



Universidade de Brasília - UnB  
Faculdade UnB Gama - FGA  
Engenharia de Software

# **Software Multiparadigma com Utilização de Métodos Matemáticos para Automação de Estratégias Financeiras no Mercado de Moedas**

Autor: Cleiton da Silva Gomes, Vanessa Barbosa Martins  
Orientador: Dr. Ricardo Matos Chaim

Brasília, DF  
2015





Cleiton da Silva Gomes, Vanessa Barbosa Martins

# **Software Multiparadigma com Utilização de Métodos Matemáticos para Automação de Estratégias Financeiras no Mercado de Moedas**

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Ricardo Matos Chaim

Coorientador: Dr<sup>a</sup>. Milene Serrano

Brasília, DF

2015

---

Cleiton da Silva Gomes, Vanessa Barbosa Martins

Software Multiparadigma com Utilização de Métodos Matemáticos para Automação de Estratégias Financeiras no Mercado de Moedas/ Cleiton da Silva Gomes, Vanessa Barbosa Martins. – Brasília, DF, 2015-

88 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Ricardo Matos Chaim

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB  
Faculdade UnB Gama - FGA , 2015.

1. Mercado de Moedas. 2. Paradigmas de Programação. I. Dr. Ricardo Matos Chaim. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Software Multiparadigma com Utilização de Métodos Matemáticos para Automação de Estratégias Financeiras no Mercado de Moedas

CDU 02:141:005.6

---

Cleiton da Silva Gomes, Vanessa Barbosa Martins

# **Software Multiparadigma com Utilização de Métodos Matemáticos para Automação de Estratégias Financeiras no Mercado de Moedas**

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho aprovado. Brasília, DF, 18 de novembro de 2014:

---

**Dr. Ricardo Matos Chaim**  
Orientador

---

**Dr<sup>a</sup>. Milene Serrano**  
Coorientador

---

**Msc. Hilmer Rodrigues Neri**  
Convidado 1

---

**Msc. Fabiana Freitas Mendes**  
Convidado 2

Brasília, DF  
2015



*Dedicamos este trabalho aos que fazem ao invés de apenas reclamar.*





# Agradecimentos

Eu, Cleiton Gomes, agradeço ao meu pai que estudou até a 7ª série do ensino fundamental, mas sempre incentivou seus filhos a estudarem mesmo diante de tantas dificuldades e hoje seus três filhos estudam na UnB. Agradeço também aos meus irmãos por toda a experiência de vida que obtivemos juntos em várias aventuras de infância/adolescência. Por fim, agradeço também a Vanessa Barbosa por todo esforço e dedicação que teve ao longo deste trabalho.

Eu, Vanessa Barbosa, agradeço à todos de que alguma maneira torceram por mim ao longo destes 5 anos. Agradeço também aos meus pais e ao Cleiton Gomes que ficaram ao meu lado nos momentos mais críticos. Por fim agradeço aquele que meu deu força para chegar até aqui, Deus.

Agradecemos aos nossos orientadores Ricardo Chaim e Milene Serrano por todas as diversas reuniões construtivas, por todas as dicas maravilhosas, pelas revisões de alto nível, pelo carinho e por todo o capricho que tiveram em nos orientar. Agradecemos também aos professores Maurício Serrano, Fabiana Freitas e Wander Cleber por todas as observações pertinentes e construtivas que fizeram ao longo do trabalho. Qualquer possível erro de semântica ou sintaxe deste trabalho são de nossa inteira responsabilidade, mas foi uma honra ter cinco professores doutores de alto nível auxiliando no nosso trabalho.



*"Sei que o meu trabalho é uma gota no oceano, mas sem ele,  
o oceano seria menor."  
(Madre Teresa de Calcutá)*



# Resumo

Este trabalho tem como objetivo desenvolver um software multiparadigma para operar de forma semiautomatizada no Mercado de Moedas: InvestMVC. Adotou-se uma metodologia de pesquisa com base nos objetivos e nos procedimentos técnicos. Para execução da pesquisa, realizou-se uma adaptação do Scrum, levando em consideração as atividades alocadas no cronograma e o contexto de Engenharia de Software aliado com o Mercado de Moedas. Criou-se um protocolo de experimentação para selecionar os métodos matemáticos que serão implementados no software InvestMVC. Definiu-se a arquitetura orientada a componentes e cada componente foi detalhado durante o trabalho. Foi reportado os resultados parciais do desenvolvimento da ferramenta InvestMVC, levando em consideração a qualidade de software com testes dinâmicos e estáticos. Por fim, foram reportadas as conclusões do trabalho.

**Palavras-chaves:** Mercado de Moedas, Métodos Matemáticos, Paradigmas de Programação, Qualidade de Software.



# Abstract

This work aims to develop a multi-paradigm software to operate semi-automated manner FOREX: InvestMVC. It was adopted a research methodology based on objectives and technical procedures. In the implementation of the research, there was a Scrum adaptation, taking into account the allocated activities on schedule and the context of Software Engineering allied with the FOREX. In this way, it has created a Protocol of Experimentation to select the Mathematical Methods that will be implemented in the software InvestMVC. Being oriented architecture defined components and each component detailed during the work. Thus, it was reported the partial results of the development of InvestMVC tool, taking into account the Quality of Software with dynamic and static tests.

**Key-words:** FOREX, Mathematical Methods, Programming Paradigms, Software Quality.





# Lista de ilustrações

Figura 1 – Reta de Suporte. . . . .	7
Figura 2 – Reta de Resistência. . . . .	8
Figura 3 – Equação da reta Mínimos Quadrados. . . . .	10
Figura 4 – Classificação da Correlação Linear. . . . .	11
Figura 5 – Determinação da Correlação Linear. . . . .	11
Figura 6 – Fórmula de Fibonacci sem uso de recorrência. . . . .	13
Figura 7 – Indicador Estocástico. . . . .	14
Figura 8 – Equação de Média Móvel. . . . .	15
Figura 9 – Indicador Média Móvel . . . . .	15
Figura 10 – Arquitetura de von Neumann. . . . .	17
Figura 11 – A essência do Paradigma Orientado a Objetos. . . . .	19
Figura 12 – Defeito x Erro x Falha. . . . .	25
Figura 13 – Níveis de teste e seu desenvolvimento. . . . .	27
Figura 14 – Intervalo para interpretação das métricas. . . . .	30
Figura 15 – Diagrama de Sequência InvestMVC . . . . .	33
Figura 16 – Diagrama de Classe InvestMVC componente Orientado a Objetos . . . .	34
Figura 17 – Diagrama de sequência InvestMVC componente Orientado a Objetos .	35
Figura 18 – Componente Funcional InvestMVC . . . . .	36
Figura 19 – Diagrama de Sequência do Componente Funcional InvestMVC . . . . .	36
Figura 20 – Diagramas de Componentes e de Sequência do Componente Estruturado InvestMVC. . . . .	37
Figura 21 – Diagrama de Classe do Componente Multiagente InvestMVC . . . . .	38
Figura 22 – Diagrama de Sequência do Componente Lógico InvestMVC. . . . .	39
Figura 23 – Diagrama de Sequência do Componente MQL InvestMVC. . . . .	40
Figura 24 – Diagrama de Sequência InvestMVC . . . . .	41
Figura 25 – Resultado da Suíte de Teste do Método Correlação Linear . . . . .	44
Figura 26 – Resultado da Suíte de Teste do Método de Fibonacci . . . . .	44
Figura 27 – Resultado da Suíte de Teste do Método Mínimos Quadrados . . . . .	44
Figura 28 – Suíte de teste da base aprendiz.pl . . . . .	45
Figura 29 – Suíte de teste da base de conhecimento . . . . .	45
Figura 30 – Cobertura de Código dos pacotes do Componente Multiagentes . . . . .	45
Figura 31 – Suite de teste do Componente Multiagentes . . . . .	46



# Lista de tabelas

Tabela 1 – Estado atual da ferramenta InvestMVC . . . . .	43
---	----



# Lista de abreviaturas e siglas

ACC	Acoplamento
ACCM	Complexidade Ciclomática
ANPM	Número de Parâmetros por Método
CPU	Unidade Central de Processamento
DIT	Herança
IEEE	Institute of Electrical and Electronics Engineers
MVC	Model-View-Controller
NPA	Encapsulamento
OO	Orientado a objetos
SC	Coesão e Acoplamento
SMA	Sistema Multiagente
US	User Story



# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>1</b>
<b>1.1</b>	<b>Objetivos</b>	<b>2</b>
<b>1.2</b>	<b>Organização do Trabalho</b>	<b>2</b>
<b>2</b>	<b>REFERENCIAL TEÓRICO</b>	<b>5</b>
<b>2.1</b>	<b>Contexto Financeiro</b>	<b>5</b>
2.1.1	Mercado de Moedas	5
2.1.2	Alavancagem	6
2.1.3	Suporte	6
2.1.4	Resistência	7
<b>2.2</b>	<b>Métodos Matemáticos</b>	<b>8</b>
2.2.1	Método de Mínimos Quadrados	8
2.2.2	Método de Correlação Linear	10
2.2.3	Método de Fibonacci	12
2.2.4	Estocástico	13
2.2.5	Média Móvel	14
<b>2.3</b>	<b>Paradigmas de Programação</b>	<b>15</b>
2.3.1	Paradigma Procedural	16
2.3.2	Paradigma Orientado a Objetos	18
2.3.3	Paradigma Orientado a Agentes	21
2.3.4	Programação Declarativa	23
2.3.4.1	Paradigma Funcional	23
2.3.4.2	Paradigma Lógico	24
<b>2.4</b>	<b>Teste de Software</b>	<b>25</b>
2.4.1	Testes Unitários	27
2.4.2	Teste de integração	28
2.4.3	Teste de Aceitação	28
<b>2.5</b>	<b>Qualidade de Software</b>	<b>28</b>
2.5.1	Métricas de Qualidade de Código Fonte	29
2.5.2	Análise Estática de Código Fonte	29
2.5.3	Ferramentas de Análise Estática	30
<b>3</b>	<b>METODOLOGIA DE PESQUISA</b>	<b>31</b>
<b>4</b>	<b>O INVESTMVC</b>	<b>33</b>
<b>4.1</b>	<b>Arquitetura Orientada a Componentes</b>	<b>33</b>

4.1.1	Componente Orientado a Objetos . . . . .	33
4.1.2	Componente Cálculos Numéricos . . . . .	35
4.1.2.1	Componente Funcional . . . . .	35
4.1.2.2	Componente Estruturado . . . . .	36
4.1.3	Componente Multiagentes . . . . .	38
4.1.4	Componente Lógico . . . . .	38
4.1.5	Componente MQL . . . . .	39
4.1.6	Fluxo de atividades da ferramenta InvestMVC . . . . .	40
<b>5</b>	<b>RESULTADOS . . . . .</b>	<b>43</b>
<b>5.1</b>	<b>Resultados testes unitários . . . . .</b>	<b>44</b>
5.1.1	Componente Funcional . . . . .	44
5.1.2	Componente Estruturado . . . . .	44
5.1.3	Componente Lógico . . . . .	44
5.1.4	Componente Multiagentes . . . . .	45
<b>6</b>	<b>CONCLUSÃO . . . . .</b>	<b>47</b>
	<b>Referências . . . . .</b>	<b>49</b>
<b>7</b>	<b>GLOSSÁRIO . . . . .</b>	<b>55</b>
	<b>APÊNDICES . . . . .</b>	<b>57</b>
	<b>APÊNDICE A – SUPORTE TECNOLÓGICO . . . . .</b>	<b>59</b>
	<b>APÊNDICE B – HISTÓRIAS DE USUÁRIO . . . . .</b>	<b>61</b>
	<b>APÊNDICE C – CRONOGRAMA INVESTMVC . . . . .</b>	<b>63</b>
	<b>APÊNDICE D – EXPERTS PARA EXPERIMENTO . . . . .</b>	<b>65</b>
	<b>APÊNDICE E – COMPONENTE OO . . . . .</b>	<b>67</b>
	<b>APÊNDICE F – COMPONENTE FUNCIONAL . . . . .</b>	<b>71</b>
	<b>APÊNDICE G – COMPONENTE ESTRUTURADO . . . . .</b>	<b>75</b>
	<b>APÊNDICE H – COMPONENTE LÓGICO . . . . .</b>	<b>77</b>
	<b>APÊNDICE I – COMPONENTE MULTIAGENTES . . . . .</b>	<b>81</b>
	<b>APÊNDICE J – COMPONENTE MQL . . . . .</b>	<b>83</b>



## **ANEXOS**

**85**

<b>ANEXO A – TEMPLATE DE PROTOCOLO EXPERIMENTAL . .</b>	<b>87</b>
---	-----------



# 1 Introdução

Operar no mercado de moedas de forma manual é muito arriscado e, portanto, não recomendado, uma vez que esse mercado é não previsível, o que pode provocar a perda do capital de um investidor em apenas alguns minutos. Em diversas situações, o mercado varia as cotações em apenas um minuto, sendo que a mesma variação pode ser feita durante horas. Para contornar esse problema, a plataforma MetaTrader<sup>1</sup> oferece as linguagens MQL4 (Paradigma Estruturado) e MQL5 (Paradigma Orientado a Objetos) para construir *Experts* que operem de forma automatizada.

A plataforma MetaTrader não oferece suporte de ferramentas de testes unitários para as linguagens MQL4 e MQL5. Após implementar um *Expert*, não é possível implementar testes unitários para verificar se as instruções programadas estão de acordo com o esperado. A única forma de verificar se o *Expert* está seguindo as estratégias programadas é usar uma conta real ou demo na plataforma e operar durante um período específico de tempo.

Não foi possível encontrar na literatura investigada até o momento ferramentas que realizem a análise estática de código fonte em MQL4 e MQL5. Portanto, torna-se difícil obter uma análise de critérios de aceitabilidade (ou orientada a métricas) no nível de código fonte dos *Experts* programados nessas linguagens.

Adicionalmente, o código da plataforma MetaTrader é fechado. Dessa forma, não é possível a colaboração da comunidade de desenvolvedores no que tange a evolução das funcionalidades do MetaTrader.

Diante das preocupações abordadas anteriormente, acredita-se que o desenvolvimento de um software de código aberto para investimento no Mercado de Moedas, orientado a modelos conceituais de diferentes paradigmas de programação bem como às boas práticas da Engenharia de Software como um todo, irá conferir ao investidor maior segurança e conforto em suas operações. Portanto, o software proposto será implementado em diferentes linguagens de programação, padrões de projeto adequados, testes unitários orientados à uma abordagem multiparadigmas e análise qualitativa de código fonte. Um Expert (implementado em linguagem MQL4 ou MQL5) ou um conjunto de Experts serão substituídos pelo software. Nesse último caso, o software terá a propriedade de controlar e/ou monitorar um ou mais *Experts*.

Este trabalho, portanto, procurará responder a seguinte questão de pesquisa: é possível desenvolver um software multiparadigma que substitua os Experts convencionais do Mercado de Moedas?

---

<sup>1</sup> <<http://www.metatrader4.com/>>

## 1.1 Objetivos

Este trabalho tem como objetivo geral desenvolver o software multiparadigma InvestMVC que utiliza métodos matemáticos para automação de estratégias financeiras no mercado FOREX.

Considerando o mercado de moedas e os paradigmas de programação estruturado, orientado a objetos, funcional, lógico e multiagentes, são objetivos específicos deste trabalho:

1. Selecionar métodos matemáticos a serem implementados no software InvestMVC com utilização de um protocolo de experimentação;
2. Caracterizar as estruturas e componentes do software InvestMVC;
3. Realizar análise estática do código fonte dos produtos de software a partir de métricas de qualidade previamente definidas;
4. Realizar análise estática do código fonte dos produtos de software a partir de métricas de qualidade previamente definidas, quando aplicável;
5. Desenvolver testes de integração e funcionais para a software InvestMVC;
6. Comparar resultados financeiros obtidos pelo software InvestMVC com os *Experts* tradicionais implementados em linguagem MQL.

## 1.2 Organização do Trabalho

No Capítulo 2, é apresentado o referencial teórico quanto ao contexto financeiro, métodos matemáticos, paradigmas de programação, teste e qualidade de Software. No contexto financeiro, são tratados os atributos atrelados ao mercado de moedas como alavancagem, suporte e resistência. Em métodos matemáticos, é realizada uma descrição dos métodos de Fibonacci, Correlação de Pearson, Mínimos Quadrados, Estocástico e Média Móvel. Em paradigmas de programação, são descritos os paradigmas: estruturado, orientado a objetos, lógico, funcional e multiagentes. Em testes de software, são evidenciados quais os tipos de testes serão utilizados neste trabalho e também quais linguagens terão testes. Por fim, em qualidade de software são externalizadas as métricas de qualidade de código fonte, critérios para interpretação das métricas e ferramentas de análise estática.

No capítulo 3, é apresentada a Metodologia de Pesquisa e seus atributos como classificação da pesquisa, atividades da pesquisa, execução da pesquisa, planejamento do protocolo de experimentação e o cronograma.

No capítulo 4, são apresentados os resultados ...

No capítulo 5, é apresentada o software InvestMVC ...

Por fim, no capítulo 6 são apresentadas as considerações finais.



## 2 Referencial teórico

O referencial teórico revela o momento de levantar o embasamento teórico sobre o tema de pesquisa. No contexto deste trabalho, faz-se necessário, dentro de outros aspectos, pesquisar sobre os entendimentos existentes do problema de pesquisa e analisar quais mecanismos devem ser adotados para se propor uma solução ([BELCHIOR, 2012](#)).

O referencial teórico deste trabalho irá levantar o embasamento teórico sobre Contexto Financeiro, Métodos Matemáticos, Paradigmas de Programação e Testes estáticos/dinâmicos para que seja possível propor uma solução para o problema de pesquisa.

### 2.1 Contexto Financeiro

Esta seção irá tratar os atributos aliados ao contexto financeiro como Alavancagem, Suporte e Resistência. Esses atributos são insumos para que se possa compreender melhor a dinâmica das estratégias para negociação no Mercado de Moedas.

#### 2.1.1 Mercado de Moedas

Mercado de Moedas ou FOREX (abreviatura de Foreign Exchange) é um mercado interbancário onde as várias moedas do mundo são negociadas. O FOREX foi criado em 1971, quando a negociação internacional transitou de taxas de câmbio fixas para flutuantes. Com o resultado do seu alto volume de negociações, o Mercado de Moedas tornou-se o principal mercado financeiro do mundo ([MARKET, 2011](#)).

A operação no Mercado de Moedas envolve a compra de uma moeda e a simultânea venda de outra. As moedas são negociadas em pares, por exemplo: euro e dólar (EUR-USD). O investidor não compra ou vende euro e dólares fisicamente, mas existe uma relação monetária de troca entre eles. O FOREX é um mercado em que são negociados, portanto, derivativos de moedas. O investidor é remunerado pelas diferenças entre a valorização (se tiver comprado) ou desvalorização (se tiver vendido) destas moedas ([CVM, 2009](#), pág. 3).

O Mercado de Moedas é descentralizado, pois as operações são realizadas por vários participantes do mercado em vários locais. É raro uma moeda manter uma cotação constante em relação a outra moeda. O câmbio entre duas moedas muda constantemente ([FXCM, 2011](#), pág. 5).

O Mercado de Moedas é constituído por transações entre as corretoras que operam no mesmo e são negociados, diariamente, contratos representando volume total entre 1 e

3 trilhões de dólares. As transações são realizadas diretamente entre as partes (investidor e corretora) por telefone e sistemas eletrônicos, desde que tenham conexão à internet. As operações ocorrem 24 horas por dia, durante 5 dias da semana (abrindo às 18h no domingo e fechando às 18h na sexta; horário de Brasília), negociando os principais pares de moedas, ao redor do mundo (CVM, 2009, pág. 4).

### 2.1.2 Alavancagem

Alavancagem no contexto de mercado, deriva do significado de alavanca na Física, relacionado com a obtenção de um resultado final maior do que ao esforço empregado (DANTAS; MEDEIROS; LUSTOSA, 2006, pág. 3).

O conceito de alavancagem é similar ao conceito de alavanca comumente empregado em física. Por meio da aplicação de uma força pequena no braço maior da alavanca, é possível mover um peso muito maior no braço menor da alavanca (BRUNI; FAMÁ, 2011, pág. 232).

A Alavancagem possui a propriedade de gerar oportunidades financeiras para empresas que possuem indisponibilidade de recursos internos e/ou próprios (ALBUQUERQUE, 2013, p 13).

No mercado FOREX, o investidor pode negociar contratos de taxas de câmbio e usar a Alavancagem para aumentar suas taxas de lucro. Se o investidor, por exemplo, realizar uma operação de compra apostando 0.01 por ponto e o mercado subir 1000 pontos, ele ganha 10 dólares ( $0.001 \times 1000$ ). Usando a técnica de Alavancagem, o investidor pode realizar a mesma operação de compra colocando sua operação alavancada a 1.0, realizando o lucro de 1000 dólares ( $1.0 \times 1000$ ) (EASYFOREX, 2014).

### 2.1.3 Suporte

Segundo MATSURA (2006, pág. 22), Suporte é o nível de preço no qual a pressão compradora supera a vendedora e interrompe o movimento de baixa. Pode-se identificar o Suporte por uma linha reta horizontal conforme a Figura 1.



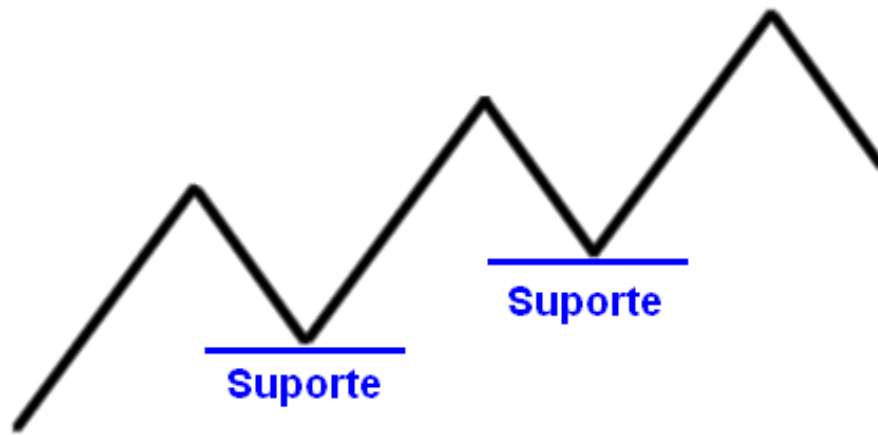


Figura 1 – Reta de Suporte.

Fonte: [MATSURA \(2006, pág. 22\)](#).

Suporte é uma região gráfica que após uma queda, os preços param e reverterem no sentido contrário. É uma área em que os investidores tendem a comprar ([DEBASTINI, 2008, p 97](#)).

Os níveis de Suporte indicam as cotações em que os investidores acreditam que vão subir. À medida em que as cotações se deslocam para a zona de Suporte, os investidores estão mais confiantes para comprar ([COLLINS et al., 2012](#)).

#### 2.1.4 Resistência

Resistência é a região do gráfico em que após um movimento de alta, os preços param e reverterem no sentido contrário. É um ponto em que os investidores tendem a vender para ter o maior lucro possível ([DEBASTINI, 2008, pág. 98](#)).

Segundo [MATSURA \(2006, pág. 23\)](#), Resistência representa o nível de preço no qual a pressão vendedora supera a compradora e interrompe o movimento de alta. A Resistência é identificada por uma linha reta horizontal, conforme a Figura 2.

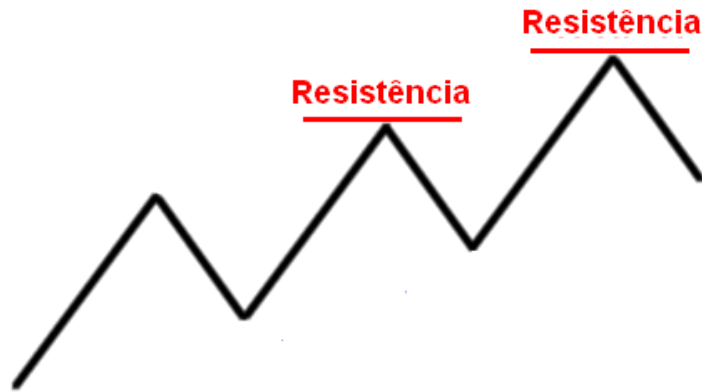


Figura 2 – Reta de Resistência.

Fonte: [MATSURA \(2006, pág. 23\)](#)

A Resistência indica os níveis das cotações em que os investidores acreditam que as mesmas vão descer. À medida que as cotações se deslocam para a zona de Resistência, os investidores estão mais confiantes para vender ([COLLINS et al., 2012](#)).

## 2.2 Métodos Matemáticos

Métodos Matemáticos é qualquer método que se utiliza da matemática para resolver um problema. É possível citar alguns exemplos desses métodos como equações polinomiais, identidades trigonométricas, geometria coordenada, frações parciais, expansões binomiais, entre outros ([RILEY; HOBSON; BENCE, 2011](#)).

Métodos Matemáticos são aplicados a área de finanças. Cálculo e álgebra linear são fundamentais para o estudo de matemática financeira e ajuda a compreender a dinâmica de mercado ([KONIS, 2014](#)).

Este capítulo irá abordar sobre os métodos matemáticos de Mínimos Quadrados, Fibonacci, Correlação Linear de Pearson, Estocástico e Média Móvel.

### 2.2.1 Método de Mínimos Quadrados

O método de Mínimos Quadrados determina o valor mais provável de quantidades não conhecidas em que a soma dos quadrados das diferenças entre valores observados e computados é mínimo ([INÁCIO, 2010, pág. 72](#)).

Usa-se o método de Mínimos Quadrados para determinar a melhor linha de ajuste que passa mais perto de todos os dados coletados, no intuito de obter a melhor linha de

ajuste, de forma que minimize as distâncias entre cada ponto de consumo (DIAS, 1985, pág. 46).

A aplicação do método de Mínimos Quadrados visa deduzir a melhor estimativa de mensurações de  $n$  medições idênticas (em condições de “repetitividade”) e não idênticas (em condições de “reprodutividade”). Dessa forma o peso estatístico de um resultado é definido (VUOLO, 1996, pág. 149).

O desvio vertical do ponto:

$$(x_i, y_i) \quad (2.1)$$

da reta:

$$Y = B0 + B1 * X_i \quad (2.2)$$

é a altura do ponto menos altura da reta. A soma dos desvios quadrados verticais dos pontos:

$$(x_1, y_1) \dots (x_i, y_i) \quad (2.3)$$

à reta é portanto:

$$f(B0, B1) = \sum y_i - (B0 + B1 * X_i)^2 \quad (2.4)$$

$$0 \leq i < \infty \quad (2.5)$$

As estimativas pontuais de  $C0$  e  $C1$ , representadas por  $K0$  e  $K1$  e denominadas estimativa de Mínimos Quadrados, são aquelas que minimizam  $f(B0, B1)$ . Em suma, para qualquer  $B0$  e  $B1$ ,  $K0$  e  $K1$  são tais que:

$$f(K0, K1) \leq f(B0, B1) \quad (2.6)$$

A reta de Regressão Estimativa ou de Mínimos Quadrados é, por conseguinte, a reta cuja equação é :

$$Y = K0 + K1X \quad (2.7)$$

Como mostrado na Figura 3 (DEVORE, 2006, pág. 441).

