# FERRAMENTA DE *SOFTWARE* PARA O AUXÍLIO AO SUPORTE E APRENDIZADO DA PLATAFORMA MIKROTIK

Marlon Cordeiro Domenech*
Herculano Haymussi de Biasi **

Resumo

Este artigo descreve a arquitetura geral e as características principais de uma ferramenta de software destinada a auxiliar o processo de aprendizagem e fornecer suporte a equipamentos da plataforma de redes Mikrotik. A ferramenta é destinada, principalmente, a usuários de nível inicial da plataforma, mas pode também ser usada por usuários mais experientes e vem para resolver a problemática do difícil suporte e aprendizado da plataforma, haja vista suas inúmeras possibilidades de configuração. Para tal, foram utilizadas técnicas de compilação e interpretação de linguagens de programação, engenharia de software, modelagem orientada a objetos por meio de Unified Modeling Language (UML) e conceitos de redes de computadores mais específicos da plataforma Mikrotik. Como resultado, foi obtido um software que faz análise sintática do arquivo de configuração de uma RouterBoard Mikrotik 493 com Router OS versão 5.11, mostrando ao usuário, em linguagem natural, qual a funcionalidade de cada uma das propriedades presentes no arquivo citado. Atualmente, o software cobre em torno de 70% das configurações possíveis da plataforma citada. Também foi obtida uma arquitetura de software que consegue "espelhar", por intermédio de objetos de alto nível, a estrutura de uma RouterBoard Mikrotik, o que permitirá aplicação de análise semântica mais aprofundada, não somente baseada nos valores das propriedades, mas também nas ligações de uma propriedade com as demais, tirando conclusões a partir disso. Deve ser salientado que não é do conhecimento do autor deste trabalho nenhuma ferramenta similar, sendo, portanto, os resultados apresentados neste artigo completamente inovadores.
Palavras-chave: Compilador. Interpretador. Mikrotik. Engenharia de software. Análise semântica. Linguagem natural.

## 1 INTRODUÇÃO

O presente artigo apresenta resultados do Trabalho de Conclusão de Curso (TCC) em Ciência da Computação. O objetivo do trabalho foi desenvolver um *software* com as funções de analisador léxico, analisador sintático e semântico e interpretador do arquivo de configuração de uma *RouterBoard* Mikrotik 493, com sistema operacional Router OS versão 5.11, de forma a auxiliar no processo de aprendizagem e oferecer suporte a equipamentos dessa plataforma. Esse auxílio torna-se relevante quando, por exemplo, uma empresa que necessita de suporte técnico ao equipamento está sem acesso a algum serviço (como, por exemplo, à Internet, à Intranet ou a qualquer outro serviço de rede) em razão de falha em uma *RouterBoard*. Se o suporte puder ser facilitado, sendo, como consequência, finalizado antes e com maior correção, o prejuízo à empresa será menor. Da mesma forma, um cenário no qual tal ferramenta seria relevante é em provedores de acesso à Internet que usam a plataforma Mikrotik. Se o suporte a um equipamento for feito em menor tempo, com maior correção, os danos financeiros e à imagem da empresa serão menores, haja vista que o período de instabilidade ou inoperância da rede que atende seus clientes será menor.

É importante salientar que, em se tratando da plataforma citada, as possibilidades de configuração são inúmeras, uma vez que o equipamento não desempenha apenas uma função. Dependendo da configuração feita, o mesmo equipamento pode desempenhar as funções de *switch*, roteador, *bridge*, *firewall*, *web-proxy* ou, ainda, várias delas ao mesmo tempo. Assim, o número de configurações tende a ser bem maior do que em equipamentos que desempenham somente uma função.

* Graduado em Ciência da Computação pela Universidade do Oeste de Santa Catarina de Videira; e-mail: marloncdomenech@gmail.com
** Mestre em Ciência da Computação pela Universidade Federal de Santa Catarina; Professor titular da Universidade do Oeste de Santa Catarina; e-mail: herculano.debiasi@gmail.com

Dessa maneira, foi procurado alcançar um nível de maturidade da ferramenta que permita mostrar ao usuário, em texto plano e linguagem natural, o que determinada propriedade presente no arquivo de configuração de uma *RouterBoard* faz exatamente. Também se buscou preparar a ferramenta para receber a implementação de análise semântica mais aprofundada, que viria a permitir correlações entre as diversas configurações feitas e problemas típicos de redes (por exemplo, uma configuração no *firewall* conflitando com uma configuração de *web-proxy*).

