



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

Ferramenta Multiparadigma para o Mercado de Moedas apoiada por Métodos Matemáticos

Autor: Cleiton da Silva Gomes, Vanessa Barbosa Martins

Orientador: Dr^a. Milene Serrano

Brasília, DF

2014



Cleiton da Silva Gomes, Vanessa Barbosa Martins

Ferramenta Multiparadigma para o Mercado de Moedas apoiada por Métodos Matemáticos

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^a. Milene Serrano

Coorientador: Dr. Ricardo Matos Chaim

Brasília, DF

2014

Cleiton da Silva Gomes, Vanessa Barbosa Martins

Ferramenta Multiparadigma para o Mercado de Moedas apoiada por Métodos Matemáticos/ Cleiton da Silva Gomes, Vanessa Barbosa Martins. – Brasília, DF, 2014-

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

Orientador: Dr^a. Milene Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2014.

1. Forex. 2. Paradigmas de Programação. I. Dr^a. Milene Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Ferramenta Multiparadigma para o Mercado de Moedas apoiada por Métodos Matemáticos

CDU 02:141:005.6

Cleiton da Silva Gomes, Vanessa Barbosa Martins

Ferramenta Multiparadigma para o Mercado de Moedas apoiada por Métodos Matemáticos

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, 01 de junho de 2013:

Dr^a. Milene Serrano
Orientador

Dr. Ricardo Matos Chaim
Coorientador

Titulação e Nome do Professor
Convidado 01
Convidado 1

Titulação e Nome do Professor
Convidado 02
Convidado 2

Brasília, DF
2014

*Dedicamos esse trabalho aos que fazem ao invés de apenas reclamar.
Olhar pelo retrovisor e dizer o outro está errado é muito fácil. Dedicamos esse trabalho
aqueles que aprendem errando*

Agradecimentos

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

Eu, Vanessa Barbosa, agradeço à todos de que alguma maneira torceram por mim ao longo destes 5 anos, meus amigos: Álex, André Cruz, André Mateus, Hebert, Jefferson, Luanna, Mônica, Ramon, Sarah. Agradeço também aos meus pais e ao Cleiton Gomes que ficaram ao meu lado nos momentos mais críticos. Por fim agradeço aquele que meu deus 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 desse TCC 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 TCC propõe o desenvolvimento de uma ferramenta multiparadigma para operar de forma automatizada no Mercado de Moedas: ferramenta InvestMVC. Cada Paradigma de Programação presente na ferramenta InvestMVC possui seu papel bem definido e suas vantagens de utilização bem esclarecidas. Foram selecionados Métodos Matemáticos para serem implementados na ferramenta através de um Protocolo Experimental bem elaborado. A Qualidade de Software foi uma vertente bem definida no escopo da ferramenta. Testes dinâmicos e estáticos foram estabelecidos. Para que os produtos de trabalho deste TCC fossem elaborados com sucesso, foi definido uma metodologia de pesquisa, com ênfase no objetivo de estudo e nos procedimentos técnicos. Para execução da pesquisa, foi realizada uma adaptação do Scrum, levando em consideração as atividades alocadas no cronograma e o contexto de Engenharia de Software aliado com Mercado de Moedas.

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

Abstract

This is the english abstract.

Key-words: latex. abntex. text editoration.