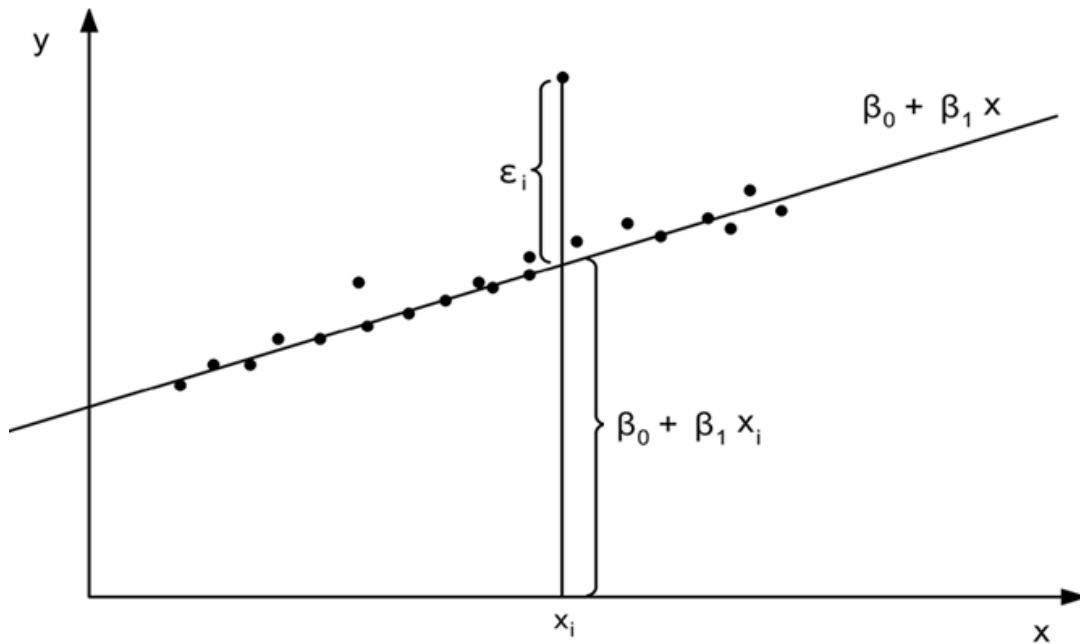


Figura 3 – Equação da reta Mínimos Quadrados.

Fonte: Devore (2006, pág. 443).

### 2.2.2 Método de Correlação Linear

Em estudos que envolvem duas ou mais variáveis, é comum o interesse em conhecer o relacionamento entre elas, além das estatísticas descritivas normalmente calculadas. A medida que mostra o grau de relacionamento entre as variáveis é chamada de Coeficiente de Correlação ou Correlação Linear ou Correlação Linear de Pearson. A Correlação Linear também é conhecida como medida de associação, interdependência, intercorrelação ou relação entre as variáveis (LIRA, 2004, pág. 62).

O Coeficiente de Correlação Linear de Pearson ( $r$ ) é uma estatística utilizada para medir força, intensidade ou grau de relação linear entre duas variáveis aleatórias (FERREIRA, 2009, pág. 664).

Segundo Lopes (2005, pág. 134), a Correlação Linear indica a relação entre duas variáveis. De modo a interpretar o coeficiente de correlação ( $r$ ), podem ser utilizados os seguintes critérios para classificar os resultados obtidos:

- De 0 a 0,50: fraca correlação;
- De 0,51 a 0,84: moderada correlação;
- A partir de 0,85: forte correlação.

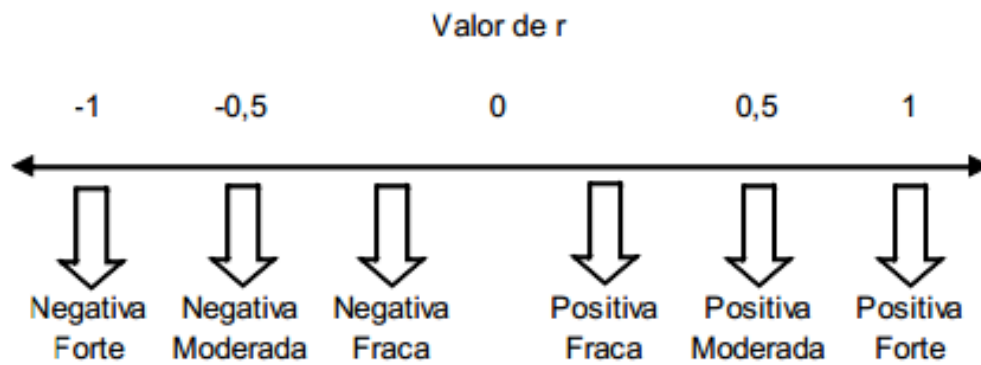


Figura 4 – Classificação da Correlação Linear.

Fonte: [Lopes \(2005, pág. 134\)](#).

Segundo [Regra \(2010, pág. 47\)](#) a Correlação Linear revela o grau de associação entre duas variáveis aleatórias. A dependência de duas variáveis X e Y é dada pelo Coeficiente de Correlação Amostral, conhecido também por coeficiente r-de-Pearson. Designa-se, normalmente, por r e é determinado de acordo com a Figura 5.

$$r = \frac{\text{Cov}(X, Y)}{s_X \cdot s_Y}$$

$$\text{Cov}(X, Y) = \frac{\sum_{i=1}^n XY}{n} - \bar{X} \bar{Y} ,$$

$$s_X = \sqrt{\frac{\sum_{i=1}^k f_i (X_i - \bar{X})^2}{n}} \text{ e } s_Y = \sqrt{\frac{\sum_{i=1}^k f_i (Y_i - \bar{Y})^2}{n}}$$

Figura 5 – Determinação da Correlação Linear.

Fonte: [Regra \(2010\)](#).

Segundo [VIALI \(2009, pág. 8\)](#) as propriedades mais importantes do Coeficiente de Correlação são:

1. O intervalo de variação vai de -1 a +1.
2. O coeficiente é uma medida adimensional, isto é, independente das unidades de medida das variáveis X e Y.
3. Quanto mais próximo de +1 for “r”, maior o grau de relacionamento linear positivo entre X e Y, ou seja, se X varia em uma direção, Y variará no mesmo sentido.
4. Quanto mais próximo de -1 for “r”, maior o grau de relacionamento linear negativo entre X e Y, isto é, se X varia em um sentido, Y variará na direção inversa.
5. Quanto mais próximo de zero estiver “r”, menor será o relacionamento linear entre X e Y. Um valor igual a zero indicará ausência apenas de relacionamento linear.

A análise da Correlação Linear fornece um número, indicando como duas variáveis variam conjuntamente e mede a intensidade e a direção da relação linear ou não-linear entre duas variáveis. Essa análise também é um indicador que atende à necessidade de estabelecer a existência ou não de uma relação entre essas variáveis sem que, para isso, seja preciso o ajuste de uma função matemática. Em suma, o grau de variação conjunta entre X e Y é igual ao de Y e X (LIRA, 2004, pág. 65).

### 2.2.3 Método de Fibonacci

A sucessão ou sequência de Fibonacci é uma sequência de números naturais, na qual os primeiros dois termos são 0 e 1, e cada termo subsequente corresponde à soma dos dois precedentes. A sequência tem o nome do matemático pisano do século XIII Leonardo de Pisa, conhecido como Leonardo Fibonacci, e os termos são chamados números de Fibonacci. Os números de Fibonacci compõem a seguinte sequência de números inteiros: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... (GAGLIARDI, 2013, pág. 6).

Devido a sequência de Fibonacci ser recursiva, é possível determinar uma fórmula capaz de encontrar o valor de qualquer número de Fibonacci,  $F_n$ , se seu lugar na sequência,  $n$ , for conhecido. Esta propriedade garante que para obter todas as soluções da equação recursiva de Fibonacci:  $F_{n+1} = F_{n-1} + F_n$ , para qualquer  $n > 1$  (ALVES., 2012, pág. 12).

Segundo Rocha (2008, pág. 32), a fórmula de Fibonacci, sem uso da recorrência, é dada de acordo com a Figura 6. Essa fórmula também é conhecida como fórmula de Binet.

$$F_n = \frac{1}{\sqrt{5}} \left( \left( \frac{1+\sqrt{5}}{2} \right)^n - \left( \frac{1-\sqrt{5}}{2} \right)^n \right)$$

Figura 6 – Fórmula de Fibonacci sem uso de recorrência.

Fonte: Rocha (2008, pág. 32).

#### 2.2.4 Estocástico

O estocástico é um Método Matemático que gera um tipo de oscilador de momento muito utilizado na análise técnica, principalmente para análises de curto prazo. Dois osciladores estocásticos são geralmente calculados para estimar variações futuras nos preços, o rápido (%K) e o devagar (%D) (ADVFN, 2013).

Segundo ADVFN (2013), o oscilador estocástico rápido e devagar variam entre 0 e 100 e são calculados como:

$$\%K = (FechHoje - Mín_n) \quad (2.8)$$

$$\%D = \sum_1^3 FechHoje - Mín_n \div \sum_1^3 Máx_n - Mín_n \quad (2.9)$$

Sendo que:

- %D = Estocástico Suave
- %K = Estocástico Rápido
- Mín<sub>n</sub> = menor preço durante os últimos "n" períodos
- Máx<sub>n</sub> = maior preço durante os últimos "n" períodos
- n = número de períodos
- $\sum_1^3$  = Somatório das 3 últimas barras

As linhas estocásticas seguem uma escala de 0 a 100 e indicam situações de compra ou venda. Conforme a imagem 7, quando as linhas estocásticas estão acima do nível 80

(linha pontilhada em vermelho), então o mercado está em situação de venda. Quando as linhas estocásticas estão abaixo de 20 (linha pontilhada em azul), então o mercado está numa situação de compra ([INVESTFOREX, 2014](#)).



Figura 7 – Indicador Estocástico.

Fonte: [InvestFOREX \(2014\)](#)

### 2.2.5 Média Móvel

Dada uma sequência de valores, a soma dos mesmos dividido pelo total de termos gera uma média móvel ([WOLFRAM, 2012](#)).



$$MM = \frac{P_1 + P_2 + \dots + P_N}{N}$$

Figura 8 – Equação de Média Móvel.

Fonte: [Wolfram \(2012\)](#)

A Média Móvel (Moving Average) pertence a classe de Métodos Matemáticos que acompanham a tendência. Graças à média móvel é possível encontrar o início e o fim da tendência e, através do ângulo da sua inclinação, determinar a aceleração do movimento ([ROBOFOREX, 2013](#)).

A média móvel é uma forma de analisar a evolução do preço ao longo do tempo. A média móvel é construída a partir do preço médio de fechamento de cotações para os últimos períodos ([INVESTFOREX, 2014](#)).



Figura 9 – Indicador Média Móvel

Fonte: [InvestFOREX \(2014\)](#)

## 2.3 Paradigmas de Programação

A palavra paradigma significa aquilo que pode ser utilizado como padrão, de forma que é um modelo a ser seguido ([FERREIRA, 2008](#)). Segundo [Normak \(2013\)](#), professor

da Universidade de Aalborg na Dinamarca, paradigma de programação é um padrão que serve como uma escola de pensamentos para a programação de computadores.

Uma linguagem de programação multiparadigma suporta mais de um paradigma de programação. O objetivo de tais linguagens é permitir que programadores usem a melhor ferramenta para o trabalho, admitindo que nenhum paradigma resolve todos os problemas da maneira mais fácil ou mais eficiente (PAQUET; MOKHOV, 2010, pág. 21). Seguem noções preliminares sobre cada paradigma, os quais serão investigados ao longo deste trabalho.

### 2.3.1 Paradigma Procedural

Programação procedural é um paradigma de programação, derivado da programação estruturada, com base no conceito da chamada de procedimento. Procedimentos, também conhecidos como rotinas, sub-rotinas, métodos ou funções, contêm uma série de passos computacionais a serem realizados. Qualquer procedimento pode ser chamado a qualquer momento durante a execução de um programa, inclusive por outros procedimentos ou a si mesmo (PAQUET; MOKHOV, 2010, pág. 22).

A linguagem de programação procedural fornece ao programador um meio de definir com precisão cada passo na execução de uma tarefa e é muitas vezes uma escolha melhor em situações que envolvem complexidade moderada ou que requerem significativa facilidade de manutenção (PAQUET; MOKHOV, 2010, pág. 22).

As linguagens procedurais foram desenvolvidas em torno da arquitetura de computadores prevalentes na época, chamada de arquitetura von Neumann, criada pelo matemático húngaro John von Neumann (SEBESTA, 2012, pág. 18).

Em um computador de von Neumann, ambos os dados e programas são armazenados na mesma memória. A unidade central de processamento (CPU), que executa as instruções, é separada da memória. Portanto, as instruções e os dados devem ser transmitidos da memória para a CPU. Os resultados das operações na CPU devem ser devolvidos à memória, conforme ilustra a Figura 10.

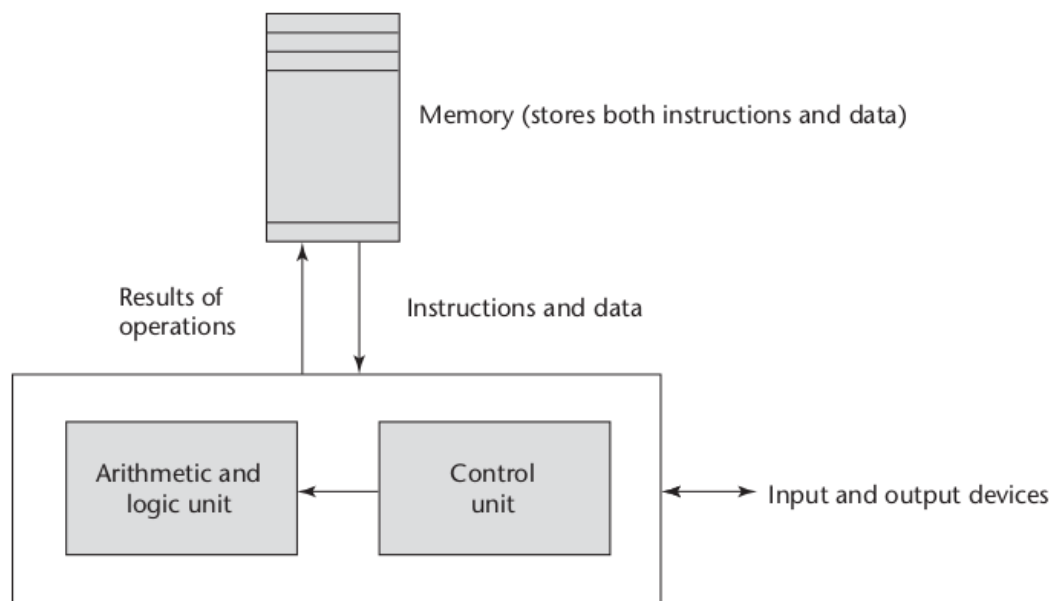


Figura 10 – Arquitetura de von Neumann.

Fonte: [SEBESTA \(2012, pág. 19\)](#).

Devido ao uso da arquitetura de von Neumann, os recursos centrais das linguagens imperativas são as variáveis, as quais modelam as células de memória, as instruções de atribuição, baseadas na operação de canalização (piping) e na forma interativa de repetição, o método mais eficiente dessa arquitetura. A iteração é rápida nos computadores de von Neumann uma vez que as instruções são armazenadas em células adjacentes da memória. Esta eficiência desencoraja o uso de recursão para repetição, embora a recursão seja frequentemente mais natural ([SEBESTA, 2012, pág. 18](#)).

Linguagens imperativas contêm variáveis e valores inteiros, operações aritméticas básicas, comandos de atribuição, sequenciamentos de comandos baseados em memórias, condições e comandos de ramificação. Essas linguagens suportam determinadas características comuns, que surgiram com a evolução do paradigma, tais como: estruturas de controle, entrada/saídas, manipulação de exceções e erros, abstração procedural, expressões e atribuição, e suporte de biblioteca para estruturas de dados ([TUCKER; NOONAN, 2009, pág. 278-279](#)).

Fortran (FORmula TRANslation) foi a primeira linguagem de alto nível a ganhar ampla aceitação, sendo esta uma linguagem imperativa. Projetada para aplicações científicas, essa linguagem conta com notação algébrica, tipos, subprogramas e entrada/saída formatada. Foi implementada em 1956 por John Backus na IBM especificamente para a máquina IBM 704. A execução eficiente foi uma grande preocupação, consequentemente, sua estrutura e comandos têm muito em comum com linguagens de montagem ([BROOKSHEAR, 2003, pág. 458](#)).

Outra linguagem de programação é o C [Ritchie \(1996\)](#), um dos criadores da linguagem, ela se tornou uma linguagem dominante na década de 90. Seu sucesso deve-se ao sucesso do Unix, um sistema operacional implementado em C.

Apesar de alguns aspectos misteriosos para o iniciante e ocasionalmente até mesmo para o adepto, a linguagem C permanece uma simples e pequena linguagem, traduzível com simples e pequenos compiladores. Seus tipos e operações são bem fundamentados naquelas fornecidas por máquinas reais, e para pessoas que usam o OO computador para trabalhar, aprender a linguagem para gerar programas em tempo – e espaço – eficientes não é difícil. Ao mesmo tempo a linguagem é suficientemente abstrata dos detalhes da máquina de modo que a portabilidade de programa pode ser alcançada ([RITCHIE, 1996](#)).

### 2.3.2 Paradigma Orientado a Objetos

Um objeto é a abstração de uma "coisa" (alguém ou algo), e esta abstração é expressa com a ajuda de uma linguagem de programação. A coisa pode ser um objeto real, ou algum conceito mais complicado. Tomando um objeto comum, como um gato, por exemplo, você pode ver que ele tem certas características (cor, nome, peso) e pode executar algumas ações (miar, dormir, esconder, fugir). As características do objeto são chamados de propriedades em Orientação a Objetos (OO) e as ações são chamadas de métodos ([STEFANOV, 2008](#), pág. 13).

O conceito de objeto não surgiu do paradigma orientado a objetos. Pode-se dizer ainda que OO foi uma evolução das práticas já existentes da época. O termo objetos surgiu quase simultaneamente em 1970 em vários ramos da ciência da computação, algumas áreas que influenciaram o paradigma OO foram: sistemas de simulação, sistemas operacionais, abstração de dados e inteligência artificial, como ilustra a figura 11 ([CAPRETZ, 2003](#), pág. 1).

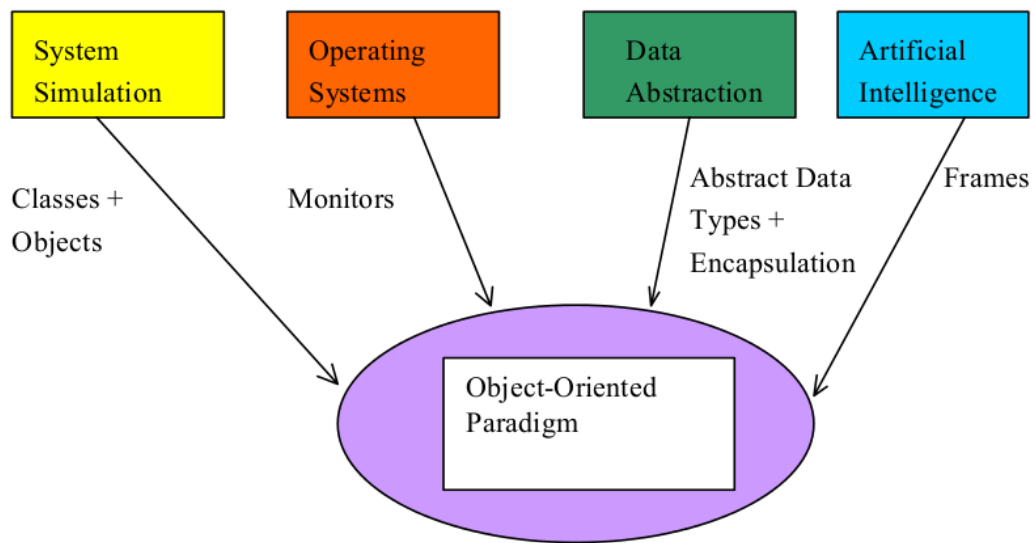


Figura 11 – A essência do Paradigma Orientado a Objetos.

Fonte: CAPRETZ (2003, pág. 1).

A programação orientada a objetos fornece um modelo no qual um programa é uma coleção de objetos que interagem entre si, passando mensagens que transformam seu estado. Neste sentido, a passagem de mensagens permite que os objetos dados se tornem ativos em vez de passivos (TUCKER; NOONAN, 2009, pág. 310).

Classes servem como modelos a partir dos quais objetos podem ser criados. Elas tem precisamente as mesmas variáveis e operações de instâncias dos objetos, mas sua interpretação é diferente: onde um objeto representa variáveis reais, variáveis de classe são, em potencial, instanciadas apenas quando um objeto é criado (WEGNER, 1990, pág. 10).

Uma das principais vantagens do uso de orientação a objetos é que o objeto não precisa revelar todos os seus atributos e comportamentos (métodos). Em um bom design OO um objeto só deve revelar as interfaces necessárias para interagir com outros objetos. Os detalhes não pertinentes para a utilização do objeto deve ser ocultado de todos os outros objetos. Por exemplo, um objeto que calcula o quadrado de um número deve fornecer uma interface para obter o resultado. No entanto, as propriedades internas e algoritmos utilizados para calcular o quadrado não têm de ser disponibilizados ao objeto solicitante. O encapsulamento é definido pelo fato de que os objetos envolvem (quase que como uma cápsula) atributos e métodos, e a ocultação de dados é uma das partes mais importantes do encapsulamento (WEISFELD, 2009, pág. 19).

Uma nova classe (designada por subclasse derivada) pode ser derivada de outra classe (designada por superclasse) por um mecanismo chamado de herança. A classe derivada herda todas as características da classe base: a sua estrutura e comportamento

(resposta a mensagens). Além disso, o mecanismo de herança é permitido mesmo sem acesso ao código-fonte da classe base. Herança dá a OO seu benefício chefe sobre outros paradigmas de programação - a reutilização de código relativamente fácil sem a necessidade de alterar o código fonte existente ([LEAVENS, 2014](#)).

Tendo em vista o conceito de Herança, é possível entender o Polimorfismo, que etimologicamente significa "muitas formas", sendo a capacidade de tratar um objeto de qualquer subclasse de uma classe base, como se fosse um objeto da própria classe base. A classe base, portanto, tem muitas formas: a própria classe base, e qualquer uma de suas subclasses. Se você precisa escrever um código que lida com uma família de tipos, o código pode ignorar detalhes específicos do tipo e apenas interagir com o tipo base da família. Mesmo que o código esteja estruturado para enviar mensagens para um objeto da classe base, a classe do objeto poderia realmente ser a classe base ou qualquer uma de suas subclasses. Isso torna o código extensível, porque outras subclasses podem ser adicionadas mais tarde para a hierarquias de classes ([VENNERS, 1996](#)).

Se tratando de métodos polimórficos pode-se ter sobrecarga e a sobrescrita. Sobrecarga de método é um recurso que permite que uma classe tem dois ou mais métodos com mesmo nome, se suas listas de parâmetros se diferentes com: números de parâmetros passados, tipo de dados dos parâmetros e sequência dos dados passados como parâmetro. Já a sobrescrita é a declaração de um método na subclasse, que já está na classe mãe ([SINGH, 2014b](#)).

Os conceitos de sobrecarga e sobrescrita são constantemente confundidos. A sobrecarga acontece em tempo de compilação, enquanto a sobrescrita ocorre em tempo de execução. A sobrecarga feita na mesma classe, enquanto são necessárias classes mães e suas filhas para o uso sobrescrita, métodos privados sobrecarregados, mas eles não podem ser sobrescritos. Isso significa que uma classe pode ter mais de um método privado mesmo nome, mas uma classe filha não pode substituir os métodos privados sua classe mãe ([SINGH, 2014a](#)).

Reusabilidade é uma das grandes promessas da tecnologia orientada a objetos. Reutilização de código, o tipo mais comum de reuso, refere-se à reutilização de código-fonte dentro de seções de uma aplicação e potencialmente através de múltiplas aplicações. Em alguns casos, reutilização de código é alcançada compartilhando-se classes, coleções de funções e rotinas comuns. A vantagem do reuso de código é que ela reduz a quantidade real de código que você precisa escrever. Há ainda a reutilização de herança, que refere-se ao uso de herança em sua aplicação para tirar vantagem de comportamento implementado em classes existentes ([AMBLER, 1998](#)).

### 2.3.3 Paradigma Orientado a Agentes

Um agente é qualquer coisa que pode perceber seu ambiente através de sensores e agir sobre esse ambiente através de atuadores. Um agente humano tem olhos, ouvidos e outros órgãos para sensores e como atuadores possui mãos, pernas, tato, voz, e assim por diante. Um agente robótico pode ter câmeras e localizadores, faixa do infravermelho como sensores e vários motores para atuadores. Um agente de software recebe as teclas digitadas, conteúdo de arquivos e pacotes de rede como entradas sensoriais e age sobre o meio ambiente por meio da exibição na tela, gravação de arquivos e envio de pacotes de rede (RUSSEL; NORVIG, 1995, pág. 34).

A exigência de continuidade e autonomia deriva nosso desejo de que um agente seja capaz de realizar atividades de uma forma flexível e inteligente, que é sensível a alterações no ambiente sem a necessidade de orientação constante humana ou de intervenção. Idealmente, um agente que funciona de forma contínua, num ambiente ao longo de um período de tempo seria capaz de aprender com sua experiência. Além disso, espera-se um agente que habita um ambiente com outros agentes e processos pode ser capaz de se comunicar e cooperar com eles (BRADSHAW, 1997, pág. 8).

O autor WOOLDRIDGE (2010, pág. 42) considera quatro tipos de arquitetura de agentes: baseados em lógica, reativos, em camadas e de crença-desejo-intenção.

Nas arquiteturas baseadas em lógica os agentes contêm um modelo simbólico do ambiente do agente, explicitamente representado em uma base de conhecimento e a decisão da ação a executar é tomada a através de raciocínio lógico (GIRARDI, 2004, pág. 3).

Arquiteturas reativas deixam o raciocínio abstrato de lado e destinam-se a lidar com comportamentos básicos. Os agentes reagem às mudanças no ambiente como uma forma de resposta ao estímulo, executando as rotinas simples que corresponde a um estímulo específico (SCHUMACHER, 2001, pág. 13).

A arquitetura em camadas, também conhecido como arquitetura híbrida, híbrida combina componentes da arquitetura baseada em lógica e da reativa. Esta arquitetura propõe um subsistema deliberativo que planeja e toma decisões da maneira proposta pela Inteligência Artificial Simbólica e um reativo capaz de reagir a eventos que ocorrem no ambiente sem se ocupar de raciocínios complexos (COSTA, 2004, pág. 44).

Na arquitetura BDI o estado do agente é representado por três estruturas: suas crenças (beliefs), que são o conhecimento do agente sobre seu ambiente; seus desejos (desires), representam objetivos ou situações que o agente gostaria de realizar ou trazer; por fim tem-se suas intenções (intentions), são suas ações que têm decidido realizar (GIRARDI, 2004, pág. 3).

Um sistema multiagente (SMA) pode ser caracterizado como um grupo de agentes que atuam em conjunto no sentido de resolver problemas que estão além das suas habilidades individuais. Os agentes realizam interações entre eles de modo cooperativo para atingir uma meta ([GIRARDI, 2004](#), pág. 6).

