

Teste de Software

1

Geração Automática de Dados de Teste

Marcio Delamaro
Francisco Carlos Souza

Geração de Dados de Teste – O que é?

2

É um processo de identificação de **dados do domínio de entrada válidos** para um determinado programa de acordo com os **critérios de teste**.

Geração de Dados de Teste

3

- **Dado de Teste:**

- São as entradas requeridas para um determinado programa.

- **Caso de Teste:**

- Um par formado por um dado de teste e o resultado esperado para a execução do programa.

- **Ex:**

$DT_1: \{5, 4, 3\}$

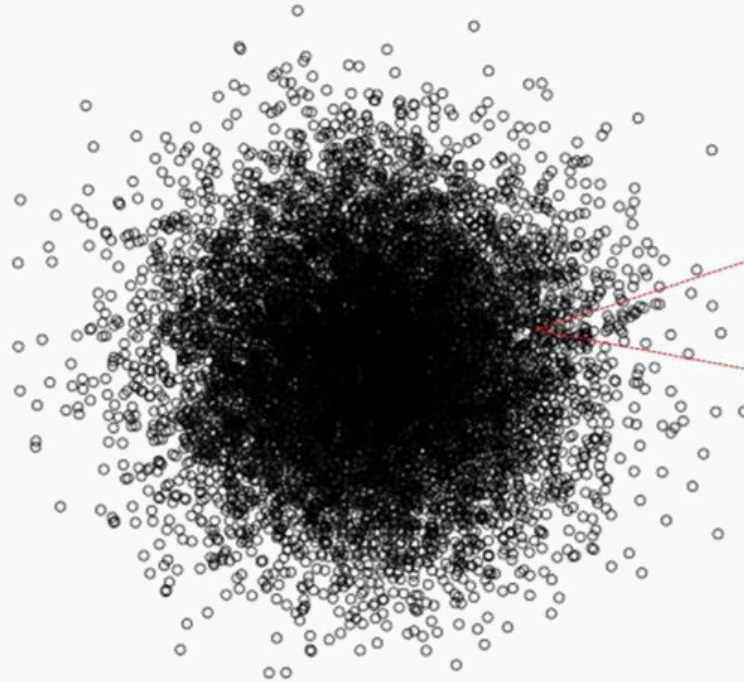
$CT_1: \{(5, 4, 3) \text{ (Escaleno)}\}$

```
a = input('Digite o tamanho do primeiro lado: ')
b = input('Digite o segundo lado: ')
c = input('Digite o terceiro lado: ')

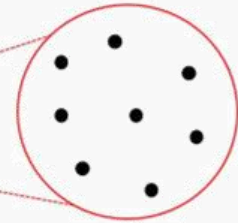
if a + b > c:
    if a == b and a == c:
        print 'Triângulo Equilátero'
    elif a == b or b == c or a == c:
        print 'Triângulo Isósceles'
    elif a != b and c or b != a and c or a != c:
        print 'Triângulo Escaleno'
else:
    print 'É impossível ser um triângulo'
```

Geração de Dados de Teste – Processo

4



Domínio de entrada



Conjunto de dados de teste

Geração de Dados de Teste – Por que automatizar?

5

- Na Prática....

1. Programador implementa a funcionalidade



Geração de Dados de Teste – Por que automatizar?

6

- Na Prática....

2. O testador inicia os testes das funcionalidades implementadas e



Geração de Dados de Teste – Por que automatizar?

7

- Na Prática....

3. Gera os dados e executa os testes **MANUALMENTE**



Geração de Dados de Teste

8

Por que, então, **NÃO** automatizar essa atividade?

Geração – Manual x Automática

9

Geração Manual	Geração Automática
Muitos casos de teste	Poucos casos de teste
Muito trabalhosa e complexa	Reduz significativamente o esforço
Demanda muito tempo	Mais rápida
Dispendiosa	O custo é reduzido
Sujeito a falhas	Alto grau de confiabilidade
	Detecção de regressões

Geração – Manual x Automática

10

- **Menor envolvimento humano**

- Humanos não são bons em realizar a **mesma tarefa repetidamente**;
- Humanos não são bons em realizar a **mesma tarefa igualmente todas as vezes**;
- Tarefas realizadas por humanos estão **sujeitas a falhas**;
- **Stress e cansaço** mascaram **falhas**.

Com a geração automática é possível gerar vários conjuntos de dados de teste de acordo com um critério sem ocasionar exaustão.

Com a automatização, o teste é executado igualmente todas as vezes garantindo maior confiabilidade.

Geração de Dados de Teste Automatizada

11

É o uso de **técnicas** e **ferramentas** combinado com **critérios de teste** para auxiliar no processo de produção de dados de teste.

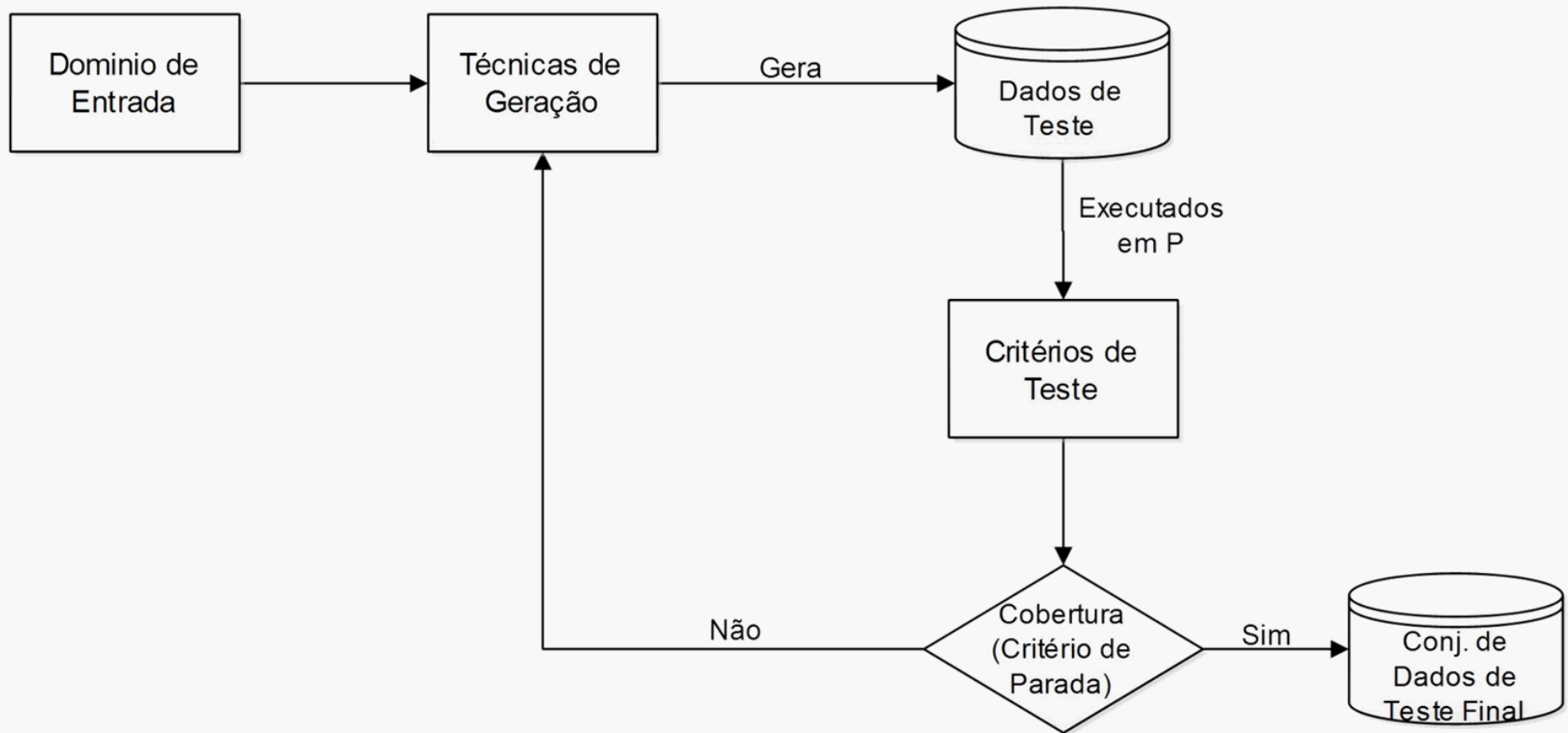
Geração de Dados de Teste Automatizada

12

- É fundamental para o teste de software, pois os **dados de teste** determinam a **qualidade do teste** durante sua execução;
- É uma tarefa **desejável**, porem **não trivial**, em função da **complexidade** de determinados domínios:
 - Tamanho do programa
 - Caminho ausente
 - Caminhos não executáveis
 - Mutantes equivalentes

Geração de Dados de Teste Automatizada

13



Técnicas de Geração de Dados de Teste

14

- Execução Simbólica
- Execução Dinâmica
- Sensíveis à defeito (Teste baseado em Restrição)
- Baseado em Casos de Uso
- **Aleatória**
- **Baseada em Busca**

Geração Aleatória

15

- **Objetivo:**
 - Gerar dados de teste aleatoriamente até que uma entrada útil seja encontrada ou um critério seja satisfeito;
- **Não é adequada** para o uso em **programas** ou **critérios de adequação complexos**, pois uma entrada teria que satisfazer requisitos muito específicos.