Lista de ilustrações

Figura 1 – Reta de Suporte.	31
Figura 2 – Reta de Resistência.	31
Figura 3 – Equação da reta Mínimos Quadrados.	33
Figura 4 – Classificação da Correlação Linear.	34
Figura 5 – Determinação da Correlação Linear.	35
Figura 6 – Fórmula de Fibonacci sem uso de recorrência.	36
Figura 7 – Indicador Estocástico.	37
Figura 8 – Equação de Média Móvel.	38
Figura 9 – Indicador Média Móvel	38
Figura 10 – Arquitetura de von Neumann.	40
Figura 11 – A essência do Paradigma Orientado a Objetos.	41
Figura 12 – Defeito x Erro x Falha.	48
Figura 13 – Níveis de teste e seu desenvolvimento.	49
Figura 14 – Intervalo para interpretação das métricas.	52
Figura 15 – Atividades da Pesquisa.	54
Figura 16 – Relatório de simulação no período agosto 2012-2013 do expert Corre- lacaoPearson.mql	65
Figura 17 – Relatório de simulação no período agosto 2013-2014 do expert Corre- lacaoPearson.mql	66
Figura 18 – Gráfico gerado pela simulação do expert CorrelacaoPearson.mql no pe- ríodo agosto 2012-2013	66
Figura 19 – Gráfico gerado pela simulação do expert CorrelacaoPearson.mql no pe- ríodo agosto 2013-2014	67
Figura 20 – Relatório de simulação no período agosto 2012-2013 do expert	67
Figura 21 – Relatório de simulação no período agosto 2012-2013 do expert	67
Figura 22 – Relatório de simulação no período agosto 2012-2013 do expert	68
Figura 23 – Relatório de simulação no período agosto 2013-2014 do expert	68
Figura 24 – Relatório de simulação no período agosto 2012-2013 do expert	69
Figura 25 – Relatório de simulação no período agosto 2013-2014 do expert	69
Figura 26 – Gráfico gerado pela simulação do expert Fibonacci.mql no período agosto 2012-2013	69
Figura 27 – Gráfico gerado pela simulação do expert Fibonacci.mql no período agosto 2013-2014	70
Figura 28 – Relatório de simulação no período agosto 2012-2013 do expert Esto- castico.mql	70

Figura 29 – Relatório de simulação no período agosto 2013-2014 do expert Estocastico.mql	71
Figura 30 – Gráfico gerado pela simulação do expert Estocastico.mql no período agosto 2012-2013	71
Figura 31 – Gráfico gerado pela simulação do expert Estocastico.mql no período agosto 2013-2014	71
Figura 32 – Relatório de simulação no período agosto 2012-2013 do expert Media-Movel.mql	72
Figura 33 – Relatório de simulação no período agosto 2013-2014 do expert Media-Movel.mql	72
Figura 34 – Gráfico gerado pela simulação do expert MediaMovel.mql no período agosto 2012-2013	73
Figura 35 – Gráfico gerado pela simulação do expert MediaMovel.mql no período agosto 2013-2014	73
Figura 36 – Diagrama de Sequência InvestMVC	82
Figura 37 – Diagrama de Classe InvestMVC componente Orientado a Objetos . . .	83
Figura 38 – Diagrama de sequência InvestMVC componente Orientado a Objetos .	84
Figura 39 – Componente Funcional InvestMVC	85
Figura 40 – Diagrama de Sequência do Componente Funcional InvestMVC	85
Figura 41 – Diagramas de Componentes e de Sequência do Componente Estruturado InvestMVC.	86
Figura 42 – Diagrama de Classe do Componente Multiagente InvestMVC	87
Figura 43 – Diagrama de Sequência do Componente Lógico InvestMVC.	88
Figura 44 – Diagrama de Sequência do Componente MQL InvestMVC.	88
Figura 45 – Diagrama de Sequência InvestMVC	89
Figura 46 – Cobertura de código fonte do Componente Estruturado InvestMVC . .	91
Figura 47 – Resultado da Qualidade de Código Fonte do Componente Estruturado	93
Figura 48 – Estrutura Interna Componente Multiagente	93
Figura 49 – Componente MQL instalado na plataforma MetaTrader	94

Lista de tabelas

Tabela 1 – Atividades e objetivos metodologia de pesquisa	55
Tabela 2 – Cronograma simplificado	58
Tabela 3 – Atividades e objetivos metodologia de pesquisa	75
Tabela 4 – Atividades e objetivos metodologia de pesquisa	92

Lista de abreviaturas e siglas

OO	Orientado a objetos
ACC	Acoplamento
ACCM	Complexidade Ciclomática
ANPM	Número de parâmetros
CPU	Unidade Central de Processamento
DIT	Herança
IEEE	Institute of Electrical and Electronics Engineers
MVC	Model-View-Controller
NPA	Encapsulamento
SC	Coesão e Acoplamento
SMA	Sistema Multi-Agente
TCC	Trabalho de Conclusão de Curso
US	User Story

