UNIVERSIDADE FEEVALE

FERNANDO STRASSBURGER ANDRADE

PROPOSTA DE LINGUAGEM DE PROGRAMAÇÃO PARA EXPRESSÃO DE ESTRATÉGIAS DE NEGOCIAÇÃO NO MERCADO DE AÇÕES

Novo Hamburgo
2014

FERNANDO STRASSBURGER ANDRADE

PROPOSTA DE LINGUAGEM DE PROGRAMAÇÃO PARA EXPRESSÃO DE ESTRATÉGIAS DE NEGOCIAÇÃO NO MERCADO DE AÇÕES

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação pela Universidade FEEVALE.

Orientador: Ricardo Ferreira de Oliveira

Novo Hamburgo
2014

FERNANDO STRASSBURGER ANDRADE

Trabalho de Conclusão do Curso de Ciências da Computação, com título **PROPOSTA DE LINGUAGEM DE PROGRAMAÇÃO PARA EXPRESSÃO DE ESTRATÉGIAS DE NEGOCIAÇÃO NO MERCADO DE AÇÕES**, submetido ao corpo docente da Universidade FEEVALE como requisito necessário para obtenção do Grau de Bacharel em Ciências da Computação.

Aprovado por:

_____
Prof. Ms. Ricardo Ferreira de Oliveira
Universidade FEEVALE
Orientador

_____
Prof. Ms. Gabriel da Silva Simões
Professor Avaliador

_____
Prof. Ms. Juliano Varella de Carvalho
Professor Avaliador

Novo Hamburgo, maio de 2014.

RESUMO

O presente trabalho aborda o desenvolvimento de uma linguagem de programação de propósito específico, a ser utilizada na geração de estratégias de negociação para o mercado de ações, bem como uma ferramenta para interpretação do código desenvolvido, que identificará a ocorrência das estratégias programadas em uma base de dados. Desta forma, pretende-se auxiliar investidores em ações na identificação de suas estratégias de negociação para compra ou venda de ações. Através da linguagem criada, será possível descrever estratégias típicas da metodologia de investimento por análise técnica, de forma a automatizar o processo de identificação destas, informando ao investidor a sua ocorrência, através de uma ferramenta desenvolvida para a interpretação da estratégia programada. A partir da linguagem desenvolvida, serão programadas algumas estratégias compostas por técnicas de análise de ações obtidas na bibliografia utilizada, e testadas sobre uma base de dados histórica de ações da BM&FBOVESPA. Ao final do trabalho, as estratégias descritas através da linguagem são identificadas no histórico da ação analisada, e o ponto de ocorrência de cada uma é informado ao investidor.

Palavras-chave: Análise Gráfica. Bolsa de Valores. Ações. Linguagens de Programação. Compiladores.

ABSTRACT

This paper discusses the development of a special purpose language programming to be used to generate trading strategies for the stock market as well as a tool for the developed code interpretation that identify the occurrence of programmed strategies in a database. Thus, it is intended to assist stock investors to identifying their trading strategies to buy or sell stock. Through the Created language, will be possible describe typical strategies of investment methodology for technical analysis in order to automate the process of identifying these, the investor informing their occurrence through a tool developed to interpret the programmed strategy. From the developed language, some strategies of stock analysis obtained in the bibliography will be programmed and tested over a historical database of BM&FBOVESPA. It is expected that, in the end, the strategies outlined by the user of the language are identified in the historical action analyzed, and the occurrence point of each is informed to the investor .

Keywords : Graphic Analysis . Stock Exchange . Actions . Programming Languages . Compilers

LISTA DE ABREVIATURAS E SIGLAS

BOVESPA	Bolsa de Valores de São Paulo
MM	Média Móvel
MMA	Média Móvel Aritmética
MME	Média Móvel Exponencial
MMS	Média Móvel Simples
IFR	Índice de Força Relativa
GLC	Gramática Livre de Contexto
BNF	Bakus-Naur Form
MQL5	MetaQuotes Language 5

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

INTRODUÇÃO

Nos dias de hoje o mercado de ações é o um dos principais focos dos investidores, em função do grande potencial de ganhos que o mesmo propicia. No Brasil, este mercado já se encontra bastante maduro, sendo que a BM&FBOVESPA, bolsa de valores de São Paulo, atingiu, em 2010, o posto de 2ª maior bolsa de valores do mundo, conforme reportagem da revista Exame (KONCHINSKI, 2013). Em 2012 a principal bolsa brasileira atingiu um total de 191.973.773 papéis negociado, resultando em pouco mais de 1,78 trilhões de reais em ativos negociados, segundo informações fornecidas pela própria BM&FBOVESPA em seu site.

O ganho financeiro nesta forma de investimento pode ocorrer basicamente de duas formas: investindo para ser tornar um “sócio” da empresa, ou operando de forma especulativa. Quando o investidor compra ações de uma empresa pensando em se associar a ela, ele o faz acreditando que esta empresa apresenta boas perspectivas de lucro e crescimento e lhe trarão retorno financeiro no médio ou longo prazo. Já quando um investidor opera de forma especulativa, seu interesse não está na perspectiva futura da empresa, ele apenas identifica a possibilidade de uma reversão na tendência vigente do preço da ação em um curto espaço de tempo, que pode ser de dias, ou até mesmo minutos. Sendo assim, ele compra uma ação, apostando que a mesma irá começar um período de alta do seu preço, e vende a mesma assim que identifica a reversão desta tendência. Seu lucro reside na diferença entre o preço de compra e o de venda.

Conforme Elder (2004), os operadores ganham dinheiro apostando em guinadas de preço. A ideia é comprar quando a leitura do mercado indica que os preços estão subindo e vender quando a tendência de alta começar a perder o gás. É possível também apostar na queda do preço de uma ação e lucrar com isto, operando em uma modalidade chamada de aluguel de ações.

A compra de ações no médio/longo prazo é amparada pelo conhecimento econômico e financeiro que se tem da empresa e o ambiente no qual ela está inserida, sendo que para esta análise dá-se o nome de Análise Fundamentalista. Já o investimento especulativo pode ser amparado por um tipo de análise conhecida

como Análise Técnica (também conhecida como Análise Gráfica[1]). O propósito desta é justamente identificar através de padrões gráficos quando uma ação chegou a um ponto de transição entre tendências. “Os analistas técnicos estudam os movimentos do mercado, tentando identificar padrões recorrentes” (ELDER, 2004 p. 36).

O estudo de padrões nos movimentos do mercado teve seu início no final do século 19, com Charles Henry Dow (BRUM, 2008) e desde então diversos indicadores gráficos surgiram. Desde os mais simples, como a análise de médias móveis, baseada na média do valor de fechamento da ação nos últimos dias, até padrões mais sofisticados, como previsões de força de um movimento envolvendo a sequência de Fibonacci. Para o investidor, a indicação de um momento de compra baseada em um padrão, se constitui em uma possibilidade, e não uma certeza. Essa possibilidade pode ser reforçada, a medida que mais indicadores reforçam as chances. Ou seja, na análise gráfica, a utilização de mais de um indicador gráfico pode ampliar a possibilidade de acerto em uma aposta.

Como a análise gráfica identifica uma possibilidade de alta ou queda em uma ação, é possível que, em um determinado momento, existam papéis com maior probabilidade de alta ou queda que outros. Logo, a diversificação dos papéis também nos ajuda a obter ganhos maiores. Quanto mais diversificada a carteira de ações do investidor, maiores as chances de obter lucro nas operações. Porém, acaba por ser demorado o trabalho de identificar pontos de compra e venda dos ativos, uma vez que se trabalha com diversas regras de negócio, podendo também se ter regras específicas para um ou mais papéis. A análise é custosa, e acaba por reduzir o tamanho da carteira de investimentos do operador, podendo deixar passar uma boa oportunidade de investimento, por uma simples impossibilidade de avaliar o gráfico do ativo.

Para tentar resolver este problema, alguns desenvolvedores de ferramentas para análise técnica de ações disponibilizam, em conjunto com seus *softwares*, um ambiente onde o operador pode programar regras de negociação que, quando identificadas no histórico de preços do papel, geram um sinal de compra ou venda (conforme especificado pelo código), possibilitando assim, que a análise seja feita

[1] Existe uma divergência entre autores e analistas de ações a respeito dos conceitos de análise técnica e análise gráfica. Enquanto alguns abordam ambas como sendo o mesmo tipo de análise, outros defendem que estas possuem conceitos diferentes. O presente trabalho não pretende entrar na discussão, e por questão de conveniência, abordará ambas como sendo nomes diferentes para o mesmo tipo de análise.

computacionalmente. Um exemplo desta solução é a linguagem MQL5 ou *MetaQuotes Language* 5. A linguagem, baseada no C++, é uma proposta da MetaQuotes Software, desenvolvedora da ferramenta MetaTrader para análise técnica de ações. Porém, a MQL5 esbarra na sua complexidade, pois exige do investidor conhecimento a respeito de tipos de variáveis, definição de funções, e até noções sobre orientação a objetos (ver trecho de código em MQL5 no Apêndice C). Tal conhecimento, comum para o desenvolvedor de sistemas, não é tão simples de ser assimilado por alguém que não seja da área de programação, tirando o foco do desenvolvimento de regras de negociação, e levando o esforço para a codificação em si.

Desta forma, define-se o objetivo geral do trabalho, que é o desenvolvimento de uma linguagem de programação de propósito específico que possibilite ao investidor definir técnicas de análise gráfica, e uma ferramenta que interprete a técnica programada e identifique sua ocorrência em uma base de dados. Assim, pode-se obter a automação do processo de análise gráfica, deixando o trabalho mecânico de identificar a ocorrência das técnicas para o *software*, enquanto o operador se dedica ao desenvolvimento de técnicas de negócio eficientes, e a execução das ordens de compra e venda das ações.

Para atingir o objetivo, o trabalho busca identificar a partir de uma pesquisa bibliográfica as principais técnicas utilizadas na Análise Gráfica. Com base no funcionamento das técnicas identificadas, será definida uma linguagem de programação que permita a formalização destas. Em seguida, será construído um compilador que transforme a linguagem desenvolvida em uma linguagem comercial. Além disso, será desenvolvida uma ferramenta que identifique a ocorrência da regra programada pelo investidor em uma base de dados de cotações de ações.

A fim de identificar se a solução criada atende ao objetivo definido, serão programadas técnicas dentro da ferramenta e as mesmas serão submetidas a uma base de dados. Ao final, espera-se que a solução identifique os pontos em que as regras programadas ocorrem.

Este trabalho está dividido em 6 capítulos. No Capítulo 1 é abordado o mercado de ações, com uma breve explicação de seu funcionamento e como os investidores obtém lucros no mesmo. O Capítulo 2 apresenta o funcionamento da análise técnica computadorizada, trazendo exemplos de métodos utilizados para identificar quando uma ação está em um momento oportuno para compra ou venda.

Já no Capítulo 3 o trabalho aborda linguagens de programação e a construção de compiladores. É feita uma apresentação das linguagens livres de contexto (classe a qual pertencem as linguagens de programação) e como estas podem ser descritas. Em seguida, é apresentado o funcionamento básico de um compilador, trazendo técnicas utilizadas para a construção de um. Ao final, são levantados alguns requisitos desejados para uma linguagem destinada especificamente á expressão de estratégias de negociação de ações. Em seguida, no Capítulo 4, é apresentada a solução desenvolvida pelo autor, descrevendo os quatro principais entregáveis do trabalho: a base de dados modelada e utilizada para testes; o software de análise, que fará a interpretação da estratégia descrita; a linguagem proposta para o operador descrever as suas estratégias; e o software de compilação desenvolvido para traduzir o código gerado pelo usuário, para uma linguagem de programação comercial. O Capítulo 5 trás os resultados obtidos a partir da solução proposta. Nele são geradas algumas estratégias de negociação por meio da linguagem proposta, e as mesmas são submetidas para análise sobre a base de dados. Ao final, tem-se a identificação das estratégias na base de cotações da ação, com o alerta de ocorrência sendo trazido ao usuário. O Capítulo 6 trás as conclusões do trabalho, onde é feito um apanhado geral sobre a teoria pesquisada e a prática desenvolvida durante o trabalho, uma análise dos resultados obtidos e uma relação de possíveis melhorias identificadas para trabalhos futuros.

1 MERCADO DE AÇÕES

O presente capítulo abordará o tema sobre o qual o trabalho foi desenvolvido: investimento em ações. Será explicado o que são ações e suas características, como funcionam as bolsas de valores, ambiente no qual ocorrem as negociações de ações, e como um investidor pode lucrar operando nesta modalidade de investimento.

1.1 O QUE SÃO AÇÕES

Conforme Cavalcante, “Uma ação representa a menor parcela do capital social[2] de uma empresa (sociedade anônima, sociedade por ações ou companhia)” (CAVALCANTE, 2005 p. 46). No momento em que uma empresa coloca ações a venda, diz-se que ela está “abrindo seu capital”, ou seja, está disponibilizando fatias de seu capital social para qualquer pessoa que tenha interesse em compra-las e, desta forma, se tornar parte da empresa, um sócio. As ações podem ser classificadas em dois tipos, que definem basicamente a preferência no recebimento dos dividendos e direito a voto em assembleia de acionistas. São elas:

- Ações ordinárias: proporciona ao titular participação nos resultados da empresa e direito a voto em assembleia;
- Ações preferenciais: proporciona ao titular prioridade no recebimento de dividendos e, no caso do encerramento da empresa, tem preferência no reembolso do capital. Não possui direito de voto em assembleia.

Uma ação pode ser negociada em troca de outra ação ou valor em dinheiro, bastando o detentor da mesma encontrar um comprador interessado. O valor da ação vai depender da cotação da mesma no momento do negócio, que por sua vez, é determinado pelo interesse do mercado em adquiri-la. Aplica-se a lei da oferta e da procura: quanto maior a procura pela ação, mais caro pode-se cobrar por ela; já no

[2] O capital social de uma empresa representa o patrimônio líquido da mesma, proveniente de investimento de seus proprietários, a partir de recursos próprios ou de resultados da própria empresa.

momento em que a oferta supera a procura, o comprador se torna o controlador do preço, puxando-o para baixo.

1.2 CONCEITO E FUNCIONAMENTO DAS BOLSAS DE VALORES

Bolsa de Valores é o local onde ações e outros tipos de ativos, como opções e debentures, são negociados. No Brasil, a maior bolsa em funcionamento, sendo também uma das maiores do mundo, é a BM&FBOVESPA.

Antes do advento da tecnologia, as negociações de ações ocorriam exclusivamente de forma física, dentro das dependências das bolsas, com a troca dos papéis entre os investidores durante o chamado “pregão viva voz”. Atualmente, as negociações ocorrem em sua grande maioria de forma informatizada, no chamado “pregão eletrônico”. Todos os negócios são executados de forma eletrônica, sendo que os dados das transações são amplamente divulgados a todos os interessados.

No site da BM&FBOVESPA (<http://www.bmfbovespa.com.br>) é possível acompanhar de modo online o pregão (com atraso de 15 minutos). Todos os dados envolvendo os negócios efetuados são informados, como as ações negociadas, preço dos negócios efetuados, volume negociado, etc.

1.3 INVESTIMENTO EM AÇÕES

Ao comprar ações de uma empresa, o investidor se torna sócio da mesma e, desta forma, recebe parte do seu lucro aferido durante o ano, que é chamado de dividendo. Também se espera que, com o tempo, a empresa venha a crescer, aumentando seu valor de mercado e, consequentemente, valorizando as suas ações. Desta forma, o investidor lucra com o dividendo (caso exista), e pode lucrar no momento em que vender os papéis que possui, obtendo o lucro da diferença entre o preço de compra e o de venda.

Outra maneira de se lucrar com ações é de forma especulativa, onde os operadores investem rapidamente, realizando operações de curto prazo. Nestes casos o investidor não compra a ação pensando em se tornar sócio da empresa e lucrar com o crescimento da mesma. Seu interesse reside nas flutuações diárias do

valor de negociação da ação, que geram altas e baixas rápidas e recorrentes. Assim, o investidor pode comprar uma ação por um valor “x”, e em apenas alguns dias, ou até mesmo algumas horas mais tarde, vender esta mesma ação por “x” + “y”, onde “y” é justamente a valorização que a ação teve durante o período, sendo este o lucro (bruto) da operação.

Identificar qual ação comprar, tanto para o investidor de curto prazo, como para o de médio e longo prazo, não é uma tarefa trivial. Para isso, existem duas formas de análise de ações bastante conhecidas no mercado. A Análise Fundamentalista, que dá suporte a operações de médio/longo prazo, e a Análise Gráfica, que auxilia principalmente nas operações de curto prazo.

