



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

Autores: Cleiton da Silva Gomes, Vanessa Barbosa Martins
Orientador: Dr. Ricardo Matos Chaim
Coorientadora: Dr^a. 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**

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^a. 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-

189 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, 7 de julho de 2015:

Dr. Ricardo Matos Chaim
Orientador

Dr^a. Milene Serrano
Coorientadora

Msc. Hilmer Rodrigues Neri
Convidado

Msc. Fabiana Freitas Mendes
Convidada

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 7^a 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 teve como objetivo desenvolver um software multiparadigma para operar de forma automatizada no Mercado de Moedas: o software InvestMVC. Foi adotada uma metodologia de pesquisa com base nos objetivos e com base nos procedimentos técnicos. Com base nos objetivos, foi realizada uma pesquisa exploratória e descritiva. Em relação aos procedimentos técnicos, foi utilizado um Estudo de Caso, pois, apesar deste trabalho ter tido uma abordagem quantitativa dos dados e envolvimento de Métodos Matemáticos para elaboração de estratégias financeiras, não é possível afirmar que os resultados podem ser generalizados para outros contextos, como por exemplo, para outro mercado que não seja o Mercado de Moedas no par de negociação euro-dólar. Para construção do software, foi realizada uma adaptação do Scrum, levando em consideração as atividades alocadas no cronograma com o desenvolvimento das Histórias de Usuário e suas tarefas. Por meio do Protocolo de Estudo de Caso, foi definido os passos metodológicos para cumprir os objetivos específicos e responder a questão de pesquisa deste trabalho. Os Métodos Matemáticos de operação do software InvestMVC e a comparação dos resultados monetários com o *expert* MQL foram os resultados mais significativos que auxiliaram a responder a questão de pesquisa. Outros resultados como arquitetura, testes unitários e análise estática de código-fonte, auxiliaram no desenvolvimento do trabalho. Também foram apresentadas funcionalidades do software InvestMVC, por exemplo, como criar uma conta, fazer login, criar um *expert*, visualizar o *expert* e ativar o mesmo para realizar operações automatizadas. Por fim, foi reportada a conclusão do trabalho.

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

Abstract

This study aimed to develop a multiparadigm software to operate automated form the Currency Market: InvestMVC software. A research methodology based on objective and based on the technical procedures was adopted. Based on the goals it was incorporated an exploratory and descriptive research. Regarding the technical procedures we used a Case Study, because although this work have had a quantitative data approach and involvement of Mathematical Methods for preparation of financial strategies, you can not say that the results can be generalized to other contexts, for example to other market than the Currency Market on the pair of euro-dollar trading. To conduct the research, an adaptation of Scrum was carried out, taking into account the activities allocated on schedule with the development of User Stories and tasks. Through the Case Study Protocol was defined methodological steps to meet specific goals and answer the research question of this work. Methods software operating Mathematical InvestMVC and the comparison of monetary results with expert MQL were the most significant results that helped to answer the research question. Other results as architecture, unit testing and static analysis of source code, also added a lot of value to work. It was also evidenced attributes InvestMVC software, such as creating an account, log in, create an expert, the expert view and enable it to perform automated operations. Finally, the conclusion was reported.

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.	12
Figura 7 – Indicador Estocástico.	14
Figura 8 – Equação de Média Móvel.	14
Figura 9 – Indicador Média Móvel	15
Figura 10 – Arquitetura de von Neumann.	16
Figura 11 – A essência do Paradigma Orientado a Objetos.	18
Figura 12 – Defeito x Erro x Falha.	24
Figura 13 – Níveis de teste e seu desenvolvimento.	25
Figura 14 – Intervalo para interpretação das métricas.	28
Figura 15 – Atividades da Pesquisa.	33
Figura 16 – Relatório de simulação no período de Agosto de 2012 a Agosto de 2013 do <i>expert</i> CorrelacaoPearson.mql	48
Figura 17 – Relatório de simulação no período de Agosto de 2013 a Agosto de 2014 do <i>expert</i> CorrelacaoPearson.mql	48
Figura 18 – Gráfico gerado pela simulação do <i>expert</i> CorrelacaoPearson.mql no pe- ríodo de Agosto de 2012 a Agosto de 2013	49
Figura 19 – Gráfico gerado pela simulação do <i>expert</i> CorrelacaoPearson.mql no pe- ríodo de Agosto de 2013 a Agosto de 2014	49
Figura 20 – Relatório de simulação no período de Agosto de 2012 a Agosto de 2013 do <i>expert</i> MinimosQuadrados.mql	50
Figura 21 – Relatório de simulação no período de Agosto de 2012 a Agosto de 2013 do <i>expert</i> MinimosQuadrados.mql	50
Figura 22 – Gráfico gerado pela simulação do <i>expert</i> MinimosQuadrados.mql no período de Agosto de 2012 a Agosto de 2013	51
Figura 23 – Gráfico gerado pela simulação do <i>expert</i> MinimosQuadrados.mql no período de Agosto de 2013 a Agosto de 2014	51
Figura 24 – Relatório de simulação no período de Agosto de 2012 s Agosto de 2013 do <i>expert</i> Fibonacci.mql	52
Figura 25 – Relatório de simulação no período agosto 2013-2014 do <i>expert</i> Fibo- nacci.mql	52

Figura 26 – Gráfico gerado pela simulação do <i>expert</i> Fibonacci.mql no período de Agosto de 2012 a Agosto de 2013	52
Figura 27 – Gráfico gerado pela simulação do <i>expert</i> Fibonacci.mql no período de Agosto de 2013 a Agosto de 2014	53
Figura 28 – Relatório de simulação no período de Agosto de 2012 a Agosto de 2013 do <i>expert</i> Estocastico.mql	53
Figura 29 – Relatório de simulação no período de Agosto de 2013 a Agosto de 2014 do <i>expert</i> Estocastico.mql	54
Figura 30 – Gráfico gerado pela simulação do <i>expert</i> Estocastico.mql no período de Agosto de 2012 a Agosto de 2013	54
Figura 31 – Gráfico gerado pela simulação do <i>expert</i> Estocastico.mql no período de Agosto de 2013 a Agosto de 2014	55
Figura 32 – Relatório de simulação no período de Agosto de 2012 a Agosto de 2013 do <i>expert</i> MediaMovel.mql	55
Figura 33 – Relatório de simulação no período de Agosto de 2013 a Agosto de 2014 do <i>expert</i> MediaMovel.mql	56
Figura 34 – Gráfico gerado pela simulação do <i>expert</i> MediaMovel.mql no período de Agosto de 2012 a Agosto de 2013	56
Figura 35 – Gráfico gerado pela simulação do <i>expert</i> MediaMovel.mql no período de Agosto de 2013 a Agosto de 2014	56
Figura 36 – Diagrama de Classe InvestMVC	58
Figura 37 – Diagrama de Classe InvestMVC componente Orientado a Objetos	59
Figura 38 – Diagrama de sequência InvestMVC componente Orientado a Objetos	60
Figura 39 – Componente Funcional InvestMVC	61
Figura 40 – Diagrama de Sequência do Componente Funcional InvestMVC	61
Figura 41 – Diagramas de Componentes e de Sequência do Componente Estruturado InvestMVC	62
Figura 42 – Diagrama de Classe do Componente Multiagente InvestMVC	64
Figura 43 – Diagrama de Sequência do Componente Lógico InvestMVC.	65
Figura 44 – Diagrama de Sequência do Componente MQL InvestMVC.	65
Figura 45 – Diagrama de Sequência InvestMVC	66
Figura 46 – Resultado da Suíte de Teste do Método Correlação Linear	67
Figura 47 – Resultado da Suíte de Teste do Método de Fibonacci	67
Figura 48 – Resultado da Suíte de Teste do Método Mínimos Quadrados	67
Figura 49 – Cobertura de código-fonte Componente Estruturado	68
Figura 50 – Suíte de teste da base aprendiz.pl	68
Figura 51 – Suíte de teste da base de conhecimento	68
Figura 52 – Cobertura de Código dos pacotes do Componente Multiagentes	69
Figura 53 – Suite de teste do Componente Multiagente	70

Figura 54 – Primeiro resultado da análise estática de código-fonte do componente Multiagente	71
Figura 55 – Segundo resultado da análise estática de código-fonte do componente Multiagente	71
Figura 56 – Gráfico de rendimento software InvestMVC	72
Figura 57 – Gráfico de rendimento do <i>expert</i> implementado em MQL	73
Figura 58 – Resultados monetários do <i>expert</i> MQL e InvestMVC	74
Figura 59 – Resultados monetários do <i>expert</i> MQL e InvestMVC em gráfico de barras	74
Figura 60 – Tela de boas vindas InvestMVC	77
Figura 61 – Criando um usuário no software InvestMVC	78
Figura 62 – Tela de <i>login</i> InvestMVC	78
Figura 63 – Suporte InvestMVC	79
Figura 64 – Tela de “Perguntas Frequentes”	79
Figura 65 – Conceitos básicos de mercado	80
Figura 66 – Criação de um <i>expert</i>	80
Figura 67 – Listagem de <i>experts</i> criados	81
Figura 68 – Tela de ativação de um <i>expert</i>	81
Figura 69 – Software InvestMVC em ação	82

Lista de tabelas

Tabela 1 – Atividades da pesquisa	33
Tabela 2 – Cronograma simplificado	35
Tabela 3 – Erro aceitável de cada Método Matemático	44
Tabela 4 – Exemplo de validação do paradigma Multiagente	44
Tabela 5 – Percentual de negociações com lucro dos Métodos Matemáticos	57
Tabela 6 – Horário de entrada no Mercado de Moedas <i>expert MQL</i> e software InvestMVC	75

Lista de abreviaturas e siglas

ACC	Acoplamento
ACCM	Complexidade Ciclomática
ANPM	Número de Parâmetros por Método
CPU	Unidade Central de Processamento
DIT	Herança
FOREX	<i>Foreign Exchange</i>
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

1	INTRODUÇÃO	1
1.1	Objetivos	2
1.2	Organização do Trabalho	2
2	REFERENCIAL TEÓRICO	5
2.1	Contexto Financeiro	5
2.1.1	Mercado de Moedas	5
2.1.2	Alavancagem	6
2.1.3	Suporte	6
2.1.4	Resistência	7
2.2	Métodos Matemáticos	8
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	12
2.2.5	Média Móvel	14
2.3	Paradigmas de Programação	15
2.3.1	Paradigma Procedural	15
2.3.2	Paradigma Orientado a Objetos	17
2.3.3	Paradigma Orientado a Agentes	20
2.3.4	Programação Declarativa	22
2.3.4.1	Paradigma Funcional	22
2.3.4.2	Paradigma Lógico	23
2.4	Teste de Software	23
2.4.1	Testes Unitários	26
2.4.2	Teste de integração	26
2.4.3	Teste de Aceitação	26
2.5	Qualidade de Software	27
2.5.1	Métricas de Qualidade de código-fonte	27
2.5.2	Análise Estática de código-fonte	28
2.5.3	Ferramentas de Análise Estática	28
3	METODOLOGIA DE PESQUISA	31
3.1	Classificação da Pesquisa	31
3.2	Atividades da Pesquisa	32
3.2.1	Descrição dos Objetivos das Atividades de Pesquisa	33

3.3	Cronograma	35
3.4	Execução da Pesquisa	38
3.5	Planejamento do Estudo de Caso	39
3.5.1	Estudos Anteriores	40
3.5.2	Projeto do Estudo de Caso	40
3.5.2.1	Projeto para Seleção dos Métodos Matemáticos	40
3.5.2.2	Projeto de Comparação dos Resultados Financeiros	41
3.5.3	Coleta de Dados	41
3.5.3.1	Coleta de Dados para Seleção dos Métodos Matemáticos	41
3.5.3.2	Coleta de Dados para Comparação dos Resultados Financeiros	42
3.5.4	Análise dos Dados	42
3.5.5	Validade do Plano do Estudo de Caso	42
3.5.5.1	Validade de Construção	42
3.5.5.2	Validade Externa	43
3.5.5.3	Validade Interna	43
3.5.5.4	Confiabilidade	43
3.5.6	Limitações do Estudo	45
4	RESULTADOS	47
4.1	Seleção dos Métodos Matemáticos	47
4.1.1	Implementação dos Métodos Matemáticos	47
4.1.2	Simulação do Método de Correlação Linear	47
4.1.3	Simulação do método de Mínimos Quadrados	49
4.1.4	Simulação do método de Fibonacci	51
4.1.5	Simulação do método de Estocástico	53
4.1.6	Simulação do método de Média Móvel	55
4.1.7	Definição dos métodos de operação software InvestMVC	57
4.2	Estruturas e Componentes do software InvestMVC	57
4.2.1	Componente Orientado a Objetos	58
4.2.2	Componente Cálculos Numéricos	60
4.2.3	Componente Funcional	60
4.2.4	Componente Estruturado	62
4.2.5	Componente Multiagente	63
4.2.6	Componente Lógico	64
4.2.7	Componente MQL	65
4.2.8	Fluxo de atividades	66
4.3	Testes unitários e cobertura de código-fonte	66
4.3.1	Componente Funcional	67
4.3.2	Componente Estruturado	67
4.3.3	Componente Lógico	68

4.3.4	Componente Multiagente	69
4.4	Análise Estática de Código-fonte	70
4.4.1	Definição das Métricas de Qualidade de Código-fonte	70
4.4.2	Qualidade de código-fonte do software InvestMVC	71
4.5	Comparação entre resultados monetários	71
4.5.1	Resultados monetários InvestMVC	72
4.5.2	Resultados monetários do <i>expert</i> MQL	73
4.5.3	Resultados monetários obtidos em MQL <i>versus</i> InvestMVC	73
4.5.4	Outros resultados associados ao rendimento monetário	75
5	O INVESTMVC	77
5.1	Tela de boas vindas InvestMVC	77
5.2	Realizando o login	77
5.3	Suporte InvestMVC	78
5.4	Criando um <i>expert</i> no software InvestMVC	80
5.5	O software InvestMVC em ação	81
6	CONCLUSÃO	83
 REFERÊNCIAS		87
7	GLOSSÁRIO	93
 APÊNDICES		95
APÊNDICE A – HISTÓRIAS DE USUÁRIO		97
APÊNDICE B – CRONOGRAMA INVESTMVC		105
APÊNDICE C – SUPORTE TECNOLÓGICO		111
APÊNDICE D – EXPERTS PARA ESTUDO DE CASO		117
APÊNDICE E – COMPONENTE OO		133
APÊNDICE F – COMPONENTE FUNCIONAL		137
APÊNDICE G – COMPONENTE ESTRUTURADO		145
APÊNDICE H – COMPONENTE LÓGICO		157
APÊNDICE I – COMPONENTE MULTIAGENTE		163

APÊNDICE J – COMPONENTE MQL	169
APÊNDICE K – ANÁLISE ESTÁTICA DE CÓDIGO-FONTE	173
APÊNDICE L – HISTÓRICO DE RESULTADOS	181
ANEXOS	185
ANEXO A – PROTOCOLO DE ESTUDO DE CASO	187

1 Introdução

Operar no mercado de moedas de forma manual é muito arriscado e, portanto, não recomendado, uma vez que esse mercado não é 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 MetaTrader¹ oferece as linguagens MQL4² (Paradigma Estruturado) e MQL5³ (Paradigma Orientado a Objetos) para construir *experts* que operem de forma automatizada.

A plataforma MetaTrader, entretanto, 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, considerou-se no 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. O software proposto foi implementado em diferentes linguagens de programação, padrões de projeto adequados, testes unitários orientados a uma abordagem multiparadigma e análise qualitativa de código-fonte. Um *expert* (implementado em linguagem MQL4 ou MQL5) ou um conjunto de *experts* são substituídos pelo software. Nesse último caso, o software tem a propriedade de controlar e/ou monitorar um ou mais *experts*.

Este trabalho, portanto, respondeu a seguinte questão de pesquisa: é possível desenvolver um software multiparadigma que substitua os *experts* convencionais do Mercado

¹ <<http://www.metatrader4.com/>>

² <<http://www.mql4.com/>>

³ <<https://www.mql5.com/>>

de Moedas?

1.1 Objetivos

Este trabalho teve como objetivo geral desenvolver o software multiparadigma InvestMVC, que utiliza Métodos Matemáticos para automação de estratégias financeiras no Mercado de Moedas e avaliar os benefícios advindos dessa abordagem.

Considerando o Mercado de Moedas e os paradigmas de programação Estruturado, Orientado a Objetos, Funcional, Lógico e Multiagentes, foram objetivos específicos deste trabalho:

1. Selecionar Métodos Matemáticos a serem implementados no software InvestMVC;
2. Caracterizar as estruturas e componentes do software InvestMVC;
3. Apurar a cobertura de código por meio de ferramentas que implementam testes unitários nos Paradigmas Estruturado (linguagem C), Multiagentes (linguagem Java), Lógico (linguagem Prolog) e Funcional (linguagem Haskell);
4. Realizar análise estática do código-fonte do componente Multiagente a partir de métricas de qualidade selecionadas;
5. 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 apresentados quais os tipos de testes mais praticados. 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 estudo de caso e o cronograma.

No capítulo 4, são apresentados os resultados, levando em consideração os objetivos específicos deste trabalho.

No capítulo 5, é apresentada o software InvestMVC, evidenciando como ele pode ser utilizado para operar no Mercado de Moedas, por meio da criação de *experts* e ativação dos mesmos.

Por fim, no capítulo 6 são apresentadas as considerações finais relacionadas a esse trabalho bem como os possíveis desdobramentos dele.

2 Referencial teórico

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

O referencial teórico deste trabalho irá apresentar conceitos relacionados ao Contexto Financeiro, Métodos Matemáticos, Paradigmas de Programação e Testes estáticos/dinâmicos para que seja possível propor uma solução para o problema desta 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 (0.001×1000). 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 (1.0×1000) ([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.

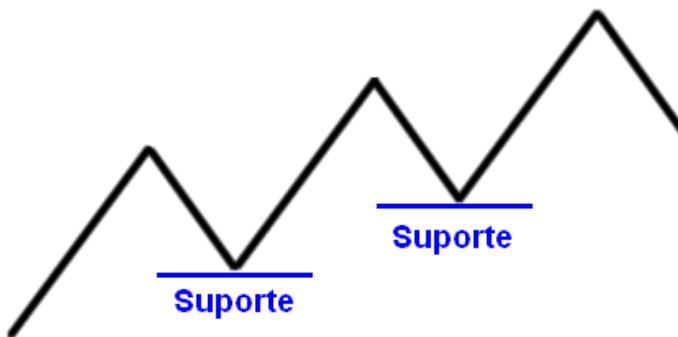


Figura 1 – Reta de Suporte.

Fonte: MATSURA (2006, pág. 22).

Suporte é uma região gráfica que após uma queda, os preços param e revertem 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 revertem no sentido contrário. É um ponto em que os investidores tendem a vender para terem 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

Método Matemático 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:

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

$$0 <= i > \infty \quad (2.5)$$

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).

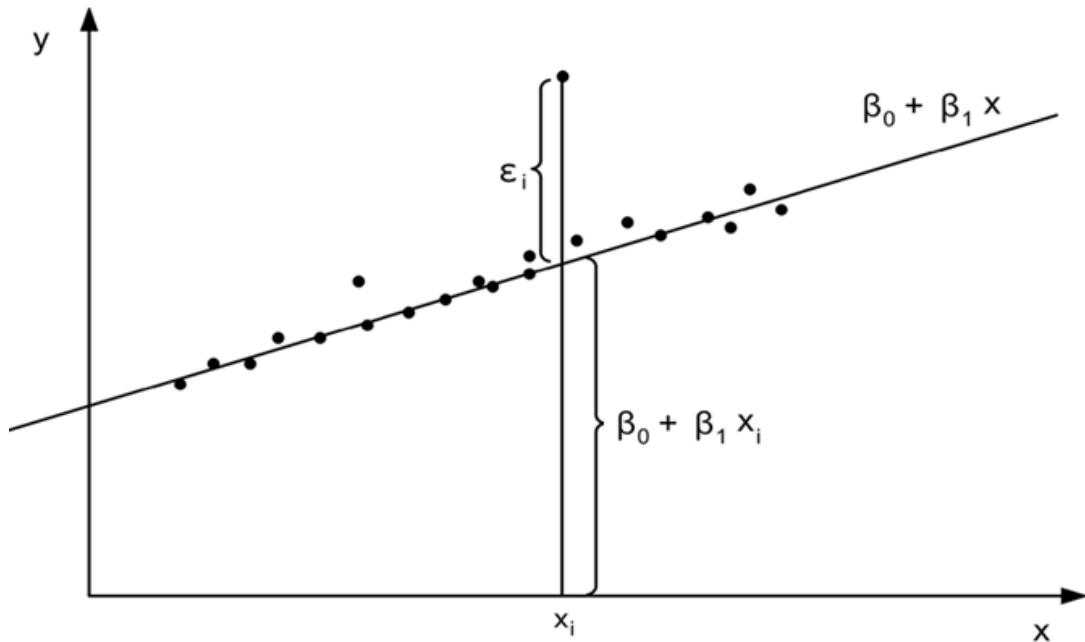


Figura 3 – Equação da reta Mínimos Quadrados.

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.

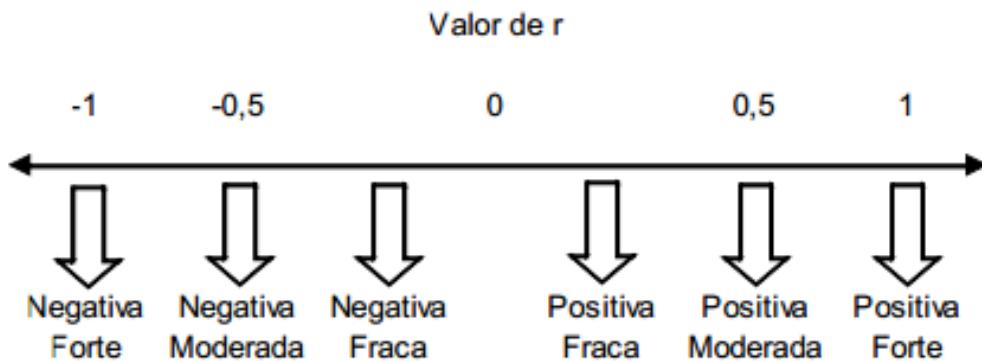


Figura 4 – Classificação da Correlação Linear.

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.

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

$$r = \frac{\text{Cov}(X, Y)}{s_x \cdot s_y} \quad s_x = \sqrt{\frac{\sum_{i=1}^k f_i (X_i - \bar{X})^2}{n}} \quad \text{e} \quad s_y = \sqrt{\frac{\sum_{i=1}^k f_i (Y_i - \bar{Y})^2}{n}}$$

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 método de 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:

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

$$\%D = \sum_1^3 FechHoje - Mínn \div \sum_1^3 Máxn - Mínn \quad (2.9)$$

Sendo que:

- %D = Estocástico Suave
- %K = Estocástico Rápido
- Mínn = menor preço durante os últimos "n" períodos
- Máxn = 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 Figura 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](#)).

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

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, conforme evidencia a linha azul da Figura 9 (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.

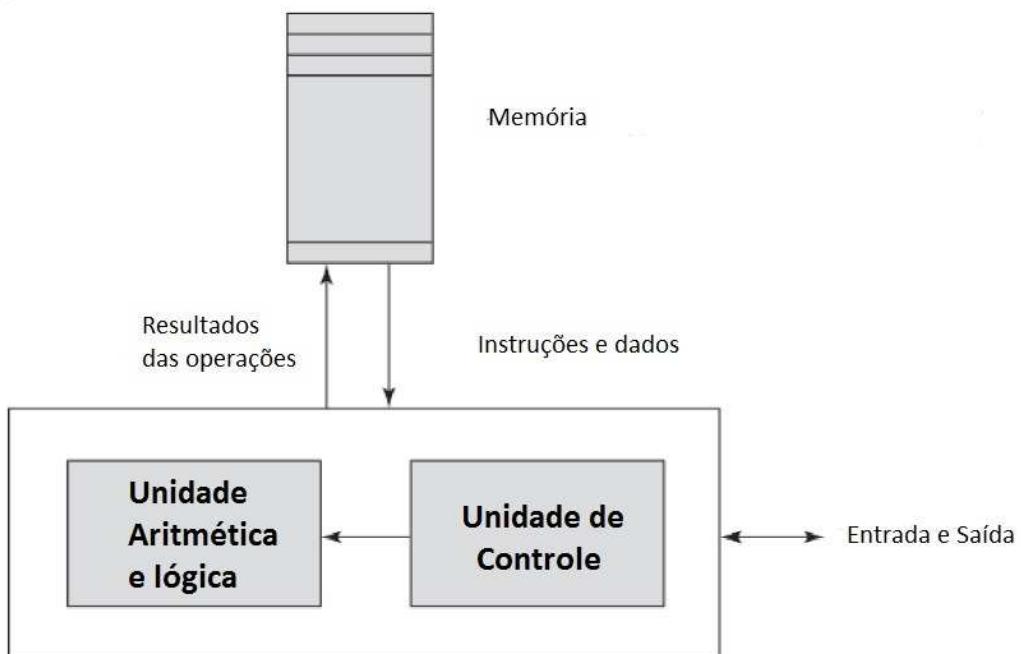


Figura 10 – Arquitetura de von Neumann.

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 TRANslator) 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, uma linguagem dominante na década de 90. Seu grande uso deve-se ao sucesso do Unix, um sistema operacional implementado em C ([RITCHIE, 1996](#)).

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" (algum ou algo), e esta abstração é expressa com a ajuda de uma linguagem de programação. 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).

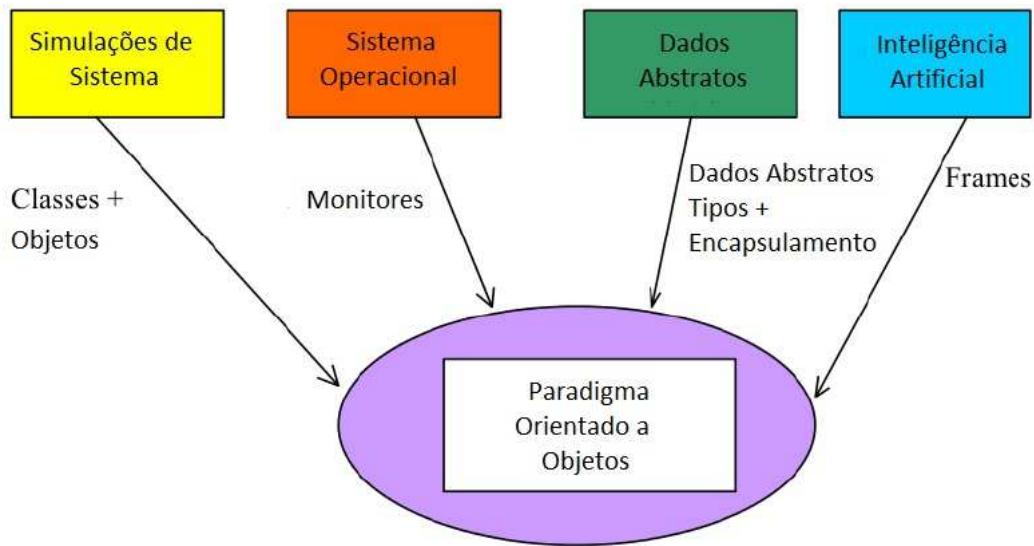


Figura 11 – A essência do Paradigma Orientado a Objetos.

