Uso de *Constraint Solvers* na Geração de Dados de Teste: Um Mapeamento Sistemático da Literatura

Trabalho de Conclusão de Curso

Engenharia da Computação

Bruno Fonseca Lins de Oliveira

Orientador: Gustavo Carvalho

Bruno Fonseca Lins de Oliveira

Uso de Constraint Solvers na Geração de Dados de Teste: Um Mapeamento Sistemático da Literatura

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco - Universidade de Pernambuco

Orientador:
Gustavo Carvalho

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

Recife - PE, Brasil

23 de março de 2012

Declaro que revisei o Trabalho de Conclusão de Curso sob o título *"Uso de Constraint Solvers na Geração de Dados de Teste: Um Mapeamento Sistemático da Literatura"*, de autoria de *Bruno Fonseca Lins de Oliveira*, e que estou de acordo com a entrega do mesmo.

Recife, _____ / _____ / _____

Gustavo Carvalho
Orientador

Agradecimentos

A todos os professores pelo conhecimento transmitido, principalmente ao meu orientador, Gustavo Carvalho, pela oportunidade, paciência, atenção e pelo apoio dado.

A família e amigos, que estiveram do meu lado durante todo o tempo, torcendo por mim, me apoiando e não me deixando desistir mesmo nos momentos de dificuldade.

A todos os meus colegas de graduação, com certeza sem o apoio deles ao longo desses 5 anos eu não teria chegado até aqui.

Resumo

Nos últimos tempos ferramentas comerciais utilizando *Constraint Solvers* têm sido adotadas por empresas para geração automática de dados de testes. Este trabalho tem como objetivo principal, sumarizar evidências existentes a fim de prover um melhor entendimento sobre os detalhes de como dados de teste são gerados automaticamente a partir do uso de *Constraint Solvers*. Mais especificamente, este trabalho também tem os seguintes objetivos

- Mapear as técnicas mais utilizadas;
- Identificar qual o tipo de *Solver* mais utilizado;
- Analisar a escalabilidade para problemas reais;
- Apontar benefícios e limitações.

Palavras-chave: *Constraint Solvers*, *Constraint Language Programming*, *Test Data*, *Test Vector*.

Abstract

In recent times commercial tools using Constraint Solvers have been adopted by companies for automatic generation of test data. This work has as main objective summarize existing evidence in order to provide a better understanding of the details of how test data are generated from the use of Constraint Solvers. More speciafically, this work also has the following objectives:

- Map the most widely used techniques;
- Identify the most widely used type of solver;
- Analyze the scalability to real problems;
- Point out the benefits and limitations.

Keywords: Constraint Solvers, Constraint Language Programming,Test Data, Test Vector

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CSP	*Constraint Satisfication Problem*
COP	*Constraint optimization problems*
ISSTA	*International Symposium on Software Testing and Analysis*
SAT	*Satisfiability*
SMT	*Satisfiability Modulo Theories*
SUT	*System Under Test*

1 Introdução

O processo de teste exerce um importante papel na garantia da qualidade de software. Durante o processo de desenvolvimento de software, diversos erros e falhas podem aparecer e o objetivo dos testes é capturar a maior quantidade de defeitos possível através da aplicação de um conjunto de atividades apuradas e bem planejadas. Essas atividades têm como objetivo verificar se todos os requisitos do sistema foram corretamente implementados, assegurar a qualidade e corretude do software produzido e reduzir custos de manutenção corretiva (MYERS, 1979).

Testar, porém, custa tempo e dinheiro. O tempo é um inimigo diário nas grandes empresas. O desafio de desenvolver software com qualidade se depara constantemente com o fator tempo. Nesse sentido, as empresas buscam alternativas de automação de testes com o intuito de assegurar a qualidade de seus produtos, mas ao mesmo tempo consumindo menos recursos na atividade de testes.

Mais especificamente, atualmente, empresas empregam uma grande quantidade de esforço e recursos durante a geração de dados de teste, pois estes são geralmente escritos manualmente, levando a erros, falhas e redundâncias. A geração automática de dados de teste surge como abordagem promissora para minimizar o esforço na criação de casos de teste, resultando em um maior controle da qualidade e na redução de custos inerentes ao processo.

Nos últimos tempos, ferramentas comerciais, utilizando *Constraint Solvers*, como: RT-Tester [1], T-Vec [2], SCR Toolset [3], entre outras, têm sido adotadas por empresas para geração automática de dados de teste. A ideia por trás da utilização de *Constraint Solvers* é resolver problemas, simplesmente afirmando restrições (*constraints*) as quais devem ser satisfeitas por uma solução do problema.

[1] http://www.verified.de/en/products/rt-tester
[2] http://www.t-vec.com/
[3] https://www.nrl.navy.mil/chacs/5546/scr_toolset/index.php

1.1 Qualificação do Problema

Como visto anteriormente, nos últimos tempos ferramentas comerciais utilizando *Constraint Solvers* têm sido adotadas por empresas para geração automática de dados de testes. Logo, é importante, saber quais os benefícios e limitações dessa técnica, o que já foi feito, e o que pode ou precisa ser melhorado.

Portanto, esse trabalho tem como problema de pesquisa: como Constraint Solvers têm sido utilizados para gerar automaticamente dados de teste? Para responder este questionamento e obter um melhor entendimento da área, propõe-se a realização de um mapeamento sistemático da literatura.

1.2 Objetivos

Este trabalho tem como principal objetivo sumarizar evidências a fim de prover um melhor entendimento sobre os detalhes de como dados de teste são gerados automaticamente a partir do uso de *Contraint Solvers*.

1.2.1 Objetivos Específicos

Mais especificamente, este trabalho também tem os seguintes objetivos:

- Mapear as técnicas mais utilizadas;
- Identificar qual o tipo de *Constraint Solver* mais utilizado;
- Analisar a escalabilidade para problemas reais;
- Apontar benefícios e limitações da técnica.

1.3 Resultados e Impactos Esperados

Este trabalho terá como principal resultado um mapeamento de *Constraint Solvers* utilizados na geração automática de dados de testes para os domínios nos quais essa abordagem tem sido utilizada.

Este mapeamento fornecerá um arcabouço para posicionar novas pesquisas, além de apoiar a geração de novas hipóteses de pesquisa. Em particular, será possível posicionar

novas pesquisas em ralação as limitações e desafios existentes atualmente no emprego desta técnica de testes de software e sistemas.

1.4 Estrutura da Monografia

Este trabalho está dividido em cinco capítulos, incluindo este capítulo introdutório. A seguir tem-se uma breve descriçao do conteúdo de cada um destes:

- O Capítulo 2 apresenta o referencial teórico, e os principais conceitos que são necessários para a compreensão deste trabalho.
- O Capítulo 3 descreve a metodologia utilizada para realização do estudo, as principais etapas da pesquisa e o processo para a realização do mapeamento sistemático da literatura, a partir da definição de um protocolo. Para finalizar, é descrito como é feita a análise dos dados extraídos e sintetize num mapeamento.
- No Capítulo 4 apresentam-se os resultados, sendo analisados inicialmente os dados quantitativos do mapeamento como principais fontes, quantidade de estudos retornados, avaliação da qualidade dos mesmos. Em seguida, é feita uma análise qualitativa das evidências com o objetivo de responder a questão de pesquisa.
- No Capítulo 5 é apresentada a conclusão do trabalho através dos dados obtidos a partir do mapeamento sistemático sobre o tema da pesquisa.

2 *Constraint Programming*

Constraint programming(ou Propagação por Restrições) é um poderoso paradigma para resolver problemas de busca combinatória, que se baseia em uma ampla gama de técnicas de inteligência artificial, ciência da computação, linguagens de programação e pesquisa operacional (ROSSI; BEEK; WALSH, 2006). *Constraint programming* atualmente é aplicada em muitos domínios, como análise de circuitos elétricos, planejamento financeiro, geração de dados de teste, etc.

É um paradigma de programação em que as relações entre as variáveis pode ser expressa na forma de restrições. Restrições diferem das primitivas comuns de outras linguagens de programação na medida em que não especificam um passo ou a sequência de passos para executar, mas sim as propriedades de uma solução a ser encontrada. *Constraint programming* pode ser utilizada pra resolver uma grande variedade de problemas de forma mais eficiente.

De acordo com Russell et al. (1996), utilizando um conjunto de variáveis, onde cada uma das quais tem um valor, um problema é solucionado quando cada variável possuir um valor que satisfaz todas as restrições impostas sobre a mesma. Um problema descrito desta maneira é chamado de *Constraint Satisfication Problem*, ou CSP.

2.1 Constraint Satisfation Problems

De acordo com Russell et al. (1996), um *Constraint Satiffaction Problem (CSP)* consiste em três componentes, X, D e C, onde:

- X é um conjunto de variáveis, $\{X_i, ..., X_n\}$;
- D é um conjunto de domínios, $\{D_i, ..., D_n\}$, um para cada variável;
- C é um conjunto de restrições que especifica possíveis combinações de valores.

Cada domínio D_i, consiste em um conjunto de possíveis valores $\{V_1, ..., V_k\}$ para a variável X_i. Cada restrição C_i consiste no par *(escopo, rel)*, onde *escopo* é uma tupla de valores que participam na restrição e *rel* é a relação que define os valores que essas variáveis podem assumir. A relação pode ser representada como uma lista de todas as tuplas de valores que satisfazem a restrição, ou como uma relação abstrata que suporta operações: testando se uma tupla é membro da relação e enumerando os membros da relação. Por exemplo, se X_1 e X_2 ambos têm o domínio {A,B}, então, a restrição que diz que essas duas variáveis devem ter valores diferentes pode ser escrita como $((X_1, X_2), \{(A, B), (B, A)\})$ ou como $((X_1, X_2), X_1 \neq X_2)$.

Para resolver um CSP, precisa-se definir um estado-espaço. Cada estado em um CSP é definido por uma atribuição de valores a algumas ou todas as variáveis. Uma atribuição que não viola nenhuma restrição é chamada de atribuição consistente. Uma atribuição parcial é aquela que atribui valores apenas a algumas variáveis. Já uma atribuição completa é uma atribuição onde cada variável é atribuída e finalmente, uma solução para um CSP é uma atribuição completa e consistente.

2.1.1 Exemplo de CSP: Coloração de Mapa

Um exemplo bastante difundido em CSP é o problema da coloração de mapas, que consiste em colorir as regiões de um determinado mapa de modo que duas regiões vizinhas não possuam a mesma cor. Neste problema, as regiões do mapa são as variáveis, o domínio é o conjunto das possíveis cores para as regiões, e a restrição é: duas regiões vizinhas não podem possuir a mesma cor.