Para [WOOLDRIDGE \(2010, pág. 3\)](#) o uso de SMA se justifica uma vez que muitas tarefas não podem ser feitas por um único agente, e há ainda tarefas que são feitas de forma mais eficaz quando realizada por vários agentes. Para tal é essencial que o SMA seja capaz de: trabalhar em conjunto para alcançar um objetivo comum, monitorar constantemente o progresso do esforço da equipe como um todo, ajudar um ao outro quando necessário, coordenação das ações individuais de modo que eles não interfiram um com o outro, comunicar sucessos e fracassos, se necessário para a equipe para ter sucesso parcial.

SMA muitas vezes baseia-se em conceitos de outras disciplinas, como a psicologia, a ciência econômica, cognitiva, lingüística, inteligência artificial, etc. Por exemplo, analisar protocolos de interação e ações de comunicação entre os agentes com base na teoria dos atos de fala, vem da filosofia e da lingüística. A abstração da postura intencional foi emprestado da ciência cognitiva para analisar e raciocinar sobre os comportamentos autônomos de agentes. Recentemente, muitas metodologias e modelos de abstrações e conceitos de organização e sociologia foram propostas para modelar, analisar e projetar SMA ([ODELL; GIORGINI; MÜLLER, 2005](#), pág. 1).

Agentes autônomos e SMA representam uma nova forma de analisar, projetar e implementar sistemas de software complexos. A visão orientada a agentes oferece um repertório poderoso de ferramentas, técnicas e metáforas que têm o potencial de melhorar consideravelmente a maneira como as pessoas conceituam e implementam muitos tipos de software. Agentes estão sendo usados em uma ampla variedade de aplicações, desde de pequenos sistemas, tais como filtros de e-mail personalizados, a sistemas grandes e complexos de missão crítica, como controle de tráfego aéreo. À primeira vista, pode parecer que tais tipos de sistema tem pouco em comum. E, no entanto este não é o caso: em ambos, a abstração chave usada é a de um agente ([JENNINGS; SYCARA; WOOLDRIDGE, 1998](#)).

Segundo [JENNINGS e WOOLDRIDGE \(1995\)](#) os agentes de software tem as seguintes propriedades: autonomia, competência social, reatividade e pró-atividade.

Os agentes mantêm uma descrição do seu próprio estado de processamento e o estado do mundo em torno deles, logo eles são ideais para aplicações de automação. Agentes autônomos são capazes de operar sem entrada ou intervenção do usuário. Podendo ser utilizados em como instalações e automação de processos ([AGENTBUILDER, 2009b](#)).

A empresa Acronymics e a Alternative Energy Systems Consultants (AESC) realizaram uma pesquisa afim de criar agentes de softwares capazes de comprar e vender



energia elétrica participando do mercado eletrônico. Cada agente apresentavam um comportamento único e individual determinado pelo seu próprio modelo econômico, esta pesquisa mostrou que os agentes podem ser usados para implementar mercados e leilões eletrônicos, e que um agente pode adotar os objetivos e intenções de seu stakeholder (AGENTBUILDER, 2009a).

### 2.3.4 Programação Declarativa

Linguagens declarativas permitem ao programador se concentrar na lógica de um algoritmo, diferentemente de linguagens imperativas que requerem do programador se concentrar tanto na lógica quanto no controle de um algoritmo, para tal se dá o nome de atribuição não-destrutiva. Nos programas declarativos, os valores associados aos nomes de variáveis não podem ser alterados. Assim, a ordem na qual definições ou equações são chamadas, não interessa. Além disso, as definições declarativas não permitem efeitos secundários, isto é, o cálculo de um valor não irá afetar outro valor (COENEN, 1999).

De fato, em algumas situações, a especificação de um problema no formato adequado já constitui a solução para o problema algorítmico. Em outras palavras, a programação declarativa torna possível escrever especificações executáveis. Entretanto, na prática, os programas obtidos desse modo são frequentemente ineficientes, visto que esta abordagem de programação tem associado, ao uso adequado de transformações dos programas (APT, 1996, pág. 2).

#### 2.3.4.1 Paradigma Funcional

O centro da programação funcional é a idéia de uma função. A linguagem de programação funcional dá um modelo simples de programação: dado um valor, o resultado é calculado com base em outros valores, as entradas da função. Devido à fundação simples, uma linguagem funcional dá uma visão mais clara das idéias centrais da computação moderna, incluindo abstração, polimorfismo e sobrecarga (THOMPSON, 1999, pág. 16).

A primeira linguagem de programação funcional foi inventada para oferecer recursos de linguagem para processamento de listas, cuja necessidade surgiu a partir das primeiras aplicações na área da inteligência artificial (TUCKER; NOONAN, 2009, pág. 361).

A programação funcional exige que as funções sejam cidadãos de primeira classe, o que significa que elas são tratadas como quaisquer outros valores e podem ser passadas como argumentos para outras funções ou serem retornadas como um resultado de uma função. Sendo cidadãos de primeira classe também significa que é possível definir e manipular funções dentro de outras funções (HOOGLÉ, 2013).

Uma linguagem de programação puramente funcional não usa variáveis ou instruções de atribuição. Isso libera o programador de preocupar-se com as células da memória

do compilador no qual o programa é executado. Sem variáveis, construções interativas não são possíveis, porque elas são controladas por variáveis. A repetição deve ser feita por meio de recursão, não por meio de laços (SEBESTA, 2012, pág. 555).

Por ser programação declarativa o paradigma funcional não tem efeitos colaterais, uma função não faz nada que não seja retornar seu valor de resultado, com isso fica fácil trazer uma experiência matemática para a programação (PIPONI, 2006).

#### 2.3.4.2 Paradigma Lógico

Na programação lógica, um programa consiste de uma coleção de declarações expressas como fórmulas da lógica simbólica. Existem regras de inferência da lógica que permitem uma nova fórmula derivada das antigas, com a garantia de que se as últimas fórmulas são verdadeiras, então a nova regra também será (SPIVEY, 1996, pág. 2).

Em outros paradigmas como o imperativo, uma pergunta terá sempre uma única resposta. A programação lógica é baseada na noção de que um programa implementa uma relação ao invés de um mapeamento. Desta forma, podemos fazer pedidos como: Dado A e B, determinar se a Relação(A, B) é verdadeira; dado A, encontrar todos os Bs tal que a Relação(A, B) é verdadeira; dado B, encontrar todos os As, tal que a Relação(A, B) é verdadeira; pesquisar os As e Bs para o qual a Relação(A, B) é verdadeira (PAQUET; MOKHOV, 2010, pág. 33).

Para garantir que o programa dará respostas corretas, o programador deve verificar se o programa contém apenas declarações verdadeiras, e em número suficiente para garantir que as soluções a serem derivadas resolvem todos os problemas que são de interesse. O programador também pode garantir que as derivações que a máquina realizará são bastante curtas, de modo que a máquina pode encontrar respostas rapidamente. No entanto, cada fórmula pode ser entendida no isolamento como uma verdadeira declaração sobre o problema a ser resolvido (SPIVEY, 1996, pág. 2).

A programação lógica surgiu como um paradigma distinto nos anos 70. Ela é diferente dos outros paradigmas porque requer que o programador declare os objetivos da computação, em vez dos algoritmos detalhados por meio dos quais esses objetivos podem ser alcançados (TUCKER; NOONAN, 2009, pág. 412).

O paradigma lógico também tem como característica a facilidade de representar conhecimento, tornando-o extremamente poderoso para resolução de problemas como: análise de teoremas matemáticos, inteligência Artificial, sistemas especialistas, processamento de linguagem natural, redes semânticas e banco de dados (ALMEIDA, 2010).

## 2.4 Teste de Software

Teste de Software é um processo de execução de um programa elaborado para garantir que código fonte faz o que foi projetado para fazer e que não faz nada de maneira não intencional, desta forma, seu objetivo encontrar erros (MYERS, 2004, pág. 8) (MYERS, 2004, pág. 8).

Para entender testes de software é fundamental que se conheça a diferença entre defeito, erro e falha. Segundo as definições estabelecidas pelo Institute of Electrical and Electronics Engineers (IEEE), defeito é uma instrução ou definição de dados incorretos; já o erro é qualquer estado intermediário incorreto ou resultado inesperado, ou seja, uma manifestação concreta de um defeito e por fim a definição de falha, é o comportamento operacional do software diferente do esperado pelo usuário (IEEE 610, 1990).

"Defeitos fazem parte do universo físico (a aplicação propriamente dita) e são causados por pessoas, por exemplo, através do mau uso de uma tecnologia. Defeitos podem ocasionar a manifestação de erros em um produto, ou seja, a construção de um software de forma diferente ao que foi especificado (universo de informação). Por fim, os erros geram falhas, que são comportamentos inesperados em um software que afetam diretamente o usuário final da aplicação (universo do usuário) e pode inviabilizar a utilização de um software"(NETO, 2005).



Figura 12 – Defeito x Erro x Falha.

Fonte: Neto (2005).

Teste e depuração são frequentemente confundidos, alguns acreditam até que são sinônimos, como foi mostrado anteriormente. A finalidade dos testes é mostrar que o programa tem erros, já o objetivo da depuração é encontrar o erro ou equívoco que levou ao fracasso do programa. Teste começa com condições conhecidas, utiliza procedimentos pré-definidos, e tem resultados previsíveis. A depuração começa a partir de condições iniciais, possivelmente desconhecidas (BEIZER, 1990).

Os testes podem ser projetados a partir de um ponto de vista funcional ou de um ponto de vista estrutural. Os testes funcionais são sujeitos a entradas, e suas saídas são verificadas afim de encontrar, ou não, conformidades com o comportamento especificado. O usuário do software deve se preocupar apenas com as funcionalidade e características (BEIZER, 1990).

Caixa-preta ou teste funcional: “Teste de que ignora o mecanismo interno de um sistema ou componente e se concentra exclusivamente nas saídas geradas em resposta a entradas selecionadas e condições de execução” (IEEE 610, 1990).

No método caixa-preta, como o próprio nome revela, vemos o programa como uma caixa preta. Seu objetivo é ser completamente indiferente sobre o comportamento interno e estrutura do programa. Em vez disso, se concentra em encontrar circunstâncias em que o programa não se comportam de acordo com as suas especificações (MYERS, 2004, pág. 13).

Caixa-branca ou teste estrutural: “Testes que leva em conta o mecanismo interno de um sistema ou componente” (IEEE 610, 1990).

As conotações indicam adequadamente que você tem total visibilidade do funcionamento interno do produto de software, especificamente, a lógica ea estrutura do código (WILLIAMS, 2006, pág. 1).

Os testes são realizados em diferentes níveis, envolvendo o todo o sistema ou parte dele. São quatro níveis de teste: os teste de unidade, de integração, de sistema e de aceitação, como mostrado na Figura 13 (NAIK; TRIPATHY, 2008, pág. 18).

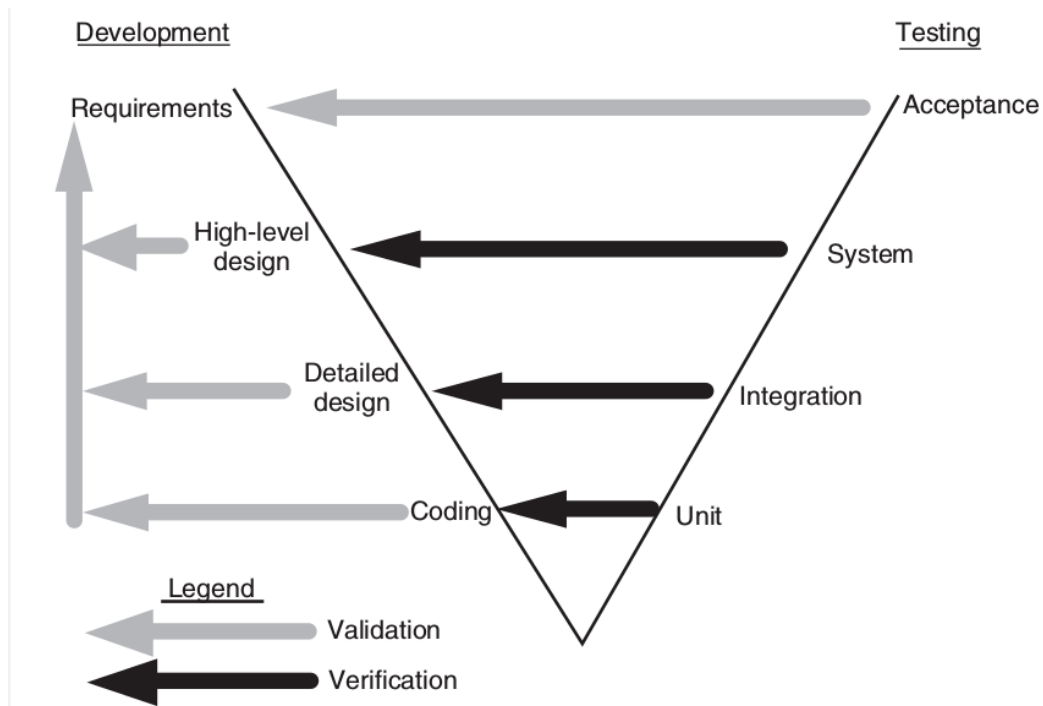


Figura 13 – Níveis de teste e seu desenvolvimento.

Fonte: [Naik e TRIPATHY \(2008, pág. 18\)](#).

### 2.4.1 Testes Unitários

"Os testes de hardware individuais ou unidades de software ou grupos de unidades relacionadas"([IEEE 610, 1990](#)).

Testa-se unidades individuais do programa, como as funções, métodos ou classes, de forma isolada. Depois de garantir que as unidades funcionam de forma satisfatória, os módulos são montados para a construção de sub-sistemas maiores, seguindo técnicas de teste de integração ([NAIK; TRIPATHY, 2008, pág. 18](#)).

Os testes unitários devem ser capazes de examinar o comportamento do código sob as mais variadas condições, ou seja, como o código deve se comportar se determinado parâmetro for passado (ou não), o que ele retorna se determinada condição for verdadeira, os efeitos colaterais que ele causa durante a sua execução, se determinada exceção é lançada ([THIAGO, 2001](#)).

O teste de unidade é importante para garantir que o código é sólido antes de ser integrado com outro código. Uma vez que o código está integrado na base de outro código, a causa de uma falha é mais difícil de ser encontrado ([WILLIAMS, 2006](#)).

### 2.4.2 Teste de integração

“Testes em que componentes de software, componentes de hardware, ou ambos são combinados e testados para avaliar a interação entre eles” (IEEE 610, 1990).

Seu objetivo é a construção de um sistema razoavelmente estável que pode suportar o rigor dos testes de nível de sistema (NAIK; TRIPATHY, 2008, pág. 18).

O teste efetuado de modo incremental através da combinação de módulos em etapas. Em cada etapa um módulo é adicionado à estrutura do programa, assim o teste se concentra neste módulo recém-adicionado. Depois de ter sido demonstrado que um módulo integra adequadamente com a estrutura do programa, um outro módulo é adicionado, e o teste continua. Este processo é repetido até que todos os módulos foram integrados e testados (LEWIS, 2009, pág. 134).

Os casos de teste são escritos para examinar explicitamente as interfaces entre as diversas unidades. Estes casos de teste podem ser de caixa preta, em que o testador entende que um caso de teste requer várias unidades do programa para interagir. Como alternativa, têm-se os casos de teste caixa-branca que são escritos para exercer explicitamente as interfaces que são conhecidos para o testador (WILLIAMS, 2006).

### 2.4.3 Teste de Aceitação

Teste de Aceitação é um teste formal realizado para permitir que um usuário, cliente ou outra entidade autorizada, para determinar se a aceita um sistema ou componente (IEEE 610, 1990).

Antes de instalar e utilizar o software na vida real outro último nível de teste deve ser executado: o teste de aceitação. Aqui, o foco está no cliente de e na ótica do usuário. O teste de aceitação pode ser a única prova de que os clientes estão efetivamente envolvidos (SPILLNER; LINZ; SCHAEFER, 2014, pág. 61).

Geralmente, é realizada por um cliente ou usuário final do programa e, normalmente, não é considerada a responsabilidade da organização de desenvolvimento. No caso de um programa contratado, a organização contratante (usuário) realiza o teste de aceitação, comparando a operação do programa com o contrato original. Como é o caso de outros tipos de testes, a melhor maneira de fazer isso é criar casos de teste que tentam mostrar que o programa não atende o contrato, se esses casos de teste não forem bem sucedidos, o programa é aceito (MYERS, 2004, pág. 104).

## 2.5 Qualidade de Software

“Qualidade é um termo que pode ter diferentes interpretações e para se estudar a qualidade de software de maneira efetiva é necessário, ini-

cialmente, obter um consenso em relação à definição de qualidade de software que está sendo abordada” (BRAGA, 2012, pág. 11).

A qualidade de software é classificada em fatores internos e externos. Fatores externos são aqueles em que usuários comuns conseguem detectar, como por exemplo, a velocidade do software ou a facilidade de uso. Os fatores externos agregam bastante valor ao usuário final, mas um bom caminho para assegurar que os fatores externos terão qualidade é caprichar na construção dos fatores internos. Portanto, técnicas de qualidade interna, como análise de qualidade de código fonte, são meios para atingir a qualidade externa (BUENO; CAMPELO, 2011, pág. 4).

### 2.5.1 Métricas de Qualidade de Código Fonte

Conceitos de qualidade são imprecisos e difíceis de serem aceitos em diversos contextos. No contexto de desenvolvimento de software, métricas de qualidade de código possuem o intuito de mensurar qualidade do produto de software (BUENO; CAMPELO, 2011, pág. 1).

As métricas de qualidade de código fonte podem ser objetivas ou subjetivas. As métricas subjetivas são obtidas através de regras bem definidas. As métricas subjetivas depende do sujeito que está realizando a medição. As métricas objetivas podem ser calculadas a partir de uma análise estática de código fonte de um software. Os resultados das métricas podem ser mapeados em intervalos com o intuito de serem interpretados quando analisados (MEIRELLES, 2013, pág. 14)

As métricas de qualidade de código fonte possuem papel fundamental no monitoramento e controle nas atividades de codificação e testes, realizados quase sempre por equipes que possuem diferentes formas de pensar e de realizar o trabalho criativo de codificação (SOMMERVILLE, 2011, pág. 341).

“Ao contrário de outras engenharias, a engenharia de software não é baseada em leis quantitativas básicas, medidas absolutas não são comuns no mundo do software. Ao invés disso, tenta-se derivar um conjunto de medidas indiretas que levam a métricas que fornecem uma indicação de qualidade de alguma representação do software. Embora as métricas para software não sejam absolutas, elas fornecem uma maneira de avaliar qualidade através de um conjunto de regras definidas” (BUENO; CAMPELO, 2011, pág. 5).

### 2.5.2 Análise Estática de Código Fonte

A análise estática de código é um processo automatizado realizado por uma ferramenta sem a necessidade de execução do programa ou software a ser verificado (CHESS; WEST, 2007, pág. 64).

Na utilização da análise estática de código fonte, falhas são descobertas mais cedo no desenvolvimento, antes do programa ser executado, ainda em versões de testes. A real causa dos defeitos é revelada e não apenas suas consequências (MELO, 2011, pág. 19).

Filho (2013), define um intervalo de interpretação das métricas de qualidade de código fonte para se saber se a qualidade do código está preocupante, regular, boa ou excelente conforme a Figura 14.

	ACC	ACCM	ANPM	DIT	NPA	SC
Excelente	[0, 2[	[0, 3[	[0, 2[	[0, 2[	[0, 1[	[0, 12[
Bom	[2, 7[	[3, 5[	[2, 3[	[2, 4[	[1, 2[	[12, 28[
Regular	[7, 15[	[5, 7[	[3, 5[	[4, 6[	[2, 3[	[28, 51[
Preocupante	[15, ∞[	[7, ∞[	[5, ∞[	[6, ∞[	[3, ∞[	[51, ∞[

Figura 14 – Intervalo para interpretação das métricas.

Fonte: Filho (2013).

### 2.5.3 Ferramentas de Análise Estática

Ferramentas de análise estática de código fonte varrem o código fonte e detectam possíveis anomalias. Também pode ser usados como parte de um processo de inspeção (SOMMERVILLE, 2011, pág. 345).

Obter os dados e extrai-los para análise estática de código fonte, não é uma tarefa trivial e requer a utilização de ferramentas automatizadas (MEIRELLES, 2013, pág. 2)

As métricas de qualidade de código fonte podem ser obtidas através do uso de uma ou mais ferramentas de extração de métricas. Existem diversas ferramentas para realização dessa atividade, mas não é possível afirmar que uma ferramenta é melhor que outras. Todas possuem pontos fortes e fracos. Para se escolher uma ferramenta de análise estática, deve-se analisar o contexto em que as mesmas estão inseridas como, por exemplo, a linguagem do projeto a ser analisado (MILLANI, 2013).



### 3 Metodologia de pesquisa



## 4 O InvestMVC

### 4.1 Arquitetura Orientada a Componentes

Arquitetura Orientada a Componentes possui o propósito de dividir para conquistar. Um grande problema é quebrado em partes menores e em seguida se desenvolve soluções mais elaboradas (CHEESMAN; DANIELS, 2001).

Por usar vários paradigmas, a ferramenta InvestMVC terá vários componentes e cada componente terá seu paradigma de programação e responsabilidade bem definida, como mostrado na Figura 15.

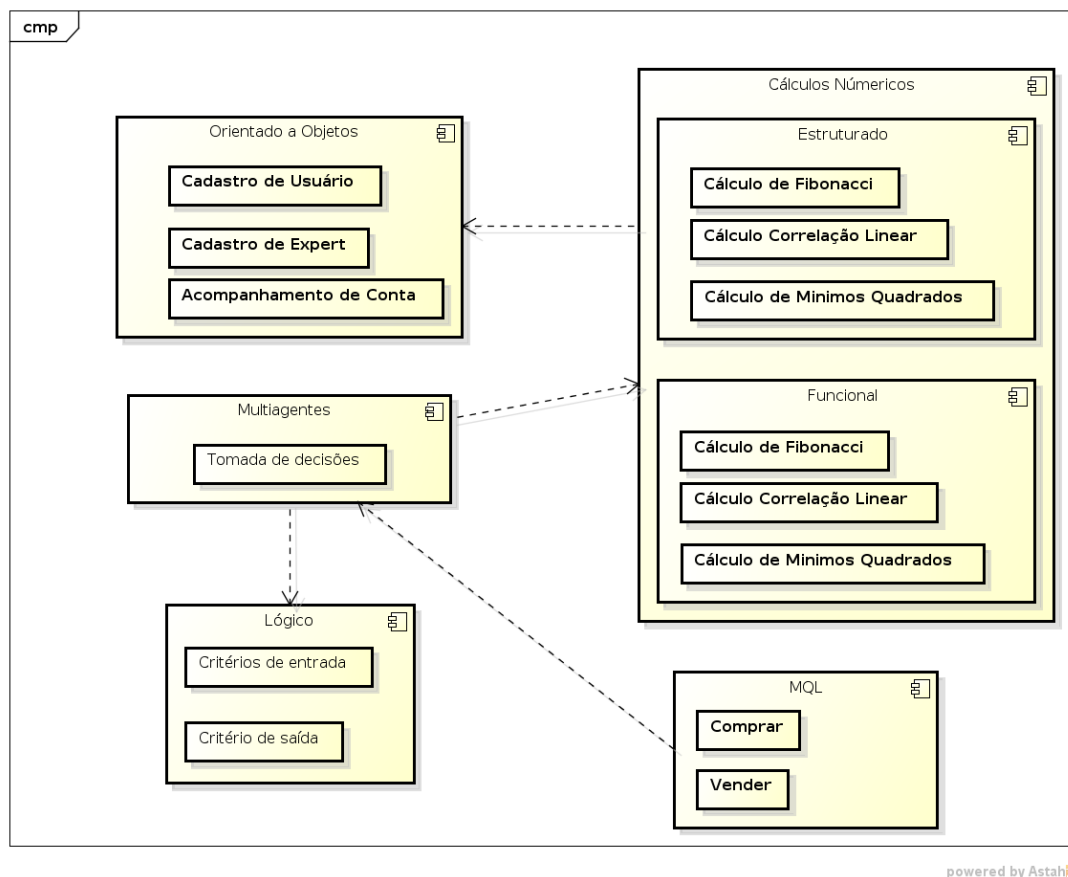


Figura 15 – Diagrama de Sequência InvestMVC

#### 4.1.1 Componente Orientado a Objetos

Este componente fará a interação com o usuário e será implementado em linguagem Groovy. A escolha dessa linguagem, se deu porque esta é voltada para aplicações web e

juntamente com o framework grails, fornece a criação de um projeto com uma arquitetura MVC definida, como demonstra a Figura 16.

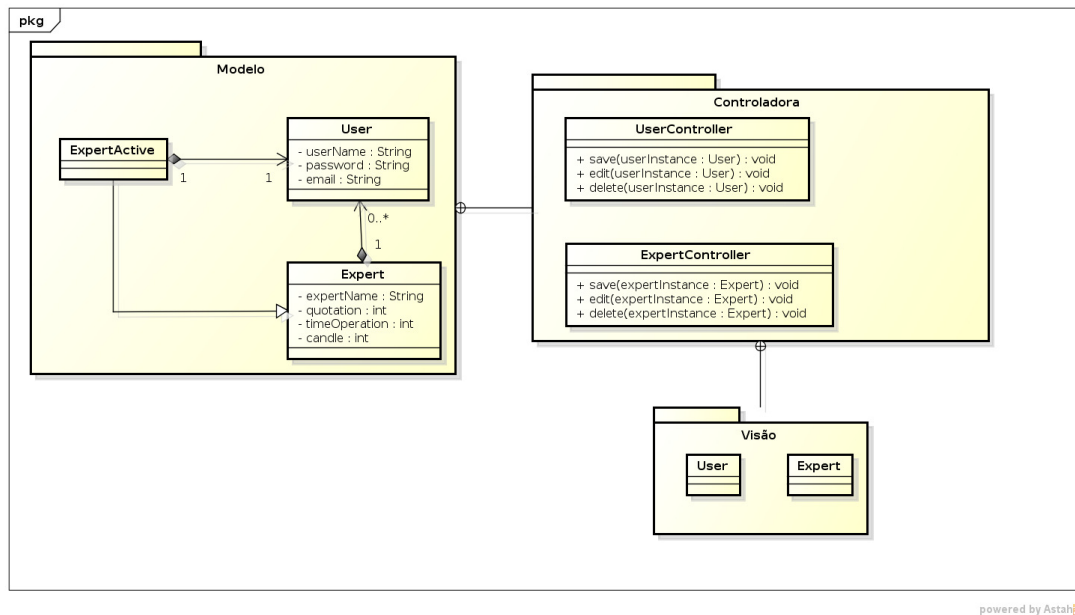


Figura 16 – Diagrama de Classe InvestMVC componente Orientado a Objetos

Segundo Lamim (2010) na arquitetura MVC o controle de fluxo de dados dentro deste módulo ocorre da seguinte forma:

1. O usuário, neste caso o investidor, interage com a Visão.
2. A Controladora manipula o evento da interface do usuário através de uma rotina.
3. A Controladora acessa o Model, atualizando-o baseado na interação do usuário.

O diagrama de sequência do componente Orientado a Objetos é evidenciado na Figura 17.