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](#)).

Tratando-se de métodos polimórficos pode-se ter sobrecarga e a sobreescrita.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 diferem com: números de parâmetros passados, tipo de dados dos parâmetros e sequência dos dados passados como parâmetro. Já a sobreescrita é a declaração de um método na subclasse, que já está na classe mãe ([SINGH, 2014b](#)).

Os conceitos de sobrecarga e sobreescrita são constantemente confundidos. A sobrecarga acontece em tempo de compilação, enquanto a sobreescrita ocorre em tempo de execução. A sobrecarga feita na mesma classe, enquanto são necessárias classes mães e suas filhas para o uso sobreescrita, métodos privados sobreescritos, mas eles não podem ser sobreescritos. 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 os dados digitados, conteúdo de arquivos e pacotes de rede como entradas sensoriais e age sobre o ambiente por meio de 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 que um agente que habita um ambiente com outros agentes e processos, seja 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 conhecida como arquitetura 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 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 e por fim, tem-se suas intenções (*intentions*), que são suas ações que o agente 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, coordenar as 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, linguística, 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 linguística. 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 pequenos sistemas, tais como filtrados 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 têm pouco em comum. 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 têm 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 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 eletricidade participando do mercado eletrônico. Cada agente apresentava um comporta-

mento 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 ([APT, 1996](#), pág. 2).

2.3.4.1 Paradigma Funcional

O centro da programação funcional é a ideia de uma função. A linguagem de programação funcional fornece 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 ideias 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 ([HOOGLE, 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 (DOETS; EIJCK, 2004).

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, pode-se 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 sejam bastante curtas, de modo que a máquina possa 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, o principal objetivo é encontrar erros (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. 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, o programa é visto como uma caixa preta. Seu objetivo é ser completamente indiferente sobre o comportamento interno e estrutura do programa. Em vez disso, concentra-se 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 se tem total visibilidade do funcionamento interno do produto de software, especificamente, a lógica e a estrutura do código (WILLIAMS, 2006, pág. 1).

Os testes são realizados em diferentes níveis, envolvendo 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).

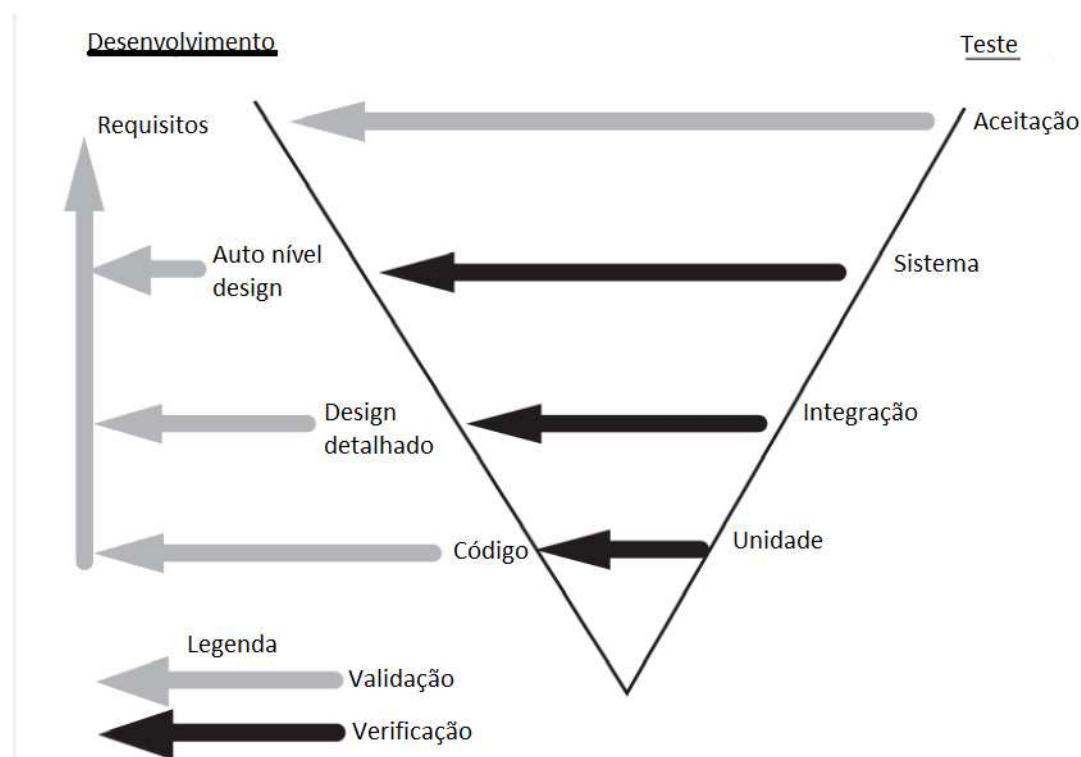


Figura 13 – Níveis de teste e seu desenvolvimento.

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 tipo de teste 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 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, testes de aceitação são realizados 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, inicialmente, 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. Fatores internos não são facilmente detectados por usuários, pois estão ligados a qualidade do código-fonte (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 dependem 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 das atividades de codificação e testes. São 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 qualitativo para 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 podem ser usados como parte de um processo de inspeção ([SOMMERVILLE, 2011](#), pág. 345).

Obter os dados e extraí-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 elas estão inseridas como, a linguagem do projeto a ser analisado ([MILLANI, 2013](#)).

3 Metodologia de pesquisa

Segundo Rodrigues (2007, pág. 2), a pesquisa deve conter um conjunto de abordagens, técnicas e processos para formular e resolver problemas do mundo real de maneira organizada e sistemática.

Para que uma pesquisa fique bem estruturada é necessário responder como os objetivos serão alcançados e como será realizada a resolução do problema de pesquisa. Para isso, deve-se classificar a pesquisa, identificar as atividades e estabelecer como as atividades serão executadas (FORCON, 2014).

3.1 Classificação da Pesquisa

Segundo Gil (2008, pág. 41), a metodologia de pesquisa é classificada através de critérios bem definidos com base em seus objetivos e procedimentos técnicos.

É usual classificar uma pesquisa com base em seus objetivos em três grandes grupos: exploratórias, descritivas e explicativas. Já em nível de procedimento técnico, a pesquisa pode ser classificada como Estudo de Caso, pesquisa-ação, survey, experimental, entre outras (GIL, 2008, pág. 41).

A pesquisa exploratória é utilizada pelo pesquisador para se familiarizar com um assunto pouco explorado. Ao decorrer ou no final da pesquisa exploratória, o pesquisador poderá estar apto para formular hipóteses (GIUDICE, 2010).

A pesquisa descritiva é usada quando se tem um conhecimento do assunto e se quer descrever um fenômeno. Hipóteses podem ser formuladas com base em conhecimentos prévios, procurando confirmá-las ou negá-las (FONSECA, 2002, pág. 21).

A pesquisa é classificada com base em procedimentos técnicos, podendo ser quantitativa ou qualitativa. A pesquisa quantitativa traduz em números os estudos realizados e se utiliza técnicas estatísticas para comprovar os fatos (RODRIGUES, 2007, pág. 9).

As pesquisas atuais revelam que o reconhecimento mútuo e integração das abordagens qualitativas e quantitativas já é reconhecido (SERAPIONI, 2005).

O Estudo de Caso é um estudo profundo, recomendável para temas muito complexos, em que é muito difícil gerar generalizações (FONSECA, 2002, pág. 33).

O Estudo de Caso vem sendo utilizado tanto em pesquisas exploratórias quanto descritivas e explicativas. É de sua natureza adotar na maioria dos casos, uma abordagem qualitativa, mas nada impede que o Estudo de Caso trabalhe com abordagens quantitativas (YIN, 2009, pág. 22).

Considerando os objetivos de estudo deste trabalho, foi incorporada uma pesquisa exploratória e uma pesquisa descritiva para evidenciar os benefícios da abordagem multiparadigma do software InvestMVC e da qualidade de código tanto a nível de testes unitários quanto a nível de análise estática de código-fonte.

Os objetivos específicos 2 (caracterizar as estruturas e componentes do software InvestMVC), 3 (apurar a cobertura de código por meio de ferramentas que implementam testes unitários) e 4 (realizar análise estática do código-fonte dos produtos de software a partir de métricas de qualidade previamente definidas) deste trabalho estão ligados a pesquisa exploratória e descritiva. Esses objetivos também estão ligados ao desenvolvimento e qualidade do produto. Conforme é descrito na seção 3.4, é utilizado o Scrum como metodologia de desenvolvimento do produto.

Em relação aos procedimentos técnicos, foi utilizado o Estudo de Caso, pois apesar deste trabalho ter uma abordagem quantitativa dos dados e envolvimento de Métodos Matemáticos para elaboração de estratégias, não é possível afirmar que os resultados podem ser generalizados para outros contextos, como por exemplo, para a bolsa de valores ou até mesmo um par de moedas diferente do euro-dólar.

3.2 Atividades da Pesquisa

É necessário evidenciar na metodologia de pesquisa quando começam as atividades e quais são as coordenadas para que as mesmas sejam cumpridas (FORCON, 2014).

A Figura 15 evidencia como foram organizadas as atividades deste trabalho (pesquisa e desenvolvimento).

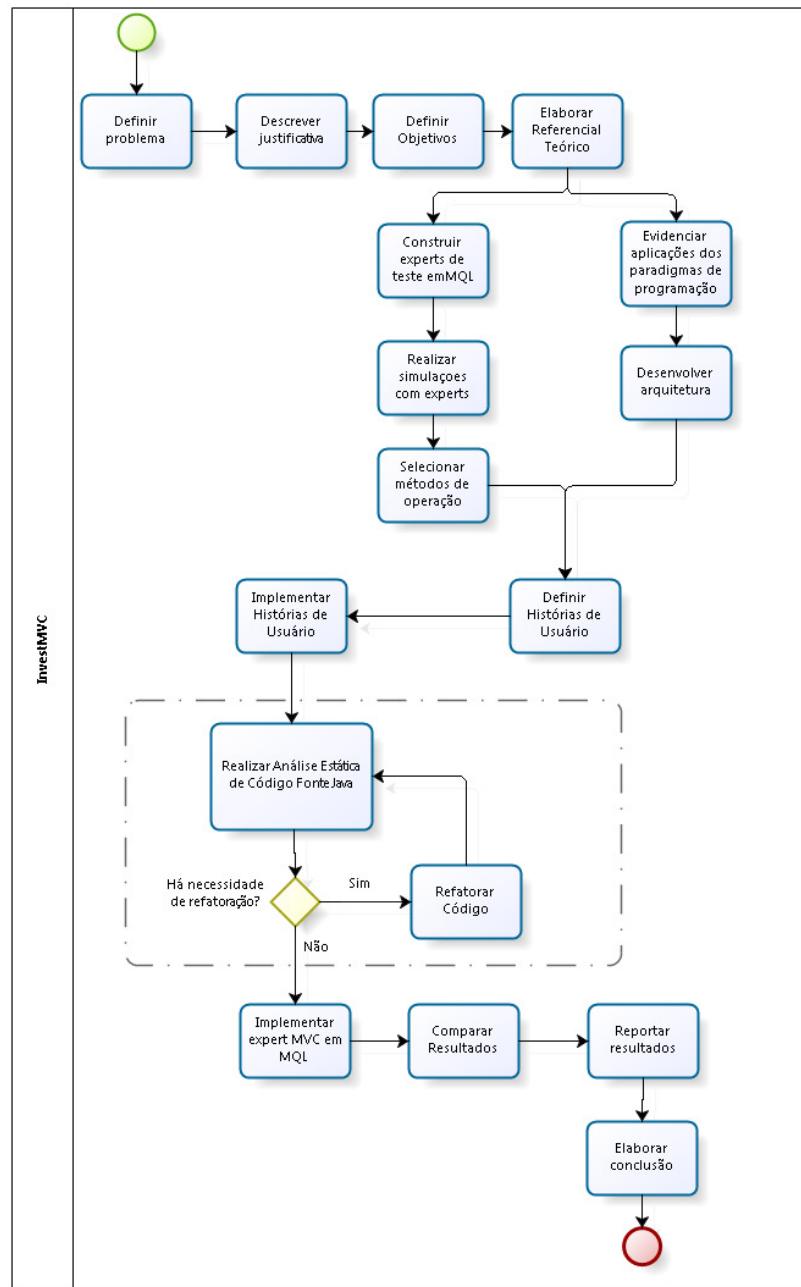


Figura 15 – Atividades da Pesquisa.

3.2.1 Descrição dos Objetivos das Atividades de Pesquisa

A Tabela 1 evidencia cada atividade da pesquisa e o objetivo de cada uma delas.

Tabela 1 – Atividades da pesquisa

Atividade	Objetivo
Definir problema	Definir o problema de pesquisa do trabalho.

Continuação na próxima página

Tabela 1 – Continuação da página anterior

Atividade	Objetivo
Descrever justificativa	Com base no problema de pesquisa, deve-se justificar a relevância do trabalho proposto.
Elaborar referencial teórico	Com base na literatura, descrever os conceitos chaves como Contexto Financeiro, Paradigmas de Programação e Qualidade de Software.
Construir <i>experts</i> em mql	Implementar <i>expert</i> Fibonacci.mql, MinimosQuadrados.mql, CorrelacaoLinear.mql, Estocastico.mql, MediaMovel.mql.
Realizar simulação com <i>experts</i>	Definir critérios de entrada e saída para cada <i>expert</i> e realizar a simulação de cada <i>expert</i> no Mercado de Moedas durante o período de 2 anos (agosto 2012-2014).
Selecionar métodos de operação	Selecionar os métodos (Fibonacci, Mínimos Quadrados, Correlação Linear, Estocástico ou Média Móvel) que obtiverem lucro durante o período de simulação.
Evidenciar aplicações dos paradigmas de programação	Evidenciar aplicações dos paradigmas: Funcional, Lógico, Estruturado e Multiagentes.
Desenvolver arquitetura	Desenvolver a arquitetura do software InvestMVC no intuito de evidenciar decisões sobre a organização do sistema.
Definir Histórias de Usuário	Definir o conjunto de funcionalidades e pontuar cada História, seguindo a sequência de Fibonacci para realizar a pontuação.
Implementar Histórias de Usuário	Desenvolver as Histórias de Usuário que foram definidas.
Realizar Análise Estática de código-fonte	Realizar a análise estática de código-fonte para aferir o nível de qualidade do software InvestMVC.
Implementar <i>expert</i> em MQL	Implementar toda a lógica do software InvestMVC em linguagem MQL.
Comparar resultados monetários entre o <i>expert</i> MQL e o software InvestMVC	Comparar os resultados monetários do <i>expert</i> em MQL com o software InvestMVC durante um tempo a se determinado de operação.

Continuação na próxima página

Tabela 1 – Continuação da página anterior

Atividade	Objetivo
Reportar resultados	Registrar os resultados da comparação do desempenho do software InvestMVC e do <i>expert</i> em MQL.
Elaborar conclusões	Desenvolver as conclusões do trabalho.

O *backlog* gerado com as histórias de usuário encontra-se no Apêndice A.

3.3 Cronograma

A Tabela 2 mostra quais foram as atividades em cada Sprint e a duração da mesma. O cronograma detalhado encontra-se no Apêndice B.

Tabela 2 – Cronograma simplificado

Sprint	Atividades	Data de início	Data de finalização
Sprint 1	1. Construir introdução 2. Implementar Métodos Numéricos	10/08/2014	31/08/2014
Sprint 2	1. Construir Referencial Teórico Paradigmas de Programação 2. Adaptar Métodos Numéricos 3. Prototipar View Projeto 4. Construir Referencial Teórico Métodos Numéricos	01/09/2014	15/09/2014

Continuação na próxima página

Tabela 2 – Continuação da página anterior

Sprint	Atividades	Data de início	Data de finalização
Sprint 3	<ul style="list-style-type: none"> 1. Refinar Referencial Teórico Paradigmas 2. Construir Referencial Teórico de Contexto Financeiro 3. Revisar Referencial Teórico Métodos Numéricos 	16/09/2014	30/09/2014
Sprint 4	<ul style="list-style-type: none"> 1. Realizar Estudo de Caso dos Métodos de Operação 2. Revisar Referencial Teórico Contexto Financeiro 3. Desenvolver <i>experts</i> em MQL4 	01/10/2014	15/10/2014
Sprint 5	<ul style="list-style-type: none"> 1. Descrever Metodologia de Pesquisa 2. Realizar Estudo de Caso dos Métodos de Operação 3. Evidenciar Aplicações Paradigmas de Programação 	16/10/2014	31/10/2014
Sprint 6	<ul style="list-style-type: none"> 1. Definir Backlog de Histórias de Usuário 2. Verificar Qualidade de código-fonte 	01/11/2014	10/11/2014

Continuação na próxima página

Tabela 2 – Continuação da página anterior

Sprint	Atividades	Data de início	Data de finalização
Sprint 7	1. Implementar US1 2. Implementar US5 3. Implementar US11 4. Implementar US12	09/03/2015	24/03/2015
Sprint 8	1. Implementar US2 2. Implementar US3 3. Implementar US4 4. Implementar US17	25/03/2015	15/04/2015
Sprint 9	1. Implementar US18 2. Implementar US7 3. Implementar US19 4. Verificar Qualidade de código-fonte	16/04/2015	07/05/2015
Sprint 10	1. Implementar US20 2. Implementar US21 3. Implementar US22	08/05/2015	29/05/2015
Sprint 11	Contingência	30/05/2015	19/05/2015

3.4 Execução da Pesquisa

Scrum é uma metodologia de desenvolvimento fundada na teoria do controle de processos empíricos. O empirismo afirma que conhecimento vem da experiência. As decisões são tomadas com base na experiência que se tem de um determinado assunto. Scrum emprega uma abordagem iterativa e incremental para otimizar a previsibilidade e controle de riscos da execução de um projeto. Existem três pilares que sustentam qualquer implementação de controle de processos empíricos: transparência, inspeção e adaptação (SCHWABER; SUTHERLAND, 2013, pág. 4).

Neste trabalho, vertentes defendidas pelo Scrum foram adaptadas e incorporadas à metodologia de pesquisa. Os produtos de trabalho foram alocados e desenvolvidos em Sprints (intervalo de tempo de 1 a 4 semanas). Durante a execução de cada Sprint, foram realizadas as adaptações necessárias para que os produtos de trabalho fossem produzidos com sucesso.

Na Sprint 1, foi construída a proposta de trabalho e foram implementados os Métodos Matemáticos em linguagem C que serviram como prova de conceito para viabilizar o desenvolvimento do trabalho.

Na Sprint 2, foi construído o referencial teórico dos Métodos Matemáticos e paradigmas de programação. Também foi feito um protótipo do software InvestMVC para evidenciar como seria a *view* da ferramenta com a dinâmica de construir um robô, selecionar e ativar um robô.

Na Sprint 3, foram refinados o referencial teórico feito na Sprint 2 e foi construído o referencial teórico do Contexto Financeiro.

Na Sprint 4, foi definida a metodologia de pesquisa e foram descritas algumas aplicações dos paradigmas de programação.

A Sprint 5 foi utilizada para atender os resultados do primeiro objetivo específico do trabalho (i.e. selecionar os Métodos Matemáticos do software InvestMVC). Para isso, foram utilizadas simulações para os métodos de fibonacci, correlação de Pearson, mínimos quadrados, média móvel e estocástico. Para realizar essas simulações, foram construídos *experts* para cada método matemático. Ao final das simulações, foram selecionados os Métodos Matemáticos de correlação de Pearson, fibonacci e mínimos quadrados.

A Sprint 6 foi utilizada para definir o backlog de Histórias de Usuário. As pontuações de cada História foram definidas com base na complexidade de cada uma. Nessa Sprint, também foi realizada a análise estática de código-fonte do paradigma estruturado e obteve-se uma qualidade de código aceitável.

Na Sprint 7, foi definido que seriam desenvolvidas as Histórias de Usuário US1 (Agente Correlação Linear), US5 (Agente gestor), US11(ativar *expert*) e US12 (desativar

expert). Entretanto, não foi possível terminar, as Histórias US11 e US12 nessa Sprint.

Na Sprint 8, foram desenvolvidas as Histórias US11 e US12 que não haviam sido terminadas na Sprint anterior. Foram desenvolvidas também as Histórias de Usuário US4 (Agente Tendência), US2 (Agente Fibonacci) e US17 (método Fibonacci em linguagem Haskell). Com essas Histórias de Usuários implementadas, já foi possível disponibilizar a primeira versão do software InvestMVC.

A Sprint 9 estava prevista para construir a US18 (método de Mínimos Quadrados em linguagem Haskell) e US7 (acompanhar retorno financeiro). Entretanto, foi refeito o planejamento dessa Sprint para atender demandas com maior prioridade. A Sprint 9 foi destinada para realizar alguns ajustes metodológicos e escrever os resultados obtidos com as Histórias de Usuário desenvolvidas nas Sprints anteriores. Além disso, foi elaborado um formulário para obter o *feedback* dos usuários que começaram a utilizar a primeira versão da ferramenta na Sprint anterior. Contudo, foi detectado que os resultados obtidos na execução dos códigos feitos em linguagem C e em linguagem Haskell não forneciam resultados similares. Como consequência nem todos os resultados previstos foram documentados nessa Sprint.

Devido aos acontecimentos ocorridos na Sprint 9, a Sprint 10 foi totalmente replanejada, criando uma nova Sprint para alocar as atividades que eram da Sprint 10.

Na Sprint 10, foi detectada a necessidade de excluir a US20 (retirar da base de conhecimento). Foi realizada a análise estática de código-fonte dos paradigmas multiagente e estruturado. Os resultados da qualidade de código e os resultados não documentados da Sprint anterior foram feitos.

Na Sprint 11, foi construída a US22 (*expert* no componente MQL), permitindo a integração entre o componente MQL e Multiagente *expert*. Após colocar o software InvestMVC para executar, percebeu-se que o software teve problemas, com "mortes" de alguns agentes. Esse problema foi contornado e o software voltou a rodar e coletar histórico das operações financeiras. Sendo assim, uma versão estável do software InvestMVC ficou pronta.

3.5 Planejamento do Estudo de Caso

Esta seção descreve o planejamento para seleção dos Métodos Matemáticos e comparação dos valores monetários do software InvestMVC com os *experts* tradicionais implementados em linguagem MQL que são, respectivamente, os objetivos específicos 1 (um) e 5 (cinco) deste trabalho. Os demais objetivos específicos são produtos de trabalho que dão suporte ao desenvolvimento do software InvestMVC e não possuem um rigor metodológico com o projeto, coleta e análise dos dados. Na seção Execução da pesquisa, é evidenciado

com clareza como foi a dinâmica para que todos os objetivos específicos deste trabalho fossem alcançados.

O planejamento deste Estudo de Caso é uma adaptação do template de protocolo de Estudo de Caso presente no Anexo A ([BRERETON et al., 2008](#)).

3.5.1 Estudos Anteriores

Não foi encontrado na literatura vigente nenhum estudo que compare os resultados monetários de Métodos Matemáticos implementados em linguagem MQL ou que defina quais são os métodos aceitáveis para serem implementados em um software multiparadigma para operar no Mercado de Moedas. Também não foi encontrado nenhum estudo que compare os resultados monetários de um software desenvolvido através de uma abordagem multiparadigma de programação com um software implementado em um único paradigma de programação.

3.5.2 Projeto do Estudo de Caso

Nesta seção é definido o projeto do Estudo de Caso para atender os objetivos específicos deste trabalho.

3.5.2.1 Projeto para Seleção dos Métodos Matemáticos

O objetivo do projeto do Estudo de Caso da seleção dos Métodos Matemáticos foi selecionar os métodos a serem implementados no software InvestMVC.

Para se selecionar os métodos de operação no Mercado de Moedas do software InvestMVC, foram definidas as seguintes variáveis e seus devidos valores:

- Alavancagem com valor de 0.25;
- Conta de simulação com valor inicial de 3.000 USD;
- Margem de negociação/alavancagem da conta igual a 1:500;
- *Stop loss* e *take profit* definido em 500 pontos;
- *Experts* programados em linguagem MQL4;
- Simulação realizada no mesmo período de tempo (Agosto de 2012 à Agosto de 2014);
- Simulações realizadas na mesma máquina e mesmo sistema operacional.

3.5.2.2 Projeto de Comparação dos Resultados Financeiros

Para realizar a comparação dos valores monetários do software InvestMVC com o software tradicional em linguagem MQL, foram definidas variáveis de operação iguais para ambos os produtos de software:

- Sistema operacional Linux (Ubuntu 12.04 ou superior);
- Alavancagem com valor de 0.1;
- Conta de simulação com valor inicial de 1.000 USD;
- Margem de negociação/alavancagem da conta igual a 1:500;
- *Stop loss* definidos de acordo com a estratégia do método Mínimos Quadrados;
- *Take profit* definidos de acordo com a estratégia do método Mínimos Quadrados;
- Quantidade de *candles* em 55;
- Critério de entrada da Correlação de Pearson maior ou igual a 0.9;
- Critério de entrada de Fibonacci em 0.38;
- Gráfico de negociação euro-dólar;
- Tipo de gráfico de negociação de 5 minutos;
- Início das negociações às 18:00 h de todo domingo;
- Fim das negociações às 18:00 h de toda sexta-feira;
- Tempo de negociação por dia de 24 horas.

3.5.3 Coleta de Dados

Esta seção define os atributos de coleta de dados para a seleção dos Métodos Matemáticos e comparação dos resultados financeiros do software InvestMVC com o software tradicional implementado em linguagem MQL.

3.5.3.1 Coleta de Dados para Seleção dos Métodos Matemáticos

Os dados coletados para seleção dos Métodos Matemáticos são: cotações do par de moedas euro-dólar (preço da cotação de abertura e fechamento); data do início e fim da ordem; tipo da ordem (compra ou venda); *stop loss*; *take profit* e resultado da operação (lucro ou prejuízo). Todos esses dados são fornecidos pelo histórico da ferramenta MetaTrader.