Neste tipo de problema a construção de um *constraint graph* ou gráfico de restrição, como mostrado na Figura 1 pode ser útil para a visualização do problema. Os nós do gráfo correspondem as variáveis do problema, e uma aresta conecta quaisquer duas variáveis que participem de uma restrição

Tomando como exemplo o grafo da Figura 1, para formular esse exemplo como um CSP, definem-se as variáveis como sendo as regiões do mapa.

$$X = \{A, B, C, D, E, F, G\}.$$

O domínio de cada variável é o conjunto $D_i = \{vermelho, verde, azul\}$. As restrições requerem que duas regiões vizinhas tenham cores distintas. O grafo de restrições possui

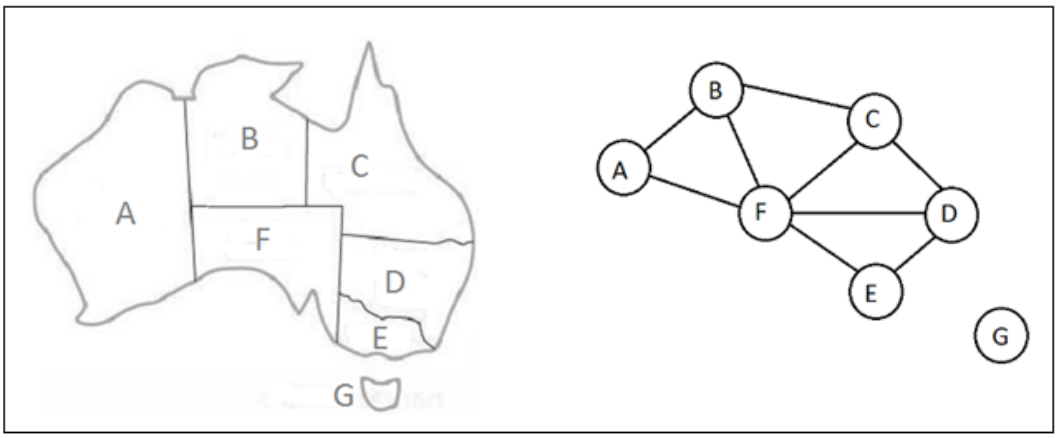

Figura 1: Grafo de restrição

nove arestas, ou seja, nove regiões de fronteira. A partir dessa observação, nove restrições são obtidas:

$$C = \{F \neq A, F \neq B, F \neq C, F \neq D, F \neq E, A \neq B, B \neq C, C \neq D, D \neq E\}.$$

$F \neq A$ é uma abreviação para $((F, A), F \neq A)$, onde $F \neq A$ pode ser enumerado por sua vez como:

$\{(vermelho, verde), (vermelho, azul), (verde, vermelho), (verde, azul),$
$(azul, vermelho), (azul, verde)\}$

Existem várias possíveis soluções para este problema, como:

$\{A = vermelho, B = verde, C = vermelho, D = verde, E = vermelho, F = azul,$
$G = vermelho\}$

Uma das razões de se modelar um problema como um CSP é que os modeols CSPs são uma representação natural para uma grande variedade de problemas. Se já possuirmos um sistema de resolução de CSP (*CSP-solving system*), é mais fácil resolver problemas utilizando esse sistema do que construir uma solução personalizada utilizando outras técnicas de busca para cada problema (RUSSELL et al., 1996). Além disso, *CSP-solvers* podem ser mais rápidos do que algoritmos de busca de estado-espaço porque os *CSP-solvers* podem eliminar rapidamente grandes amostras do espaço de busca. Por exemplo, uma vez que atribuímos $\{F = Azul\}$ no problema anterior, pode-se concluir que nenhum das cinco regiões vizinhas pode assumir a cor azul. Sem tirar vantagem da *constraint propagation* (propagação de restrições) (RUSSELL et al., 1996) , um procedimento de busca teria que considerar $3^5 = 243$ atribuições para as cinco variáveis vizinhas à variável F; já

com a *Constraint Propagation* não se precisa considerar "azul" como um valor, logo, se tem apenas $2^5 = 32$ atribuições possíveis. Uma redução de 87%.

Com modelos CSPs, uma vez descoberto que uma atribuição parcial não é uma solução, podemos imediatamente descartar futuros refinamentos sobre essa atribuição. Além disso, podemos observar quais variáveis infringem a restrição, e então concentrar a atenção sobre as variáveis que são importantes. Como resultado, muitos problemas que são intratáveis utilizando busca em estado-espaço podem ser resolvidos rapidamente quando formulados como um CSP (RUSSELL et al., 1996).

2.1.2 Exemplo de CSP: Planejamento de Tarefas

Outro exemplo de problema que pode ser descrito como um CSP é o problema de planejar um dia de trabalho em uma fábrica, o qual está sujeito a varias restrições. Na pratica, muitos desses problemas são resolvidos com técnicas de CSP (RUSSELL et al., 1996). Considere o problema de planejamento de uma montadora de carro. Todo o trabalho é composto de tarefas e cada tarefa pode ser modelada como uma variável, onde o valor de cada variável é o momento em que a tarefa é iniciada, expresso como um numero inteiro de minutos. Restrições podem afirmar que uma tarefa deve ocorrer antes de outra, por exemplo, a roda do carro deve ser fixada antes que a calota seja colocada, e que apenas um número determinado de tarefas pode ser executado por vez.

Para ilustrar considerarmos apenas uma pequena parte da montagem do carro, consistindo de 15 tarefas: montar eixos (dianteiro e traseiro), fixar as quatro rodas, apertar as porcas de cada roda, fixar as calotas e inspecionar toda a montagem. As tarefas podem ser representadas por 15 variáveis:

$$\{X = Eixo_d, Eixo_t, Roda_{de}, Roda_{dd}, Roda_{te}, Roda_{td}, Porca_{de}, Porca_{dd}, Porca_{te},$$

$$Porc_{td}, Cal_{de}, Cal_{dd}, Cal_{te}, Cal_{dd}, Inspec\}$$

O valor de cada variável é o tempo em que a tarefa é iniciada. O próximo passo é representar as restrições de precedência entre duas variáveis individuais. Sempre que uma tarefa T_1 precise ocorrer antes que a tarefa T_2, e a tarefa T_1 dure D_1 para ser completada, é adicionada uma restrição aritmética da forma $T_1 + d_1 \leq T_2$.

Neste exemplo os eixos devem ser colocados antes que as rodas, e a fixação dos eixos demora 10 minutos para ser completada, assim:

$$Eixo_d + 10 \leq Roda_{de}; Eixo_d + 10 \leq Roda_{dd};$$

$$Eixo_t + 10 \leq Roda_{te}; Eixo_t + 10 \leq Roda_{td}.$$

Para cada roda, é preciso colocar a roda (1 minuto), depois apertar as roscas (2 minutos) e depois colocar a calota (1 minuto).

$$Roda_{de} + 1 \leq Porca_{de}; Porca_{de} + 2 \leq Cal_{de};$$

$$Roda_{dd} + 1 \leq Porca_{dd}; Porca_{dd} + 2 \leq Cal_{dd};$$

$$Roda_{te} + 1 \leq Porca_{te}; Porca_{te} + 2 \leq Cal_{te};$$

$$Roda_{td} + 1 \leq Porca_{td}; Porca_{td} + 2 \leq Cal_{td}.$$

Suponha que exista quatro trabalhadores para instalar as rodas, mas eles compartilham uma ferramenta que ajuda a colocar o eixo no lugar. É necessário uma restrição disjuntiva para dizer que $Eixo_d$ e $Eixo_t$ não deve sobrepor-se no tempo:

$$(Eixo_d + 10 \neq Eixo_t) ou (Eixo_t + 10 \neq Eixo_d).$$

Esta restrição parece ser mais complicada, combinando aritmética e lógica. Mas ainda reduz-se a um conjunto de pares de valores que $Eixo_d$ e $Eixo_e$ podem assumir.

Também é necessário afirmar que a inspeção vem por ultimo e demora 3 minutos. Para cada variável exceto Inspec é adicionada a restrição da forma $X + d_x \neq Inspec$. Finalmente, supondo que há uma exigência para que toda a montagem seja feita em 30 minutos. Pode-se conseguir isso limitando o domínio de todas as variáveis:

$$D_i = \{1, 2, 3, ...27\}.$$

Apesar da resolução desse problema em particular, ser trivial, CSPs tem sido aplicados a problemas de planejamento como esse com milhares de variáveis. Em alguns casos, há restrições complicadas que são difíceis de especificar utilizando a modelagem CSP.

2.1.3 Variações da Modelagem CSP

O tipo mais simples de CSP envolve variáveis que possuem domínio finito e discreto, problemas como os de coloração de mapa e planejamento de tarefas são desse tipo.

Entretanto, um domínio discreto pode ser infinito, como o conjunto dos inteiros ou *strings* (cadeias de caracteres). Com domínios infinitos, não é mais possível descrever restrições enumerando todos as combinações de valores possíveis. Ao invés disso, deve ser usada uma linguagem de restrições (*constraint language*) (ROSSI; BEEK; WALSH, 2006), que entenda diretamente restrições como $(T_1 + d_1 \leq T_2)$, sem enumerar o conjunto de pares dos possíveis valores para (T_1, T_2). Existem algoritmos especiais para solução de restrições lineares sobre variáveis inteiras, onde cada variável aparece apenas na forma linear (RUSSELL et al., 1996).

Já os *Constraint satisfaction problems* com domínios contínuos são comum em problemas reais e são amplamente estudados no campo de pesquisas operacionais (RUSSELL et al., 1996). Por exemplo, o planejamento dos experimentos com o telescópio espacial Hubble, exige um tempo muito preciso de observações; o inicio e o fim de cada observação e manobra são variáveis de valores contínuos que devem obedecer uma variedade de restrições astronômicas, de precedência e de energia (RUSSELL et al., 1996).

A categoria mais conhecida de CSPs com domínio continuo é a de problemas de programação linear (*Linear Programming Problems*), onde as restrições devem ser igualdades ou desigualdades lineares. *Linear Programming Problems* podem ser solucionados em tempo polinomial de acordo com o número das variáveis. Problemas com diferentes tipos de restrições e funções objetivo também tem sido estudados (RUSSELL et al., 1996).

Além disso, para examinar os tipos de variáveis que podem aparecer em modelos CSP, é vantajoso observar os tipos de restrições. O tipo mais simples é a restrição unária (*unary constraint*), que restringe o valor de uma única variável. Por exemplo, no problema da coloração de mapa visto na seção anterior, poderia ser o caso em que a região A não tolerasse a cor verde, isso pode ser expresso com a restrição unária $(A \neq verde)$.

Uma restrição binária (*binary constraint*) relaciona duas variáveis, por exemplo, $A \neq B$ é uma restrição binária. Um CSP binário é um problema apenas com restrições binárias e pode ser representado como um grafo de restrição, como na Figura 1.