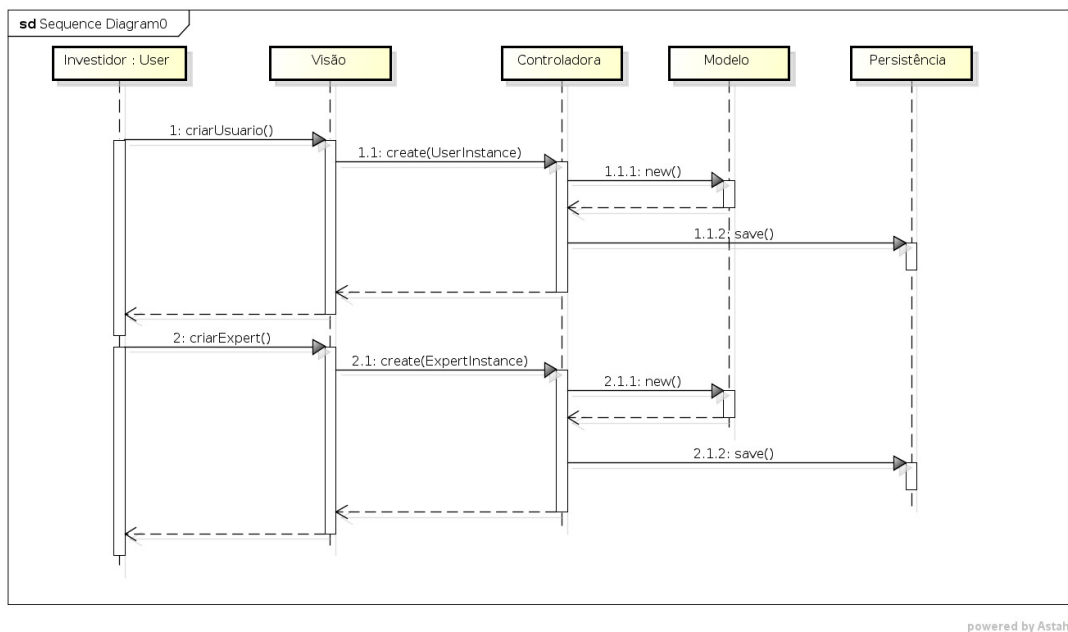


Figura 17 – Diagrama de sequência InvestMVC componente Orientado a Objetos

### 4.1.2 Componente Cálculos Numéricos

O componente Cálculos Numéricos será responsável por calcular os Métodos Matemáticos presentes na ferramenta InvestMVC. Este módulo é composto por dois outros módulos: módulo Estruturado e módulo Funcional.

O componente Estruturado da ferramenta InvestMVC será programado em linguagem C e o módulo Funcional em linguagem Haskell. Ambos os componentes irão realizar os mesmos cálculos. Isso aumenta a probabilidade de não ocorrer erros nos cálculos dos Métodos Matemáticos. Caso um dos componentes por algum motivo não seja executado no momento correto, a tendência é que o outro módulo realize os cálculos.

#### 4.1.2.1 Componente Funcional

Por ser uma linguagem de programação funcional sua "gramática" está próxima das funções matemáticas, logo a implementação dos métodos algébricos e numéricos se torna muito intuitiva. (HOOGLE, 2013).

O paradigma funcional é declarativo, por limitar o uso de atribuições à variáveis suas funções são mais precisas do que em outros paradigmas (PIPONI, 2006).

Devido aos fatos externalizados, o paradigma funcional, foi eleito para implementar os Métodos Matemáticos de Correlação Linear, Mínimos Quadrados e Fibonacci. Por ser simples, este componente será formado apenas por seus 4 arquivos haskell, cada arquivo realiza o cálculo de um Método Numérico, com exceção do arquivo Arquivos.hs, que faz a leitura de arquivos.

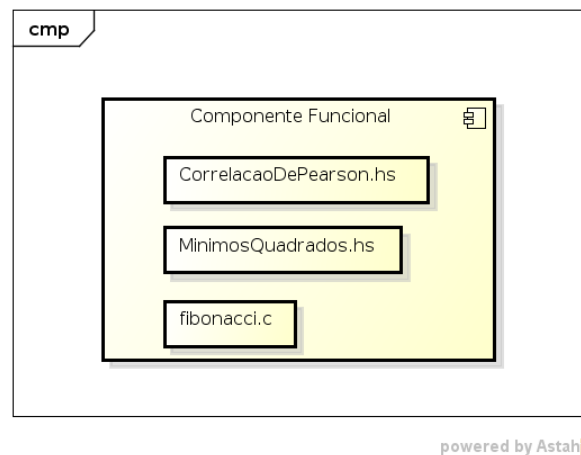


Figura 18 – Componente Funcional InvestMVC

O componente Funcional espera uma solicitação de cálculo do componente Multiagente, logo após a solicitação o componente busca na persistência (um arquivo) as cotações do mercado, com estas cotações o componente é capaz de realizar o cálculo do Método Numérico que é esperado pelo componente Multiagente, como é mostrado na Figura 19.

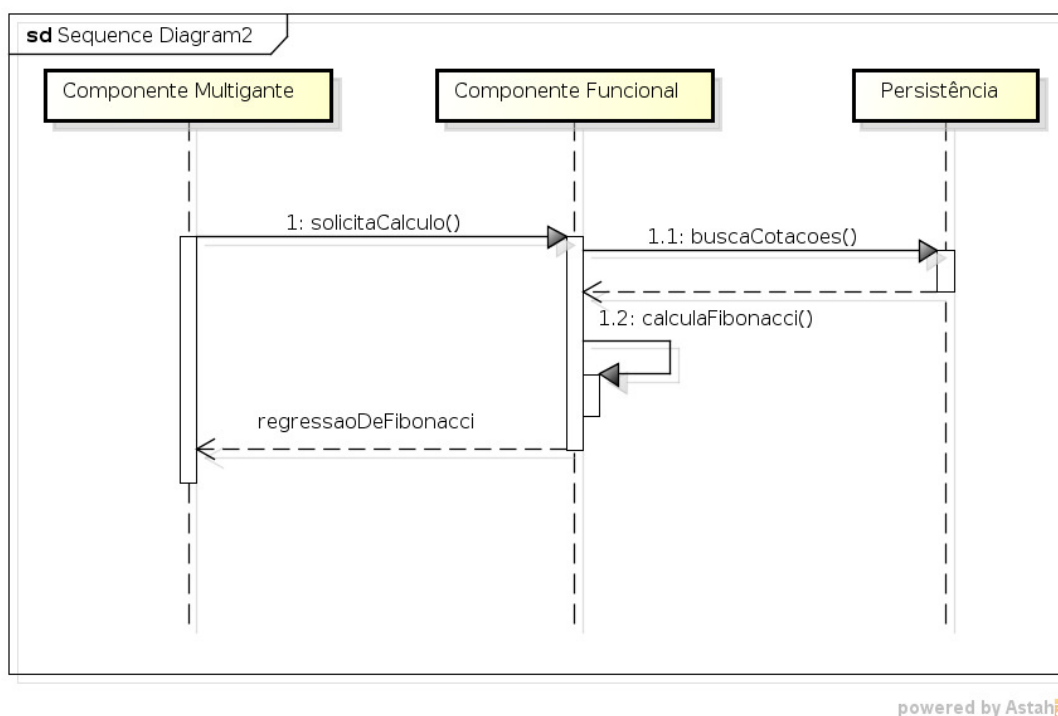


Figura 19 – Diagrama de Sequência do Componente Funcional InvestMVC

#### 4.1.2.2 Componente Estruturado

A linguagem de programação C é estruturada e possui a vantagem da velocidade de execução do código fonte. Também é uma linguagem bastante utilizada para realizar

cálculos numéricos e algébricos (JUNGTHON; GOULART, 2009).

Devido aos fatos externalizados, o paradigma estruturado utilizando a linguagem C, também foi eleito para implementar os Métodos Matemáticos de Correlação Linear, Mínimos Quadrados e Fibonacci.

A arquitetura e sequência do fluxo de dados do Componente Estruturado segue a mesma lógica do Componente Funcional, como ilustrado na Figura 20.

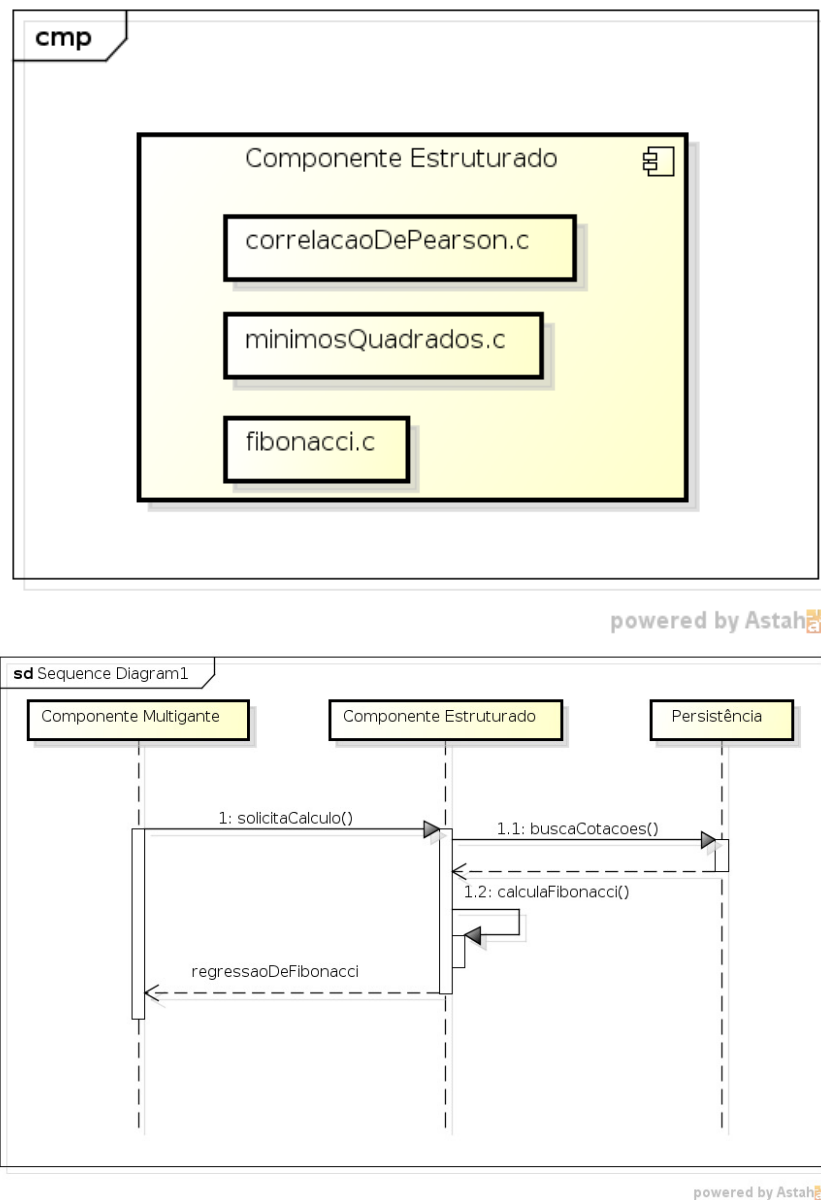


Figura 20 – Diagramas de Componentes e de Sequência do Componente Estruturado InvestMVC.

### 4.1.3 Componente Multiagentes

O componente Multiagente será implementado usando o paradigma Multiagente com linguagem Java.

Agentes de software são entidades autônomas e com capacidades sociais, o uso deste paradigma justifica-se na tomada de decisões (AGENTBUILDER, 2009b).

Os agentes da ferramenta InvestMVC possuem uma arquitetura reativa, pois suas ações se dão pelas variações que ocorrem nas cotações do Mercado de Moedas.

A arquitetura do Componente Multiagente está modularizado por pacotes: o pacote comportamentos será formado por comportamentos que serão usados pelos Agentes de Software, o pacote metodosNumericos será formado por Agentes que acerão o componente Cálculos Numéricos, o pacote investidores será composto por Agentes que vão interagir com o Componente MQL e o pacote execucao iniciará a execução do SMA.

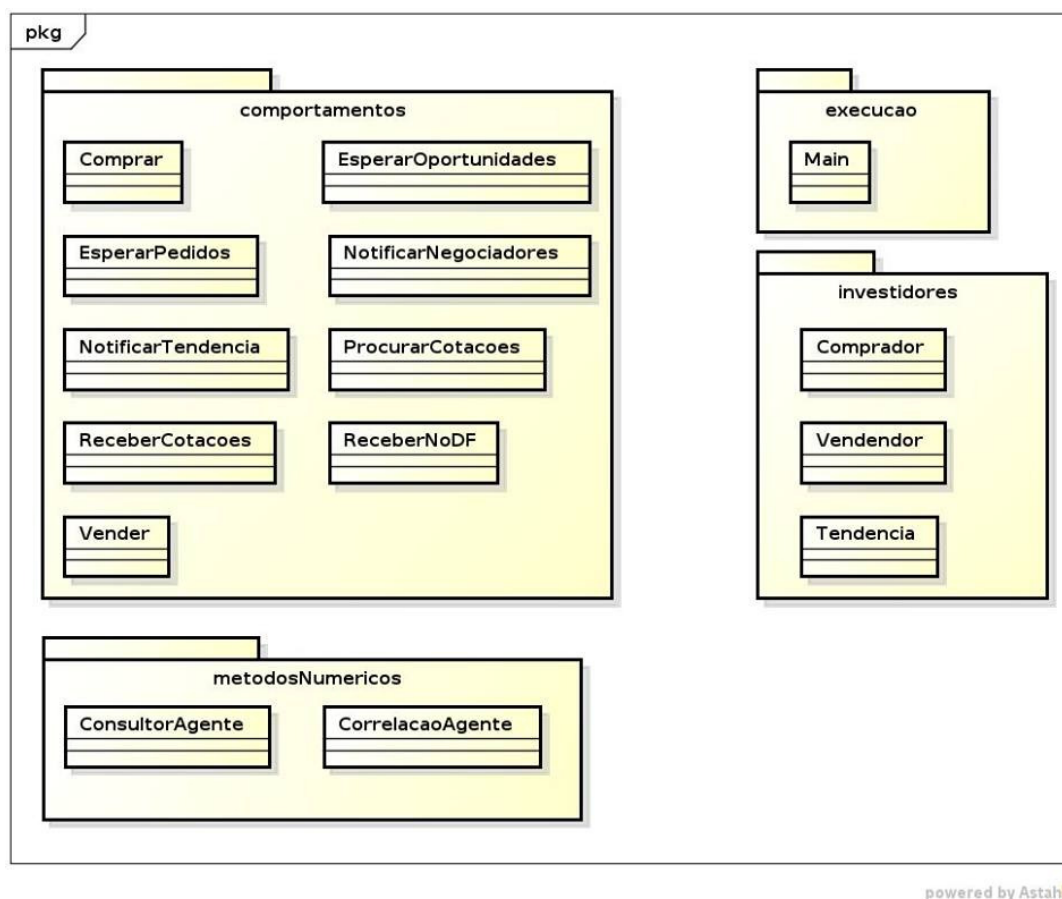


Figura 21 – Diagrama de Classe do Componente Multiagente InvestMVC

### 4.1.4 Componente Lógico

O componente Lógico será produzido em linguagem Prolog e vai definir uma base de conhecimento que servirá como critério de entrada e saída no Mercado de Moedas.



O paradigma Lógico facilita a representação, inserção e recuperação de conhecimento, por isso é muito usado em aplicações com Inteligência Artificial (ALMEIDA, 2010).

A interação do Componente Lógico com o Componente Multiagentes é evidenciada na Figura 22.

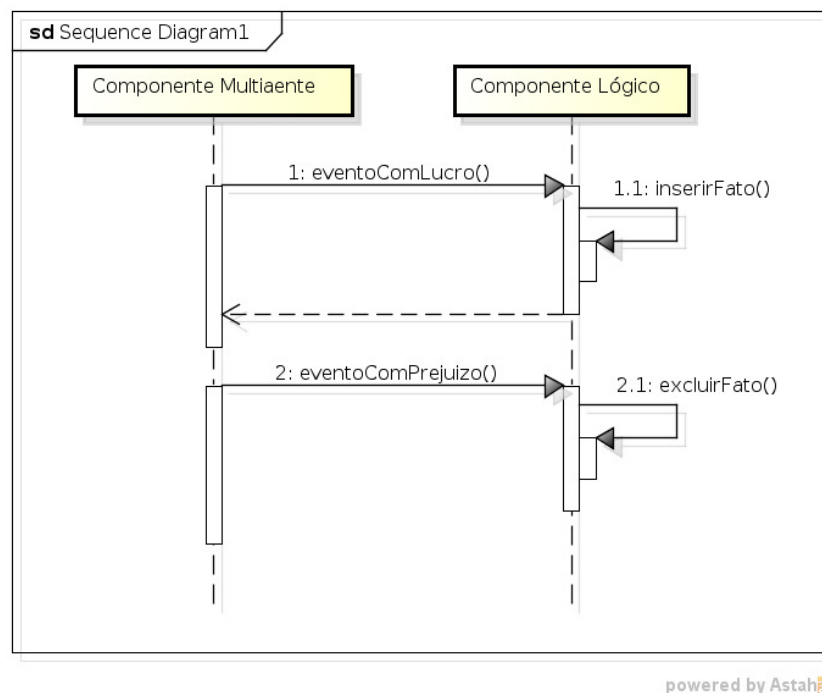


Figura 22 – Diagrama de Sequência do Componente Lógico InvestMVC.

#### 4.1.5 Componente MQL

O componente MQL será responsável por receber a resposta do Componente Multiagentes para realizar uma compra ou venda. Também será recebido outros atributos relacionados a compra ou venda, como alavancagem, stop loss e take profit.

A interação do Componente MQL com o Componente Multiagentes é evidenciada na Figura 23.

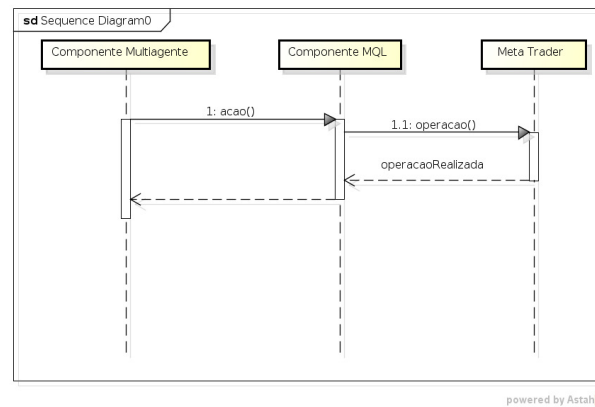


Figura 23 – Diagrama de Sequência do Componente MQL InvestMVC.

#### 4.1.6 Fluxo de atividades da ferramenta InvestMVC

O investidor interage apenas com o componente Orientado a Objetos, criando seu usuário e Experts, no qual serão persistidos. Além disso, o investidor também poderá ativar um Expert.

O componente Multiagentes vai verificar a tendência do Mercado de Moedas, por meio da plataforma MetaTrader. Sendo assim, o componente Multiagentes buscará na persistência o Expert que está ativo. Sabendo qual o Expert que foi ativado, o componente Multiagente faz a requisição de cálculos para os módulos C e Haskell. A partir desse resultado, o componente Multiagente procurará no Módulo Base de Conhecimento, a alavancagem (quanto deve arriscar) e os valores de entrada para o método de Correlação de Pearson, Fibonacci e Mínimos Quadrados. Caso todas as especificações para o módulo Multiagentes sejam obedecidas, ele informa ao componente MQL para realizar uma compra ou venda.

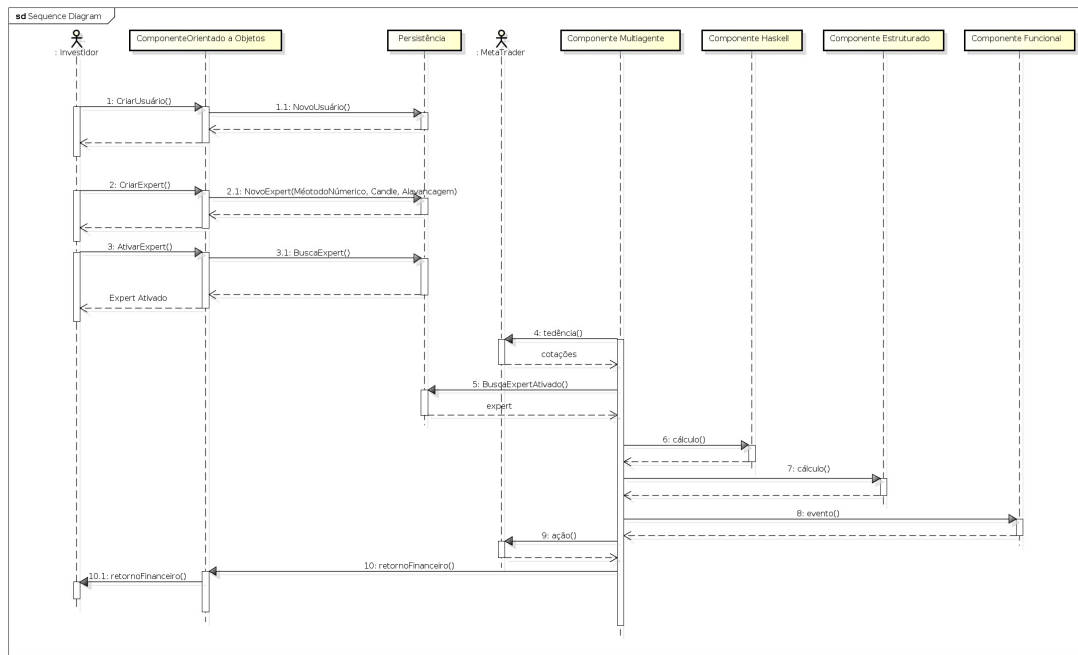


Figura 24 – Diagrama de Sequência InvestMVC



## 5 Resultados

Na tabela 1 pode ser visualizado as Histórias de Usuário, a pontuação e o estado (a fazer, em progresso ou feito) de cada História. Foram implementados 12 pontos do total de 48 pontos estimados. Sendo assim, já foi implementado 25% dos pontos da ferramenta InvestMVC.

Tabela 1 – Estado atual da ferramenta InvestMVC

Histórias de Usuário	Pontuação	Estado
US1 - Agente Correlação Linear	3	Feito
US2 - Agente Fibonacci	3	Feito
US3 - Agente Mínimos Quadrados	3	Feito
US4 - Agente Tendência	2	Feito
US5 - Agente Gestor/Consultor	2	Feito
US6 - Criar conta de usuário	2	Feito
US7 - Acompanhar retorno financeiro	5	A fazer
US8 - Criar Experts	2	Feito
US9 - Editar Experts	2	Feito
US10 - Excluir Experts	2	Feito
US11 - Ativar Expert	1	Feito
US12 - Desativar Expert	2	A fazer
US13 - Método Correlação Linear em linguagem C	2	Feito
US14 - Método Fibonacci em linguagem C	2	Feito
US15 - Método Mínimos Quadrados em linguagem C	2	Feito
US16 - Método Correlação Linear em linguagem Haskell	2	Feito
US17 - Método Fibonacci em linguagem Haskell	2	Feito
US18 - Método Mínimos Quadrados em linguagem Haskell	2	Feito
US19 - Inserir na Base de Conhecimento	2	Feito
US20 - Retirar na Base de Conhecimento	2	A fazer
US21 - Calcular Critério de Entrada	3	Feito
US22 - Realizar Operação no MetaTrader	1	A fazer

## 5.1 Resultados testes unitários

### 5.1.1 Componente Funcional

Foram realizados os testes unitários na linguagem haskell utilizando o framework HUnit. O framework não fornece a cobertura de código fonte, mas é possível visualizar a quantidade de casos de teste, quantidade de testes realizados, quantidade de erros e quantidade de falhas. Os resultados dos testes unitários dos métodos de Correlação de Pearson, Fibonacci e Mínimos Quadrados podem ser visualizados na Figura 25, 26 e 27.

```
*TesteCorrelacaoDePearson> main
Cases: 8   Tried: 8   Errors: 0   Failures: 0
Counts {cases = 8, tried = 8, errors = 0, failures = 0}
```

Figura 25 – Resultado da Suíte de Teste do Método Correlação Linear

```
*TesteFibonacci> main
Cases: 3   Tried: 3   Errors: 0   Failures: 0
Counts {cases = 3, tried = 3, errors = 0, failures = 0}
```

Figura 26 – Resultado da Suíte de Teste do Método de Fibonacci

```
*TesteMinimosQuadrados> main
Cases: 2   Tried: 2   Errors: 0   Failures: 0
Counts {cases = 2, tried = 2, errors = 0, failures = 0}
```

Figura 27 – Resultado da Suíte de Teste do Método Mínimos Quadrados

Os códigos referentes aos testes unitários em linguagem Haskell encontram-se no apêndice F.

### 5.1.2 Componente Estruturado

### 5.1.3 Componente Lógico

Foi utilizada a ferramenta Swi-prolog para realização dos testes unitários na linguagem prolog. Não foi possível obter a cobertura de código fonte da linguagem prolog utilizando a ferramenta Swi-prolog (os testes ficaram com cobertura fixa em 4.5). Entretanto, os testes foram realizados com sucesso, conforme pode ser visualizado nas Figuras 28 e 29.

```
?- show_coverage(run_tests).
% PL-Unit: aprendizMock ... done
% All 3 tests passed

=====
                        Coverage by File
=====
File                                Clauses    %Cov %Fail
=====
/usr/lib/swi-prolog/library/test_cover.pl    22      4.5  4.5
=====
true.
?- 
```

Figura 28 – Suíte de teste da base aprendiz.pl

```
?- show_coverage(run_tests).
% PL-Unit: baseConhecimentoMock ..... done
% All 12 tests passed

=====
                        Coverage by File
=====
File                                Clauses    %Cov %Fail
=====
/usr/lib/swi-prolog/library/test_cover.pl    22      4.5  4.5
=====
true.
?- 
```

Figura 29 – Suíte de teste da base de conhecimento

Os códigos referentes aos testes unitários em linguagem Prolog encontram-se no apêndice H.

#### 5.1.4 Componente Multiagentes

Foram utilizados os frameworks Junit e Easy-mock para realização dos testes unitários na linguagem Java. Para obter a cobertura de código fonte, foi utilizada a ferramenta EclEmma. Obteve-se uma cobertura de código fonte de 84,8% nos testes unitários, conforme pode ser visualizado na Figura 30.

▼ 📁 src	🟢 84,8 %	481	86
▶ 📁 comportamentosComuns	🟢 82,9 %	369	76
▶ 📁 metodosNumericos	🟢 86,5 %	64	10
▶ 📁 investidores	🟢 100,0 %	48	0

Figura 30 – Cobertura de Código dos pacotes do Componente Multiagentes

O comportamento dos agentes são executados através dos métodos actions e muitos comportamentos precisavam de um tempo de até 6 (seis) segundos para serem executados. Devido a isso, os testes unitários quebraram, pois nenhum comportamento era executado a tempo. Para tentar solucionar esse problema, foram colocados delays para que desse tempo dos agentes se comunicarem. A execução dos testes demorou mais de 18 (dezoito) segundos conforme pode ser visualizado no canto superior esquerdo da Figura 31. Apesar de todos os testes passarem pelo Junit e o Easy-mock com os delays, a ferramenta Eclemma não pegava a cobertura de código dos métodos actions e devido a isso esses métodos foram ignorados no percentual de cobertura de código fonte.