A ferramenta MetaTrader já fornece um relatório completo e detalhado do rendimento dos métodos que são colocados em simulação. Portanto, não tem necessidade de fazer um plano para armazenamento dos dados coletados. Os resultados das simulações (gráficos e relatórios de desempenho), são salvos em arquivos com extensões csv ou html.

3.5.3.2 Coleta de Dados para Comparaçāo dos Resultados Financeiros

Da mesma forma que a seleção dos Métodos Matemáticos, os dados coletados para comparação dos resultados financeiros são: cotações do par de moedas euro-dólar (preço da cotação de abertura e fechamento); data do início e fim da ordem; tipo da ordem (compra ou venda); *stop loss*; *take profit* e resultado da operação (lucro ou prejuízo). Esses resultados também são oferecidos nos históricos gerados pela ferramenta MetaTrader.

Os históricos dos resultados financeiros são salvos em planilhas ou em arquivos com extensão html. Sendo assim, é possível armazenar esses dados para fazer análises dos valores monetários extraídos dos históricos das operações.

3.5.4 Análise dos Dados

O critério para seleção dos Métodos Matemáticos e comparação dos valores monetários é o rendimento sob o capital investido. Os demais dados descritos na seção de coleta de dados (como cotações, data de início e fim da ordem), dão suporte e ajuda a explicar o motivo do rendimento financeiro ser maior ou menor de acordo com um valor de investimento. Entretanto, a análise dos demais dados não fazem parte do escopo deste trabalho.

O critério para análise e seleção dos Métodos Matemáticos foi o percentual de lucro sob o capital investido. Os Métodos Matemáticos que obtiveram 10% de lucro sob capital inicial, foram selecionados para serem implementados no software InvestMVC.

A análise dos resultados monetários é feita com base no histórico das operações. É considerado o resultado da operação (lucro ou prejuízo) como critério para realização da análise.

3.5.5 Validade do Plano do Estudo de Caso

O validade do plano do Estudo de Caso é definido por meio dos atributos de validade da construção, validade interna, validade externa e confiabilidade ([BRERETON et al., 2008](#)).

3.5.5.1 Validade de Construção

Segundo [Yin \(2009\)](#), a validade da construção está presente na fase de coleta de dados do Estudo de Caso. Posteriormente, os dados podem ser analisados para responder

a questão de pesquisa do trabalho.

Conforme descrito na Seção 3.5.3, a coleta de dados fornece o horário de entrada no mercado, horário de saída, tipo da operação, resultado da operação (lucro ou prejuízo), entre outras informações. O resultado da operação e tempo de entrada da operação são as duas variáveis que ficaram em análise neste trabalho e auxiliaram a responder a questão de pesquisa.

3.5.5.2 Validade Externa

Uma pesquisa possui validade externa quando ela permite ao pesquisador generalizar os resultados obtidos a outros contextos. A validade externa de uma pesquisa vai depender de se poder mostrar que os resultados obtidos nesta pesquisa não são dependentes da amostra ou da situação particular da pesquisa, mas que suas conclusões são verdadeiras também para outros contextos. A melhor estratégia para se garantir a validade externa de uma pesquisa é compor a amostra que será estudada com sujeitos que sejam selecionados aleatoriamente da população alvo, de modo que a amostra seja representativa da população (BANDEIRA, 2012).

Este trabalho não possui validade externa, visto que as negociações são exclusivas para o Mercado de Moedas no par de negociações euro-dólar com as demais configurações descritas na Seção 3.5.2.

3.5.5.3 Validade Interna

Para Yin (2009), a validade interna de um Estudo de Caso é gerada pela técnica de triangulação em que é usada várias fontes de dados e métodos de coleta para confirmar se os resultados convergem.

Neste trabalho, foram selecionadas duas fontes de dados para os cálculos dos Métodos Matemáticos. O *expert* implementado em linguagem MQL consumiu os dados de abertura e fechamento direto das funções da linguagem MQL (funções open e close). Já o componente Funcional e Estruturado consumiu os dados de abertura e fechamento gravados em arquivos com extensão .csv, que por sua vez, foram gravados por um *script*.

3.5.5.4 Confiabilidade

Falhas são inevitáveis na maioria dos contextos, mas as consequências das falhas podem ser evitadas pelo uso adequado de técnicas de tolerâncias a falhas. Um tipo de técnica comum é colocar dois componentes de software para realizarem a mesma atividade e gerar os mesmos resultados (WEBER, 2009).

Considerando o Mercado de Moedas, com negociações no par de moedas euro-dólar, foi definida a validade interna do Estudo de Caso considerando a técnica de tole-

rância a falhas dos paradigmas Funcional e Estruturado.

Para que o software InvestMVC realize uma operação de compra ou venda, devem ser realizados cálculos matemáticos pelos paradigmas Estruturado e Funcional. Ambos paradigmas realizam os mesmos cálculos para validar os resultados dos Métodos Matemáticos para realização das operações.

O paradigma Multiagente é responsável por receber os cálculos dos métodos de Mínimos Quadrados, Fibonacci e Correlação Linear através dos paradigmas Estruturado e Lógico. Feito isso, os resultados fornecidos são validados pelo paradigma Multiagente com um nível erro de acordo com a Tabela 3. O nível de erro foi definido de forma empírica, considerando a complexidade dos cálculos dos Métodos Matemáticos. Quanto maior a complexidade do cálculo matemático, maior foi o valor do nível de erro aceitado.

Tabela 3 – Erro aceitável de cada Método Matemático

Método Matemático	Nível de erro para validação
Correlação Linear	0.0010
Fibonacci	0.0001
Mínimos Quadrados	0.0100

A Tabela 4 evidencia possíveis resultados dos Métodos Matemáticos nos paradigmas Estruturado e Funcional e qual deve ser a resposta do paradigma Multiagente, levando em consideração o nível de erro para validação.

Tabela 4 – Exemplo de validação do paradigma Multiagente

Método Matemático	Cálculo para-paradigma Estruturado	Cálculo para-paradigma Funcional	Nível de erro para validação	Validado pelo paradigma Multiagente?
Correlação Linear	0.9000000	0.9100000	0.0010	NÃO
Fibonacci	0.6182578	0.6182577	0.0001	SIM
Mínimos Quadrados	0.912778	0.912779	0.0100	SIM

Como os resultados dos Métodos Matemáticos possuem uma validade do plano, não é necessário validar os resultados dos valores monetários, pois os mesmos são originados através das operações de compra ou venda que, por sua vez, são realizadas por meio dos cálculos matemáticos.

3.5.6 Limitações do Estudo

A linguagem MQL4 não possui suporte para teste unitário. Portanto, os *experts* tradicionais programados para este Estudo de Caso, não possuem testes automatizados. A linguagem MQL4 também não possui nenhuma ferramenta para realizar uma análise estática de código. Logo, não é possível determinar o nível de qualidade de código-fonte dos códigos desenvolvidos.

O software InvestMVC só executa no sistema operacional Linux e utiliza o Wine para rodar o MetaTrader, pois o MetaTrader só roda no sistema operacional Windows. Portanto, nada garante que o Wine não trave em algum momento e faça com que o software InvestMVC deixe de realizar uma operação de compra ou venda.

O simulador do MetaTrader possui código- fonte fechado. Portanto, não foi possível realizar nenhuma adaptação da ferramenta para auxiliar no desenvolvimento do software InvestMVC. Por exemplo, a ferramenta MetaTrader fornece ao usuário o histórico de operações, mas não é possível o software InvestMVC utilizar esse histórico. Nesse caso, tem que ser implementado o código para calcular o histórico dos resultados monetários.

4 Resultados

Este capítulo irá apresentar os resultados obtidos com a execução deste trabalho. Para tanto, optou-se mostrá-los considerando os objetivos mostrados na Seção 1.1. Deve-se observar que cada uma das seções deste capítulo corresponde a um objetivo específico definido para esta pesquisa. Para alcançar os resultados, foi necessário utilizar ferramentas de auxílio, elas encontram-se no Apêndice C.

4.1 Seleção dos Métodos Matemáticos

Nesta seção é evidenciado os resultados das simulações dos *experts* em linguagem MQL4 (produto da implementação dos Métodos Matemáticos), através de relatórios e gráficos gerados pela plataforma MetaTrader. Conforme estabelecido na metodologia de pesquisa, os Métodos Matemáticos que obtiveram 10% de rendimento sob o capital inicial, foram selecionados para serem implementados no software InvestMVC. Este resultado atende o objetivo específico 1 (selecionar Métodos Matemáticos a serem implementados) deste trabalho.

4.1.1 Implementação dos Métodos Matemáticos

Os métodos de Correlação Linear, Média Móvel, Mínimos Quadrados, Estocástico e Fibonacci foram implementados em linguagem MQL4 e assim foi construído um *expert* para cada método. Esses produtos de software receberam os nomes, respectivamente, CorrelacaoPearson.mql, MediaMovel.mql, MinimosQuadrados.mql, Estocastico.mql e Fibonacci.mql. Cada *expert* encontra-se no Apêndice D.

4.1.2 Simulação do Método de Correlação Linear

O *expert* CorrelacaoPearson.mql obteve o percentual de negociações com lucros de 56.86% no período de Agosto de 2012 a Agosto de 2013. Nesse período, o *expert* teve um lucro de 1981.60 USD. No período de Agosto de 2013 a Agosto de 2014, o percentual de negociações com lucros foi de 55.56% e obteve-se o lucro de 1119.05 USD. Os relatórios completos das simulações podem ser visualizados nas Figuras 16 e 17.

Barras em teste	7244	Ticks modelados	17462779
Mismatched charts errors	0		
Depósito Inicial	3000.00		
Lucro líquido total	1981.60	Lucro Bruto	9428.13
Fator de lucro	1.27	Compensação esperada	38.85
diminuição absoluta	431.18	Perda máxima	1804.15 (5.71%)
Total de negociações	51	Posições de Venda (ganhos %)	0 (0.00%)
		Negociações com Lucro (% do total)	29 (56.86%)
		Maior	Negociações com lucro
			337.32
		Média	Negociações com lucro
			325.11
		Máximo	Ganhos consecutivos (lucro em dinheiro)
			5 (1682.60)
		Máximo	ganhos consecutivos (contagem de ganhos)
			1682.60 (5)
		Média	ganhos consecutivos
			3

Configurações | Resultados | Gráfico | Relatório | Diário |

Figura 16 – Relatório de simulação no período de Agosto de 2012 a Agosto de 2013 do *expert* CorrelacaoPearson.mql

Barras em teste	7244	Ticks modelados	10799187
Mismatched charts errors	8		
Depósito Inicial	3000.00		
Lucro líquido total	1119.05	Lucro Bruto	5029.78
Fator de lucro	1.29	Compensação esperada	41.45
diminuição absoluta	610.55	Perda máxima	1350.33 (25.32%)
Total de negociações	27	Posições de Venda (ganhos %)	0 (0.00%)
		Negociações com Lucro (% do total)	15 (55.56%)
		Maior	Negociações com lucro
			337.50
		Média	Negociações com lucro
			335.32
		Máximo	Ganhos consecutivos (lucro em dinheiro)
			5 (1679.45)
		Máximo	ganhos consecutivos (contagem de ganhos)
			1679.45 (5)
		Média	ganhos consecutivos
			3

Configurações | Resultados | Gráfico | Relatório | Diário |

Figura 17 – Relatório de simulação no período de Agosto de 2013 a Agosto de 2014 do *expert* CorrelacaoPearson.mql

Foram gerados os gráficos de simulação de 2012 a 2013 e de 2013 a 2014, conforme ilustrado nas Figuras 18 e 19. É possível perceber que o método de Correlação de Pearson perde dinheiro em alguns períodos, mas os ganhos são superiores às perdas.

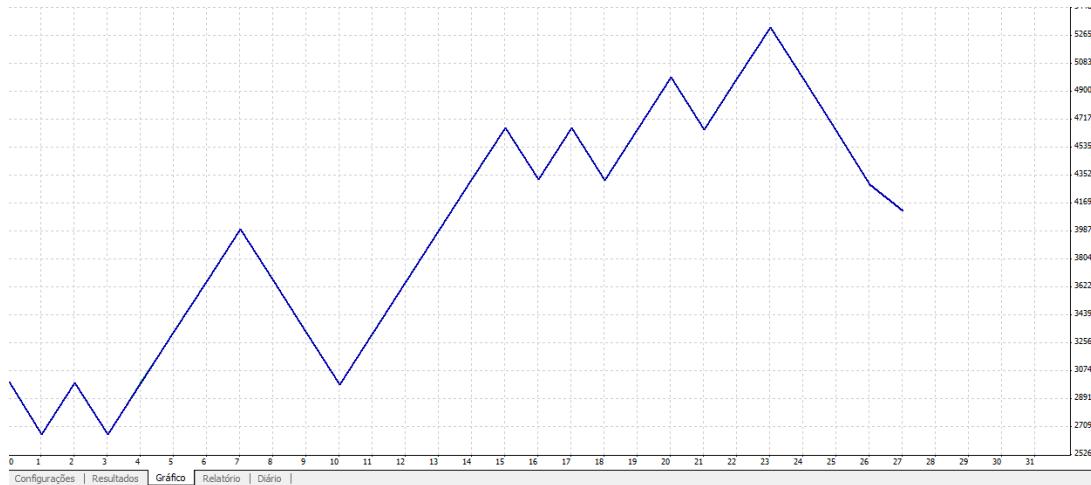


Figura 18 – Gráfico gerado pela simulação do *expert* CorrelacaoPearson.mql no período de Agosto de 2012 a Agosto de 2013

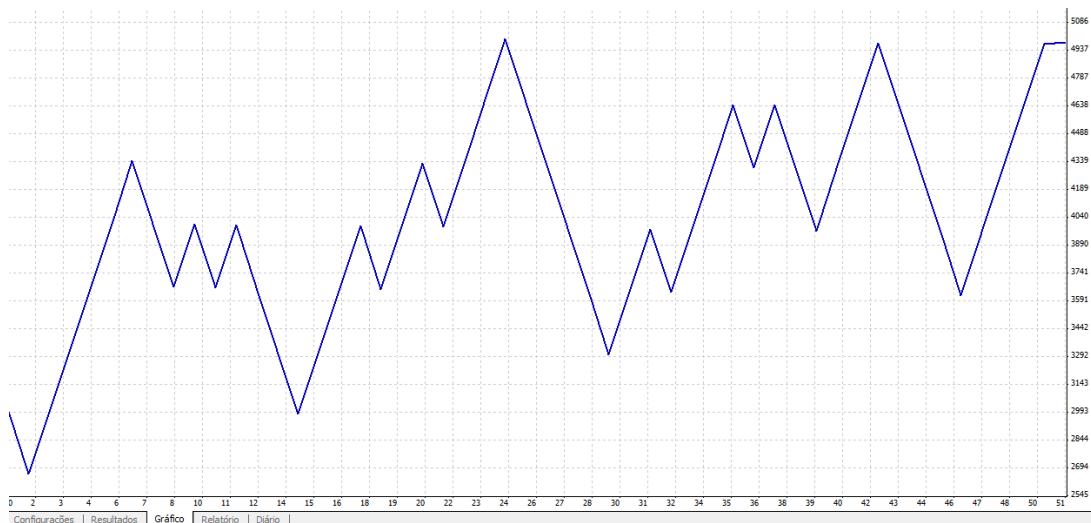


Figura 19 – Gráfico gerado pela simulação do *expert* CorrelacaoPearson.mql no período de Agosto de 2013 a Agosto de 2014

4.1.3 Simulação do método de Mínimos Quadrados

O *expert* MinimosQuadrados.mql obteve o percentual de negociações com lucros de 77.88% no período de Agosto de 2012 a Agosto de 2013, e o lucro nesse período foi de 1341.88 USD. No período de Agosto de 2013 a Agosto de 2014, o percentual de negociações com lucros foi de 85.71%, e obteve-se o lucro de 1026 USD. Os relatórios completos das simulações podem ser visualizados nas Figuras 20 e 21.

Barras em teste	7244	Ticks modelados	17462779
Mismatched charts errors	0		
Deposito Inicial	3000.00		
Lucro líquido total	1341.88	Lucro Bruto	5739.88
Fator de lucro	1.31	Compensação esperada	11.88
diminuição absoluta	570.47	Perda máxima	2394.14 (43.31%)
Total de negociações	113	Posições de Venda (ganhos %)	0 (0.00%)
		Negociações com Lucro (% do total)	88 (77.88%)
		Maior Negociações com lucro	764.22
		Média Negociações com lucro	65.23
		Máximo Ganhos consecutivos (lucro em dinheiro)	16 (2859.37)
		Máximo ganhos consecutivos (contagem de ganhos)	2859.37 (16)
		Média ganhos consecutivos	6

Configurações | Resultados | Gráfico | Relatório | Diário |

Figura 20 – Relatório de simulação no período de Agosto de 2012 a Agosto de 2013 do *expert* MinimosQuadrados.mql

Barras em teste	6645	Ticks modelados	10000081
Mismatched charts errors	8		
Deposito Inicial	3000.00		
Lucro líquido total	1026.33	Lucro Bruto	3057.94
Fator de lucro	1.51	Compensação esperada	13.33
diminuição absoluta	637.51	Perda máxima	1230.65 (25.26%)
Total de negociações	77	Posições de Venda (ganhos %)	0 (0.00%)
		Negociações com Lucro (% do total)	66 (85.71%)
		Maior Negociações com lucro	344.36
		Média Negociações com lucro	46.33
		Máximo Ganhos consecutivos (lucro em dinheiro)	19 (1032.92)
		Máximo ganhos consecutivos (contagem de ganhos)	1032.92 (19)
		Média ganhos consecutivos	9

Configurações | Resultados | Gráfico | Relatório | Diário |

Figura 21 – Relatório de simulação no período de Agosto de 2012 a Agosto de 2013 do *expert* MinimosQuadrados.mql

O *expert* MinimosQuadrados.mql, teve altos e baixos nas simulações durante os dois anos (2012 a 2013 e 2013 a 2014). Mas, no desempenho geral, conforme é evidenciado nos gráficos das Figuras 22 e 23, o *expert* teve um lucro acima de 10% do capital inicial.

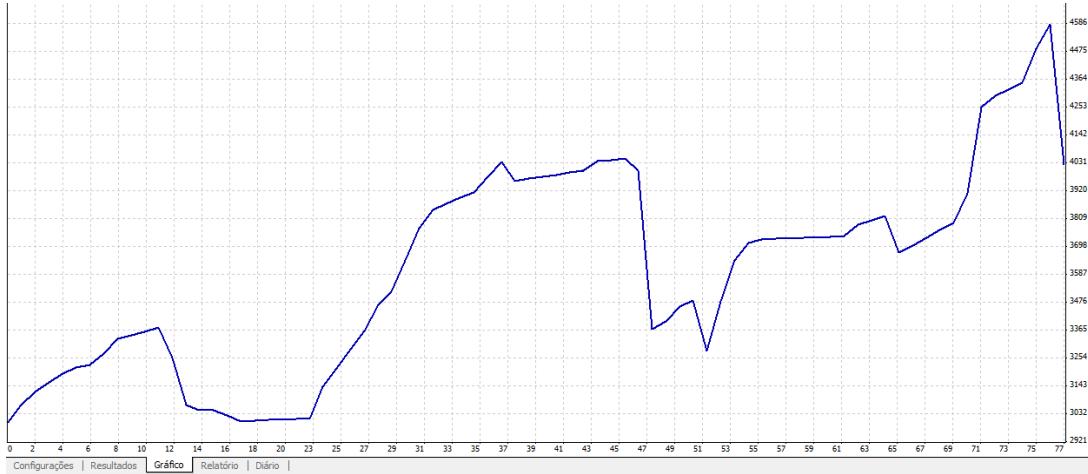


Figura 22 – Gráfico gerado pela simulação do *expert* MinimosQuadrados.mql no período de Agosto de 2012 a Agosto de 2013

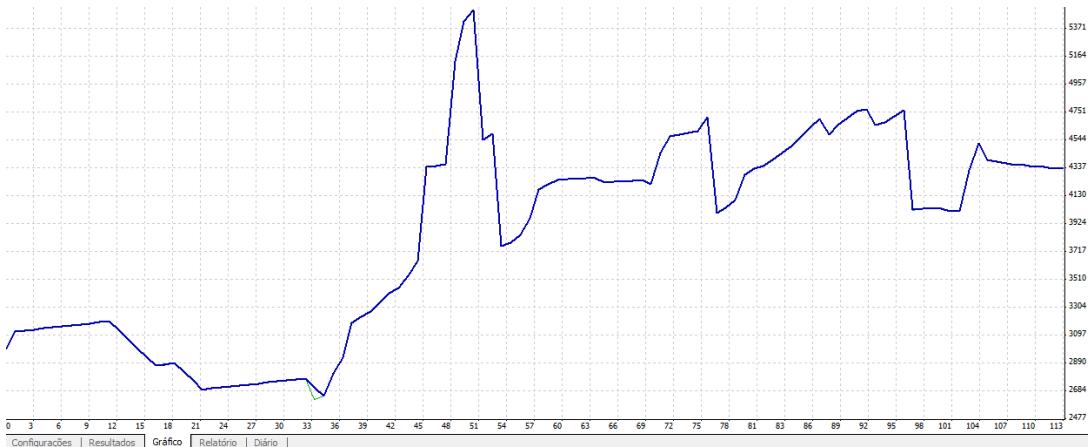


Figura 23 – Gráfico gerado pela simulação do *expert* MinimosQuadrados.mql no período de Agosto de 2013 a Agosto de 2014

4.1.4 Simulação do método de Fibonacci

O *expert* Fibonacci.mql obteve o percentual de negociações com lucros de 72.73% no período de Agosto de 2012 a Agosto de 2013, e o lucro nesse período foi de 341.20 USD. No período de Agosto de 2013 a Agosto de 2014, o percentual de negociações com lucros foi de 56.00%, e obteve-se o lucro de 659.05 USD. Apesar do percentual de acerto nesse período ter sido menor quando comparado ao período de agosto 2012-2013, o lucro obtido foi 51.77% maior. Isso se deve ao fato do *expert* ter negociado mais vezes (25 nesse período contra 11 no anterior) no período de Agosto 2013-2014.

Os relatórios completos das simulações podem ser visualizados nas Figuras 24 e 25.

Barras em teste	7244	Ticks modelados	10799187
Mismatched charts errors	8		
Depósito Inicial	3000.00		
Lucro líquido total	341.20	Lucro Bruto	798.35
Fator de lucro	1.75	Compensação esperada	31.02
diminuição absoluta	187.07	Perda máxima	347.08 (10.73%)
Total de negociações	11	Posições de Venda (ganhos %)	11 (72.73%)
		Negociações com Lucro (% do total)	8 (72.73%)
		Maior Negociação com lucro	99.92
		Média Negociações com lucro	99.79
		Máximo Ganhos consecutivos (lucro em dinheiro)	4 (399.18)
		Máximo ganhos consecutivos (contagem de ganhos)	399.18 (4)
		Média ganhos consecutivos	3

Configurações | Resultados | Gráfico Relatório Diário |

Figura 24 – Relatório de simulação no período de Agosto de 2012 s Agosto de 2013 do *expert Fibonacci.mql*

Barras em teste	7244	Ticks modelados	17462779
Mismatched charts errors	0		
Depósito Inicial	3000.00		
Lucro líquido total	659.05	Lucro Bruto	3379.25
Fator de lucro	1.24	Compensação esperada	26.36
diminuição absoluta	17.75	Perda máxima	1264.43 (27.00%)
Total de negociações	25	Posições de Venda (ganhos %)	0 (0.00%)
		Negociações com Lucro (% do total)	14 (56.00%)
		Maior Negociação com lucro	246.75
		Média Negociações com lucro	241.38
		Máximo Ganhos consecutivos (lucro em dinheiro)	5 (1229.90)
		Máximo ganhos consecutivos (contagem de ganhos)	1229.90 (5)
		Média ganhos consecutivos	2

Configurações | Resultados | Gráfico Relatório Diário |

Figura 25 – Relatório de simulação no período agosto 2013-2014 do *expert Fibonacci.mql*

É possível visualizar nos gráficos das simulações os lucros de capital que o *expert Fibonacci.mql* gerou, conforme ilustrado nas Figuras 26 e 27.

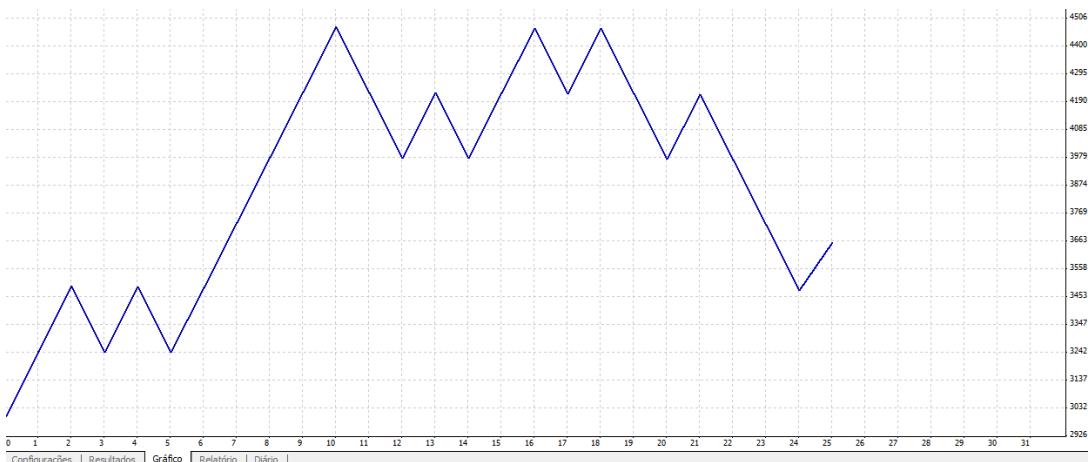


Figura 26 – Gráfico gerado pela simulação do *expert Fibonacci.mql* no período de Agosto de 2012 a Agosto de 2013

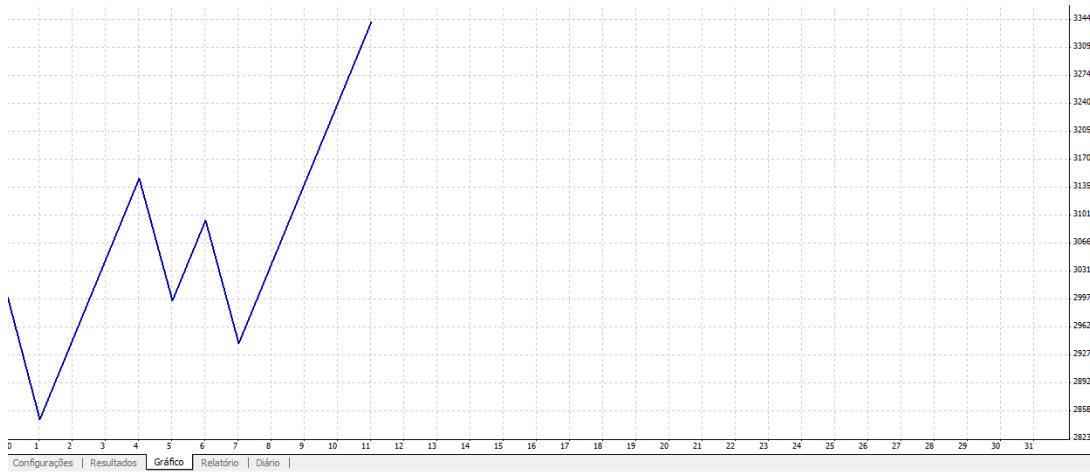


Figura 27 – Gráfico gerado pela simulação do *expert* Fibonacci.mql no período de Agosto de 2013 a Agosto de 2014

4.1.5 Simulação do método de Estocástico

O *expert* Estocastico.mql obteve o percentual de negociações com lucros de 47.47% no período agosto 2012-2013. Portanto, o percentual de negociações com perdas foi de 52.53%. Nesse período, o *expert* teve um prejuízo de 1110.88 USD.