Geração Aleatória – Por que usar?

16

- Pode **economizar** mais **tempo** e **esforço** que outros métodos/técnicas de geração de dados (se o domínio for bem definido);
- Para realizar testes em **programas mais simples**;
- Para fins de **comparação com outras técnicas**.

Geração Aleatória – Como usar?

17

1. Identificar o domínio de entrada;
2. Selecionar os dados de teste independentemente do domínio;
3. Executar o programa em teste com os dados selecionados. Esses dados constituem um conjunto de teste aleatório;
4. Comparar os resultados com a especificação do programa. O teste é considerado falho se qualquer entrada leva a um resultado incorreto; caso contrário ele é bem sucedido.

Geração Aleatória – Quando e onde usar?

18

- **Quando usar:**

- Se não tiver muito tempo disponível para realizar o teste, mas o programa deve ser testado;
- Para garantir que os testes sejam suficientemente aleatórios para cobrir diferentes especificações;

- **Onde usar:**

- Em qualquer projeto simples;
- Projetos maiores caso haja tempo disponível;

Geração Aleatória – Vantagens x Desvantagens

19

Vantagens	Desvantagens
Relativamente barata	Dados de teste podem ser redundantes
Permite executar mais casos de teste do que manualmente	Não garante cobertura de trechos específicos
Garante bons resultados para programas simples	Impossível recriar um teste se o dado de teste não for armazenado

Técnicas Baseado em Busca

20

- Técnicas provenientes da área de Inteligência Artificial (IA);
- Problemas de otimização.
- Resolução automática
 - Métodos inteligentes
 - Adequação dos dados

Técnicas Baseado em Busca

21

- Conceitos:
 - **Espaço de Busca:** é um conjunto de todas as possíveis soluções;
 - **Busca Local:** opera em um único estado e movimenta-se para a vizinhança desse estado;
 - **Busca Global:** pode operar em vários estados e movimenta-se para diferentes pontos do espaço de busca;
 - **Função objetivo:** avalia quão próximo de uma solução ótima está a solução atual;

Técnicas Baseado em Busca – Por que usar?

22

- Pode ser usado em problemas que **não se conhece a solução**;
- Para testar programas que **não** se tem conhecimento suficiente do **domínio de entrada**;
- São tipicamente usados para gerar **dados de teste mais próximos ao adequado**;
- Para realizar testes em **programas mais complexos**;

Técnicas Baseado em Busca – Como usar?

23

1. Definir o domínio de entrada;
2. Representar as soluções candidatas (dados de teste);
3. Definir uma função objetivo para guiar a buscar;
4. Gerar os dados de teste;
5. Executar e avaliar os dados gerados no programa em teste;
6. Se os dados de teste forem adequados, a geração deve ser finalizada;
7. Caso contrário, melhorar os dados de acordo com a estratégia da técnica utilizada e retornar ao passo 5;

Técnicas Baseado em Busca– Quando e onde usar?

24

- **Quando usar:**

- Quando não existe um método exato de otimização para resolver o problema em análise.;
- São geralmente aplicadas em problemas que não se conhece algoritmo eficiente;

- **Onde usar:**

- Em projetos críticos que necessitam dos melhores dados de teste ou próximo dos melhores;

Técnicas Baseado em Busca

25

- **Heurística**

- Usa as informações coletadas durante a busca;
- Auxilia na decisão de como a solução seguinte deve ser produzida;
- Específica.

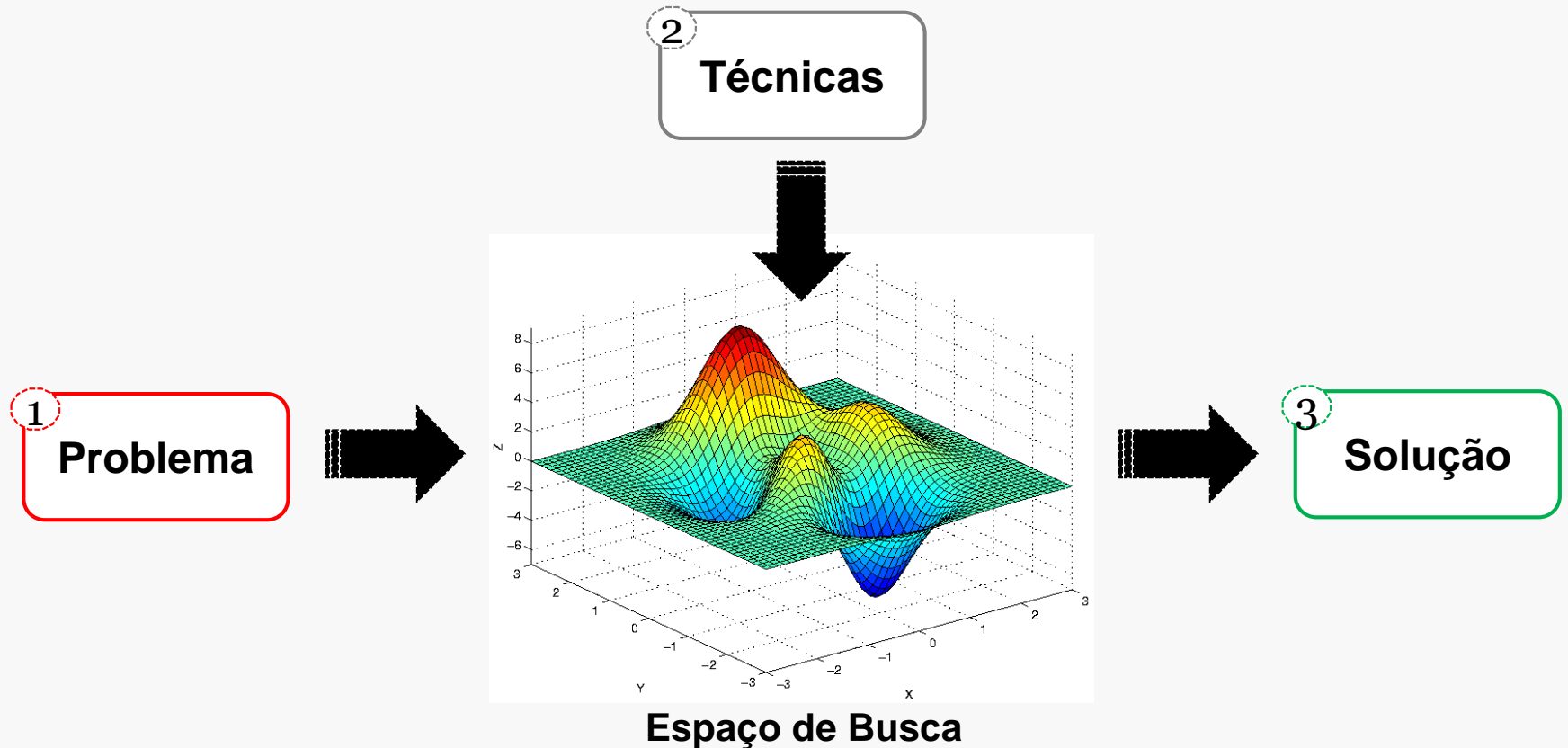
- **Meta-heurística**

- Combina funções objetivos com heurísticas;
- Utiliza estatísticas obtidas de amostras em um espaço de busca;
- Busca Local e Global.

