4
Uma Linguagem Baseada em Máquinas de Estado

4.1.
A Linguagem

Acredita-se nesse trabalho que características reativas e fortemente baseadas em modelos tornam necessária a criação de uma linguagem específica capaz de representar de forma mais eficiente os comportamentos do sistema. A linguagem proposta fornece uma forma de especificar máquinas de estado que descrevam os comportamentos da aplicação. As máquinas de estado possuem três elementos principais:

- Estados: Representam a situação do(s) módulo(s) sob teste em um determinado instante.
- Eventos: Situações que ocorrem no sistema e podem ou não causar mudanças de estado no(s) módulo(s) sob teste.
- Transições: Definem as possíveis mudanças de estado no(s) módulo(s) sob teste, levando a máquina de um estado para outro quando ocorre um determinado evento.

As máquinas de estado definidas por esta ferramenta são do tipo determinístico, ou seja, dados um estado e um evento apenas uma transição pode ocorrer. Caso ocorra um evento desconhecido, ou seja, que não está em nenhuma das transições definidas para o estado em que a máquina se encontra, este será descartado. A definição das máquinas de estado é feita através de um script que contém estados, eventos e as possíveis transições.

Um exemplo de aplicação para a linguagem proposta vem de uma das situações de não conformidade verificadas pelo controle de alarmes do sistema alvo, o controle de velocidade. De forma simplificada, existem os seguintes estados, mutuamente exclusivos, para um veículo em um determinado momento: em alta velocidade, e em velocidade regular. Os eventos que causam as transições entre esses estados são obtidos a partir das informações recebidas do equipamento GPS do veículo, podendo ser uma informação de alta velocidade ou uma

informação de velocidade regular. Portanto, um veículo no estado velocidade regular para o qual se recebe uma informação de velocidade acima da permitida deve ir para o estado de alta velocidade. Caso a informação recebida para esse veículo indique velocidade permitida, o veículo continuará no seu estado anterior, de velocidade regular. De forma semelhante, um veículo que se encontre no estado de alta velocidade deve ir para o estado de velocidade regular caso se receba informações de velocidade regular, ou deve permanecer no estado alta velocidade caso a informação seja de alta velocidade. Esse comportamento é descrito na máquina de estados da figura 17.

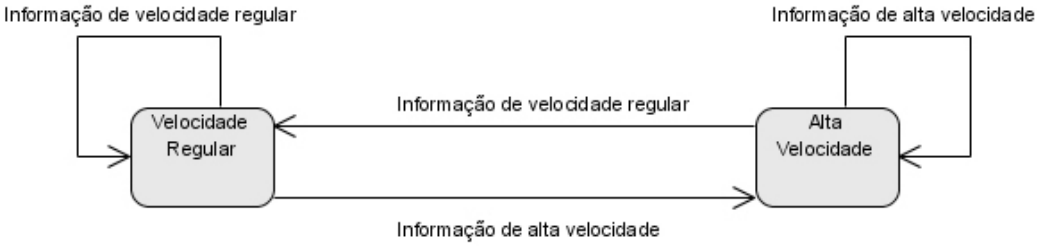

Figura 17: Máquina de estados descrevendo o controle de alarmes de velocidade

A linguagem proposta foi uma extensão da linguagem mostrada no exemplo da seção 3.4.1, adicionando os comandos necessários para a manipulação de máquinas de estado. Além dos comandos definidos para a declaração da máquina de estados, através da declaração de seus estados e possíveis transições, mecanismos são fornecidos para que se force a máquina a ir a um determinado estado e para a geração de eventos que causem as transições. Tais operações podem ser realizadas através dos seguintes comandos:

- **Existe transição <estado de origem> <evento> <estado de destino>**: Define uma transição para o estado de destino sempre que a máquina se encontrar no estado de origem e o evento especificado ocorrer. Além de criar a transição, este comando também está declarando os estados e evento envolvidos na transição.
- **Vai para <estado>**: É o comando disponível para forçar que a máquina de estados esteja em um estado específico. Esse comando é útil para criar as pré-condições para a realização de determinado caso de testes.

- **Acontece evento <evento>**: Faz com que ocorra o evento especificado. Dependendo do estado em que a máquina se encontra e das transições definidas tendo esse estado como origem este comando pode gerar uma transição. Caso uma transição seja gerada, é verificado posteriormente se o estado alcançado pelo módulo sob teste após a transição é compatível com aquilo que foi declarado para a máquina de estados no *script* de teste, verificando dessa forma se o comportamento que era esperado de fato ocorreu.

Após a execução do *script* de testes, é gerado um relatório contendo as linhas do *script*, com seus comandos e comentários, e para cada comando executado são apresentados os resultados da execução, com indicação de falha ou sucesso, mensagens geradas pelo módulo específico de testes e, caso exista, a exceção não capturada que impediu a sua execução. Espera-se que esse relatório possua informações suficientes para auxiliar na detecção dos defeitos causadores das falhas encontradas durante a execução.

