Tutorial de execução da ferramenta Code Maat

Code Maat é uma ferramenta escrita em Clojure, que possibilita a criação de aplicações concorrentes que rodam sobre a JVM. Por isso, para automatizar a compilação do código, utilizaremos o leiningen, que por sua vez consegue automatizar a compilação e gerenciamento de dependências para projetos escritos em Clojure.

A instalação do leningen para ambiente Windows pode ser feita intuitivamente através do seguinte link: leiningen-win-installer.

Tendo devidamente instalado o leningen, o próximo passo é obter o código do Code Maat.

```
git clone https://github.com/adamtornhill/code-maat.git
```

Em seguida, devemos construir a aplicação. Para isso utilizaremos, como já citado anteriormente, utilizaremos o leiningen. Na pasta do Code Maat executamos:

```
lein uberjar
```

Esse comando criará um novo diretório dentro da pasta do projeto, intitulado *target*. Ali estarão os .jar construídos, referentes ao Code Maat.

Para fazer a verificação das dependências lógicas de um determinado projeto hospedado no git, primeiro é necessário fazer o clone do mesmo. Uma vez que o Code Maat aceita como entrada o arquivo de log de um projeto, devemos extraí-lo daquele projeto que queremos analisar. Dois determinados tipos de formatação de arquivos de log são aceitos pela ferramenta; esses tipos podem ser obtidos através dos seguintes comandos, quando executados dentro da pasta local do repositório que queremos analisar:

```
git log --pretty=format:'[%h] %aN %ad %s' --date=short --numstat --after=YYYY-MM-DD > logfile.log
```

```
git log --all --numstat --date=short --pretty=format:'--%h--%ad--%aN' --no-renames --after=YYYY-MM-DD > logfile.log
```

OBS.: Segundo a documentação do code-maat, o segundo formato apresentado é mais tolerante e rápido de analisar e por isso é preferível em relação ao primeiro.

Tendo extraído o arquivo de log, podemos, enfim, analisar as dependências lógicas com ajuda do Code Maar. Existem quatro tags relevantes que devemos entender para enfim executar a ferramenta de análise de dados de um sistema de controle de versões:

- -l: entrada (arquivo log) a ser analisado.
- -c: controle de versão. Os formatos suportados são: svn, git, git2, hg, p4, e tfs.

  O tipo git analisa arquivos de log gerados pelo primeiro formato de arquivo; o segundo formato (aquele que é mais tolerante e rápido) é aceito pelo tipo git2.
- -a: tipo de análise a ser executada.

  Há várias opções (que são listadas e melhor explicadas na documentação), mas a que nos interessa aqui é a *coupling*, que estuda o acoplamento lógico de um sistema.
- -o: arquivo de saída; escreve o resultado da análise em um determinado arquivo.

Entendendo essas principais tags, podemos agora executar o Code Maat para estudar as dependências lógicas de um projeto. Para isso, no diretório gerado pela execução do leiningen sobre o código do Code Maat, rodamos:

```
java -jar code-maat-1.1-SNAPSHOT-standalone.jar -l logfile.txt -c git2 -a coupling -o result.cvs
```

Deve ser obtido, através disso, o resultado da análise do acoplamento lógico em um arquivo CVS.

OBS[1].: O Leiningen pode gerar um .jar que contenha ou não “SNAPSHOT” em seu nome, dependendo da versão do code-maat a ser construída. O .jar a ser executado no comando acima é aquele que contém “standalone” em seu nome.

OBS[2].: No comando acima, estamos considerando que o arquivo de log a ser analisado está na mesma pasta que os .jar gerados pelo Leiningen. É possível passar o caminho para o arquivo também, se isso não acontecer.

OBS[3].: Notar a definição do controle de versão como git2. Isso implica que o arquivo log logfile.txt foi extraído através do segundo comando apresentado.