Também podem ser descritas restrições de ordem superior, tais como assegurar que o valor de Y está entre X e Z, como a restrição ternária *Between*(X,Y,Z). Uma restrição envolvendo um número arbitrário de variáveis é chamada de restrição global (*global cons-*

traint). Apesar do nome, uma restrição global não precisa envolver todas as variáveis de um problema. Uma das restrições globais mais comuns é *Alldiff*, que afirma que todas as variáveis envolvidas na restrição devem ter valores diferentes. Por exemplo, a restrição *Alldiff*(A,B,C,D) quer dizer que as variáveis A, B, C e D devem ter valores diferentes entre si.

É importante destacar que toda restrição de domínio finito pode ser reduzida a um conjunto de restrições binárias, se forem introduzidas variáveis auxiliares (RUSSELL et al., 1996). Desse modo, qualquer CSP pode ser transformado em um CSP apenas com restrições binárias. Por exemplo, a restrição global *Alldiff*(A,B,C,D) pode ser descrita por um conjunto de restrições binárias; $\{A \neq B, A \neq C, A \neq D, B \neq C, B \neq D, C \neq D\}$. Isso torna os algoritmos de *constraint solving* mais simples.

Uma forma simples de detectar inconsistencia para restrições *Alldiff* funciona da seguinte maneira: se m variáveis estão envolvidas na restrição, e se existem n possíveis valores distintos no total, e $m > n$, então a restrição não pode ser satisfeita. Isso leva ao simples algoritmo: primeiro, remove-se qualquer variável na restrição que tem um domínio *singleton* (isto é, com apenas um valor), e exclui-se o valor desta variável dos domínios das variáveis restantes. Repete-se enquanto houver variáveis *singleton*. Se em algum momento um domínio vazio é produzido ou há mais variáveis do que os valores restantes do domínio, então uma inconsistência foi detectada.

Outra maneira de converter qualquer CSP em um CSP com restrições binárias é a *dual grapfh transformation* (RUSSELL et al., 1996): criar um novo grafo onde haverá uma variável para cada restrição do grafo original, e uma restrição binária para cada par de restrições no grafo original que compartilhem variáveis. Por exemplo, se o grafo original tiver as variáveis X, Y, Z e as restrições $((X, Y, Z), C_1)$ e $((X, Y), C_2)$, então o *dual graph* tem duas variáveis C_1, C_2 com a restrição binária $((X, Y), R_1)$, onde (X,Y) são as variáveis compartilhadas e R_1 é uma nova relação que define a restrição entre as variáveis compartilhadas, como especificado pelo original $C_1 e C_2$.

No entanto, existem duas razões pelas quais deve se preferir restrições globais como *Alldiff* ao invés de um conjunto de restrições binárias. Primeiro, é mais fácil e menos suscetível a erros escrever a descrição do problema usando restrições globais. Segundo, é possível projetar algoritmos de inferência para restrições globais que não são disponíveis para um conjunto de restrições mais primitivas (RUSSELL et al., 1996).

As restrições descritas até agora, são todas restrições absolutas, isto é, a violação dessas restrições exclui possíveis soluções. Muitos problemas do mudo real incluem res-

trições preferenciais (*preference constraints*), indicando quais soluções são preferidas. Por exemplo, numa universidade, o problema de alocação de aulas possui restrições absolutas, onde, nenhum professor pode estar em duas aulas ao mesmo tempo. Mas também podem existir restrições preferenciais: Professor X pode preferir ensinar no período da manhã, enquanto que Professor Z prefere ensinar no período da tarde. Restrições preferenciais podem muitas vezes ser codificadas como custos sobre atribuições de variáveis individuais, por exemplo, atribuir uma vaga da tarde para o Professor X custa 2 pontos na função objetivo global, enquanto uma vaga da manha custa 1. Com essa formulação, modelos CSP com preferencias podem ser resolvidos com métodos de busca otimizados. Esse tipo de problema é chamado de *constraint optimization problems* ou COP (RUSSELL et al., 1996). *Linear programming problems* fazem esse tipo de otimização.

2.2 Constraint Solvers

A ideia de *constraint programming* é resolver problemas simplesmente declarando restrições (condições, propriedades) e devem ser satisfeitas por uma solução do problema. (FRUHWIRTH; ABDENNADHER, 2005)

Quando executado, um *constraint program* gera sucetivamente restrições. Um programa especial, o *constraint solver*, armazena, combina e simplifica as restrições até uma solução ser encontrada. O *solver* reúne as restrições que chegam de forma incremental a partir de um ou mais programas em execução e armazena essas restrições em uma *constraint store*, uma estrutura de dados para as restrições. Então o *solver* simplifica e, se possível, encontra uma solução para o problema (FRUHWIRTH; ABDENNADHER, 2005).

2.2.1 SAT e SMT Solvers

Na ultima década houveram importantes avanços nas técnicas e ferramentas baseadas em lógica. Avanços significativos ocorreram no campo da *propositional satisfiability (SAT)* a tal ponto que, atualmente, *SAT solvers* são capazes de resolver problemas do mundo real modelados utilizando lógica de primeira ordem, como o problema de coloração de mapas.

As técnicas de SAT tem sido adaptadas para lógicas mais expressivas. Por exemplo, o caso da *SAT Modulo Theories* (SMT). Uma instancia de SMT é uma generalização de uma instancia de SAT em que algumas variáveis proposicionais são substituídas por predicados de teorias subjacentes, e pode conter formulas como:

$$f(f(x) - f(y)) \neq f(z) \wedge x + z \leq y \wedge y \leq x \Rightarrow z < 0,$$

fornecendo uma linguagem de modelagem muito mais rica que simples formulas proposicionais (FRUHWIRTH; ABDENNADHER, 2005).

A principal área de aplicação de SMT é a verificação de hardware e software (FRUHWIRTH; ABDENNADHER, 2005). Entretanto, as teorias disponíveis não restringem o uso de SMT a problemas de verificação, e, de fato, permite codificar vários problemas fora da área de verificação de uma maneira natural.

2.3 Teste de Software

Durante o desenvolvimento de um *software*, o processo de testes exerce um importante papel para garantir a qualidade do *software*. O orocesso de testes é um conjunto de atividades apuradas e bem planejadas. Essas atividades têm como objetivo verificar se todos os requisitos do sistema foram implementados corretamente e assegurar a qualidade e corretude do *software* produzido (MYERS, 1979).

Um ponto importante no processo de testes é a geração dos dados que serão utilziados nessas atividades. Atualmente, empresas empregam uma grande quantidade de esforço e recursos durante a geração dos dados de teste, pois esse dados geralmente são escritos manualmente, levando a erros, falhas e redundâncias.

A geração automática de dados de teste surge como uma abordagem promissora para minimizar o esforço na geração de dados de teste, resultando em um maior controle de qualidade e na redução dos custos inerentes ao processo.

3 Método de Pesquisa

Este Capítulo apresenta a metodologia aplicada neste trabalho, sua qualificação, estapas, critérios de inclusão e exclusão, processo de seleção, estratégia de estração e sintese dos dados e apresentação dos resultados.

3.1 Qualificação do Método de Pesquisa

Este trabalho optou pela utilização do método de mapeamento sistemático da literatura. Os mapeamentos sistemáticos têm por objetivo apresentar uma avaliação justa de um tópico de investigação, usando uma confiável, rigorosa e verificável metodologia (KITCHENHAM, 2007).

3.2 Etapas do Método de Pesquisa

Um mapeamento sistemático começa com a definição do protocolo que especifica as questões de investigação e os métodos que serão utilizados para conduzir o mapeamento. De acordo com Kitchenham (2007), além das razões e objetivos da pesquisa devem fazer parte do protocolo:

- As questões de investigação que a pesquisa pretende responder. A estratégia que será usada para procurar os estudos primários, incluindo os termos de pesquisa e recursos necessários.
- Critérios de seleção do estudo que serão usados para determinar quais estudos serão incluídos ou excluídos do mapeamento sistemático.
- Uma descrição de como os critérios de seleção serão aplicados, por exemplo, como avaliar cada estudo primário, e como divergências entre os avaliadores serão resolvidas.

- O protocolo também deve conter listas de verificação da qualidade e procedimentos de avaliação desta.
- Uma estratégia de extração de dados que define como e quais informações serão extraídas de cada estudo primário. Se os dados exigem manipulação ou suposições e inferências a serem feitas, o protocolo deve especificar um processo de validação adequado.
- Uma estratégia de síntese dos dados. Esta informação deve esclarecer se uma meta-análise formal se aplica e caso afirmativo, quais técnicas serão usadas.
- Por fim, uma estratégia de divulgação. Esta define como as conclusões oriundas da pesquisa serão apresentadas e divulgadas para terceiros interessados no mesmo tema.

3.3 Questões de Pesquisa

A partir do que está apresentado anteriormente, a principal questão de pesquisa deste estudo é: **Como *Constraint Solvers* têm sido utilizados para gerar automaticamente dados de teste?**

Para um mapeamento sistemático mais preciso, foram elaboradas as se seguintes subquestões:

- Qual a escalabilidade para problemas reais da utilização de *Constraint Solvers* para geração automática de dados de teste?
- Quais os benefícios da utilização de *Constraint Solvers* para geração automática de dados de teste?
- Quais as limitações da utilização de *Constraint Solvers* para geração automática de dados de teste?
- Quais as técnicas mais utilizadas para gerar dados de teste utilizando *Constraint Solvers*?
- Qual o tipo de *Constraint Solver* mais utilizado?

3.4 Estratégia de Busca

Segundo Kitchenham (2007), uma estratégia de busca deve ser usada para a pesquisa dos estudos primários. Devem ser definidas palavras chaves, termos de busca e decididas em que bibliotecas digitais, jornais e conferências, os termos serão submetidos. Devido à limitação de tempo disponível. A estratégia usada nessa pesquisa é apresentada nas próximas subseções.

3.5 Termos Chaves da Pesquisa

A partir da questão de investigação, os principais termos podem ser identificados. Após a identificação, é realizada a tradução desses termos para a língua inglesa, por ser a língua utilizada nas bases de dados eletrônicas pesquisadas e nos principais jornais e conferencias dos tópicos de investigação.

Os termos e sinônimos identificados são apresentados abaixo:

- **Dados de Teste:** Test Data, Test Vector.
- **Constraint Solver:** Constraint Solver, Constraint Solving, SAT, SMT, CLP.

3.6 Termos e Predicado de Busca

Segundo Kitchenham (2007), os termos são construídos a partir das estruturas das questões de pesquisa e podem sofrer adaptações de acordo com as necessidades especificas de cada base da dados. Assim, a busca é realizada a partir da combinação dos termos chave e sinônimos usando *OR* (ou) e *AND* (e), e possíveis particularidades das bibliotecas digitais. Nesta pesquisa apenas uma expressão de busca será utilizada.

((“test data” OR “test vector”)) AND (“constraint solving” OR “constraint solver” OR SAT OR SMT OR CLP)

3.7 Fontes de Busca

Os critérios de seleção das fontes de busca são:

- Disponibilidade de consulta dos artigos na web.