Não deixa de ser relevante considerar que tal ferramenta, ou similar, não foi encontrada no mercado pelo autor, sendo, portanto, os resultados deste trabalho de caráter inovador.

## 2 **METODOLOGIA**

O trabalho foi iniciado com a revisão bibliográfica referente a redes de computadores, a fim de se obter o conhecimento técnico necessário para poder explicar, em linguagem natural, a função de uma configuração para o usuário final. Foi realizado um estudo acerca de Sistemas Operacionais, com o objetivo de elucidar os principais conceitos que podem permear a utilização do RouterOS, sistema operacional das *RouterBoards* Mikrotik. Também foi feito um estudo sobre linguagens formais e autômatas, a fim de proporcionar embasamento à análise bibliográfica de compiladores. Ambos servem para o correto desenvolvimento da ferramenta. Foi revisado o tema *UML* e linguagem de programação JAVA, a fim de poder projetar, implementar e documentar a ferramenta citada. O próximo passo foi o estudo da plataforma Mikrotik, suas funcionalidades e possíveis implementações. Foi, então, abordado o estudo do arquivo de configuração de uma *RouterBoard* Mikrotik, com o intuito de entender a linguagem usada e o encadeamento lógico, léxico, sintático e semântico dos componentes desse arquivo.

A seguir, foi feito o levantamento de requisitos da plataforma e sua modelagem, utilizando padrões *UML*. Após, foi iniciado o desenvolvimento do analisador léxico, sintático e semântico e do interpretador do arquivo de configuração, sendo baseado nos conhecimentos adquiridos e em conhecimentos específicos da plataforma, obtidos por meio de documentação do fabricante, mendiante testes de campo e discussão em fóruns sobre o assunto.

## 3 **O ESTUDO DA LINGUAGEM**

Segundo Ferreira, Anjos e Ferreira (2004, p. 1213), uma linguagem é "O uso da palavra articulada ou escrita como meio de expressão e de comunicação entre pessoas" e, também, é "A forma de expressão pela linguagem própria de um indivíduo, grupo, classe, etc." Pode-se concluir, portanto, que uma linguagem é a forma de o indivíduo expressar ideias e transmiti-las a outros, como, por exemplo, a uma máquina ou a um indivíduo semelhante.

Porém, para que a comunicação entre indivíduos ou entre indivíduo e máquina ocorra, é necessário que o que for transmitido/recebido possa ser entendido/interpretado pelas partes que interagem na comunicação. Para que isso aconteça e a linguagem venha a ter uma estrutura bem definida, são necessárias a definição de palavras válidas dessa linguagem e suas regras gramaticais. Segundo Campani (2006, p. 10), esses dois elementos são os constituintes de uma linguagem.

Contudo, normalmente, as linguagens usadas para configuração/operação de máquinas são diferentes das linguagens usadas pelo ser humano. Seria o grupo de linguagens formais. Segundo Campani (2006, p. 10), esse é um grupo distinto de linguagens, as quais podem ser representadas por meio de um sistema com sustentação matemática, basicamente, formada por sintaxe e semântica.

Segundo Menezes (2001, p. 2), "A sintaxe trata das propriedades livres da linguagem como, por exemplo, a verificação gramatical de programas." Já a semântica, segundo o mesmo autor, "[...] objetiva dar uma interpretação para a linguagem como, por exemplo, um significado ou valor para um determinado programa."

Essa sustentação matemática, que torna a especificação de uma linguagem sem ambiguidades e precisa, é de suma importância para a comunicação homem-máquina, pois um computador apenas executa comandos em uma linguagem conhecida. Se esses comandos forem ambíguos, os programas escritos terão comportamentos incertos, ora apresentando um comportamento, ora apresentando outro.