4.2. Instanciação do Módulo de Linguagem

Para a criação da instância da ferramenta contendo a linguagem de máquinas de estado foi necessário criar um novo módulo de linguagem, capaz de gerar os novos elementos no modelo de *script*, e novos componentes de execução para os novos elementos de *script* definidos. A figura 18 mostra o arquivo de configuração com as alterações necessárias para a criação dessa instância.

```
1  SCRIPT_READER_CLASS_NAME=br.pucrio.inf.jautotest.reader.DefaultScriptReader
2  TEST_ENGINE_CLASS_NAME=br.pucrio.inf.jautotest.test.DefaultTestEngine
3  REPORT_PRINTER_CLASS_NAME=br.pucrio.inf.jautotest.report.OutputFileReportPrinter
4
5  ELEMENT_PREFIX_LIST=//->recupera->caso->executa>procedimento->chama->vai para->acontece
   evento>existe transição
6
7  SCRIPT_ELEMENT_PARSE_CLASS_//=br.pucrio.inf.jautotest.reader.command.CommandParseComment
8  SCRIPT_ELEMENT_PARSE_CLASS_caso=br.pucrio.inf.jautotest.reader.command.CommandParseTestCase
9  SCRIPT_ELEMENT_PARSE_CLASS_executa=br.pucrio.inf.jautotest.reader.command.CommandParseTestCommand
10 SCRIPT_ELEMENT_PARSE_CLASS_recupera=br.pucrio.inf.jautotest.reader.command.CommandParseRecovery
11 SCRIPT_ELEMENT_PARSE_CLASS_procedimento=
   br.pucrio.inf.jautotest.reader.command.CommandParseProcedureDefinition
12 SCRIPT_ELEMENT_PARSE_CLASS_chama=br.pucrio.inf.jautotest.reader.command.CommandParseProcedureCall
13 SCRIPT_ELEMENT_PARSE_CLASS_vaisppara=br.pucrio.inf.jautotest.reader.command.CommandParseForceState
14 SCRIPT_ELEMENT_PARSE_CLASS_acontecespevento=
   br.pucrio.inf.jautotest.reader.command.CommandParseGenerateEvent
15 SCRIPT_ELEMENT_PARSE_CLASS_existesptransição=
   br.pucrio.inf.jautotest.reader.command.CommandParseDefineTransition
16
17 COMMAND_RUN_TEST_CLASS_Comment=br.pucrio.inf.jautotest.test.command.ExecuteComment
18 COMMAND_RUN_TEST_CLASS_TestCase=br.pucrio.inf.jautotest.test.command.ExecuteTestCase
19 COMMAND_RUN_TEST_CLASS_Procedure=br.pucrio.inf.jautotest.test.command.ExecuteProcedureDefinition
20 COMMAND_RUN_TEST_CLASS_DefineTransition=br.pucrio.inf.jautotest.test.command.ExecuteDefineTransition
21 COMMAND_RUN_TEST_CLASS_ForceState=br.pucrio.inf.jautotest.test.command.ExecuteForceState
22 COMMAND_RUN_TEST_CLASS_GenerateEvent=br.pucrio.inf.jautotest.test.command.ExecuteGenerateEvent
23
24 COMMAND_RUN_TEST_COMMAND_CLASS_TestCommand=br.pucrio.inf.jautotest.test.command.ExecuteTestCommand
25 COMMAND_RUN_TEST_COMMAND_CLASS_TestCommandRecovery=
   br.pucrio.inf.jautotest.test.command.ExecuteTestCommandRecovery
26 COMMAND_RUN_TEST_COMMAND_CLASS_ProcedureCall=
   br.pucrio.inf.jautotest.test.command.ExecuteProcedureCall
```

Figura 18: Arquivo de configuração para a instância com linguagem de máquinas de estado

Para cada um dos novos comandos foi criado um novo elemento de *script*. Os elementos criados foram:

- DefineTransaction, representando o comando “Existe transição”.
- ForceState, representando o comando “Vai para”.
- GenerateEvent, Representando o comando “Acontece evento”.

Como a linguagem utilizada é puramente textual, foi possível criar o novo módulo apenas através da configuração do módulo de linguagem ***DefaultScriptReader***, descrito na seção 3.4, como é possível ver na linha 1 da figura. Na linha 5 estão declarados os 3 novos comandos criados pela linguagem. Os componentes que realizam a leitura são registrados para seus respectivos comandos nas linhas 13 a 15.

A execução dos novos elementos de script é realizada por novos componentes de execução que são registrados nas linhas 20 a 22. Esses componentes recebem seus respectivos elementos de script, os executam, e adicionam os resultados execução ao relatório de resultados, que será processado pelo módulo de resultados. Novos elementos de relatório foram criados para conter os resultados de cada novo comando. Caso seja interessante a construção

de um módulo de linguagem gráfico, os componentes de execução e dos modelos de especificação e resultados não precisam ser alterados, bastando para tal alterar apenas o módulo de linguagem, utilizando os elementos disponíveis na criação do modelo de especificação.

A figura 19 exibe um diagrama explicando a instanciação do framework.