Sumário

1	INTRODUÇÃO	25
1.1	Contextualização	25
1.2	Problema de Pesquisa	25
1.3	Justificativa	25
1.4	Objetivos	26
1.4.1	Objetivo Geral	26
1.4.2	Objetivos Específicos	26
1.5	Organização do Trabalho	27
2	REFERENCIAL TEÓRICO	29
2.1	Contexto Financeiro	29
2.1.1	Mercado de Moedas	29
2.1.2	Alavancagem	30
2.1.3	Suporte	30
2.1.4	Resistência	31
2.2	Métodos Matemáticos	32
2.2.1	Método de Mínimos Quadrados	32
2.2.2	Método de Correlação Linear	33
2.2.3	Método de Fibonacci	35
2.2.4	Estocástico	36
2.2.5	Média Móvel	37
2.3	Paradigmas de Programação	38
2.3.1	Paradigma Procedural	39
2.3.2	Paradigma Orientado a Objetos	41
2.3.3	Paradigma Orientado a Agentes	43
2.3.4	Programação Declarativa	45
2.3.4.1	Paradigma Funcional	45
2.3.4.2	Paradigma Lógico	46
2.4	Teste de Software	47
2.4.1	Testes Unitários	48
2.4.2	Teste de integração	49
2.4.3	Teste de Aceitação	50
2.5	Qualidade de Software	50
2.5.1	Métricas de Qualidade de Código Fonte	51
2.5.2	Análise Estática de Código Fonte	51

2.5.3	Ferramentas de Análise Estática	51
3	METODOLOGIA DE PESQUISA	53
3.1	Classificação da Pesquisa	53
3.2	Atividades da Pesquisa	54
3.2.1	Descrição dos Objetivos das Atividades de Pesquisa	55
3.3	Execução da Pesquisa	56
4	CRONOGRAMA	57
5	SUPORTE TECNOLÓGICO	59
5.1	Ferramentas para teste unitário e teste de integração	59
5.1.1	CUnit	59
5.1.2	JUnit	59
5.1.3	HUnit	59
5.1.4	PIUnit	60
5.2	Ferramenta para teste funcional: Cucumber	60
5.3	Ferramentas de cobertura de teste	60
5.3.1	EclEmma	60
5.3.2	HPC	60
5.4	Ferramenta de Análise Estática de Código Fonte	60
5.4.1	Analizo	60
5.4.2	Sonar	61
5.5	Ferramentas de Mercado de Moedas	61
5.5.1	MetaTrade	61
5.5.2	MetaEditor	61
5.5.3	Alpari-UK	61
5.5.4	FXDD	61
6	PROTOCOLO DE EXPERIMENTAÇÃO PARA SELEÇÃO DE MÉ- TODOS MATEMÁTICOS	63
6.1	Preparação para o Protocolo	63
6.2	Controle de Versão do Protocolo	63
6.3	Projeto do Protocolo de Experimentação	63
6.3.1	Variáveis dependentes e independentes	64
6.3.2	Critério para seleção dos Métodos Matemáticos	64
6.3.3	Definições para simulação dos métodos de operação	64
6.4	Execução e Análise dos dados	65
6.4.1	Implementação dos Métodos Matemáticos	65
6.4.2	Simulação do método de Correlação Linear	65
6.4.3	Simulação do método de Mínimos Quadrados	66

6.4.4	Simulação do método de Fibonacci	68
6.4.5	Simulação do método de Estocástico	70
6.4.6	Simulação do método de Média Móvel	72
6.5	Limitações do experimento	73
6.6	Definição métodos de operação ferramenta MVC	73
7	PROPOSTA DA SOLUÇÃO	75
7.1	Backlog de Histórias de Usuário ferramenta InvestMVC	75
7.1.1	Histórias de Usuário	75
7.1.1.1	Histórias de Usuário Componente MultiAgente	75
7.1.1.2	Histórias de Usuário Componente Orientado a Objetos	77
7.1.1.3	Histórias de Usuário Componente Estruturado	79
7.1.1.4	Histórias de Usuário Componente Funcional	80
7.1.1.5	Histórias de Usuário Componente Lógico	80
7.1.1.6	História de Usuário Componente MQL	81
7.2	Arquitetura Orientada a Componentes	82
7.2.1	Componente Orientado a Objetos	82
7.2.2	Componente Cálculos Numéricos	83
7.2.2.1	Componente Funcional	84
7.2.2.2	Componente Estruturado	84
7.2.3	Componente Multiagentes	86
7.2.4	Componente Lógico	87
7.2.5	Componente MQL	87
7.2.6	Fluxo de atividades da ferramenta InvestMVC	88
8	RESULTADOS	91
8.0.7	Resultados do Componente Orientado a Objetos	91
8.0.8	Resultados do Componente Funcional	91
8.0.9	Resultados do Componente Estruturado	91
8.0.10	Resultados do Componente Multiagente	93
8.0.11	Resultados do Componente Lógico	94
8.0.12	Resultados do Componente MQL	94
9	CONSIDERAÇÕES FINAIS	95
	Referências	97
10	GLOSSÁRIO	103