Dessa linguagem, decorrem diversas estruturas. As principais são os símbolos, estruturas abstratas básicas, caracteres; o alfabeto, o qual é um conjunto finito de símbolos; a palavra, como uma sequência finita de símbolos de um alfabeto; e a gramática, a qual serve para especificar uma linguagem que poderia ser infinita, de forma finita,

formalmente. É essa última que diz que uma palavra faz parte de uma linguagem ou não (MENEZES, 2001, p. 21; CAMPANI, 2006, p. 15).

Segundo Menezes (2001, p. 23), uma gramática é uma construção matemática sem ambiguidades, definida por uma quádrupla (V, T, P, S): V é um conjunto de símbolos variáveis; T é um conjunto de símbolos terminais, tal que $V \cap T = \emptyset V \cap T = \emptyset$; P é o conjunto de regras de produção, as quais definem as regras para a geração de palavras de uma linguagem (se uma construção de palavras puder ser gerada a partir dessas regras, ela faz parte da linguagem); e S é um elemento de V, denominado variável inicial.

Por se tratar de uma plataforma fechada, a Mikrotik não disponibiliza publicamente a linguagem e a gramática usadas na construção do arquivo de configuração de uma *RouterBoard*. Portanto, foi necessário o estudo desse arquivo e diversos testes com as mais diversas configurações, a fim de captar as minúcias da gramática usada. Como resultado, até o momento foi obtido um arquivo da ordem de 41 páginas A4 definindo a gramática utilizada no arquivo de configuração. O arquivo contém regras de produção como as apresentadas na Figura 1. Foi utilizada, em toda a gramática, a Forma Normal de Backus, ou BNF.

Figura 1 – Parte Inicial da gramática

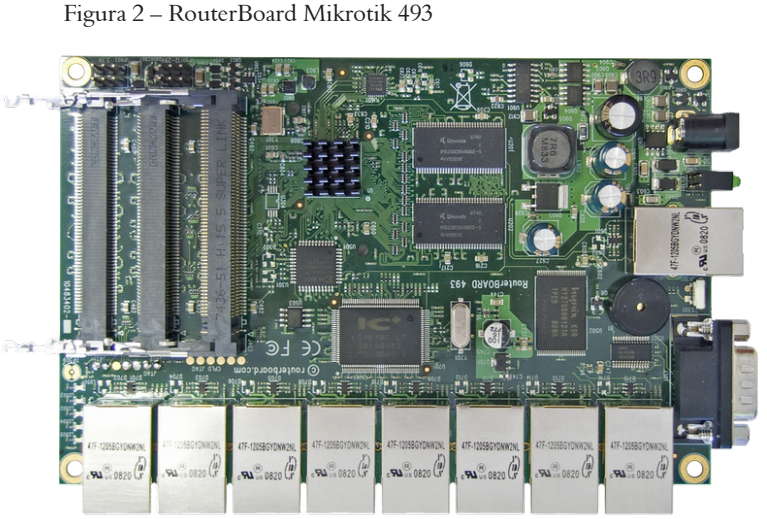
```
<ConfigFile>::= "# " <data> <tempo_horario>  " by Router OS "
<versaoROS> "# software ID = " <softID>  "# "  <ROS>

<ROS>::= <interface_> <ip>

<interface_>::= "/interface " <interfaceopts>

<interfaceopts>::= <bridge> | <ethernet>|  <wireless>

<ip>::= "/ip" <ip_opts>

<ip_opts>::= <ip_accounting> | <ip_address> | <ip_dhcp_server> |
<ip_dhcp_relay> | <ip_dhcp_client> | <ip_arp> | <ip_dns> |
<ip_service> | <ip_socks> | <ip_ssh> | <ip_traffic_flow> |
<ip_neighbor> | <ip_packing> | <ip_pool> | <ip_tftp> | <ip_proxy> |
<ip_route>

•
•
•
```

Fonte: os autores.

A partir dessa gramática, a qual foi sendo construída incrementalmente, conforme ocorriam os estudos acerca do arquivo de configurações e parte do desenvolvimento, foi possível realmente moldar o *software* para se adequar a essa gramática e conseguir realizar o trabalho proposto.