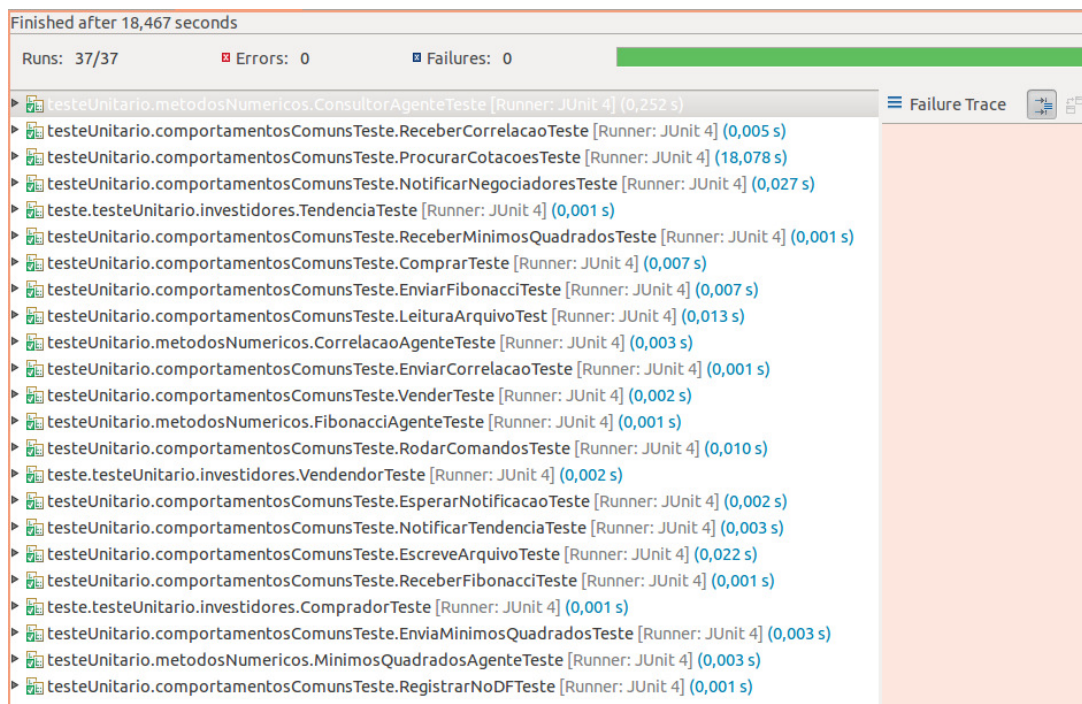


Figura 31 – Suite de teste do Componente Multiagentes

As classes de teste mais significativas encontram-se no apêndice I.



## 6 Conclusão

Através do Protocolo de Experimentação, os Métodos Matemáticos de Correlação Linear, Fibonacci e Mínimos Quadrados foram selecionados para serem implementados na ferramenta InvestMVC.

Cada Paradigma de Programação presente na ferramenta InvestMVC teve seu papel bem definido através da arquitetura guiada por componentes.

No Componente Estruturado já foram implementados e realizados testes unitários de todos os Métodos Matemáticos (Correlação Linear, Mínimos Quadrados e Fibonacci). Também foi realizada a análise estática de código fonte e a mesma evidencia que os códigos dos Métodos Matemáticos estão excelentes na maioria das métricas. No Componente Funcional foi implementado o método de Correlação Linear e seus respectivos testes unitários. No Componente Multiparadigma, ainda não foi finalizado nenhum agente, mas a estrutura desse Componentes e partes do código do mesmo já estão bem adiantadas. No Componente Lógico, ainda não foi implementado nenhuma linha de código. Mas, a ferramenta de teste unitário PIUnit já foi instalada e testada no ambiente da ferramenta InvestMVC. Por fim, o Componente MQL já foi totalmente implementado e já está em comunicação com a plataforma MetaTrader.

Conforme explanado no capítulo de Resultados, a ferramenta InvestMVC encontra-se 30,61% finalizada, pois 15 pontos foram concluídos dos 49 pontos definidos no backlog. Qualquer História de Usuário, só é considerada como pronta quando é realizado todos os testes unitários pertinentes. Acredita-se que o percentual de 30,61% em breve será aumentado, pois existem várias Histórias de Usuário que estão sendo testadas para serem consideradas finalizadas.

A mistura harmônica de Mercado de Moedas, Métodos Matemáticos, Paradigmas de Programação e Qualidade de Software está gerando a grande ferramenta InvestMVC. Espera-se no TCC 2, uma evolução bastante significativa da ferramenta para que este trabalho não se torne apenas um trabalho de consulta na biblioteca da UnB, mas sim um trabalho que será utilizado pela sociedade.



# Referências

ADVFN. Technical analysis. 2013. Disponível em: <<http://advfn.com/educacional/analise-tecnica>>. Citado na página 13.

AGENTBUILDER. Auction agents for the electric power industry. 2009. Disponível em: <<http://www.agentbuilder.com/Documentation/EPRI/index.html>>. Citado na página 23.

AGENTBUILDER. Mercado forex: Série alertas. 2009. Disponível em: <<http://www.cvm.gov.br/port/Alertas/mercadoForex.pdf>>. Citado 2 vezes nas páginas 22 e 38.

ALBUQUERQUE, A. A. *Alavancagem Financeira e Investimento*. Monografia (Especialização) — Faculdade de Administração, Universidade de São Paulo, São Paulo, 2013. Citado na página 6.

ALMEIDA, R. J. de A. Automatizando a criação de uma base de conhecimento em prolog para gerenciar os acessos a um site. 2010. Disponível em: <<http://www.vivaolinux.com.br/artigo/Automatizando-a-criacao-de-uma-base-de-conhecimento-em-Prolog-para-gerenciar-os-acessos-a-um>>. Citado 2 vezes nas páginas 24 e 39.

ALVES., T. *Um Passeio na Sequência de Fibonacci*. Monografia (Trabalho de Conclusão de Curso em Matemática) — Universidade Estadual da Paraíba, Paraíba, 2012. Citado na página 12.

AMBLER, S. *Uma Visão Realística da Reutilização em Orientação a Objetos*. [S.l.], 1998. Citado na página 20.

APT, K. R. *From Logic programming to Prolog*. 1. ed. [S.l.], 1996. Citado na página 23.

BANDEIRA, M. *Tipos de Pesquisa*. Monografia (Artigo) — Faculdade de Psicologia, FUNREI, Minas Gerais, 2012. Nenhuma citação no texto.

BEIZER, B. *Software Testing Techniques*. 2. ed. [S.l.], 1990. Citado 2 vezes nas páginas 25 e 26.

BELCHIOR, G. P. *Oficina de Metodologia Científica: Elaboração de Projetos de Pesquisa*. [S.l.], 2012. Citado na página 5.

BRADSHAW, J. M. *Software Agents*. [S.l.], 1997. Citado na página 21.

BRAGA, R. Qualidade de software: Avaliação de sistemas computacionais. 2012. Disponível em: <[http://disciplinas.stoa.usp.br/pluginfile.php/57546/mod\\_resource/content/1/Aula8-QualidadeSoftware.pdf](http://disciplinas.stoa.usp.br/pluginfile.php/57546/mod_resource/content/1/Aula8-QualidadeSoftware.pdf)>. Citado na página 29.

BROOKSHEAR, J. G. *Ciência da Computação: Uma Visão Abrangente*. 3. ed. [S.l.], 2003. Citado na página 17.

- BRUNI, A. L.; FAMÁ, R. *Gestão de custos e formação de preços*. São Paulo, 2011. Citado na página 6.
- BUENO, C. S.; CAMPELO, G. B. *Qualidade de Software*. Monografia (Artigo) — Departamento de Informática, Universidade Federal de Pernambuco, Pernambuco, 2011. Citado na página 29.
- CAPRETZ, L. F. *A Brief History of the Object-Oriented Approach*. [S.l.], 2003. Citado 2 vezes nas páginas 18 e 19.
- CHEESMAN, J.; DANIELS, J. *UML Components*. [S.l.], 2001. Citado na página 33.
- CHESS, B.; WEST, J. *Secure Programming with Static Analysis*. [S.l.], 2007. Citado na página 29.
- COENEN, F. Tópicos de tratamento de informação: Linguagens declarativas. 1999. Disponível em: <<http://cgi.csc.liv.ac.uk/~frans/OldLectures/2CS24/declarative.html#detail>>. Citado na página 23.
- COLLINS, V. et al. Support and resistance. 2012. Disponível em: <<http://www.markets.com/pt/education/technical-analysis/support-resistance.html>>. Citado 2 vezes nas páginas 7 e 8.
- COSTA, S. L. *Uma Ferramenta e Técnica para o Projeto Global e Detalhado de Sistemas Multiagente*. Monografia (Mestrado em Engenharia de Eletricidade) — Departamento de Engenharia Elétrica, Universidade Federal do Maranhão, Maranhão, 2004. Citado na página 21.
- CVM. Mercado forex: Série alertas. 2009. Disponível em: <<http://www.cvm.gov.br/port/Alertas/mercadoForex.pdf>>. Citado 2 vezes nas páginas 5 e 6.
- DANTAS, J. A.; MEDEIROS, O. R.; LUSTOSA, P. R. B. Reação do mercado à alavancagem operacional. 2006. Disponível em: <<http://www.scielo.br/pdf/rf/v17n41/v17n41a06.pdf>>. Citado na página 6.
- DEBASTINI, C. A. *Análise técnica de ações: identificando oportunidades de compra e venda*. 1. ed. [S.l.], 2008. Citado na página 7.
- DEVORE, J. L. *Probabilidade e Estatística para Engenharia e Ciências*. 6. ed. [S.l.], 2006. Citado 2 vezes nas páginas 9 e 10.
- DIAS, M. A. P. *Administração de materiais: uma abordagem logística*. 2. ed. [S.l.], 1985. Citado na página 9.
- EASYFOREX. Leveraged forex trading: What is leverage in forex trading? 2014. Disponível em: <<http://www.easy-forex.com/int/leveragedtrading/>>. Citado na página 6.
- FERREIRA, A. B. de H. *Novo Dicionário da Língua Portuguesa*. 2. ed. [S.l.], 2008. Citado na página 15.
- FERREIRA, D. F. *Estatística básica*. 2. ed. [S.l.], 2009. Citado na página 10.

FILHO, C. *Kalibro: interpretação de métricas de código-fonte*. Monografia (Mestrado em Ciência da Computação) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2013. Citado na página 30.

FORCON. Metodologias para monografia. 2014. Disponível em: <<http://www.fazermonografia.com.br/metodologia-para-monografia>>. Nenhuma citação no texto.

FXCM. Forex basic. 2011. Disponível em: <<http://www.fxcm.com/forex-basics/>>. Citado na página 5.

GAGLIARDI, J. D. *Fundamentos de Matemática*. Monografia (Monografia) — Universidade Federal de Campinas, São Paulo, 2013. Citado na página 12.

GIL, A. C. *Como elaborar projetos de pesquisa*. 4. ed. [S.l.], 2008. Nenhuma citação no texto.

GIRARDI, R. *Engenharia de Software baseada em Agentes*. [S.l.], 2004. Citado 2 vezes nas páginas 21 e 22.

HOOGLÉ. Functional programming. 2013. Disponível em: <[http://www.haskell.org/haskellwiki/Functional\\_programming](http://www.haskell.org/haskellwiki/Functional_programming)>. Citado 2 vezes nas páginas 23 e 35.

HOPPEN, C. *Metodologia Científica: Pesquisa Experimental*. Monografia (Graduação) — Universidade Federal do Paraná, 2010. Nenhuma citação no texto.

IEEE 610. *IEEE Standard Glossary of Software Engineering Terminology*. [S.l.], 1990. Citado 4 vezes nas páginas 25, 26, 27 e 28.

INVESTFOREX. Análise técnica para mercado forex. 2014. Disponível em: <<http://www.investforex.pt/aprender-forex/analise-tecnica>>. Citado 2 vezes nas páginas 14 e 15.

INÁCIO, J. F. S. *Análise do Estimador de Estado por Mínimos Quadrados Ponderados*. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, 2010. Citado na página 8.

JENNINGS, N. R.; SYCARA, K.; WOOLDRIDGE, M. *A Roadmap of Agent Research and Development*. [S.l.], 1998. Citado na página 22.

JENNINGS, N. R.; WOOLDRIDGE, M. *Intelligent agents: theory and practice*. [S.l.], 1995. Citado na página 22.

JUNGTHON, G.; GOULART, C. M. *Paradigmas de Programação*. Monografia (Monografia) — Faculdade de Informática de Taquara, Rio Grande do Sul, 2009. Citado na página 37.

KONIS, K. Mathematical methods for quantitative finance. 2014. Disponível em: <<https://www.coursera.org/course/mathematicalmethods>>. Citado na página 8.

LAMIM, J. Mvc - o padrão de arquitetura de software. 2010. Disponível em: <[http://www.oficinadanet.com.br/artigo/1687/mvc\\_-\\_o\\_padrao\\_de\\_arquitetura\\_de\\_software](http://www.oficinadanet.com.br/artigo/1687/mvc_-_o_padrao_de_arquitetura_de_software)>. Citado na página 34.

- LEAVENS, G. T. Major programming paradigms. 2014. Disponível em: <<http://www.eecs.ucf.edu/~leavens/ComS541Fall97/hw-pages/paradigms/major.html#object>>. Citado na página 20.
- LEWIS, W. E. *Software Testing and Continuous Quality Improvement*. 3. ed. [S.l.], 2009. Citado na página 28.
- LIRA, S. A. *Análise Correlação: Abordagem Teórica e de Construção dos Coeficientes com Aplicações*. Monografia (Dissertação Pós-Graduação) — Universidade Federal do Paraná, Paraná, 2004. Citado 2 vezes nas páginas 10 e 12.
- LOPES, F. D. *Caderno didático: estatística geral*. [S.l.], 2005. Citado 2 vezes nas páginas 10 e 11.
- MARKET. About what is forex. 2011. Disponível em: <<http://www.markets.com/pt/education/forex-education/what-is-forex.html>>. Citado na página 5.
- MATSURA, E. *Comprar ou Vender? Como investir na Bolsa Utilizando Análise Gráfica*. 7. ed. [S.l.], 2006. Citado 3 vezes nas páginas 6, 7 e 8.
- MEIRELLES, P. *Monitoramento de métricas de código-fonte em projetos de Software Livre*. Monografia (Tese de Doutorado) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2013. Citado 2 vezes nas páginas 29 e 30.
- MELO, J. R. F. *Análise Estática de Código*. Monografia (Monografia) — Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, Natal, 2011. Citado na página 30.
- MILLANI, L. F. G. *Análise de Correlação entre Métricas de Qualidade de Software e Métricas Físicas*. Monografia (Monografia) — Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, 2013. Citado na página 30.
- MYERS, G. J. *The art of Software Testing*. 2. ed. [S.l.], 2004. Citado 3 vezes nas páginas 25, 26 e 28.
- NAIK, K.; TRIPATHY, P. *SOFTWARE TESTING AND QUALITY ASSURANCE Theory and Practice*. 2. ed. [S.l.], 2008. Citado 3 vezes nas páginas 26, 27 e 28.
- NETO, A. C. D. Introdução a teste de software. 2005. Disponível em: <<http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035>>. Citado na página 25.
- NORMAK, K. Programming paradigms. 2013. Disponível em: <[http://people.cs.aau.dk/~normark/prog3-03/html/notes/paradigms\\_themes-paradigms.html#paradigms\\_the-word\\_title\\_1](http://people.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigms.html#paradigms_the-word_title_1)>. Citado na página 15.
- ODELL, J.; GIORGINI, P.; MÜLLER, J. P. *Agent-Oriented Software Engineering V*. [S.l.], 2005. Citado na página 22.
- PAQUET, J.; MOKHOV, S. *Comparative Studies of Programming Languages*. [S.l.], 2010. Citado 2 vezes nas páginas 16 e 24.
- PIPONI, D. Eleven reasons to use haskell as a mathematician. 2006. Disponível em: <<http://blog.sigfpe.com/2006/01/eleven-reasons-to-use-haskell-as.html>>. Citado 2 vezes nas páginas 24 e 35.

- REGRA, C. M. *Tese de Mestrado em Estatística Computacional*. Monografia (Monografia) — Universidade Aberta, 2010. Citado na página 11.
- RILEY, F.; HOBSON, M. P.; BENCE, S. J. *Mathematical Methods for Physics and Engineering: A Comprehensive Guide*. [S.l.], 2011. Citado na página 8.
- RITCHIE, D. M. O desenvolvimento da linguagem c. 1996. Disponível em: <<http://cm.bell-labs.com/cm/cs/who/dmr/chistPT.html>>. Citado na página 18.
- ROBOFOREX. Indicators used in forex market. 2013. Disponível em: <<http://www.roboforex.pt/beginner/start/technical-analysis>>. Citado na página 15.
- ROCHA, R. A. *Algumas Evidências Computacionais da Infinitude dos Números Primos de Fibonacci*. Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) — Universidade Federal do Rio Grande do Norte, Natal, 2008. Citado 2 vezes nas páginas 12 e 13.
- RODRIGUES, W. C. *Metodologia Científica*. [S.l.], 2007. Nenhuma citação no texto.
- RUSSEL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3. ed. [S.l.], 1995. Citado na página 21.
- SCHUMACHER, M. *Objective Coordination in Multi-Agent System Engineering: design and implementation*. 1. ed. [S.l.], 2001. Citado na página 21.
- SCHWABER, K.; SUTHERLAND, J. The definitive guide to scrum: The rules of the game. 2013. Disponível em: <[https://www.scrum.org/portals/0/documents/scrum%20guides/scrum\\_guide.pdf](https://www.scrum.org/portals/0/documents/scrum%20guides/scrum_guide.pdf)>. Nenhuma citação no texto.
- SEBESTA, R. W. *Concepts of programming languages*. 10. ed. [S.l.], 2012. Citado 3 vezes nas páginas 16, 17 e 24.
- SINGH, C. Difference between method overloading and overriding in java. 2014. Disponível em: <<http://beginnersbook.com/2014/01/difference-between-method-overloading-and-overriding-in-java/>>. Citado na página 20.
- SINGH, C. Method overloading in java with examples. 2014. Disponível em: <<http://beginnersbook.com/2013/05/method-overloading/>>. Citado na página 20.
- SOMMERVILLE, I. *Engenharia de Software*. 9. ed. [S.l.], 2011. Citado 2 vezes nas páginas 29 e 30.
- SPILLNER, A.; LINZ tilo; SCHAEFER, H. *Software Testing Foundations*. 4. ed. [S.l.], 2014. Citado na página 28.
- SPIVEY, M. *An introduction to logic programming through Prolog*. 1. ed. [S.l.], 1996. Citado na página 24.
- STEFANOV, S. *Object-Oriented JavaScript*. [S.l.], 2008. Citado na página 18.
- THIAGO, A. As vantagens do teste unitário. 2001. Disponível em: <<http://andrethiago.wordpress.com/2011/04/06/as-vantagens-do-teste-unitario/>>. Citado na página 27.

- THOMPSON, S. *Haskell: The Craft of Functional Programming*. 2. ed. [S.l.], 1999. Citado na página 23.
- TUCKER, A. B.; NOONAN, R. E. *Linguagens de programação : Princípios e paradigmas*. 2. ed. [S.l.], 2009. Citado 4 vezes nas páginas 17, 19, 23 e 24.
- VENNERS, B. Polymorphism and interfaces. 1996. Disponível em: <<<http://www.artima.com/objectsandjava/webuscript/PolymorphismInterfaces1.html>>>. Citado na página 20.
- VIALI, L. M. *Estatística Básica*. Monografia (Monografia) — Instituto de Matemática da Universidade Federal do Rio Grande do Sul, 2009. Citado na página 11.
- VUOLO, J. H. *Fundamentos da Teoria de Erros*. 2. ed. [S.l.], 1996. Citado na página 9.
- WEGNER, P. *Concepts and paradigms of object-oriented programming*. 1. ed. [S.l.], 1990. Citado na página 19.
- WEISFELD, M. A. *The Object-Oriented Thought Process*. 3. ed. [S.l.], 2009. Citado na página 19.
- WILLIAMS, L. *White-Box Testing*. [S.l.], 2006. Citado 3 vezes nas páginas 26, 27 e 28.
- WOLFRAM. Built with mathematica technology: Movingaverage. 2012. Disponível em: <<http://mathworld.wolfram.com/MovingAverage.html>>. Citado 2 vezes nas páginas 14 e 15.
- WOOLDRIDGE, M. *Teamwork in Multi-agent systems: A Formal Approach*. 1. ed. [S.l.], 2010. Citado 2 vezes nas páginas 21 e 22.



## 7 Glossário

**Cotação** - valor monetário do par de moedas em um determinado período de tempo.

**Corretagem** - Cobrança em valor monetário por alguma operação de compra ou venda.

**Corretora** - Broker que intermedia a compra ou venda no Mercado de Moedas

**CSV** - Extensão que permite manipular arquivos em planilhas eletrônicas.

**FGA** - Faculdade do Gama

**GPL3** - permite que os programas sejam distribuídos e reaproveitados

**LGPL3** - permite que os programas sejam distribuídos e reaproveitados com outras licenças que não sejam GPL ou LGPL.

**MQL4** - Linguagem de programação em paradigma estruturado que permite implementar experts.

**MQL5** - Linguagem de programação em paradigma orientado a objetos que permite implementar experts.

**Stop loss** - define a quantidade de pontos que o investidor deseje perder.

**Take profit** - define a quantidade de pontos que o investidor deseje ganhar.

**Tendência** - revela a direção em que o mercado está indo.

**USD** - dólares americanos

**UnB** - Universidade de Brasília



## Apêndices



## APÊNDICE A – Suporte Tecnológico

## 1. Ferramentas para teste unitário e teste de integração

XUnit<sup>1</sup> é um framework para construção de testes unitários e de integração. Nesse capítulo, serão apresentados frameworks que se basearam no XUnit para serem construídos.

Esta seção descreve os frameworks CUnit, JUnit, HUnit e PIUnit que respectivamente auxiliam na criação de testes em linguagem C, Java, Haskell e Prolog.

### CUnit

CUnit<sup>2</sup> é um framework para escrita e execução de testes automatizados em linguagem C e C++. O framework usa uma estrutura simples para a construção de estruturas de teste e fornece um rico conjunto de afirmações para testar tipos de dados comuns. Além disso, várias interfaces diferentes são fornecidas para a execução de testes e comunicação de resultados. Essas interfaces atualmente incluem saídas automatizadas para arquivo xml não interativas, console de interface (ansi C) interativa e interface gráfica Curses (Unix) interativa.

### JUnit

JUnit<sup>3</sup> é um framework para criação de testes automatizados na linguagem de programação Java. O framework facilita a criação de código para a automação de testes com apresentação dos resultados, verificando se cada método de uma classe funciona da forma esperada. Os resultados são exibidos via interface, sendo que os erros aparecem em cor vermelha, as falhas cor azul e os testes aceitáveis em cor verde.

### HUnit

HUnit<sup>4</sup> é um framework para criação de testes automatizados em linguagem Haskell. Os testes são executados via terminal. Após executar os testes é possível visualizar no terminal, os resultados da quantidade de testes com erros ou falhas.

### PIUnit

---

<sup>1</sup> <https://xunit.codeplex.com>

<sup>2</sup> <http://cunit.sourceforge.net/>

<sup>3</sup> <http://junit.org/index.html>

<sup>4</sup> <http://hunit.sourceforge.net/>

PIUnit<sup>5</sup> é um framework para criação de testes automatizados em linguagem Prolog. Os testes são executados via terminal usando o suporte swi<sup>6</sup>. Ao executar os testes, os erros e falhas são mostrados via terminal.

## **2. Ferramenta para teste funcional: Cucumber**

Cucumber<sup>7</sup> permite que as equipes de desenvolvimento de software descrevam como o software deve se comportar com apoio de textos simples. O texto é escrito em uma linguagem específica de domínio e com base nesse texto, é construído o teste funcional da aplicação.

## **3. Ferramentas de cobertura de teste**

Esta seção descreve as ferramentas de cobertura de teste Eclemma (linguagem Java) e HPC (linguagem Haskell). O Cunit e o PIUnit já fornecem a cobertura de código e portanto não é necessário instalar nenhum plugin adicional.

### **Eclemma**

Eclemma<sup>8</sup> é uma ferramenta gratuita para fazer cobertura de código Java na IDE Eclipse. Esta ferramenta não exige qualquer alteração no projeto a ser inspecionado, fornecendo um resultado rápido no próprio editor de texto.

### **HPC**

HPC<sup>9</sup> é um tool-kit para exibir e armazenar o cobertura de código fonte de programas Haskell.

## **4. Ferramenta de Análise Estática de Código Fonte**

Esta seção descreve quais foram as ferramentas selecionadas para análise estática de código fonte.

### **Analizo**

Analizo<sup>10</sup> é ferramenta de análise estática de código fonte que roda projetos em linguagens C, C++ e Java. A ferramenta roda sistema operacional Linux, fornece 20 métricas e possui licença GPL3.

---

<sup>5</sup> <http://www.swi-prolog.org/pldoc/package/plunit.html>

<sup>6</sup> [http://www.swi-prolog.org/pldoc/doc\\_for?object=manual](http://www.swi-prolog.org/pldoc/doc_for?object=manual)

<sup>7</sup> <http://cukes.info/>

<sup>8</sup> <http://www.eclemma.org/>

<sup>9</sup> [https://www.haskell.org/haskellwiki/Haskell\\_program\\_coverage#Hpc\\_tools](https://www.haskell.org/haskellwiki/Haskell_program_coverage#Hpc_tools)

<sup>10</sup> <http://www.analizo.org/>

## **Sonar**

Sonar <sup>11</sup> é uma ferramenta de análise estática de código fonte que roda projetos em mais de 20 linguagens, incluindo C, C++, Java, PHP, Groovy, entre outras. A ferramenta roda nos sistemas operacionais Windows, Linux e Mac OS X. A licença de uso é a LGPL3.

## **5. Ferramentas de Mercado de Moedas**

Esta seção descreve as ferramentas de Mercado de Moedas MetaTrader, MetaEditor, Alpari-UK e FXDD.

### **MetaTrade**

MetaTrader<sup>12</sup> é uma plataforma de negociação eletrônica com capacidade de negociações automatizadas. É possível programar experts em linguagem mql4 <sup>13</sup> (paradigma estruturado) e linguagem mql5<sup>14</sup> (paradigma orientado a objetos).

### **MetaEditor**

MetaEditor<sup>15</sup> é uma IDE para linguagem MQL4 e MQL5. É possível editar e compilar experts para operar de forma automatizada no Mercado de Moedas através de uma corretora.

### **Alpari-UK**

Alpari-UK<sup>16</sup> é uma corretora com sede oficial na Inglaterra. Possui tecnologia de negociação que inclui a plataforma MetaTrader. Através da Alpari-UK é possível comprar ou vender no Mercado de Moedas, pois a mesma intermedia as negociações através da cobrança de uma corretagem.

### **FXDD**

FXDD<sup>17</sup> é uma corretora com sede oficial nos Estados Unidos e possui as mesmas características tecnológicas que a Alpari-UK. Inclui as tecnologias da plataforma MetaTrader e intermedia as negociações através da cobrança de uma corretagem para o investidor.