APÊNDICES	105
APÊNDICE A – CRONOGRAMA INVESTMVC	107
APÊNDICE B – EXPERTS PARA EXPERIMENTO	115
APÊNDICE C – COMPONENTE MULTIAGENTE	131
APÊNDICE D – COMPONENTE FUNCIONAL	133
APÊNDICE E – COMPONENTE ESTRUTURADO	137
APÊNDICE F – COMPONENTE MQL	149
 ANEXOS	 153
ANEXO A – TEMPLATE DE PROTOCOLO EXPERIMENTAL . .	155

1 Introdução

Este capítulo aborda o contexto, o problema de pesquisa, a justificativa, os objetivos e como o trabalho está organizado.

1.1 Contextualização

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. É possível operar nesse mercado de forma manual ou automatizada (CVM, 2009).

No contexto de operações automatizadas, no Mercado de Moedas estão inseridos os produtos de software conhecidos como Experts, que processam métodos numéricos para gerar critérios de entrada e saída para comprar ou vender nesse mercado.

1.2 Problema de Pesquisa

Este TCC procurará responder a seguinte questão: é possível desenvolver uma ferramenta multiparadigma que substitua os Experts convencionais do Mercado de Moedas?

1.3 Justificativa

Operar no Mercado de Moedas de forma manual é muito arriscado e, portanto, não recomendado, uma vez que esse mercado é não previsível, o que pode provocar a perda do capital de um investidor em apenas alguns minutos. Em diversas situações, o mercado varia as cotações em apenas um minuto, sendo que a mesma variação pode ser feita durante horas. Para contornar esse problema, a plataforma MetaTrader¹ oferece as linguagens MQL4 (Paradigma Estruturado) e MQL5 (Paradigma Orientado a Objetos) para construir Experts que operem de forma automatizada.

A plataforma MetaTrader não oferece suporte de ferramentas de teste unitário para as linguagens MQL4 e MQL5. Após implementar um Expert, não é possível implementar testes de unidade 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.

¹ <<http://www.metatrader4.com/>>

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 da ferramenta anteriormente.

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

1.4 Objetivos

1.4.1 Objetivo Geral

Desenvolver uma ferramenta multiparadigma para operar de forma semi-automatizada no Mercado de Moedas.

1.4.2 Objetivos Específicos

Considerando o Mercado de Moedas e os Paradigmas de Programação Estruturado, Orientado a Objetos, Funcional, Lógico e Multiagentes, são objetivos específicos desse TCC:

1. Selecionar Métodos Matemáticos para serem implementados na ferramenta MVC através de um protocolo de experimentação;
2. Caracterizar as implementações dos códigos da ferramenta MVC através dos Paradigmas de Programação;
3. Comparar resultados financeiros obtidos pelo código no Paradigma Estruturado com códigos produzidos nos demais Paradigmas;

4. Realizar análise estática do código fonte dos produtos de software a partir de métricas de qualidade previamente definidas;
5. 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);
6. Desenvolver testes de integrações e funcionais para a ferramenta MVC.

1.5 Organização do Trabalho

No capítulo 2, é apresentado o Referencial Teórico quanto ao Contexto Financeiro, Métodos Numéricos e aos Paradigmas de Programação. No Contexto Financeiro são tratados os atributos atrelados ao Mercado de Moedas como alavancagem, suporte e resistência. Em Métodos Numéricos, é realizada uma descrição dos métodos de Fibonacci, Correlação de Pearson e Mínimos Quadrados. Em Paradigmas de Programação são descritos os paradigmas: Estruturado, Orientado a Objetos, Lógico, Funcional e Multiagentes. Em Testes de Software, são evidenciados quais os tipos de testes serão utilizados neste TCC e também quais linguagens terão testes. Por fim, em Qualidade de Software são externalizadas as métricas de qualidade de código fonte, critérios para interpretação das métricas e ferramentas de análise estática.