Para que o estudo da linguagem pudesse ser feito, utilizou-se uma *RouterBoard* Mikrotik 493, com sistema operacional RouterOS 5.11, mostrada na Figura 2. Nela foram realizados diversos testes de possibilidades de configuração, a fim de obter quais seriam as propriedades passíveis de configuração, quais os valores possíveis e qual o comportamento em cada valor. Para tal propósito, também foi utilizada a documentação do fabricante.

Figura 2 – RouterBoard Mikrotik 493



Fonte: Routerboard (2012).

Como exemplo, pode-se citar o arquivo de configuração (gerado a partir do comando */ ip route export file=route1* digitado na *console* do equipamento), que mostra as políticas de roteamento e as rotas adicionadas manualmente. Tal arquivo é mostrado na Figura 3.

É possível perceber no arquivo que, após algumas informações sobre a *RouterBoard* (linhas 1 a 3), começa a configuração. Percebeu-se que o arquivo se divide em estruturas hierárquicas. Como exemplo, a estrutura *ip* contida na raiz, contém a estrutura *route*, e esta contém a estrutura *rule*. Dentro de *route*, estão as entradas manuais na tabela de roteamento. É possível notar que há duas entradas, cada uma começando com a palavra *add*. Após essa palavra, estão as propriedades do item, algumas opcionais, outras não (há dois itens, com algumas configurações presentes em um e ausentes em outro). Cada valor de cada propriedade fará com que o equipamento apresente um comportamento diferente. Por exemplo, no segundo item de */ip route,* caso a propriedade *gateway* seja alterada, o pacote que passa pelo equipamento será enviado a um destino diferente.

Figura 3 – Arquivo de configuração

```
# jan/01/2002 01:00:06 by RouterOS 5.11
# software id = 12MC-C1BV
#
/ip route
add bgp-as-path=513,8220,7018,701703,80 bgp-atomic-aggregate=yes \
    bgp-communities=0:0,no-advertise bgp-local-pref=20 bgp-med=0 bgp-
origin=\
    incomplete bgp-prepend=0 check-gateway=ping comment=comerntario
disabled=\
    no distance=1 dst-address=192.168.8.0/24 gateway=\
    "ether4,192.168.6.1,(unknown)" pref-src=192.168.7.2 routing-mark=\
    marcaDeRoteamento scope=30 target-scope=10
add disabled=no distance=1 dst-address=0.0.0.0/0 gateway=192.168.7.1
scope=30 \
    target-scope=10
/ip route rule
add action=lookup comment=coment disabled=no dst-
address=192.168.8.0/24 \
    interface=ether7 routing-mark=haha src-address=192.168.7.0/24
table=\
    marcaDeRoteamento
add action=lookup disabled=no table=main
```

Fonte: os autores.

É importante que cada uma das propriedades configuradas no equipamento, explícitas no arquivo de configuração, possa ser interpretada corretamente, mostrando ao usuário, de forma clara, a semântica envolvida nessa pro-

priedade. Também é imprescindível que durante a interpretação a estrutura sintática do arquivo, explícita na gramática (exemplo na Figura 1), possa ser validada, a fim de se assegurar de que ele não foi alterado e que os valores das propriedades são válidos. Para tais propósitos, foi construída uma planilha que denota a hierarquia de comandos (sua estrutura sintática) e a semântica de cada um deles. Essa planilha tem um exemplo no Quadro 1.

Quadro 1 – Planilha de hierarquia e semântica

| Nível | | Comando Inicial | Propriedades | Neg | Opt | Observação | Semântica | Valores Possíveis |
|---|---|---|---|---|---|---|---|---|
| Interface | | | | | | | | |
| | Bridge | | | | | | | |
| | | add | - | | sim | só para o caso de haver alguma interface bridge criada. O MAC associado a bridge é o MAC da menor interface associada a essa | | |
| | | | admin-mac | não | não | Tem efeito se a propriedade auto-mac=no | MAC Address Estático (manual) para a interface. | MAC-Address |
| | | | ageing-time | não | não | | Por quanto tempo a informação de um host será mantida na base de dados de | tempo - **Default: 00:05:00** |
| | | | arp | não | não | | Define o funcionamento do protocolo | disabled, **enabled**, proxy-arp, reply-only |
| | | | auto-mac | não | não | | Se for yes, seleciona automaticamente o menor MAC de alguma interface como | **yes**, no |
| | | | comment | não | sim | | - | texto |
| | | | disabled | não | não | | - | yes,no |

Fonte: os autores.