1.3.1 Análise fundamentalista

Também conhecida como Análise Fundamental, a Análise Fundamentalista é o campo de estudo de investimentos onde o investidor toma suas decisões de compra e venda de uma ação com base em uma análise econômica. Conforme Brum,

> [...] a análise fundamentalista tem por objetivo avaliar alternativas de investimento a partir do processamento de informações obtidas junto às companhias, partindo do entendimento do contexto macroeconômico e do panorama setorial no qual a companhia se insere (microeconômico). (BRUM, 2008, p. 66)

Desta forma, diversos aspectos que permeiam a empresa na qual se pretende investir são analisados com o intuito de tentar calcular qual seria o valor “correto” para as ações da mesma no presente, identificando se ela encontra-se sub ou sobre valorizada, e identificar se a empresa possui perspectivas de crescimento no médio/longo prazo, com expectativas de bons lucros e valorização de mercado.

Para este tipo de análise, é necessário conhecimento de economia, contabilidade, e outras disciplinas envolvidas na área financeira empresarial. Busca-se identificar diversos aspectos, como o potencial econômico no qual a empresa está inserida, a saúde financeira da empresa através da análise do seu balanço, a capacidade da diretoria de levar a companhia a atingir bons resultados, a conjuntura econômica nacional e internacional, etc.

A análise fundamentalista é mais comum a bancos, fundos de investimentos e corretoras de valores. O investidor que utiliza a análise fundamentalista para orientar

suas aplicações dificilmente opera de forma especulativa. Nestes casos, o interesse é na valorização da empresa no médio/longo prazo, e na distribuição dos dividendos.

1.3.2 Análise técnica

A análise técnica de ações, também conhecida como análise gráfica, tem por objeto de estudo o gráfico formado pelas cotações de uma ação em um dado período de tempo. Teve seu início no final do século 19, quando o analista Charles Henry Dow começou a estudar o gráfico formado pelos preços das ações (BRUM, 2008). Uma das teses do que mais tarde ficou conhecido como Teoria de Dow, o preço de uma ação incorpora todo o conhecimento que o mercado tem a respeito dela naquele momento. Conforme Elder “cada preço é o consenso momentâneo dos participantes do mercado sobre o valor de uma ação” (2004, p. 77), ou seja, para o analista técnico, não é necessário ter conhecimento a respeito da empresa, tampouco entender de economia, toda a informação conhecida a respeito destes está contida no preço que, consequentemente, gera o gráfico.

Outro ponto importante que a Teoria de Dow aborda e que serve de base para a análise técnica, é que o mercado possui tendências, e se move de acordo com elas. Isso significa que, baseado na análise do gráfico de uma ação, pode-se identificar qual a tendência do seu preço. Desta forma, apenas analisando o gráfico de uma ação é possível identificar qual é a tendência vigente (se é de alta, baixa, ou estável) e prever se ela irá continuar; se poderá sofrer uma reversão; ou se irá manter-se estável. De acordo com Matsura, “Não é importante saber por que os preços se movem [...] O importante é saber como os preços se movem.” (2006, p.2). O “como” ao qual o autor se refere é justamente a identificação da tendência do preço.

Para identificar uma tendência ou um ponto de reversão, o analista técnico busca por movimentos recorrentes na cotação das ações. “Os analistas técnicos estudam os movimentos do mercado, tentando identificar padrões recorrentes” (ELDER, 2004, p. 36). A identificação de padrões em gráficos possibilita encontrar pontos em que as chances de ocorrer uma alta ou uma baixa no valor de uma ação são maiores, gerando assim, indicativos de compra e venda para o papel. Estes

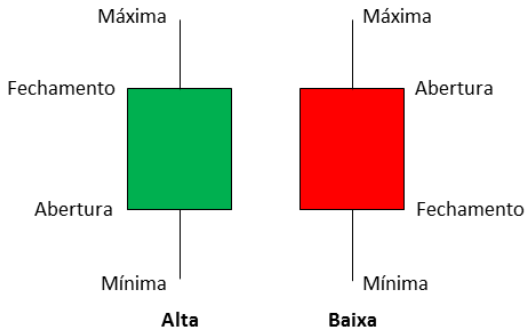
padrões podem ser figuras geométricas formadas pelo gráfico das cotações, relações obtidas através de cálculos realizados com os preços, etc.

Para trabalhar com a análise gráfica existem alguns conceitos básicos, que são apresentados a seguir:

Cotação da ação: é o valor da ação em um determinado momento. São formadas com base em um período, e podem possuir diversas informações a respeito da movimentação do preço neste período. Por exemplo: uma cotação diária representa o valor da ação no dia, podendo trazer informações como os valores máximo e mínimo que a ação atingiu, qual o preço de abertura e qual foi o de fechamento, o preço médio, etc.

Uma das formas mais usuais para se representar uma cotação é o "*candlestick*", ou simplesmente "*candle*". É representado por uma barra que apresenta cinco informações sobre a ação: o preço de abertura/fechamento (corpo), a máxima/mínima (linha vertical acima e abaixo do corpo) e a cor (uma cor representa uma cotação de alta, outra representa de baixa). A Figura 1 demonstra esse gráfico.

Figura 1 - Candlestick



Fonte: Do autor

Gráfico de cotações: é o gráfico formado por um conjunto de cotações em um determinado período. Pode ser formado por cotações mensais, semanais, diárias, horárias, etc. A seguir tem-se um exemplo com um gráfico diário.

Figura 2 - Exemplo de gráfico de ação

Fonte: Do Autor

No exemplo da Figura 2, as cotações estão representadas por *candles*, e trazem valores diários, ou seja, cada *candle* representa um dia de negociação da ação. No eixo horizontal, na parte inferior do gráfico, é mostrado o dia que cada *candle* representa. No eixo vertical direito fica a escala de preços.

Suporte e resistência: conforme Matsura (2006, p.22), suporte e resistência representam o nível de preço no qual a pressão vigente (compradora ou vendedora) supera a oposta, interrompendo a direção atual dos preços. Caracteriza-se como uma linha de suporte ou resistência a recorrência deste ponto, permitindo o traçado de uma linha. Quando o preço da ação encontra-se acima da linha, chama-se de “suporte”, e quando se encontra abaixo, “resistência”. Na Figura 3 tem-se uma linha de resistência desenhada.

É possível observar que o preço da ação atinge duas vezes a linha de resistência, revertendo o movimento de alta. Somente na terceira tentativa o preço consegue romper a linha e seguir o movimento.

Figura 3 - Gráfico com linha de resistência

Fonte: Do autor

Tendência: segundo explica Elder (2004, p. 87), na análise técnica uma tendência se manifesta quando o valor de uma ação sobe ou desce ao longo do tempo. Quando os preços sobem por um período de tempo, tem-se uma tendência de alta, ou seja, a tendência é que os preços continuem subindo, e caso os preços estejam em queda, tem-se uma tendência de baixa. Este movimento não é constante, pois dentro de um movimento de alta ou baixa, os preços oscilam, conforme explica Matsura, “Qualquer que seja a tendência, o movimento não é uniforme: há momentos em que a força entre compradores e vendedores se alternam, mesmo que de forma transitória.” (2006, p.15).

Uma linha de tendência é a ligação dos topos ou fundos de uma tendência, conforme a figura 4. Nesta, é exemplificada uma linha de tendência de alta, pois a mesma conecta os fundos formados pela variação dos preços dentro da tendência de alta. No caso de uma linha de tendência de baixa, a ligação ocorre entre os topos formados em um movimento descendente.

Figura 4 - Gráfico com linha de tendência

Fonte: Do autor

Com base no funcionamento do mercado de ações, e de que forma um investidor pode operar para obter lucros, o próximo capítulo trás uma visão mais aprofundada da análise técnica, apresentando alguns métodos que, exclusivamente a partir do histórico de cotações de uma ação, permitem ao investidor identificar um ponto de reversão desta.

2 ANÁLISE TÉCNICA COMPUTADORIZADA

No início da análise técnica, o desenho do gráfico e posterior análise eram feitas de forma manual. Com uma folha de papel, uma régua e um lápis, o analista desenhava o gráfico da ação, ponto a ponto e, em seguida, traçava linhas de tendência, suporte, resistência, e procurava por padrões geométricos, a fim de encontrar um ponto para comprar ou vender suas ações. “O processo de análise ficava restrito ao gráfico de preço, no qual predominava a identificação de figuras e o traçado das retas de tendência, suporte e resistência.” (MATSURA, 2006 p.68). Era extremamente trabalhoso para o analista acompanhar diversos papéis, pois para cada nova cotação da ação exigia o seu desenho no papel. Acompanhar ações em períodos de tempo *intraday*[3] era praticamente inviável.

Com o advento do computador na análise de ações, o investidor ganhou muito em termos de eficiência. A partir de um servidor que disponibiliza os preços das ações, um *software* especializado pode ler as cotações e desenhar automaticamente o gráfico, atualizando-o sempre que houver um novo valor disponível.

> O computador pode ajuda-lo a acompanhar e a analisar mais mercados em profundidade. Pode assumir tarefas rotineiras e liberar sua mente para raciocínios e reflexões. O computador cria condições para que se utilizem mais indicadores e se localizem mais oportunidades no mercado. O computador é poderosa ferramenta para o processamento de mais informações. (ELDER, 2004 p. 124).

Além de permitir ao analista acompanhar um número maior de ações ao mesmo tempo, o computador também oferece a facilidade de calcular e relacionar diversas variáveis disponíveis em um gráfico, como os preços (de abertura, fechamento, máximo, mínimo e médio), volume negociado de ações, entre outros. Desta forma, foi aberta a possibilidade para a criação de diversas novas técnicas de análise gráfica. As técnicas desenvolvidas desde então, são separadas basicamente em duas classes: rastreadores de tendência e osciladores.

Os rastreadores de tendência são indicadores que monitoram uma ação, tentando identificar algum momento onde o preço da ação diverge ou converge de sua tendência. Se convergir, indica a possibilidade de continuidade da tendência, caso contrário, pode ser que a tendência seja revertida. Já os osciladores são

[3] *Intraday* é o nome dado ao gráfico de cotações com períodos inferiores a um dia. Cotações *intraday* podem ser fechadas a cada 1 hora, meia hora, 10 minutos, etc.

indicadores que procuram indicar pontos específicos de reversão do preço de uma ação. Existem também os chamados mistos, que trabalham sobre um conjunto de ações, e podem tanto indicar um tendência como um ponto de oscilação.

Abaixo são apresentados três indicadores técnicos muito utilizados atualmente: médias móveis, índice de força relativa e estocástico. O primeiro é um rastreador de tendência, enquanto que os outros dois são osciladores.

2.1 MÉDIAS MÓVEIS

Uma das técnicas mais populares para realizar a análise gráfica de uma ação é a análise por médias móveis (MM). Este método se baseia na avaliação do valor médio de fechamento da ação em uma janela de tempo definida pelo analista. A técnica possui diversas variações, como a média móvel aritmética (também conhecida como média móvel simples), média móvel exponencial, média móvel ponderada, entre outras, e também pode ser utilizada de mais de uma forma, como a análise por cruzamento de médias e a análise de suporte e resistência.

Para o cálculo da MMA (média móvel aritmética) de uma ação, é necessário o número de períodos que irão compor a média, e os preços da ação analisada (que vai depender da escala de tempo utilizada no gráfico) no período estipulado.

Quadro 1 - Equação de Média Móvel Aritmética

$$MMA = \frac{V_1 + V_2 + \cdots + V_n}{n}$$

Onde:

- *MMA* = valor da média aritmética
- V = valor da ação (normalmente o preço de fechamento)
- n = número de período para o cálculo

Fonte: Elder (2004)

Para calcular a MMA de 10 períodos (n), a partir de uma base de dados diários de uma ação, soma-se o preço de fechamento (V) da ação nos últimos 10

dias, e divide-se pelos mesmos 10. O resultado é a MMA para aquele dia. No momento em que a ação receber um novo valor de fechamento, a janela da MMA se “move” para a direita, excluindo o primeiro valor da relação (V_1) e incluindo o novo valor no inicio (V_n), gerando a MMA para o novo dia. No final, tem-se para cada dia da ação a sua MMA correspondente.

Outro tipo de média móvel muito utilizado no mercado é a média móvel exponencial (MME). Sua fórmula mais usual é apresentada abaixo, embora possam ocorrer variações:

Quadro 2 - Equação da Média Móvel Exponencial

$$MME = \text{Preço} \cdot \text{K} + MME_{ontem} \cdot (1 - \text{K})$$

$$K = \frac{2}{N + 1}$$

Onde:

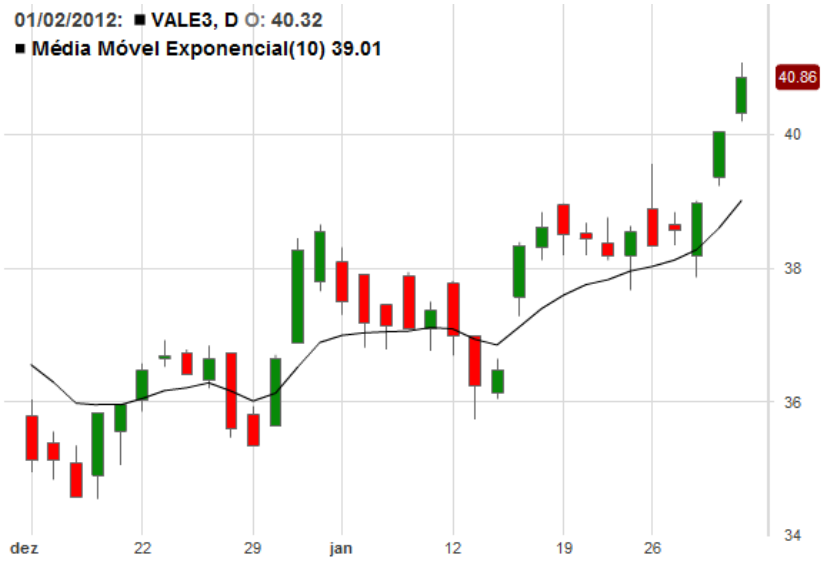
- MME = valor da média exponencial
- Preço = valor da ação (normalmente o preço de fechamento)
- MME_{ontem} = MME do dia anterior
- N = número de períodos

Fonte: Elder (2004)

A grande diferença entre as duas fórmulas é que na MMA, todas as cotações possuem o mesmo peso no cálculo, enquanto que na MME, a última cotação tem peso superior à penúltima, já a penúltima tem peso superior a sua anterior, e assim por diante. Na prática, a MMA dá a mesma importância para todos os valores que se encontram dentro da janela de análise, enquanto que na MME, a importância de um preço no cálculo vai diminuindo à medida que a janela de análise anda sobre o gráfico. Também vale ressaltar que na MMA, quando um valor sai do período, ele não influência mais na média, enquanto que na MME, mesmo fora do período, o valor continua a gerar influência no cálculo, pois permanece no valor da média do período passado (MME_{ontem}).

Para gerar o gráfico das médias, basta fazer a ligação dos pontos obtidos pelos cálculos, conforme segue:

Figura 5 - Gráfico de Média Móvel



Fonte: Do autor

Como as MM são rastreadores de tendência, sua principal função é representar a tendência da ação para aquele momento. Uma MM ascendente indica que a tendência é de alta, dando suporte para operações de compra, enquanto que uma média móvel descendente indica o oposto, ou seja, um momento de baixa.

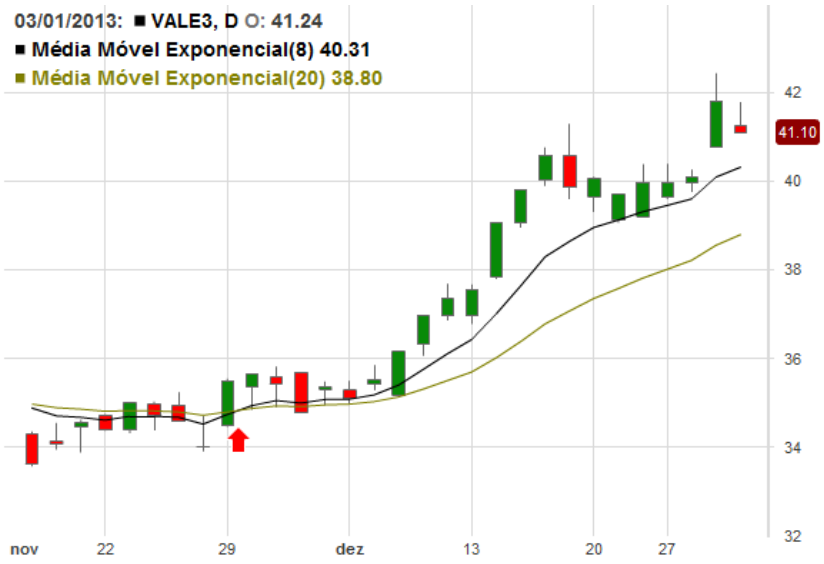
A velocidade com que a média móvel irá responder a tendência vai depender do número de períodos (*n*) utilizados no cálculo. Utilizando um número pequeno de períodos, a média irá responder de forma mais rápida às variações de preço da ação, enquanto que médias calculadas com períodos maiores terão uma inércia maior, levando mais tempo para responder a variação. Em função disto, diz-se que uma média móvel pode ser lenta (período de análise grande) ou rápida (período de análise curto).