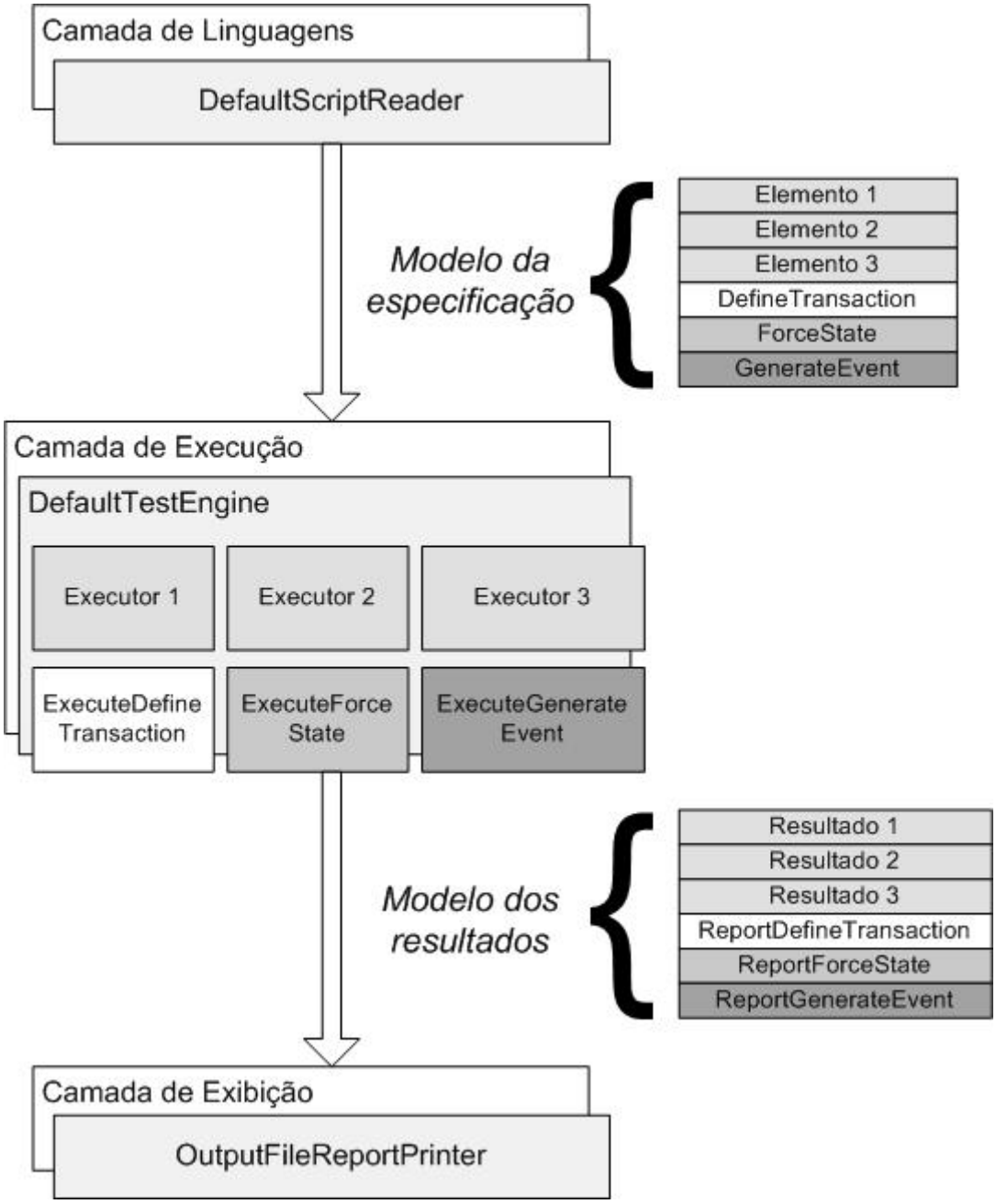

Figura 19: Diagrama da ferramenta após a instanciação com a linguagem de máquinas de estado.

Da mesma forma que na linguagem de exemplo, os scripts são utilizados como uma linguagem de mais alto nível que realiza os testes invocando métodos

do módulo específico de testes, de acordo com a lógica do *script*. Para esse novo conjunto de comandos, são necessários quatro tipos de métodos:

- **void force<Nome do estado>()**: Deve existir um método desse tipo para cada estado criado. É o método invocado pela ferramenta no momento em que o comando **Vai para <nome do estado>** é executado.
- **boolean check<Nome do Estado>()**: Também deve existir um desses para cada um dos estados criados. É o método invocado após a ocorrência de uma transição, para verificar se o estado alcançado foi de fato o esperado.
- **void generate<Nome do evento>()**: Deve haver um método desse tipo para cada evento disponível no *script*. Este método é invocado durante a execução do comando **acontece evento <Nome do evento>** para gerar o evento passado como parâmetro.
- **String getCurrentState()**: Obtém o estado corrente da máquina. É implementado apenas uma vez por módulo específico de testes e é utilizado para obter o estado da máquina em caso de falha, ou seja, quando o estado esperado não é alcançado após uma transição.

Os resultados são exibidos pelo mesmo módulo utilizado na seção 3.4, o ***OutputFileReportPrinter***, que imprime os resultados em um arquivo texto.