No período de Agosto de 2013 a Agosto de 2014, o percentual de negociações com lucros foi de 47.70% (percentual com perdas de 52.30%), e obteve-se o prejuízo de 459.17 USD. Os relatórios completos das simulações podem ser visualizados nas Figuras 28 e 29.

Barras em teste	7244	Ticks modelados	10799187
Mismatched charts errors	8		
Depósito Inicial	3000.00		
Lucro líquido total	-1110.88	Lucro Bruto	11891.60
Fator de lucro	0.91	Compensação esperada	-5.55
diminuição absoluta	1295.07	Perda máxima	2521.47 (59.66%)
Total de negociações	200	Posições de Venda (ganhos %)	99 (47.47%)
		Negociações com Lucro (% do total)	96 (48.00%)
		Maior Negociação com lucro	126.22
		Média Negociações com lucro	123.87
		Máximo Ganhos consecutivos (lucro em dinheiro)	6 (750.85)
		Máximo ganhos consecutivos (contagem de ganhos)	750.85 (6)
		Média ganhos consecutivos	2

Configurações | Resultados | Gráfico | Relatório | Diário |

Figura 28 – Relatório de simulação no período de Agosto de 2012 a Agosto de 2013 do *expert* Estocastico.mql

Barras em teste	6212	Ticks modelados	9423746
Mismatched charts errors	7		
Deposito Inicial	3000.00		
Lucro líquido total	-459.17	Lucro Bruto	10327.40
Fator de lucro	0.96	Compensação esperada	-2.64
diminuição absoluta	1085.30	Perda máxima	1755.08 (47.83%)
Total de negociações	174	Posições de Venda (ganhos %)	84 (44.05%)
		Negociações com Lucro (% do total)	83 (47.70%)
		Maior Negociação com Lucro	126.40
		Média Negociação com Lucro	124.43
		Máximo Ganhos consecutivos (lucro em dinheiro)	5 (624.20)
		Máximo ganhos consecutivos (contagem de ganhos)	624.20 (5)
		Média ganhos consecutivos	2

Configurações | Resultados | Gráfico | Relatório | Diário |

Figura 29 – Relatório de simulação no período de Agosto de 2013 a Agosto de 2014 do *expert* Estocastico.mql

É possível visualizar nos gráficos das simulações, as perdas de capital que o *expert* Estocastico.mql gerou, conforme ilustrado nas Figuras 30 e 31.

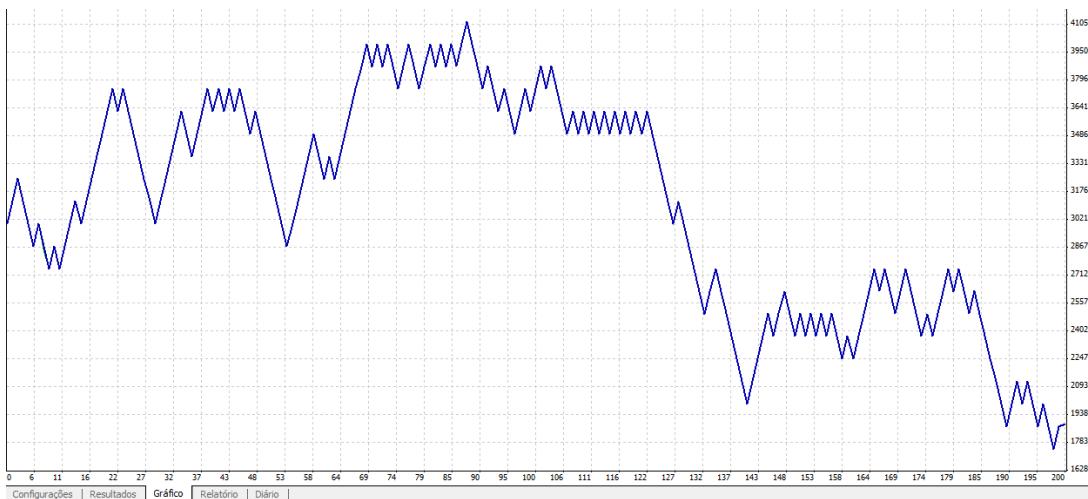


Figura 30 – Gráfico gerado pela simulação do *expert* Estocastico.mql no período de Agosto de 2012 a Agosto de 2013

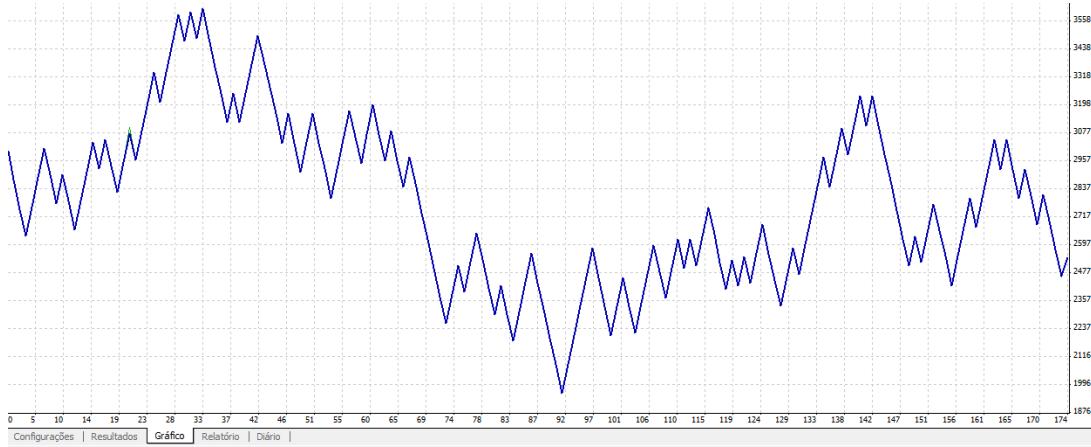


Figura 31 – Gráfico gerado pela simulação do *expert* Estocastico.mql no período de Agosto de 2013 a Agosto de 2014

4.1.6 Simulação do método de Média Móvel

O *expert* MediaMovel.mql obteve o percentual de negociações com lucros de 43.55% no período de Agosto de 2012 a Agosto de 2013. Portanto, o percentual de negociações com perdas foi de 56.45%. Nesse período, o *expert* obteve um prejuízo de 2987.00 USD.

No período de Agosto de 2013 a Agosto de 2014, o percentual de negociações com lucros foi de 48.48%, e o percentual de negociação com prejuízos foi de 51.82%. Foi obtido um prejuízo de 459.17 USD.

Os relatórios completos das simulações podem ser visualizados nas Figuras 32 e 33.

Barras em teste	7244	Ticks modelados	17462779
Mismatched charts errors	0		
Depósito Inicial	3000.00		
Lucro líquido total	-2987.00	Lucro Bruto	10124.18
Fator de lucro	0.77	Compensação esperada	-10.06
diminuição absoluta	2987.00	Perda máxima	3064.50 (99.58%)
Total de negociações	186	Posições de Venda (ganhos %)	82 (41.46%)
		Negociações com Lucro (% do total)	81 (43.55%)
		Maior Negociações com lucro	125.60
		Média Negociações com lucro	124.99
		Máximo Ganhos consecutivos (lucro em dinheiro)	4 (500.80)
		Máximo ganhos consecutivos (contagem de ganhos)	500.80 (4)
		Média ganhos consecutivos	2

Configurações | Resultados | Gráfico | Relatório (destacado) | Diário |

Figura 32 – Relatório de simulação no período de Agosto de 2012 a Agosto de 2013 do *expert* MediaMovel.mql

Barras em teste	7244	Ticks modelados	10799187
Mismatched charts errors	8		
Depósito Inicial	3000.00		
Lucro líquido total	-644.88	Lucro Bruto	12004.78
Fator de lucro	0.95	Compensação esperada	-3.26
diminuição absoluta	989.10	Perda máxima	1834.60 (47.71%)
Total de negociações	198	Posições de Venda (ganhos %)	96 (47.92%)
		Negociações com Lucro (% do total)	96 (48.48%)
		Maior Negociação com lucro	126.40
		Média Negociações com lucro	125.05
		Máximo Ganhos consecutivos (lucro em dinheiro)	5 (625.65)
		Máximo ganhos consecutivos (contagem de ganhos)	625.65 (5)
		Média ganhos consecutivos	2

Configurações | Resultados | Gráfico | Relatório **Relatório** | Diário |

Figura 33 – Relatório de simulação no período de Agosto de 2013 a Agosto de 2014 do *expert* MediaMovel.mql

É possível visualizar nos gráficos das simulações, as perdas de capital que o *expert* MediaMovel.mql gerou, conforme ilustrado nas Figuras 34 e 35.

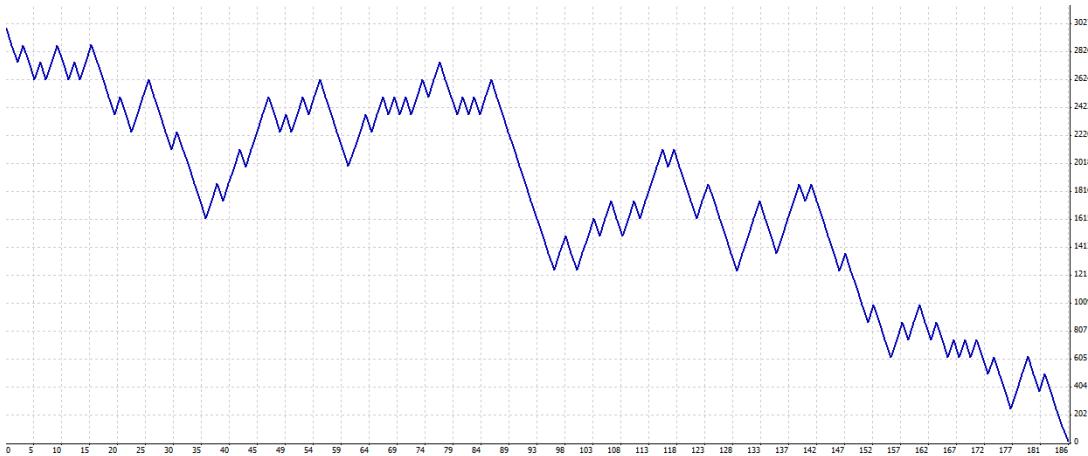


Figura 34 – Gráfico gerado pela simulação do *expert* MediaMovel.mql no período de Agosto de 2012 a Agosto de 2013

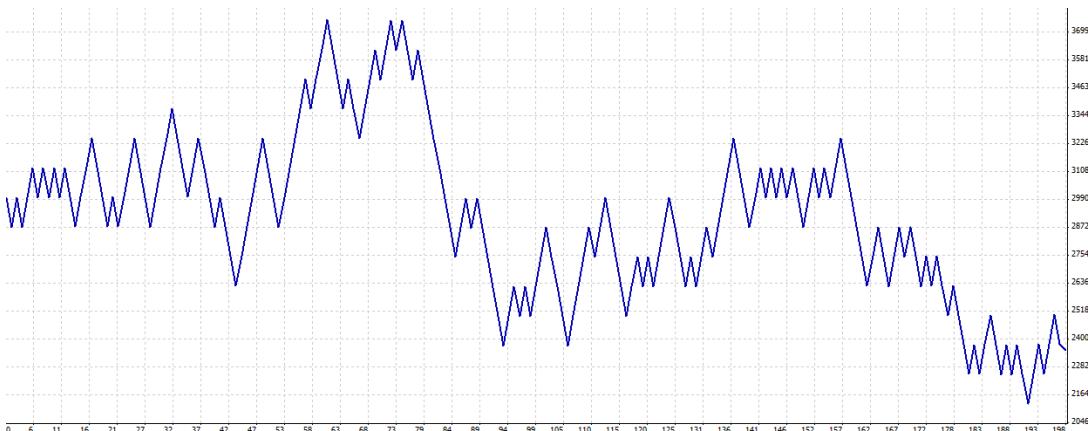


Figura 35 – Gráfico gerado pela simulação do *expert* MediaMovel.mql no período de Agosto de 2013 a Agosto de 2014

4.1.7 Definição dos métodos de operação software InvestMVC

Os Métodos Matemáticos de Correlação Linear, Fibonacci e Mínimos Quadrados tiveram êxito nos dois anos de simulação (Agosto 2012 a Agosto de 2014) e obtiveram lucro superior a 10% sob o capital inicial. Os métodos de Estocástico e Média Móvel tiveram prejuízo nos dois anos de simulação. Portanto, foram escolhidos os métodos de Correlação Linear, Fibonacci e Mínimos Quadrados como métodos de estratégia financeira do software InvestMVC.

A Tabela 5 evidencia os resultados financeiros de cada Método Matemático em percentual de negociações com lucro. Isso reforça o motivo da escolha dos métodos Correlação Linear, Fibonacci e Mínimos Quadrados.

Tabela 5 – Percentual de negociações com lucro dos Métodos Matemáticos

Método Matemático	Percentual de lucro em 2012 – 2013	Percentual de lucro em 2013 – 2014
Mínimos Quadrados	77.88%	85.71%
Fibonacci	72.73%	56.00%
Correlação Linear	56.86%	55.56%
Média Móvel	43.48%	48.48%
Estocástico	47.47%	47.70%

4.2 Estruturas e Componentes do software InvestMVC

Esta seção evidencia as estruturas e componentes do software InvestMVC. Este resultado atende o Objetivo Específico 2 (caracterizar as estruturas e componentes) deste trabalho. Uma arquitetura Orientada a Componentes possui o propósito de dividir para conquistar. Um grande problema é subdividido em partes menores e em seguida se desenvolve soluções mais elaboradas (CHEESMAN; DANIELS, 2001).

Por usar vários paradigmas, o software InvestMVC tem vários componentes, sendo cada qual implementado em um paradigma de programação mais adequado às necessidades do componente bem como contendo responsabilidades bem definidas. A Figura 36 procura ilustrar esses componentes usando a representação de digrama de classes.

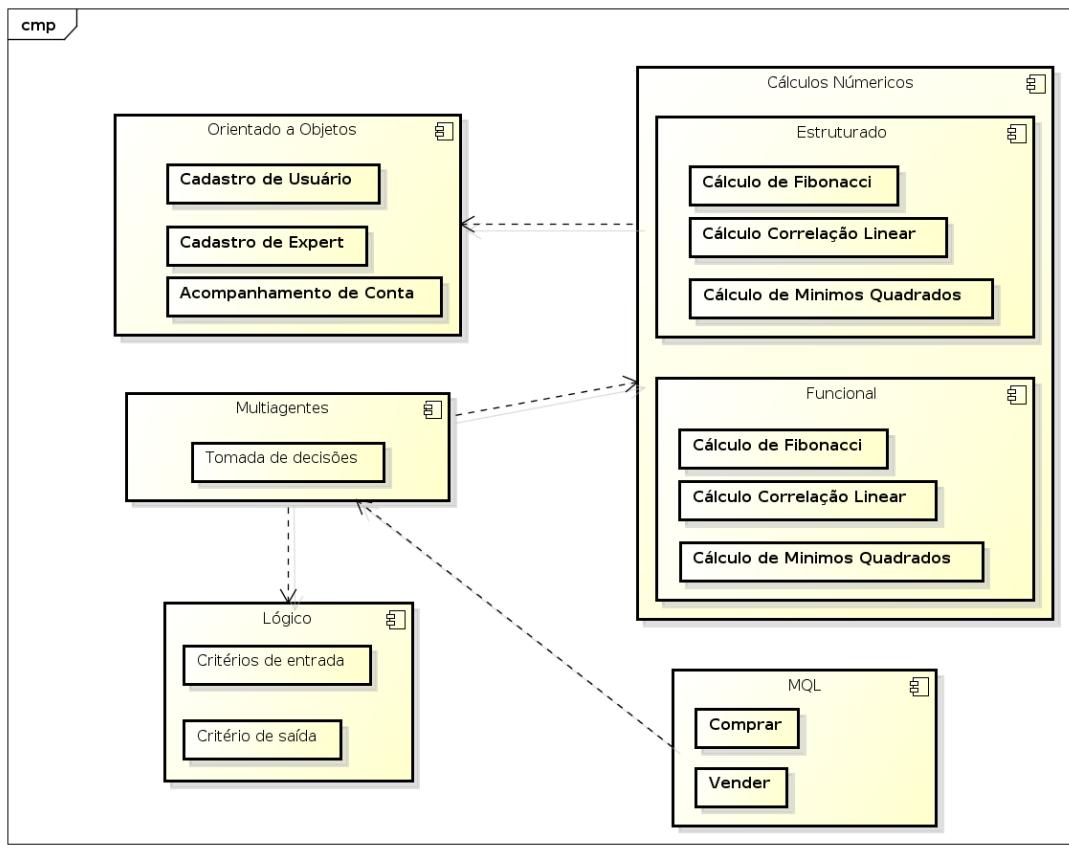


Figura 36 – Diagrama de Classe InvestMVC

4.2.1 Componente Orientado a Objetos

Este componente é o responsável pela interação com o usuário e foi 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 37.

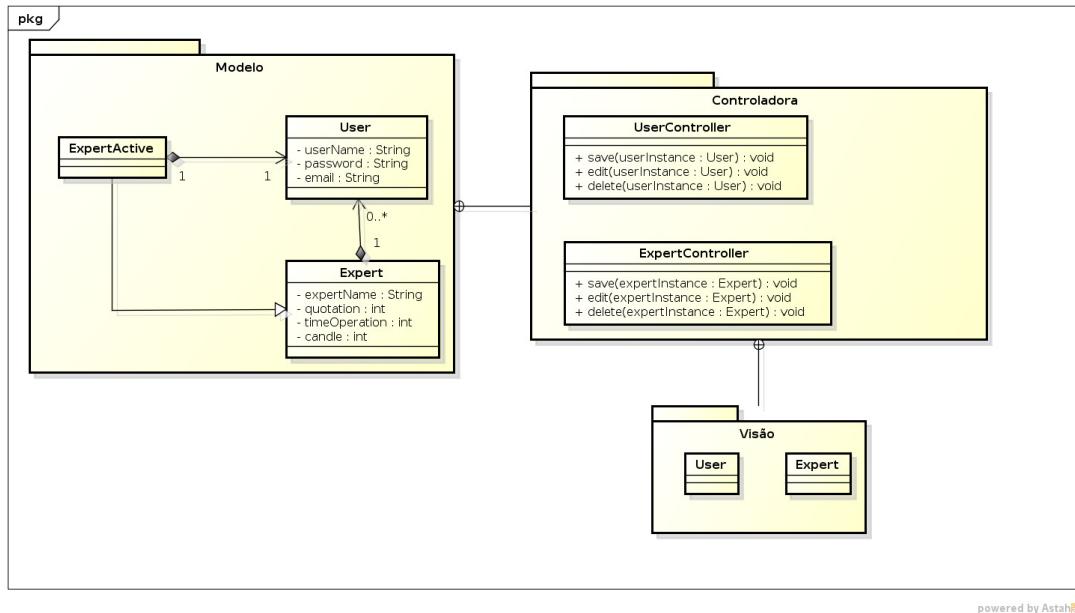


Figura 37 – Diagrama de Classe InvestMVC componente Orientado a Objetos

Segundo Lamim (2010) na arquitetura MVC, o controle de fluxo de dados ocorre de forma específica. Orientando-se por esse autor, o fluxo de dados dentro do componente Orientado a Objetos do software InvestMVC 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 por meio de uma rotina.
3. A Controladora acessa a camada de Modelo, atualizando-a com base nas interações do usuário.

O diagrama de sequência do componente Orientado a Objetos é evidenciado na Figura 38.

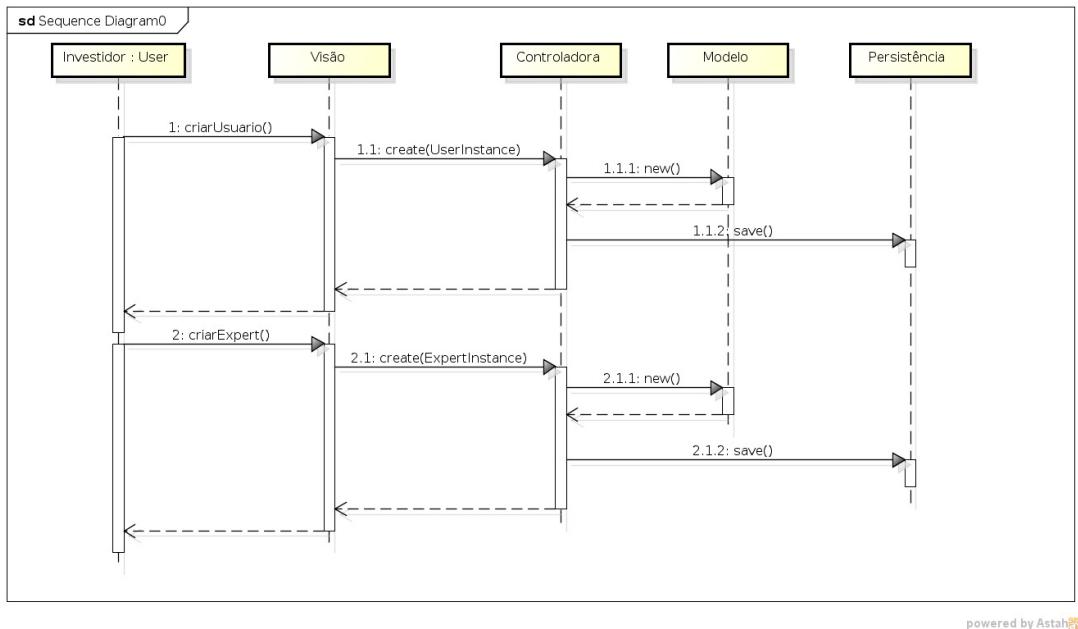


Figura 38 – Diagrama de sequência InvestMVC componente Orientado a Objetos

As classes mais relevantes do componente Orientado a Objetos encontram-se no Apêndice E.

4.2.2 Componente Cálculos Numéricos

O componente Cálculos Numéricos é responsável por calcular os Métodos Matemáticos presentes no software InvestMVC. Este módulo é composto por dois outros componentes: componente Estruturado e componente Funcional.

O componente Estruturado do software InvestMVC foi programado em linguagem C e o módulo Funcional em linguagem Haskell. Ambos os componentes realizam 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.

4.2.3 Componente Funcional

Na linguagem Haskell, 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 ([HOOOGLE, 2013](#)).

O paradigma funcional é declarativo. Por limitar o uso de atribuições às variáveis, evitando a noção de estados (comum no paradigma estruturado bem como na Orientação a Objetos), os resultados das funções em Haskell são mais precisos do que em outros paradigmas ([DOETS; EIJCK, 2004](#)).

Devido estes fatos, o paradigma funcional foi utilizado para implementar os Métodos Matemáticos de Correlação Linear, Mínimos Quadrados e Fibonacci. Por ser simples, este componente será formado apenas por quatro arquivos em Haskell, cada arquivo realiza o cálculo de um método matemático.

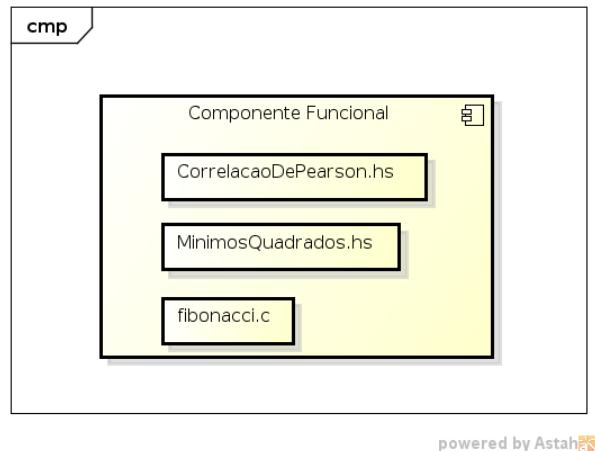


Figura 39 – Componente Funcional InvestMVC

O componente Funcional espera uma solicitação de cálculo do componente Multiagente. Logo após a solicitação, o componente busca na camada de persistência (representado por um sistema de arquivos) as cotações do mercado. Com essas cotações o componente é capaz de realizar o cálculo do método matemático, o qual é esperado pelo componente Multiagente, como é mostrado na Figura 40.