Com as MMs, pode-se realizar alguns tipos de análise. Uma delas é o cruzamento de médias. A técnica de análise através do cruzamento de médias móveis exige o cálculo de duas médias sobre uma mesma ação, uma média rápida e uma lenta. Plota-se as médias sobre o gráfico, e procura-se o ponto em que estas se cruzam. No ponto do cruzamento, tem-se um indicativo de compra ou venda. Se

a média lenta estiver subindo, o indicativo é de alta, caso contrário, tem-se um indicativo de baixa.

> O cruzamento de duas médias de períodos diferentes pode ser utilizada como uma regra de compra ou venda, [...]. Quando a média de menor período cruzar de baixo para cima a média de maior período é um sinal de compra. O sinal de venda aparece quando a média de menor período cruzar de cima para baixo a média de maior período. (MATSURA, 2006 p. 72).

Figura 6 - Cruzamento de médias móveis



Fonte: Do autor

Na figura 6 tem-se um cruzamento de médias ocorrendo no ponto indicado pela seta vermelha. A linha de cor amarelada é uma MME de 20 períodos, enquanto que a preta é uma MME de 8 períodos.

São indicadas diversas configurações de período para a análise por cruzamento, como 3/8, 8/20 e 15/35 (média rápida / média lenta). A escolha vai depender da análise do operador. Este tipo de análise pode gerar falsos indicativos, pois pode haver um cruzamento momentâneo, que não representa uma tendência. Para eliminar estes casos, utiliza-se um valor percentual de diferença entre os valores das médias após o cruzamento, ou até antes do mesmo. Desta forma, pode-se verificar se o cruzamento ocorreu com uma diferença relevante entre os valores, e não apenas uma flutuação nos valores. Também se pode aguardar uma quantidade *n* de períodos após o cruzamento para validar o indicador.

2.2 ÍNDICE DE FORÇA RELATIVA (IFR)

O Índice de Força Relativa (em inglês *Relative Strength Index*) é um indicador da classe dos osciladores criado em 1978 por Weller Wilder. A proposta deste indicador é sinalizar quando uma ação está em um nível chamado de “sobrecomprado” ou “sobrevendido”, ou seja, os movimentos de compra ou venda do papel chegaram a uma situação de exaustão. Desta forma, o IFR ajuda o analista a identificar quando um movimento está perdendo sua força, ou quando chega a um provável momento de reversão. “[...] o que se mede é a força relativa dos compradores (fechamentos em alta) em relação aos vendedores (fechamentos em baixa).” (MATSURA, 2006 p. 81).

A fórmula deste indicador é apresentada no quadro 3:

Quadro 3 - Equação do Índice de Força Relativa

$$IFR = 100 - (\frac{100}{1 + \frac{U}{D}})$$

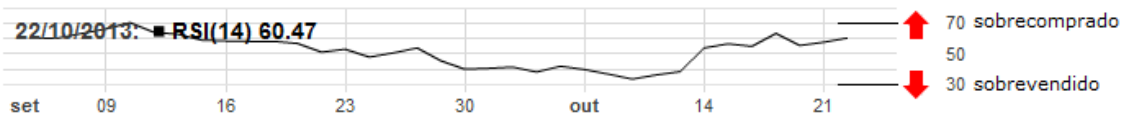
Onde:

- *U* = média das cotações nos últimos *n* dias em que o papel teve alta
- *D* = média das cotações nos últimos *n* dias em que o papel teve baixa

Fonte: Elder (2004)

O IFR fornece um percentual, que é plotado em um gráfico linear vertical, seguindo a mesma escala de tempo do gráfico da ação em análise. Os níveis de “sobrecomprado” e “sobrevendido” são definidos como duas áreas no sentido vertical do gráfico, que normalmente são configuradas como 70% e 30% do total. Abaixo é apresentado um exemplo de um gráfico IFR plotado, bem como as regiões de “sobrecomprado” e “sobrevendido” definidas.

Figura 7 - Representação gráfica do IFR



Fonte: Do autor

No exemplo da figura 7, tem-se um gráfico de IFR. As faixas de 70% e 30% estão sinalizadas a direita.

Em relação ao número de períodos para o cálculo, o criador do indicador sugere a utilização de 14 períodos, mas também é habitual seu cálculo com 9 ou 25 períodos.

O IFR pode ser utilizado de mais de uma forma no apoio à tomada de decisão no momento de compra ou venda de uma ação. Uma das mais conhecidas é a entrada e saída das regiões de “sobrecompra” e “sobrevenda”. No momento em que o gráfico do índice de força relativa deixar uma região de “sobrecompra”, cruzando a linha da região superior e entrando na zona central, tem-se um indicativo de que o papel pode iniciar um movimento de queda do seu valor, o que sinaliza um momento de venda. O raciocínio é análogo à saída da região de “sobrevenda”, ou seja, saindo da faixa inferior no gráfico e entrando na zona central, tem-se um possível movimento de alta, gerando uma possibilidade de compra.

> Quando o RSI ergue-se acima da sua linha de referência superior, isso mostra que os touros [vendedores] estão fortes, mas o mercado está *overbought* [sobrecomprado] e entrando na zona de venda. Quando o RSI cai abaixo de sua linha de referência inferior, isso mostra que os ursos [compradores] estão fortes, mas que o mercado está *oversold* [sobrevendido] e entrando na zona de compra. (ELDER, 2004 p. 175).

Na figura 8, têm-se estes momentos indicados.

Figura 8 - Pontos de sobrecompra e sobrevenda no IFR

Fonte: Do autor

No período compreendido entre setembro e outubro, o IFR entra e logo em seguida sai da região de “sobrecomprado”. Em novembro, tem-se a situação oposta, com o IFR entrando e também em seguida saindo da região de “sobrevendido”.

2.3 ESTOCÁSTICO

Criado em 1950 por George Lane, o oscilador Estocástico calcula a taxa de variação do fechamento da ação em relação ao máximo e mínimo que a mesma atingiu nos últimos *n* dias, e analisa este valor com a média desta mesma taxa nos últimos *n* dias. A ideia por trás deste indicador é que a cotação de uma ação tende a ficar próximo a sua média dos últimos dias. Caso isso não ocorra, tem-se a indicação de um possível movimento de alta ou baixa dos preços. “O estudo conhecido como estocástico enfatiza a relação do preço de fechamento com os máximos e mínimos mais recentes, criando assim regiões de “sobrecompra” e “sobrevenda” “ (MATSURA, 2006 p. 85).

Para gerar um gráfico do indicador estocástico, utilizam-se as duas fórmulas apresentadas a seguir:

Quadro 4 - Equação do indicador Estocástico

$$\%K = \frac{Fech_{hoje} - Min_n}{Max_n - Min_n} \times 100$$

$$\%D = \frac{\sum_3 (Fech_{hoje} - Min_n)}{\sum_3 (Max_n - Min_n)} \times 100$$

Onde:

- $Fech_{hoje}$ = valor de fechamento da ação no dia
- Min_n = valor mínimo da ação nos últimos *n* dias
- Max_n = valor máximo da ação nos últimos *n* dias
- $\sum_3$ = somatório dos últimos 3 $\%K$

Fonte: Elder (2004)

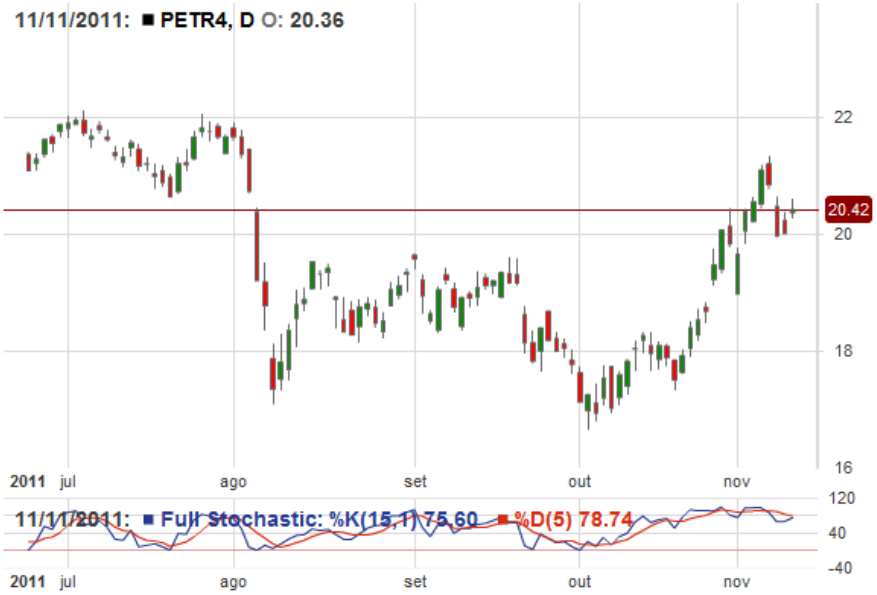
A primeira fórmula, que calcula o valor de $\%K$, indica o quanto o valor de fechamento atual da ação representa dentro da faixa de variação desta ação nos últimos *n* dias. Já o valor de $\%D$, nada mais é que a média de $\%K$ nos últimos 3 dias. Desta forma, observa-se que $\%K$ é um oscilador rápido, pois o movimento atual da ação impacta rapidamente em seu valor, enquanto que $\%D$ é um oscilador mais lento, visto que o último movimento do preço entra em um cálculo de média.

Calculando os valores de $\%K$ e $\%D$ para uma ação, obtêm-se dois gráficos. Ambos são plotados em separado do gráfico da ação. Na figura 9 é apresentado um exemplo. As linhas na parte inferior são %K (linha azul) e %D (linha vermelha).

Através do indicador estocástico, algumas técnicas podem ser exploradas para auxiliar na tomada de decisão na compra ou venda de uma ação.

De forma semelhante a outros indicadores, como o IFR (índice de força relativa), o estocástico indica momentos de “sobrecompra” ou “sobrevenda”. Ou seja, quando o indicador superar um valor percentual, é possível dizer que a ação está “sobrecomprada”, e quando ficar abaixo de um valor percentual, está “sobrevendida”. Na técnica estocástica, estas regiões são normalmente configuradas em 85% e 15%. Na figura 10, é apresentado um exemplo indicando estes pontos.

Figura 9 - Representação do indicador Estocástico



Fonte: Do autor

É possível ver no mês de agosto ambas as situações de esgotamento de níveis de “sobrecomprado” e “sobrevendido”. Inicialmente o papel encontra-se “sobrecomprado”, em seguida, o preço tem uma queda muito grande, indicando um movimento muito forte de venda do mesmo, que o leva a zona de “sobrevenda”.

Figura 10 - Regiões de sobrecomprado e sobrevendido no gráfico Estocástico

Fonte: Do autor

No capitulo a seguir, serão abordadas as etapas envolvidas no processo de compilação de um código, com uma introdução a respeito de linguagens livres de contexto, classe às quais pertencem as linguagens de programação. Também serão elencadas algumas características desejadas para uma linguagem com o propósito específico de descrever estratégias de negociação de ações, baseadas na análise técnica.

3 COMPILADORES

Quando um problema é passado para um computador para ser resolvido, a descrição do mesmo é feita através de uma linguagem de programação, que segundo Price e Toscani (2001), é o meio de comunicação entre o indivíduo que deseja resolver o problema e o computador escolhido para ajuda-lo. Porém, um computador é capaz de entender apenas a linguagem binária. “Um programa de computador é uma sequencia de 0s e 1s armazenada na sua memória” (DELAMARO 2004 p. 1). Desta forma, à medida que os problemas computacionais ganham complexidade, se torna inviável seu desenvolvimento na linguagem da máquina (também chamada de linguagem de baixo nível), sendo necessária a criação de linguagens mais próximas da humana, chamadas de linguagens de alto nível.

Para que uma linguagem de alto nível se torne operacional, é necessário que ela seja traduzida para uma linguagem que o computador possa interpretar. Conforme Price e Toscani (2001), essa conversão é feita por sistemas especializados – compiladores e interpretadores – que recebem uma representação textual do problema, dita linguagem fonte, e produzem uma representação do mesmo algoritmo expresso em outra linguagem, dita linguagem objeto (ou linguagem alvo).

Figura 11 - Trabalho de um compilador

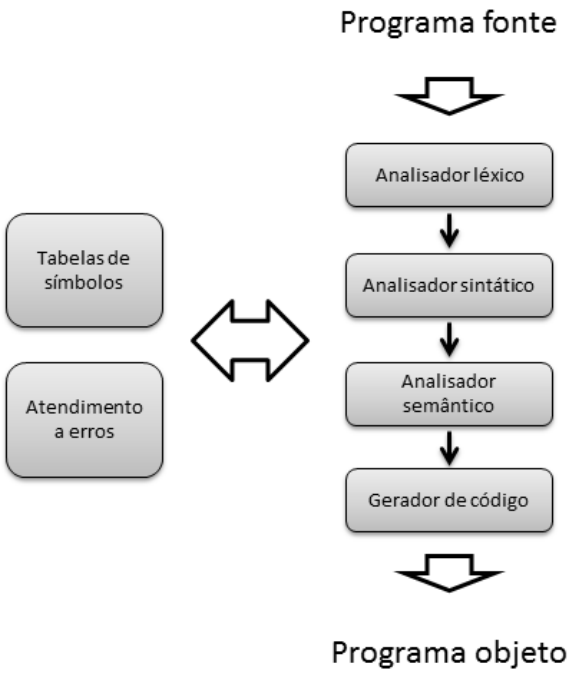
Fonte: Price e Toscani (2001, p.5)

Sendo assim, um compilador é basicamente “um programa que recebe como entrada um programa em uma linguagem de programação e o traduz para um programa equivalente em outra linguagem.” (AHO, SETHI e ULLMAN 2008 p.1).

Ainda conforme Price e Toscani (2001), compiladores em geral são programas bastante complexos, mas devido à experiência acumulada ao longo dos

anos, pode-se chegar a uma estrutura básica para eles. A seguir é representada uma estrutura básica dos componentes de um compilador.

Figura 12 - Exemplo de estrutura de um compilador básico



Fonte: Do autor

Conforme apresentado pela figura 12, o programa fonte é inicialmente tratado por três tipos de analisadores: o léxico, o sintático e o semântico. Estes três compõem a fase chamada de “análise”. Conforme Aho, Sethi e Ullman (2008), nesta etapa o compilador avalia se o programa fonte utiliza a gramática disponível, se está bem formado sintaticamente, e se a semântica está correta. Caso algo de errado seja encontrado, o usuário deve ser informado a respeito, repassando ao mesmo as informações pertinentes ao erro encontrado. Como saída da etapa de análise tem-se uma tabela de símbolos, gerada durante a análise a partir da coleta de informações do programa; e uma representação intermediária do programa, que é passada para a etapa de geração do código, chamada de “síntese”. Ainda segundo Aho, Sethi e Ullman (2008), a parte da síntese constrói o programa objeto a partir da representação intermediária resultante da análise e das informações na tabela de símbolos. Ao final, tem-se o programa objeto pronto para a execução.

Antes de entrar em detalhes a respeito do funcionamento do compilador, é pertinente discutir a formalização das linguagens de programação em geral. O subtítulo seguinte aborda as Gramáticas Livres de Contexto, forma comum de representação de uma linguagem de programação.

3.1 GRAMÁTICAS LIVRES DE CONTEXTO

Conforme Delamaro (2004), “Em geral, uma linguagem de programação pertence a uma classe de linguagens chamada de linguagens livres de contexto. Uma maneira de se definir tais linguagens é por meio das gramáticas livres de contexto.” Ainda segundo Delamaro, uma gramática livre de contexto, ou GLC, pode ser definida da seguinte forma:

Quadro 5 - Formalização de uma gramática livre de contexto (GLC)

(Ω, Σ, *S*, *P*) , onde

Ω – é um conjunto não vazio de símbolos não terminais;
Σ – é o alfabeto sobre o qual a linguagem é definida;
S – é o símbolo inicial da gramática;
P – é um conjunto de produções da forma A → α, onde A ϵ Ω e α ϵ (Σ ∪ Ω)

Fonte: Delamaro (2004)

O Σ é o conjunto dos símbolos terminais, pois representam o alfabeto sobre o qual as palavras da gramática serão formadas. O Ω representa outro conjunto de símbolos, chamado de não terminais. São utilizados para auxiliar na definição das produções *P* da gramática.

Para verificar se uma palavra pertence a uma gramática, a partir do símbolo inicial *S* aplica-se as regras de produção *P* substituindo o símbolo não terminal à esquerda de “→” por uma cadeia de símbolos à direita, que pode ser composta por símbolos terminal, não terminal, ou então “λ”, que representa uma cadeia vazia. A substituição deve ser feita até que restem apenas símbolos terminais à direita, o que

significa que a palavra pertence à gramática. Caso restem símbolos não terminais à direita, e nenhuma regra de produção é capaz de elimina-los, conclui-se que a palavra não pertence à gramática definida.