Técnicas Baseado em Busca

26

- Consistem em formular problemas de teste para problemas de busca



Técnicas Baseado em Busca

27

- Algoritmo Genético (*Genetic Algorithm – GA*);
- Subida da Encosta (*Hill Climbing – HC*).
- Otimização por Enxame de Partículas (*Particle Swarm Optimization – PSO*);

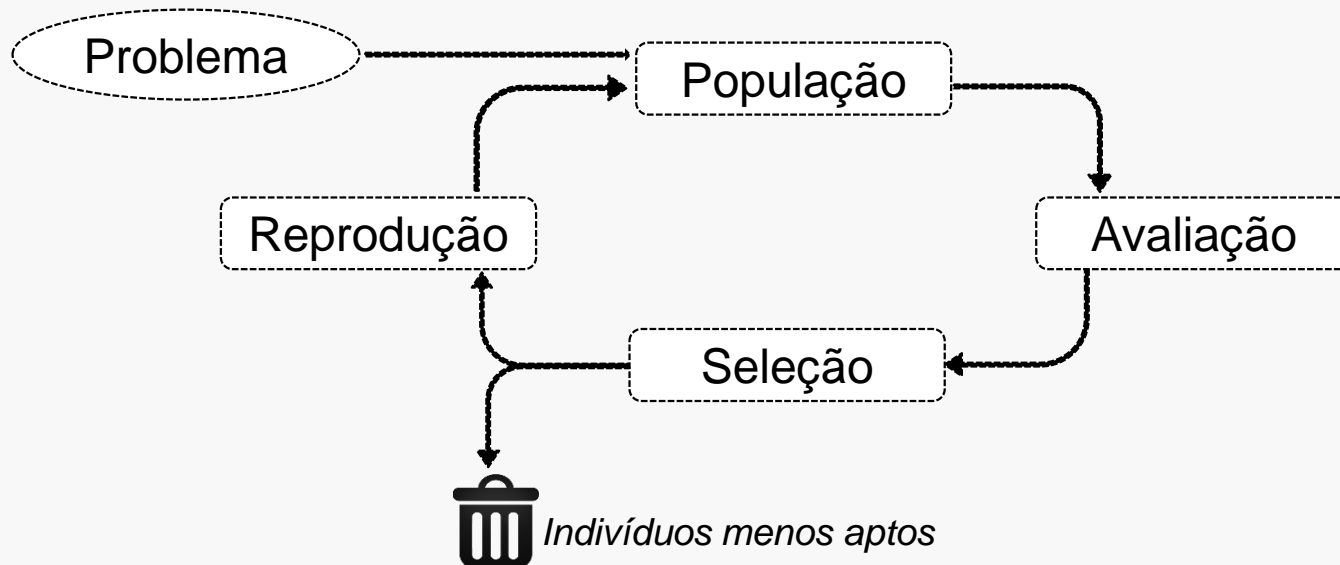
Técnicas Baseado em Busca

28



- **Algoritmo Genético:**

- São inspirados na teoria de evolução de **Charles Darwin**;
- Utiliza um conjunto de soluções candidatas denominadas de **população**, avaliadas por uma **função de *fitness***;



Técnicas Baseado em Busca

29

- **Algoritmo Genético: Vantagens X Desvantagens**

Vantagens	Desvantagens
Útil para problemas complexos	Custo \ alocação de memória
Flexível para ser utilizado em diferentes problemas	Tempo de processamento
Não ficam presos em ótimos locais	Requer um grande número de avaliações de função de fitness
Apresentam bom desempenho para a maioria dos problemas	Dificuldade de achar o ótimo global

Técnicas Baseado em Busca

30



- **Algoritmo Genético: Operadores**

- Seleção
 - ✦ Roleta
 - ✦ Ranking
 - ✦ Amostragem universal estocastica
- Recombinação
 - ✦ Cruzamento com ponto de corte
 - ✦ Máscara binária
- Mutação

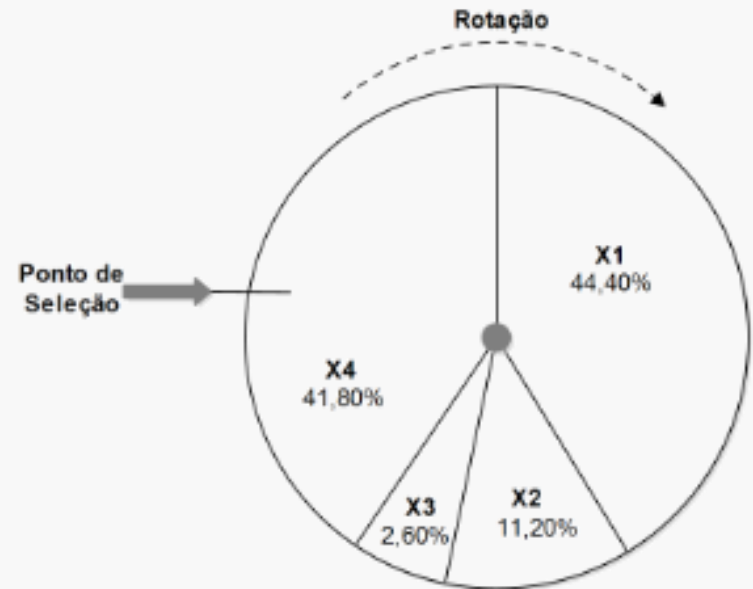
Técnicas Baseado em Busca

31

- **Algoritmo Genético: Operador de Seleção**

- **Roleta:**

- ✦ A seleção ocorre como um sorteio, em que os indivíduos com maior valor de fitness possuem maior probabilidade de serem selecionados.
 - ✦ Esse método simula uma roleta, em que cada indivíduo da população corresponde a uma fatia do total.



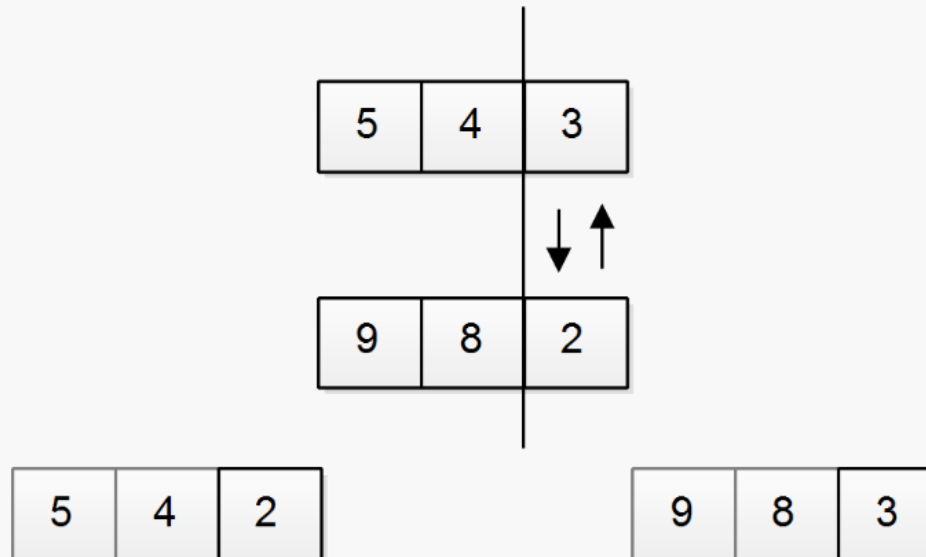
Técnicas Baseado em Busca

32

- **Algoritmo Genético: Operador de Recombinação**

- **Cruzamento:**

- ✦ O objetivo é fazer com que descendentes gerados herdem características dos indivíduos mais aptos da população, juntamente com os métodos de seleção.