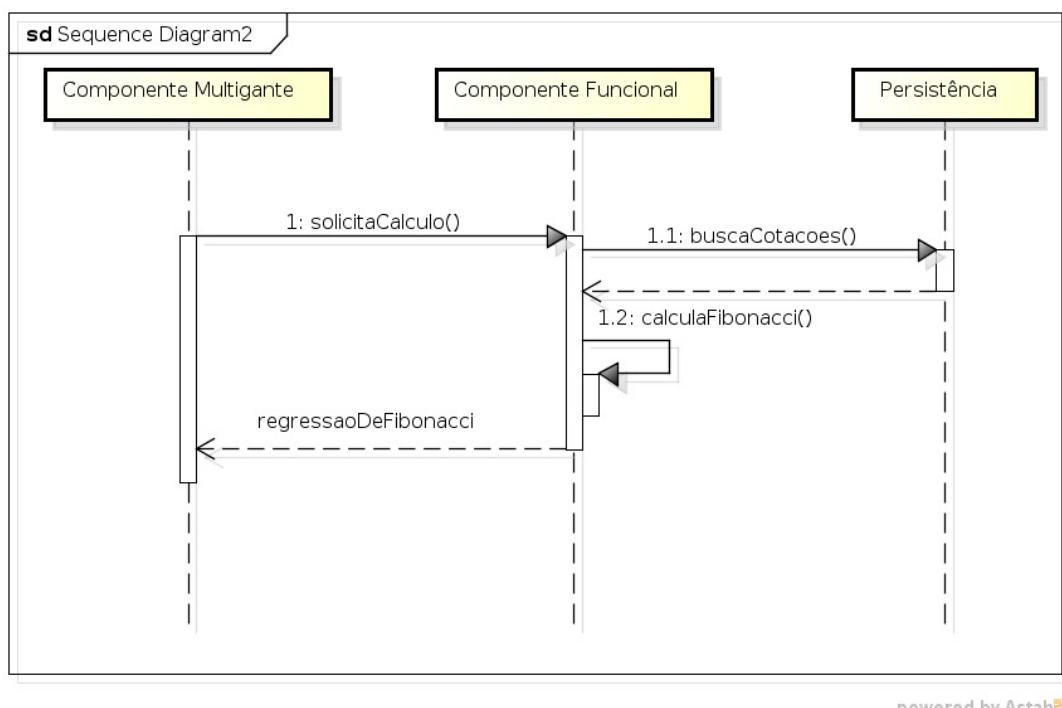


Figura 40 – Diagrama de Sequência do Componente Funcional InvestMVC

4.2.4 Componente Estruturado

A linguagem de programação C é estruturada, possuindo a vantagem da velocidade de execução do código-fonte. Também é uma linguagem bastante utilizada para realizar cálculos numéricos e algébricos (JUNGTHON; GOULART, 2009).

O paradigma estruturado utilizando a linguagem C também foi utilizado para implementar os Métodos Matemáticos de Correlação Linear, Mínimos Quadrados e Fibonacci.

A arquitetura bem como a sequência do fluxo de dados do Componente Estruturado seguem a mesma lógica do Componente Funcional, como ilustrado na Figura 41.

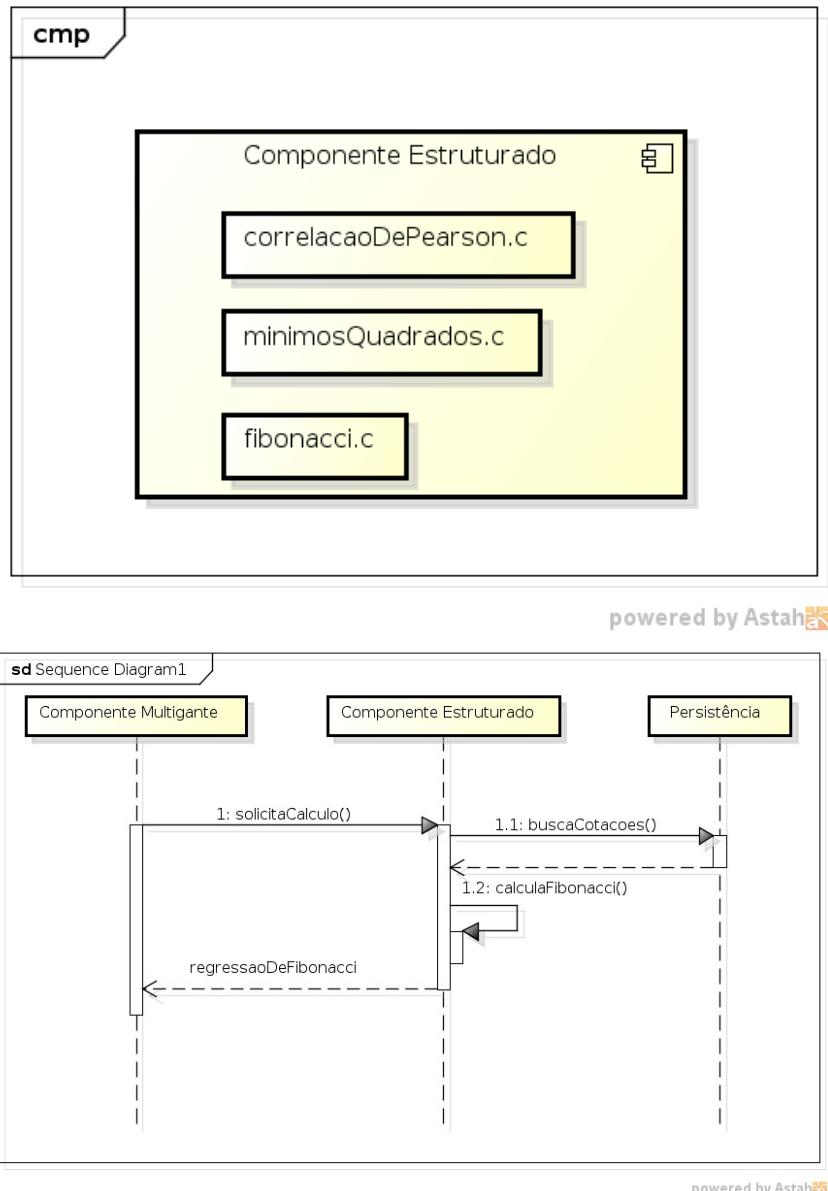


Figura 41 – Diagramas de Componentes e de Sequência do Componente Estruturado InvestMVC.

4.2.5 Compartimento Multiagente

O compartimento Multiagente foi implementado usando o paradigma Multiagente com a linguagem Java e o apoio da Plataforma JADE.

O JADE se trata de um *framework* de software, independente de aplicação específica, capaz de prover funcionalidades de camada *middleware*. Ele possui uma infraestrutura bastante flexível para o desenvolvimento de aplicações onde o elemento de abstração é um agente de software. Para isso ele oferece suporte ao ciclo de vida e a lógica do núcleo de agentes em si, além de oferecer uma variedade de ferramentas gráficas que favorecem o desenvolvimento. A tecnologia é totalmente escrita em Java, o que traz benefícios a partir do enorme conjunto de recursos e bibliotecas oferecidas pela linguagem, que por sua vez oferece um vasto conjunto de abstrações de programação e possibilita a construção de sistemas baseados em agentes com competências relativamente mínimas em relação à teoria de agentes. Com isso, o programador pode economizar o tempo com o trabalho inicial necessário para construir uma infraestrutura baseada em agentes, e dedicar seus esforços para o negócio da aplicação propriamente dito ([TEIXEIRA, 2010](#)).

Agentes de software são entidades autônomas e com capacidades sociais. O uso deste paradigma foi justificado dada as colocações realizadas na etapa de tomada de decisões desse trabalho ([AGENTBUILDER, 2009b](#)).

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

A arquitetura do Compartimento Multiagente está modularizada por pacotes: o pacote comportamentos é formado por comportamentos que são usados pelos Agentes de Software; o pacote metodosMatemáticos é formado por Agentes que acessam o componente Cálculos matemáticos; o pacote investidores é composto por agentes que interagem com o Compartimento MQL, e o pacote execucao inicia a execução do SMA.

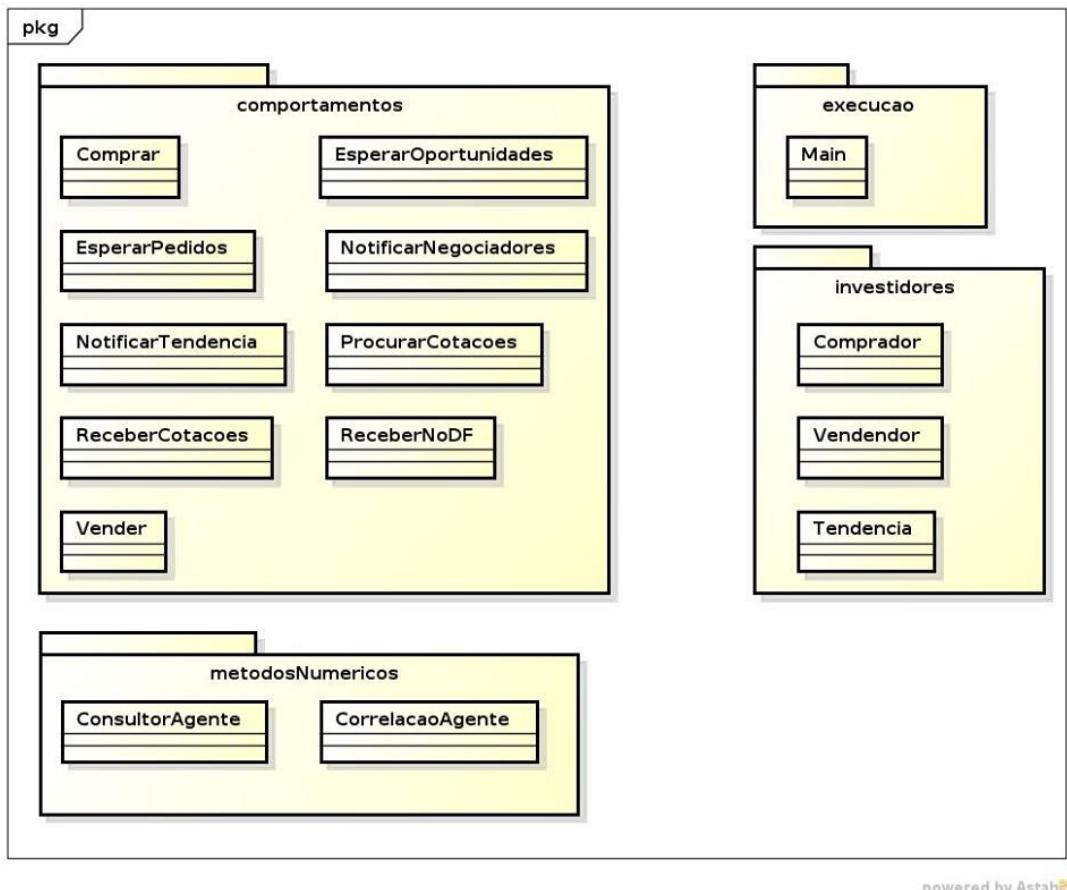


Figura 42 – Diagrama de Classe do Componente Multiagente InvestMVC

4.2.6 Componente Lógico

O componente Lógico foi produzido em linguagem Prolog, definindo uma base de conhecimento, a qual serve 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 43.

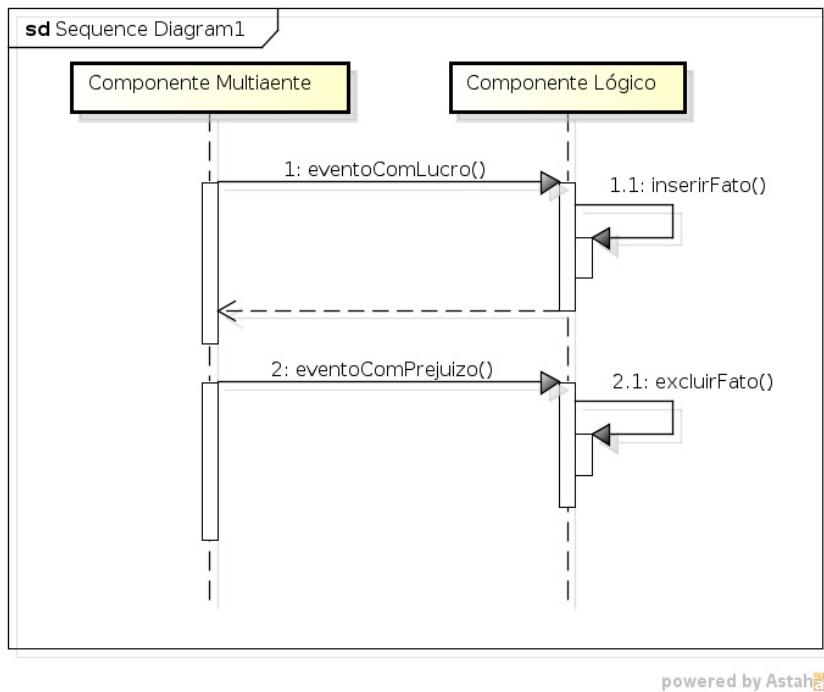


Figura 43 – Diagrama de Sequência do Componente Lógico InvestMVC.

4.2.7 Componente MQL

O componente MQL é responsável por receber a resposta do Componente Multiagente para realizar uma compra ou venda. Adicionalmente, são recebidos outros atributos relacionados à compra ou venda, como alavancagem, *stop loss* e *take profit*, a implementação deste componente encontra-se no Apêndice J.

A interação do Componente MQL com o Componente Multiagente é evidenciada na Figura 44.

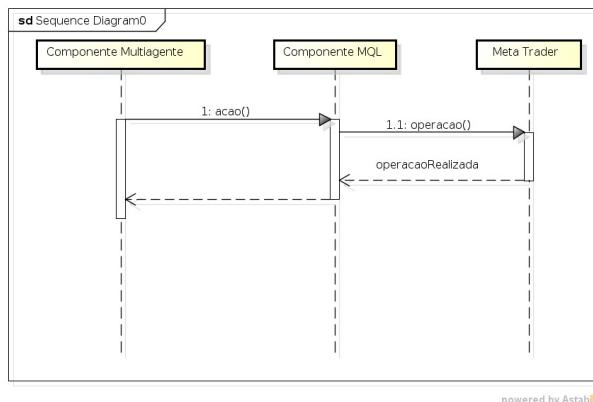


Figura 44 – Diagrama de Sequência do Componente MQL InvestMVC.

4.2.8 Fluxo de atividades

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 componente Multiagente sejam obedecidas, ele informa ao componente MQL para realizar uma compra ou venda. Esta interação é evidenciada na Figura 45.

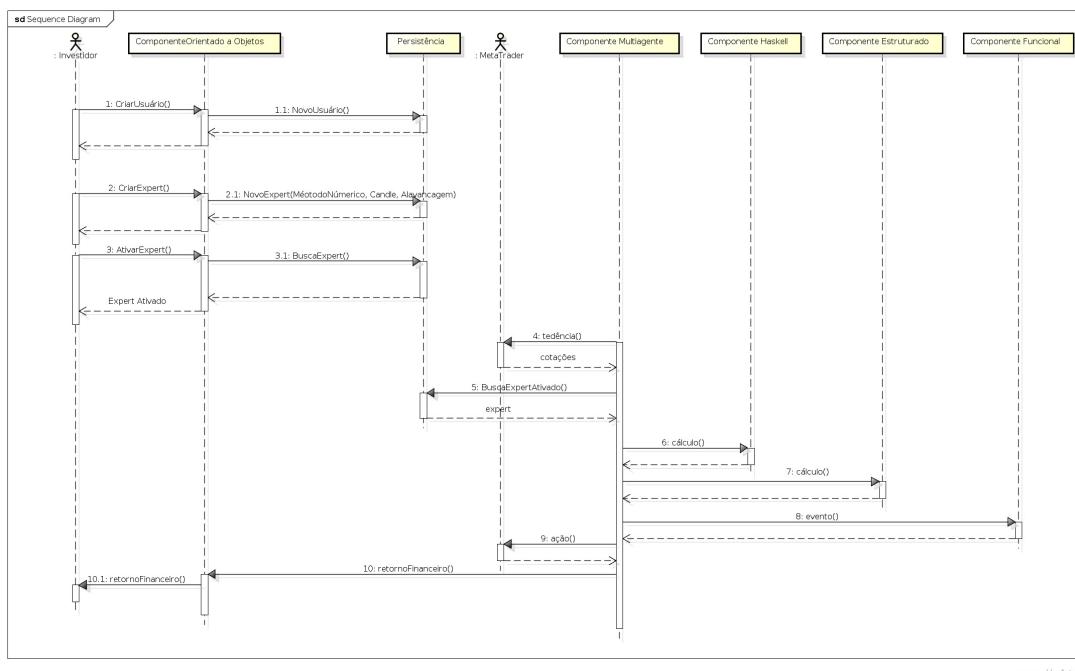


Figura 45 – Diagrama de Sequência InvestMVC

4.3 Testes unitários e cobertura de código-fonte

Esta seção evidencia o resultado dos testes unitários dos componentes Estruturado, Lógico, Funcional e Multiagente. Este resultado atende o objetivo específico 3 (apurar a cobertura de código por meio de ferramentas que implementam testes unitários) deste trabalho.

4.3.1 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 46, 47 e 48.

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

Figura 46 – 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 47 – 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 48 – Resultado da Suíte de Teste do Método Mínimos Quadrados

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

4.3.2 Componente Estruturado

Encontra-se no Apêndice G , o código-fonte das Histórias de Usuário 13 (Método de Correlação Linear em linguagem C), 14 (Método de Fibonacci em linguagem C) e 15 (Método de Mínimos Quadrados em linguagem C). No mesmo apêndice, segue o teste unitário de cada História de Usuário.

Na Figura 49, é possível ver o resultado da suite de teste do Componente Estruturado. A cobertura de código foi de 100%.