- Presença de mecanismos de busca utilizando palavras-chave.
- Importância e relevância das fontes.

Assim, com a expressão de busca definida, as fontes de pesquisa utilizadas para a busca dos estudos primários são listadas abaixo:

- ACM Digital Library (portal.acm.org/dl.cfm)
  - Seguindo o seguinte procedimento: (i) entrar no site Portal ACM (*http* : *//dl.acm.org/*); (ii) clicar em *Search*; (iii) considerando a base "*The ACM Guide to Computing Literature*". clicar em "*Advanced Search*"e (iv) digitar a *string* de busca acima no campo localizado abaixo do texto "*Edit the query directly, or use the form below*".
- IEEE Xplore (ieeexplore.ieee.org/Xplore/dynhome.jsp)
  - Seguindo o seguinte procedimento: (i) entrar no site do IEEE XPlore (*http* : *//ieeexplore.ieee.org/*); (ii) clicar em "*Advanced Search*"; (iii) clicar em "*Switch to Command Search*"; (iv) escolher a opção "*Full Text & Metadata*"e (v) digitar a string de busca acima no campo principal.
- SpringerLink (www.springer.com)
  - Seguindo o seguinte procedimento: (i) entrar no site do Springer Link (*http* : *//www.springerlink.com/*) e (ii) digitar a *string* de busca acima no campo principal ("*Search For* ").
- Science Direct (www.sciencedirect.com)
  - Seguindo o seguinte procedimento: (i) entrar no site do Science Direct (*http* : *//www.sciencedirect.com/*) e (ii) digitar a *string* de busca acima no campo principal ("*All Fields*").

Para conferir uma maior confiabilidade à pesquisa, evitando a não consideração de artigos relevantes não retornados pela busca, idealmente, deveria ser feita uma pesquisa em segundo nível utilizando o processo *snow-balling* (GREENHALGH; PEACOCK, 2005), ou seja, para todos os artigos selecionados, as suas referências seriam analisadas para identificar trabalhos relevantes não capturados pela expressão de busca.

Esse processo e a expressão de busca foram validados a partir de sete trabalhos já conhecidos e considerados relevantes, estes trabalhos estão listados no Apêndice B . A lista final com os trabalhos selecionados deve ser avaliada por um pesquisador mais experiente, de forma que este possa validá-la e porventura incluir trabalhos relevantes não selecionados. Essa avaliação está sujeita a disponibilidade de tempo, e a disponibilidade do pesquisador escolhido.

3.8 Seleção dos Estudos

Os estudos que fazem parte desta pesquisa são:

- Artigos em revistas, conferências e congressos;
- Relatórios técnicos;
- Dissertações e teses.

Uma vez obtidos os estudos primários, os mesmos passarão por uma análise para comprovar sua relevância para a pesquisa, e para que estudos não relevantes sejam descartados. Segundo Travassos (2007), critérios de inclusão e exclusão devem ser baseados nas questões de pesquisa. Portanto, os critérios de inclusão e exclusão foram baseados nos trabalhos de Travassos (2007) e Kitchenham (2007) e serão apresentados a seguir.

3.9 Critérios de Inclusão

Para um trabalho ser considerado relevante para esta pesquisa são executadas duas etapas. Estas etapas se aplicam tanto aos artigos retornados pela aplicação da expressão de busca nas bases de dados antes citados, como também na pesquisa em segundo nível, antes explicada.

Na primeira etapa são analisados o título e as palavras-chave e o resumo de cada artigo, verificando se os mesmos estão de acordo com o tema deste estudo. Na segunda etapa, através da leitura da introdução e conclusão, é verificado se o trabalho obedece aos critérios listados abaixo:

- Os documentos devem estar inteiramente disponíveis na internet

- Os textos devem apresentar informações sobre geração automática de dados de teste utilizando *Constraint Solvers*.

3.10 Critérios de Exclusão

Serão descartados trabalhos que se enquadrem em algum dos critérios abaixo:

- Estudos que não sejam disponíveis livremente na web;
- Estudos irrelevantes para a pesquisa, de acordo com a questão de pesquisa;
- Estudos repetidos: se determinado estudo estiver disponível em diferentes fontes de busca, a primeira pesquisa será considerada;
- Estudos duplicados: caso dois trabalhos apresentam estudos semelhantes, apenas o mais recente e/ou mais completos será considerado, a menos que tenham informação complementar;
- Estudos que apresentem texto, conteúdo e resultados incompletos, ou seja, normalmente, trabalhos com menos de 2 páginas.

3.11 Processo de Seleção dos Estudos Primários

Após a definição das questões de pesquisa, da estratégia de busca e dos critérios de inclusão e exclusão, as etapas do processo de seleção dos estudos primários são descritas abaixo:

1. Em uma data fixa, a expressão de busca será aplicada nas bases de dados selecionadas, e a lista com todos os resultados será armazenada. Desta forma, será criado uma fotografia em cima do qual a seleção dos artigos ocorrerá.
2. De forma paralela e independente, o autor deste trabalho e o seu orientador irão analisar o documento pdf antes mencionado e criar três listas: uma lista com os artigos que devem ser incluídos, uma lista com os artigos que devem ser excluídos e outra lista de indecisos com artigos para os quais não foi possível identificar se o mesmo se enquadra no tema desta pesquisa. Isto será feito a partir da leitura do título, palavras-chave e resumo. Em seguida, estas listas serão unificadas da seguinte maneira:

- Se o artigo consta na lista de incluídos, ou na lista de indecisos, de pelo menos um dos pesquisadores, o mesmo será incluído.
- Se o artigo consta na lista de excluído dos dois autores, o mesmo será excluído.

De acordo com Kitchenham (2007), as buscas iniciais retornam uma grande quantidade de estudos que não são relevantes, não respondendo às questões ou mesmo não tendo relação com o tópico em questão. Portanto, é provável que a quantidade de artigos incluídos represente um pequeno percentual do total de artigos retornados.

3. Após este primeiro filtro (etapa 2), os dois pesquisadores, mais uma vez de forma paralela e independente, irão repetir o processo descrito anteriormente, contudo com a seguinte diferença: após a leitura da introdução e da conclusão de cada trabalho.

4. Os artigos na nova lista de incluídos serão lidos por completo. Caso o volume de trabalhos não seja compatível com o tempo disponível pela pesquisa, um corte temporal deve ser analisado. Por exemplo, considerar apenas os trabalhos dos últimos 5 anos. Para este trabalho serão lidos em torno de 25 trabalhos.

5. Após a leitura dos artigos, as etapas 2 e 3 são aplicadas de forma análoga, contudo, considerando agora as referências citadas pelos artigos lidos. Como dito anteriormente, o que constitui o segundo nível do mapeamento.

Os estudos incluídos são documentados através das Tabela 1. Enquanto que todos os trabalhos excluídos e o critério que definiu sua exclusão serão documentados na Tabela 2.

Tabela 1: Trabalhos incluídos.
[Fonte: elaboração própria]

Fonte	Título	1º Autor	Ano

Tabela 2: Trabalhos excluídos.
[Fonte: elaboração própria]

Fonte	Título	1º Autor	Ano	Critério de Exclusão

É realizada uma análise de confiabilidade para avaliar a concordância entre os investigadores. Para esta análise é utilizado o coeficiente kappa (k), o coeficiênte kappa é calculado a partir da seguinte fórmula: $k = \frac{P_o - P_e}{1 - P_e}$, onde P_o é a concordância obtida e P_e a concordância esperada (COHEN, 1960). Para interpretar a magnitude do coeficiente kappa obtido, serão utilizadas diretrizes propostas por (LANDIS; KOCH, 1977), como pode ser visto na Tabela 3. Para este trabalho será admitido o coeficiente kappa maior que 0.60, ou seja, nível de concordância substâncial.

Tabela 3: Coeficiente Kappa.
[Fonte: elaboração própria]

Coeficiente Kappa	**Nível de Concordância**
<0.00	Pobre
0.00 - 0.20	Leve
0.21 - 0.4	Razoável
0.41 - 0.6	Moderado
0.61 - 0.80	Substancial
0.81 - 1.00	Quase perfeito

3.12 Leitura dos Trabalhos Selecionados

Cada artigo é lido por somente um pesquisador, no caso, o autor deste trabalho. Durante a leitura dos artigos, é coletada a maior quantidade possível de informações relevantes para responder a questão de pesquisa deste mapeamento. As Tabelas 4, 5, 6, 7 e 8, apresentam um fichamento que define quais informações são coletadas e como elas são organizadas.

3.13 Avaliação da Qualidade dos Estudos

Além das informações coletadas a partir da Tabela 4, é importante considerar também a qualidade dos estudos selecionados. Apesar de não existir uma definição universal do que seja qualidade de estudo, a maioria dos *checklists* incluem questões que objetivam avaliar a extensão em que o viés é minimizado e a validação interna e externa são maximizadas (KITCHENHAM, 2007). De acordo com Kitchenham (2004), esta análise traz os seguintes benefícios:

Tabela 4: Informações gerais.
[Fonte: elaboração própria]

Informações gerais
Título do Trabalho:
Nº:
Autores:
Fonte de Pesquisa:
Local de Publicação:
Ano:
Tipo de Estudo: • Experimento • Teórico • Revisão Sistemática • Relato de Experiência Industrial
Tipo de Publicação: • Artigo • Periódico • Dissertação • Tese • Relatório Técnico • Capítulo de Livro • Livro
Principal contribuição do artigo (Resumo em 2 frases)

Tabela 5: Caracterização da contribuição.
[Fonte: elaboração própria]

Caracterização da contribuição.
O que espera de entrada, o que gera de saída? Em qual formalistmo a entrada e a saída estão representadas?
Diferencial em relação as outras estratégias.
Limitações.
Propriedades (Cobertura, corretude, etc.). Apresenta prova formal ou argumentação intuitiva? Relação de conformidade Implicita x Explicita? Verificação automática? Composicional?

Tabela 6: Caracterização das estapas.
[Fonte: elaboração própria]

Caracterização das estapas existentes na aplicação da contribuição.
Demais formalismos utilizados internamente.
Técnicas utilizadas internamente.
As etapas são automáticas ou há intervenção manual?

Tabela 7: Caracterização do apoio ferramental.
[Fonte: elaboração própria]

Caracterização do apoio ferramental (se aplicável)
Existe apoio ferramental?
Faz uso de que tipo de solver (SAT, SMT, outros)?
A ferramenta é paga ou gratuita? Se gratuita, onde baixar/requisitar?
Tem uso prático por parte de empresas? Quais?