---

<sup>11</sup> <http://www.sonarqube.org/>

<sup>12</sup> <http://www.metaquotes.net/>

<sup>13</sup> <http://www.mql4.com/>

<sup>14</sup> <http://www.mql5.com/>

<sup>15</sup> <http://book.mql4.com/metaeditor/index>

<sup>16</sup> <http://www.alpari.co.uk/>

<sup>17</sup> <http://www.fxdd.com/>



## **6. Editores de texto**

Esta seção descreve os editores de texto selecionados para implementação da ferramenta InvestMVC.

### **Eclipse**

Eclipse<sup>18</sup> é um IDE para desenvolvimento em linguagem Java. Com uso de plugins, é possível programar em outras linguagens como C/C++, PHP e Python.

### **Sublime**

Sublime<sup>19</sup> é um editor de texto que suporta diversas linguagens como C, C++, Java, Groovy, entre outras.

## **7. Ferramenta organizacional**

### **ScrumMe**

ScrumMe<sup>20</sup> é uma ferramenta on line para criação, controle e acompanhamento de projetos. É possível criar Sprints, atividades dentro de cada Sprint e colocar o responsável por cada atividade. Também é possível obter relatórios de acompanhamento do projeto.

---

<sup>18</sup> <https://www.eclipse.org>

<sup>19</sup> <http://www.sublimetext.com/>

<sup>20</sup> <http://www.scrumme.com.br>



## APÊNDICE B – Histórias de Usuário

### **US1 - Agente Correlação Linear**

Como agente de software, gostaria de usar o método de Correlação de Pearson para verificar sua eficácia no Mercado de Moedas.

Critérios de aceitação US1:

1. A Correlação de Pearson deve ser maior que 0.8 e o mercado estar subindo para se realizar a venda.
2. A Correlação de Pearson deve ser maior que 0.8 e o mercado estar caindo para se realizar a compra.
3. O take profit e o stop loss devem ser os mesmos.

### **US2 - Agente Fibonacci**

Como agente de software, gostaria de usar o método de Fibonacci para verificar sua eficácia no Mercado de Moedas.

Critérios de aceitação US2:

1. Na situação em que o mercado estiver caindo e a regressão de Fibonacci estiver em 0.62, deve-se apenas vender no mercado com take profit e stop loss em 500 pontos.
2. Na situação em que o mercado estiver subindo e a regressão de Fibonacci estiver em 0.62, deve-se apenas comprar no mercado com take profit e stop loss em 500 pontos.

### **US3 - Agente Mínimos Quadrados**

Como agente de software, gostaria de usar o método de Mínimos Quadrados para verificar sua eficácia no Mercado de Moedas.

Critérios de aceitação US3:

1. Não deve existir stop loss ou take profit fixos, pois os mesmos são variáveis de acordo com a estratégia do método de Mínimos Quadrados.
2. O método deve seguir a tendência do mercado. Se o mercado estiver subindo, deve-se comprar e se estiver caindo, deve-se vender.

#### **US4: Agente Tendência**

Como agente de software, gostaria de verificar a tendência do mercado para estimar um ponto de compra ou venda.

Critérios de aceitação US4:

1. As cotações obtidas na persistência não podem ser nulas.
2. A tendência deve ser calculada de acordo com o gráfico escolhido como 1 minuto, 5 minutos e assim por diante.

#### **US5: Agente Gestor/Consultor**

Como agente de software, gostaria de informar ao componente mql uma operação para que seja possível realizar uma compra ou venda.

Critérios de aceitação US5:

1. Só deve ser informado ao componente mql 1 para compra, -1 para venda ou 0 para fazer nada.
2. Enquanto o componente mql estiver com uma ordem de compra ou venda aberta, não é necessário informar a operação que deve ser feita ao componente.

#### **US6 - Criar conta de usuário**

Como investidor, gostaria de criar uma conta na ferramenta InvestMVC para que eu possa realizar simulações e/ou investimentos reais.

Critério de aceitação US6:

1. O usuário deve cadastrar um login que não esteja presente no banco de dados.
2. O usuário deve cadastrar um senha com 6 dígitos.

#### **US7 - Acompanhar retorno financeiro**

Como investidor, gostaria de visualizar o status da minha conta de investidor, para saber o lucro/prejuízo do expert.

Critérios de aceitação US7:

1. O retorno é um número real maior que 0.
2. O tempo de resposta para visualização do saldo deve ser inferior a 2 segundos.
3. Para acompanhar o retorno financeiro, o usuário deve estar logado na ferramenta InvestMVC.

### **US8 - Criar Experts**

Como investir, gostaria de criar um expert na ferramenta investMVC para que eu possa investir no Mercado de Moedas.

Critérios de aceitação US8:

1. Os experts disponíveis devem ser Mínimos Quadrados, Fibonacci e Correlação de Pearson.
2. Não se pode criar um expert sem ter selecionado ao menos um Método Matemático.
3. Para criar um expert, o usuário deve estar logado na ferramenta InvestMVC.

### **US9 - Editar Experts**

Como investir, gostaria de editar um expert na ferramenta investMVC para que eu possa investir no Mercado de Moedas.

Critérios de aceitação US9:

1. Os experts disponíveis devem ser Mínimos Quadrados, Fibonacci e Correlação de Pearson.
2. Não se pode editar um expert sem ter selecionado ao menos um Método Matemático para edição.
3. Para editar um expert, o usuário deve estar logado na ferramenta InvestMVC.

### **US10 - Excluir Experts**

Como investir, gostaria de excluir um expert na ferramenta investMVC para que não mais visualizar o mesmo para investir no Mercado de Moedas.

Critérios de aceitação US10:

1. Os experts disponíveis devem ser Mínimos Quadrados, Fibonacci e Correlação de Pearson.
2. Não se pode excluir um expert sem ter selecionado ao menos um Método Matemático para exclusão.
3. Para excluir um expert, o usuário deve estar logado na ferramenta InvestMVC.

### **US11 - Ativar Expert**

Como investidor, gostaria de ativar um Expert para que o mesmo esteja disponível para operar de forma automatizada.

Critérios de aceitação US11:

1. Só pode ser ativado um expert que acabou de ser criado ou que está desativado.
2. Para ativar um expert, o usuário deve estar logado na ferramenta InvestMVC.

### **US12 - Desativar Expert**

Como investidor, gostaria de desativar um Expert para que o mesmo esteja disponível para operar de forma automatizada.

Critérios de aceitação US12:

1. Só pode ser desativado um expert que esteja ativado.
2. Para desativar um expert, o usuário deve estar logado na ferramenta InvestMVC.

### **US13 - Método Correlação Linear em linguagem C**

Como investidor, gostaria de usar o método de Correlação de Pearson para operar no Mercado de Moedas.

Critérios de aceitação US13:

1. A Correlação de Pearson deve ser um número real menor que 1 e maior que -1.
2. Caso, as cotações lidas da persistência sejam nulas, o cálculo é abortado.

#### **US14 - Método de Fibonacci em linguagem C**

Como investidor, gostaria de usar o método de Fibonacci para operar no Mercado de Moedas.

Critérios de aceitação US14:

1. As regressões de Fibonacci devem ser maior que 0 e menor que 1.
2. Caso, as cotações lidas da persistência sejam nulas, o cálculo é abortado.

#### **US15- Método de Mínimos Quadrados em linguagem C**

Como investidor, gostaria de usar o método de Mínimos Quadrados para operar no Mercado de Moedas.

Critérios de aceitação US15:

1. O coeficiente angular e linear da reta de Mínimos Quadrados são números reais menores que 100.
2. Caso, as cotações lidas da persistência sejam nulas, o cálculo é abortado.

#### **US16- Método Correlação Linear em linguagem Haskell**

Como investidor, gostaria de usar o método de Correlação de Pearson para operar no Mercado de Moedas.

Critérios de aceitação US16:

1. A Correlação de Pearson deve ser um número real menor que 1 e maior que -1.
2. Caso, as cotações lidas da persistência sejam nulas, o cálculo é abortado.

#### **US17 - Método de Fibonacci em linguagem Haskell**

Como investidor, gostaria de usar o método de Fibonacci para operar no Mercado de Moedas.



Critérios de aceitação US17:

1. As regressões de Fibonacci devem ser maior que 0 e menor que 1.
2. Caso, as cotações lidas da persistência sejam nulas, o cálculo é abortado.

### **US18 - Método de Mínimos Quadrados em linguagem Haskell**

Como investidor, gostaria de usar o método de Mínimos Quadrados para operar no Mercado de Moedas.

Critérios de aceitação US18:

1. O coeficiente angular e linear da reta de Mínimos Quadrados são números reais menores que 100.
2. Caso, as cotações lidas da persistência sejam nulas, o cálculo é abortado.

### **US19 - Inserir na Base de Conhecimento**

Como agente de software, gostaria de enviar o retorno do Método Matemático de Correlação de Pearson, Mínimos Quadrados ou Fibonacci para que possa ser inserido na Base de Conhecimento.

Critérios de aceitação US19:

1. O valor a ser inserido na Base de Conhecimento é um número real.
2. Não devem ser inseridos valores nulos na Base de Conhecimento.
3. Caso, seja inserido um valor já existente na Base de Conhecimento, esse valor ganhar um peso a mais na base.

### **US20 - Retirar na Base de Conhecimento**

Como agente de software, gostaria de enviar o retorno do Método Matemático de Correlação de Pearson, Mínimos Quadrados ou Fibonacci para que possa ser retirado na Base de Conhecimento.

Critérios de aceitação US20:

1. Após o valor ser retirado, deve-se verificar se o mesmo não existe na Base de Conhecimento ou perdeu um peso caso tenha mais de um valor na base.
2. Deve ser verificado se o valor existe antes do mesmo ser retirado da Base de Conhecimento.

### **US21 - Calcular Critério de Entrada**

Como agente de software, gostaria de solicitar o retorno do Método Matemático de Correlação de Pearson, Mínimos Quadrados ou Fibonacci da Base de Conhecimento para que seja possível operar no Mercado de Moedas.

Critérios de aceitação US21:

1. O critério de entrada deve variar de acordo com a estratégia escolhida.
2. Não devem ser fornecidos critérios de entrada nulos.

### **US22 - Realizar Operação no MetaTrader**

Como investidor, gostaria de realizar uma operação na plataforma MetaTrader para que eu possa realizar uma compra ou venda.

Critérios de aceitação US22:

1. Deve ser definido uma alavancagem maior que zero para compra ou venda
2. Deve ser apresentado um take profit maior que zero
3. Deve ser apresentado um stop loss maior que zero.

## APÊNDICE C – Cronograma InvestMVC

ScrumMe (beta)  
Project:  
TCC Invest MVC<sup>2</sup>

Sprint/Story/Task	Team	Finalized	TD	IP	VR	DN
-------------------	------	-----------	----	----	----	----

Sprint: Sprint1

Construir Introdução

Definir Problema de Pesquisa	Vanessa	10/28/2014				
Definir Metodologia Parcial	Vanessa	10/28/2014				
Definir Contexto	Cleiton	10/28/2014				
Definir Justificativa	Vanessa	10/28/2014				
Definir Objetivos	Cleiton	10/28/2014				

Implementar Métodos Numéricos

Implementar e testar mínimos quadrados	Cleiton	9/6/2014				
Implementar e testar Fibonacci	Cleiton	9/6/2014				
Implementar e testar Pearson	Vanessa	9/6/2014				

Sprint: Sprint2

Construir Referencial Teórico Paradigmas de Programação

Definir paradigma MultiAgente	Vanessa	10/28/2014	■
Definir paradigma Lógico	Vanessa	10/28/2014	■
Definir paradigma Funcional	Vanessa	10/28/2014	■
Definir paradigma Estruturado	Vanessa	10/28/2014	■
<b>Adaptar Métodos Numéricos</b>			
Criar CSV dinâmico	Cleiton	10/28/2014	■
Realizar Comunicação com MetaTrader	Cleiton	10/28/2014	■
Realizar Comunicação com Grails	Vanessa	10/28/2014	■
<b>Protótipo View Projeto</b>			
Implementar View	Vanessa	10/28/2014	■
Estudar Kickstart	Vanessa	9/11/2014	■
<b>Construir Referencial Teórico Métodos Numéricos</b>			
Definir Correlação Linear	Cleiton	10/28/2014	■
Definir Mínimos Quadrados	Cleiton	10/28/2014	■
Definir Fibonacci	Cleiton	10/28/2014	■
<b>Sprint: Sprint3</b>			
<b>Refinar Referencial Teórico Paradigmas</b>			
Definir paradigma Estruturado	Vanessa	10/31/2014	■

Definir paradigma Funcional	Vanessa	10/28/2014	■
Definir paradigma MultiAgente	Vanessa	10/28/2014	■
Definir paradigma Lógico	Vanessa	10/28/2014	■

#### Construir Referencial Teórico de Contexto Financeiro

Definir Alavancagem	Cleiton	10/28/2014	■
Definir Suporte e Resistência	Cleiton	10/28/2014	■
Definir Mercado Forex	Cleiton	10/28/2014	■

#### Revisar Referencial Teórico Métodos Numéricos

Refinar Método de Correlação Linear	Cleiton	10/28/2014	■
Refinar Método de Fibonacci	Cleiton	10/28/2014	■
Refinar Método de Mínimos Quadrados	Cleiton	10/28/2014	■

### Sprint: Sprint4

#### Realizar Experimentos Métodos de Operação

Realizar Experimento Método de Fibonacci	Cleiton		■
Realizar Experimento Método de Média Móvel	Cleiton	■	
Realizar Experimento Método de Estocástico	Cleiton	■	
Realizar Experimento Método de Correlação Linear	Cleiton		■

Realizar Experimento Método de Mínimos Quadrados	Cleiton			
<b>Evidenciar Aplicações Paradigmas de Programação</b>				
Evidenciar aplicações paradigma MultiAgente	Vanessa			
Evidenciar aplicações paradigma Funcional	Vanessa			
Evidenciar aplicações paradigma Estruturado	Vanessa			
Evidenciar aplicações paradigma Lógico	Vanessa			
<b>Revisar Referencial Teórico Contexto Financeiro</b>				
Revisar Suporte e Resistência	Cleiton	10/28/2014		
Revisar Mercado Forex	Vanessa	10/28/2014		
Revisar Alavancagem	Cleiton	10/28/2014		
<b>Desenvolver Experts em MQL4</b>				
Desenvolver Expert MediaMovel	Vanessa	10/28/2014		
Desenvolver Expert MinimosQuadrados	Cleiton	10/28/2014		
Desenvolver Expert CorrelacaoLinear	Cleiton	10/28/2014		
Desenvolver Expert Estocastico	Vanessa	10/28/2014		
Desenvolver Expert Fibonacci	Cleiton	10/28/2014		
<b>Sprint: Sprint5</b>				

### Descrever Metodologia de Pesquisa

Descrever Conceito de Objeto	Cleiton	10/28/2014	■
Descrever Conceito de Procedimentos Técnicos	Cleiton	10/28/2014	■

### Realizar Experimentos Métodos de Operação

Realizar experimento Método de Mínimos Quadrados	Cleiton	10/28/2014	■
Realizar experimento Método de Estocástico	Cleiton	10/28/2014	■
Realizar experimento Método de Média Móvel	Cleiton	10/28/2014	■
Realizar experimento Método de Correlação Linear	Cleiton	10/28/2014	■
Realizar experimento Método de Fibonacci	Cleiton	10/28/2014	■

### Evidenciar Aplicações Paradigmas de Programação

Evidenciar aplicações paradigma MultiAgente	Vanessa	10/28/2014	■
Evidenciar aplicações paradigma Lógico	Vanessa	11/9/2014	■
Evidenciar aplicações paradigma Estruturado	Vanessa	11/9/2014	■
Evidenciar aplicações paradigma Funcional	Vanessa	10/28/2014	■

### Sprint: Sprint6

#### Verificar Qualidade de Código

Realizar Análise Estática	Cleiton	11/9/2014	■
---------------------------	---------	-----------	---



Realizar Teste Unitário em Haskell	Vanessa	11/9/2014	■
<b>Definir Backlog de Histórias</b>			
Componente Estruturado	Cleiton	11/9/2014	■
Componente Orientado a Objetos	Vanessa	11/9/2014	■
Componente Multiagente	Cleiton	11/9/2014	■
Componente Lógico	Cleiton	11/9/2014	■
Componente Funcional	Vanessa	11/9/2014	■



## APÊNDICE D – Experts para Experimento

## Experts em linguagem MQL4

### CorrelacaoPearson.mql

```
1. #property copyright "Copyright 2014, Cleiton Gomes"
2. #property link      "cleitoncsg@gmail.com"

3. #define TAKE_PROFIT 500
4. #define STOP_LOSS 500
5. #define ALAVANCAGEM 0.25
6. #define CORRELACAO_ACEITAVEL 0.89
7. #define SEXTA_FEIRA 5
8.
9. int ticket=0;
10. string nome = "CSG";
11. bool realizaOrdem;
12. double estado_mercado;
13.
14. int start(){
15.     bool venda, compra;
16.
17.     if(correlacao_pearson(55) > CORRELACAO_ACEITAVEL && correlacao_pearson(34) >
        CORRELACAO_ACEITAVEL &&
18.         correlacao_pearson(21) > CORRELACAO_ACEITAVEL && correlacao_pearson(13) >
        CORRELACAO_ACEITAVEL){
19.         realizaOrdem = true;
20.     }
21.
22.     if( DayOfWeek() != SEXTA_FEIRA ){
23.         if(realizaOrdem == true && estado_mercado > 0 ){
24.             compra = true;
25.         }
26.         if(realizaOrdem == true && estado_mercado < 0 ){
27.             venda = true;
28.         }
```

```

29.     }
30.
31.     if( ((compra == true  && OrdersTotal() == 0)) ){
32.         RefreshRates();
33.         while (IsTradeContextBusy()) Sleep(5);
34.         ticket= OrderSend(Symbol(),OP_BUY,ALAVANCAGEM,Ask,0,Ask - TAKE_PROFIT*Point,
35.             Ask + TAKE_PROFIT*Point,nome,AccountNumber(),0,Yellow);
36.     }
37.     if( ((venda == true && OrdersTotal() == 0)) ){
38.         RefreshRates();
39.         while (IsTradeContextBusy()) Sleep(5);
40.         ticket= OrderSend(Symbol(),OP_SELL,ALAVANCAGEM,Bid,0,Bid + STOP_LOSS*Point,
41.             Bid - TAKE_PROFIT*Point,nome,AccountNumber(),0,Green);
42.     }
43.     double ponto_positivo, ponto_negativo;
44.
45.     for(int j=0; j < OrdersHistoryTotal();j++){
46.         OrderSelect(j,SELECT_BY_POS,MODE_HISTORY);
47.         if(OrderSymbol()!=Symbol()) continue;
48.         if(OrderMagicNumber() != AccountNumber()) continue;
49.         if(OrderProfit() > 0){
50.             ponto_positivo++;
51.         }
52.         else{
53.             ponto_negativo++;
54.         }
55.     }
56.
57.     Comment(
58.         "Margem da Conta = ", AccountMargin() ,"\n",
59.         "Ordens em lucro = ", ponto_positivo ,"\n",
60.         "Ordens em prejuízo = ", ponto_negativo ,"\n",
61.         "STOP LOSS = ", STOP_LOSS ,"\n",
62.         "TAKE PROFIT = ", TAKE_PROFIT ,"\n",

```

```

63.         "CORRELAÇÃO LINEAR 55 ", correlacao_pearson(55) , "\n",
64.         "CORRELAÇÃO LINEAR 34 ", correlacao_pearson(34) , "\n",
65.         "CORRELAÇÃO LINEAR 21 ", correlacao_pearson(21) , "\n",
66.         "CORRELAÇÃO LINEAR 13 ", correlacao_pearson(13) , "\n",
67.         ""
68.     );
69.     return(0);
70. }
71. double correlacao_pearson(int tempoCorrelacao){
72.     int c =0;
73.     double soma_ordenadas = 0;
74.     double soma_abcissas = 0;
75.     double soma_ordenadas_quadrado = 0;
76.     double soma_abcissas_quadrado = 0;
77.     double numero_abcissa;
78.     double numero_ordenada;
79.     double soma_X_vezes_Y = 0;
80.     double numerador, denominador_1,denominador,correlacao;
81.
82.     for(c=0; c<tempoCorrelacao; c++){
83.         numero_abcissa = NormalizeDouble(Open[c],5);
84.         numero_ordenada =NormalizeDouble(Close[c],5);
85.         soma_abcissas = soma_abcissas + numero_abcissa;
86.         soma_abcissas_quadrado = (soma_abcissas_quadrado) +
            (numero_abcissa)*(numero_abcissa);
87.         soma_ordenadas = soma_ordenadas + numero_ordenada;
88.         soma_ordenadas_quadrado = (soma_ordenadas_quadrado) +
            (numero_ordenada)*(numero_ordenada);
89.         soma_X_vezes_Y = soma_X_vezes_Y + (numero_ordenada*numero_abcissa);
90.     }
91.
92.     numerador
    =((tempoCorrelacao*soma_X_vezes_Y)-((soma_abcissas)*(soma_ordenadas)));

```

```

93.                                                                 denominador_1
    =((tempoCorrelacao*soma_abcissas_quadrado)-(soma_abcissas*soma_abcissas))*
94.
    ((tempoCorrelacao*soma_ordenadas_quadrado)-(soma_ordenadas*soma_ordenadas));
95.
96.         denominador = MathPow(denominador_1,1.0/2.0);
97.
98.         if(denominador != 0)
99.             correlacao = numerador/denominador;
100.        else
101.            correlacao = 0;
102.
103.        estado_mercado = soma_abcissas - soma_ordenadas;
104.
105.        return (correlacao);
106.    }

```

## Estocastico.mql

```

1. #property copyright "Copyright 2014, Cleiton Gomes"
2. #property link      "cleitoncsg@gmail.com"
3.
4. #define TAKE_PROFIT 500
5. #define STOP_LOSS 500
6. #define ALAVANCAGEM 0.25
7. #define TEMPO_OPERACAO 60
8. #define SEXTA_FEIRA 5
9. #define AJUSTE_TEMPORAL_MAIOR 5
10. #define AJUSTE_TEMPORAL_MENOR 3
11.
12. int ticket=0;
13. string nome = "CSG";
14. bool realizaOrdem;
15.
16. int start(){
17.     bool venda, compra;

```

```

18.
19.   HideTestIndicators(TRUE);
20.   RefreshRates();
21.   double estocastico
      =iStochastic(NULL,TEMPO_OPERACAO,AJUSTE_TEMPORAL_MAIOR,AJUSTE_TEMPORAL_MENOR,AJUSTE_TEMP
      ORAL_MENOR,MODE_SMA,0,MODE_MAIN,1);
22.   double sinal
      =iStochastic(NULL,TEMPO_OPERACAO,AJUSTE_TEMPORAL_MAIOR,AJUSTE_TEMPORAL_MENOR,AJUSTE_TEMP
      ORAL_MENOR,MODE_SMA,0,MODE_SIGNAL,1);
23.
24.   if( DayOfWeek() != SEXTA_FEIRA ){
25.
26.       if (estocastico > sinal){
27.           compra = true;
28.       }
29.
30.       if (estocastico < sinal){
31.           venda = true;
32.       }
33.   }
34.
35.   if( ((compra == true && OrdersTotal() == 0 )) ){
36.       RefreshRates();
37.       while (IsTradeContextBusy()) Sleep(5);
38.       ticket= OrderSend(Symbol(),OP_BUY,ALAVANCAGEM,Ask,0,Ask - TAKE_PROFIT*Point,
39.       Ask + TAKE_PROFIT*Point,nome,AccountNumber(),0,Yellow);
40.
41.   }
42.   if( ((venda == true && OrdersTotal() == 0 )) ){
43.       RefreshRates();
44.       while (IsTradeContextBusy()) Sleep(5);
45.       ticket= OrderSend(Symbol(),OP_SELL,ALAVANCAGEM,Bid,0,Bid + STOP_LOSS*Point,
46.       Bid - TAKE_PROFIT*Point,nome,AccountNumber(),0,Green);
47.   }

```



```

48.
49.     double ponto_positivo, ponto_negativo;
50.
51.     for(int j=0; j < OrdersHistoryTotal();j++){
52.         OrderSelect(j,SELECT_BY_POS,MODE_HISTORY);
53.         if(OrderSymbol()!=Symbol()) continue;
54.         if(OrderMagicNumber() != AccountNumber()) continue;
55.         if(OrderProfit() > 0){
56.             ponto_positivo++;
57.         }
58.         else{
59.             ponto_negativo++;
60.         }
61.     }
62.     Comment(
63.         "Margem da Conta = ", AccountMargin() ,"\n",
64.         "Ordens em lucro = ", ponto_positivo ,"\n",
65.         "Ordens em prejuizo = ", ponto_negativo ,"\n",
66.         "STOP LOSS = ", STOP_LOSS ,"\n",
67.         "TAKE PROFIT = ", TAKE_PROFIT ,"\n",
68.         ""
69.     );
70.     return(0);
71. }

```

## Fibonacci.mql

```

1. #property link      "cleitoncsg@gmail.com"
2. #include "suporteResistencia.mq4"
3.
4. #define TAKE_PROFIT 500
5. #define STOP_LOSS 500
6. #define ALAVANCAGEM 0.25
7. #define FATOR_RETRACAO 0.38
8. #define MAX_CANDLES 34
9. #define SEXTA_FEIRA

```

```

10. #define ESTADO_VALIDO 0.01
11. double retracao_fibo;
12.
13. int ticket=0;
14. string nome = "CSG";
15.
16. int start(){
17.     bool venda;
18.
19.     if(NormalizeDouble(retracao_fibonacci()+ suporte(),4) == Bid &&
        estadoMercado(MAX_CANDLES) > ESTADO_VALIDO){
20.         venda = true;
21.     }
22.
23.     if( ((venda == true && OrdersTotal() == 0)) ){
24.         RefreshRates();
25.         while (IsTradeContextBusy()) Sleep(5);
26.
27.         ticket= OrderSend(Symbol(),OP_SELL,ALAVANCAGEM,Bid,0,Bid + STOP_LOSS*Point,
28.         Bid - TAKE_PROFIT*Point,nome,AccountNumber(),0,Green);
29.     }
30.
31.     double ponto_positivo, ponto_negativo;
32.
33.     for(int j=0; j < OrdersHistoryTotal();j++){
34.         OrderSelect(j,SELECT_BY_POS,MODE_HISTORY);
35.
36.         if(OrderSymbol()!=Symbol()) continue;
37.         if(OrderMagicNumber() != AccountNumber()) continue;
38.         if(OrderProfit() > 0){
39.             ponto_positivo++;
40.         }
41.         else{
42.             ponto_negativo++;

```

```

43.         }
44.     }
45.     Comment(
46.         "Margem da Conta = ", AccountMargin() ,"\n",
47.         "Ordens em lucro = ", ponto_positivo ,"\n",
48.         "Ordens em prejuizo = ", ponto_negativo ,"\n",
49.         "Suporte = ", suporte() ,"\n",
50.         "Resistencia = ", resistencia() ,"\n",
51.         "Retracao Fibo = ", retracao_fibonacci() ,"\n",
52.         ""
53.     );
54.     return(0);
55. }
56. double estadoMercado(int tempoCorrelacao){
57.     double soma_ordenadas = 0, soma_abcissas = 0;
58.     double numero_abcissa, numero_ordenada;
59.
60.     for(int c=0; c<tempoCorrelacao; c++){
61.         numero_abcissa = NormalizeDouble(Open[c],5);
62.         numero_ordenada =NormalizeDouble(Close[c],5);
63.         soma_abcissas = soma_abcissas + numero_abcissa;
64.         soma_ordenadas = soma_ordenadas + numero_ordenada;
65.     }
66.     return ( soma_abcissas - soma_ordenadas);
67. }
68. double retracao_fibonacci(){
69.     double retracao = ( resistencia() - suporte())*FATOR_RETRACAO;
70.
71.     return (retracao);
72. }

```

## MediaMovel.mql

```

1. #property copyright "Copyright 2014, Cleiton Gomes"
2. #property link      "cleitoncsg@gmail.com"
3.

