes in Computer Science*, pages 2705–2705. Springer Berlin / Heidelberg.

Li, Z., Yin, X., and Wu, J. (2004b). Distributed Testing of Multi Input/Output Transition System. In *Proceedings of the Second International Conference on Software Engineering and Formal Methods. SEFM 2004*, pages 271–280.

Machado, P. D., Silva, D. A., and Mota, A. C. (2007). Towards Property Oriented Testing. *Electronic Notes in Theoretical Computer Science*, 184(0):3–19. Proceedings of the Second Brazilian Symposium on Formal Methods (SBMF 2005).

Muccini, H., Bertolino, A., and Inverardi, P. (2004). Using Software Architecture for Code Testing. *IEEE Transactions on Software Engineering*, 30(3):160–171.

Nachmanson, L., Veanes, M., Schulte, W., Tillmann, N., and Grieskamp, W. (2004). Optimal Strategies for Testing Nondeterministic Systems. In *Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis*, ISSTA '04, pages 55—64, New York, NY, USA. ACM.

Noroozi, N., Khosravi, R., Mousavi, M., and Willemse, T. (2011). Synchronizing Asynchronous Conformance Testing. In Barthe, G., Pardo, A., and Schneider, G., editors, *Software Engineering and Formal Methods*, volume 7041 of *Lecture Notes in Computer Science*, pages 334–349. Springer Berlin / Heidelberg.

Pardo, J., Núñez, M., and Ruiz, M. (2010). Specification and Testing of E-Commerce Agents Described by Using UIOLTSs. In Hatcliff, J. and Zucca, E., editors, *Formal Techniques for Distributed Systems*, volume 6117 of *Lecture Notes in Computer Science*, pages 78–86. Springer Berlin / Heidelberg.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic Mapping Studies in Software Engineering. In *Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77, Swinton, UK. British Computer Society.

Petrenko, A. and Yevtushenko, N. (2002). Queued Testing of Transition Systems with Inputs and Outputs. In *Proceedings of the Workshop on Formal Approaches to Testing of Software, FATES'02, A Satellite Workshop of CONCUR'02. Czech Republic.*

Petrenko, A., Yevtushenko, N., and Huo, J. (2003). Testing Transition Systems with Input and Output Testers. In Hogrefe, D. and Wiles, A., editors, *Testing of Communicating Systems*, volume 2644 of *Lecture Notes in Computer Science*, pages 609–609. Springer Berlin / Heidelberg.

Rollet, A. and Saad-Khorchef, F. (2007). A Formal Approach to Test the Robustness of Embedded Systems using Behaviour Analysis. In *Software Engineering Research, Management Applications, 2007. SERA 2007. 5th ACIS International Conference on*, pages 667–674. IEEE.

Rollet, A. and Salva, S. (2008). Two Complementary Approaches to Test Robustness of Reactive Systems. In *Automation, Quality and Testing, Robotics, 2008. AQTR 2008. IEEE International Conference on*, volume 1, pages 47–53. IEEE Computer Society.

Rollet, A. and Salva, S. (2009). Testing Robustness of Communicating Systems using ioco-based Approach. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 67–72. IEEE Computer Society.

Ruibing, H. and Jianping, W. (1998). A Formal Approach to Protocol Interoperability Testing. *Journal of Computer Science and Technology*, 13(1):79–90.

Rusu, V., Marchand, H., and Jéron, T. (2005). Automatic Verification and Conformance Testing for Validating Safety Properties of Reactive Systems. In Fitzgerald, J., Hayes, I., and Tarlecki, A., editors, *FM 2005: Formal Methods*, volume 3582 of *Lecture Notes in Computer Science*, pages 189–204. Springer Berlin Heidelberg.

Sampaio, A., Nogueira, S., and Mota, A. (2009). Compositional verification of Input-Output Conformance via CSP Refinement Checking. In Breitman, K. and Cavalcanti, A., editors, *Formal Methods and Software Engineering*, volume 5885 of *Lecture Notes in Computer Science*, pages 20–48. Springer Berlin / Heidelberg.