No exemplo do Quadro 1, o nível mostrado refere-se àquele que mostra a configuração de interfaces lógicas *bridge* (/interface bridge). Cada item é identificado por meio de uma palavra 'add' inserida no arquivo, o que na planilha é mostrado na coluna Comando Inicial. Cada propriedade tem uma linha, e seu nome está na coluna Propriedades. A coluna Neg indica se a propriedade pode ser negada (por meio do símbolo '!' em frente ao valor); a coluna Opt define se a propriedade sempre deve aparecer no arquivo ou não; o campo Observação fornece alguma explicação sobre exceções à regra, funcionamento de um item da hierarquia ou se há dependência de uma propriedade com outra (por exemplo, se uma propriedade somente tem sentido se outra tiver determinado valor); a coluna Semântica define, em linguagem natural, qual o significado da propriedade e o comportamento esperado em cada valor; e a coluna Valores Possíveis determina quais os valores esperados para a propriedade, estando de acordo com a gramática para esse item.

## 4 CORRELAÇÃO COM COMPILADORES

Dessa forma, tendo identificado parte da linguagem, foi possível criar o *software* que conseguisse interpretar essa linguagem. É importante salientar que o *software* usa conceitos de compiladores e interpretadores, mas não é um compilador ou interpretador típico. Por compilador típico, pode-se usar a definição de Aho, Sethi e Ullman (1995, p. 1), que diz que "Um compilador é um programa que lê um programa escrito numa linguagem, a linguagem *fonte*, e o traduz num programa equivalente numa outra linguagem, a linguagem *alvo*." Portanto, é mais orientado a passar uma linguagem de alto nível para linguagem de baixo nível. Já um interpretador, segundo Aho, Sethi e Ullman (1995, p. 2), em vez de produzir um programa em uma linguagem de mais baixo nível, a partir de um programa em linguagem de mais alto nível, realiza as operações especificadas pelo programa fonte. Ainda, segundo Aho, Sethi e Ullman (1995, p. 1), são duas as partes da compilação: análise e síntese. A análise divide o programa-fonte em partes e cria uma representação intermediária dele, enquanto a síntese constrói o programa-alvo a partir dessa representação intermediária. O *software* aqui descrito utiliza apenas a parte de análise, ou seja, análise léxica, sintática e semântica, não executando nenhum comando do código passado (arquivo de configuração da *RouterBoard*), apenas traduzindo-o em expressões da linguagem natural.

Dessa maneira, optou-se pela utilização de uma análise sintática *top-down*, chamada Recursiva Preditiva. Segundo Aho, Sethi e Ullman (1995, p. 20), esse é um método no qual cada não terminal da gramática é representado por um procedimento no compilador. Nele, o primeiro *token* do lado direito de cada produção é único (considerando o mesmo lado esquerdo), determinando de forma única o procedimento a ser selecionado ao se ler um *token* da entrada. Assim, evita-se o retrocesso e o processo fica mais rápido. Além disso, as regras de recursividade desse modelo ficam implícitas nas chamadas feitas em *software*. Todos esses detalhes contribuíram para a escolha desse método, que tornou o desenvolvimento da ferramenta mais facilitado.

Quanto aos demais passos (análise léxica e semântica), estão implícitos no decorrer da análise sintática. Esse método chama-se Tradução Dirigida por Sintaxe que, segundo Price e Toscani (2001, p. 85), "[...] É uma técnica que permite realizar tradução (geração de código) concomitantemente com a análise sintática." Também, para aplicar todos os passos junto à análise sintática, foram utilizados esquemas de tradução. Conforme Aho, Sethi e Ullman (1995, p. 17), "Um esquema de tradução é uma gramática livre de contexto, na qual fragmentos de programas chamados de ações semânticas são inseridos nos lados direitos das produções." Nesse método, a avaliação das regras semânticas e das variáveis da gramática é feita de forma explícita, facilitando o entendimento. Contudo, em razão da extensão do código-fonte e das regras semânticas que deveriam ser inseridas na gramática, tornando-a quase ilegível e excessivamente extensa, optou-se por trabalhar com uma planilha que descreve a semântica envolvida em cada propriedade do arquivo de configuração. Tal planilha está exemplificada no Quadro 1.