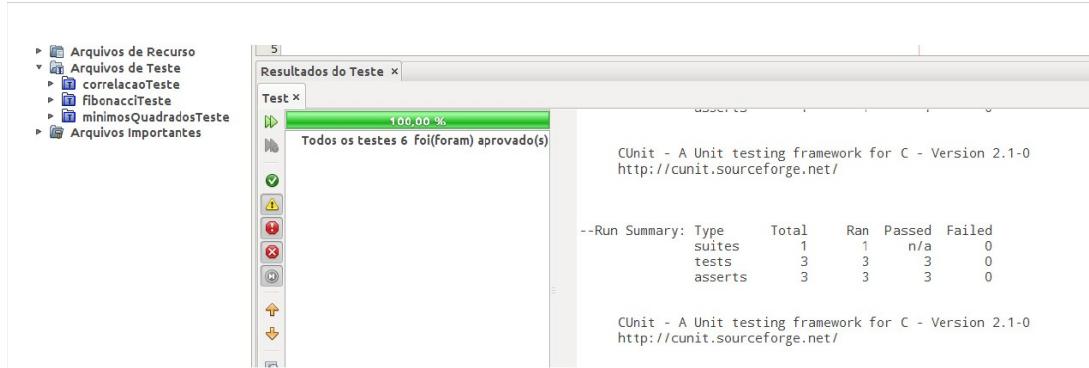


Figura 49 – Cobertura de código-fonte Componente Estruturado

4.3.3 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 50 e 51.

```
?- 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 50 – 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 51 – Suíte de teste da base de conhecimento

Os códigos referentes aos testes unitários em linguagem Prolog, bem como a implementação da base de conhecimento, encontram-se no Apêndice [H](#).

4.3.4 Compartimento Multiagente

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 [52](#).



Figura 52 – Cobertura de Código dos pacotes do Componente Multiagentes

O comportamento dos agentes são executados por meio dos métodos *actions* e muitos comportamentos precisavam de um tempo de até 6 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 o tempo fosse ajustado, viabilizando a comunicação entre os agentes. A execução dos testes demorou mais de 18 (dezoito) segundos conforme pode ser visualizado no canto superior esquerdo da Figura [53](#). Apesar de todos os testes passarem pelo Junit e o Easy-mock com os *delays*, a ferramenta Eclemma não registrava a cobertura de código dos métodos *actions*. Assim, esses métodos foram desconsiderados no percentual de cobertura de código-fonte.

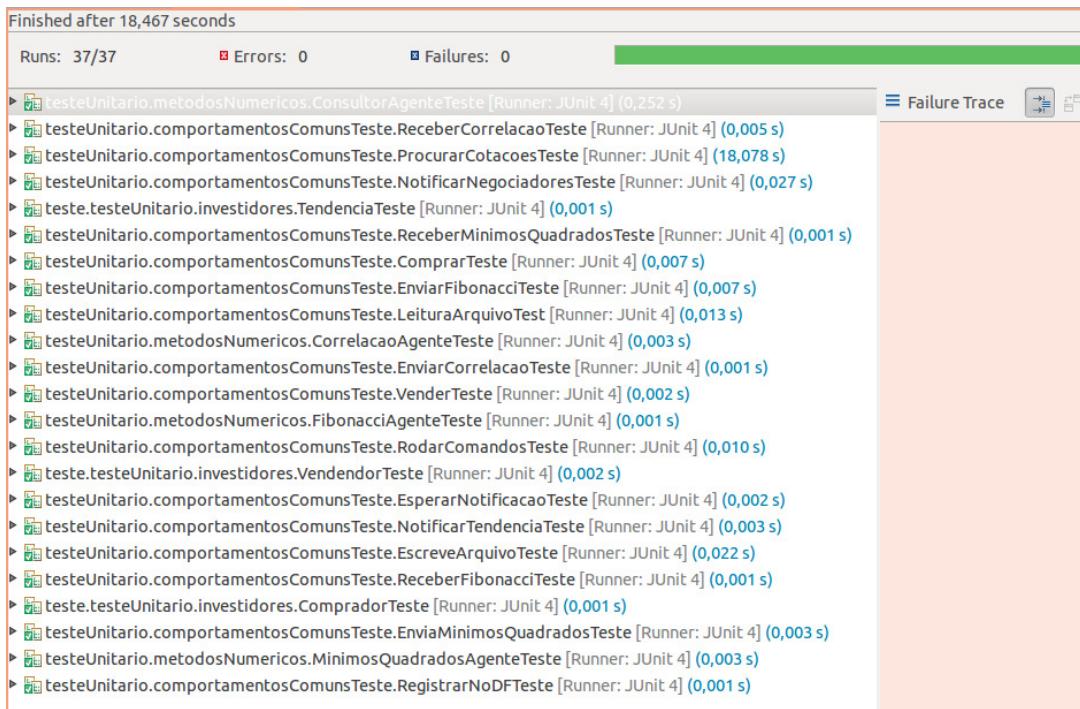


Figura 53 – Suite de teste do Componente Multiagente

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

4.4 Análise Estática de Código-fonte

Esta seção apresenta o resultado da qualidade de código-fonte do componente Multiagente, através da realização de análise estática e interpretação de métricas de qualidade de código-fonte. O resultado atende o objetivo específico 4 (realizar análise estática do código-fonte do componente Multiagente) deste trabalho. Não é objetivo deste trabalho evidenciar as relações entre as métricas como, por exemplo, a alta coesão tender a gerar um baixo acoplamento.

4.4.1 Definição das Métricas de Qualidade de Código-fonte

Para realizar a interpretação das métricas de qualidade de código-fonte, escolheu-se os valores de referência de acordo com a dissertação de (FILHO, 2013). Segundo ele, os valores foram obtidos baseando-se nos resultados da análise de um ou mais "projetos modelo", analisados no Capítulo 4 da tese de Meirelles (2013).

Segundo BRAGA (2012), métricas estruturais de tamanho e profundidade são boas referências para saber se um código está com uma qualidade aceitável. Diante disso, foram escolhidas de forma empírica as métricas estruturais de coesão, acoplamento e complexidade ciclomática. As métricas de tamanho escolhidas foram número de parâmetros

públicos e número de parâmetros por método. Por fim, escolheu-se a métrica de profundidade herança.

4.4.2 Qualidade de código-fonte do software InvestMVC

Foi realizada a análise estática de código-fonte no intuito de obter a qualidade de código-fonte do componente Multiagente. A Figura 54 revela o resultado da primeira análise estática. Percebe-se que a métrica número de parâmetros públicos (NPA) ficou com nível regular no pacote investidores. Com esse resultado, o código foi refatorado e o pacote investidores saiu do nível regular para o nível bom, conforme consta na Figura 55. Além disso, foi atribuída uma menor responsabilidade para o agente tendência no pacote comportamentosComuns e com isso, a complexidade ciclomática (ACCM) nesse pacote evoluiu de um nível bom para excelente. No Apêndice K, é possível visualizar os resultados da análise estática de código-fonte de forma mais detalhada, pois os resultados de cada classe são evidenciados.

	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[
Peculiar	[15, ∞[[7, ∞[[5, ∞[[6, ∞[[3, ∞[[51, ∞[
Pacotes						
comportamentosComuns	0	3,5	2	1	1	0
execucao	2	3	0	1	0	1
investidores	2	4	0	1	2	1
metodosNumericos	0	2	0	1	0	0

Figura 54 – Primeiro resultado da análise estática de código-fonte do componente Multiagente

	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[
Peculiar	[15, ∞[[7, ∞[[5, ∞[[6, ∞[[3, ∞[[51, ∞[
Pacotes						
comportamentosComuns	0	2,5	2	1	1	0
execucao	2	3	0	1	0	1
investidores	2	4	0	1	1	1
metodosNumericos	0	2	0	1	0	0

Figura 55 – Segundo resultado da análise estática de código-fonte do componente Multiagente

4.5 Comparação entre resultados monetários

Esta seção evidencia o resultado dos rendimentos monetários do software InvestMVC e do *expert* implementado em MQL. Os parâmetros (projeto, coleta dos dados, interpretação, validação) para comparar os valores monetários em dólares americanos dos produtos de software, foram definidos na metodologia. O período de coleta dos resultados monetários foi de 18 de Maio de 2015 à 12 de Junho de 2015 (20 dias de negociações,

24 horas por dia, sem contar sábados e domingos). O resultado atende o objetivo específico 5 (comparar resultados financeiros obtidos pelo software InvestMVC com os *experts* tradicionais implementados em linguagem MQL) deste trabalho.

Utilizou-se o gráfico de linhas e de barras para evidenciar os resultados monetários, bem como o desvio padrão amostral e o método de Correlação Linear para evidenciar o resultados do tempo de entrada das operações. O relatório completo das negociações, encontra-se no Apêndice L.

4.5.1 Resultados monetários InvestMVC

O software InvestMVC ficou rodando no Mercado de Moedas durante 20 dias de negociações e fez 19 operações. Dessas 19 operações, 8 (oito) operações tiveram lucro, o que evidencia um índice de acerto de 42.10%. Apesar do índice de acerto ter sido inferior a 50%, o software InvestMVC obteve um lucro de 120.58 USD (cento e vinte dólares e cinquenta e oito centavos de dólares). Sendo assim, obteve-se um lucro de 12,58% durante as operações.

A Figura 56, evidencia o resultado do gráfico das operações em função do rendimento do software InvestMVC.



Figura 56 – Gráfico de rendimento software InvestMVC

4.5.2 Resultados monetários do *expert* MQL

Da mesma forma que o software InvestMVC, o *expert* implementado em linguagem MQL também foi executado no Mercado de Moedas durante 20 dias e fez 19 operações. Desses 19 operações, 8 (oito) negociações tiveram lucro, o que evidencia um índice de acerto de 42.10%.

Foi obtido um lucro de 114.86 USD (cento e quatorze dólares e oitenta e seis centavos de dólares). Portanto, foi gerado um lucro de 11,48% durante as operações.

A Figura 57, evidencia o resultado do gráfico das operações em função do rendimento do *expert* implementado em MQL.



Figura 57 – Gráfico de rendimento do *expert* implementado em MQL

4.5.3 Resultados monetários obtidos em MQL versus InvestMVC

O tempo das operações (20 dias, 24 horas por dia), a quantidade de negociações (19) e índice de acerto nas operações (42.10%) do *expert* implementado em linguagem MQL foram iguais ao software InvestMVC.

O software InvestMVC obteve um lucro 5.72% a mais que *expert* implementado em MQL nos resultados monetários.

A Figura 58 evidencia o gráfico de rendimento do *expert* implementado em MQL (linha vermelha) e do software InvestMVC (linha azul). A linha vermelha e azul ficam juntas na maior parte do gráfico, o que evidencia que os resultados monetários foram semelhantes na maior parte do tempo.



Figura 58 – Resultados monetários do *expert MQL* e *InvestMVC*

É possível que ambos produtos de software, obteveram lucros e perdas, mas o somatório dos lucros foi maior que o somatório das perdas.

A Figura 59 representa o gráfico da comparação dos resultados financeiros do *expert MQL* e do software *InvestMVC*. As barras em vermelho e em azul representam, respectivamente, os rendimentos do *expert MQL* e do software *InvestMVC*, no qual o eixo das abscissas representa as operações feitas pelos *experts* implementados. Da mesma forma que no gráfico de linhas, é possível observar que os resultados monetários de ambos produtos de software ficaram bem próximos.

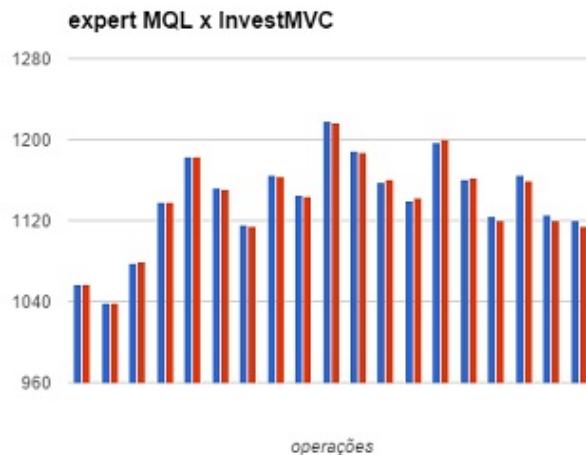


Figura 59 – Resultados monetários do *expert MQL* e *InvestMVC* em gráfico de barras

4.5.4 Outros resultados associados ao rendimento monetário

Verificou-se o horário exato em que foram feitas as operações do expert MQL e do software InvestMVC no Mercado de Moedas. A Tabela 6, evidencia os horários de entrada das operações, ou seja, o horário exato em que foi emitida uma ordem de compra ou venda.

Tabela 6 – Horário de entrada no Mercado de Moedas
expert MQL e software InvestMVC

Horário operação <i>expert MQL</i>	Horário Operação software <i>InvestMVC</i>
2015.05.18 18:01:02	2015.05.18 18:01:17
2015.05.19 06:06:08	2015.05.19 06:06:34
2015.05.19 09:50:07	2015.05.19 09:50:01
2015.05.19 16:33:47	2015.05.19 16:33:56
2015.06.01 05:29:26	2015.06.01 05:29:34
2015.06.01 12:41:19	2015.06.01 12:41:22
2015.06.01 20:58:47	2015.06.01 20:58:55
2015.06.03 10:53:32	2015.06.03 10:53:35
2015.06.04 10:39:09	2015.06.04 10:39:31
2015.06.04 17:00:17	2015.06.04 17:00:33
2015.06.05 02:41:23	2015.06.05 02:41:21
2015.06.05 15:29:45	2015.06.05 15:29:56
2015.06.08 01:30:42	2015.06.08 01:30:43
2015.06.09 10:53:21	2015.06.09 10:53:22
2015.06.09 16:19:47	2015.06.09 16:19:43
2015.06.11 15:32:39	2015.06.11 15:32:44
2015.06.12 07:50:36	2015.06.12 07:50:46
2015.06.12 13:03:29	2015.06.12 13:03:43
2015.06.12 23:24:18	2015.06.12 23:24:22

Foi calculado o desvio padrão amostral dos tempos de entrada do *expert MQL* e do software *InvestMVC* e obteve-se um desvio de 4.23 segundos. Também foi calculado o coeficiente de Correlação Linear dos tempos de entrada e obteve-se um coeficiente de 0.83.

O desvio de 4.23 segundos é considerado relativamente alto, visto que em tese ambos produtos de software deveriam entrar em pontos semelhantes e, portanto, também em tempos de entrada semelhantes. Segundo Lopes (2005)[pág. 134], o resultado de uma

Correlação Linear é considerada como forte se for acima de 0.85. Portanto, o resultado da Correlação Linear é significativo, mas não é considerado como forte.

5 O InvestMVC

Este capítulo evidencia como utilizar o software InvestMVC. O usuário pode logar no sistema, construir um robô de acordo com os parâmetros de entrada desejáveis e ativar o robô para que ele opere de forma automatizada no Mercado de Moedas. Também é possível desativar o robô e consultar o histórico de operações (lucros ou prejuízos). Além disso, caso o usuário seja leigo no assunto de Mercado de Moedas, existe o FAQ com perguntas frequentes e conceitos chaves de mercado.

O código do software InvestMVC é aberto e pode ser clonado ou baixado no [github](#)¹.

5.1 Tela de boas vindas InvestMVC

Ao acessar o software investMVC por meio do endereço do navegador localhost, irá aparecer a tela de boas vindas com animações. No canto superior direito, é possível logar no sistema (botão Entrar em cor verde) ou criar um novo usuário (botão Registrar em cor azul) conforme é mostrado na Figura 60.



Figura 60 – Tela de boas vindas InvestMVC

5.2 Realizando o login

Caso o usuário não tenha um login, basta criar um usuário com nome e senha conforme é evidenciado na Figura 61. Após o usuário criado, basta fazer *login* conforme

¹ <https://github.com/cleitoncsg/investMVC/>

é mostrado na Figura 62.

The screenshot shows a user creation form titled 'Novo User'. It includes fields for 'Email do Usuário*' (input: investMVC@com) and 'Senha*' (input: *****). There are toggle switches for 'Enabled' (On), 'Account Expired' (Off), 'Account Locked' (Off), and 'Password Expired' (Off). At the bottom are 'Criar' and 'Reiniciar' buttons.

Email do Usuário*	investMVC@com
Senha*	*****
Enabled	On
Account Expired	Off
Account Locked	Off
Password Expired	Off
Criar	Reiniciar

Figura 61 – Criando um usuário no software InvestMVC

Por favor faça login

The login screen has fields for 'Usuário' (admin@com) and 'Senha' (*****). A 'Lembrar' toggle switch is set to 'Off'. A green 'Login' button is at the bottom.

Usuário:	admin@com
Senha:	*****
Lembrar	Off
Login	

Figura 62 – Tela de *login* InvestMVC

5.3 Suporte InvestMVC

O software InvestMVC ajuda o usuário que tem pouco conhecimento sobre o Mercado de Moedas. Clicando no canto superior direito no botão “Suporte”, é aberta uma aba com os botões “FAQ” e “Conceitos” conforme é evidenciado na figura 63.

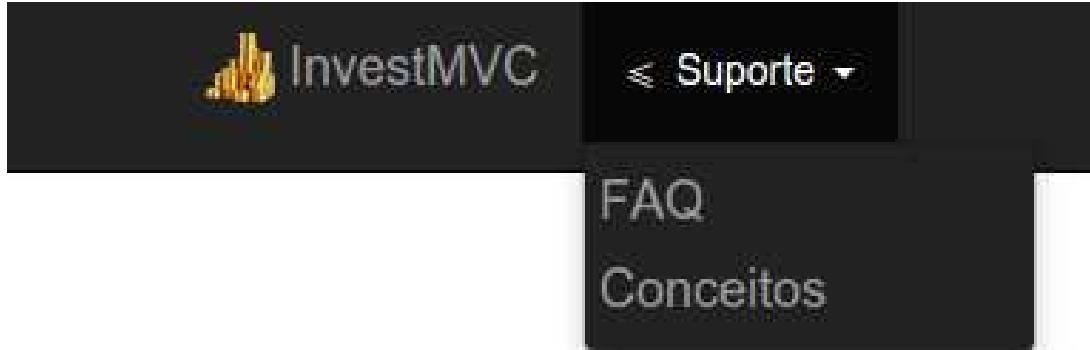


Figura 63 – Suporte InvestMVC

Se o usuário selecionar a opção FAQ, é aberta a tela de “Perguntas Frequentes”, conforme é mostrado na Figura 64. Essa tela ensina o que é o Mercado de Moedas, como funciona, como abrir uma conta para operar, quais são os riscos nas operações, dentre outras informações.

Perguntas Frequentes

Expert Histórico

1. **O que é FOREX, quando e como funciona?** É o mercado de câmbio ou mercado de moedas. O mercado de moedas funciona 24/5 (24 horas por dia, das 18h do domingo às 18h da sexta-feira). Nesse mercado é possível comprar ou vender uma moeda em relação a outra, como euro-dólar, por exemplo. Se o investidor compra euro-dólar e o mercado subir, ele ganha dinheiro. Se vender e o mercado cair, ele também ganha dinheiro. Se o resultado das duas situações anteriores forem ao contrário, ele perde dinheiro.
2. **Qualquer pessoa pode operar em FOREX?** Sim, desde que tenha idade maior que 18 anos, identidade e um comprovante de endereço no próprio nome.
3. **Qual o mínimo de dinheiro para abrir uma conta real em FOREX?** Não tem o mínimo de dinheiro para abrir uma conta real. As corretoras abrem a conta real desde que o investidor mande sua identidade e um comprovante de endereço. Feito isso, o investidor envia qualquer valor acima de 100 dólares e já está liberado para operar no FOREX.
4. **Como eu envio dinheiro para operar em FOREX?** Depende das formas de depósito que a sua corretora aceite. Geralmente as corretoras aceitam depósito bancário, por cartão de crédito, paypal, neteller e boleto bancário.
5. **Encontrei uma corretora de FOREX brasileira, deve abrir uma conta nela?** A Comissão de Valores Mobiliários do Brasil (CVM) não permite que corretoras brasileiras operem em FOREX. Se encontrou uma corretora brasileira que permite operar em FOREX, ela está irregular. Portanto, você pode cair em um golpe. Não abra a conta!
6. **Encontrei uma corretora no exterior para operar em FOREX. Como eu sei que ela é confiável?** Verifique em quais órgãos a corretora está regulamentada. Toda corretora é obrigada a informar na sua página web em quais órgãos está regulamentada. Você pode verificar em qual Comissão de Valores Mobiliários (CVM) e Autoridade de Conduta Financeira (ACF) a corretora está regulamentada. Verifique também na web se existem reclamações sobre essa corretora.

Figura 64 – Tela de “Perguntas Frequentes”

Para o usuário que tem pouca noção de conceitos de mercado, o software InvestMVC oferece a tela de “Conceitos básicos” com mais de 15 (quinze) conceitos, conforme consta na Figura 65.

The screenshot shows a web browser window titled 'Conceitos Básicos' at the URL 'localhost:8080/InvestMVC/conceitos'. The browser's toolbar includes 'Vanessa' as the user, 'Sair', and a flag icon. The page content is titled 'Conceitos Básicos' and contains five numbered definitions:

- Alavancagem**: É o quanto o investidor está disposto a arriscar. O resultado do valor monetário da conta é o produto da alavancagem com os pontos de variação cambial. Se o investidor realizar uma compra no par de moedas euro-dólar com alavancagem 1.0 (um), por exemplo, e o mercado subir 1000 (mil) pontos, o investidor irá ganhar 1000 USD (mil dólares). Entretanto, se a alavancagem for de 0.1 para 1.0, o investidor irá ganhar 100 (cem dólares). No caso de a alavancagem ser de 0.01 para 1.0, o investidor irá ganhar 10 (dez dólares). Se no lugar da compra fosse realizada uma venda, os exemplos anteriores de lucros seriam contabilizados como prejuízos.
- Compra**: Operação que possui como objetivo ganhar dinheiro com a subida do mercado.
- Corretagem**: Taxa que se paga para realizar uma operação de compra ou venda. No mercado FOREX, a corretagem varia conforme a corretora.
- Corretora**: Instituição responsável por emitir as ordens de compra ou venda dos investidores no mercado de moedas. Liquidez
- Liquidez**: Ação em que o investidor fecha suas operações de compra ou venda e se retira do mercado.

Figura 65 – Conceitos básicos de mercado

5.4 Criando um *expert* no software InvestMVC

É possível criar um *expert* no software InvestMVC para operar de forma automatizada no Mercado de Moedas. Para isso, o usuário deve clicar no botão “Novo Expert” e selecionar o tipo de gráfico, o tipo de ordem (venda, por exemplo), os Métodos Matemáticos desejáveis para as operações de compra ou venda, o nome do *expert* e quantidade de *candles* (quantidade de gráfico de velas). Após preencher todos os parâmetros, basta clicar no botão “Criar”. Toda essa dinâmica de criação de um *expert* é exibida na Figura 66.

The screenshot shows the 'Novo Expert' creation form. The fields filled in are:

- Tipo de Gráfico***: M5
- Tipo de Ordem***: Sell
- Métodos Matemáticos***:
 - Correlação
 - Fibonacci
 - Mínimos Quadrados
- Nome**: InvestMVC
- Quantidade de Candles***: 55

At the bottom are two buttons: 'Criar' (Create) and 'Reiniciar' (Reset).

Figura 66 – Criação de um *expert*

Após criar um *expert*, ele irá aparecer na listagem de experts criados conforme

consta na Figura 67.

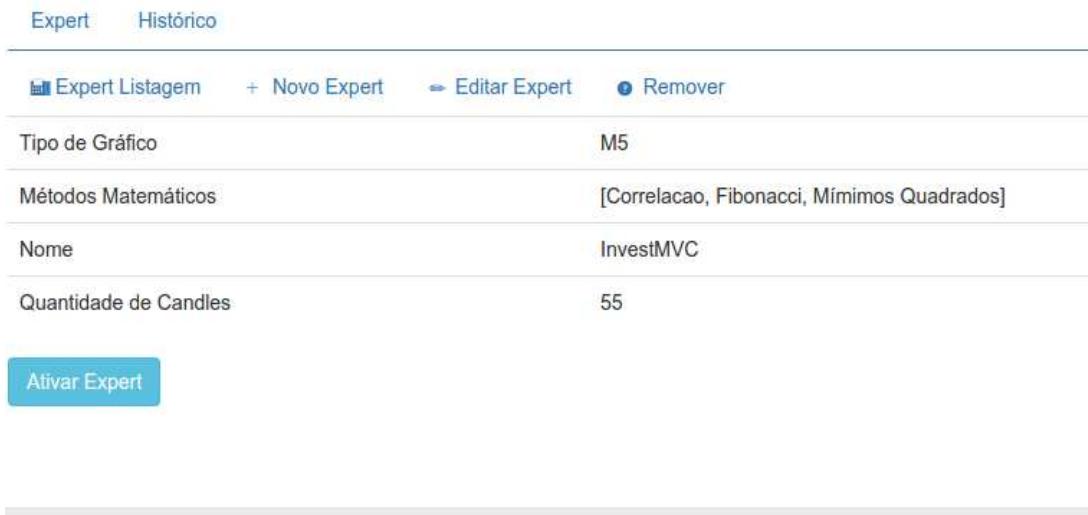


The screenshot shows a software interface with a header 'Expert' and 'Histórico'. Below is a button 'Expert Listagem' and a blue button '+ Novo Expert'. A message 'Expert InvestMVC actived!' is displayed. A table lists one expert:

Nome	Métodos Matemáticos	Tipo de Gráfico
InvestMVC	[Correlacao, Fibonacci, Mínimos Quadrados]	M5

Figura 67 – Listagem de *experts* criados

Ao clicar no *expert* da listagem é aberta uma tela mostrando todos os atributos pertencentes ao *expert* e o botão de “Ativar” (canto inferior esquerdo), conforme consta na Figura 68. Ao ativar o *expert*, ele irá ficar disponível para realizar operações de compra ou venda no Mercado de Moedas no par de moedas euro-dólar.



The screenshot shows a configuration screen for the 'InvestMVC' expert. At the top are tabs 'Expert' and 'Histórico', and buttons for 'Expert Listagem', '+ Novo Expert', 'Editar Expert', and 'Remover'. The expert details are listed below:

- Tipo de Gráfico: M5
- Métodos Matemáticos: [Correlacao, Fibonacci, Mínimos Quadrados]
- Nome: InvestMVC
- Quantidade de Candles: 55

At the bottom is a large blue button labeled 'Ativar Expert'.

Figura 68 – Tela de ativação de um *expert*

5.5 O software InvestMVC em ação

Após criar um *expert* e ativa-lo, é aberta a tela do MetaTrader mostrando os gráficos de operação e o *expert* criado em ação. No retângulo em vermelho, é possível ver o *expert* ativado fazendo os cálculos e esperando oportunidades para comprar ou vender no Mercado de Moedas conforme consta na Figura 69.

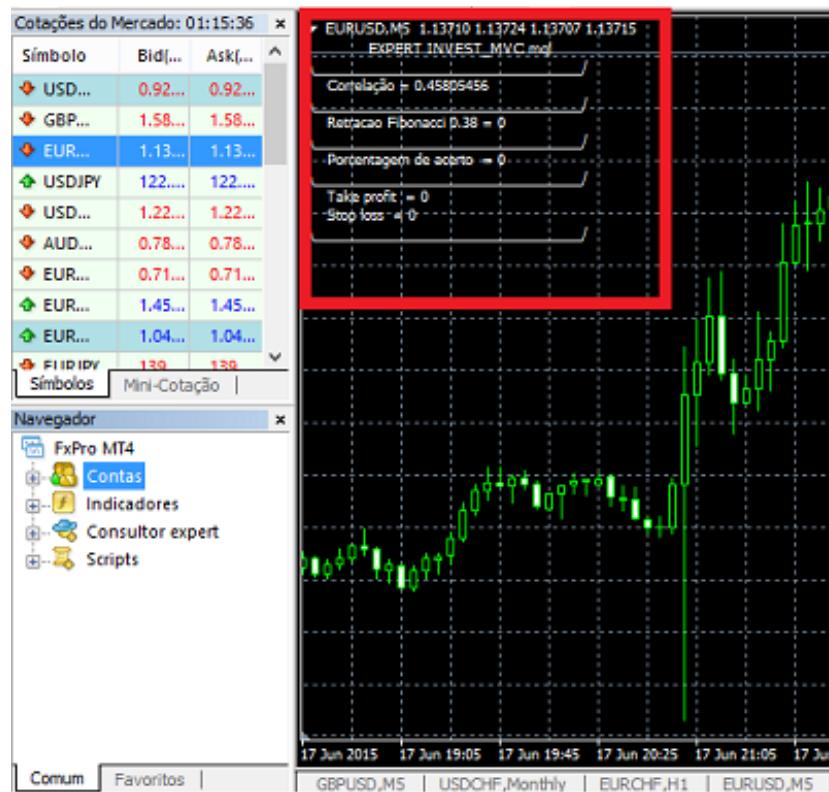


Figura 69 – Software InvestMVC em ação

6 Conclusão

Este trabalho teve o objetivo de desenvolver o software InvestMVC que, por sua vez, é um software multiparadigma que realiza operações financeiras (compra ou venda) no Mercado de Moedas de forma automatizada. Sendo assim, é possível que um usuário que não conheça o Mercado de Moedas, opere através do software e ganhe dinheiro de acordo com a(s) estratégia(s) selecionada(s). Para que o objetivo geral fosse alcançado, foram definidos os objetivos específicos (produtos de trabalho). O primeiro e último objetivo específico (selecionar Métodos Matemáticos e comparar resultados financeiros) tiveram maior peso neste trabalho, pois através dos mesmos foi possível definir passos metodológicos para obter os resultados monetários dos Métodos Matemáticos e responder a questão de pesquisa deste trabalho.

A metodologia foi definida com base nos objetivos e com base nos procedimentos técnicos. Com base nos objetivos foi incorporada uma pesquisa exploratória e descritiva. Em relação aos procedimentos técnicos, foi utilizado um Estudo de Caso, pois apesar deste trabalho ter tido uma abordagem quantitativa dos dados e envolvimento de Métodos Matemáticos para elaboração de estratégias financeiras, não é possível afirmar que os resultados podem ser generalizados para outros contextos, como por exemplo, para outro mercado como a bolsa de valores. O rigor metodológico para o Estudo de Caso foi estabelecido através do planejamento do Estudo de Caso, projeto, coleta dos dados, análise dos dados, validade do plano e limitações de estudo. Outros atributos atrelados a metodologia também foram definidos como atividades da pesquisa, execução da pesquisa e cronograma. A execução da pesquisa não ocorreu conforme o que foi planejado no cronograma e nem todas as atividades da pesquisa foram agregadas ao trabalho. Ajustes foram necessários no decorrer do processo de desenvolvimento do trabalho para que os objetivos específicos fossem alcançados.

Cada Paradigma de Programação presente no software InvestMVC teve seu papel bem definido na arquitetura guiada por componentes. No componente Estruturado foram implementados e realizados testes unitários de todos os Métodos Matemáticos (Correlação Linear, Mínimos Quadrados e Fibonacci). No componente Funcional foi implementado e testado os mesmos Métodos Matemáticos do componente Estruturado. Desta forma, o paradigma Funcional e Estruturado trabalharam juntos nos mesmos cálculos matemáticos e geram um mecanismo de tolerância a falhas. No componente Lógico, foi implementada e testada a inserção dos dados das operações financeiras (tipo de ordem, tipo de método usado, resultado da operação financeira). No componente Multiagente, foram implementados e testados os diversos agentes com suas responsabilidades específicas. Além disso, foi elaborada a comunicação desse paradigma com os demais paradigmas do

software InvestMVC. Por exemplo, o componente Multiagente recebe a resposta dos cálculos numéricos dos componentes Estruturado e Funcional e verificam se é possível emitir uma ordem de compra ou venda para o componente MQL. Por fim, o Componente MQL foi implementado para realizar as operações de compra ou venda de acordo com a ordem do componente Multiagente, porém não foi possível implementar testes unitários para o mesmo, pois a linguagem MQL não possui nenhum mecanismo que permita realizar testes unitários.

Foi necessário selecionar Métodos Matemáticos para serem implementados no software InvestMVC. Então, foi proposto o objetivo específico de “selecionar Métodos Matemáticos para serem implementados no software InvestMVC”. Através da metodologia de pesquisa, foram definidos passos para se alcançar esse objetivo específico. Depois de realizar simulações dos métodos, foi selecionado o método de Correlação de Pearson, Fibonacci e Mínimos Quadrados para serem implementados no software InvestMVC.

Para desenvolver o software InvestMVC, seria necessário analisar as arquiteturas existentes dos paradigmas de programação e estudar como os paradigmas poderiam se comunicar de forma harmônica. Além disso, era necessário explorar o que cada paradigma poderia oferecer de melhor. Então foi elaborado o objetivo específico “caracterizar as estruturas e componentes do software InvestMVC”. A arquitetura desenvolvida neste trabalho foi guiada por componentes e cada componente possui um paradigma de programação com suas responsabilidades específicas.

Para obter a qualidade do software InvestMVC, seria viável propor testes unitários e realização de análise estática de código-fonte. Então, foi proposto os objetivos específicos “realizar análise estática de código-fonte do paradigma multiagente a partir de métricas previamente definidas” e “apurar a cobertura de código por meio de ferramentas que implementam testes unitários nos Paradigmas Estruturado (linguagem C), Multiagentes (linguagem Java), Lógico (linguagem Prolog) e Funcional (linguagem Haskell)”. Foram realizados testes unitários de todos os componentes do software InvestMVC, exceto o componente MQL, pois a linguagem MQL não possui nenhuma ferramenta para desenvolvimento de testes unitários. A cobertura de código-fonte foi de 100% no componente Estruturado e acima de 80% no componente Multiagente, contudo não foi possível obter a cobertura dos componentes Funcional e Lógico, apesar de terem sido desenvolvidos os testes unitários. Em relação a análise estática de código-fonte, foi obtido a qualidade de código-fonte do componente Multiagente e a qualidade ficou em um nível aceitável. A maioria das métricas deram resultado excelente.

Foi mostrado neste trabalho, como pode ser utilizado o software InvestMVC, desde a criação de um usuário, *login*, criação de um *expert*, ativação do mesmo e visualização dos resultados. Também foi mostrado como um usuário que não tenha conhecimento sobre o Mercado de Moedas, pode ser auxiliado através da opção de “Suporte”.

Para responder a questão de pesquisa deste trabalho, seria necessário colocar o software InvestMVC e o *expert* MQL para operar no Mercado de Moedas e obter os resultados monetários. Então, foi desenvolvido o objetivo específico “comparar resultados monetários do software InvestMVC e do *expert* implementado em linguagem MQL”. Foram selecionados os métodos de operação, tempo de operação, tipo de gráfico e demais atributos para os produtos de software operarem conforme estabelecido na metodologia de pesquisa. Os resultados do software InvestMVC e do *expert* MQL foram semelhantes. Os pontos de entrada e saída no mercado não ficaram iguais, porém próximos. Com isso, os valores monetários também ficaram próximos. O software InvestMVC obteve pouco mais de 5% de lucro em relação ao *expert* MQL.

Sugere-se como trabalhos futuros, analisar os tempos de resposta do *expert* implementado em linguagem MQL e do software InvestMVC. Sugere-se também, analisar os pontos de entrada e saída das operações de compra e venda. Por fim, em relação a qualidade do produto, sugere-se realizar outros tipos de teste como: teste funcional e teste de integração entre os componentes.

Neste trabalho foi proposta a seguinte questão de pesquisa: “é possível que o software InvestMVC substitua os *experts* tradicionais do Mercado de Moedas?”. Diante dos resultados deste trabalho, oriundos dos objetivos específicos, há fortes indícios que sim, o software InvestMVC pode ser uma alternativa aos *experts* tradicionais utilizados para operarem no Mercado de Moedas.

A expectativa dos autores é que o software InvestMVC possa ser usado por pessoas que tenham conhecimento do Mercado de Moedas ou até mesmo que não entendam muito sobre este mercado. Sendo assim, espera-se que este trabalho incentive pessoas a buscarem conhecimento sobre este mercado e as aplicações da Engenharia de Software neste contexto.

Referências

- ADVFN. Technical analysis. 2013. Disponível em: <<http://adfn.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 22.
- AGENTBUILDER. Mercado forex: Série alertas. 2009. Disponível em: <<http://www.cvm.gov.br/port/Alertas/mercadoForex.pdf>>. Citado 2 vezes nas páginas 21 e 63.
- 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-site>>. Citado 2 vezes nas páginas 23 e 64.
- 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 19.
- APT, K. R. *From Logic programming to Prolog*. 1. ed. [S.l.], 1996. Citado na página 22.
- BANDEIRA, M. *Tipos de Pesquisa*. Monografia (Artigo) — Faculdade de Psicologia, FUNREI, Minas Gerais, 2012. Citado na página 43.
- BEIZER, B. *Software Testing Techniques*. 2. ed. [S.l.], 1990. Citado na página 24.
- 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 20.
- 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>. Citado 2 vezes nas páginas 27 e 70.
- BRERETON, P. et al. *Using a Protocol Template for Case Study Planning*. [S.l.], 2008. Citado 2 vezes nas páginas 40 e 42.
- 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 2 vezes nas páginas 27 e 28.

CAPRETZ, L. F. *A Brief History of the Object-Oriented Approach*. [S.l.], 2003. Citado na página 18.

CHEESMAN, J.; DANIELS, J. *UML Components*. [S.l.], 2001. Citado na página 57.

CHESS, B.; WEST, J. *Secure Programming with Static Analysis*. [S.l.], 2007. Citado na página 28.

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 22.

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 20.

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/rcf/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 8.

DOETS, K.; EIJCK, J. van. *The Haskell Road to Logic, Maths and Programming*. [S.l.], 2004. Citado 2 vezes nas páginas 23 e 60.

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 2 vezes nas páginas 28 e 70.
- FONSECA j j s. *Metodologia da pesquisa científica*. [S.l.], 2002. Citado na página 31.
- FORCON. Metodologias para monografia. 2014. Disponível em: <<http://www.fazermanografia.com.br/metodologia-para-monografia>>. Citado 2 vezes nas páginas 31 e 32.
- 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. Citado na página 31.
- GIRARDI, R. *Engenharia de Software baseada em Agentes*. [S.l.], 2004. Citado 2 vezes nas páginas 20 e 21.
- GIUDICE, J. C. S. Metodoogia científica e métodos de pesquisa. 2010. Disponível em: <http://www.oficinadapesquisa.com.br/APOSTILAS/METODOL/_OF.TIPOS_PESQUISA.PDF>. Citado na página 31.
- HOOGLE. Functional programming. 2013. Disponível em: <http://www.haskell.org/haskellwiki/Functional_programming>. Citado 2 vezes nas páginas 22 e 60.
- IEEE 610. *IEEE Standard Glossary of Software Engineering Terminology*. [S.l.], 1990. Citado 4 vezes nas páginas 24, 25, 26 e 27.
- 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.
- INVESTFOREX. Análise técnica para mercado forex. 2014. Disponível em: <<http://www.investforex.pt/aprender-forex/analise-tecnica>>. Citado 3 vezes nas páginas 13, 14 e 15.
- JENNINGS, N. R.; SYCARA, K.; WOOLDRIDGE, M. *A Roadmap of Agent Research and Development*. [S.l.], 1998. Citado na página 21.
- JENNINGS, N. R.; WOOLDRIDGE, M. *Intelligent agents: theory and practice*. [S.l.], 1995. Citado na página 21.
- JUNGTHON, G.; GOULART, C. M. *Paradigmas de Programação*. Monografia (Monografia) — Faculdade de Informática de Taquara, Rio Grande do Sul, 2009. Citado na página 62.
- 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>. Citado na página 59.
- 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 19.
- LEWIS, W. E. *Software Testing and Continuous Quality Improvement*. 3. ed. [S.l.], 2009. Citado na página 26.
- 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 3 vezes nas páginas 10, 11 e 75.
- 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 3 vezes nas páginas 27, 28 e 70.
- 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 28.
- 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 29.
- MYERS, G. J. *The art of Software Testing*. 2. ed. [S.l.], 2004. Citado 3 vezes nas páginas 23, 25 e 27.
- NAIK, K.; TRIPATHY, P. *SOFTWARE TESTING AND QUALITY ASSURANCE Theory and Practice*. 2. ed. [S.l.], 2008. Citado 2 vezes nas páginas 25 e 26.
- 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 24.
- 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>. Citado na página 15.
- ODELL, J.; GIORGINI, P.; MÜLLER, J. P. *Agent-Oriented Software Engineering V*. [S.l.], 2005. Citado na página 21.

- PAQUET, J.; MOKHOV, S. *Comparative Studies of Programming Languages*. [S.l.], 2010. Citado 3 vezes nas páginas 15, 16 e 23.
- 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 17.
- ROBOFOREX. Indicators used in forex market. 2013. Disponível em: <<http://www.roboforex.pt/beginner/start/technical-analysis>>. Citado na página 14.
- 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 na página 12.
- RODRIGUES, W. C. *Metodologia Científica*. [S.l.], 2007. Citado na página 31.
- RUSSEL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3. ed. [S.l.], 1995. Citado na página 20.
- SCHUMACHER, M. *Objective Coordination in Multi-Agent System Engineering: design and implementation*. 1. ed. [S.l.], 2001. Citado na página 20.
- 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>. Citado na página 38.
- SEBESTA, R. W. *Concepts of programming languages*. 10. ed. [S.l.], 2012. Citado 3 vezes nas páginas 16, 17 e 22.
- SERAPIONI m. *Métodos qualitativos e quantitativos na pesquisa social*. [S.l.], 2005. Citado na página 31.
- 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 19.
- SINGH, C. Method overloading in java with examples. 2014. Disponível em: <<http://beginnersbook.com/2013/05/method-overloading/>>. Citado na página 19.
- SOMMERVILLE, I. *Engenharia de Software*. 9. ed. [S.l.], 2011. Citado na página 28.
- SPILLNER, A.; LINZ tilo; SCHAEFER, H. *Software Testing Foundations*. 4. ed. [S.l.], 2014. Citado na página 27.
- SPIVEY, M. *An introduction to logic programming through Prolog*. 1. ed. [S.l.], 1996. Citado na página 23.
- STEFANOV, S. *Object-Oriented JavaScript*. [S.l.], 2008. Citado na página 17.

TEIXEIRA, F. V. Jade: Java agent development framework. 2010. Disponível em: <http://www.dca.unicamp.br/~gudwin/courses/IA009/artigos/IA009_2010_12.pdf>. Citado na página 63.

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 26.

THOMPSON, S. *Haskell: The Craft of Functional Programming*. 2. ed. [S.l.], 1999. Citado na página 22.

TUCKER, A. B.; NOONAN, R. E. *Linguagens de programação : Princípios e paradigmas*. 2. ed. [S.l.], 2009. Citado 4 vezes nas páginas 17, 18, 22 e 23.

VENNERS, B. Polymorphism and interfaces. 1996. Disponível em: <<<http://www.artima.com/objectsandjava/webuscript/PolymorphismInterfaces1.html>>>. Citado na página 19.

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.

WEBER, T. S. *Tolerância a falhas: conceitos e exemplos*. Monografia (Especialização) — Universidade Federal do Rio Grande do Norte, Rio Grande do Norte, 2009. Citado na página 43.

WEGNER, P. *Concepts and paradigms of object-oriented programming*. 1. ed. [S.l.], 1990. Citado na página 18.

WEISFELD, M. A. *The Object-Oriented Thought Process*. 3. ed. [S.l.], 2009. Citado na página 18.

WILLIAMS, L. *White-Box Testing*. [S.l.], 2006. Citado 2 vezes nas páginas 25 e 26.

WOLFRAM. Built with mathematica technology: Movingaverage. 2012. Disponível em: <<http://mathworld.wolfram.com/MovingAverage.html>>. Citado na página 14.

WOOLDRIDGE, M. *Teamwork in Multi-agent systems: A Formal Approach*. 1. ed. [S.l.], 2010. Citado 2 vezes nas páginas 20 e 21.

YIN, R. K. *Estudo de caso: planejamento e método*. 4. ed. [S.l.], 2009. Citado 3 vezes nas páginas 31, 42 e 43.

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.

Expert - Software que realiza operações de compra ou venda no Mercado de Moedas.

FGA - Faculdade do Gama

Framework - Captura a funcionalidade comum a várias aplicações

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.

Middleware - Software que atua como um mediador, entre dois programas existentes e independentes.

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 deseja perder.

Take profit - Define a quantidade de pontos que o investidor deseja ganhar.

Tendência - Revela a direção em que o mercado está indo.

USD - Dólares americanos

UnB - Universidade de Brasília

Apêndices

APÊNDICE A – 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 B – Cronograma InvestMVC

Project:

TCC Invest MVC²

ScrumMe (beta)

Sprint/Story/Task	Team	Finalized	TD	IP	VR	DN
Sprint: Sprint 10						
Implementar US21						
Elaborar comunicação metaTrader com multiagente	Cleiton					■
Invocar métodos matemáticos	Cleiton					■
Lançar ordem de compra ou venda	Cleiton	6/17/2015				■
Implementar US22						
Emitir compra	Cleiton					■
Emitir Venda	Vanessa					■
Implementar US20						
Excluir correlações dispensáveis	Cleiton					■
Calcular correlações ideais	Cleiton					■
Escrever resultados						
U20	Cleiton	6/17/2015				■
US22	Vanessa	6/17/2015				■
US21	Vanessa	6/17/2015				■
Sprint: Sprint 11						
Atualizar trabalho escrito						
Objetivo Específico 5	Cleiton					■
Elaborar conclusões	Cleiton	6/17/2015				■
Atualizar objetivo específico 2	Vanessa					■
Sprint: Sprint 7						
Implementar US11						
Criar arquivo de execução	Vanessa	6/17/2015				■
Criar Controladora	Cleiton	6/17/2015				■
Criar botão na view	Vanessa	6/17/2015				■
Ajustar metodologia						
Verificar Estudo de Caso	Cleiton	6/17/2015				■
Verificar Pesquisa Ação	Cleiton	6/17/2015				■
Implementar US 12						
Criar botão na view	Cleiton	6/17/2015				■
Apagar persistência da ativação	Cleiton	6/17/2015				■

Criar Controladora	Cleiton	6/17/2015	■
Implementar US5			
Criar critério de gestão	Cleiton	6/17/2015	■
Escrever na persistência	Vanessa	6/17/2015	■
Escrever resultados			
US11	Cleiton	6/17/2015	■
US05	Vanessa	6/17/2015	■
US12	Vanessa	6/17/2015	■
Sprint: Sprint 8			
Implementar US17			
Implementar funções recursivas	Vanessa		■
Implementar resistência e suporte	Vanessa		■
Escrever resultados			
US02	Vanessa	6/17/2015	■
US04	Cleiton	6/17/2015	■
US17	Cleiton	6/17/2015	■
Implementar US04			
Comunicar com agente de compra e venda	Cleiton		■
Verificar melhor método de tendência	Cleiton	6/17/2015	■
Implementar US02			
Criar registros no DF	Vanessa	6/17/2015	■
Criar comunicação componente estruturado	Cleiton	6/17/2015	■
Criar comunicação componente funcional	Vanessa	6/17/2015	■
Sprint: Sprint 9			
Implementar US18			
Implementar coeficiente linear	Cleiton		■
Implementar coeficiente angular	Cleiton	6/17/2015	■
Elaborar regressão de fibo	Cleiton		■
US07			
Construir histórico na view	Vanessa		■
Desenvolver integração com multiagente	Vanessa		■
Construir faq de apoio	Vanessa	6/17/2015	■
Implementar US19			
Inserir Correlação Linear	Cleiton		■

Retirar Correlação Linear	Cleiton	■
Colocar resultados na persistência	Vanessa	■
Escrever resultados		
US18	Cleiton	6/17/2015
US19	Vanessa	6/17/2015
US07	Vanessa	6/17/2015

Sprint: Sprint1

Construir Introdução

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

Implementar Métodos Numéricos

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

Sprint: Sprint2

Construir Referencial Teórico Paradigmas de Programação

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

Adaptar Métodos Numéricos

Realizar Comunicação com Grails	Vanessa	10/28/2014	■
Realizar Comunicação com MetaTrader	Cleiton	10/28/2014	■
Criar CSV dinâmico	Cleiton	10/28/2014	■

Protótipo View Projeto

Estudar Kickstart	Vanessa	9/11/2014	■
Implementar View	Vanessa	10/28/2014	■

Construir Referencial Teórico Métodos Numéricos

Definir Fibonacci	Cleiton	10/28/2014	■
Definir Mínimos Quadrados	Cleiton	10/28/2014	■
Definir Correlação Linear	Cleiton	10/28/2014	■

Sprint: Sprint3

Refinar Referencial Teórico Paradigmas

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	10/31/2014	■

Construir Referencial Teórico de Contexto Financeiro

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

Revisar Referencial Teórico Métodos Numéricos

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

Sprint: Sprint4**Realizar Experimentos Métodos de Operação**

Realizar Experimento Método de Correlação Linear	Cleiton	6/17/2015	■
Realizar Experimento Método de Mínimos Quadrados	Cleiton	6/17/2015	■
Realizar Experimento Método de Estocástico	Cleiton	6/17/2015	■
Realizar Experimento Método de Fibonacci	Cleiton	6/17/2015	■
Realizar Experimento Método de Média Móvel	Cleiton	6/17/2015	■

Evidenciar Aplicações Paradigmas de Programação

Evidenciar aplicações paradigma MultiAgente	Vanessa	6/17/2015	■
Evidenciar aplicações paradigma Lógico	Vanessa	6/17/2015	■
Evidenciar aplicações paradigma Estruturado	Vanessa	6/17/2015	■
Evidenciar aplicações paradigma Funcional	Vanessa	6/17/2015	■

Revisar Referencial Teórico Contexto Financeiro

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

Desenvolver Experts em MQL4

Desenvolver Expert MinimosQuadrados	Cleiton	10/28/2014	■
Desenvolver Expert MediaMovel	Vanessa	10/28/2014	■
Desenvolver Expert Fibonacci	Cleiton	10/28/2014	■

Desenvolver Expert CorrelacaoLinear	Cleiton	10/28/2014	■
Desenvolver Expert Estocastico	Vanessa	10/28/2014	■

Sprint: Sprint5**Descrever Metologia de Pesquisa**

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

Realizar Experimentos Métodos de Operação

Realizar experimento Método de Correlação Linear	Cleiton	10/28/2014	■
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 Fibonacci	Cleiton	10/28/2014	■

Evidenciar Aplicações Paradigmas de Programação

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

Sprint: Sprint6**Verificar Qualidade de Código**

Realizar Teste Unitário em Haskell	Vanessa	11/9/2014	■
Realizar Análise Estática	Cleiton	11/9/2014	■

Definir Backlog de Histórias

Componente Orientado a Objetos	Vanessa	11/9/2014	■
Componente Estruturado	Cleiton	11/9/2014	■
Componente Lógico	Cleiton	11/9/2014	■
Componente Multiagente	Cleiton	11/9/2014	■
Componente Funcional	Vanessa	11/9/2014	■

APÊNDICE C – Suporte Tecnológico

1. Ferramentas para teste unitário e teste de integração

XUnit¹ 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

Cunit² 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 fornecidos 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

JUnit³ 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

HUnit⁴ 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.

¹ <https://xunit.codeplex.com>

² <http://cunit.sourceforge.net/>

³ <http://junit.org/index.html>

⁴ <http://hunit.sourceforge.net/>

PIUnit

PIUnit⁵ é um framework para criação de testes automatizados em linguagem Prolog. Os testes são executados via terminal usando o suporte swi⁶. Ao executar os testes, os erros e falhas são mostrados via terminal.

EasyMock

EasyMock⁷ é um framework para criação de teste unitário utilizando mocks ou dublês. Esse tipo de ferramenta é ideal para testar métodos sem retorno, por exemplo.

2. Ferramenta para teste funcional: Cucumber

Cucumber⁸ 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

EclEmma⁹ 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

HPC¹⁰ 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.

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

⁶ http://www.swi-prolog.org/pldoc/doc_for?object=manual

⁷ <http://easymock.org/>

⁸ <http://cukes.info/>

⁹ <http://www.eclemma.org/>

¹⁰ https://www.haskell.org/haskellwiki/Haskell_program_coverage#Hpc_tools

Analizo

Analizo¹¹ 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.

Sonar

Sonar¹² 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.

MetaTrader

MetaTrader¹³ é uma plataforma de negociação eletrônica com capacidade de negociações automatizadas. É possível programar experts em linguagem mql4¹⁴ (paradigma estruturado) e linguagem mql5¹⁵ (paradigma orientado a objetos).

MetaEditor

MetaEditor¹⁶ 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-UK¹⁷ 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.

¹¹ <http://www.analizo.org/>

¹² <http://www.sonarqube.org/>

¹³ <http://www.metaquotes.net/>

¹⁴ <http://www.mql4.com/>

¹⁵ <http://www.mql5.com/>

¹⁶ <http://book.mql4.com/metaeditor/index>

¹⁷ <http://www.alpari.co.uk/>

FXDD

FXDD¹⁸ 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.

6. Editores de texto

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

Eclipse

Eclipse¹⁹ 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

Sublime²⁰ é um editor de texto que suporta diversas linguagens como C, C++, Java, Groovy, entre outras.

7. Ferramenta organizacional

Esta seção descreve a ferramenta selecionada para organizar o processo de desenvolvimento do trabalho.

ScrumMe

ScrumMe²¹ é uma ferramenta online 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.

8. Ferramentas de Apoio

Esta seção descreve as ferramentas que dão suporte ao funcionamento do InvestMVC.

JADE

JADE²² é um software framework, que simplifica a implementação de sistemas multiagente através de um middleware que está em conformidade com as especificações do FIPA e através de um conjunto de ferramentas gráficas que suportam as fases de depuração e implantação.

Wine

O Wine²³ é uma camada de tradução (um lançador de programas) capaz de executar aplicações Windows em Linux e outros sistemas operativos compatíveis. Os programas

¹⁸ <http://www.fxdd.com/>

¹⁹ <https://www.eclipse.org>

²⁰ <http://www.sublimetext.com/>

²¹ <http://www.scrumme.com.br>

²² <http://jade.tilab.com/>

²³ <https://www.winehq.org/>

Windows que rodam no Wine agem como se fossem nativos, executando sem as penalidades de desempenho ou uso de memória de um emulador, com um visual semelhante às outras aplicações do seu computador.

APÊNDICE D – Experts para Estudo de Caso

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 prejuizo = ", ponto_negativo ,"\n",
61.        "STOP LOSS = ", STOPLOSS ,"\n",
62.        "TAKE PROFIT = ", TAKEPROFIT ,"\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) +
87.             (numero_abcissa)*(numero_abcissa);
88.             soma_ordenadas = soma_ordenadas + numero_ordenada;
89.             soma_ordenadas_quadrado = (soma_ordenadas_quadrado) +
90.             (numero_ordenada)*(numero_ordenada);
91.             soma_X_vezes_Y = soma_X_vezes_Y + (numero_ordenada*numero_abcissa);
92.     }
93.     numerador
=((tempoCorrelacao*soma_X_vezes_Y)-((soma_abcissas)*(soma_ordenadas)));