2 Referencial teórico

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

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

2.1 Contexto Financeiro

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

2.1.1 Mercado de Moedas

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

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

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

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

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

2.1.2 Alavancagem

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

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

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

No mercado FOREX, o investidor pode negociar contratos de taxas de câmbio e usar a Alavancagem para aumentar suas taxas de lucro. Se o investidor, por exemplo, realizar uma operação de compra apostando 0.01 por ponto e o mercado subir 1000 pontos, ele ganha 10 dólares (0.001×1000). 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.

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

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

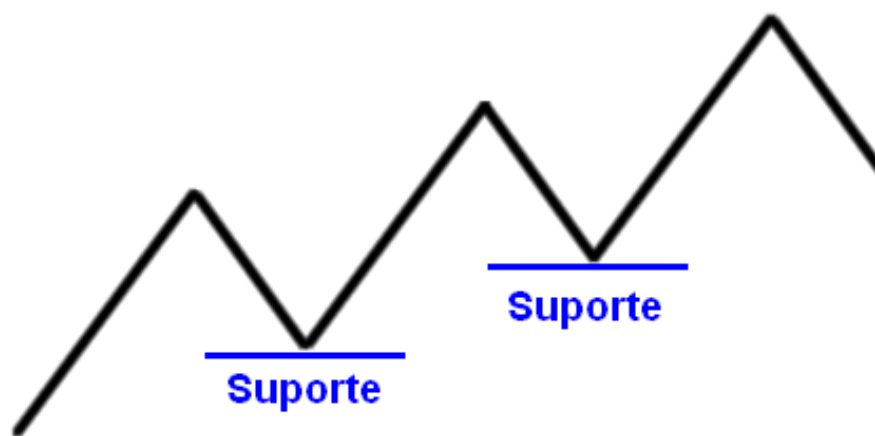


Figura 1 – Reta de Suporte.

Fonte: MATSURA (2006, pág. 22).

2.1.4 Resistência

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

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

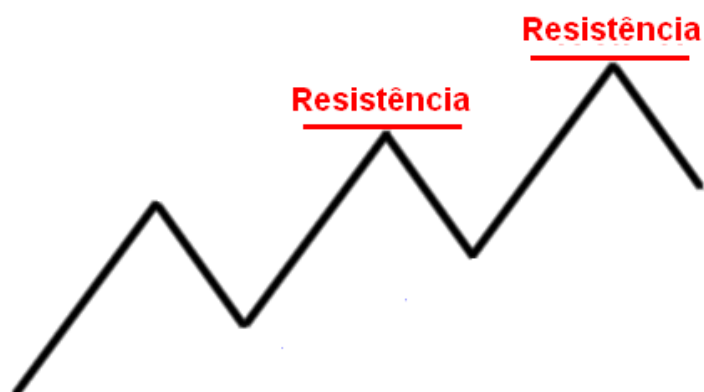


Figura 2 – Reta de Resistência.

Fonte: MATSURA (2006, pág. 23)

A Resistência indica os níveis das cotações em que os investidores acreditam que as mesmas vão descer. À medida que as cotações se deslocam para a zona de Resistência,

os investidores estão mais confiantes para vender (COLLINS et al., 2012).

2.2 Métodos Matemáticos

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

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

Este capítulo irá abordar sobre os Métodos Matemáticos de Mínimos Quadrados, Fibonacci e Correlação Linear de Pearson.

2.2.1 Método de Mínimos Quadrados

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

Usa-se o método de Mínimos Quadrados para determinar a melhor linha de ajuste que passa mais perto de todos os dados coletados, no intuito de obter a melhor linha de ajuste, de forma que minimize as distâncias entre cada ponto de consumo (DIAS, 1985, pág. 46).