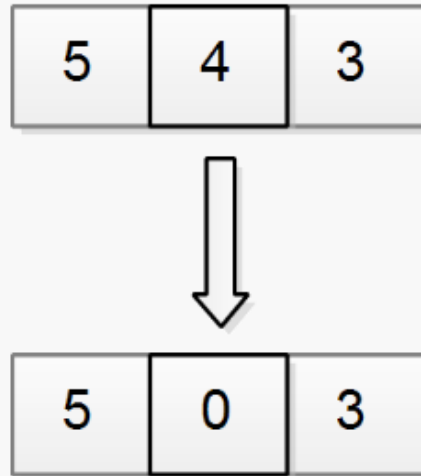


Técnicas Baseado em Busca

33

- **Algoritmo Genético: Operador de Mutação**

- O objetivo é realizar alteração aleatória de um gene do cromossomo;
- Funciona como um operador de manutenção da diversidade genética, pois modifica as características de um individuo aleatoriamente



Técnicas Baseado em Busca

34

- **Subida da Encosta:**

- Analogia da subida progressiva em uma encosta de uma paisagem
 - ✧ Encosta = Vizinhança
 - ✧ Paisagem = Espaço de Pesquisa
- Algoritmo de busca local
- A subida caracteriza o **algoritmo iterativo de melhoria**
 - ✧ Solução inicial aleatória
 - ✧ Melhorando passo-a-passo
 - ✧ Investigação da vizinhança da solução corrente



Técnicas Baseado em Busca

35

- **Subida da Encosta: Vantagens X Desvantagens**

Vantagens	Desvantagens
Flexível e fácil implementação	Grandes chances de ficar preso em ótimos locais
Aplicável em uma variedade de problemas	Depende de uma função objetivo bem definida
Rápida convergência	Não há garantia de encontrar o ótimo global

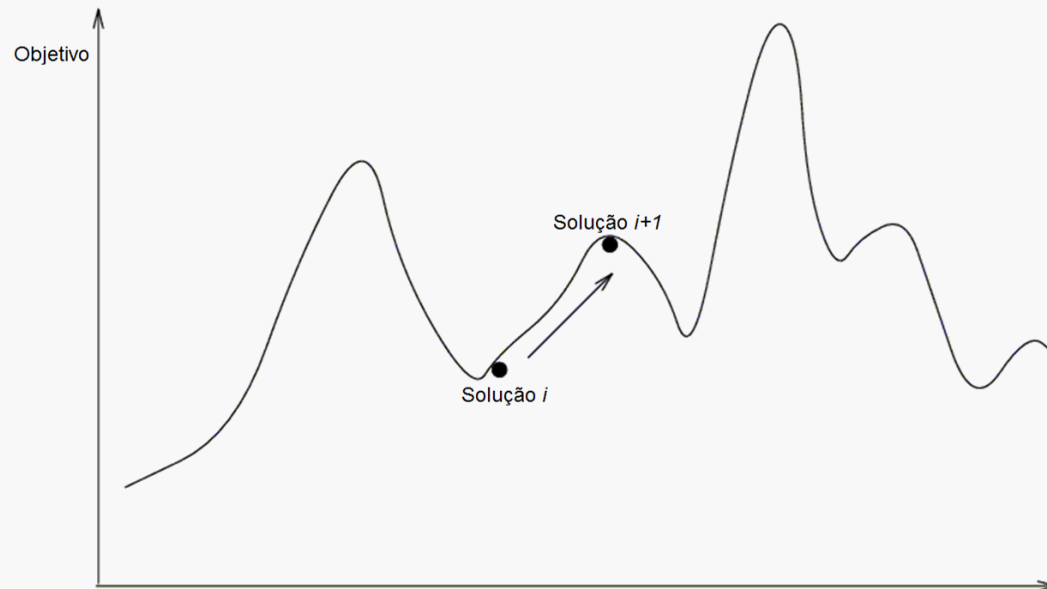
Técnicas Baseado em Busca

36

- **Subida da Encosta:**

- **Estratégia da Busca**

- ✦ Visa a chegar ao pico mais alto apenas examinando os caminhos mais próximos (**vizinhos**) e **termina** quando alcança um pico em que **nenhum vizinho tem o valor mais alto**.

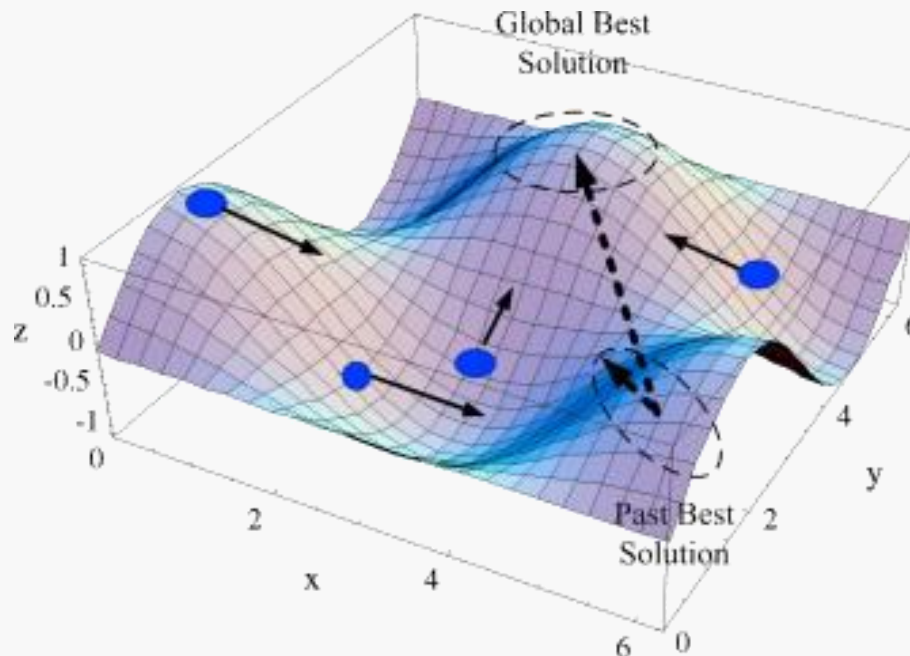


Técnicas Baseado em Busca

37

- **Otimização por Enxame de Partículas:**

- Analogia ao comportamento social de pássaros;
- Utiliza **partículas** que constituem um **enxame** que se movimenta no espaço de busca à procura da **melhor solução**;



Técnicas Baseado em Busca

38

- **Otimização por Enxame de Partículas : Vantagens X Desvantagens**

Vantagens	Desvantagens
Maior exploração do espaço de busca	Custo de processamento
Rápida convergência para o local das melhores soluções	Lento para refinar um solução
Bom desempenho em diversas aplicações	Dificuldade de achar o ótimo global

Ferramentas

39

- AgitarOne
- AutoTest
- CATG
- CAUT
- CREST
- **EvoSuite**
- GRT
- GUITAR
- Jalangi
- JSeft
- Jtest
- **JTExpert**
- KLEE
- MergePoint/Mayhem
- PathCrawler
- Pex
- QuickCheck
- Randoop
- Symbolic PathFinder
- T3
- **AUSTIN**

Ferramentas

40

- **EvoSuite:** <http://www.evosuite.org/>
 - **Objetivo:**
 - ✦ Gerar automaticamente casos de teste com assertivas para classes escritas em **Java**.
 - **Técnica:**
 - ✦ Utiliza **Algoritmo Genético** para gerar e otimizar os conjuntos de testes a fim de satisfazer um critério de cobertura.
 - **Critérios:**
 - ✦ Teste estrutural
 - ✦ Teste de mutação

Ferramentas

41

- **EvoSuite:**
 - **Ano:** 2011
 - **Tipo:** Acadêmica e *Open Source*
 - **Interação:**
 - ✦ Linha de Comando
 - ✦ **Plugin Eclipse**
 - ✦ Plugin IntelliJ IDEA
 - ✦ Plugin Maven



Ferramentas

42

- **AUSTIN:**
 - **Objetivo:**
 - ✦ Gerar dados de teste para programas em **C**
 - **Técnicas:**
 - ✦ Subida da encosta
 - ✦ Busca aleatória.
 - **Critério:**
 - ✦ Teste estrutural



austin-sbst

Ferramentas

43

- **AUSTIN:**
 - **Ano:** 2010
 - **Tipo:** *Open Source*
 - **Interação:**
 - ✦ Linha de Comando



austin-sbst

Ferramentas

44

- **JTExpert:**
 - **Objetivo:**
 - ✦ Gerar automaticamente um conjunto de testes para classes em Java
 - **Técnica:**
 - ✦ Busca aleatória guiada.
 - **Critério:**
 - ✦ Cobertura de ramos

Ferramentas

45

- **JTExpert :**
 - **Ano:** 2015
 - **Tipo:** Acadêmica
 - **Interação:**
 - ✦ Linha de Comando

Geração de Dados de Teste Automatizada

46

- Dificuldades:
 - *Arrays* e Ponteiros;
 - Objetos;
 - Loops;
 - Caminhos não executáveis;
 - Mutantes equivalentes

Geração de Dados de Teste usando a ferramenta EvoSuite


EvoSuite – Instalação

48


- Abrir o Eclipse
- Help > Install New Software >
- <http://www.evosuite.org/update/> > Add >
- Name > http://www.evosuite.org/update > Ok >
- Select EvoSuite JUnit Test Generation >
- Next


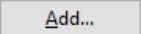
EvoSuite - Instalação

49


 Install

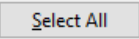
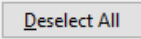
Available Software
Check the items that you wish to install.



Work with:  

Find more software by working with the ["Available Software Sites"](#) preferences.