3.1.1 Forma de Backus-Naur

A Forma de Backu-Naur (BNF) é, segundo Louden, “uma ferramenta poderosa para representar a estrutura sintática de uma linguagem de programação.” (2004, p. 130), sendo esta frequentemente utilizada para este fim (PRINCENTON, 2013). Um exemplo de sua notação é apresentado no quadro 6:

Quadro 6 - Notação de Backus-Naur

```
<não terminal> ::= ‘TERMINAL’ <não terminal>
                 | ‘TERMINAL’
                 | <não terminal>
                 | λ
```

Fonte: Do autor

Na BNF os símbolos terminais e não terminais são todos apresentados dentro das regras de produção, não sendo necessário defini-los em conjuntos separados. A seguir é apresentado um exemplo de gramática definida pela BNF:

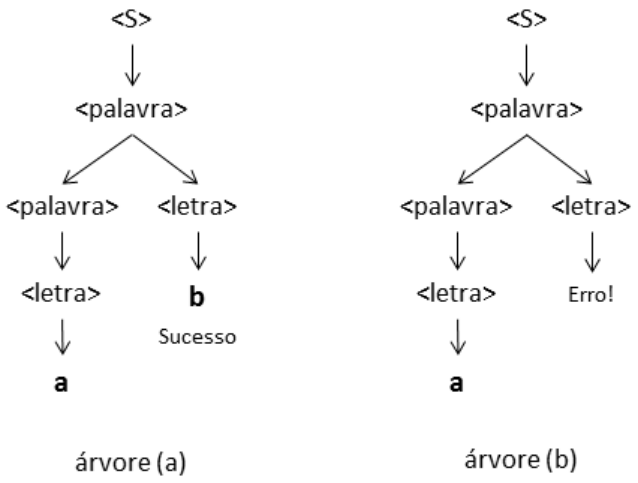
Quadro 7 - Gramática definida na BNF

```
      <S> ::= <palavra>
<palavra> ::= <palavra> <letra>
            | <letra>
  <letra> ::= ‘a’ | ‘b’ | ‘cc’
```

Fonte: Do autor

Para a linguagem definida apresentada no Quadro anterior, a palavra “ab” seria uma produção válida, enquanto que uma produção “ac” seria inválida. Para cada uma destas palavras, tem-se uma árvore de derivação[4] das produções a seguir:

Figura 13 – Exemplo de árvore de derivação



Fonte: Do autor

Na árvore (a) tem-se a derivação da palavra “ab”, que ocorre com sucesso, pois é possível chegar a esta palavra utilizando-se das regras de produção pertencentes a gramática. Já na árvore (b), não é possível derivar, pois não existe nenhuma regra de derivação que leve ao terminal “c”, apenas ao terminal “cc”.

3.2 ANÁLISE LÉXICA

Segundo Aho, Sethi e Ullman (2008), o analisador léxico é responsável por ler o fluxo de caracteres que compõem o programa fonte e agrupa-lo em sequências significativas, chamadas de *lexemas*. Para cada *lexema*, o analisador léxico produz uma saída chamada de *token*. Conforme Price e Toscani (2001), os *tokens* são a unidade básica do texto do programa. Eles representam categorias (também

[4] Conforme AHO, SETHI E ULLMAN (2008), uma árvore de derivação é uma representação gráfica de uma derivação, onde é evidenciado a ordem em que as produções são aplicadas.

chamadas de “classes”) de palavras da gramática utilizada na linguagem fonte. Louden cita alguns exemplos:

> Exemplos típicos são palavras-chave, como *if* e *while*, que são cadeiras fixas de caracteres; identificadores, que são cadeias de caracteres definidas pelo usuário, usualmente compostas por letras e números começando por uma letra; símbolos especiais, como os símbolos aritméticos + e *; e alguns símbolos com mais de um caractere, como >= e <> (2004).

De acordo com Price e Toscani (2001), o analisador léxico interpreta o programa fonte como uma sequência de palavras de uma linguagem regular.

Conforme Louden (2004), o trabalho do analisador léxico é fazer o casamento de padrões, sendo que este casamento pode ser feito por algumas técnicas, entre elas, a de autômatos finitos (sobre autômatos finitos, ver HOPCROFT, ULLMAN e MOTWANI, 2002).

3.3 ANÁLISE SINTÁTICA

Conforme Price e Toscani, o analisador sintático tem como função “verificar se as construções usadas no programa estão gramaticalmente corretas.” (2001 p. 29). Para realizar a tarefa, Aho, Sethi e Ullman, explicam que “o analisador sintático recebe do analisador léxico uma cadeia de *tokens* representando o programa fonte, [...], e verifica se essa cadeia de *tokens* pertence à linguagem gerada pela gramática”. (2008, p. 122). Caso pertença, ele produz uma árvore de derivação para o programa fonte, que será utilizada para gerar o código do programa objeto. Caso não pertença, ele deve lançar um erro, informando que existe um erro sintático no fonte.

Para fazer a análise sintática, pode-se recorrer a duas estratégias básicas, que são a *top-down*, ou descendente; e a *bottom-up*, ou ascendente (também chamada de redutiva).

a) Análise sintática *top-down*: a árvore de derivação para a cadeia de *tokens* recebida é criada de cima para baixo. A análise inicia na raiz e desce até atingir as folhas, sempre percorrendo os nós que se encontram mais a esquerda. Conforme Louden (2004, p.143) existem duas formas de analisadores sintáticos descendentes: analisadores com retrocesso e os analisadores preditivos (como o LL(1)).

b) Análise sintática *bottom-up*: a árvore de derivação é criada a partir das folhas em direção à raiz, através de derivações mais à direita. Também é chamada de análise redutiva, pois o seu processo efetua operações de *redução*, onde a entrada é consumida até restar apenas o símbolo inicial da gramática. Conforme Louden (2004, p.199), os algoritmos de análise sintática ascendente são, de maneira geral, mais poderosos que os descendentes. O analisador sintático LR canônico e analisador sintático LALR são exemplos deste tipo.

3.3.1 Análise sintática descendente recursiva

Neste tipo de analisador tem-se um procedimento a ser executado para cada produção da gramática. Segundo Louden, “A regra gramatical para um *A* não terminal é vista como uma definição de procedimento para reconhecer um *A.*” (2004, p.144). O mesmo traz o seguinte exemplo:

Quadro 8 - Exemplo análise sintática descendente recursiva

```
1  if-decl ::= if (exp) declaração
2            | if (exp) declaração else declaração
3
4
5  procedure declIf
6  begin
7     casamento( if )
8     casamento( ( )
9     exp();
10    casamento())
11    Declaracao();
12    if marca = else then
13       casamento(else);
14       Declaracao();
15    end if;
16 end declIf;
```

Fonte: Louden (2007, p.145)

Pela gramática definida no exemplo (linhas 1 e 2), o não terminal “if-decl” representa o comando “if”, que pode ou não ter a declaração “else”. O procedimento de leitura da produção procura por todos os terminais (em negrito na gramática), e

chama procedimentos de leitura específicos para os “não terminais” (como por exemplo, a chamada do procedimento “exp()” na linha 9). Se o procedimento chegar ao final significa que a análise sintática foi feita com sucesso, caso contrário pode ser lançado um erro (não implementado no exemplo pelo autor) de código.

Louden completa que “A análise sintática descendente recursiva é bastante versátil e é o método mais adequado para um analisador sintático escrito manualmente.” (LOUDEN, 2004 p.143).

3.4 ANÁLISE SEMÂNTICA

Conforme Delamaro (2004), a etapa de análise semântica é responsável por verificar aspectos do código relacionados ao significado de cada comando. Não basta um programa estar gramaticalmente correto, ele também deve fazer sentido. Desta forma, o analisador semântico realiza atividades como identificar se as associações feitas durante o programa podem ser feitas, como, por exemplo, atribuir um texto a uma variável declarada como numérica, encontrar variáveis não declaradas sendo utilizadas, e verificar se regras definidas pela linguagem são seguidas, de forma que o programa seja executável. É também durante esta análise que o código objeto é gerado, como resultado da tradução do código fonte.

3.4.1 Tradução dirigida por sintaxe

A tradução dirigida por sintaxe consiste em uma técnica de análise semântica onde a geração do código objeto ocorre de forma concomitante à análise sintática (PRICE e TOSCANI, 2001, p.85). Desta forma, Louden destaca que “[...] o conteúdo semântico de um programa deve ser fortemente relacionado com sua sintaxe” (2004, p.260), ou seja, deve haver uma relação direta entre uma dada estrutura de palavras de um programa e o significado desta estrutura.

Para efetuar a tradução dirigida por sintaxe, associa-se à uma regra de produção da gramática um bloco de ações semânticas, como gerar um código objeto, armazenar valores ou informações de variáveis, etc. No momento em que uma produção definida na gramática é localizada no código fonte pelo analisador sintático, o bloco semântico correspondente à produção é executado, podendo

assim gerar na saída o código objeto referente ao trecho reconhecido pelo analisador sintático. (PRICE e TOSCANI, 2001, p.85).

Atualmente, todas as linguagens de programação modernas utilizam a tradução dirigida por sintaxe para implementar a análise semântica (LOUDEN, 2004, p.260).

3.5 CARACTERÍSTICAS DESEJADAS DA LINGUAGEM DESENVOLVIDA

Conforme visto anteriormente, o analista técnico dispõe de diversas técnicas computacionais para a análise de ações, que lhe dão suporte na tomada de decisão quando realiza operações de compra e venda. O objetivo da linguagem a ser definida no presente trabalho é dar ao operador a possibilidade de automatizar o trabalho de análise, permitindo-lhe programar em que situação um alerta de compra ou venda deve ser gerado com base em indicadores técnicos, como seguidores de tendência ou osciladores, ambos abordados anteriormente.

A linguagem a ser desenvolvida será de propósito específico, ou seja, será uma linguagem dedicada à expressão de um tipo particular e conhecido de problema. Esta estratégia é mais interessante do que o uso de uma linguagem de propósito geral, pois conforme Price e Toscani, “O desenvolvimento de um programa torna-se mais fácil se a linguagem de programação em uso estiver próxima ao problema a ser resolvido.” (2001, p.1). Para que a linguagem atenda ao propósito, foram definidas algumas características básicas que são citadas abaixo:

Linguagem de alto nível: a gramática e sintaxe da linguagem devem ser de fácil compreensão pelo programador, exigindo o mínimo possível de conhecimento sobre programação. A necessidade de utilização de lógicas algorítmicas, como laços de repetição e recursão devem ser evitadas, pois podem gerar demasiada dificuldade na sua elaboração.

Programação procedural: deve-se exigir do programador apenas os procedimentos a serem calculados e quais os resultados (ou relacionamento entre estes) que devem gerar uma saída. Desta forma, não se exige do operador o entendimento do conceito de objetos, instâncias, herança, etc.

Estrutura dos procedimentos: a estrutura dos procedimentos a serem definidos pelo operador deve ser volta à criação de regras, tanto de compra como de

venda. Deve ser voltada a expressão de quais as condições a serem atendidas para a geração de um sinal, e o que deve ser trazido pelo sinal (como, por exemplo, uma mensagem de texto). Também deve ser fornecida uma estrutura complementar para a realização de cálculos e armazenamento de valores em variáveis. Desta forma, permite-se o desenvolvimento de cálculos em uma parte do programa, e posterior utilização de seus resultados nas regras de negócio, melhorando a legibilidade do código.

Quanto à tipagem de variáveis, a linguagem deve ser do tipo *não tipada*, ou seja, não exige a declaração do tipo das variáveis. Opta-se por esta forma, pois a linguagem utilizará apenas variáveis numéricas, que podem ser preços ou números auxiliares. Não serão utilizadas variáveis de texto, pois o propósito ao qual a linguagem pretende atender não exige operações sobre texto, apenas sobre números.

Como o objeto de estudo da análise técnica é o gráfico da ação, e a menor parte deste é a cotação, é conveniente a existência de uma variável de sistema que possua este valor e permita ao programador trabalhar com ela. Com isto abre-se uma gama de possibilidades para o usuário, que passaria a poder criar suas próprias técnicas de análise com base nas cotações fornecidas.

As regras devem ser criadas sem a especificação das ações sobre as quais elas irão funcionar, sendo que será de responsabilidade da ação chamar a regra e passar por parâmetro seu histórico de cotações para análise. Ou seja, seleciona-se a ação, e inclui-se a regra de negociação a ela. Desta forma, não é exigido do usuário nenhuma passagem de parâmetros, sendo este trabalho do sistema que executará as funções.

4 SOLUÇÃO PROPOSTA

Neste capítulo será apresentada a solução desenvolvida pelo autor para atender ao problema identificado. Inicialmente será apresentada a linguagem modelada para permitir ao investidor definir suas estratégias de negociação. Em seguida, será apresentado o compilador desenvolvido, ou seja, o *software* que faz a tradução do código gerado pelo usuário para um código equivalente em uma linguagem comercial, passível de execução em um computador pessoal. Ao final do capitulo, tem-se um tópico abordando a base de dados de ações e cotações, modelada e populada para ser analisada e servir de subsídios para os testes; e por fim, o *software* de análise desenvolvido, que permite ao usuário indicar as ações a serem monitoradas e definir as regras de negociação para estes papéis, além de interpretar as estratégias de negociação e apresentar o resultado da análise ao usuário.

4.1 LINGUAGEM PROPOSTA

Para atender ao problema identificado, foi modelada uma linguagem de programação, atendendo as características descritas no subcapitulo 3.5, “CARACTERÍSTICAS DESEJADAS DA LINGUAGEM DESENVOLVIDA”. A estrutura geral da linguagem é apresentada na figura 14.

Cada ação possui um conjunto de estratégias de negociação, chamado de “Regras”, sendo este conjunto único para cada ação. Dentro deste, o usuário cria diversas estratégias, chamadas de “Regra”, cada uma para um tipo de alerta que o mesmo deseja ser informado caso ocorra. A expressão da regra ocorre dentro do bloco “Comandos”, que contém “Condições” a serem atendidas para a regra ser dada por ocorrida; “Atribuições”, que são operações para atribuir valores a variáveis; e uma “Saída”, que é a mensagem de alerta gerada pela regra, avisando que ela ocorreu e foi identificada.

Figura 14 – Estrutura geral da linguagem modelada

Fonte: Do autor

A gramática da linguagem é formalmente apresentada no próximo quadro, por meio da BNF. Em seguida, são explicados os principais não terminais e suas produções, bem como exemplos de códigos aceitos.

Quadro 9 – Gramática da linguagem proposta

```
    <regras> ::= <regra> ' ' <regras>
               | <regra>

     <regra> ::= <nome> '{'<comandos>'}'

  <comandos> ::= <comando> ';' <comandos>
               | <comando>';'

   <comando> ::= <atribuicao>
               | <condicao>
               | <saida>
```

```
<atribuicao> ::= <nome> '=' <expressao>

  <condicao> ::= '(' <lista> ')' <comparador> <expressao>

     <saida> ::= 'saida' '(' <texto> ')'

     <lista> ::= <expressao> '&' <lista>
               | <expressao>

 <expressao> ::= <expressao> '+' <termo>
               | <expressao> '-' <termo>
               | <termo>

     <termo> ::= <termo> '*' <fator>
               | <termo> '/' <fator>
               | <fator>

     <fator> ::= '(' <expressao> ')'
               | <funcao>
               | <numero>
               | <nome>

<comparador> ::= [> <]

    <funcao> ::= <nome>'('<parametros>')'

<parametros> ::= <numero> ',' <parametros>
               | <numero>

      <nome> ::= [a-zA-Z][a-zA-Z0-9_]*

     <texto> ::= [a-zA-Z0-9_-!#.]+

    <numero> ::= [0-9]+
```

Fonte: Do autor

- *<regras>* e *<regra>*:

Quadro 10 – Gramática: construção das regras

```
<regras> ::= <regra> ' ' <regras>
         | <regra>
<regra>  ::= <nome> '{' <comandos> '}'
```

Fonte: Do autor

O não terminal *<regras>* representa a estrutura principal que abriga todas as regras para uma determinada ação. Na linguagem objeto (resultante da compilação da linguagem proposta), o não terminal *<regras>* abriga toda a declaração da classe de estratégias para a ação e contém os métodos da mesma (que são as regras e métodos auxiliares, este último não visível pelo usuário e discutido mais a frente). A seguir tem-se um exemplo de construção, onde são apresentadas 3 regras (“Regra1”, “Regra2” e “Regra_N”):

Quadro 11 – Exemplo de construção de regras

```
Regra1 {
...
}

Regra2 {
...
}

Regra_N {
...
}
```

Fonte: Do autor

Já o não terminal *<regra>* define o formato de uma regra em particular. As regras são constituídas por um nome e seu conteúdo é declarado dentro de chaves (“{“ ... “}”). Cabe salientar que duas regras definidas para uma mesma ação não

podem ter o mesmo nome. No quadro anterior são apresentados exemplos de construção, como a “Regra1”.