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

O desvio vertical do ponto:

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

da reta:

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

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

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

à reta é portanto:

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

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

As estimativas pontuais de C_0 e C_1 , representadas por K_0 e K_1 e denominadas estimativa de Mínimos Quadrados, são aquelas que minimizam $f(B_0, B_1)$. Em suma, para qualquer B_0 e B_1 , K_0 e K_1 são tais que:

$$f(K_0, K_1) \leq f(B_0, B_1) \quad (2.6)$$

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

$$Y = K_0 + K_1 X \quad (2.7)$$

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

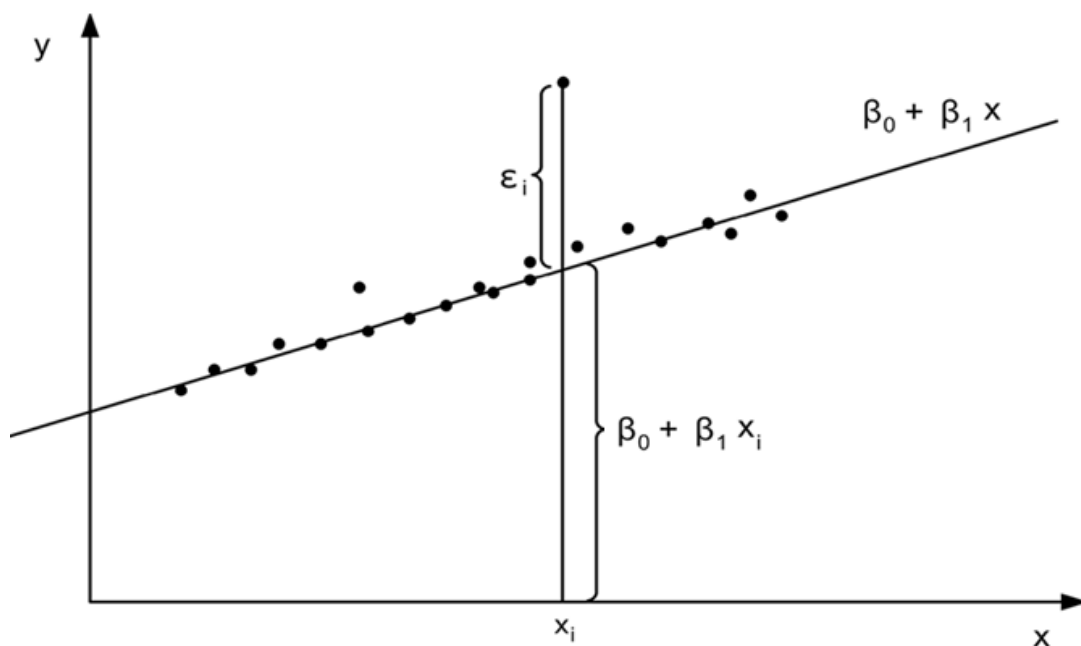


Figura 3 – Equação da reta Mínimos Quadrados.

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.

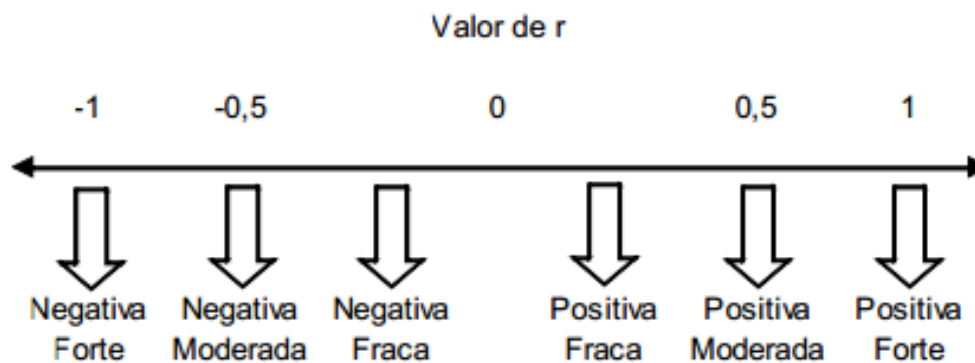


Figura 4 – Classificação da Correlação Linear.

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.

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.

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

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

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

Figura 5 – Determinação da Correlação Linear.

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

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

2.2.3 Método de Fibonacci

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

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

Segundo [Rocha \(2008, pág. 32\)](#), a fórmula de Fibonacci, sem uso da recorrência, é

dada de acordo com a figura 6. Essa fórmula também é conhecida como fórmula de Binet.