Name	Version
<input checked="" type="checkbox"/>  EvoSuite JUnit Test Generation	1.0.2.201512261016

  1 item selected

Details

☐ Show only the latest versions of available software


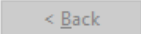
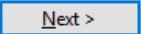
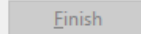
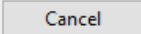
☐ Hide items that are already installed

☐ Group items by category

What is [already installed](#)?

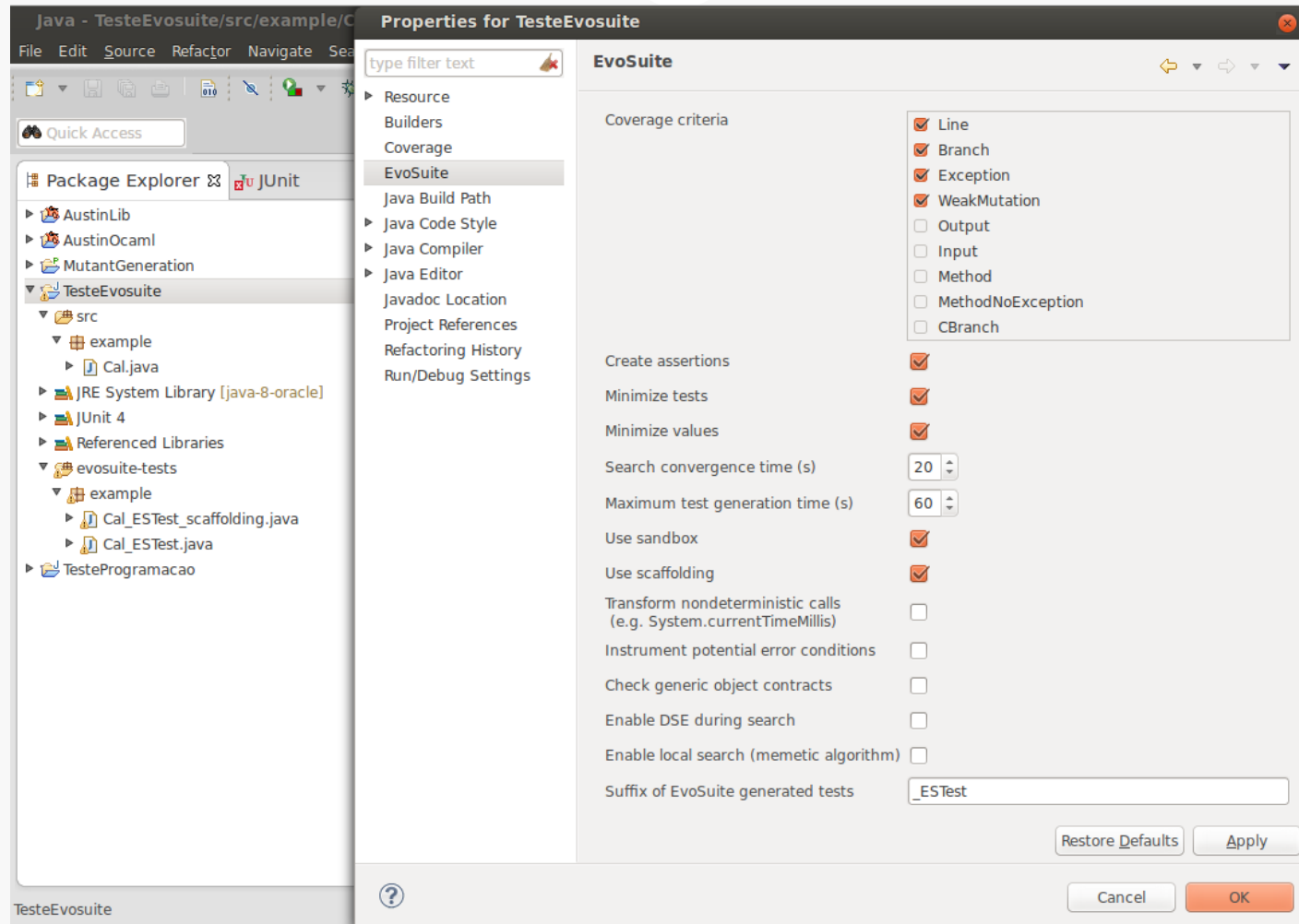
☐ Show only software applicable to target environment