Schmaltz, J. and Tretmans, J. (2008). On Conformance Testing for Timed Systems. In Cassez, F. and Jard, C., editors, *Formal Modeling and Analysis of Timed Systems*, volume 5215 of *Lecture Notes in Computer Science*, pages 250–264. Springer Berlin / Heidelberg.

Scollo, G. and Zecchini, S. (2005). Architectural Unit Testing. *Electronic Notes in Theoretical Computer Science*, 111(0):27–52. Proceedings of the Workshop on Model Based Testing (MBT 2004) – Workshop on Model Based Testing 2004.

Simao, A. and Petrenko, A. (2011). Generating Asynchronous Test Cases from Test Purposes. *Information and Software Technology*, 53(11):1252–1262. AMOST 2010.

Simão, A. (2007). *Introdução ao Teste de Software*, chapter Teste Baseado em Modelos, pages 27–45. Elsevier.

Tretmans, G. and Brinksma, H. (2003). TorX: Automated Model-Based Testing. In *First European Conference on Model-Driven Software Engineering*, pages 31–43.

Tretmans, J. (1996). Test Generation with Inputs, Outputs and Repetitive Quiescence. *Software - Concepts and Tools*, 17(3):103–120.

Tretmans, J. (2008). Model Based Testing with Labelled Transition Systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4949 LNCS:1–38.

Tretmans, J. (2011). Model-Based Testing and Some Steps towards Test-Based Modelling. In Bernardo, M. and Issarny, V., editors, *Formal Methods for Eternal Networked Software Systems*, volume 6659 of *Lecture Notes in Computer Science*, pages 297–326. Springer Berlin / Heidelberg.

Utting, B. L. M. (2010). Model-Based Testing – Next Generation Functional Software Testing. *Journal of Software Technology*, 12(4).

van Beek, H. and Mauw, S. (2004). Automatic Conformance Testing of Internet Applications. In Petrenko, A. and Ulrich, A., editors, *Formal Approaches to Software Testing*, volume 2931 of *Lecture Notes in Computer Science*, pages 1106–1106. Springer Berlin / Heidelberg.

van der Bijl, M., Rensink, A., and Tretmans, J. (2004). Compositional Testing with ioco. In Petrenko, A. and Ulrich, A., editors, *Formal Approaches to Software Testing*, volume 2931 of *Lecture Notes in Computer Science*, pages 1102–1102. Springer Berlin / Heidelberg.

van der Bijl, M., Rensink, A., and Tretmans, J. (2005). Action Refinement in Conformance Testing. In Khendek, F. and Dssouli, R., editors, *Testing of Communicating Systems*, volume 3502 of *Lecture Notes in Computer Science*, pages 363–363. Springer Berlin / Heidelberg.

Weiglhofer, M. and Aichernig, B. (2010). Unifying Input Output Conformance. In Butterfield, A., editor, *Unifying Theories of Programming*, volume 5713 of *Lecture Notes in Computer Science*, pages 181–201. Springer Berlin / Heidelberg.

Weiglhofer, M., Fraser, G., and Wotawa, F. (2009). Using Coverage to Automate and Improve Test Purpose Based Testing. *Information and Software Technology*, 51(11):1601–1617. Third IEEE International Workshop on Automation of Software Test (AST 2008) – Eighth International Conference on Quality Software (QSIC 2008).

Weiglhofer, M. and Wotawa, F. (2009). Asynchronous Input-Output Conformance Testing. In *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*, volume 1, pages 154–159.

Willemse, T. (2007). Heuristics for ioco-Based Test-Based Modelling. In Brim, L., Haverkort, B., Leucker, M., and van de Pol, J., editors, *Formal Methods: Applications and Technology*, volume 4346 of *Lecture Notes in Computer Science*, pages 132–147. Springer Berlin / Heidelberg.

Xie, G. and Dang, Z. (2006). Testing Systems of Concurrent Black-Boxes – An Automata-Theoretic and Decompositional Approach. In Grieskamp, W. and Weise, C., editors, *Formal Approaches to Software Testing*, volume 3997 of *Lecture Notes in Computer Science*, pages 170–186. Springer Berlin / Heidelberg.