## 5 A MODELAGEM DA FERRAMENTA

Como forma de projetar e documentar o *software*, foi utilizada a modelagem orientada a objetos utilizando *UML*. Segundo Melo (2004, p. 33-34), "UML [...] é uma linguagem para especificação, visualização, construção e documentação de artefatos de sistemas de *software*." Portanto, ela serviu para que se fizesse a especificação das responsabilidades de cada um de seus componentes, a fim de atender às necessidades do usuário. Além disso, serviu como forma de documentar os relacionamentos entre as diversas partes desse *software*. Foram feitos quatro tipos de diagramas: de pacotes, de casos de uso, de classes e de sequência; todos disponibilizados para acesso Web, em decorrência de seu tamanho. O Diagrama de Casos de Uso está disponível em: <http://bit.ly/KXweCk>, o Diagrama de Sequência em: <http://bit.ly/KNzket>, o Diagrama de Classes, em duas partes, disponível em: <http://bit.ly/KNzhiQ> e <http://bit.ly/LB2URK> e o Diagrama de Pacotes está em: <http://bit.ly/KNzyCn>.

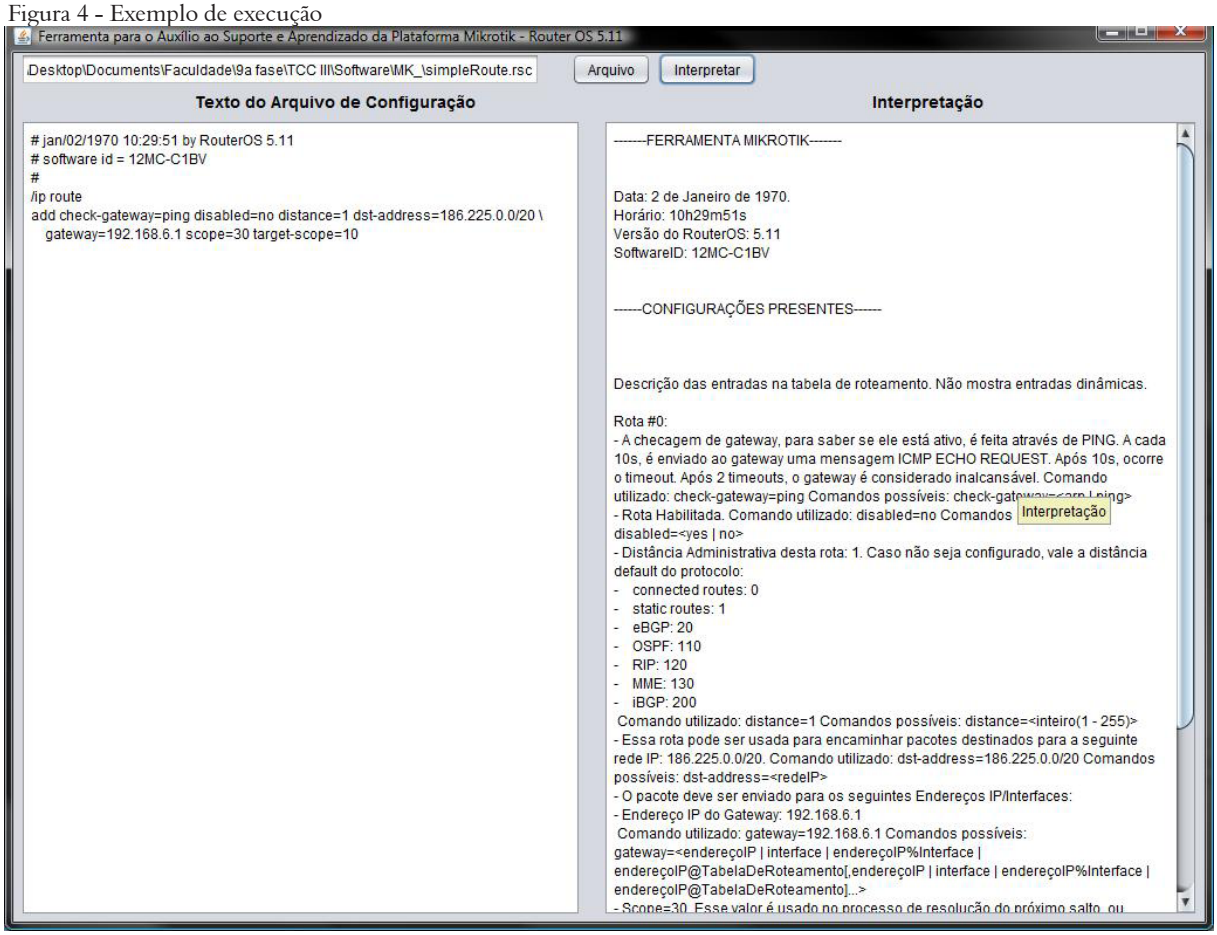
A ferramenta foi idealizada para ser usada em casos em que o usuário busca o aprendizado sobre algum tópico, ou ainda ajuda na resolução de algum problema, por meio do uso de um arquivo de configuração da *RouterBoard*. Dessa maneira, o levantamento de requisitos apontou apenas dois casos de uso. Será fornecido pelo usuário um arquivo de configuração, e o *software* irá interpretá-lo, retornando o texto em linguagem natural.