☐ Contact all update sites during install to find required software

EvoSuite - Configuração

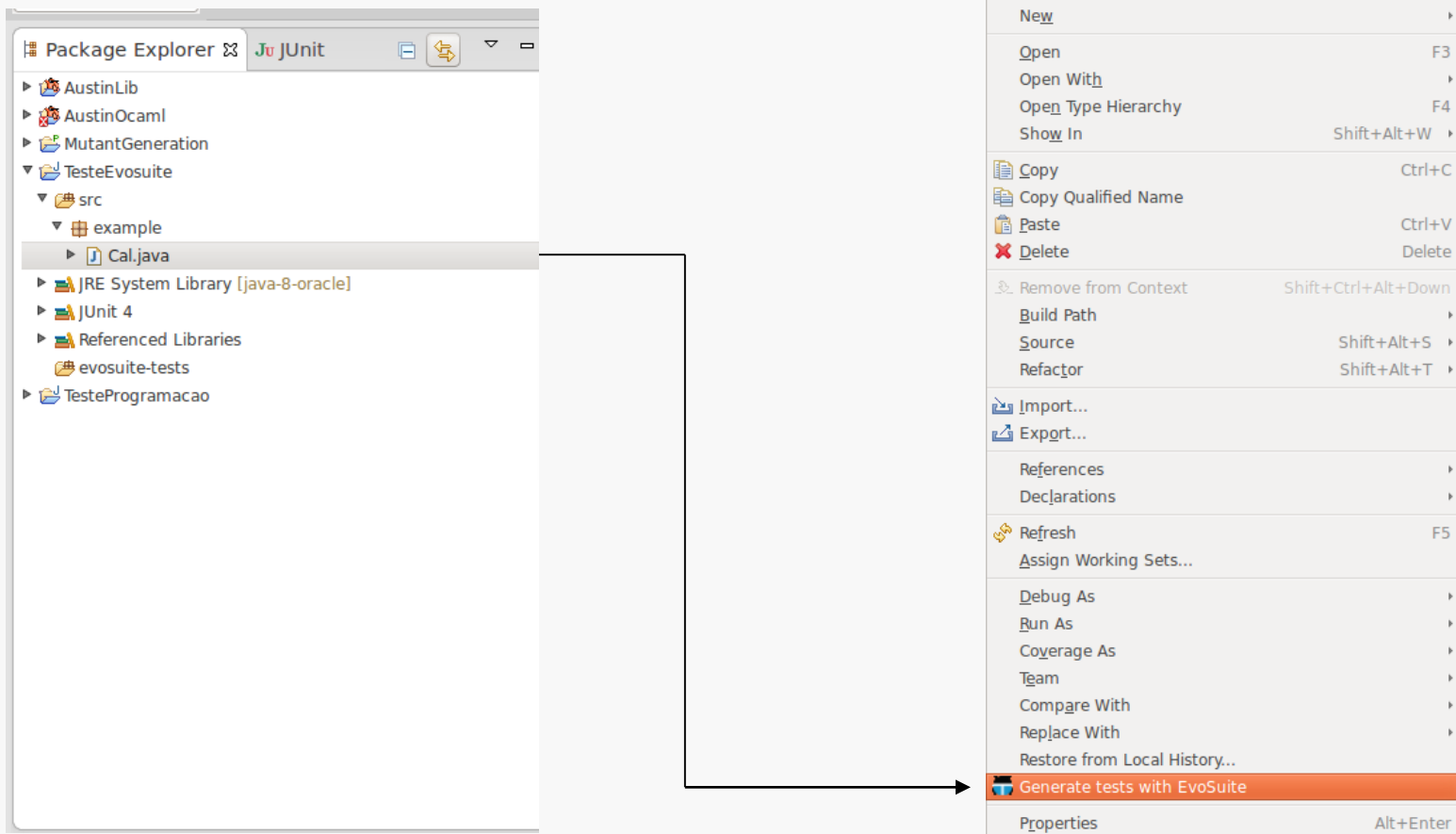
50



EvoSuite - Execução

51

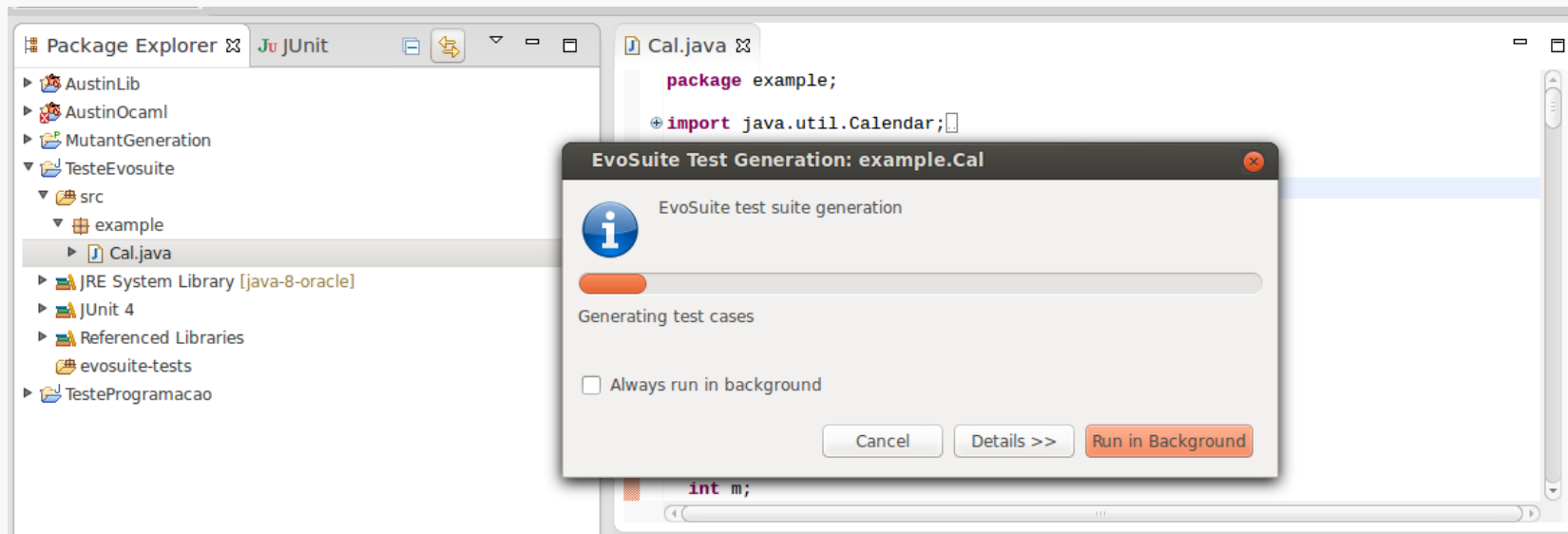
- Selecionar a classe que será testada (Cal.java)



EvoSuite - Execução

52

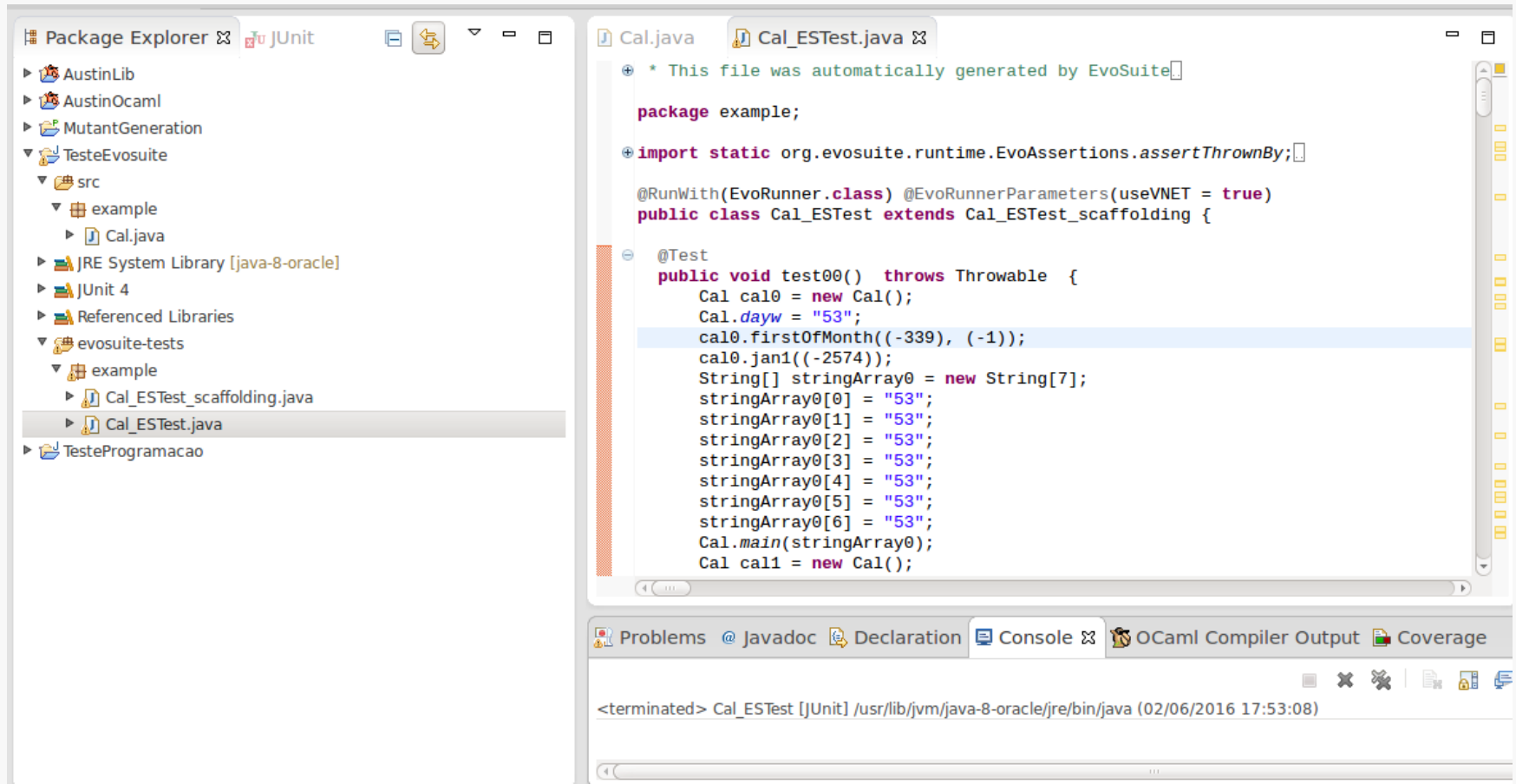
- Geração do caso de Teste:
 - Cal_ESTest.java



EvoSuite - Execução

53

- Execução a classe Cal_ESTest.java com o JUnit



EvoSuite - Execução

54

- Resultados do teste:

The screenshot displays the Eclipse IDE interface during the execution of an EvoSuite test. The Package Explorer on the left shows the test results for the package `example`. The test `example.Cal_ESTest` [Runner: JUnit 4] (0,449 s) is highlighted. Below it, a list of tests is shown with their execution times: `test06` (0,047 s), `test17` (0,003 s), `test07` (0,002 s), `test18` (0,001 s), `test08` (0,001 s), `test19` (0,001 s), `test09` (0,015 s), `test20` (0,320 s), and `test10` (0,001 s). The Failure Trace section shows the error details for `test06`: `java.lang.AssertionError: Expecting exception: ArrayIndexOutOfBoundsException` at `org.junit.Assert.fail(Assert.java:88)` and `example.Cal_ESTest.test06(Cal_ESTest.java:144)`.

The main editor shows the source code of `Cal.java` and `Cal_ESTest.java`. The `Cal_ESTest.java` file is highlighted, showing the test method `test00()` which is annotated with `@RunWith(EvoRunner.class)` and `@EvoRunnerParameters(useVNET = true)`. The test method `test00()` is annotated with `@Test` and `throws Throwable`. The test method `test00()` is annotated with `@Test` and `throws Throwable`.