```

```

93.                                         denominador_1
94.                                         =((tempoCorrelacao*soma_abcissas_quadrado)-(soma_abcissas*soma_abcissas))* 
95.                                         ((tempoCorrelacao*soma_ordenadas_quadrado)-(soma_ordenadas*soma_ordenadas));
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
22.        =iStochastic(NULL,TEMPO_OPERACAO,AJUSTE_TEMPORAL_MAIOR,AJUSTE_TEMPORAL_MENOR,AJUSTE_TEMP
23.          ORAL_MENOR,MODE_SMA,0,MODE_MAIN,1);
24.    double sinal
25.        =iStochastic(NULL,TEMPO_OPERACAO,AJUSTE_TEMPORAL_MAIOR,AJUSTE_TEMPORAL_MENOR,AJUSTE_TEMP
26.          ORAL_MENOR,MODE_SMA,0,MODE_SIGNAL,1);
27.
28.
29.
30.    if( DayOfWeek() != SEXTA_FEIRA ){
31.
32.        if (estocastico > sinal){
33.            compra = true;
34.
35.        }
36.
37.        if (estocastico < sinal){
38.            venda = true;
39.
40.        }
41.
42.        if( ((compra == true && OrdersTotal() == 0 )) ){
43.            RefreshRates();
44.            while (IsTradeContextBusy()) Sleep(5);
45.            ticket= OrderSend(Symbol(),OP_BUY,ALAVANCAGEM,Ask,0,Ask - TAKE_PROFIT*Point,
46.              Ask + TAKE_PROFIT*Point,nome,AccountNumber(),0,Yellow);
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
21.         mediaMovelRapidaCorrente=iMA(NULL,TEMPO_OPERACAO,mediaMovelRapida,0,MODE_EMA,PRICE_CLOSE
22. ,0);
23.     double mediaMovelLentaCorrente
24.         =iMA(NULL,TEMPO_OPERACAO,mediaMovelLenta,0,MODE_EMA,PRICE_CLOSE,0);
25.
26.     if( DayOfWeek() != SEXTA_FEIRA ){
27.         if (mediaMovelLentaCorrente < mediaMovelRapidaCorrente){
28.             compra = true;
29.         }
30.         if (mediaMovelLentaCorrente > mediaMovelRapidaCorrente){
31.             venda = true;
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 = ", STOPLOSS ,"\n",
67.     "TAKE PROFIT = ", TAKEPROFIT ,"\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_conficienteAngular_cotacao;
20.
21.     produto_conficienteAngular_cotacao = calculoCoeficienteLinear(QUANTIDADE_CANDLES);
22.
23.     if( DayOfWeek() != SEXTA_FEIRA ){
24.         if(coeficienteAngular < 0){
25.             coeficienteAngular = coeficienteAngular*(-1);
26.         }
27.         if(produto_conficienteAngular_cotacao > 1){
28.             compra = true;
29.         }
29.     }
30. }
```

```

30.     if(produto_conficienteAngular_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 -
96.         (variacaoLinear*soma_x/quantidadeVelas);

```

```
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 ?
37.             springSecurityService.encodePassword(password) : password
38.     }

```


APÊNDICE F – Componente Funcional

CorrelacaoDePearson.hs

```
1 module CorrelacaoDePearson
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.QuickCheck
4 import CorrelacaoDePearson
5
6 testaMediaListaVazia :: Test
7 testaMediaListaVazia = TestCase (assertEqual "Média de lista vazia" 0
(calculaMedia []))
8
9 testaMediaLista :: Test
10 testaMediaLista = TestCase (assertEqual "Média 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 quadrática de uma
lista vazia" 0 (somaQuadradoVetor []))
26
27 testaSomaQuadradoVetor :: Test
28 testaSomaQuadradoVetor = TestCase (assertEqual "Soma quadrática 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 "ResistÃ¢ncia de uma lista Comum" 4
(resistencia [0,1,2,3,4]))
11
12 testaRetracao :: Test
13 testaRetracao = TestCase (assertEqual "RetraÃ§Ã£o 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

correlacaoDePearson.c

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4. #include <math.h>
5. #define QUANTIDADE_CANDLES 100
6. #define TAMANHO_STRING 50
7.
8. double leituraCotacoes[QUANTIDADE_CANDLES];
9. char nomeRobo[50], nomeTipoGrafico[2];
10. double metodoCorrelacao(int tempoCorrelacao);
11. void detectaRoboETipoDeGrafico();
12.
13. int main(){
14.     metodoCorrelacao(21);
15.     printf("%f\n", metodoCorrelacao(21));
16.     return 0;
17. }
18.
19. double metodoCorrelacao(int tempoCorrelacao){
20.     FILE *arquivo;
21.     double somaOrdenadas = 0, somaAbcissas = 0,
22.             somaOrdenadasQuadrado = 0, somaAbcissasQuadrado = 0,
23.             somaXvezesY = 0, correlacao,
24.             numeroAbcissa, numeroOrdenada,
25.             numerador, denominador_1, denominador;
26.     int c;
27.     detectaRoboETipoDeGrafico();
28.
29.     if( (strcmp(nomeTipoGrafico,"M1")) == 0)
30.         arquivo = fopen("tabela1Minuto.csv","rt");
31.     else if( (strcmp(nomeTipoGrafico,"M5")) == 0)
32.         arquivo = fopen("tabela5Minutos.csv","rt");
33.     else if( (strcmp(nomeTipoGrafico,"H1")) == 0)
```

```

34.         arquivo = fopen("tabela1Hora.csv", "rt");
35.     else
36.         printf("Erro, tabela nao encontrada\n");
37.
38.     for(c=0; c<QUANTIDADE_CANDLES; c++){
39.         fscanf(arquivo, "%lf", &leituraCotacoes[c]);
40.     }
41.
42.     for(c=0; c<tempoCorrelacao; c++){
43.         numeroAbcissa = leituraCotacoes[c];
44.         numeroOrdenada = leituraCotacoes[c+1];
45.
46.         somaAbcissas = somaAbcissas + numeroAbcissa;
47.         somaAbcissasQuadrado += (numeroAbcissa*numeroAbcissa);
48.         somaOrdenadas = somaOrdenadas + numeroOrdenada;
49.         somaOrdenadasQuadrado += (numeroOrdenada*numeroOrdenada);
50.         somaXvezesY = somaXvezesY + (numeroOrdenada*numeroAbcissa);
51.     }
52.
53.     numerador =((tempoCorrelacao*somaXvezesY)-((somaAbcissas)*(somaOrdenadas)));
54.     denominador_1 =((tempoCorrelacao*somaAbcissasQuadrado)-(somaAbcissas*somaAbcissas))*((tempoCorrelacao*somaOrdenadasQuadrado)-(somaOrdenadas*somaOrdenadas));
55.
56.
57.     denominador = sqrt(denominador_1);
58.     correlacao = numerador/denominador;
59.
60.     return correlacao;
61.
62.     printf("%f\n",correlacao);
63.     fclose(arquivo);
64. }
65. void detectaRoboETipoDeGrafico(){
66.     FILE *arquivo;
67.