A atividade de interpretação é denotada pelo diagrama de sequência dessa atividade, exemplificado para o caso de interpretação de parte do arquivo da Figura 3. Nele, é possível notar que é feito o *parsing* (por meio dos objetos com nome iniciando com *Parser*), obedecendo à hierarquia de comandos. Por exemplo, há o objeto Parser_Geral, que inicia o processo de interpretação, criando o objeto *RouterBoard* (que espelha as configurações da *RouterBoard* Mikrotik pra futuras implementações de análise semântica mais profunda) e o *ParserRouterboard* (que interpreta as informações iniciais sobre a *RouterBoard*). Após, é feita a interpretação de */ip* pelo objeto Parser_IP, seguida do mesmo processo para *route*, feito pelo Parser_IP_Route. Então, é feita a interpretação de um item dessa hierarquia pelo objeto Parser_IP_Route_Item, o qual faz o *matching* da palavra *add* e irá interpretar todas as propriedades de um item de */ip route*. São usados métodos comuns do objeto *Common*, para evitar a reescrita de código em diversas classes de interpretação. Cada propriedade lida é armazenada em um objeto *Bean*, que espelha um objeto da plataforma Mikrotik. Esse objeto é armazenado no objeto *Routerboard*. Após as operações de interpretação de propriedades, o *Bean* é validado por meio do método *isComplete():Boolean*, o qual define se o *Bean* está com todas as propriedades necessárias para ser válido. Essa validação tem profunda importância para a análise semântica atual e futura, pois se faltar alguma propriedade indispensável para um item, a semântica ficará inconsistente.

## 6 INTERFACE COM USUÁRIO E EXEMPLO DE EXECUÇÃO

A interface com o usuário foi feita de forma simples e intuitiva. Basta selecionar um arquivo de configuração completo (gerado a partir do comando *export file=nomeDoArquivo*) ou parcial (mesmo comando, entretanto, dentro da hierarquia desejada) e clicar em "Interpretar". Um exemplo de execução é mostrado na Figura 4. É possível notar que para cada comando interpretado é mostrado qual foi o comando utilizado, os comandos possíveis e, se houver, qual é o valor padrão da propriedade.

Figura 4 - Exemplo de execução




Fonte: os autores.

Como citado, o *software* está cobrindo em torno de 70% das configurações possíveis. Também é importante ressaltar que a cobertura do *software* é para a versão do RouterOS 5.11. Para novas versões, é necessário acompanhar a documentação de atualizações do fabricante. Outro ponto que está fora do escopo é a interpretação de *scripts*, desenvolvidos por meio da API da Mikrotik. Também fica fora do escopo a aplicação em outros modelos, haja vista que o *chipset* de *switching* de cada *RouterBoard* tem atribuições diferentes, conforme o *hardware* em que está operando. Não é coberta a interpretação de comandos na console do equipamento.