Tabela 8: Caracterização da avaliação.
[Fonte: elaboração própria]

Caracterização da avaliação da contribuição.
A avaliação seguiu um planejamento bem estruturado?
Os instrumentos utilizados na avaliação estão disponíveis publicamente.
Se sim, onde baixa-los?
Detalhes da aplicação considerada: Domínio (ex: aviação, automobilismo, etc.).
Tamanho (LOC, quantidade de variáveis, restrições, etc.).
Resumo dos resultados Configuração do computador (se aplicável). Detalhes do projeto experimental (se aplicável)
Análise de escalabilidade para problemas reais?
Trabalhos futuros propostos

- Investigar se as diferenças de qualidade fornecem uma explicação para as diferenças dos resultados;
- Meio de ponderação da importância dos estudos individuais, quando os resultados estão sendo sintetizados;
- Orientar a interpretação dos resultados e determinar a força das inferências;
- Guia de recomendação para futuras pesquisas.

3.13.1 Critérios de Avaliação

Para avaliar a qualidade dos estudos para esta pesquisa, são levados em conta onze critérios baseados nos critérios propostos pelo *Critical Appraisal Skills Programme* e nos princípios de boas praticas para a realização de estudos empíricos em engenharia de software (DYBÅ; DINGSØYR, 2008).

De acordo com Dybå e Dingsøyr (2008), estes onze critérios abrangem três questões principais relativas à qualidade, que precisam ser consideradas ao avaliar os estudos identificados:

- Rigor: tem sido aplicada uma abordagem completa e apropriada aos métodos de investigação fundamentais no estudo?
- Credibilidade: os resultados são significativos e bem apresentados?
- Relevância: quão uteis os resultados são para a indústria de software e a comunidade acadêmica?

Dentre os onze critérios, três critérios estão relacionados com a qualidade dos resultados, objetivos e contexto do estudo. Assim, para cada estudo será avaliado se:

1. O estudo relatou uma pesquisa empírica ou foi apenas um relato de "lições aprendidas" baseadas na opinião de um especialista.
2. As metas e objetivos foram claramente relatados (incluindo uma razão do por quê o estudo foi realizado).
3. Houve uma descrição adequada do contexto em qual a pesquisa foi realizada.

Cinco critérios estão relacionados ao rigor dos métodos de pesquisa empregados para estabelecer a validade das ferramentas de coleta de dados e dos métodos de análise, e, portanto a confiabilidade dos resultados. Consequentemente para cada estudo será avaliado se:

4. O projeto de pesquisa foi apropriado para alcançar os objetivos da pesquisa.
5. Houve uma descrição adequada da amostra usada e dos métodos usados para identificar e recrutar essa amostra.
6. Algum grupo de controle foi usado para comparar os tratamentos.
7. Foram descritos e utilizados métodos de coleta de dados apropriados.
8. Houve uma descrição adequada dos métodos utilizados para analisar os dados, e se foram utilizados métodos apropriados para garantir que a analise dos dados foi fundamentada nos dados.

Além disso, dois outros critérios estão relacionados com a credibilidade dos métodos utilizados pelo estudo, para assegurar que os resultados são válidos e significativos. Em relação a isso, será verificado se:

9. A relação entre o pesquisador e os participantes foi considerada com adequada.
10. O estudo forneceu claramente resultados confiáveis e conclusões fundamentadas.

O critério final está relacionado com a relevância do estudo para a indústria de software em geral e a comunidade acadêmica. Assim, será verificado se:

11. O estudo forneceu valor para a pesquisa ou pratica.

Para a avaliação da qualidade dos estudos é utilizada a escala Likert-5, que permite o pesquisador responder as questões dos critérios de qualidade com respostas gradativas. O pesquisador pode usar os seguintes níveis de concordância ou discordância: (concordo totalmente, concordo parcialmente, neutro, discordo parcialmente e discordo totalmente). Para a avaliação, devem ser consideradas as seguintes observações:

- **Concordo totalmente (4)**: deve ser concedido no caso em que o trabalho atenda totalmente aos critérios da questão;

- **Concordo parcialmente (3)**: deve ser concedido no caso em que o trabalho atenda parcialmente aos critérios da questão;
- **Neutro (2)**: deve ser concedido no caso em que o trabalho não deixe claro se atende ou não a questão;
- **Discordo parcialmente(1)**: dever ser concedido no caso em que o trabalho não atenda aos critérios contidos na questão;
- **Discordo totalmente (0)**: deve ser concedido no caso em que o trabalho não atenda de forma alguma os critérios de avaliação, isto é, não existe nada no trabalho que atenda aos critérios da questão.

Os estudos primários avaliados podem então se enquadrar em 5 níveis de qualidade, conforme classificação de Beecham et al. (2007), a partir dos valores finais da avaliação de cada estudo, conforme mostra a Tabela 9

Tabela 9: Níveis de qualidade.
[Fonte: elaboração própria]

Faixa de Notas	**Avaliação**
>86%	Excelente
66% - 85%	Muito Boa
46% - 65%	Boa
26% - 45%	Média
<26%	Baixa

3.14 Estratégia de Extração Dos Dados

Para Kitchenham (2007), o objetivo desta etapa é criar formas de extração dos dados para registrar com precisão as informações obtidas a partir dos estudos primários. Esta deve ser projetada para coletar as informações necessárias as questões. Um formulário eletrônico é sugerido por vários trabalhos, pois segundo especialistas, o uso pode facilitar a análise posterior. Logo, para apoiar a extração e registro dos dados e posterior análise, será utilizada a ferramenta JabRef [1] , um gerenciador de referencias código aberto que permite a customização e facilidades na importação/exportação e dados (KITCHENHAM, 2007) (TRAVASSOS, 2007).

[1] http://jabref.sourceforge.net/

3.15 Síntese dos Dados Coletados

Após a coleta dos dados, as informações dever ser tabuladas de acordo com as questões de pesquisa, as tabelas devem ser estruturadas de forma a destacar as semelhanças e diferenças entre os resultados do estudo (KITCHENHAM, 2007).

De acordo com Kitchenham (2007), a síntese dos dados pode ser qualitativa e/ou quantitativa, sendo que a primeira necessariamente seria tratada como uma meta análise. Para esta pesquisa serão realizadas sínteses qualitativas e quantitativas.

3.16 Documentação e Apresentação dos Resultados

A fase final de uma revisão sistemática envolve a redação dos resultados de análise e divulgação dos resultados aos potenciais interessados. Alguns estudos indicam alguns tópicos necessários para a apresentação de uma revisão sistemática: Título (de acordo com as questões de pesquisa); Autores; Resumo do trabalho (contexto, objetivos, método, resultados e conclusões); *Background* (justificativa da necessidade da revisão); Questões de pesquisa; Método de revisão (estratégia de busca, seleção dos estudos, avaliação da qualidade, extração e síntese dos dados); Estudos incluídos e excluídos; Resultados; Discussão, e Conclusões (KITCHENHAM, 2007).

A partir da síntese dos dados, são sumarizadas evidencias existentes a fim de prover um melhor entendimento sobre os detalhes de como *constraint solvers* têm sido utilizados para gerar dados de teste.

4 Resultados

Neste capítulo são apresentados os resultados do estudo e a análise dos mesmos. Para melhor compreensão dos resultados obtidos, os mesmos são apresentados em duas seções:

- **4.1 Análise quantitativa do mapeamento sistemático** - apresenta dados gerais da revisão, como: processo de seleção com o número final de estudos primários, distribuição ao longo dos anos, locais de publicação, tipos de estudos e avaliação da qualidade.
- **4.2 Análise qualitativa das evidencias** - apresentação das evidencias identificadas pelo mapeamento sistemático, com o intuito de responder a questão de pesquisa.

4.1 Análise Quantitativa

A revisão sistemática foi executada de acordo com o que foi definido no protocolo que se encontra no capítulo anterior. A partir da expressão de busca e fontes definidas, as buscas primárias retornaram um total de 8.844 trabalhos, dos quais, 2.960 foram identificados no IEEE, 1.775 na ACM, 2.414 no ScienceDirect, e 1.695 no SpringerLink.

Devido a grande quantidade de trabalhos primários retornados pelo processo de busca e o limite de tempo disponível para a realização desta pesquisa, foram considerados apenas os trabalhos identificados na base da ACM, por ter sido a fonte que mais retornou trabalhos entre os sete trabalhos já conhecidos e considerados relevantes.

Também em consequência da indisponibilidade de tempo, não foi possível a realização de uma pesquisa em segundo nível utilizando o processo *snow-balling*, onde, para todos os artigos selecionados, as suas referências seriam analisadas para identificar trabalhos relevantes não capturados pela expressão de busca.

Através do processo de seleção e exclusão, o numero de trabalhos foi bastante reduzido. A partir da primeira seleção por título e palavra-chave, dos 1.775 trabalhos identificados

na ACM, apenas 95 foram identificados como estudos potencialmente relevantes para a pesquisa.

Com a leitura do resumo e conclusão dos estudos potencialmente relevantes, e utilizando-se os critérios de inclusão e exclusão, chegou-se a 42 estudos primários. Assim, 53 trabalhos considerados potencialmente relevantes na primeira seleção foram excluídos, e o principal motivo para a exclusão foi que os estudos foram considerados irrelevantes de acordo com a questão de pesquisa.

Como foi mencionado no processo de seleção do Capítulo 3, só seriam lidos em torno de vinte e cinco trabalhos, se mais de 30 trabalhos fossem selecionados, seria necessário um corte temporal que retornasse uma quantidade de trabalhos compatível com o tempo disponível. Desse modo, como foram selecionados 42 estudos primários, foram selecionados os trabalhos que foram publicados nos últimos 5 anos, ou seja, a partir de 2007. Foram identificados, então, 24 trabalhos publicados nos últimos 5 anos.

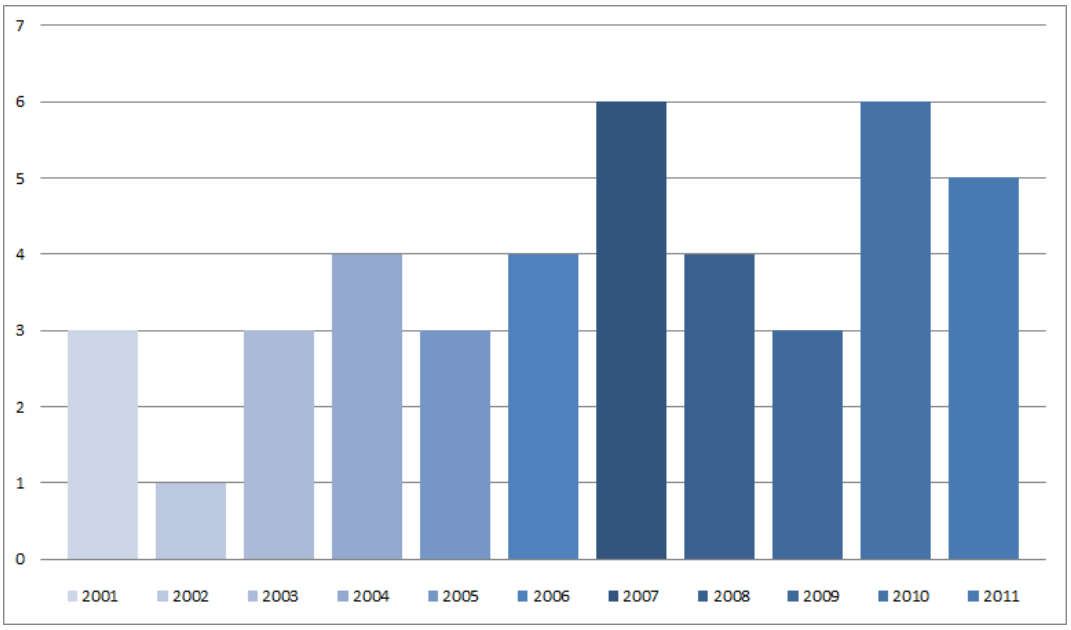

Figura 2: Distribuição dos estudos primários ao longo dos anos

A Figura 2 ilustra a distribuição dos estudos primários ao longo dos anos. Pode-se perceber que a utilização de *constraint solvers* para geração de dados de teste tem sido tema recorrente de pesquisas nos últimos 12 anos.