No código objeto, resultante da compilação, o bloco <*regras*> é representado por uma classe associada a uma ação. Dentro desta classe, cada <*regra*> é traduzida para uma declaração de método, onde o nome da regra será o nome do método. Mais detalhes a respeito da classe que contém as estratégias serão explicados no subtítulo 4.4 “SOFTWARE DE ANÁLISE”.

- <*comandos*> e <*comando*>:

Quadro 12 – Gramática: comandos

```
<comandos> ::= <comando> ';' <comandos>
             | <comando> ';'
<comando>  ::= <atribuicao>
             | <condicao>
             | <saida>
```

Fonte: Do autor

O bloco <*comandos*> especifica a construção das regras. Este pode conter:

- comandos de atribuição: quando é gerada uma variável e atribui-se um valor a ela;
- condições: são as regras de negociação, ou seja, as condições que devem ser atendidas para a regra ser dada por ocorrida;
- saída: informa a saída a ser gerada caso as regras sejam atendidas.

Além de permitir a utilização de uma ou mais variáveis, a construção permite que sejam utilizadas diversas condições a serem atendidas dentro da regra de negociação para que uma saída seja gerada. Esta definição pretende atender a regras mais elaboradas, que podem depender de mais que uma condição para serem geradas.

- <*atribuicao*>:

Quadro 13 – Gramática: comando de atribuição

```
<atribuicao> ::= <nome> '=' <expressao>
```

Fonte: Do autor

Uma atribuição ocorre quando se tem um “nome”, que é o nome da variável, o terminal ‘=’ (sinal de igualdade), e uma expressão. Em exemplo é apresentado a seguir:

Quadro 14 – Exemplo de atribuição

```
var1 = 55;
varX = var1 + 10;
```

Fonte: Do autor

Na compilação, este trecho é traduzido para uma declaração de variáveis do tipo *int*, o restante permanece idêntico ao código fonte.

- <*condicao*>:

Quadro 15 – Gramática: especificação de uma condição da regra

```
<condicao> ::= '(' <lista> ')' <comparador> <expressao>
<lista>    ::= <expressao> '&' <lista>
             | <expressao>
```

Fonte: Do autor

O não terminal <*condicao*> define a construção de uma condição específica dentro da regra de negociação. Como as condições são formadas basicamente por comparações entre valores, a sua estrutura define uma lista de valores e os compara com outro valor, resultante de uma expressão. A seguir é demonstrado um exemplo:

Quadro 16 – Exemplo de condição

```
(media3 & media8 & media20) > cotacao.fechamento();
```

Fonte: Do autor

Segundo a condição definida, é feita a comparação entre o valor das variáveis “media3”, “media8” e “media20” e “cotacao.fechamento()”. A condição é considerada verdadeira caso o valor armazenado em cada uma das 3 variáveis informadas na lista seja maior que o valor retornado pela função “cotacao.fechamento()”. Optou-se por este formato uma vez que são comuns as comparações de diversos valores diferentes com um valor básico (como, por exemplo, o preço de fechamento de uma ação), aperfeiçoando-se o código para comparações, facilitando sua entrada pelo usuário.

No código objeto, as condições são incluídas em um comando *if* e separadas pelo operador “&&” (operador lógico “E”). Exemplos deste trecho após a compilação podem ser vistos no capitulo 5, “RESULTADOS OBTIDOS”.

- <*saida*>:

Quadro 17 – Gramática: comando de saída

```
<saida> ::= 'saida' '(' <texto> ')'
```

Fonte: Do autor

A saída de uma regra é especificada por <*saida*>, que define a mensagem a ser gerada pela estratégia, caso esta ocorra. Um exemplo é apresentado a seguir:

Quadro 18 – Exemplo de saida

```
saida(mensagem de saída);
```

Fonte: Do autor

Conforme definido no exemplo, quando a regra ocorrer, a mensagem “mensagem de saída” será lançada, informando que esta foi identificada. Caberá posteriormente ao *software* de análise fazer o tratamento desta mensagem para apresentar ao usuário (esta etapa será abordada no subtítulo 4.4, “SOFTWARE DE ANÁLISE”).

A saída é declarada dentro do comando *if* do código compilado. A declaração deste comando consiste em setar um atributo de um objeto do tipo “Saida” e em seguida incluir este objeto em um vetor deste tipo de objeto, para posteriormente o *software* de análise fazer sua leitura. Uma melhor explicação a respeito desta etapa é dada no subtítulo 4.4, “SOFTWARE DE ANÁLISE”, e alguns exemplos do código após a compilação podem ser vistos no capitulo 5, “RESULTADOS OBTIDOS”.

- <*expressao*>:

Quadro 19 – Gramática: expressões

```
<expressao> ::= <expressao> '+' <termo>
            | <expressao> '-' <termo>
            | <termo>
<termo>     ::= <termo> '*' <fator>
            | <termo> '/' <fator>
            | <fator>
<fator>     ::= '(' <expressao> ')'
            | <funcao>
            | <numero>
            | <nome>
```

Fonte: Do autor

O não terminal <*expressao*> permite a criação de expressões aritméticas utilizando-se as 4 operações básicas, variáveis e funções. A utilização das 2 primeiras é comum na grande maioria dos algoritmos e não serão abordadas de forma aprofundada neste trabalho.

Já as funções possuem um papel muito específico. Para permitir a expressão das estratégias, além de utilizar variáveis e cálculos matemáticos, a linguagem deve fornecer dois subsídios: acesso aos dados de cotações e indicadores previamente programados, e são as funções que garantem o acesso a estes dados. Ambos foram feitos diretamente na linguagem Java, em função da necessidade de utilizar *loops* e manipulação de matrizes.

O acesso aos dados de cotações é fornecido através da função "cotacao.<dado desejado>(<parametro>)", onde o "dado desejado" é o valor que o usuário quer obter do histórico de cotação e o "parametro" é alguma especificação requerida pela função. Um exemplo da utilização deste recurso é apresentado a seguir:

Quadro 20 – Exemplo de função

```
cotacao.aberturaAnterior(2)
cotacao.fechamento()
```

Fonte: Do autor

A primeira função retorna o valor de abertura da ação de 2 dias atrás (indicado pelo número "2" passado como parâmetro). A segunda função retorna o preço de fechamento da última cotação, sendo que para esta não é necessária a passagem de nenhum parâmetro. Desta forma, o usuário não necessita acessar tabelas do banco de dados, manipular matrizes ou trabalhar com *loops* para obter um valor. Uma lista das funções sobre dados de cotações é fornecida no apêndice A, "LISTA DE FUNÇÕES SOBRE DADOS DE COTAÇÕES".

Os indicadores são fornecidos de forma semelhante aos dados de cotações, ou seja, como funções, porém, sem nenhum prefixo. O usuário entra com o nome do indicador (que é o nome da função), os parâmetros exigidos e recebe o valor

calculado. No apêndice B, “LISTA DE INDICADORES”, é apresentada a relação de indicadores disponibilizados pelo software.

As funções, já no código objeto, são convertidas em chamadas de métodos com o mesmo nome, existentes em classes auxiliares. A instanciação destas classes bem como a declaração dos objetos para invocar os métodos são geradas pelo compilador. Exemplos do código final podem ser vistos no capitulo 5, “RESULTADOS OBTIDOS”.

4.2 SOFTWARE DE COMPILAÇÃO

O *software* de compilação é o encarregado de traduzir o código gerado pelo usuário para uma linguagem comercial. Para o trabalho, a linguagem escolhida foi a Java, principalmente em função de ser a mesma linguagem utilizada para desenvolver o *software* de análise, mas também por ser uma linguagem de domínio do autor.

O compilador foi desenvolvido utilizando-se as técnicas de análise sintática descendente recursiva, para a análise sintática, e a tradução dirigida por sintaxe, para a análise semântica. A análise léxica, mais simples das 3 etapas, foi desenvolvida a partir de um método que busca os *tokens* no texto e devolve estes ao programa principal.

4.2.1 Analisador léxico

Para identificar os *tokens*, utilizou-se dois métodos chamados “moveLookAhead” e “buscaProximoToken”. A análise léxica realizada por estes métodos consiste em ler caractere a caractere o código digitado pelo usuário, gerando *lexemas*, e classifica-los como *tokens*. O primeiro lê o texto digitado pelo usuário, pulando posições em branco, verificando final de linha e final do arquivo. O segundo gerencia a leitura destes caracteres, formando *lexemas* a partir da concatenação dos caracteres retornados, e buscando o *token* correspondente. A seguir, no quadro 21, é apresentado um resumo do algoritmo que identifica e classifica os *tokens* encontrados:

Quadro 21 – Resumo do algoritmo de análise léxica

```
Se inicia com letra
      Le o resto da palavra (letras, números e “_”)
      Token = nome
      Se lexema = saída, token = saida
      Se próximo igual a “.”
            Token = cotação
            Le resto do lexema
      Se próximo igual a “(“
            Se não é cotação ou saída, token é funcao
            Se é saída, le a saída e concatena ao lexema atual
            Senao, le e concatena ao lexema atual
Se inicia com número, token é numero
Se não é letra nem numero, é um símbolo, e busca na lista de símbolos o
token correspondente
```

Fonte: Do autor

No algoritmo, existe uma verificação geral, que tenta classificar a entrada do usuário como um nome, um número, ou um símbolo (*token* formado por um único caractere, como o sinal de igual “=”). Sendo um símbolo ou um número, a verificação é simples, em função da reduzida quantidade de possibilidades que existe para classificar o *lexema* lido. Já quando a entrada inicia por uma letra, a verificação é mais complexa, pois é necessário identificar se o *lexema* se trata de um nome, um comando de saída, ou uma função. Durante toda a leitura, é verificado se os caracteres digitados pertencem à gramática definida pela linguagem para o tipo de entrada. Caso não pertença, é lançado um erro informando a posição do caractere inesperado.

4.2.2 Analisador sintático

O analisador sintático foi construído a partir da técnica descendente recursiva, que empilha funções recursivamente à medida que estas vão sendo identificadas, para, ao fim, informar se o programa gerado pelo usuário pertence à gramática que define a linguagem. Além disso, é o analisador sintático o responsável por ordenar a

busca pelos *tokens* do código e consumi-los, ou seja, a análise léxica ocorre ao mesmo tempo em que a sintática e sob o controle desta.

Para realizar a análise sintática descendente recursiva, foram criados métodos correspondentes aos não terminais da gramática, e dentro dos métodos foram expressas as regras gramaticais dos mesmos, ou seja, as regras que identificam se uma produção é valida ou não. Um exemplo retirando do código é apresentado e explicado a seguir:

Quadro 22 – Exemplo do código de análise sintática

```
// Gramática:
//<regras> ::= <regra> ' ' <regras>
//          | <regra>
//<regra>  ::= <nome> '{' <comandos> '}'

buscaProximoToken();
static void regras() {
      regra();
      if (token == T_NOME) {
            regras();
      }
      if (token != T_FIM_FONTE) {
            erroSintatico("FIM DO FONTE");
      }
}

static void regra() {
      if (token == T_NOME) {
            buscaProximoToken();
            if (token == T_ABRE_CHAVES) {
                  buscaProximoToken();
                  comandos();
                  if (token == T_FECHA_CHAVES) {
                        buscaProximoToken();
                  } else {
                        erroSintatico("FECHA CHAVES");
                  }
            } else {
                  erroSintatico("ABRE CHAVES");
            }
      } else {
            erroSintatico("NOME ESPERADO");
```

```
        }
}
```

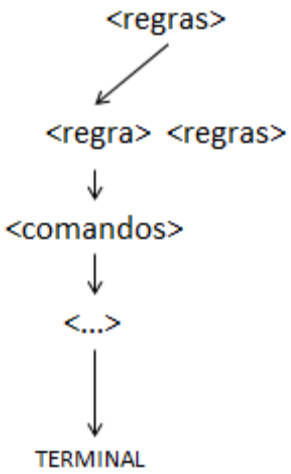
Fonte: Do autor

Conforme definido pela gramática, um bloco de regras (<*regras*>) pode conter uma ou mais regras (<*regra*>). Sendo assim, o método “regras()” (que verifica as produções para o não terminal <*regras*>) valida a primeira <*regra*> especificada e verifica se após esta existe um “nome” (que indica o início de uma nova regra). Caso exista, o método “regras()” é chamado novamente, e examina a próxima <*regra*> encontrada. Desta forma, é possível criar quantos blocos de regras forem necessários para uma ação. Após a validação da última regra, o próximo *token* esperado é o que indica o final do código fonte (código digitado pelo usuário) , sendo assim, é verificado se o *token* lido é igual a “T_FIM_FONTE”, que representa esta situação.

Já no método que define uma regra (“regra()”), é verificado se o código entrado pelo usuário inicia com um “nome”, e em seguida verifica-se a existência do terminal “{“. Caso exista, o método que define os comandos é chamado (“comandos()”). Após, verifica-se o fechamento do bloco com o símbolo “}” e retorna-se ao método anterior, no caso “regras()”. Se o símbolo lido não for o esperado o programa lança um erro, através do método “erroSintatico()”, indicando o que deveria ter sido encontrado.

A análise sintática descendente recursiva opera realizando a derivação dos não terminais mais a esquerda. O exemplo de código apresentado pode ser exemplificado através da figura 15. Ao ser aberto um bloco <regras>, o mesmo é derivado em uma de suas produções, no caso <regra>, primeiro não terminal da esquerda para a direita, e parte para validar o mesmo. Esta validação pode implicar em uma nova derivação, indo para a validação de outro não terminal (<comandos>), e assim sucessivamente, até que seja encontrado um símbolo terminal. Neste ponto, o volta um nível e valida o próximo não terminal, e assim por diante, até que restem apenas símbolos terminais, indicando que as produções encontradas estão de acordo com o definido pela gramática da linguagem.

Figura 15 – Derivação dos não terminais mais a esquerda



Fonte: Do autor

No código, o primeiro método chamado (“Regras()”) somente se encerra ao final da verificação de todo o código. Caso nenhum erro tenha ocorrido durante a análise sintática, o programa finaliza com sucesso.

4.2.3 Analisador semântico

A última etapa da compilação é de responsabilidade do analisador semântico, e este foi desenvolvido através da técnica de tradução dirigida pela sintaxe. Neste método, a tradução do código ocorre durante a análise sintática, ou seja, ao se identificar uma produção da gramática, o código objeto referente a mesma é gerado logo em seguida. O quadro 23 apresenta um exemplo da análise semântica:

Quadro 23 – Exemplo do código de análise semântica

```
1 static void regra() {
2     if (token == T_NOME) {
3           buscaProximoToken();
4           regraSemantica(3);
5           if (token == T_ABRE_CHAVES) {
6                 buscaProximoToken();
7                 comandos();
8                 if (token == T_FECHA_CHAVES) {
```

```
9                       buscaProximoToken();
10                      regraSemantica(4);
11                } else {
12                      erroSintatico("FECHA CHAVES");
13                }
14          } else {
15                erroSintatico("ABRE CHAVES");
16          }
17    } else {
18          erroSintatico("NOME ESPERADO");
19    }
20 }
21
22 ...
23
24 static void regraSemantica(int numeroRegra) {
25    switch (numeroRegra) {
26    ...
27    case 3:
28          codigoJava.append("\n\n\tpublic SinalBean ");
29          codigoJava.append(ultimoLexema);
30          codigoJava.append("(ArrayList<CotacaoBean> cotacoes) {\n");
31          codigoJava.append("Indicadores i = new Indicadores();\n");
32          codigoJava.append("ComandosCotacao c = new
ComandosCotacao(cotacoes);\n");
33          codigoJava.append("SinalBean s = new SinalBean();\n\n");
34          condicaoSe = new StringBuffer();
35          break;
36    case 4:
37          codigoJava.append("\t\tsinais.add(s);\n");
38          codigoJava.append("\t}\n\n\n");
39          break;
      ...
```

Fonte: Do autor

Durante a análise da produção de uma <*regra*>, ao se identificar o nome da regra, é chamado o método gerador de código (“regraSemantica(n)”, linha 4), e como parâmetro passa-se um número, referente a regra semântica a ser utilizada para a produção identificada. Já dentro do método “regraSemantica”, verifica-se qual é a regra a ser acionada (linha 26) e é concatenado o código Java para aquela produção em um “StringBuffer” (linhas 28 a 33). No exemplo apresentado

anteriormente, foi identificada uma nova <*regra*>, que possui um “nome”. Ao gerar o código Java, este “nome” é utilizado para nomear um método (linha 29). Feita a geração do código, é retornado ao método “regra()”.

Seguindo o código da produção de uma <*regra*>, ao fechar um parêntese é gerada uma nova regra semântica, agora com o código “4” (linha 10). Esta regra inclui outro trecho ao código, e novamente retorna ao ponto chamador. Desta forma, verifica-se que à medida que as produções sintáticas são identificadas no código do usuário, o código Java correspondente é criado, concatenando-se o identificado ao já existente, sendo que ao final da análise sintática, a semântica também se dá por concluída.