The Console at the bottom displays the output of the test run, including the failure trace for `test06`. The output shows the following log messages:

```
<terminated> Cal_ESTest [JUnit] /usr/lib/jvm/java-8-oracle/jre/bin/java (02/06/2016 17:53:08)
17:53:08.776 [main] INFO o.evosuite.runtime.agent.AgentLoader - dynamically loading javaagent
17:53:08.782 [main] INFO o.evosuite.runtime.agent.AgentLoader - Using JavaAgent in /opt/eclipse/plugins/org.
17:53:08.786 [main] INFO o.evosuite.runtime.agent.AgentLoader - Classpath: /home/fcarlos/workspace/evosuite/
17:53:08.787 [main] INFO o.evosuite.runtime.agent.AgentLoader - Going to attach agent to process 25853
17:53:09.030 [Attach Listener] INFO o.e.runtime.agent.InstrumentingAgent - Executing agentmain of JavaAgent
17:53:09.280 [Thread-0] DEBUG o.e.r.agent.TransformerForTests - Going to instrument: example.Cal
17:53:09.301 [Thread-0] INFO o.e.r.i.MethodCallReplacementClassAdapter - Adding mock interface to class exam
```

Geração de Dados de Teste usando a ferramenta AUSTIN

AUSTIN – Instalação

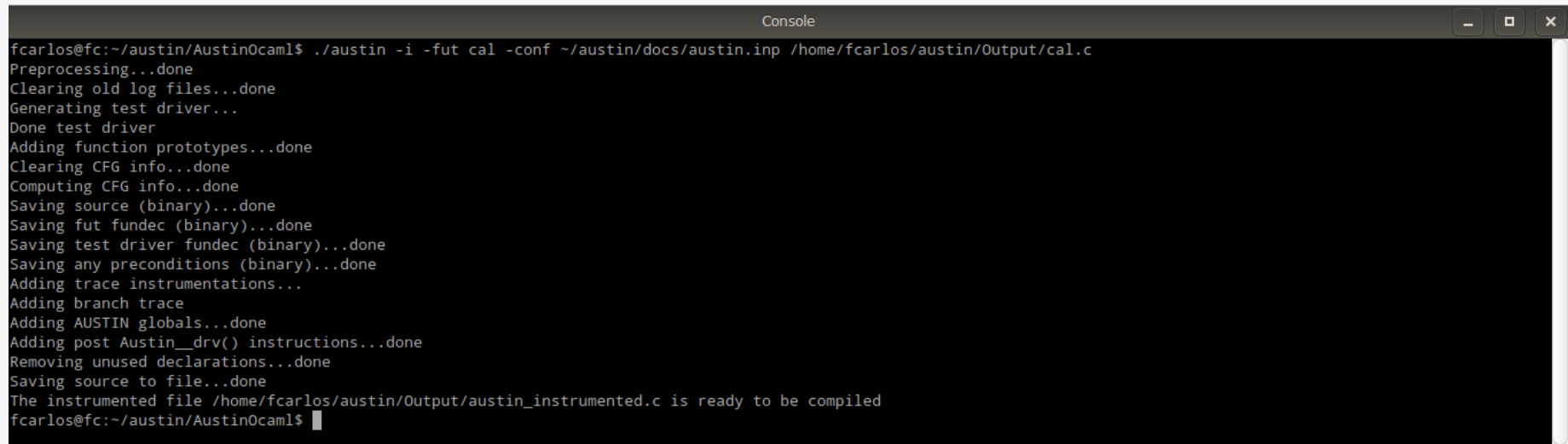
56

- Instalar Ocaml versão 3.11.2 ou superior
 - <http://caml.inria.fr/download.en.html>
- Instalar CIL versão 1.3.7
 - <http://sourceforge.net/projects/cil>
- Configurar a variável de ambiente CILHOME de acordo com o diretório que o CIL foi instalado
 - Ex: `CILHOME=/Users/fcarlos/cil-1.3.7/`

AUSTIN – Execução

57

- **Instrumentação:**



```
Console
fcarlos@fc:~/austin/Austin0caml$ ./austin -i -fut cal -conf ~/austin/docs/austin.inp /home/fcarlos/austin/Output/cal.c
Preprocessing...done
Clearing old log files...done
Generating test driver...
Done test driver
Adding function prototypes...done
Clearing CFG info...done
Computing CFG info...done
Saving source (binary)...done
Saving fut fundec (binary)...done
Saving test driver fundec (binary)...done
Saving any preconditions (binary)...done
Adding trace instrumentations...
Adding branch trace
Adding AUSTIN globals...done
Adding post Austin_drv() instructions...done
Removing unused declarations...done
Saving source to file...done
The instrumented file /home/fcarlos/austin/Output/austin_instrumented.c is ready to be compiled
fcarlos@fc:~/austin/Austin0caml$
```

Example:
~~~~~

To instrument your source files a.c, b.c and d.c call

```
./austin -i -fut function-to-test -conf <path to austin configuration file> a.c b.c d.c
```

AUSTIN will produce a single instrumented source file (called austin\_instrumented.c). Compile the instrumented source with a C compiler e.g.

# AUSTIN – Execução

58

- **Compilação:**

```
Console
fcarlos@fc:~/austin/AustinOcaml$ gcc -c /home/fcarlos/austin/Output/austin_instrumented.c
```

```
Console
fcarlos@fc:~/austin/AustinOcaml$ g++ -o /home/fcarlos/austin/Output/cal austin_instrumented.o -L ~/austin/AustinLib/ -lAustinLib -ldl -lm
fcarlos@fc:~/austin/AustinOcaml$
```

AUSTIN will produce a single instrumented source file (called `austin_instrumented.c`). Compile the instrumented source with a C compiler e.g.

```
gcc -c austin_instrumented.c
```

Then link the object file using a C++ compiler

```
g++ -o sut.exe austin_instrumented.o -L<path to AustinLib> -lAustinLib -ldl -lm
```

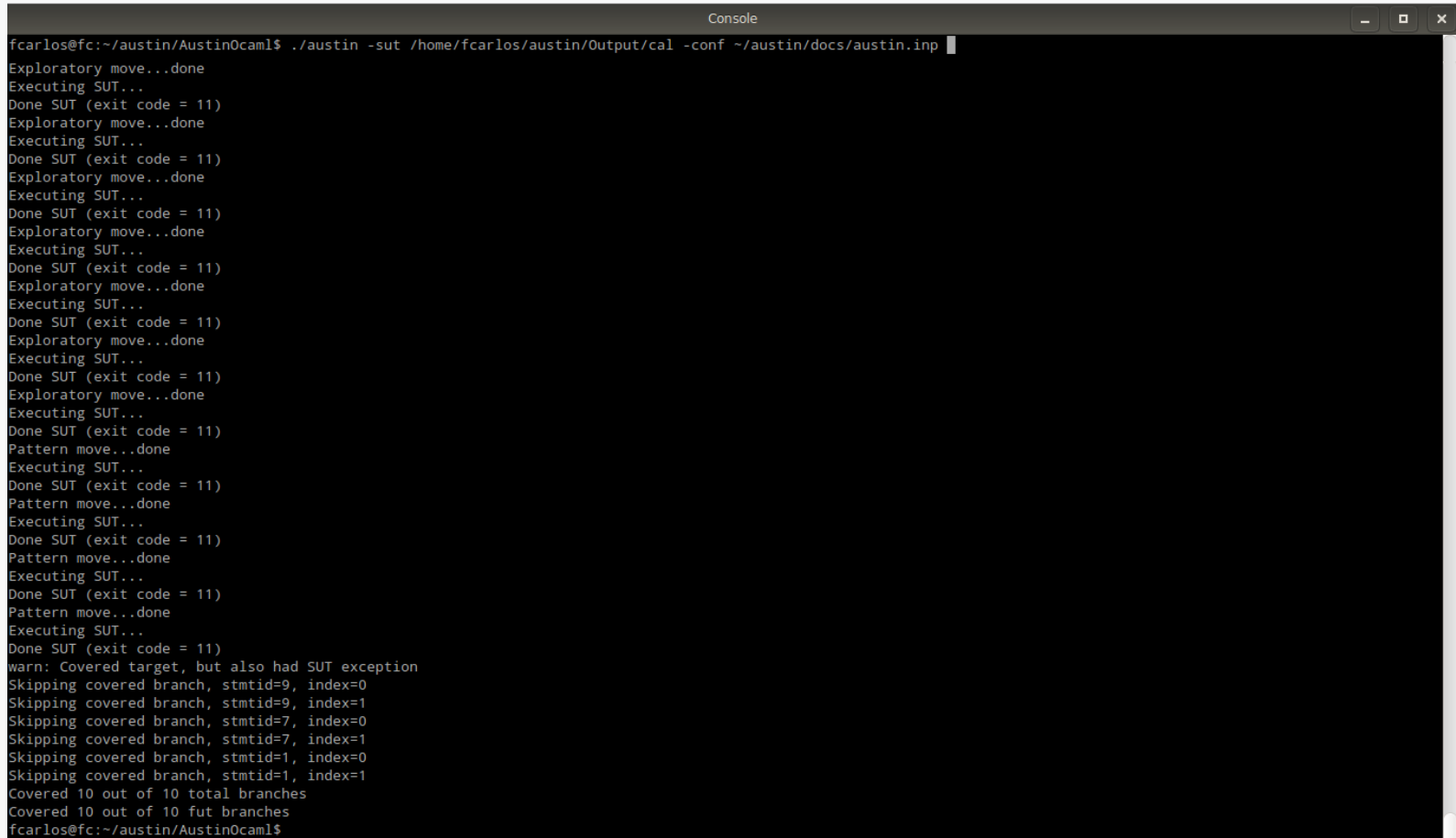
Then run AUSTIN