```

```
68.     arquivo = fopen("criterioEntrada.txt","rt");
69.     fgets(nomeRobo, 50,arquivo);
70.     fgets(nomeTipoGrafico, 3,arquivo);
71.     fclose(arquivo);
72. }
```

testeCorrelacaoDePearson.c

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <CUnit/Basic.h>
4.
5. int init_suite(void) {
6.     return 0;
7. }
8. int clean_suite(void) {
9.     return 0;
10. }
11.
12. double metodoCorrelacao(int tempoCorrelacao);
13.
14. void testMetodoCorrelacao() {
15.     int tempoCorrelacao = 21;
16.     double resultadoCorrelacao = metodoCorrelacao(tempoCorrelacao);
17.
18.     CU_ASSERT_DOUBLE_EQUAL(0.748820,resultadoCorrelacao, 0.001 );
19. }
20. int main() {
21.     CU_pSuite pSuite = NULL;
22.
23.     /* Initialize the CUnit test registry */
24.     if (CUE_SUCCESS != CU_initialize_registry())
25.         return CU_get_error();
26.
27.     /* Add a suite to the registry */
```

```

28.     pSuite = CU_add_suite("correlacaoLinearTeste", init_suite, clean_suite);
29.     if (NULL == pSuite) {
30.         CU_cleanup_registry();
31.         return CU_get_error();
32.     }
33.     /* Add the tests to the suite */
34.     if ((NULL == CU_add_test(pSuite, "testMetodoCorrelacao", testMetodoCorrelacao))) {
35.         CU_cleanup_registry();
36.         return CU_get_error();
37.     }
38.     /* Run all tests using the CUnit Basic interface */
39.     CU_basic_set_mode(CU_BRM_VERBOSE);
40.     CU_basic_run_tests();
41.     CU_cleanup_registry();
42.     return CU_get_error();
43. }
```

fibonacci

```

1. #include <stdio.h>
2. #include <stdlib.h>
3.
4. double calculoSuporte(int quantidadeVelas);
5. double calculoResistencia(int quantidadeVelas);
6. double calculoRegressaoFibonacci(double fatorDeRegressao, int quantidadeVelas);
7.
8. int main(){
9.
10.     printf("Suporte = %lf, resistencia =
11.             %lf\n", calculoSuporte(13), calculoResistencia(13));
12.     printf("Regressao De Fibonacci = %lf\n", calculoRegressaoFibonacci(0.23, 13));
13.     return 0;
14.
15. double calculoSuporte(int quantidadeVelas){
16.     FILE *arquivo;
```

```
17.     double cotacao[quantidadeVelas];
18.     double suporte = 0;
19.     int i;
20.     arquivo = fopen("dadosFibonacci.txt","rt");
21.
22.     for(i = 0; i < quantidadeVelas; i++){
23.         fscanf(arquivo, "%lf",&cotacao[i]);
24.
25.         if(suporte < cotacao[i])
26.             suporte = cotacao[i];
27.     }
28.     fclose(arquivo);
29.     return suporte;
30. }
31. double calculoResistencia(int quantidadeVelas){
32.     FILE *arquivo;
33.     double cotacao[quantidadeVelas];
34.     double resistencia = 777;
35.     int i;
36.
37.     arquivo = fopen("dadosFibonacci.txt","rt");
38.
39.     for(i = 0; i < quantidadeVelas; i++){
40.         fscanf(arquivo, "%lf",&cotacao[i]);
41.         if(resistencia > cotacao[i])
42.             resistencia = cotacao[i];
43.     }
44.     fclose(arquivo);
45.     return resistencia;
46. }
47.
48. double calculoRegressaoFibonacci(double fatorDeRegressao, int quantidadeVelas){
49.     double variacaoDePontos = calculoSuporte(quantidadeVelas) -
```

```
50.     return variacaoDePontos*fatorDeRegressao;
51. }
```

testeFibonacci

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <CUUnit/Basic.h>
4.
5. int init_suite(void) {
6.     return 0;
7. }
8. int clean_suite(void) {
9.     return 0;
10. }
11. double calculoRegressaoFibonacci(double fatorDeRegressao, int quantidadeVelas);
12.
13. void testCalculoRegressaoFibonacci() {
14.     double resultadoRegressaoFibonacci = calculoRegressaoFibonacci(0.23, 13);
15.     CU_ASSERT_DOUBLE_EQUAL(0.162380,resultadoRegressaoFibonacci, 0.001 );
16. }
17.
18. double calculoResistencia(int quantidadeVelas);
19.
20. void testCalculoResistencia() {
21.     int quantidadeVelas = 13;
22.     CU_ASSERT_DOUBLE_EQUAL(136.290,calculoResistencia(quantidadeVelas), 0.001 );
23. }
24.
25. double calculoSuptore(int quantidadeVelas);
26.
27. void testCalculoSuptore() {
28.     int quantidadeVelas = 13;
29.     CU_ASSERT_DOUBLE_EQUAL(136.996,calculoSuptore(quantidadeVelas), 0.001 );
30. }
```

```

31.
32. int main() {
33.     CU_pSuite pSuite = NULL;
34.     /* Initialize the CUnit test registry */
35.     if (CUE_SUCCESS != CU_initialize_registry())
36.         return CU_get_error();
37.     /* Add a suite to the registry */
38.     pSuite = CU_add_suite("fibonacciTeste", init_suite, clean_suite);
39.     if (NULL == pSuite) {
40.         CU_cleanup_registry();
41.         return CU_get_error();
42.     }
43.     /* Add the tests to the suite */
44.     if ((NULL == CU_add_test(pSuite, "testCalculoRegressaoFibonacci",
45.                             testCalculoRegressaoFibonacci)) ||
46.         (NULL == CU_add_test(pSuite, "testCalculoResistencia",
47.                             testCalculoResistencia)) ||
48.         (NULL == CU_add_test(pSuite, "testCalculoSuporTe", testCalculoSuporTe))) {
49.         CU_cleanup_registry();
50.         return CU_get_error();
51.     }
52.     /* Run all tests using the CUnit Basic interface */
53.     CU_basic_set_mode(CU_BRM_VERBOSE);
54.     CU_basic_run_tests();
55.     CU_cleanup_registry();
56.     return CU_get_error();
57. }
```

minimosQuadarados.c

```

1. #include <stdio.h>
2. #include <stdlib.h>
3.
4. double calculoCoeficienteLinear();
5. double calculoCoeficienteAngular();
6.
```

```
7. double calculoCoeficienteLinear(int quantidadeVelas){
8.     FILE *arquivo;
9.     double x[quantidadeVelas], y[quantidadeVelas];
10.    double soma_x = 0, soma_y = 0;
11.    double numerador, denominador;
12.    double variacaoLinear;
13.    int i;
14.
15.    arquivo = fopen("dadosMinimosQuadrados.txt", "rt");
16.
17.    for(i = 1; i < quantidadeVelas; i++){
18.        fscanf(arquivo, "%lf", &x[i]);
19.        fscanf(arquivo, "%lf", &y[i]);
20.        soma_x = soma_x + x[i];
21.        soma_y = soma_y + y[i];
22.    }
23.
24.    for(i = 1; i < quantidadeVelas; i++){
25.        numerador = x[i]*(y[i] - soma_x/quantidadeVelas);
26.        denominador = y[i]*(x[i] - soma_y/quantidadeVelas);
27.    }
28.    variacaoLinear = numerador/denominador;
29.    fclose(arquivo);
30.    return variacaoLinear;
31. }
32.
33. double calculoCoeficienteAngular(int quantidadeVelas){
34.     FILE *arquivo;
35.     double x[quantidadeVelas], y[quantidadeVelas];
36.     double soma_x = 0, soma_y = 0;
37.     double variacaoAngular;
38.     int i;
39.     arquivo = fopen("dadosMinimosQuadrados.txt", "rt");
40. }
```

```

41.     for(i = 1; i < quantidadeVelas; i++){
42.         fscanf(arquivo, "%lf",&x[i]);
43.         fscanf(arquivo, "%lf",&y[i]);
44.         soma_x = soma_x + x[i];
45.         soma_y = soma_y + y[i];
46.     }
47.
48.     variacaoAngular = soma_y/quantidadeVelas -
        (calculoCoeficienteLinear(quantidadeVelas)*soma_x/quantidadeVelas);
49.
50.     fclose(arquivo);
51.     return variacaoAngular;
52. }
```

testeMinimosQuadrados

```

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <CUUnit/Basic.h>
4.
5. int init_suite(void) {
6.     return 0;
7. }
8.
9. int clean_suite(void) {
10.    return 0;
11. }
12.
13. double calculoRegressaoFibonacci(double fatorDeRegressao, int quantidadeVelas);
14.
15. void testCalculoRegressaoFibonacci() {
16.     double resultadoRegressaoFibonacci = calculoRegressaoFibonacci(0.23, 13);
17.
18.     CU_ASSERT_DOUBLE_EQUAL(0.162380, resultadoRegressaoFibonacci, 0.001 );
19. }
20.
```

```
21. double calculoResistencia(int quantidadeVelas);  
22.  
23. void testCalculoResistencia() {  
24.     int quantidadeVelas = 13;  
25.  
26.     CU_ASSERT_DOUBLE_EQUAL(136.290,calculоНdResistencia(quantidadeVelas), 0.001 );  
27. }  
28.  
29. double calculoSuporte(int quantidadeVelas);  
30.  
31. void testCalculoSuporte() {  
32.     int quantidadeVelas = 13;  
33.  
34.     CU_ASSERT_DOUBLE_EQUAL(136.996,calculоНdSuporte(quantidadeVelas), 0.001 );  
35. }  
36.  
37. int main() {  
38.     CU_pSuite pSuite = NULL;  
39.  
40.     /* Initialize the CUnit test registry */  
41.     if (CUE_SUCCESS != CU_initialize_registry())  
42.         return CU_get_error();  
43.  
44.     /* Add a suite to the registry */  
45.     pSuite = CU_add_suite("fibonacciTeste", init_suite, clean_suite);  
46.     if (NULL == pSuite) {  
47.         CU_cleanup_registry();  
48.         return CU_get_error();  
49.     }  
50.  
51.     /* Add the tests to the suite */  
52.     if ((NULL == CU_add_test(pSuite, "testCalculoRegressaoFibonacci",  
testCalculoRegressaoFibonacci)) ||
```

```
53.         (NULL == CU_add_test(pSuite, "testCalculoResistencia",
54.             testCalculoResistencia)) ||
55.         (NULL == CU_add_test(pSuite, "testCalculoSuporte", testCalculoSuporte))) {
56.     CU_cleanup_registry();
57.     return CU_get_error();
58. }
59. /* Run all tests using the CUnit Basic interface */
60. CU_basic_set_mode(CU_BRM_VERBOSE);
61. CU_basic_run_tests();
62. CU_cleanup_registry();
63. return CU_get_error();
64. }
```

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 correlações "), write(Resposta), nl,
9   qtde(Resposta, QuantidadeCorrelações),
10  write("Quantidade correlações "), write(QuantidadeCorrelações), nl,
11  somaCorrelações(Resposta, Soma),
12  write("Quantidade correlações "), write(Soma), nl,
13  mediaCorrelações(MediaCorrelações),
14  write("Media correlações vencedoras: "), write(MediaCorrelações),
15  %tell abre o arquivo para escrita
16  open('respostaProlog.txt', write, Arquivo),
17  write(Arquivo, MediaCorrelações),
18  close(Arquivo),
19  nl, told.
20
21 % Soma de todos os elementos da lista de correlações
22 somaCorrelações([],0).
23 somaCorrelações([Elem|Cauda],Soma):-somaCorrelações(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 mediaCorrelações(Media):-
31   findall(CLVencedora, metodosNumericos(ganhou, X, Y, CLVencedora), Resposta),
32   somaCorrelações(Resposta, Soma),
33   qtde(Resposta, QuantidadeCorrelações),
34   Media is Soma/QuantidadeCorrelações.
```

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(ganhou, X, Y, CLVencedora), Resposta),  
18     qtde(Resposta, QuantidadeCorrelacoes),  
19     assertion(QuantidadeCorrelacoes == 12).  
20  
21 test(somaCorrelacoes) :-  
22     findall(CLVencedora, metodosNumericos(ganhou, X, Y, CLVencedora), Resposta),  
23     somaCorrelacoes(Resposta, Soma),  
24     assertion((Soma == 10.800000000000002)).  
25  
26 test(mediaCorrelacoes) :-  
27     findall(CLVencedora, metodosNumericos(ganhou, 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 Multiagente

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("\r\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("\r\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(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(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
76         FileReader("../criterioEntrada.txt")).useDelimiter("\\|\\n");
77     scanner.nextLine();
78     tipoGrafico = scanner.nextLine();
79     tempoEmMiliSegundos = converteTipoGraficoEmTempo(tipoGrafico);
80 } catch (FileNotFoundException e) {
81     e.printStackTrace();
82 }
83 return tempoEmMiliSegundos;
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 serviço
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. }

```


APÊNDICE K – Análise estática de código-fonte

InvestMVC - Análise de Qualidade de Código Fonte Inicial								
	ACC	ACCM	ANPM	DIT	NPA	SC	ACC	Acoplamento
Excelente	[0, 2[[0, 3[[0, 2[[0, 2[[0, 1[[0, 12[ACCM	Complexidade Ciclomática
Bom	[2, 7[[3, 5[[2, 3[[2, 4[[1, 2[[12, 28[ANPM	Números de parâmetros por método
Regular	[7, 15[[5, 7[[3, 5[[4, 6[[2, 3[[28, 51[DIT	Herança
Preocupante	[15, ∞[[7, ∞[[5, ∞[[6, ∞[[3, ∞[[51, ∞[NPA	Encapsulamento
SC							SC	Coesão e Acoplamento
Pacote comportamentos								
Comprar.java	0	3	0	1	0	0		
EnviaMinimosQuadrados.java	0	3,5	0	1	0	0		
EnviarCorrelacao.java	0	3,5	0	1	0	0		
EnviarFibonacci.java	0	3,5	0	1	0	0		
EscreveArquivo.java	0	1	2	0	0	0		
EsperarNotificacao.java	0	3	0	1	0	0		
LeituraArquivo.java	0	2,090909091	0,2727272727	0	0	0		
ManipulaStrings.java	0	1	0	0	0	0		
NotificarNegociadores.java	0	2	0	1	0	0		
NotificarTendencia.java	0	1,5	0	1	0	0		
ProcurarCotacoes.java	0	1,75	0,5	1	1	0		
RegistrarNoDF.java	0	1,5	1	1	0	0		
RodarComandos.java	0	2,5	1	0	0	0		
Vender.java	0	3	0	1	0	0		
Pacote investidores								
Comprador.java	2	1,5	0	1	2	0		
Controlador.java	0	4	0	1	0	1		
Controlador.java	0	3	0	1	0	1		
Controlador.java	0	1	0	1	0	1		
Tendencia.java	0	1	0	1	0	1		
Vendedor.java								
Pacote metodosNumericos								
ConsultorAgente.java	0	1,5	0	1	0	0		

CorrelacaoAgente.java	0	1	0	1	0	0			
FibonacciAgente.java	0	1	0	1	0	0			
MinimosQuadradosAgente.java									

InvestMVC - Análise de Qualidade de Código Fonte Final								
	ACC	ACCM	ANPM	DIT	NPA	SC	ACC	Acoplamento
Excelente	[0, 2[[0, 3[[0, 2[[0, 2[[0, 1[[0, 12[ACCM	Complexidade Ciclomática
Bom	[2, 7[[3, 5[[2, 3[[2, 4[[1, 2[[12, 28[ANPM	Números de parâmetros por método
Regular	[7, 15[[5, 7[[3, 5[[4, 6[[2, 3[[28, 51[DIT	Herança
Preocupante	[15, ∞[[7, ∞[[5, ∞[[6, ∞[[3, ∞[[51, ∞[NPA	Encapsulamento
SC							SC	Coesão e Acoplamento
Pacote comportamentos								
Comprar.java	0	3	0	1	0	0		
EnviaMinimosQuadrados.java	0	2,5	0	1	0	0		
EnviarCorrelacao.java	0	3,5	0	1	0	0		
EnviarFibonacci.java	0	2,5	0	1	0	0		
EscreveArquivo.java	0	1	2	0	0	0		
EsperarNotificacao.java	0	3	0	1	0	0		
LeituraArquivo.java	0	2,090909091	0,2727272727	0	0	0		
ManipulaStrings.java	0	1	0	0	0	0		
NotificarNegociadores.java	0	2	0	1	0	0		
NotificarTendencia.java	0	1,5	0	1	0	0		
ProcurarCotacoes.java	0	1,75	0,5	1	1	0		
RegistrarNoDF.java	0	1,5	1	1	0	0		
RodarComandos.java	0	2,5	1	0	0	0		
Vender.java	0	3	0	1	0	0		
Pacote investidores								
Comprador.java	2	1,5	0	1	0	0		
Controlador.java	0	4	0	1	0	1		
Controlador.java	0	3	0	1	0	1		
Controlador.java	0	1	0	1	0	1		
Tendencia.java	0	1	0	1	0	1		
Vendedor.java								
Pacote metodosNumericos								
ConsultorAgente.java	0	1,5	0	1	0	0		

CorrelacaoAgente.java	0	1	0	1	0	0			
FibonacciAgente.java	0	1	0	1	0	0			
MinimosQuadradosAgente.java									
Pacotes									
comportamentosComuns									
execucao	2	3	0	1	0	1			
investidores	2	4	0	1	1	1			
metodosNumericos	0	2	0	1	0	0			
teste	0	1	1	0	0	0			

InvestMVC - Análise de Qualidade de Código Fonte Inicial por pacote						
	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, ∞[
Pacotes						
comportamentosComuns	0	3,5	2	1	1	0
execucao	2	3	0	1	0	1
investidores	2	4	0	1	2	1
metodosNumericos	0	2	0	1	0	0

InvestMVC - Análise de Qualidade de Código Fonte Final por pacote						
	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, ∞[
Pacotes						
comportamentosComuns	0	2,5	2	1	1	0
execucao	2	3	0	1	0	1
investidores	2	4	0	1	1	1
metodosNumericos	0	2	0	1	0	0

APÊNDICE L – Histórico de Resultados

FXPRO Financial Services Ltd

Account: 6640392

Name: Robô MQL TCC2

Currency: USD Leverage: 1:500

Ticket	Horário Entrada	Tipo	Item	Volume	Preço	S / L	T / P	Lucro	Horário Saída	Balanço
29019743	2015.05.18 18:01:02	sell	eurusd	0.10	1.13114	1.14044	1.13114	56.70	2015.05.18 21:03:23	1056.70
29019744	2015.05.19 06:06:08	sell	eurusd	0.10	1.13167	1.13167	1.12596	-17.90	2015.05.19 08:12:47	1038.80
29019745	2015.05.19 09:50:07	sell	eurusd	0.10	1.12519	1.13108	1.12519	40.00	2015.05.19 10:03:37	1078.80
29019746	2015.05.19 16:33:47	sell	eurusd	0.10	1.10969	1.11954	1.10969	59.18	2015.05.29 12:35:27	1137.98
29019747	2015.06.01 05:29:26	sell	eurusd	0.10	1.09044	1.09731	1.09044	44.80	2015.06.01 11:35:45	1182.78
29019748	2015.06.01 12:41:19	sell	eurusd	0.10	1.09502	1.09502	1.08658	-31.90	2015.06.01 15:30:42	1150.88
29019749	2015.06.01 20:58:47	sell	eurusd	0.10	1.09583	1.09583	1.08617	-36.81	2015.06.02 09:54:21	1114.07
29019750	2015.06.03 10:53:32	sell	eurusd	0.10	1.10969	1.11745	1.10969	49.20	2015.06.03 15:30:24	1163.27
29019751	2015.06.04 10:39:09	sell	eurusd	0.10	1.12647	1.12647	1.12048	-19.40	2015.06.04 10:41:47	1143.87
29019752	2015.06.04 17:00:17	sell	eurusd	0.10	1.12138	1.13397	1.12138	72.69	2015.06.05 01:40:24	1216.56
29019753	2015.06.05 02:41:23	sell	eurusd	0.10	1.12337	1.12337	1.11542	-29.40	2015.06.05 09:00:42	1187.16
29019754	2015.06.05 15:29:45	sell	eurusd	0.10	1.12652	1.12652	1.11897	-26.90	2015.06.05 15:30:27	1160.26
29019755	2015.06.08 01:30:42	sell	eurusd	0.10	1.11265	1.11265	1.10686	-18.30	2015.06.08 09:52:10	1141.96
29019756	2015.06.09 10:53:21	sell	eurusd	0.10	1.12302	1.13252	1.12302	57.70	2015.06.09 15:38:13	1199.66
29019757	2015.06.09 16:19:47	sell	eurusd	0.10	1.12838	1.12838	1.11885	-37.50	2015.06.09 18:55:27	1162.16
29019758	2015.06.11 15:32:39	sell	eurusd	0.10	1.12611	1.12611	1.11561	-42.40	2015.06.11 15:43:49	1119.76
29019759	2015.06.12 07:50:36	sell	eurusd	0.10	1.11980	1.12569	1.11980	40.00	2015.06.12 11:49:27	1159.76
29019760	2015.06.12 13:03:29	sell	eurusd	0.10	1.12260	1.12260	1.11253	-40.30	2015.06.12 15:20:19	1119.46
29019761	2015.06.12 23:24:18	sell	eurusd	0.10	1.12662	1.12795	1.12224	-4.60	2015.06.12 23:59:17	1114.86

FXPRO Financial Services Ltd

Account: 6640380

Name: InvestMVC TCC2

Currency: USD Leverage: 1:500

Ticket	Horário Entrada	Tipo	Item	Volume	Preço	S / L	T / P	Lucro	Horário Saída	Balanco
30324567	2015.05.18 18:01:17	sell	eurusd	0.10	1.13117	1.14057	1.13064	56.20	2015.05.18 21:03:02	1056.20
30334456	2015.05.19 06:06:34	sell	eurusd	0.10	1.13169	1.13169	1.12586	-17.80	2015.05.19 08:12:55	1038.40
30345567	2015.05.19 09:50:01	sell	eurusd	0.10	1.12519	1.13108	1.12519	40.00	2015.05.19 10:03:43	1078.40
30353224	2015.05.19 16:33:56	sell	eurusd	0.10	1.10964	1.11954	1.10869	60.18	2015.05.29 12:35:32	1137.58
30353352	2015.06.01 05:29:34	sell	eurusd	0.10	1.09032	1.09731	1.09044	44.80	2015.06.01 11:35:33	1183.38
30353386	2015.06.01 12:41:22	sell	eurusd	0.10	1.09451	1.09452	1.08634	-31.40	2015.06.01 15:30:42	1151.98
30360234	2015.06.01 20:58:55	sell	eurusd	0.10	1.09586	1.09586	1.08611	-36.75	2015.06.02 09:54:32	1115.23
30478743	2015.06.03 10:53:35	sell	eurusd	0.10	1.10968	1.11744	1.10967	49.22	2015.06.03 15:30:34	1164.45
31346578	2015.06.04 10:39:31	sell	eurusd	0.10	1.12642	1.12642	1.12038	-19.36	2015.06.04 10:41:43	1145.09
31356789	2015.06.04 17:00:33	sell	eurusd	0.10	1.12139	1.13397	1.12138	72.69	2015.06.05 01:40:53	1217.78
31374356	2015.06.05 02:41:21	sell	eurusd	0.10	1.12337	1.12337	1.11542	-29.40	2015.06.05 09:00:22	1188.38
31407623	2015.06.05 15:29:56	sell	eurusd	0.10	1.12692	1.12692	1.11856	-30.90	2015.06.05 15:30:27	1157.48
32435678	2015.06.08 01:30:43	sell	eurusd	0.10	1.11269	1.11269	1.10676	-17.30	2015.06.08 09:52:12	1140.18
32456788	2015.06.09 10:53:22	sell	eurusd	0.10	1.12301	1.13251	1.12301	57.60	2015.06.09 15:38:11	1197.78
32657843	2015.06.09 16:19:43	sell	eurusd	0.10	1.12827	1.12827	1.11883	-36.40	2015.06.09 18:55:37	1161.38
32657943	2015.06.11 15:32:44	sell	eurusd	0.10	1.12610	1.12610	1.11521	-37.30	2015.06.11 15:43:43	1124.08
32723456	2015.06.12 07:50:46	sell	eurusd	0.10	1.11985	1.12565	1.11982	40.20	2015.06.12 11:49:34	1164.28
33452378	2015.06.12 13:03:43	sell	eurusd	0.10	1.12250	1.12250	1.11242	-39.30	2015.06.12 15:20:21	1124.98
33478798	2015.06.12 23:24:22	sell	eurusd	0.10	1.126620	1.12795	1.12227	-4.20	2015.06.12 23:59:23	1120.58

Anexos

ANEXO A – Protocolo de Estudo de Caso

CASE STUDY PROTOCOL

1. Background

- a) identify previous research on the topic
- b) define the main research question being addressed by this study
- c) identify any additional research questions that will be addressed 2.

2. Design

- a) identify whether single-case or multiple-case and embedded or holistic designs will be used, and show the logical links between these and the research questions
- b) describe the object of study (e.g. a new testing procedure; a new feature in a browser)
- c) identify any propositions or sub-questions derived from each research question and the measures to be used to investigate the propositions

3. Case Selection

- a) Criteria for case selection

4. Case Study Procedures and Roles

- a) Procedures governing field procedures
- b) Roles of case study research team members

5. Data Collection

- a) identify the data to be collected b) define a data collection plan c) define how the data will be stored

6. Analysis

- a) identify the criteria for interpreting case study findings
- b) identify which data elements are used to address which research question/sub question/proposition and how the data elements will be combined to answer the question
- c) consider the range of possible outcomes and identify alternative explanations of the outcomes, and identify any information that is needed to distinguish between these
- d) the analysis should take place as the case study task progresses

7. Plan Validity

- a) general: check plan against Höst and Runeson's (2007) checklist items for the design and the data collection plan
- b) construct validity - show that the correct operational measures are planned for the concepts being studied. Tactics for ensuring this include using multiple sources of evidence, establishing chains of evidence, expert reviews of draft protocols and reports
- c) internal validity - show a causal relationship between outcomes and intervention/treatment (for explanatory or causal studies only).
- d) external validity – identify the domain to which study finding can be generalized. Tactics include using theory for single-case studies and using multiple-case studies to investigate outcomes in different contexts.

8. 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).

9. Reporting

Using a Protocol Template for Case Study Planning EASE 2008 Identify target audience, relationship to larger studies (YIN, 2003).

10. Schedule

Give time estimates for all of the major steps: Planning, Data Collection, Data Analysis, Reporting. Note Data Collection and Data Analysis are not expected to be sequential stages

11. Appendices

- a) Validation: report results of checking plan against Höst and Runeson's (2007) checklist items
- b) Divergences: update while conducting the study by noting any divergences from the above steps.