Ao final da compilação, a variável “codigoJava” contém o código Java correspondente ao código informado pelo usuário. Esta variável é salva pelo programa em um arquivo para posteriormente ser consumida pelo *software* de análise.

4.3 BASE DE DADOS

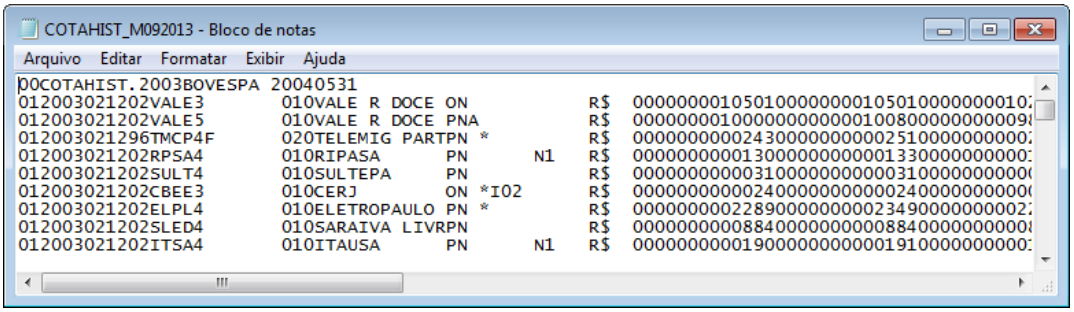
Os dados a serem utilizados foram obtidos diretamente no site da BM&FBOVESPA em duas fontes citadas abaixo. Como a bolsa apenas disponibiliza arquivos com cotações diárias, optou-se trabalhar com estes.

- títulos negociados: arquivo de texto (.txt) disponível em <http://www.bmfbovespa.com.br/cias-listadas/titulos-negociaveis/BuscaTitulosNegociaveis.aspx?Idioma=pt-br>, que contém todos os títulos negociados na bolsa, tipo de mercado, entre outros dados;
- cotações: arquivo de texto (.txt) disponível em < <http://www.bmfbovespa.com.br/pt-br/cotacoes-historicas/FormSeriesHistoricasArq.asp>. Contém as cotações diárias para as ações em séries diárias, mensais e anuais.

A modelagem do *software* também permite trabalhar com cotações *intraday*, bastando gerar um conector para obter de alguma fonte os dados e persisti-los no banco. Para o *software*, a leitura dos dados é a mesma.

A especificação do *layout* do documento é fornecida nas mesmas páginas onde é feito o *download*. Na figura 16 tem-se um exemplo de como as informações são fornecidas no arquivo.

Figura 16 - Fonte de cotações BM&FBOVESPA

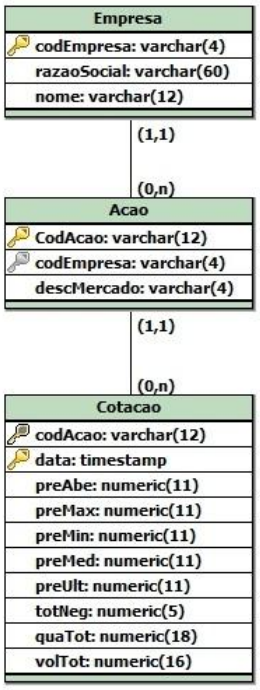


Fonte: Do autor

A base de dados do trabalho foi criada sobre o SGBD HSQLDB, banco de dados relacional Java, que possui algumas características desejadas, como a portabilidade (não necessita instalação na máquina do usuário), suporte a linguagem SQL, e pouco espaço ocupado em disco (menos de 700kb). Dentro do banco, foram criadas 3 tabelas para armazenar os dados, conforme apresentado pela figura 17.

Conforme o esquema apresentado, cada ação pode não possuir cotação (caso em que o papel ainda não foi negociado no período analisado) ou então diversas cotações (quando o papel é negociado no período). Cada ação, por sua vez, pertence (ou é emitida) por uma empresa. A tabela “Empresa”, apesar de não ser necessária para o objetivo do trabalho, foi adicionada ao esquema, em função da informação já estar presente em um dos arquivos, não ter um custo alto de carga, e contribuir com a riqueza de informações acerca do papel negociado.

Figura 17 – Modelo da base de dados



Fonte: Do autor

4.4 SOFTWARE DE ANÁLISE

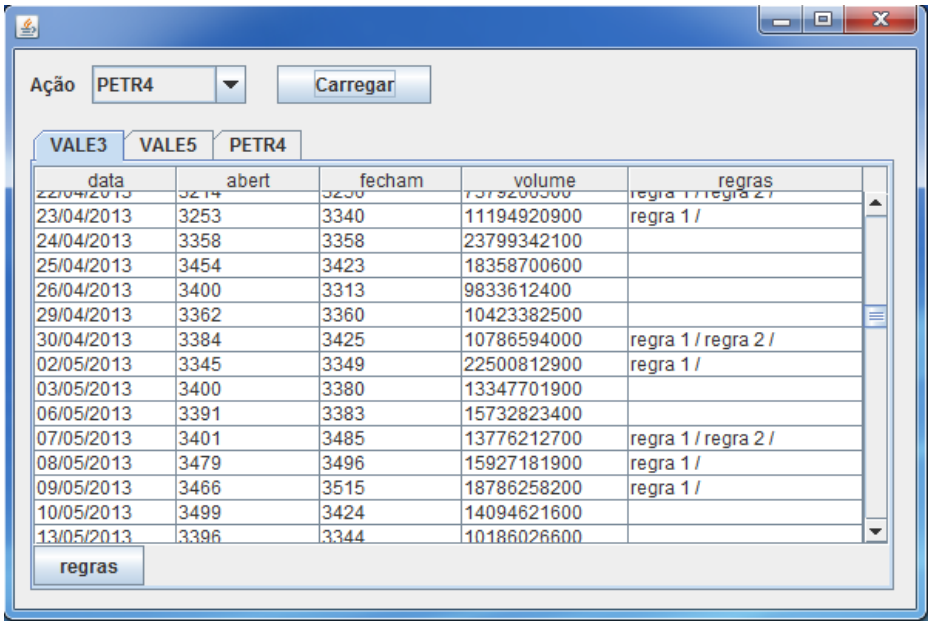
Para integrar todas as partes da solução desenvolvidas até o momento, bem como permitir o usuário opera-las, foi desenvolvido um *software* de análise. Seus requisitos são listados abaixo:

- Disponibilizar uma interface gráfica para operar as suas funcionalidades;
- Dar acesso ao usuário à relação de ações negociadas na bolsa de valores (especificamente a BM&FBOVESPA);
- Permitir ao usuário monitorar ações, tendo acesso às cotações históricas destas, e visualizar os momentos em que as estratégias codificadas para a mesma ocorreram;
- Fornecer um ambiente onde o usuário possa codificar as regras de negociação da ação monitorada, bem como modifica-las quando necessário;
- Compilar as estratégias geradas;

- Fazer uma varredura do histórico das ações monitoradas, buscando os momentos em que as regras especificadas para as mesmas ocorreram, e mostrar este resultado ao usuário;

Tanto o *software* de análise como a interface foram desenvolvidos para ambiente *desktop* utilizando-se a linguagem de programação Java. A figura 18 é apresentada a interface do programa:

Figura 18 – Tela do software de análise



Fonte: Do autor

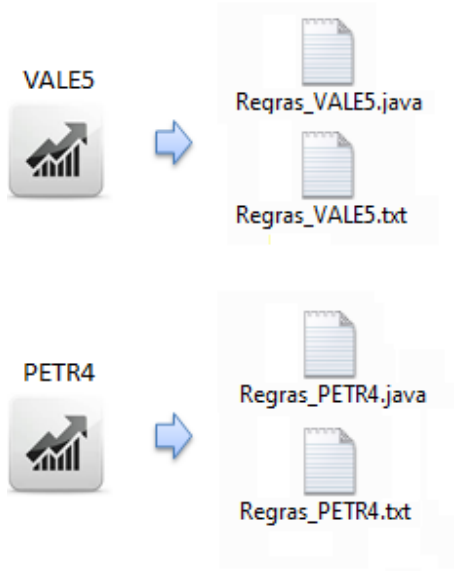
A interface conta com uma *combobox* onde o usuário seleciona o papel que deseja monitorar; um botão para carregar o papel selecionado; um painel com guias onde são apresentadas as cotações dos papéis em monitoração; e um botão onde é possível visualizar e editar as regras específicas da ação que está sendo visualizada.

Ao abrir o programa, o *software* carrega a tabela “Acao” na *combobox* e aguarda o usuário selecionar um papel e clicar em “Carregar”. A carga do papel ocorre buscando o código do papel selecionado através da *combobox* na tabela “Cotacao”, e gerando uma aba para este, contendo uma tabela e o botão “regras”. Como cada cotação possui diversos dados, selecionaram-se apenas quatro para

serem apresentados na tabela: data (que é o dia da cotação), preço de abertura, preço de fechamento e volume.

Conforme já mencionado no subtítulo 4.1, “LINGUAGEM PROPOSTA”, foi definido que as regras de negociação são armazenadas como métodos em uma classe (arquivo “*.java”). Junto a este arquivo, fica o arquivo que contém o código gerado pelo usuário na linguagem modelada (arquivo “*.txt”). A figura 19 ajuda a demonstrar a solução:

Figura 19 – Arquivos com o código fonte e compilado



Fonte: Do autor

Quando o papel é carregado, o *software* executa a classe Java que contém as regras associadas a ele (“*.java”). A seguir, é apresentado o trecho do código que faz esta execução e uma explicação de seu funcionamento:

Quadro 24 – Código de execução das regras da ação

```
1  private String getSinalNegociacao(ArrayList<CotacaoBean>
   listaParaAnalise, AcaoBean acao) {
2        StringBuilder sinal = new StringBuilder();
3        Class cls = Class.forName("regras.Regras_" +
   acao.getCodAcao());
4        Constructor ctr =
   cls.getConstructor(listaParaAnalise.getClass());
5        Object obj = ctr.newInstance(listaParaAnalise);
6
```

```
7         Method getSinais = cls.getMethod("getSinais", null);
8         Object retobj = getSinais.invoke(obj, null);
9         ArrayList<SinalBean> sinais = (ArrayList<SinalBean>) retobj;
10
11        Iterator<SinalBean> iter = sinais.iterator();
12        while (iter.hasNext()) {
13              String regra = iter.next().toString();
14              if (regra != null) {
15                    sinal.append(regra + " / ");
16              }
17        }
18  ...
```

Fonte: Do autor

Conforme exemplificado na figura 18, as classes são chamadas de “Regras_<código da ação>”. Desta forma, a linguagem Java nos permite instanciar um objeto do tipo “*Class*” a partir do código do papel selecionado (linha 3). Tendo a classe que contém as regras do papel selecionado, obtêm-se o construtor da classe, e em seguida, um objeto desta é instanciado (linhas 4 e 5). Internamente, a classe que contém as regras irá executar todos os métodos existentes nela (ou seja, todas as regras de negociação declaradas), e salvará o retorno destes em uma variável da classe (esta etapa será explicada em seguida, quando for analisada a classe que contém as regras). Em seguida, invoca-se um método chamado “getSinais()”, que retorna uma lista contendo todos os sinais gerados pelas regras executadas e identificadas (linhas 7 e 8).

Para apresentar os sinais obtidos, é gerada uma linha de texto onde todas as mensagens de saídas recebidas são concatenadas, separadas por uma barra (“/”) (linhas 10 até 15). Por fim, esta linha é devolvida ao programa principal e apresentada na coluna “regras” da tabela.

A seguir, no quadro 25, tem-se o código existente na classe das regras de negociação de uma ação (o exemplo utiliza as regras da VALE3). Ao ser instanciada, a classe recebe uma lista de cotações por meio de seu construtor (linha 5), que é a lista de cotações sobre a qual todas as regras serão analisadas. Em seguida, é obtida uma lista de todos os métodos existentes na classe (linha 8), e é executado um laço, que invoca todos estes métodos (linhas 9 e 10). Cada método executado verifica a ocorrência da regra (linha 19) sobre a lista de cotações fornecida e, caso identifique a sua ocorrência, insere na lista de sinais a sua saída (linha 20 e 22).

Quadro 25 – Código executado na classe das regras da ação

```
1  …
2  private ArrayList<CotacaoBean> cotacoes;
3  private ArrayList<SinalBean> sinais;
4
5  public Regras_VALE3(ArrayList<CotacaoBean> cotacoes) throws
   IllegalArgumentException, IllegalAccessException,
   InvocationTargetException {
6        this.cotacoes = cotacoes;
7        sinais = new ArrayList<SinalBean>();
8        Method m[] = this.getClass().getDeclaredMethods();
9        for (int i = 0; i < m.length; i++) {
10             m[i].invoke(this, null);
11       }
12 }
13
14  private void regra1() {
15       Indicadores i = new Indicadores();
16       ComandosCotacao c = new ComandosCotacao(cotacoes);
17       SinalBean s = new SinalBean();
18
19       if (c.fechamento() > c.abertura()) {
20            s.setMensagem("regra 1");
21       }
22       sinais.add(s);
23  }
24
25  public ArrayList<SinalBean> getSinais() {
26       return sinais;
27  }
28
```

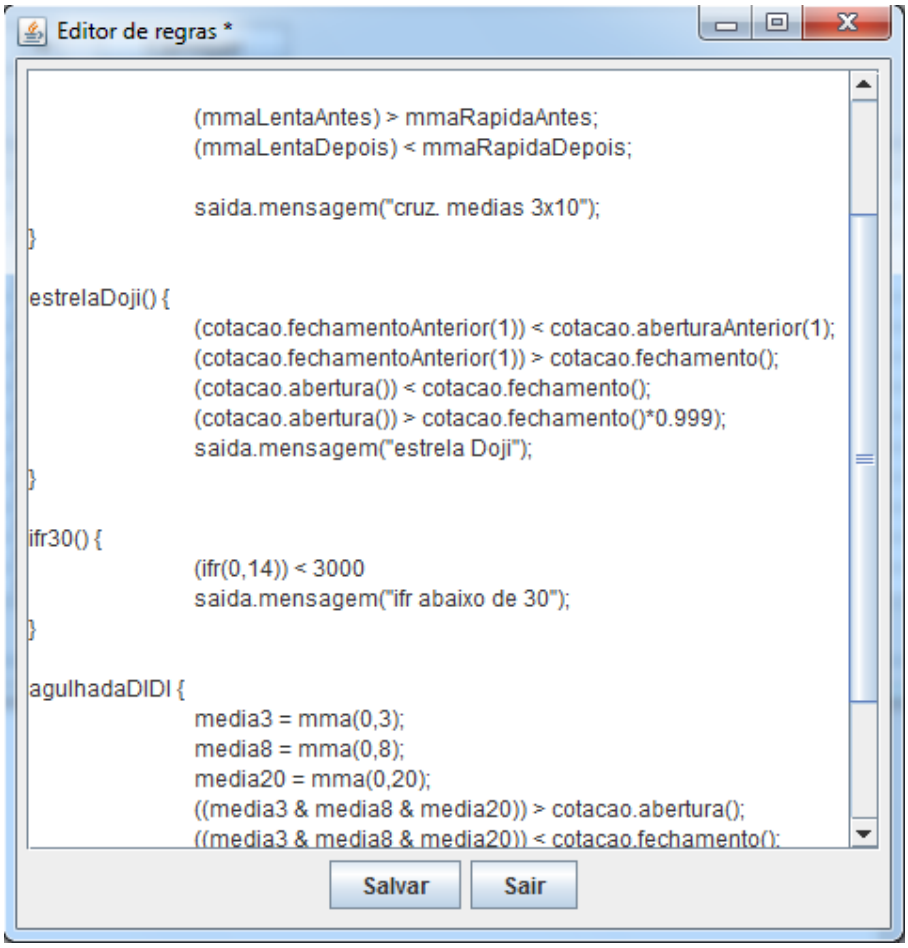
Fonte: Do autor

Cada regra cria um objeto do tipo “SinalBean”, que representa uma ocorrência de regra. Este objeto possui um método chamado “setMensagem(<mensagem>)” que permite definir uma mensagem de alerta. Ao final da regra, este objeto é inserido em um vetor de objetos “SinalBean”, e é fornecido um método chamado “getSinais()” que retorna este vetor. Assim, quando a classe é executada, ela gera um vetor contendo todas as mensagens geradas pelas regras codificadas, e permite obter este vetor através de um método “get”.

Ainda na classe que contém as regras, pode-se observar a criação de outros 2 objetos no corpo do método “regra1”: um do tipo “Indicadores” e outro “ComandosCotacao”. Os objetos destas classes atendem justamente às funções, abordadas no subtítulo 4.1, “LINGUAGEM PROPOSTA”. Através destes objetos, o código acessa os métodos que retornam dados de cotações e valores calculados pelos indicadores comentados.