```

```

4. #define TAKE_PROFIT 500
5. #define STOP_LOSS 500
6. #define ALAVANCAGEM 0.25
7. #define TEMPO_OPERACAO 60
8. #define SEXTA_FEIRA 5
9.
10. extern int mediaMovelRapida = 12;
11. extern int mediaMovelLenta = 26;
12.
13. int ticket=0;
14. string nome = "CSG";
15. bool realizaOrdem;
16.
17. int start(){
18.     bool venda, compra;
19.
20.     double
        mediaMovelRapidaCorrente=ima(NULL,TEMPO_OPERACAO,mediaMovelRapida,0,MODE_EMA,PRICE_CLOSE
        ,0);
21.     double mediaMovelLentaCorrente
        =ima(NULL,TEMPO_OPERACAO,mediaMovelLenta,0,MODE_EMA,PRICE_CLOSE,0);
22.
23.     if( DayOfWeek() != SEXTA_FEIRA ){
24.
25.         if (mediaMovelLentaCorrente < mediaMovelRapidaCorrente){
26.             compra = true;
27.         }
28.
29.         if (mediaMovelLentaCorrente > mediaMovelRapidaCorrente){
30.             venda = true;
31.         }
32.     }
33.
34.     if( ((compra == true && OrdersTotal() == 0)) ){

```

```

35. RefreshRates();
36. while (IsTradeContextBusy()) Sleep(5);
37. ticket= OrderSend(Symbol(),OP_BUY,ALAVANCAGEM,Ask,0,Ask - TAKE_PROFIT*Point,
38. Ask + TAKE_PROFIT*Point,nome,AccountNumber(),0,Yellow);
39.
40. }
41. if( ((venda == true && OrdersTotal() == 0)) ){
42. RefreshRates();
43. while (IsTradeContextBusy()) Sleep(5);
44. ticket= OrderSend(Symbol(),OP_SELL,ALAVANCAGEM,Bid,0,Bid + STOP_LOSS*Point,
45. Bid - TAKE_PROFIT*Point,nome,AccountNumber(),0,Green);
46. }
47.
48. double ponto_positivo, ponto_negativo;
49.
50. for(int j=0; j < OrdersHistoryTotal();j++){
51. OrderSelect(j,SELECT_BY_POS,MODE_HISTORY);
52. if(OrderSymbol()!=Symbol()) continue;
53. if(OrderMagicNumber() != AccountNumber()) continue;
54. if(OrderProfit() > 0){
55. ponto_positivo++;
56. }
57. else{
58. ponto_negativo++;
59. }
60. }
61.
62. Comment(
63. "Margem da Conta = ", AccountMargin() ,"\\n",
64. "Ordens em lucro = ", ponto_positivo ,"\\n",
65. "Ordens em prejuizo = ", ponto_negativo ,"\\n",
66. "STOP LOSS = ", STOP_LOSS ,"\\n",
67. "TAKE PROFIT = ", TAKE_PROFIT ,"\\n",
68. ""

```

```
69.     );  
70.  
71.     return(0);  
72. }
```

## MinimosQuadrados.mql

```
1.  #property copyright "Copyright 2014, Cleiton Gomes"  
2.  #property link      "cleitoncsg@gmail.com"  
3.  
4.  #define ALAVANCAGEM 0.25  
5.  #define QUANTIDADE_CANDLES 34  
6.  #define AJUSTE_SL 8  
7.  #define AJUSTE_CA 0.1  
8.  #define SEXTA_FEIRA 5  
9.  
10. extern double take_profit_fixo, stop_loss_fixo;  
11.  
12. int ticket=0;  
13. string nome = "CSG";  
14. double coeficienteAngular;  
15.  
16. int start(){  
17.     bool venda, compra;  
18.     double take_profit, stop_loss;  
19.     double produto_coeficienteAngular_cotacao;  
20.  
21.     produto_coeficienteAngular_cotacao = calculoCoeficienteLinear(QUANTIDADE_CANDLES);  
22.  
23.     if( DayOfWeek() != SEXTA_FEIRA ){  
24.         if(coeficienteAngular < 0){  
25.             coeficienteAngular = coeficienteAngular*(-1);  
26.         }  
27.         if(produto_coeficienteAngular_cotacao > 1){  
28.             compra = true;  
29.         }
```

```

30.     if(produto_coeficienteAngular_cotacao < 0){
31.         venda = true;
32.     }
33.     take_profit = (Ask + (coeficienteAngular)*AJUSTE_CA);
34.     stop_loss = (Bid - AJUSTE_SL*(coeficienteAngular)*AJUSTE_CA);
35. }
36.
37. if( ((compra == true  && OrdersTotal() == 0 && venda != true)) ){
38.     take_profit_fixo = take_profit;
39.     stop_loss_fixo = stop_loss;
40.     RefreshRates();
41.     while (IsTradeContextBusy()) Sleep(5);
42.     ticket= OrderSend(Symbol(),OP_BUY,ALAVANCAGEM,Ask,0,stop_loss_fixo,
43.         take_profit_fixo,nome,AccountNumber(),0,Yellow);
44. }
45. if( ((venda == true  && OrdersTotal() == 0 && compra != true)) ){
46.     take_profit_fixo = take_profit;
47.     stop_loss_fixo = stop_loss;
48.
49.     RefreshRates();
50.     while (IsTradeContextBusy()) Sleep(5);
51.     ticket= OrderSend(Symbol(),OP_SELL,ALAVANCAGEM,Bid,0,stop_loss_fixo,
52.         take_profit_fixo,nome,AccountNumber(),0,Green);
53. }
54.
55.     double ponto_positivo, ponto_negativo;
56.
57.     for(int j=0; j < OrdersHistoryTotal();j++){
58.         OrderSelect(j,SELECT_BY_POS,MODE_HISTORY);
59.         if(OrderSymbol()!=Symbol()) continue;
60.         if(OrderMagicNumber() != AccountNumber()) continue;
61.         if(OrderProfit() > 0){
62.             ponto_positivo++;
63.         }

```

```

64.         else{
65.             ponto_negativo++;
66.         }
67.     }
68.     Comment(
69.         "Margem da Conta = ", AccountMargin() ,"\n",
70.         "Ordens em lucro = ", ponto_positivo ,"\n",
71.         "Ordens em prejuizo = ", ponto_negativo ,"\n",
72.         "STOP LOSS ", stop_loss_fixo ,"\n",
73.         "TAKE PROFIT", take_profit_fixo ,"\n",
74.         "COEFICIENTE LINEAR ", calculoCoeficienteLinear(34) ,"\n",
75.         "COEFICIENTE ANGULAR ", coeficienteAngular ,"\n",
76.         ""
77.     );
78.     return(0);
79. }
80. double calculoCoeficienteLinear(int quantidadeVelas){
81.     double soma_x = 0, soma_y = 0;
82.     double numerador, denominador;
83.     double variacaoLinear;
84.     int i;
85.
86.     for(i = 1; i < quantidadeVelas; i++){
87.         soma_x = soma_x + Open[i];
88.         soma_y = soma_y + Close[i];
89.     }
90.     for(i = 1; i < quantidadeVelas; i++){
91.         numerador = Open[i]*(Close[i] - soma_x/quantidadeVelas);
92.         denominador = Close[i]*(Open[i] - soma_y/quantidadeVelas);
93.     }
94.     variacaoLinear = numerador/denominador;
95.     coeficienteAngular = soma_y/quantidadeVelas -
        (variacaoLinear*soma_x/quantidadeVelas);
96.

```



```
97.     return variacaoLinear;
98. }
```

### ResistenciaSuporte.mql

```
1.  #property copyright "Copyright 2014, Cleiton Gomes"
2.  #property link      "cleitoncsg@gmail.com"
3.
4.  #define QUANTIDADE_CANDLES 13
5.
6.  double resistencia(){
7.      double maior = -99;
8.
9.      for(int i = 0; i < QUANTIDADE_CANDLES;i++){
10.         if(Open[i] > maior)
11.             maior = Open[i];
12.     }
13.     return maior;
14. }
15. double suporte(){
16.     double menor = 99;
17.
18.     for(int i = 0; i < QUANTIDADE_CANDLES;i++){
19.         if(Close[i] < menor)
20.             menor = Close[i];
21.     }
22.     return menor;
23. }
```



## APÊNDICE E – Componente OO

## Expert.groovy

```
package investmvc
```

```
class Expert {
```

```
    String name
```

```
    int quote
```

```
    static constraints = {
```

```
    }
```

```
}
```

## User.groovy

```
1. package investmvc.security
```

```
2.
```

```
3. class User {
```

```
4.     transient springSecurityService
```

```
5.
```

```
6.     String username
```

```
7.     String password
```

```
8.     boolean enabled = true
```

```
9.     boolean accountExpired
```

```
10.    boolean accountLocked
```

```
11.    boolean passwordExpired
```

```
12.    static transients = ['springSecurityService']
```

```
13.
```

```
14.    static constraints = {
```

```
15.        username blank: false, unique: true
```

```
16.        password blank: false
```

```
17.    }
```

```
18.    static mapping = {
```

```
19.        password column: '`password`'
```

```
20.    }
```

```
21.
```

```
22.     Set<Role> getAuthorities() {
23.         UserRole.findAllByUser(this).collect { it.role }
24.     }
25.     def beforeInsert() {
26.         encodePassword()
27.     }
28.
29.     def beforeUpdate() {
30.         if (isDirty('password')) {
31.             encodePassword()
32.         }
33.     }
34.
35.     protected void encodePassword() {
36.         password = springSecurityService?.passwordEncoder ?
springSecurityService.encodePassword(password) : password
37.     }
38. }
```

## Tela : Criar Expert

# Criar Expert

[\\_DemoPage](#) [Dbdoc](#) **Expert** [Home](#) [Login](#) [Logout](#) [User](#)

Expert Listagem

+ Novo Expert

Name

Quote\*

0

Criar

Reset

## Tela : Criar Usuário

# User

[\\_DemoPage](#) [Dbdoc](#) [Expert](#) [Home](#) [Login](#) [Logout](#) **User**

User Listagem

+ Novo User

Username	Password
admin@com	\$2a\$10\$9/c/PedPNFa963.6mSu3t.kYv0PKUiPlogaAf6nA52/1Xns.GH3Nm
user@user.com	\$2a\$10\$fxQE7ePwj7B9/c40x.fQUeGk8K9fYLZotHGwQ4neVfXn2VSNu4IGi

Anterior

Próximo

## APÊNDICE F – Componente Funcional

## CorrelacaoDePearson.hs

```
1 module
CorrelacaoDePearson(calculaMedia,vetorX,vetorY,qtdCandles,somaQuadradoVetor)
where

2 import System.IO
3 import Foreign.Marshal.Unsafe
4 import ArquivosForex(cotacoes,detectaQuantidadeCandle)
5
6
7 intToFloat :: Int -> Float
8 intToFloat n = fromIntegral n ::Float
9
10 calculaMedia [] = 0
11 calculaMedia (cabeca:calda) = sum (cabeca:calda) / fromIntegral(length
(cabeca:calda))
12
13 vetorX [] = []
14 vetorX (cabeca : calda) = init (cabeca : calda)
15
16 vetorY [] = []
17 vetorY (cabeca : calda) = tail (cabeca : calda)
18
19 vetorXY [] [] = 0
20 vetorXY (cabecaX : caldax) (cabecaY : calday) = (cabecaX*cabecaY) + (vetorXY
caldax calday)
21
22 somaQuadradoVetor [] = 0;
23 somaQuadradoVetor (cabeca:calda) = (cabeca*cabeca) + (somaQuadradoVetor
calda)
24
25 somaAbcissas = sum(vetorX cotacoes)
26 somaAbcissasQuadrado = somaQuadradoVetor (vetorX cotacoes)
27 somaOrdenadas = sum (vetorY cotacoes)
28 somaOrdenadasQuadrado = somaQuadradoVetor (vetorY cotacoes)
29 xy = vetorXY (vetorX cotacoes) (vetorY cotacoes)
30 qtdCandles = intToFloat (unsafeLocalState detectaQuantidadeCandle)
31
32 numerador = (qtdCandles*xy)-(somaAbcissas*somaOrdenadas)
33 denominador =sqrt( (
(qtdCandles*somaAbcissasQuadrado)-(somaAbcissas*somaAbcissas)
```



```

)*((qtdCandles*somaOrdenadasQuadrado)-(somaOrdenadas*somaOrdenadas)) )
34
35 correlacao = numerador / denominador

```

## Fibonacci.hs

```

1 module Fibonacci (fibonacci,retracao,suporte,resistencia) where
2 import ArquivosForex(tendencia, cotacoes)
3 import Foreign.Marshal.Unsafe
4
5 suporte (cabeca:calda)= minimum (cabeca:calda)
6
7 resistencia (cabeca:calda)= maximum (cabeca:calda)
8
9 retracao s r n = (r - s)*n + s
10
11 fibonacci n
12   |unsafeLocalState(tendencia) < 0 = (retracao (suporte cotacoes) (resistencia
cotacoes) n)
13   |otherwise = (retracao (resistencia cotacoes) (suporte cotacoes) n)

```

## MinimosQuadrados.hs

```

1 module MinimosQuadrados (numerador, denominador, variacaoAngular,
variacaoLinear, coeficienteAngular, coeficienteLinear) where
2 import ArquivosForex(cotacoes)
3 import CorrelacaoDePearson(calculaMedia,qtdCandles,vetorX,vetorY)
4
5 numerador [] [] = 0
6 numerador (cabecaX:caldaX) (cabecaY: caldaY) =
7   ( (cabecaX-(calculaMedia (cabecaX:caldaX)))*(cabecaY-(calculaMedia
(cabecaY:caldaY))) )+ (numerador caldaX caldaY)
8
9 denominador [] [] = 0
10 denominador (cabecaX:caldaX) =
11   ((cabecaX - (calculaMedia (cabecaX:caldaX)))*(cabecaX - (calculaMedia
(cabecaX:caldaX)))) + (denominador caldaX)
12
13 variacaoAngular (cabecaX:caldaX) (cabecaY: caldaY) =
14   (numerador (cabecaX:caldaX) (cabecaY: caldaY))/(denominador

```

(cabecaX:caldaX))

15

16 **variacaoLinear** (cabecaX:caldaX) (cabecaY:caldaY) =

17     (calculaMedia (cabecaY:caldaY)) - ((variacaoAngular (cabecaX:caldaX)  
(cabecaY:caldaY)) \* (calculaMedia (cabecaX:caldaX)))

18

19 **coeficienteAngular** = variacaoAngular (vetorX cotacoes) (vetorY cotacoes)

20

21 **coeficienteLinear** = variacaoLinear (vetorX cotacoes) (vetorY cotacoes)

## TesteCorrelacaoDePearson.hs

```
1 module TesteCorrelacaoDePearson(main) where
2
3 import Test.HUnit
4 import CorrelacaoDePearson
5
6 testaMediaListaVazia :: Test
7 testaMediaListaVazia = TestCase (assertEqual "MÃ©dia de lista vazia" 0
(calculaMedia []))
8
9 testaMediaLista :: Test
10 testaMediaLista = TestCase (assertEqual "MÃ©dia de uma lista" 3 (calculaMedia
[3,3,3]))
11
12 testaVetorXVazia :: Test
13 testaVetorXVazia = TestCase (assertEqual "Vetor X" 0 (length(vetorX [])) )
14
15 testaVetorX :: Test
16 testaVetorX = TestCase (assertEqual "Vetor X" [1,2,3] (vetorX [1,2,3,4]))
17
18 testaVetorYVazia :: Test
19 testaVetorYVazia = TestCase (assertEqual "Vetor Y" 0 (length(vetorY [])) )
20
21 testaVetorY :: Test
22 testaVetorY = TestCase (assertEqual "Vetor Y" [1,2,3] (vetorY [0,1,2,3]))
23
24 testaSomaQuadradoVetorVazia :: Test
25 testaSomaQuadradoVetorVazia = TestCase (assertEqual "Soma quadrÃ¡tica de uma
lista vazia" 0 (somaQuadradoVetor []))
26
27 testaSomaQuadradoVetor :: Test
28 testaSomaQuadradoVetor = TestCase (assertEqual "Soma quadrÃ¡tica de uma lista
[1,2,3]" 14 (somaQuadradoVetor [1,2,3]))
29
30 suiteDeTeste :: Test
31 suiteDeTeste = TestList [testaMediaListaVazia,testaMediaLista,testaVetorX,
testaVetorXVazia, testaVetorY, testaVetorYVazia,
testaSomaQuadradoVetor,testaSomaQuadradoVetorVazia]
32
33 main :: IO Counts
```

```
34 main = runTestTT suiteDeTeste
```

### TesteFibonacci.hs

```
1 module TesteFibonacci(main) where
2
3 import Test.HUnit
4 import Fibonacci
5
6 testaSuporte :: Test
7 testaSuporte = TestCase (assertEqual "Suporte de uma lista Comum" 0 (suporte
[0,1,2,3,4]))
8
9 testaResistencia :: Test
10 testaResistencia = TestCase (assertEqual "Resistência de uma lista Comum" 4
(resistencia [0,1,2,3,4]))
11
12 testaRetracao :: Test
13 testaRetracao = TestCase (assertEqual "Retração Simples" 1 (retracao 1 1 0))
14
15 suiteDeTeste :: Test
16 suiteDeTeste = TestList [testaRetracao,testaSuporte,testaResistencia]
17
18 main :: IO Counts
19 main = runTestTT suiteDeTeste
```

### TesteMinimosQuadrados.hs

```
1 module TesteMinimosQuadrados(main) where
2
3 import Test.HUnit
4 import MinimosQuadrados
5
6 testaNumeradorVazio :: Test
7 testaNumeradorVazio = TestCase (assertEqual "Numerador com lista vazia" 0
(numerador [] []))
8
9 testaDenominadorVazio :: Test
10 testaDenominadorVazio = TestCase (assertEqual "Denominador com lista vazia" 0
```

(denominador []))

11

12 **suiteDeTeste** :: **Test**

13 **suiteDeTeste** = **TestList** [testaNumeradorVazio, testaDenominadorVazio]

14

15 **main** :: **IO Counts**

16 **main** = runTestTT suiteDeTeste



## APÊNDICE G – Componente Estruturado

## correlacaoLinear.c

```
1 //+-----+
2 /|          MÃ©todo de CorrelacÃ£o Linear |
3 /|          Copyright © 2014, Cleiton Gomes; Vanessa Barbosa |
4 /|          http://www.softwarecsg.com.br |
5 //+-----+
6
7 #include <stdio.h>
8 #include <stdlib.h>
9 #include <string.h>
10 #include <math.h>
11 #define QUANTIDADE_CANDLES 100
12 #define TAMANHO_STRING 50
13
14 char nomeRobo[TAMANHO_STRING], nomeTipoGrafico[2];
15 double quantidadeCandes;
16 double tendencia;
17
18 double calculaCorrelacao(int tempoCorrelacao);
19 void detectaRoboETipoDeGrafico();
20
21 int main(){
22     FILE *arquivo;
23     FILE *arquivoTendencia;
24
25     detectaRoboETipoDeGrafico();
26     printf("METODO CORRELACAO EM C LIGADO\n");
27     //printf("Correlacao Linear em C: %f\n",calculaCorrelacao(quantidadeCandes));
28
29     arquivo = fopen("../correlacaoResposta.txt", "wt");
30     fprintf(arquivo, "%f", calculaCorrelacao(quantidadeCandes));
31
32     fclose(arquivo);
33     arquivoTendencia = fopen("../tendencia.txt", "wt");
34     //printf("tendencia : %f\n", tendencia);
35     fprintf(arquivoTendencia, "%f", tendencia);
36
37     fclose(arquivoTendencia);
38
39     return 0;
40 }
```



```

41
42 void detectaRoboETipoDeGrafico(){
43     FILE *arquivo;
44     char temporariaQuantidadeCandle[10];
45
46     arquivo = fopen("../criterioEntrada.txt","rt");
47
48     if(arquivo == NULL){
49         printf("Arquivo nulo\n");
50     }
51
52     fscanf(arquivo,"%s",nomeRobo);
53     //printf("nome robo: %s\n", nomeRobo);
54     fscanf(arquivo,"%s",nomeTipoGrafico);
55     //printf("Nome tipo grafico: %s\n", nomeTipoGrafico);
56     fscanf(arquivo,"%s",temporariaQuantidadeCandle);
57     //printf("Quantidade candles %s\n", temporariaQuantidadeCandle);
58     quantidadeCandles = atoi(temporariaQuantidadeCandle);
59
60     fclose(arquivo);
61 }
62
63 double calculaCorrelacao(int tempoCorrelacao){
64
65     FILE *arquivo;
66     int c;
67     double somaOrdenadas = 0, somaAbcissas = 0,
68         somaOrdenadasQuadrado = 0, somaAbcissasQuadrado = 0,
69         somaXvezesY = 0, correlacao,
70         numeroAbcissa, numeroOrdenada,
71         numerador, denominador_1, denominador;
72
73     double leituraCotacoes[tempoCorrelacao];
74
75     if( (strcmp(nomeTipoGrafico,"M1")) == 0)
76         arquivo = fopen("../MQL4/Files/M1.csv","rt");
77     else if( (strcmp(nomeTipoGrafico,"M5")) == 0)
78         arquivo = fopen("../MQL4/Files/M5.csv","rt");
79     else if( (strcmp(nomeTipoGrafico,"M15")) == 0)
80         arquivo = fopen("../MQL4/Files/M15.csv","rt");
81     else if( (strcmp(nomeTipoGrafico,"M30")) == 0)
82         arquivo = fopen("../MQL4/Files/M30.csv","rt");

```

```

83  else if( (strcmp(nomeTipoGrafico,"H1")) == 0)
84      arquivo = fopen("../MQL4/Files/H1.csv","rt");
85  else if( (strcmp(nomeTipoGrafico,"H4")) == 0)
86      arquivo = fopen("../MQL4/Files/H4.csv","rt");
87  else if( (strcmp(nomeTipoGrafico,"D1")) == 0)
88      arquivo = fopen("../MQL4/Files/D1.csv","rt");
89  else if( (strcmp(nomeTipoGrafico,"MN1")) == 0)
90      arquivo = fopen("../MQL4/Files/MN1.csv","rt");
91  else if( (strcmp(nomeTipoGrafico,"W1")) == 0)
92      arquivo = fopen("../MQL4/Files/W1.csv","rt");
93  else
94      printf("Erro, tabela nao encontrada\n");
95
96  for(c=0; c<= tempoCorrelacao; c++){
97      fscanf(arquivo, "%lf",&leituraCotacoes[c]);
98  }
99
100  for(c=0; c< tempoCorrelacao; c++){
101      numeroAbcissa = leituraCotacoes[c];
102      numeroOrdenada = leituraCotacoes[c+1];
103
104      somaAbcissas = somaAbcissas + numeroAbcissa;
105      somaAbcissasQuadrado += (numeroAbcissa*numeroAbcissa);
106      somaOrdenadas = somaOrdenadas + numeroOrdenada;
107      somaOrdenadasQuadrado += (numeroOrdenada*numeroOrdenada);
108      somaXvezesY = somaXvezesY + (numeroOrdenada*numeroAbcissa);
109  }
110
111  numerador
112  =((tempoCorrelacao*somaXvezesY)-((somaAbcissas)*(somaOrdenadas)));
113  denominador_1
114  =((tempoCorrelacao*somaAbcissasQuadrado)-(somaAbcissas*somaAbcissas))*
115  ((tempoCorrelacao*somaOrdenadasQuadrado)-(somaOrdenadas*somaOrdenadas));
116  denominador = sqrt(denominador_1);
117  printf("Denominador %lf\n", denominador);
118  tendencia = somaAbcissas - somaOrdenadas;
119  correlacao = numerador/denominador;
120
121  fclose(arquivo);

```

```
122 printf("Correlacao: %lf\n", correlacao);
123 return correlacao;
124 }
```

## fibonacci.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define TAMANHO_STRING 50
4
5 char nomeRobo[TAMANHO_STRING], nomeTipoGrafico[2];
6
7 void detectaRoboETipoDeGrafico();
8 double calculoSuporte(int quantidadeVelas);
9 double calculoResistencia(int quantidadeVelas);
10 double calculoRegressaoFibonacci(double fatorDeRegressao, int quantidadeVelas);
11 void lerTendencia();
12
13 double quantidadeCandes;
14 double tendencia;
15
16 int main(){
17     FILE *arquivo;
18     FILE *arquivoSuporteResistencia;
19
20     detectaRoboETipoDeGrafico();
21     lerTendencia();
22     //printf("Suporte = %lf, resistencia =
23     //printf("Regressao De Fibonacci = %lf\n", calculoRegressaoFibonacci(0.23, 13));
24     printf("METODO FIBONACCI EM C LIGADO\n");
25
26     arquivo = fopen("../fibonacciResposta.txt", "wt");
27
28     //fprintf(arquivo, "%f\n", calculoRegressaoFibonacci(0.23, quantidadeCandes));
29     fprintf(arquivo, "%f\n", calculoRegressaoFibonacci(0.38, quantidadeCandes));
30     //fprintf(arquivo, "%f\n", calculoRegressaoFibonacci(0.62, quantidadeCandes));
31     fclose(arquivo);
32
33     arquivoSuporteResistencia = fopen("../suporteResistencia.txt", "wt");
34     fprintf(arquivoSuporteResistencia, "%f\n", calculoSuporte(quantidadeCandes));
```

```

35  fprintf(arquivoSuporteResistencia, "%f\n", calculoResistencia(quantidadeCandes));
36
37  return 0;
38 }
39
40 void detectaRoboETipoDeGrafico(){
41  FILE *arquivo;
42  char temporariaQuantidadeCandle[10];
43
44  arquivo = fopen("../criterioEntrada.txt", "rt");
45
46  if(arquivo == NULL){
47      printf("Arquivo nulo\n");
48  }
49
50  fscanf(arquivo, "%s", nomeRobo);
51  //printf("nome robo: %s\n", nomeRobo);
52  fscanf(arquivo, "%s", nomeTipoGrafico);
53  //printf("Nome tipo grafico: %s\n", nomeTipoGrafico);
54  fscanf(arquivo, "%s", &temporariaQuantidadeCandle);
55  //printf("Quantidade candles %s\n", temporariaQuantidadeCandle);
56  quantidadeCandes = atoi(temporariaQuantidadeCandle);
57
58  fclose(arquivo);
59 }
60
61 double calculoSuporte(int quantidadeVelas){
62  FILE *arquivo;
63  double cotacao[quantidadeVelas];
64  double suporte = 777;
65  int i;
66
67  if( (strcmp(nomeTipoGrafico, "M1")) == 0)
68      arquivo = fopen("../MQL4/Files/M1.csv", "rt");
69  else if( (strcmp(nomeTipoGrafico, "M5")) == 0)
70      arquivo = fopen("../MQL4/Files/M5.csv", "rt");
71  else if( (strcmp(nomeTipoGrafico, "M15")) == 0)
72      arquivo = fopen("../MQL4/Files/M15.csv", "rt");
73  else if( (strcmp(nomeTipoGrafico, "M30")) == 0)
74      arquivo = fopen("../MQL4/Files/M30.csv", "rt");
75  else if( (strcmp(nomeTipoGrafico, "H1")) == 0)
76      arquivo = fopen("../MQL4/Files/H1.csv", "rt");

```