A Figura 3 apresenta a distribuição dos trabalhos por local de publicação. Através dela podemos verificar que a grande maioria dos estudos primários (96%) foi publicada em anais de eventos (Conferências, Workshops e Simpósios). Apenas 1 trabalho (4%) foi

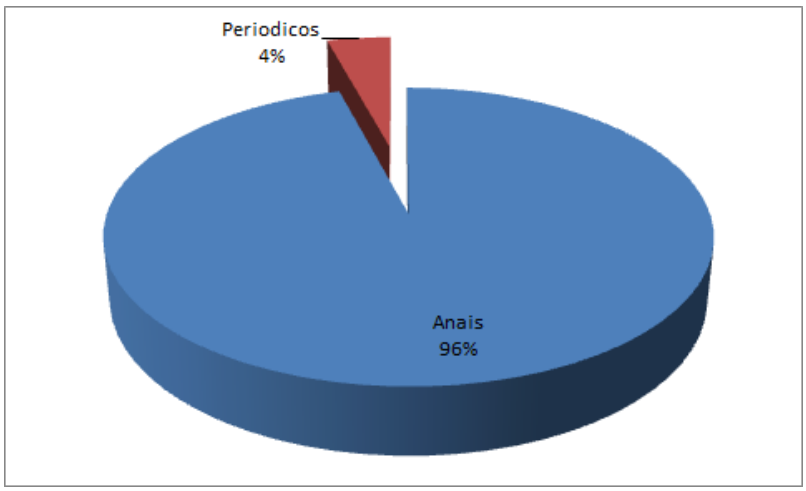

Figura 3: Distribuição dos estudos primários por local de publicação

publicado em periódicos.

A Tabela 10 apresenta a quantidade de estudos primários retornados por evento. Pode-se perceber que o *International Symposium on Software Testing and Analysis* (ISSTA) se destaca entre os demais eventos como sendo o evento onde ocorreram mais publicações sobre a utilização de *constarint solvers* na geração de dados de teste. Dos estudos primários selecionados, 16,6% foram publicados nesse evento. Apenas um estudo primário foi publicado em um periódico (*Journal Information and Software Technology*).

Dos 24 estudos primários avaliados, 88% se caracterizam como estudos experimentais (estudos baseados em evidencias ou experimentos, *Empirical Studies*, em inglês), 8% como teóricos (estudos conceituais baseados em um entendimento de uma área, referenciando outros trabalhos relacionados), e apenas 4%, ou seja, apenas 1 estudo primário foi caracterizado com revisão sistemática da literatura. Nenhum estudo primário foi caracterizado como relato de experiência industrial.

A Figura 4 mostra a distribuição dos trabalhos por tipo de estudo. Dentro os estudos classificados como experimento, todos foram classificados como experimento controlado de acordo com a classificação de Easterbrook et al. (2007).

A Figura 5 apresenta a distribuição dos estudos por local dos autores, desse modo será possível identificar quais países e ou centros de pesquisa são mais atuantes na área de pesquisa. Ainda nesta Figura, pode ser observado que os Estados Unidos aparecem como principal país, com 10 trabalhos (42%) entre os trabalhos selecionados. Dentre os 10 trabalhos publicados por autores americanos, 5 trabalhos (50%) foram publicados pela Universidade do Texas. A França também se destaca, com 5 trabalhos (2%) entre

Tabela 10: Quantidade de estudos primários por evento.
[Fonte: elaboração própria]

Evento	**Quantidade de Trabalhos**
International Symposium on Software Testing and Analysis	4
Foundations of Software Engineering	3
International Conference on Software Engineering	3
Automated Software Engineering	2
European Conference on Object-Oriented Programming	2
Genetic and Evolutionary Computation Conference	2
ACM SIGPLAN workshop on Partial evaluation and program manipulation	1
ACM Symposium On Applied Computing	1
Design, Automation, and Test in Europe	1
GLSVLSI Great Lakes Symposium on VLSI	1
Library-Centric Software Design	1
NASA Formal Methods Symposium	1
Principles and Practice of Declarative Programming	1

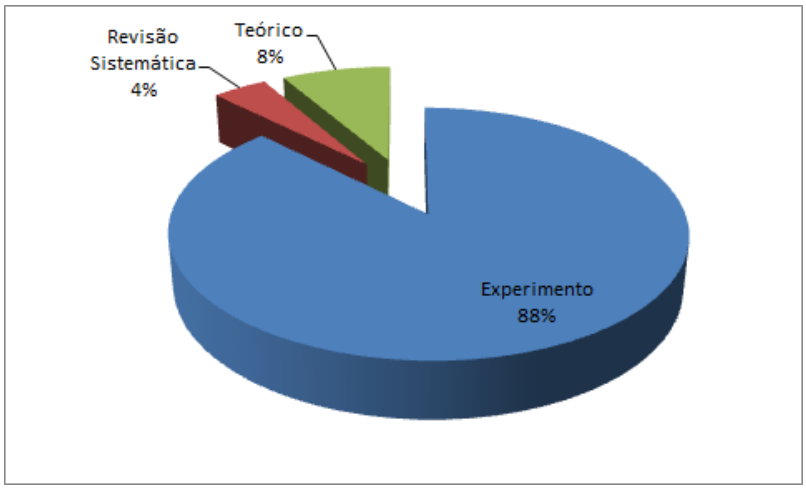

Figura 4: Distribuição dos estudos primários por tipo de estudo.

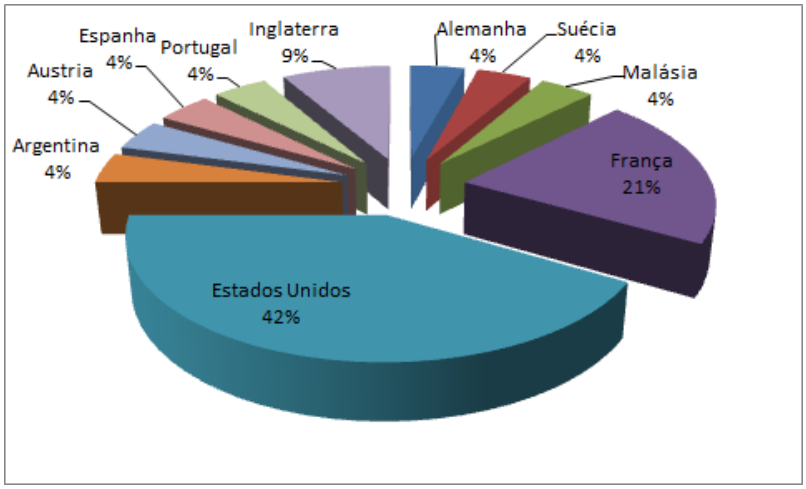

Figura 5: Distribuição dos estudos primários por local dos autores.

os selecionados.

Os resultados da avaliação de qualidade são apresentados pela Tabela 11 e na Figura 6. O valor máximo que um estudo pode alcançar, com base nos critérios de avaliação descritos no capitulo 3, seria de 44 pontos. Com base na nota alcançada pelo estudo, o mesmo será analisado conforme a Tabela 9, baseada em Beecham et al. (2007), indicando a classificação do estudo: Excelente, Muito Boa, Boa, Média e Baixa.

Como pode ser observado, apenas 1 estudo encontra-se na faixa Baixa e 1 estudo na Média. Enquanto 5 estudos (21%) estão na faixa boa, 12 estudos (50%) estão na faixa muito boa, e 5 estudos (21%) estão na faixa Excelente. Portanto a maioria dos trabalhos analisados apresentam qualidade acima da média de acordo com os critérios utilizados.

Tabela 11: Qualidade dos estudos primários.
[Fonte: elaboração própria]

	Classificação do estudo					
	Baixa	Média	Boa	Muito Boa	Excelente	Total
Quantidade	1	1	5	12	5	24

É importante destacar que quanto melhor avaliado for um trabalho, maior deve ser a importância dada às evidências fornecidas por ele. A qualidade atribuída ao estudo é levada em consideração na análise das evidências.

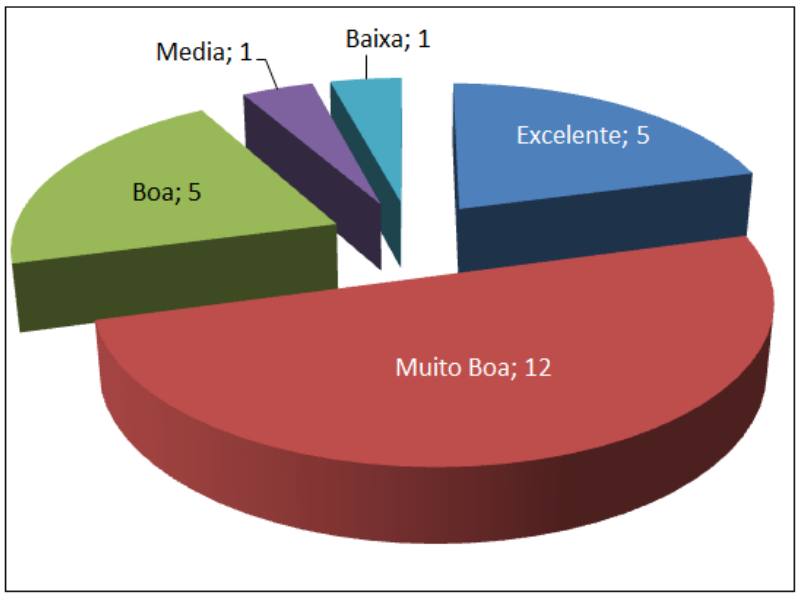

Figura 6: Distribuição por qualidade dos estudos.

4.2 Análise Qualitativa das Evidências

Nessa seção, são apresentadas as evidências que ajudam a responder a questão de pesquisa deste estudo. Todas as evidências são referenciadas pelos 24 estudos primários, e os números de identificação são precedidos por EP (Estudo Primário) que se encontram disponíveis no apêndice X.