Na aba da ação que está sendo monitorada, o usuário tem acesso às regras definidas para a mesma, através do botão “regras”. Ao clicar no mesmo, a tela a seguir é apresentada:

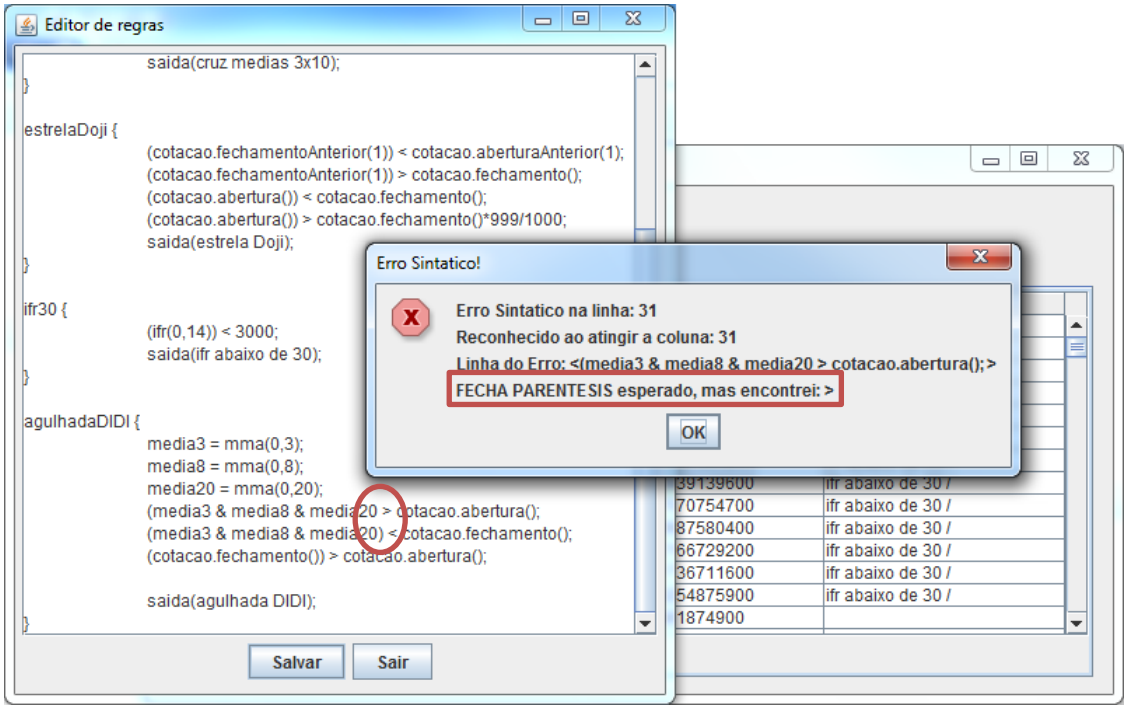
Figura 20 – Editor de estratégias



Fonte: Do autor

Na tela de edição de regras, o usuário acessa as regras do papel, carregadas a partir do arquivo “*.txt” apresentado no exemplo da figura 18, podendo editá-las, criar novas, ou exclui-las. Ao salvar, o editor chama o compilador, que recebe o código do usuário e gera um código Java correspondente, resultando no arquivo “*.java”. Caso seja identificado algum erro no código gerado pelo usuário, o programa emite uma mensagem de erro, informando onde ele ocorreu e o que era esperado naquele trecho. Um exemplo é apresentado na figura 21:

Figura 21 – Mensagem de erro durante compilação



Fonte: Do autor

Conforme destacado na imagem, o compilador aguardava um fechamento de parêntesis, mas encontrou o sinal de comparação “maior que”.

Tendo a solução proposta desenvolvida e funcionando, alguns testes foram realizados. O próximo capitulo trás algumas estratégias de negociação programadas na linguagem proposta, e a análise da ocorrência destas na base de dados de cotações.

5 RESULTADOS OBTIDOS

Neste capítulo serão apresentadas algumas regras de negociação desenvolvidas utilizando-se a linguagem proposta, bem como, o retorno das mesmas ao *software* de análise e apresentação ao usuário.

No momento em que uma nova cotação é adicionada ao histórico da ação, as regras definidas para esta são chamadas, verificando-se quais são atendidas. O retorno é dado conforme explicado no subcapítulo “4.2 SOFTWARE DE ANÁLISE”, com a apresentação das mensagens retornadas pelas regras na tabela de cotações, na mesma linha da data base de análise.

5.1 CRUZAMENTO DE ALTA

A regra de “Cruzamento de Alta” ocorre quando uma média móvel rápida cruza de baixo para cima uma média móvel mais lenta. Para identificar este cruzamento, precisa-se encontrar dois momentos: o primeiro, em que a rápida está abaixo da lenta, e um segundo momento, em que a rápida está acima da lenta.

A seguir, no quadro 26, é apresentada uma regra capaz de identificar o cruzamento entre uma média de 3 períodos e outra de 10 períodos.

Quadro 26 – Codificação da estratégia “Cruzamento de Alta 3x10”

```
cruzamentoAlta3x10 {
      mmaLentaAntes = mma(8,10);
      mmaLentaDepois = mma(0,10);

      mmaRapidaAntes = mma(8,3);
      mmaRapidaDepois = mma(0,3);

      (mmaLentaAntes) > mmaRapidaAntes
      (mmaLentaDepois) < mmaRapidaDepois

      saida(cruz. medias 3x10);
}
```

Fonte: Do autor

Já no quadro 27, é apresentado o código Java resultante da compilação da estratégia programada.

Quadro 27 – Código Java resultante da compilação da estratégia "Cruzamento de Alta 3x10"

```
private void cruzamentoAlta3x10() {
      Indicadores i = new Indicadores(cotacoes);
      ComandosCotacao c = new ComandosCotacao(cotacoes);
      SinalBean s = new SinalBean();

      int mmaLentaAntes = i.mma(8,10);
      int mmaRapidaAntes = i.mma(8,3);
      int mmaLentaDepois = i.mma(0,10);
      int mmaRapidaDepois = i.mma(0,3);
      if (mmaLentaAntes > mmaRapidaAntes &&
            mmaLentaDepois < mmaRapidaDepois ){
                  s.setMensagem("cruz medias 3x10");
      }
      sinais.add(s);
}
```

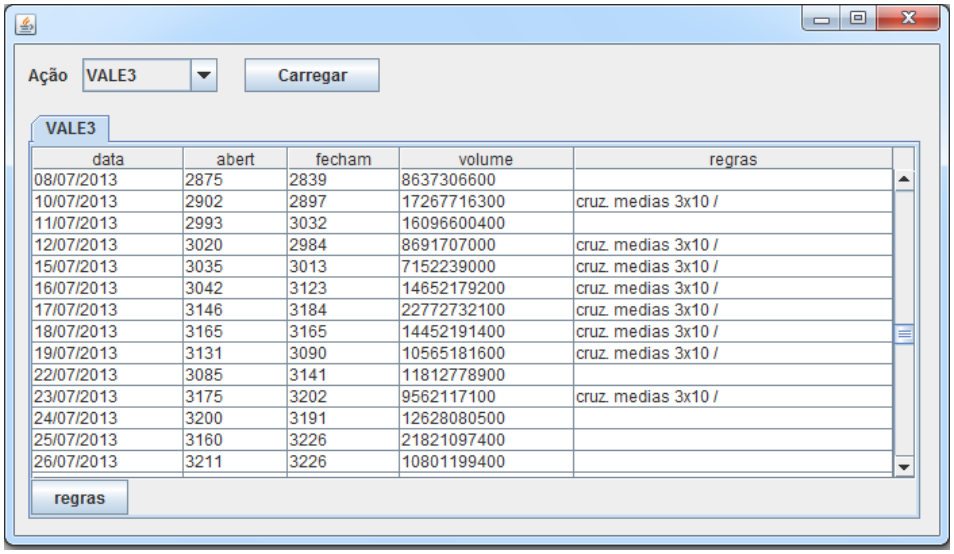
Fonte: Do autor

Inicialmente são gerados 2 pares de variáveis. O primeiro deles armazena os valores obtidos através do cálculo da média móvel aritmética (função "mma()") de 10 e 3 períodos para 8 períodos anteriores ("mmaLentaAntes" e "mmaRapidaAntes"). O segundo par armazena os valores obtidos pelo mesmo cálculo, mas para o período atual. Em seguida, verifica-se:

- se a 8 períodos atrás a média lenta estava acima da média rápida;
- se atualmente a média lenta está abaixo da média rápida.

Caso ocorra a condição, significa que as médias se cruzaram dentro do intervalo de 8 períodos anteriores ao período atual. A seguir, na figura 22, é apresentado um exemplo de ocorrência desta identificado pelo *software*:

Figura 22 – Identificação de ocorrência da estratégia Cruzamento de Alta 3x10



data	abert	fecham	volume	regras
08/07/2013	2875	2839	8637306600	
10/07/2013	2902	2897	17267716300	cruz. medias 3x10 /
11/07/2013	2993	3032	16096600400	
12/07/2013	3020	2984	8691707000	cruz. medias 3x10 /
15/07/2013	3035	3013	7152239000	cruz. medias 3x10 /
16/07/2013	3042	3123	14652179200	cruz. medias 3x10 /
17/07/2013	3146	3184	22772732100	cruz. medias 3x10 /
18/07/2013	3165	3165	14452191400	cruz. medias 3x10 /
19/07/2013	3131	3090	10565181600	cruz. medias 3x10 /
22/07/2013	3085	3141	11812778900	
23/07/2013	3175	3202	9562117100	cruz. medias 3x10 /
24/07/2013	3200	3191	12628080500	
25/07/2013	3160	3226	21821097400	
26/07/2013	3211	3226	10801199400	

Fonte: Do autor

A partir do dia 10/07 ocorre a identificação do cruzamento das médias configuradas. Por identificar a ocorrência do cruzamento dentro de um período, a estratégia gera uma saída mais de uma vez além do momento do cruzamento, pois a regra continua sendo verdadeira. A seguir é apresentado o gráfico analisado pelo *software*, onde também estão plotadas as médias de 3 e 10 períodos. Nele é identificada a data em que ocorre o cruzamento das médias, coincidente com a data sinalizada pelo analisador.

Figura 23 – Estratégia "Cruzamento de Alta 3x10" no gráfico de cotações

Fonte: Do autor

5.2 ESTRELA DOJI

Uma “Estrela Doji” ocorre quando o preço de abertura e fechamento ficam muito próximos um do outro, formando uma figura semelhante a uma estrela. Além disso, o preço de fechamento deve ficar abaixo do fechamento da cotação anterior, e esta última deve ser de baixa (fechamento abaixo da abertura).

Quadro 28 - Codificação da estratégia “Estrela Doji”

```
estrelaDoji() {
      (cotacao.fechamentoAnterior(1))< cotacao.aberturaAnterior(1);
      (cotacao.fechamentoAnterior(1))> cotacao.fechamento();
      (cotacao.abertura()) < cotacao.fechamento()
      (cotacao.abertura()) > cotacao.fechamento()*999/1000)
      saida(estrela Doji);
}
```

Fonte: Do autor

A estratégia de negociação não foi identificada nas ações avaliadas, portanto, sua ocorrência não foi sinalizada. No quadro 29, é apresentado o código Java resultante da compilação da estratégia “Estrela Doji”:

Quadro 29 - Código Java resultante da compilação da estratégia “Estrela Doji”

```
private void estrelaDoji() {
      Indicadores i = new Indicadores(cotacoes);
      ComandosCotacao c = new ComandosCotacao(cotacoes);
      SinalBean s = new SinalBean();

      if (c.fechamentoAnterior(1) < c.aberturaAnterior(1) &&
            c.fechamentoAnterior(1) > c.fechamento() &&
            c.abertura() < c.fechamento() &&
            c.abertura() > c.fechamento()*999/1000 ){
                  s.setMensagem("estrela Doji");
      }
      sinais.add(s);
}
```

Fonte: Do autor

5.3 IFR ABAIXO DE 30

Conforme abordado no subtítulo “2.2 INDICE DE FORÇA RELATIVA”, a região de “sobrevenda” indica um momento em que as vendas da ação estão sem força, podendo ocorrer uma reversão nos preços a medida que as compras se acentuam. A regra a seguir gera um aviso indicando que a ação em análise encontra-se “sobrevendida”.

Quadro 30 - Codificação da estratégia “IFR abaixo de 30”

```
ifr30() {
      (ifr(0,14)) < 3000
      saida(ifr abaixo de 30);
}
```

Fonte: Do autor

A seguir é apresentado o código Java resultante da compilação da estratégia “IFR Sobrevendido”:

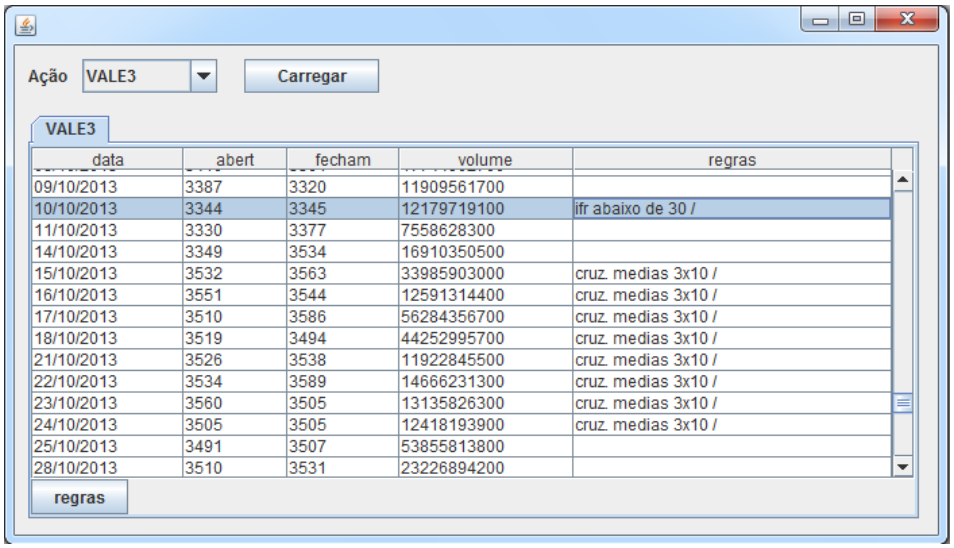
Quadro 31 - Código Java resultante da compilação da estratégia “IFR abaixo de 30”

```
private void ifr30() {
      Indicadores i = new Indicadores(cotacoes);
      ComandosCotacao c = new ComandosCotacao(cotacoes);
      SinalBean s = new SinalBean();

      if (i.ifr(0,14) < 3000 ){
                  s.setMensagem("ifr abaixo de 30");
      }
      sinais.add(s);
}
```

Fonte: Do autor

Através da função “ifr()” calcula-se o atual índice de força relativa da ação, e é verificado se este está abaixo de 30%. Caso esteja, é registrada a mensagem de sinalização da regra.