```
./austin -sut <path to sut.exe> -conf <path to austin configuration file -- austin.inp>
```

# AUSTIN – Execução

59

- Geração dos dados e execução dos testes:



```
fcarlos@fc:~/austin/AustinOcaml$ ./austin -sut /home/fcarlos/austin/Output/cal -conf ~/austin/docs/austin.inp
Exploratory move...done
Executing SUT...
Done SUT (exit code = 11)
Exploratory move...done
Executing SUT...
Done SUT (exit code = 11)
Exploratory move...done
Executing SUT...
Done SUT (exit code = 11)
Exploratory move...done
Executing SUT...
Done SUT (exit code = 11)
Exploratory move...done
Executing SUT...
Done SUT (exit code = 11)
Exploratory move...done
Executing SUT...
Done SUT (exit code = 11)
Exploratory move...done
Executing SUT...
Done SUT (exit code = 11)
Exploratory move...done
Executing SUT...
Done SUT (exit code = 11)
Pattern move...done
Executing SUT...
Done SUT (exit code = 11)
Pattern move...done
Executing SUT...
Done SUT (exit code = 11)
Pattern move...done
Executing SUT...
Done SUT (exit code = 11)
Pattern move...done
Executing SUT...
Done SUT (exit code = 11)
Pattern move...done
Executing SUT...
Done SUT (exit code = 11)
warn: Covered target, but also had SUT exception
Skipping covered branch, stmtid=9, index=0
Skipping covered branch, stmtid=9, index=1
Skipping covered branch, stmtid=7, index=0
Skipping covered branch, stmtid=7, index=1
Skipping covered branch, stmtid=1, index=0
Skipping covered branch, stmtid=1, index=1
Covered 10 out of 10 total branches
Covered 10 out of 10 fut branches
fcarlos@fc:~/austin/AustinOcaml$
```

# AUSTIN – Relatório

60

csvBranchCoverage\_1.csv - LibreOffice Calc

Arquivo Editar Exibir Inserir Formatar Ferramentas Dados Janela Ajuda

Liberation Sans 10

H20

|    | A             | B            | C              | D                   | E                  | F | G | H | I | J |
|----|---------------|--------------|----------------|---------------------|--------------------|---|---|---|---|---|
| 1  | <u>nodeid</u> | <u>index</u> | <u>covered</u> | <u>fitnessEvals</u> | <u>numExecuted</u> |   |   |   |   |   |
| 2  | 18            | 1            | 1              | 477                 | 1                  |   |   |   |   |   |
| 3  | 18            | 0            | 1              | 477                 | 60658              |   |   |   |   |   |
| 4  | 10            | 1            | 1              | 2034                | 1                  |   |   |   |   |   |
| 5  | 10            | 0            | 1              | 96                  | 28                 |   |   |   |   |   |
| 6  | 9             | 1            | 1              | 143                 | 957                |   |   |   |   |   |
| 7  | 9             | 0            | 1              | 96                  | 29                 |   |   |   |   |   |
| 8  | 7             | 1            | 1              | 143                 | 986                |   |   |   |   |   |
| 9  | 7             | 0            | 1              | 1                   | 1617               |   |   |   |   |   |
| 10 | 1             | 1            | 1              | 1                   | 2603               |   |   |   |   |   |
| 11 | 1             | 0            | 1              | 136                 | 4                  |   |   |   |   |   |
| 12 |               |              |                |                     |                    |   |   |   |   |   |

# Pesquisas em Geração de Dados de Teste

61

- **Problemas:**

- Redução de tempo e custo;
- Produção de dados de teste adequados;
- Caminhos não executáveis;
- Tratamento de mutantes equivalentes;

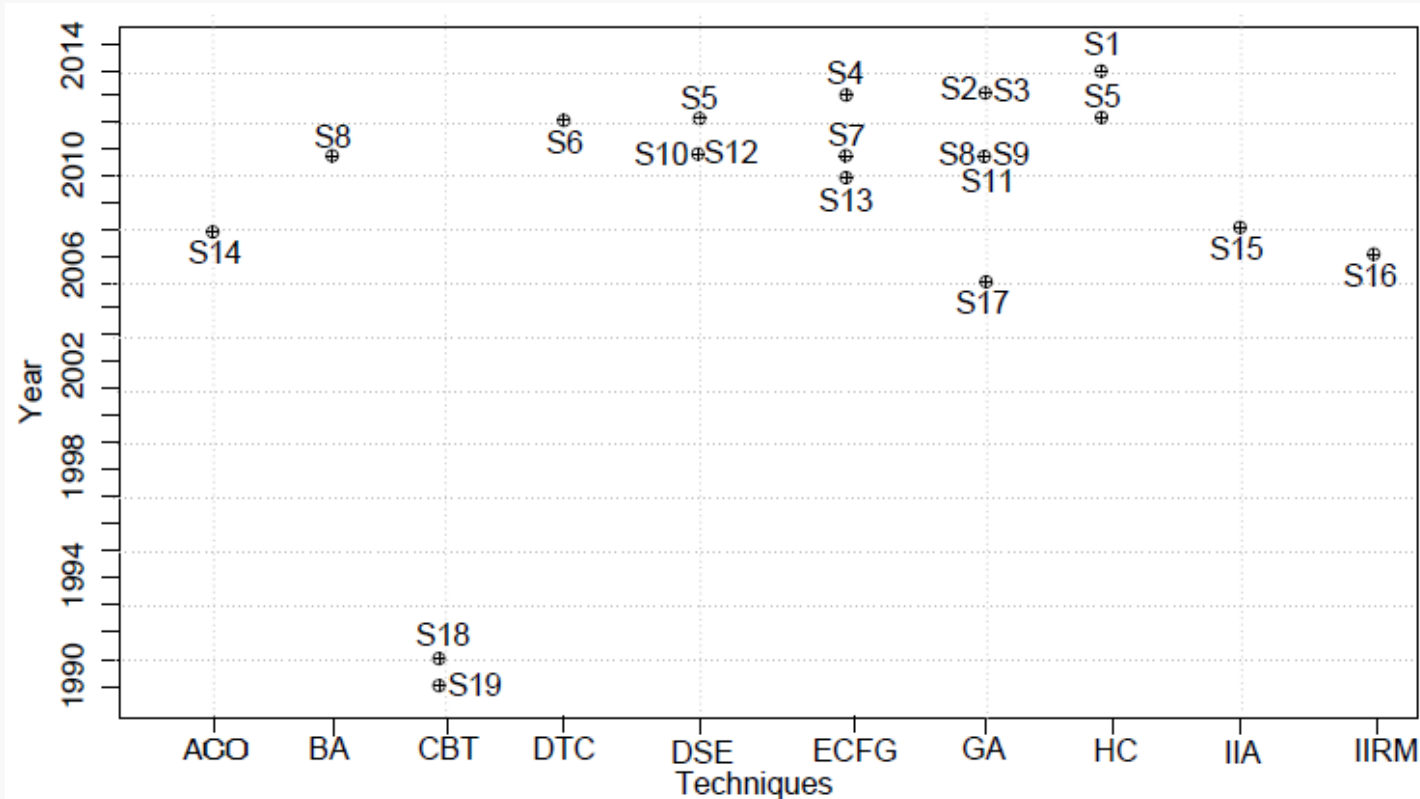
- **Contextos:**

- Teste estrutural;
- Teste funcional;
- **Teste de mutação**

# Pesquisas em Geração de Dados de Teste

62

- Histórico da geração de dados para teste de mutação



# Pesquisas em Geração de Dados de Teste

63

- **Geração de dados de teste para teste de mutação**
  - **Técnica de Busca**
    - ✦ Subida da Encosta
    - ✦ Três funções de fitness
  - **Critérios de Teste**
    - ✦ Teste de Mutação
    - ✦ Teste Estrutural
  - **Ferramentas**
    - ✦ AUSTIN
    - ✦ PROTEUM

# Pesquisas em Geração de Dados de Teste

64

- **Geração de dados de teste para teste de mutação**