4.2.1 Principais Técnicas Utilizadas

Na tentativa de responder a questão de pesquisa, nesta seção serão apresentadas as principais técnicas mais utilizadas pelas abordagens propostas pelos estudos primários.

A técnica mais citada pelos trabalhos foi a técnica de execução simbólica (*symbolic execution* em inglês), sendo citada por 7 dos 21 trabalhos classificados como experimento. A principal da *symbolic execution* é utilizar valores simbólicos como entrada do programa, ao invés de valores reais, e utilizar expressões simbólicas para representar as variáveis do programa. Como resultado, os valores de saída computados pelo programa são expressos como uma função dos valores de entrada simbólicos (LAKHOTIA; HARMAN; MCMINN, 2008).

Algumas limitações da *symbolic execution* foram identificadas pelos estudos primários. As principais limitações encontradas foram: o problema da explosão de caminho (*The*

Path Explosion Problem) e a capacidade limitada de "raciocinar" sobre determinadas construções do código, como objetos do sistema, chamadas a funções externas, etc.

- EP_09 - *"Symbolic execution suffers from scalability issues since the number of symbolic paths that need to be explored is very large (or even infinite)..."*
- EP_12 - *"while in hardware we can perform symbolic simulation easily, in software a symbolic simulation will miss some of the aspects of a program."*

Para compensar essas limitações, alguns dos trabalhos lidos fizeram uso da técnica *concolic testing* (LAKHOTIA; HARMAN; MCMINN, 2008). *Concolic testing* é uma técnica que combina a execução concreta de um programa com a execução simbólica do mesmo.

- EP_16 - *"One of the principle strenghts of concolic testing is the way in which concrete values are used to overcome many of the problems associated with symbolic execution."*
- EP_21 - *"Simultaneously, the concrete execution helps to retain precision in the symbolic computations by allowing dynamics values to be used in the symbolic executor."*

O principal beneficio das abordagens que utilizaram as técnicas citadas anteriormente, juntamente com técnicas de *constraint solving*, foi alcançar o critério de cobertura utilizando um numero mínimo de casos de teste. Critérios de cobertura são heurísticas que tentam estimar o quão bem o programa é exercitado por um conjunto de testes. Exemplos de critérios de cobertura são: *statement coverage*, onde requer que cada linha do código seja executada; *path coverage*, onde requer que todos os caminhos possíveis de todas as funções sejam executados, etc.

- EP_19 - *"The full branch and code coverage is not a surprese since symbolic execution generates tests for all reachable branches and all the branches in the event handlers..."*

4.2.2 Escalabilidade para Problemas Reais

Escalabilidade é uma característica desejável em todo o sistema, em uma rede ou em um processo, que indica sua habilidade de manipular uma porção crescente de trabalho

de forma uniforme, ou estar preparado para crescer (BONDI, 2000). Nesta seção será apresentada uma análise de escalabilidade para problemas reais sobre os estudos primários selecionados.

Dentre os 21 estudos primários classificados como experimento, 5 (EP_01, EP_02, EP_04, EP_18, EP_20) apresentaram resultados para sistemas do mundo real. A avaliação feita pelo EP_01 foi realizada utilizando 5 modelos do mundo real para testes de sistema de controle de funções automotivas. Os modelos utilizados são propriedade intelectual da Daimler (http://www.daimler.com/).

O EP_02 avaliou sua abordagem utilizando 14 projetos Ruby[1] do mundo real.

O EP_04 avaliou sua abordagem realizando testes em uma aplicação industrial de tempo real utilizada por um fabricante de veículos e automóveis. A aplicação fornecida pela Geensoft (http://www.geensoft.com/) é um controlador que controla várias funções relacionadas com as luzes de um carro. Uma completa descrição , junto com seu código está disponível na web[2].

A aplicação do mundo real utilizada pelo EP_18 foi um gerador de treino utilizado pelo *Sports Clubs Apolon*. A aplicação possui 649 linhas de código e recebe como entrada características pessoais do usuário, como altura, peso, idade, metabolismo e nível de experiência, e com base nesses dados de entrada, a aplicação gera um programa de treino adequado.

O EP_20 não utiliza nenhuma aplicação do mundo real para avaliar sua abordagem, mas o autor afirma que a ferramenta está sendo avaliada em aplicações reais e os resultados preliminares sugerem que a abordagem é escalável.

- EP_21 - *"The tool is under evaluation on realistic applications and the preliminar results suggest that the approach is scalable".*

Estes resultados mostram que apesar do grande número de experimentos realizados na área (21 dos 24 estudos primários selecionados foram classificados como experimento), apenas 5 (23%) deles apresentaram resultados que sugerem que suas abordagens são escaláveis para problemas do mundo real.

[1] http://www.ruby-lang.org/pt/
[2] http://users.polytech.unice.fr/~rueher/Benchs/FM/

4.2.3 Benefícios Da Utilização de *Constraint Solvers*

Dentre os principais benefícios apontados pelos estudos primários analisados, estão a maior velocidade das abordagens que utilizam técnicas *constraint solving* em relação a outras técnicas, uma vez que os *solvers* podem eliminar rapidamente grandes amostras do espaço de busca.

- EP_24 - *"in practice current state-of-the-art SAT solvers can often determine the satisfiability of boolean formulas with tens of thousands of variables in a reasonable amount of time"*

Abordagens que utilizam técnicas de *constraint solving* também demonstraram gerar casos de teste com melhor qualidade que outras abordagens. Os estudos analisados mostraram que essas abordagens encontraram melhores níveis de cobertura do que outras abordagens.

- EP_02 - *"Experiments on 14 cases taken from real-world Ruby projects show that RuTeG achieves full or higher statement coverage on more cases and does so faster than randomly generated test cases. ... RuTeG could achieve full code coverage in 11 of 14 cases, while the random test case generator only succeeded on 4 occasions. In 10 of 14 cases RuTeG found test cases that showed a significantly higher code coverage than random generation."*
- EP_19 - *"Experimental results using our prototype show that it provides significantly better performance compared to random input generation, in terms of line and branch coverage."*

4.2.4 Limitações Da Utilização de *Constraint Solvers*

Algumas limitações foram identificadas durante a leitura dos estudos primários. A grande maioria dessas limitações ocorrem nas abordagens que esperam como entrada o código fonte do SUT (*System Under Test*).

Essas limitações não foram encontradas em abordagens que esperam de entrada modelos do SUT especificados em linguagens de modelagem (*Modeling Language*, em inglês), já que a maioria dessas limitações são intrínsecas a linguagens de programação. As linguagens de programação mais utilizadas nas abordagens foram as linguagens Java e C. A Figura 7 apresenta a distribuição dos experimentos por linguagem.

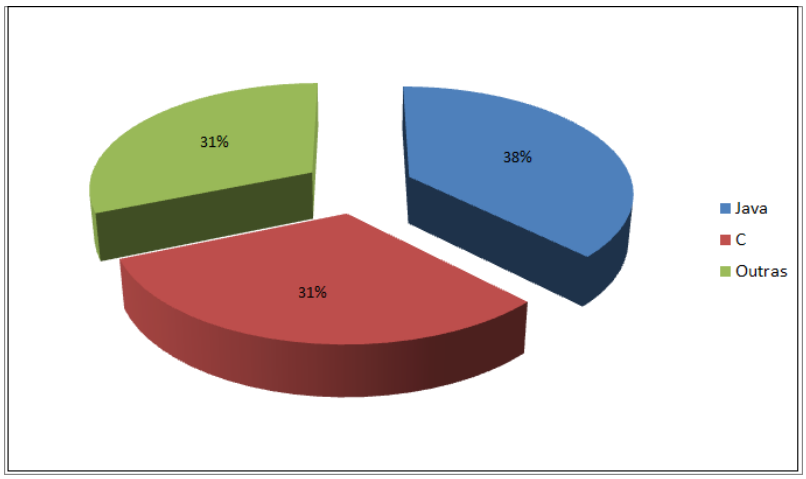

Figura 7: Distribuição por linguagem de programação.

Muitas dessas limitações podem ser observadas nas transcrições abaixo:

- EP_02 -*"This approach was tested on a small number of test programs, which showed its applicability. However, this approach is limited to simple dynamic data structures, such as binary trees."*
- EP_04 - *"... input data are restricted to Booleans, integers and arrays of Booleans and integers. Pointers are not handled and only runtime error-free programs are treated (i.e. errors like dividing by zero or exceptions are not handled)."*
- EP_05 - *"Additional challenges are associated with the need to adequately model class hierarchies and inheritance, dynamic arrays, data types including arbitrary-precision integers, strings and pointers."*
- EP_06 - *"Code involving linked data structures with rich invariants (such as circular lists, red-black trees, AVL trees or binomial heaps) is hard to analyze using these techniques."*
- EP_16 - *"..the limitations of many constraint solvers, especially in the presence of floating point arithmetic.."*
- EP_21 - *"...such test input generation techniques run into certain problems when dealing with database-driven programs. Technical problems remain to handle properly function pointers (second-order programming) and recursive calls."*
- EP_24 - *"Our current implementation does not support checking the exceptional behavior of programs or generating inputs with multi-dimensional-array-based com-*

ponents. ...inheritance is fundamental in object-oriented programming. So far, we have not addressed how to utilize class hierarchies in test generation"

Essas limitações podem ser as responsáveis pelo baixo número de experimentos (dentre os estudos primários selecionados) escaláveis para problemas reais, já que muitas dessas situações como utilização de ponteiros, herança, chamadas a banco de dados, chamadas recursivas, etc., são comumente encontradas em sistemas reais.

4.2.5 Tipos de Solvers

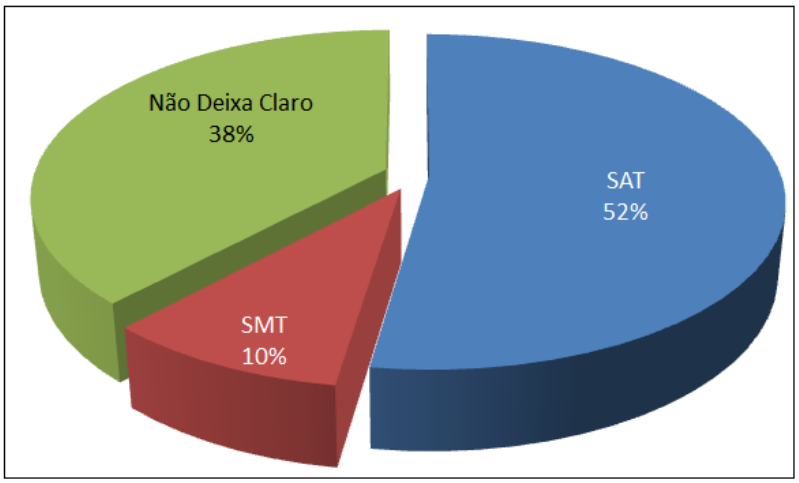

Figura 8: Distribuição por tipo de solver utilizado.

