5
Tecnologias Utilizadas

5.1.
Inversão de Controle e Injeção de Dependências

A inversão de controle é uma técnica utilizada para inverter a forma como um programa realiza algum tipo de operação em comparação a arquiteturas mais tradicionais [21]. Esta técnica permite, por exemplo, que um código genérico seja escrito sem que se conheçam a priori todas as possibilidades de comportamento da aplicação, deixando que lacunas sejam preenchidas em tempo de execução. A injeção de dependências é uma forma específica de inversão de controle onde a inversão ocorre na forma como as dependências necessárias são carregadas [21]. Ao utilizar esta técnica, existe uma interface que todas as dependências de um tipo específico devem implementar, e a dependência concreta que será de fato utilizada é passada para a aplicação que a utilizará, ao contrário do fluxo usual onde o código que utiliza a dependência já conhece a implementação. Neste trabalho essa prática é necessária para que diferentes módulos de linguagem de especificação e de apresentação de resultados possam ser criados de forma separada do código da ferramenta, e posteriormente utilizados sem a necessidade de se re-compilar o framework. Essas configurações podem ser vistas no exemplo da figura 20. Os nomes das classes que implementam os módulos são configurados em um arquivo textual. Em tempo de execução o sistema carregará as classes passadas nesses parâmetros, e utilizará estas implementações específicas para realizar as computações necessárias.

```
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
```

Figura 20: Exemplo de configuração dos módulos utilizando injeção de dependências

5.2. Reflexão Computacional

É o termo utilizado para denominar o comportamento exibido por sistemas reflexivos [22]. Sistemas reflexivos são sistemas de computação relacionados a si mesmos de forma causalmente conectada [22]. Um sistema computacional causalmente conectado ao seu domínio é aquele que possui uma representação do que ocorre em seu domínio em forma de um modelo, de maneira que alterações no domínio alterem este modelo, e da mesma forma, alterações no modelo devem provocar mudanças no domínio [22]. Portanto, sistemas reflexivos são capazes de manter modelos de sua estrutura interna, sua auto-representação [22], fornecendo também mecanismos para que alterações em sua auto-representação reflitam em modificações na estrutura e comportamento do sistema.

Esta possibilidade traz a um *software* a capacidade de auto-configuração, permitindo que técnicas como a injeção de dependências, citada anteriormente, sejam aplicadas através de arquivos que descrevem as dependências que serão utilizadas. Desta forma, torna-se mais simples configurar a ferramenta para utilizar os novos componentes criados.

Novamente utilizando o exemplo da figura 20, no qual os módulos responsáveis por entradas, saídas e execução dos testes são definidos no arquivo de configuração, o *framework* recebe apenas o nome da classe que deve ser carregada, e envia um comando através da *api* de reflexão de Java [23] solicitando um objeto que representa a classe correspondente ao nome passado. A classe obtida é então instanciada através de nova chamada à *api* de reflexão, e o novo objeto, o qual não se conhecia em tempo de compilação, é retornado para ser utilizado normalmente no restante do código, através de sua interface. Dessa forma, o comportamento original da aplicação pode ser alterado, uma vez que o novo objeto que será utilizado pertence a uma classe que não necessariamente se conhecia no momento do desenvolvimento do *framework*. Na figura 21 é apresentado um exemplo simples de carga dinâmica de uma classe através da api de reflexão de Java [23]. No exemplo é carregada uma implementação de uma interface Lista que não era necessariamente conhecida em tempo de compilação.

```
1  Class classeConcretaDaLista = Class.forName( "br.pucrio.inf.ListaDuplamenteEncadeada" );
2  Lista lista = ( Lista ) classeConcretaDaLista.newInstance();
```

Figura 21: Exemplo de instanciação de objeto utilizando reflexão computacional em Java