Figura 24 - Identificação de ocorrência da estratégia “IFR abaixo de 30”



Fonte: Do autor

Figura 25 - Estratégia “IFR abaixo de 30” no gráfico de cotações

Fonte: Do autor

5.4 AGULHADA DIDI

Esta é a estratégia de negociação que busca identificar o momento em que 3 médias móveis se encontram dentro do corpo de um *candle*. As médias devem ser configuradas para 3, 8 e 20 períodos. Em seguida é apresentada a codificação desta estratégia:

Quadro 32 - Codificação da estratégia "Agulhada DIDI"

```
agulhadaDIDI {
      media3 = mma(0,3);
      media8 = mma(0,8);
      media20 = mma(0,20);
      ((media3 & media8 & media20)) > cotacao.abertura();
      ((media3 & media8 & media20)) < cotacao.fechamento();
      ((cotacao.fechamento()) > cotacao.abertura();

      saida(agulhada DIDI);
}
```

Fonte: Do autor

A seguir é apresentado o código Java resultante da compilação da estratégia "Agulhada DIDI":

Quadro 33 - Código Java resultante da compilação da estratégia "Agulhada DIDI"

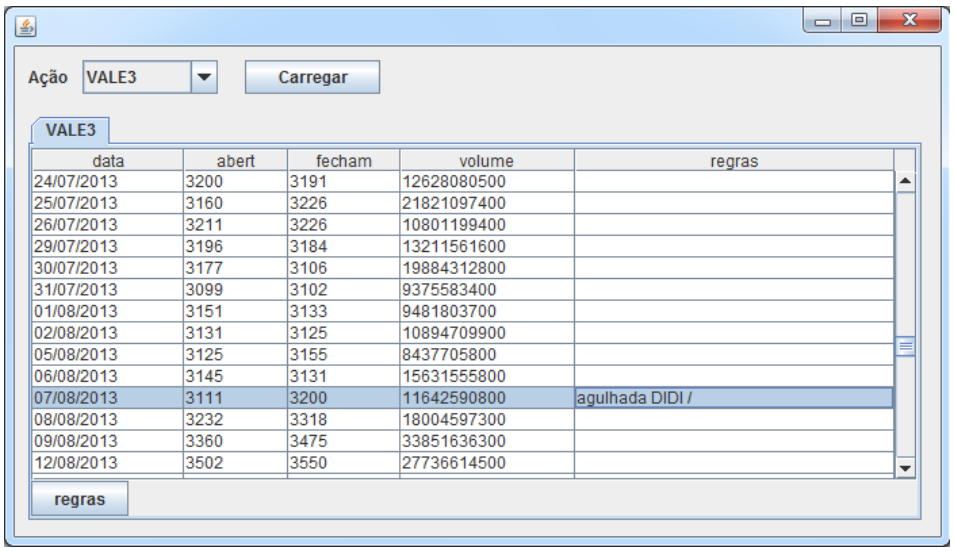
```
private void agulhadaDIDI() {
      Indicadores i = new Indicadores(cotacoes);
      ComandosCotacao c = new ComandosCotacao(cotacoes);
      SinalBean s = new SinalBean();

      int media3 = i.mma(0,3);
      int media8 = i.mma(0,8);
      int media20 = i.mma(0,20);
      if (media20 > c.abertura() &&
            media8 > c.abertura() &&
            media3 > c.abertura() &&
            media20 < c.fechamento() &&
```

```
            media8 < c.fechamento() &&
            media3 < c.fechamento() &&
            c.fechamento() > c.abertura() ){
                  s.setMensagem("agulhada DIDI");
            }
      sinais.add(s);
}
```

Fonte: Do autor

No início do código, são definidas 3 variáveis, cada uma delas armazenando o valor da média móvel aritmética para um período diferente, sempre referentes à última cotação disponível. Em seguida, são definidas as duas condições da agulhada, que buscam identificar se os valores das médias estão entre a abertura e fechamento também da última cotação. Uma terceira condição adicional é inserida, apenas para confirmar se após a agulhada ocorre uma alta dos preços. Em seguida é apresentada a identificação desta regra, bem como a identificação de sua ocorrência no gráfico de cotações:

Figura 26 - Identificação de ocorrência da estratégia "Agulhada DIDI"



Fonte: Do autor

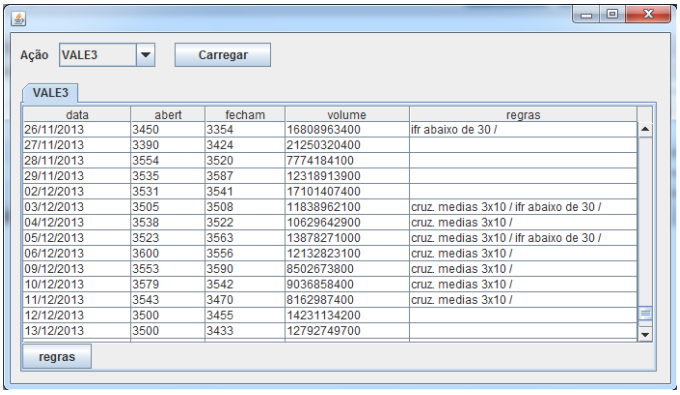
Figura 27 - Estratégia “Agulhada DIDI” no gráfico de cotações

Fonte: Do autor

5.5 MULTIPLAS REGRAS

Em um mesmo momento, pode ser identificada a ocorrência de mais de uma das estratégias programadas. O exemplo a seguir serve para demonstrar a identificação destas e sua apresentação ao usuário. No caso, tem-se a regra “Cruzamento de Alta” e “IFR Sobrevendido” ocorrendo ao mesmo tempo em 2 momentos. Como saída, as mensagens repassadas por ambas as regras são apresentadas de forma concatenada, conforme explicado no capítulo “4.2 SOFTWARE DE ANÁLISE”.

Figura 28 - Identificação de ocorrência de mais de uma estratégia



Fonte: Do autor

Com os exemplos apresentados no decorrer do capítulo, é possível fazer um avaliação a respeito das possibilidades do projeto desenvolvido, bem como verificar se o obtido condiz com o que foi inicialmente proposto e planejado. Esta avaliação é feita no Capítulo 6, onde são expostas as conclusões a respeito do trabalho.

CONCLUSÃO

Ao final do presente trabalho, pode-se concluir que o objetivo do trabalho foi atingido. A linguagem proposta pelo autor permite o desenvolvimento de estratégias para a negociação de ações, sem a necessidade de declaração de tipos de variáveis, manipulação de vetores, laços, ou outros componentes próprios de linguagens de programação de propósito geral, e que poderiam vir a dificultar o trabalho de expressão destas estratégias. Para demonstrar isto, foram expressas algumas regras de negociação, onde o código era composto apenas por termos utilizados na expressão destas regras, sem nenhuma necessidade de código extra, seja na declaração das variáveis, da regra, ou mesmo da saída. Também foi desenvolvido o compilador para a linguagem, e uma ferramenta capaz de identificar a ocorrência das estratégias desenvolvidas em uma base de dados, repassando a ocorrência destas ao usuário por meio da interface.

Apesar de o trabalho ter sido desenvolvido utilizando-se uma base de dados estática, nada impede que o sistema trabalhe com uma base de dados dinâmica, podendo receber cotações em tempo real, interpretando-as conforme a estratégia desenvolvida, e devolvendo ao usuário a ocorrência destas também em tempo real. Quanto maior o número de ações monitoradas e mais diversificadas as estratégias utilizadas, mais proeminente se torna a solução, uma vez que ela pode exceder com larga vantagem a capacidade de análise manual de um operador, limitando-se apenas pela potência da máquina onde ele está sendo executado.

Foram apresentadas algumas das principais técnicas utilizadas atualmente pela análise técnica, descrevendo seu funcionamento, apresentando exemplos, e demonstrando os casos em que estas geram informações úteis ao operador. Com base nas técnicas apresentadas, foram desenvolvidas estratégias de negociação utilizando-se a linguagem proposta, explicando seu funcionamento e demonstrando suas ocorrências na base de dados utilizada.

Além da proposta de linguagem, o trabalho possui outros dois entregáveis: o *software* de análise e o compilador. O primeiro demonstrou ser capaz de identificar a ocorrência das regras programadas pelo usuário na base de dados utilizada, e apresentar ao usuário suas ocorrências. Já o compilador, desenvolvido a partir das técnicas de compilação estudadas e apresentadas no trabalho, cumpriu o objetivo

de traduzir o código do usuário para uma linguagem comercial, no caso, a linguagem Java.

Sendo assim, pode-se considerar que os objetivos definidos para o trabalho foram todos alcançados.

O trabalho foi desenvolvido com a meta de atender aos objetivos definidos, porém, existe uma gama muito grande de possibilidades futuras. Algumas são listadas a seguir:

- Apresentação dos resultados em gráfico: ao invés de apresentar as cotações e as ocorrências das regras em uma tabela, existe a possibilidade de desenvolvimento de uma interface que gere um gráfico com as cotações, e aponte no próprio gráfico a ocorrência das estratégias;
- Desenvolvimento de novos indicadores: disponibilizar ao usuário a possibilidade de desenvolvimento de indicadores personalizados, não ficando dependendo dos indicadores fornecidos pelo *software*;
- Permitir outros tipos de saída: ao invés de retornar apenas um texto, uma estratégia de negociação poderia retornar imagens, alertas sonoros, etc. Esta melhoria provavelmente dependeria de uma interface que plotasse as cotações na forma de gráfico.
- Conexão para cotações em tempo real: incluir no software um conector que permita obter cotações em tempo real, junto a BM&FBOVESPA, ou outra empresa autorizada a fazer a redistribuição do sinal.

Cabe colocar também o grande valor de aprendizado para o autor, que teve a oportunidade com o mesmo de aprofundar seus conhecimentos em conteúdos abordados durante todo o curso, pondo em prática diversos assuntos vistos em aula.

REFERÊNCIAS BIBLIOGRÁFICAS

AHO, Alfred V.; SETHI, Ravi.; LAM, Monica S.; ULLMAN, Jeffrey D. **Compiladores princípios, técnicas e ferramentas**. 2. ed. São Paulo: Pearson Addison Wesley, 2008. Disponível em : <http://feevale.bv3.digitalpages.com.br/users/publications/9788588639249/pages/107> Acessado em: 27 out 2013.

BEIGHLEY, Lynn. **Use a cabeça:** SQL. Rio de Janeiro, RJ: Alta Books, 2010. xxxiv, 454 p.

BM&FBOVESPA. **Cotações históricas.** Disponível em: < http://www.bmfbovespa.com.br/shared/iframe.aspx?idioma=pt-br&url=http://www.bmfbovespa.com.br/pt-br/cotacoes-historicas/FormSeriesHistoricas.asp >. Acessado em: 03 nov 2013.

BM&FBOVESPA. **Cotações históricas.** Disponível em: <http://www.bmfbovespa.com.br/cias-listadas/titulos-negociaveis/BuscaTitulosNegociaveis.aspx?Idioma=pt-br>. Acessado em: 03 nov 2013.

BRUM, Carlos A. H. **Investindo em ações com estratégia e disciplina**. Rio de Janeiro, RJ: Ciência Moderna, 2008. xxx, 241 p.

CAVALCANTE, Francisco; MISUMI, Jorge Yoshio; RUDGE, Luiz Fernando. **Mercado de capitais:** o que é, como funciona. 6. ed., rev. e atual. Rio de Janeiro, RJ: Campus, 2005. xxxiv, 371 p.

DEITEL, Paul J.; DEITEL, Harvey M.,. **Java:** como programar. 8. ed. São Paulo, SP: Pearson, 2010. 1114 p.

DELAMARO, Márcio. **Como construir um compilador utilizando ferramentas Java**. São Paulo, SP: Novatec, c2004. xi, 308 p.

ELDER, Alexander. **Como se transformar em um operador e investidor de sucesso:** entenda a psicologia do mercado financeiro : técnicas poderosas de negociação. 10. ed. Rio de Janeiro, RJ: Campus, 2004. xi, 305 p.

ELDER, Alexander. **Aprenda a operar no mercado de ações**: um guia completo para o trading. 4. ed. Rio de Janeiro, RJ: Campus, 2006.

GRUNE, Dick; BAL, Henri E.; JACOBS, Ceriel J.H.; LANGENDOEN, Koen G. **Projeto moderno de compiladores:** implementação e aplicações. Rio de Janeiro: Campus, 2001. 671 p.

HOFFMEISTER, Bruno Vier. **Eficiencia da análise gráfica no mercado de ações brasileiro.** 2013. 117 f. Monografia (Conclusão do Curso de Ciência da Computação) - Feevale, Novo Hamburgo-RS, 2013 Disponível em : <http://biblioteca.feevale.br/Monografia/MonografiaBrunoHoffmeister.pdf>. Acesso em : 26 nov. 2013.

HOPCROFT, John E.; ULLMAN, Jeffrey D.; MOTWANI, Rajeev. **Introdução à teoria dos autômatos, linguagens e computação.** Rio de Janeiro, RJ: Campus, 2002, c2003. Não paginado

KONCHINSKI, Vinicius. **Bovespa já é a 2ª maior do mundo em valor de mercado**. Exame, Disponível em: <http://exame.abril.com.br>. Acessado em: 3 set 2013.

LOUDEN, Kenneth C**. Compiladores:** princípios e práticas. São Paulo, SP: Cengage Learning, 2004. xiv, 569 p.

MATSURA, Eduardo. **Comprar ou vender?:** como investir na bolsa utilizando análise gráfica. 3. ed. São Paulo, SP: Saraiva, 2006. 124 p.

NILSON, Steve. **Japanese Candlestick Charting Techniques**. 2. ed. New York: New York Institute of Finance, 2001. 299p.

PRICE, Ana Maria de Alencar; TOSCANI, Simão Sirineo**. Implementação de linguagens de programação:** compiladores. 2. ed. Porto Alegre, RS: Sagra Luzzatto, Instituto de Informática da UFRGS, 2001. Não paginado

PRINCENTON. **Backus-Naur Form.** Disponível em: <http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Backus%E2%80%93Naur_Form.html>. Acessado em: 02 nov 2013.

RICARTE, Ivan. **Introdução à compilação**. Rio de Janeiro, RJ: Campus, 2008. xi, 264 p.

RICARTE, Ivan L. M. **Programação de Sistemas:** Uma Introdução. Disponível em: <http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/progsist2003.html>. Acessado em: 27 out 2013.

APÊNDICES

APÊNDICE A – LISTA DE FUNÇÕES SOBRE DADOS DE COTAÇÕES

cotacao.abertura()

Parâmetros:

-

Retorno:

Preço de abertura da última cotação disponível

cotacao.aberturaAnterior(N)

Parâmetros:

N: número de dias (N = 1 -> 1 dia atrás)

Retorno:

Preço de abertura de N dias atrás

cotacao.mediaAberturaUltimasAltas(N)

Parâmetros:

N: número de cotações em que ocorreram altas

Retorno:

Média aritmética da abertura das ultimas N cotações em que ocorreram altas

cotacao.mediaAberturaUltimasBaixas(N)

Parâmetros:

N: número de cotações em que ocorreram baixas

Retorno:

Média aritmética da abertura das ultimas N cotações em que ocorreram baixas

cotacao.fechamento()

Parâmetros:

-

Retorno:

Preço de fechamento da última cotação disponível

cotacao.fechamentoAnterior(N)

Parâmetros:

N: número de dias (N = 1 -> 1 dia atrás)

Retorno:

Preço de abertura de N dias atrás

cotacao.mediaFechamentoUltimasAltas(N)

Parâmetros:

N: número de cotações em que ocorreram altas

Retorno:

Média aritmética do fechamento das ultimas N cotações em que ocorreram altas

cotacao.mediaFechamentoUltimasBaixas(N)

Parâmetros:

N: número de cotações em que ocorreram baixas

Retorno:

Média aritmética do fechamento das ultimas N cotações em que ocorreram baixas

cotacao.maximo()

Parâmetros:

-

Retorno:

Preço máximo da última cotação disponível

cotacao.maximoAnterior(N)

Parâmetros:

N: número de dias (N = 1 -> 1 dia atrás)

Retorno:

Preço máximo de N dias atrás

cotacao.maximoPeriodo(N)

Parâmetros:

N: número de cotações do período de análise

Retorno:

Preço máximo das ultimas N cotações

cotacao.minimo()

Parâmetros:

-

Retorno:

Preço mínimo da última cotação disponível

cotacao.minimoAnterior(N)

Parâmetros:

N: número de dias (N = 1 -> 1 dia atrás)

Retorno:

Preço mínimo de N dias atrás

cotacao.minimoPeriodo(N)

Parâmetros:

N: número de cotações do período de análise

Retorno:

Preço mínimo das ultimas N cotações

cotacao.volume()

Parâmetros:

-

Retorno:

Volume de negócios da última cotação disponível

cotacao.volumeAnterior(N)

Parâmetros:

N: número de dias (N = 1 -> 1 dia atrás)

Retorno:

Volume de negócios de N dias atrás

APÊNDICE B – LISTA DE INDICADORES

mma(periodo, numPeriodos)

Parâmetros:

periodo: número de dias (N = 1 -> 1 dia atrás)

numPeriodos: quantidade de períodos a serem utilizados para calcular a média

Retorno:

A média aritmética do período informado, calculada com base no número de períodos informados.

IFR(periodo, numPeriodos)

Parâmetros:

periodo: número de dias (N = 1 -> 1 dia atrás)

numPeriodos: quantidade de períodos a serem utilizados para calcular o índice de força relativa

Retorno:

O percentual (variando de 0 a 100) do índice de força relativa para o período informado, utilizando o número de períodos informados.

APÊNDICE C – TRECHO DE CÓDIGO NA LINGUAGEM MQL5

Figura 29 – Trecho de código em MQL5

```
void myInit(const int bars,
            const double &Open[],
            const double &High[],
            const double &Low[],
            const double &Close[])
  {
//---
   int index,index1;
   int bars1=bars-1;
   fPoint_i=0;
   h=High[bars1];
   l=Low[bars1];
   for(index=bars-2; index>=0; index--)
     {
      index1=index+1;
      if(High[index]>High[index1] || Low[index]<Low[index1])
        {
         if(High[index]>h && High[index]>High[index1]) s_up++;
         if(Low[index]<l && Low[index]<Low[index1]) s_dn++;
        }
      else continue;
      if(s_up==s_dn && s_up==GSv_range)
        {
         h=High[index];
         l=Low[index];
         sPoint_i=bars1-index;
         if(Close[index]>=Open[index])
           {
            s_dn=0;
            IndBuffer[bars1]=Low[bars1];
            IndBuffer[index]=High[index];
            draw_up=1;
            break;
           }
         else
           {
            s_up=0;
            IndBuffer[bars1]=High[bars1];
            IndBuffer[index]=Low[index];
            draw_dn=1;
            break;
```

Fonte: http://www.mql5.com/