## 7 **CONCLUSÃO**

A partir dos resultados alcançados é possível afirmar que, em se tratando de um trabalho que envolve a concepção da arquitetura de um *software* que venha a ser robusto o suficiente para suportar a expansão para uma ferramenta de análise semântica aprofundada, o estudo da linguagem de um arquivo de configuração de uma *RouterBoard*, a montagem de uma gramática condizente e de um *software* que consiga interpretar, com capacidade de análise semântica, um arquivo de configuração completo ou parcial (permitindo *troubleshooting* em parte da configuração), os objetivos foram alcançados.

Ainda, a partir desses resultados, é possível vislumbrar diversos trabalhos futuros, como a implementação de uma análise semântica mais profunda, a fim de apresentar resultados com um nível mais alto de "inteligência". Ainda, pode ser estendida a ferramenta para a API e para comandos digitados na *console* do equipamento, bem como para a atividade de simulação de configurações e interação com outros componentes de rede, a fim de testar os resultados antes de colocar o equipamento real em campo. Tais trabalhos são perfeitamente suportados pela arquitetura desenvolvida até agora, portanto, é possível concluir, a partir dos resultados obtidos, que é possível desenvolver um *software* completo para o auxílio ao processo de aprendizagem e suporte a equipamentos da plataforma Mikrotik, a partir da arquitetura proposta e do que foi desenvolvido até agora.

Ainda, deve-se citar que tal ferramenta pode vir a facilitar muito o trabalho e o aprendizado com esses equipamentos, haja vista que não há nada igual a essa ferramenta no mercado. No estado atual, já poderia ser fornecida uma versão *on-line* para testes com usuários, o que seria de grande valia para a validação dos resultados.

***Software tool to aid the technical support and learning of the mikrotik plataform***

*Abstract*

*This article describes the general architecture and the meaningful characteristics of a software tool intended to aid the learning and technical support process of Mikrotik Network Plataform's equipments. It's targeted primarily to users at beginner level of experience in the plataform, but it can be used by more experienced users. The tool comes to solve the problem of the difficult of supporting and learning about the plataform, because of its numerous configuration possibilities. For this purpose, it has been used programming language's compilation and interpretation techniques, software engineering, object oriented modeling through UML and computer network's concepts, mainly those more specific of the Mikrotik Plataform. As a result, it has been gotten a software that parses the configuration file of a RouterBoard Mikrotik 493, running Router OS version 5.11, showing to the user, in natural language, what is the functionality of each one of the present properties in the file. Currently, the software covers about 70% of the possible configuration in the plataform. It has been also gotten a software architecture that can 'mirror', through high level objects, the structure of a RouterBoard Mikrotik, what is going to allow the application of a deeper semantic analysis, not just based on the properties value, but also in the connections of a property with the others, drawing conclusions from this. It should be noted that it's not known to the author some kind of similar tool, therefore, the results shown in this article are completely innovative.*

*Keywords: Compiler. Interpreter. Mikrotik. Software engineering. Semantic analysis. Natural language.*

## REFERÊNCIAS

AHO, A. V.; SETHI, R.; ULLMAN, J. D. **Compiladores**: princípios, técnicas e ferramentas. Rio de Janeiro: Guanabara Koogan, 1995.

CAMPANI, C. A. P. **Disciplina de Linguagens Formais**. 2006. Disponível em: <http://www.ufpel.tche.br/~campani/lingformal.pdf>. Acesso em: 02 fev. 2010.

FERREIRA, A. B. de H.; ANJOS, Margarida dos; FERREIRA, Marina Baird. **Novo dicionário Aurélio da língua portuguesa**. 3. ed. Curitiba: Positivo, 2004.

MELO, A. C. **Desenvolvendo Aplicações com UML 2.0**. Rio de Janeiro: Brasport, 2004.

MENEZES, P. B. **Linguagens Formais e Autômatos**. Porto Alegre: Sagra Luzzatto, 2001.

PRICE, A. M. de A.; TOSCANI, S. S. **Implementação de linguagens de programação**: compiladores. São Paulo: Sagra DC Luzzatto, 2001.

ROUTERBOARD. **RB493**. Disponível em: <routerboard.com/RB493>. Acesso em: 27 maio 2012.