A Figura 8 apresenta a distribuição do tipo de *solver* utilizado nos estudos que são experimentos. Entre os 21 trabalhos classificados como experimento, 8 deles não deixam claro que tipo de solver foi utilizado no experimento. Entre os 13 trabalhos restantes, 11 estudos (85%) utilizaram *SAT-solvers*, 2 estudos utilizaram *SMT-solvers*. Portanto, através da Figura 8 percebe-se a predominância dos *SAT-solvers* entre os solvers mais utilizados.

5 *Considerações Finais*

Este capítulo apresenta as considerações acera do uso de *Constraint Solvers* na geração de dados de teste.

5.1 Conclusões

Os resultados desta pesquisa mostram que a utilização de *constraint solvers* na geração de dados de teste tem sido um tema de pesquisa recorrente, com 24 trabalhos publicados nesta área nos últimos 5 anos.

Apesar de 88% dos estudos primários analisados nessa pesquisa serem classificados como experimento, apenas 19% deles apresentaram ser escaláveis para problemas reais. Isso mostra que apesar do grande número de experimentos realizados na área, apenas alguns encontram-se maduros o suficiente para serem adotadas no mundo real.

A partir dos resultados obtidos, foi observado que apesar de suas limitações, a utilização de *constraint solvers* na geração de dados de teste é uma abordagem promissora, já que apresentou melhores resultados em relação à desempenho e qualidade dos testes, atingindo melhores níveis de cobertura do que outras abordagens.

Além de sumarizar evidências existentes a fim de prover um melhor entendimento sobre os detalhes de como dados de teste são gerados automaticamente a partir da utilização de *constraint solvers*, este trabalho buscou contribuir com a Engenharia de Software baseada em evidências (KITCHENHAM, 2007). O trabalho apresentou um processo detalhado para a realização de um mapeamento sistemático da literatura, processo que pode ser seguido por outros trabalhos.

Para servir de apoio para trabalhos futuros, são listadas abaixo algumas lições aprendidas durante este trabalho:

1. Os termos de busca devem ser definidos antes de iniciar o protocolo, já que os

mesmos guiarão o trabalho;

2. Os termos de busca precisam ser adequados de acordo com as particularidades de cada base da dados que será utilizada na pesquisa. Na maioria das bases de dados é necessária uma licença especial para vizualização dos trabalhos;

3. As buscas retornam muitos trabalhos irrelevantes para o escopo da pesquisa. Os critérios de inclusão e exclusão devem garantir a imparcialidade na seleção;

4. É importante evitar que o processo se torne muito complexo e assim "engessar" o trabalho. Além disso, o processo pode não garantir que trabalhos dentro do escopo da pesquisa sejam selecionados;

5. O protocolo de pesquisa precisa ser claro e objetivo, não deixando margem para ambiguidade. Depois da definição dos principais pontos, o mapeamento pode ser iniciado, e o protocolo evoluir paralelamente;

6. A definição dos critérios de qualidade é um dos maiores desafios na definição do protocolo e é deixado de lado em muitos trabalhos. Essa questão merece um cuidado especial, já que a avaliação pode dar uma maior relevância a estudos de maior qualidade;

5.2 Limitações e Trabalhos Futuros

Apesar da preocupação em utilizar um método rigoroso, esta pesquisa apresentou algumas limitações:

- Devido a restrição de tempo, a pesquisa apenas considerou o portal ACM como base de pesquisa, não levando em consideração bases de dados sugeridas por Kitchenham (2007), como SpringerLink, IEEE e Science Direct.

- Também devido a restrição de tempo e a grande quantidade de estudos primários retornados pelo processo de seleção, foi necessário a realização de um corte temporal, selecionando apenas trabalhos publicados nos últimos 5 anos.

- Apenas o processo de busca foi revisado por mais de um pesquisador, como é sugerido pelos guias atuais. A maior parte da extração dos dados foi realizada apenas pela autor deste trabalho.

As limitações encontradas oferecem caminhos claros para novas pesquisas, como:

- Este estudo pode ser refinado, expandido para outras bases de dados sugeridas por Kitchenham (2007), aumentando o número de estudos primários analisados no mapeamento sistemático.
- O mapeamento sistemático pode ser extendido, levando em consideração também trabalhos publicados a mais de 5 anos, aumentando também o número de estudos primários analisados.
- Para conferir uma maior confiabilidade à pesquisa, pode ser realizada uma pesquisa em segundo nível utilizando o processo *snow-balling* (GREENHALGH; PEACOCK, 2005), ou seja, para todos os trabalhos selecionados, analisar suas referências, na tentativa de identificar trabalhos relevantes não capturados pelos termos de busca.
- Outros pesquisadores podem participar do processo de extração e síntese dos dados, como sugere Kitchenham (2007), garantindo maior confiabilidade aos resultados obtidos pelo mapeamento sistemático.

Referências

BEECHAM, S. et al. Motivation in software engineering: A systematic literature review. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 50, n. 9-10, p. 860–878, ago. 2007. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2007.09.004>.

BONDI, A. B. Characteristics of scalability and their impact on performance. In: *Proceedings of the 2nd international workshop on Software and performance*. New York, NY, USA: ACM, 2000. (WOSP '00), p. 195–203. ISBN 1-58113-195-X. Disponível em: <http://doi.acm.org/10.1145/350391.350432>.

COHEN, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, Durham, v. 20, n. 1, p. 37–46, 1960. Disponível em: <http://epm.sagepub.com/cgi/doi/10.1177/001316446002000104>.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 50, n. 9-10, p. 833–859, ago. 2008. ISSN 0950-5849.

EASTERBROOK, S. et al. *Selecting Empirical Methods for Software Engineering Research*. 2007.

FRUHWIRTH, T.; ABDENNADHER, S. *Principles of Constraint Systems and Constraint Solvers*. 2005.

GREENHALGH, T.; PEACOCK, R. Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. *BMJ (Clinical research ed.)*, v. 331, n. 7524, p. 1064–1065, Nov 5 2005. LR: 20060518; PUBM: Print-Electronic; DEP: 20051017; JID: 8900488; CIN: Evid Based Dent. 2006;7(1):19. PMID: 16557254; RF: 4; 2005/10/17 [aheadofprint]; ppublish.

KITCHENHAM, B. Procedures for performing systematic reviews. *Keele UK Keele University*, Citeseer, v. 33, n. TR/SE-0401, p. 28, 2004.

KITCHENHAM, B. Guidelines for performing systematic literature reviews in software engineering. *Engineering*, ACM Press, v. 2, n. EBSE 2007-001, p. 1051, 2007.

LAKHOTIA, K.; HARMAN, M.; MCMINN, P. Handling dynamic data structures in search based testing. In: *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2008. (GECCO '08), p. 1759–1766. ISBN 978-1-60558-130-9. Disponível em: <http://doi.acm.org/10.1145/1389095.1389435>.

LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. *Biometrics*, International Biometric Society, v. 33, n. 1, p. 159–174, 1977. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/843571>.

MYERS, G. J. *Art of Software Testing*. New York, NY, USA: John Wiley & Sons, Inc., 1979. ISBN 0471043281.

ROSSI, F.; BEEK, P. v.; WALSH, T. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. New York, NY, USA: Elsevier Science Inc., 2006. ISBN 0444527265.

RUSSELL, S. J. et al. *Artificial intelligence: a modern approach*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN 0-13-103805-2.

TRAVASSOS, G. Revisões sistemáticas aplicadas a engenharia de software. *XXI SBES - Brazilian Symposium on Software Engeneering, 2007.*, v. 1, 2007.

APÊNDICE A – Trabalhos Incluídos

Id	Fonte	Título	1º Autor	Ano
EP_01	ACM	Automated test case generation with SMT-solving and abstract interpretation	Jan Peleska	2011
EP_02	ACM	Search-based software testing and test data generation for a dynamic programming language	Stefan Mairhofer	2011
EP_03	ACM	A comparative evaluation of state-of-the-art web service composition testing approaches	Hazlifah Mohd Rusli	2011
EP_04	ACM	A dynamic constraint-based BMC strategy for generating counterexamples	Thierry Gueguen	2011
EP_05	ACM	The challenges of constraint-based test generation	Vitaly Lagoon	2011
EP_06	ACM	Analysis of invariants for efficient bounded verification	Juan P. Galeotti	2010
EP_07	ACM	Abstract path testing with PathCrawler	Nicky Williams	2010
EP_08	ACM	AIana: an AI planning system for test data generation	Stefan J. Galler	2010
EP_09	ACM	Parallel symbolic execution for structural test generation	Matt Staats	2010

EP_10	ACM	PET: a partial evaluation-based test case generation tool for Java bytecode	German Puebla	2010
EP_11	ACM	Representation dependence testing using program inversion	G. Ramalingam	2010
EP_12	ACM	A MILP-based approach to path sensitization of embedded software	José C. Monteiro	2009
EP_13	ACM	MYGEN: automata-based on-line test generator for assertion-based verification	Zeljko Zilic	2009
EP_14	ACM	Using formal specifications to support testing	Jonathan P. Bowen	2009
EP_15	ACM	Efficient solving of structural constraints	Bassem Elkarablieh	2008
EP_16	ACM	Handling dynamic data structures in search based testing	Phil McMinn	2008
EP_17	ACM	Query-Aware Test Generation Using a Relational Constraint Solver	Sarfraz Khurshid	2008
EP_18	ACM	Test generation for graphical user interfaces based on symbolic execution	Dewayne E Perry	2008
EP_19	ACM	A specification-based approach to testing software product lines	Don Batory	2007

EP_20	ACM	Automatic generation of test data generators for synchronous programs: Lutess V2	Ioannis Parissis	2007
EP_21	ACM	Dynamic test input generation for database applications	Ioannis Parissis	2007
EP_22	ACM	Goal-oriented test data generation for pointer programs	Arnaud Gotlieb	2007
EP_23	ACM	Scalable automatic test data generation from modeling diagrams	Ranjith Subramanian	2007
EP_24	ACM	Whispec: white-box testing of libraries using declarative specifications	Danhua Shao	2007

APÊNDICE B – Trabalhos Relevantes

Título	**1º Autor**	**Ano**
A clp framework for computing structural test data	Arnaud Gotlieb	2000
A Dynamic Constraint-Based BMC Strategy For Generating Counterexamples	Hélène Collavizza	2011
A Real-World Benchmark Model for Testing Concurrent Real-Time Systems in the Automotive Domain	Jan Peleska	2011
Automated Test Case Generation with SMT-Solving and Abstract Interpretation	Jan Peleska	2011
Automatic test data generation using constraint logic programming and symbolic execution	Christophe Meudec	2001
Automatic Test Data Generation using Constraint Solving Techniques	Arnaud Gotlieb	1998
Test Data Generation of Bytecode by CLP Partial Evaluation	Elvira Albert	2009

...