```

77     else if( (strcmp(nomeTipoGrafico,"H4")) == 0)
78         arquivo = fopen("../MQL4/Files/H4.csv","rt");
79     else if( (strcmp(nomeTipoGrafico,"D1")) == 0)
80         arquivo = fopen("../MQL4/Files/D1.csv","rt");
81     else if( (strcmp(nomeTipoGrafico,"MN1")) == 0)
82         arquivo = fopen("../MQL4/Files/MN1.csv","rt");
83     else if( (strcmp(nomeTipoGrafico,"W1")) == 0)
84         arquivo = fopen("../MQL4/Files/W1.csv","rt");
85     else
86         printf("Erro, tabela nao encontrada\n");
87
88     for(i = 0; i <= quantidadeVelas; i++){
89         fscanf(arquivo, "%lf",&cotacao[i]);
90         if(suporte > cotacao[i])
91             suporte = cotacao[i];
92     }
93
94     fclose(arquivo);
95     return suporte;
96 }
97
98 double calculoResistencia(int quantidadeVelas){
99     FILE *arquivo;
100     double cotacao[quantidadeVelas];
101     double resistencia = 0;
102     int i;
103
104     if( (strcmp(nomeTipoGrafico,"M1")) == 0)
105         arquivo = fopen("../MQL4/Files/M1.csv","rt");
106     else if( (strcmp(nomeTipoGrafico,"M5")) == 0)
107         arquivo = fopen("../MQL4/Files/M5.csv","rt");
108     else if( (strcmp(nomeTipoGrafico,"M15")) == 0)
109         arquivo = fopen("../MQL4/Files/M15.csv","rt");
110     else if( (strcmp(nomeTipoGrafico,"M30")) == 0)
111         arquivo = fopen("../MQL4/Files/M30.csv","rt");
112     else if( (strcmp(nomeTipoGrafico,"H1")) == 0)
113         arquivo = fopen("../MQL4/Files/H1.csv","rt");
114     else if( (strcmp(nomeTipoGrafico,"H4")) == 0)
115         arquivo = fopen("../MQL4/Files/H4.csv","rt");
116     else if( (strcmp(nomeTipoGrafico,"D1")) == 0)
117         arquivo = fopen("../MQL4/Files/D1.csv","rt");
118     else if( (strcmp(nomeTipoGrafico,"MN1")) == 0)

```

```

119     arquivo = fopen("../MQL4/Files/MN1.csv","rt");
120     else if( strcmp(nomeTipoGrafico,"W1") == 0)
121         arquivo = fopen("../MQL4/Files/W1.csv","rt");
122     else
123         printf("Erro, tabela nao encontrada\n");
124
125     for(i = 0; i <= quantidadeVelas; i++){
126         fscanf(arquivo, "%lf",&cotacao[i]);
127
128         if(resistencia < cotacao[i])
129             resistencia = cotacao[i];
130     }
131
132     fclose(arquivo);
133     return resistencia;
134 }
135
136 double calculoRegressaoFibonacci(double fatorDeRegressao, int quantidadeVelas){
137     double variacaoDePontos;
138     double fibonacci;
139
140     if(tendencia>0){
141         variacaoDePontos =
calculoSuporte(quantidadeVelas)-calculoResistencia(quantidadeVelas);
142         fibonacci = (variacaoDePontos*fatorDeRegressao) +
calculoResistencia(quantidadeVelas);
143         return fibonacci;
144     }
145
146     else{
147         variacaoDePontos =
calculoResistencia(quantidadeVelas)-calculoSuporte(quantidadeVelas);
148         fibonacci = (variacaoDePontos*fatorDeRegressao) +
calculoSuporte(quantidadeVelas);
149         return fibonacci;
150     }
151 }
152
153 void lerTendencia(){
154     FILE *arquivoTendencia;
155     double temporariaTendencia;
156

```

```

157  arquivoTendencia = fopen("../tendencia.txt", "rt");
158
159  if(arquivoTendencia == NULL){
160      printf("Arquivo nulo\n");
161  }
162
163  fscanf(arquivoTendencia, "%lf", &temporariaTendencia);
164  tendencia = temporariaTendencia;
165  //printf("Tendencia: %lf\n", tendencia);
166
167  fclose(arquivoTendencia);
168 }

```

### minimosQuadrados.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #define TAMANHO_STRING 50
4
5  double calculoCoeficienteLinear();
6  double calculoCoeficienteAngular();
7
8  char nomeRobo[TAMANHO_STRING], nomeTipoGrafico[2];
9
10 void detectaRoboETipoDeGrafico();
11 int quantidadeVelas;
12
13 int main(){
14     FILE *arquivo;
15
16     detectaRoboETipoDeGrafico();
17     printf("METODO MINIMOS QUADRADOS LIGADO\n");
18
19     arquivo = fopen("../minimosQuadradosResposta.txt", "wt");
20
21     fprintf(arquivo, "%f\n", calculoCoeficienteLinear(quantidadeVelas));
22     fprintf(arquivo, "%f\n", calculoCoeficienteAngular(quantidadeVelas));
23
24     //printf("Angular: %lf\n", calculoCoeficienteAngular(quantidadeVelas));
25     //printf("Linear: %lf\n", calculoCoeficienteLinear(quantidadeVelas));
26     fclose(arquivo);

```

```

27
28     return 0;
29 }
30
31 void detectaRoboETipoDeGrafico(){
32     FILE *arquivo;
33     char temporariaQuantidadeCandle[10];
34
35     arquivo = fopen("../criterioEntrada.txt","rt");
36
37     if(arquivo == NULL){
38         printf("Arquivo nulo\n");
39     }
40
41     fscanf(arquivo,"%s",nomeRobo);
42     //printf("nome robo: %s\n", nomeRobo);
43     fscanf(arquivo,"%s",nomeTipoGrafico);
44     //printf("Nome tipo grafico: %s\n", nomeTipoGrafico);
45     fscanf(arquivo,"%s",temporariaQuantidadeCandle);
46     //printf("Quantidade candles %s\n", temporariaQuantidadeCandle);
47     quantidadeVelas = atoi(temporariaQuantidadeCandle);
48
49     fclose(arquivo);
50 }
51
52 double calculoCoeficienteLinear(int quantidadeVelas){
53     double variacaoLinear;
54     double media_y,media_x;
55     FILE *arquivo;
56     double x[quantidadeVelas], y[quantidadeVelas];
57     double soma_x = 0, soma_y = 0;
58     double variacaoAngular;
59     int i,c;
60     double leituraCotacoes[quantidadeVelas+1];
61
62     if( (strcmp(nomeTipoGrafico,"M1")) == 0)
63         arquivo = fopen("../MQL4/Files/M1.csv","rt");
64     else if( (strcmp(nomeTipoGrafico,"M5")) == 0)
65         arquivo = fopen("../MQL4/Files/M5.csv","rt");
66     else if( (strcmp(nomeTipoGrafico,"M15")) == 0)
67         arquivo = fopen("../MQL4/Files/M15.csv","rt");
68     else if( (strcmp(nomeTipoGrafico,"M30")) == 0)

```



```

69     arquivo = fopen("../MQL4/Files/M30.csv", "rt");
70     else if( (strcmp(nomeTipoGrafico, "H1")) == 0)
71         arquivo = fopen("../MQL4/Files/H1.csv", "rt");
72     else if( (strcmp(nomeTipoGrafico, "H4")) == 0)
73         arquivo = fopen("../MQL4/Files/H4.csv", "rt");
74     else if( (strcmp(nomeTipoGrafico, "D1")) == 0)
75         arquivo = fopen("../MQL4/Files/D1.csv", "rt");
76     else if( (strcmp(nomeTipoGrafico, "MN1")) == 0)
77         arquivo = fopen("../MQL4/Files/MN1.csv", "rt");
78     else if( (strcmp(nomeTipoGrafico, "W1")) == 0)
79         arquivo = fopen("../MQL4/Files/W1.csv", "rt");
80     else
81         printf("Erro, tabela nao encontrada\n");
82
83     for(c=0; c<= quantidadeVelas; c++){
84         fscanf(arquivo, "%lf", &leituraCotacoes[c]);
85     }
86
87     for(c=0; c<= quantidadeVelas; c++){
88         x[c] = leituraCotacoes[c];
89         y[c] = leituraCotacoes[c+1];
90     }
91
92     for(i = 0; i < quantidadeVelas; i++){
93         soma_x = soma_x + x[i];
94         soma_y = soma_y + y[i+1];
95     }
96
97     media_x=soma_x/quantidadeVelas;
98     media_y=soma_y/quantidadeVelas;
99
100     variacaoLinear= media_y - (calculoCoeficienteAngular(quantidadeVelas) *
media_x);
101     fclose(arquivo);
102
103     return variacaoLinear;
104 }
105
106 double calculoCoeficienteAngular(int quantidadeVelas){
107     FILE *arquivo;
108     double x[quantidadeVelas], y[quantidadeVelas];
109     double soma_x = 0, soma_y = 0;

```

```

110  double variacaoAngular;
111  int i,c;
112  double numerador=0, denominador=0;
113  double leituraCotacoes[quantidadeVelas+1];
114  double media_x, media_y;
115
116  if( (strcmp(nomeTipoGrafico,"M1")) == 0)
117      arquivo = fopen("../MQL4/Files/M1.csv","rt");
118  else if( (strcmp(nomeTipoGrafico,"M5")) == 0)
119      arquivo = fopen("../MQL4/Files/M5.csv","rt");
120  else if( (strcmp(nomeTipoGrafico,"M15")) == 0)
121      arquivo = fopen("../MQL4/Files/M15.csv","rt");
122  else if( (strcmp(nomeTipoGrafico,"M30")) == 0)
123      arquivo = fopen("../MQL4/Files/M30.csv","rt");
124  else if( (strcmp(nomeTipoGrafico,"H1")) == 0)
125      arquivo = fopen("../MQL4/Files/H1.csv","rt");
126  else if( (strcmp(nomeTipoGrafico,"H4")) == 0)
127      arquivo = fopen("../MQL4/Files/H4.csv","rt");
128  else if( (strcmp(nomeTipoGrafico,"D1")) == 0)
129      arquivo = fopen("../MQL4/Files/D1.csv","rt");
130  else if( (strcmp(nomeTipoGrafico,"MN1")) == 0)
131      arquivo = fopen("../MQL4/Files/MN1.csv","rt");
132  else if( (strcmp(nomeTipoGrafico,"W1")) == 0)
133      arquivo = fopen("../MQL4/Files/W1.csv","rt");
134  else
135      printf("Erro, tabela nao encontrada\n");
136
137  for(c=0; c<= quantidadeVelas; c++){
138      fscanf(arquivo, "%lf",&leituraCotacoes[c]);
139  }
140
141  for(c=0; c<= quantidadeVelas; c++){
142      x[c] = leituraCotacoes[c];
143      y[c] = leituraCotacoes[c+1];
144  }
145
146  for(i = 0; i < quantidadeVelas; i++){
147      soma_x = soma_x + x[i];
148      soma_y = soma_y + y[i+1];
149  }
150
151  media_x=soma_x/quantidadeVelas;

```

```
152  media_y=soma_y/quantidadeVelas;
153
154  for (i = 0; i < quantidadeVelas; ++i)
155  {
156      numerador+=(x[i]-media_x)*(y[i]-media_y);
157      denominador+= (x[i]-media_x)*(x[i]-media_x);
158  }
159  variacaoAngular=numerador/denominador;
160  fclose(arquivo);
161
162  return variacaoAngular;
163 }
```



## APÊNDICE H – Componente Lógico

## aprendiz.pl

```
1 :-dynamic metodosNumericos/3.
2
3 :-include('baseConhecimentoMock.pl').
4
5 % Função principal para gerar o executável
6 main:-
7     findall(CLVencedora, metodosNumericos(ganhou, X, Y, CLVencedora), Resposta),
8     write("Lista correlacoes "), write(Resposta), nl,
9     qtde(Resposta, QuantidadeCorrelacoes),
10    write("Quantidade correlacoes "), write(QuantidadeCorrelacoes), nl,
11    somaCorrelacoes(Resposta, Soma),
12    write("Quantidade correlacoes "), write(Soma), nl,
13    mediaCorrelacoes(MediaCorrelacoes),
14    write("Media correlacoes vencedoras: "), write(MediaCorrelacoes),
15    %tell abre o arquivo para escrita
16    open('respostaProlog.txt',write,Arquivo),
17    write(Arquivo, MediaCorrelacoes),
18    close(Arquivo),
19    nl, told.
20
21 % Soma de todos os elementos da lista de correlações
22 somaCorrelacoes([],0).
23 somaCorrelacoes([Elem|Cauda],Soma):-somaCorrelacoes(Cauda,Proximo),Soma is
Elem+Proximo.
24
25 % Quantidade de elementos de uma lista.
26 qtde([],0).
27 qtde([_|Tamanho],Soma):-qtde(Tamanho,Proximo),Soma is 1+Proximo.
28
29 % Pega a média das correlações vencedoras
30 mediaCorrelacoes(Media):-
31     findall(CLVencedora, metodosNumericos(ganhou, X, Y, CLVencedora), Resposta),
32     somaCorrelacoes(Resposta, Soma),
33     qtde(Resposta, QuantidadeCorrelacoes),
34     Media is Soma/QuantidadeCorrelacoes.
```

## baseConhecimento.pl

```
1 metodosNumericos(ganhou, venda, 0.23, 0.9).
2 metodosNumericos(ganhou, venda, 0.38, 0.9).
3 metodosNumericos(ganhou, venda, 0.62, 0.9).
4 metodosNumericos(ganhou, compra, 0.23, 0.9).
5 metodosNumericos(ganhou, compra, 0.38, 0.9).
6 metodosNumericos(ganhou, compra, 0.62, 0.9).
7 metodosNumericos(perdeu, venda, 0.23, 0.8).
8 metodosNumericos(perdeu, venda, 0.38, 0.8).
9 metodosNumericos(perdeu, venda, 0.62, 0.8).
10 metodosNumericos(perdeu, compra, 0.23, 0.8).
11 metodosNumericos(perdeu, compra, 0.38, 0.8).
12 metodosNumericos(perdeu, compra, 0.62, 0.8).
```

### AprendizTeste.pl

```
1 %:- use_module(library(plunit)).
2 :- use_module(library(test_cover)).
3
4 :-dynamic aprendizTeste/0.
5
6 :-consult(aprendizMock).
7
8 :-include('baseConhecimentoMock.pl').
9 :-include('aprendizMock.pl').
10 %:-include('test_cover.pl').
11
12
13 :-begin_tests(aprendizMock).
14
15
16 test(qtde) :-
17     findall(CLVencedora, metodosNumericos(ganhau, X, Y, CLVencedora), Resposta),
18     qtde(Resposta, QuantidadeCorrelacoes),
19     assertion(QuantidadeCorrelacoes == 12).
20
21 test(somaCorrelacoes) :-
22     findall(CLVencedora, metodosNumericos(ganhau, X, Y, CLVencedora), Resposta),
23     somaCorrelacoes(Resposta, Soma),
24     assertion((Soma == 10.8000000000000002)).
25
26 test(mediaCorrelacoes) :-
27     findall(CLVencedora, metodosNumericos(ganhau, X, Y, CLVencedora), Resposta),
28     mediaCorrelacoes(MediaCorrelacoes),
29     assertion((MediaCorrelacoes == 0.9000000000000002)).
30 :-end_tests(aprendizMock).
```

### baseConhecimentoTeste.pl

```
1 :- use_module(library(plunit)).
2 :-consult(baseConhecimentoMock).
3 :-begin_tests(baseConhecimentoMock).
4
```



```
5 test(metodosNumericos) :-
6   metodosNumericos(ganhou, venda, 0.23, 0.9), !.
7 test(metodosNumericos) :-
8   metodosNumericos(ganhou, venda, 0.38, 0.9), !.
9 test(metodosNumericos) :-
10  metodosNumericos(ganhou, venda, 0.62, 0.9), !.
11
12 test(metodosNumericos) :-
13  metodosNumericos(ganhou, compra, 0.23, 0.9), !.
14 test(metodosNumericos) :-
15  metodosNumericos(ganhou, compra, 0.38, 0.9), !.
16 test(metodosNumericos) :-
17  metodosNumericos(perdeu, venda, 0.62, 0.8), !.
18
19 test(metodosNumericos) :-
20  metodosNumericos(perdeu, venda, 0.23, 0.8), !.
21 test(metodosNumericos) :-
22  metodosNumericos(perdeu, venda, 0.38, 0.8), !.
23 test(metodosNumericos) :-
24  metodosNumericos(perdeu, compra, 0.62, 0.8), !.
25
26 test(metodosNumericos) :-
27  metodosNumericos(perdeu, compra, 0.23, 0.8), !.
28 test(metodosNumericos) :-
29  metodosNumericos(perdeu, compra, 0.38, 0.8), !.
30 test(metodosNumericos) :-
31  metodosNumericos(ganhou, compra, 0.62, 0.9), !.
32
33 :-end_tests(baseConhecimentoMock).
34
35 :-run_tests.
```



## APÊNDICE I – Componente Multiagentes

## LeituraArquivo.java

```
1 package comportamentosComuns;
2
3 import java.io.FileNotFoundException;
4 import java.io.FileReader;
5 import java.io.IOException;
6 import java.util.ArrayList;
7 import java.util.Scanner;
8
9
10 public class LeituraArquivo{
11     static String correlacao;
12     static String tendencia;
13
14     public static String leituraCorrelacao() throws IOException{
15         Scanner scanner = new Scanner(new FileReader("../correlacaoResposta.txt"))
16         .useDelimiter("\\|\\n");
17         while (scanner.hasNext()) {
18             correlacao = scanner.next();
19         }
20         scanner.close();
21         return correlacao;
22     }
23
24     public static String leituraTendencia() throws IOException{
25         //Aqui vou ter que chamar um programa em C, e verificar o arquivo que foi reescrito
26         Scanner scanner = new Scanner(new FileReader("../tendencia.txt"))
27         .useDelimiter("\\|\\n");
28         while (scanner.hasNext()) {
29             tendencia = scanner.next();
30             //System.out.println("MINHA TENDENCIA: "+tendencia);
31         }
32         scanner.close();
33         return tendencia;
34     }
35
36     public static ArrayList<String> leituraFibonacci() throws FileNotFoundException{
37         ArrayList<String> fibonacci = new ArrayList<String>();
```

```

38     Scanner scanner = new Scanner(new FileReader("../FibonacciResposta.txt"));
39
40     fibonacci.add(scanner.next());
41     fibonacci.add(scanner.next());
42     fibonacci.add(scanner.next());
43
44     return fibonacci;
45 }
46
47 public static String leituraMetodo() throws FileNotFoundException {
48     String metodo = new String();
49     Scanner scanner = new Scanner(
49     new FileReader("../criterioEntrada.txt")).useDelimiter("\\|\\n");
50
51     while (scanner.hasNext()){
52         metodo = scanner.next();
53     }
54     return metodo;
55 }
56
57 public static double lerAlavancaProlog(){
58     double alavanca;
59     Scanner scanner = null;
60
61     try {
62         scanner = new Scanner(
62         new FileReader("../prologResposta.txt")).useDelimiter("\\|\\n");
63     } catch (FileNotFoundException e) {
64         e.printStackTrace();
65     }
66     alavanca = Double.parseDouble(scanner.next());
67     return alavanca;
68 }
69
70 public static long lerTipoGrafico() {
71     long tempoEmMiliSegundos=0;
72     String tipoGrafico = new String();
73     Scanner scanner;
74     try {

```

```

75 scanner = new Scanner(new
FileReader("../criterioEntrada.txt")).useDelimiter("\\|\\n");
76 scanner.nextLine();
77 tipoGrafico = scanner.nextLine();
78 tempoEmMiliSegundos = converteTipoGraficoEmTempo(tipoGrafico);
79 } catch (FileNotFoundException e) {
80 e.printStackTrace();
81 }
82 return tempoEmMiliSegundos;
83 }
84
85 public static long converteTipoGraficoEmTempo(String tipoGrafico) {
86
87 if(tipoGrafico == "M1"){
88 return 60000;
89 }
90 else if (tipoGrafico == "M5") {
91 return 60000*5;
92 }
93 else return 60000*60;
94 }
95
96 }

```

## RegistrarNoDF.java

```

1 package comportamentosComuns;
2
3 import jade.core.behaviours.OneShotBehaviour;
4 import jade.domain.DFService;
5 import jade.domain.FIPAException;
6 import jade.domain.FIPAAgentManagement.DFAgentDescription;
7 import jade.domain.FIPAAgentManagement.ServiceDescription;
8
9 public class RegistrarNoDF extends OneShotBehaviour{
10 private static final long serialVersionUID = -5125123631192783579L;
11
12 private String tipo;

```

```

13  private String nome;
14
15  public RegistrarNoDF(String tipo, String nome) {
16      this.tipo = tipo;
17      this.nome = nome;
18  }
19
20  @Override
21  public void action() {
22
23      DFAgentDescription descricaoAgente = new DFAgentDescription();
24      descricaoAgente.setName(myAgent.getAID()); //Registra o nome do agente no DF
25
26      //Criando um servi  o
27      ServiceDescription servicoMetodoNumerico = new ServiceDescription();
28      servicoMetodoNumerico.setType(tipo);
29      servicoMetodoNumerico.setName(nome);
30      descricaoAgente.addServices(servicoMetodoNumerico);
31
32      //Registrando o agente no DF
33      try {
34          DFService.register(myAgent, descricaoAgente);
35          System.out.println("Registrado o Agente "+myAgent+" no DF");
36      } catch (FIPAException erro) {
37          erro.printStackTrace();
38      }
39
40
41  }
42
43 }

```

### RodarComandos.java

```

1  package comportamentosComuns;
2
3  import java.io.BufferedReader;
4  import java.io.IOException;

```

```

5 import java.io.InputStreamReader;
6
7 public class RodarComandos {
8
9     public static void rodarComandoNoTerminal(String comando) throws IOException{
10         Runtime.getRuntime().exec(comando);
11     }
12 }

```

### ConsultorAgente.java

```

1 package metodosNumericos;
2
3 import java.io.FileNotFoundException;
4
5 import comportamentosComuns.LeituraArquivo;
6 import comportamentosComuns.ProcurarCotacoes;
7 import jade.core.Agent;
8
9 public class ConsultorAgente extends Agent{
10     private static final long serialVersionUID = -1528819378039323293L;
11
12     public void setup(){
13         System.out.println("Iniciado o agente consultor");
14         try {
15             // addBehaviour(new ProcurarCotacoes(this,LeituraArquivo.lerTipoGrafico()));
16             addBehaviour(new ProcurarCotacoes(this,6000));
17         } catch (FileNotFoundException e) {
18             e.printStackTrace();
19         }
20
21     }
22 }

```



## APÊNDICE J – Componente MQL

## Código Componente MQL

```
1. #property copyright "Copyright 2014, Cleiton da Silva Gomes"
2. #property link      "http://www.softwarecsg.com.br"
3.
4. int stop_loss = 500;
5. int take_profit = 500;
6. int ticket;
7. string nome = "csg";
8.
9. int start(){
10.     int arquivo = FileOpen("respostaMultiagente.txt", FILE_CSV|FILE_WRITE, ';');
11.     bool compra, venda;
12.
13.     if(compra == true && OrdersTotal() == 0){
14.         realizaCompra();
15.     }
16.
17.     if(venda == true && OrdersTotal() == 0){
18.         realizaVenda();
19.     }
20.
21.     exibeInformacaoNaTela();
22.
23.     FileClose(arquivo);
24.
25.     return (0);
26.
27. }
28.
29. void realizaCompra(){
30.     RefreshRates();
31.     while (IsTradeContextBusy())
32.         Sleep(5);
33.     ticket= OrderSend(Symbol(),OP_BUY,ALAVANCAGEM,Ask,0,Ask - stop_loss*Point,
```

```

34.     Ask + take_profit*Point,nome,AccountNumber(),0,Yellow);
35. }
36.
37. void realizaVenda(){
38.     RefreshRates();
39.     while (IsTradeContextBusy()) Sleep(5);
40.     ticket= OrderSend(Symbol(),OP_SELL,ALAVANCAGEM,Bid,0,Bid + stop_loss*Point,
41.         Bid - take_profit*Point,nome,AccountNumber(),0,Red);
42.
43. }
44.
45. void exibeInformacaoNaTela(){
46.     double ponto_positivo, ponto_negativo;
47.
48.     for(int j=0; j < OrdersHistoryTotal();j++){
49.         OrderSelect(j,SELECT_BY_POS,MODE_HISTORY);
50.
51.         if(OrderSymbol()!=Symbol()) continue;
52.         if(OrderMagicNumber() != AccountNumber()) continue;
53.         if(OrderProfit() > 0){
54.             ponto_positivo++;
55.         }
56.         else{
57.             ponto_negativo++;
58.         }
59.     }
60.
61.     Comment(
62.         "Quantidade ordens positivas = ", ponto_positivo+"\n",
63.         "Quantidade ordens negativas = ", ponto_negativo+"\n",
64.         ""
65.     );
66. }

```



# Anexos



# ANEXO A – Template de Protocolo Experimental

## Template for a Laboratory Experiment Protocol

### 1. Change Record

This should be a list or table summarizing the main updates and changes embodied in each version of the protocol and (where appropriate), the reasons for these.

### 2. Background

- a) identify previous research on the topic.
- b) define the main research question being addressed by this study and the associated hypothesis and null hypothesis.
- c) identify any additional research questions that will be addressed along with the relevant hypotheses and null hypotheses.

### 3. Design

- a) determine the independent and dependent variables.
- b) identify any variables that will need to be controlled.
- c) identify the population to be studied (e.g. practitioners, students, novices, ...).
- d) describe how the participants will be selected (recruited).
- e) determine the form of the study (between-subject or within-subject).
- f) describe the objects of study and how these will be prepared (if necessary), for example how errors will be seeded in a class for a testing study etc.
- g) specify how the treatment will be allocated to participants, such as the randomization mechanism to be used.
- h) describe how the protocol is to be reviewed (e.g. by supervisor, domain expert, etc.).

### 4. Data Preparation and Collection

- a) describe how the material for the study will be prepared.
- b) define a data collection plan and how the dependent variable(s) will be measured.

- c) define how the data will be stored.

## 5. Analysis

- a) the plan should identify which data elements are used to address which research question and how the data elements will be combined to answer the question.
- b) describe any statistical forms or graphical forms to be used.
- c) assess the threats to validity (construct, internal, external)

## 6. Study Limitations

Specify residual validity issues including potential conflicts of interest (i.e. that are inherent in the problem, rather than arising from the plan).

## 7. Reporting

Identify target audience, ways of providing data (e.g. scatter plots)

## 8. Schedule

Give time estimates for all of the major steps