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

Figura 6 – Fórmula de Fibonacci sem uso de recorrência.

Fonte: Rocha (2008, pág. 32).

2.2.4 Estocástico

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

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

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

$$\%D = \sum_3 FechHoje - Mín_n \div \sum_3 Máxn - Mín_n \quad (2.9)$$

Sendo que:

- %D = Estocástico Suave
- %K = Estocástico Rápido
- Mín_n = menor preço durante os últimos "n" períodos
- Máxn = maior preço durante os últimos "n" períodos
- n = número de períodos
- \sum_3 = Somatório das 3 últimas barras

As linhas estocásticas seguem uma escala de 0 a 100 e indicam situações de compra ou venda. Conforme a imagem 7, quando as linhas estocásticas estão acima do nível 80 (linha pontilhada em vermelho), então o mercado está em situação de venda. Quando as linhas estocásticas estão abaixo de 20 (linha pontilhada em azul), então o mercado está numa situação de compra (INVESTFOREX, 2014).



Figura 7 – Indicador Estocástico.

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

2.2.5 Média Móvel

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

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

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

Figura 8 – Equação de Média Móvel.

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

tendência e, através do ângulo da sua inclinação, determinar a aceleração do movimento ([ROBOFOREX, 2013](#)).

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



Figura 9 – Indicador Média Móvel

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

2.3 Paradigmas de Programação

A palavra paradigma significa aquilo que pode ser utilizado como padrão, de forma que é um modelo a ser seguido ([FERREIRA, 2008](#)). Segundo [Normak \(2013\)](#) Kurt Normark (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 é uma linguagem de programação que 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 desse TCC.

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.

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

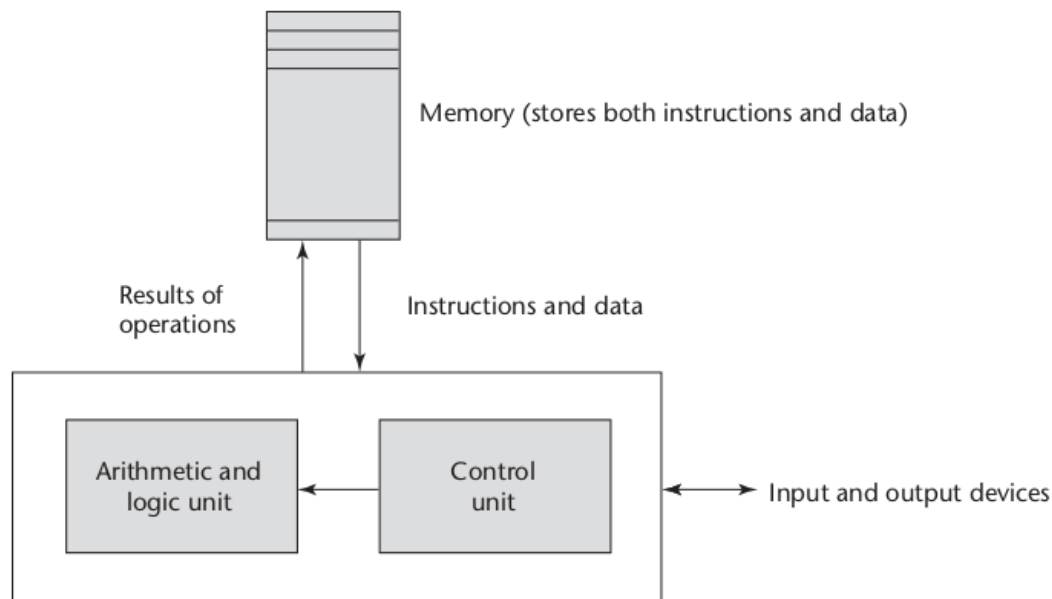


Figura 10 – Arquitetura de von Neumann.

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

controle, entrada/saídas, manipulação de exceções e erros, abstração procedural, expressões e atribuição, e suporte de biblioteca para estruturas de dados ([TUCKER; NOONAN, 2009](#), pág. 278-279).

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

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

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

2.3.2 Paradigma Orientado a Objetos

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

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

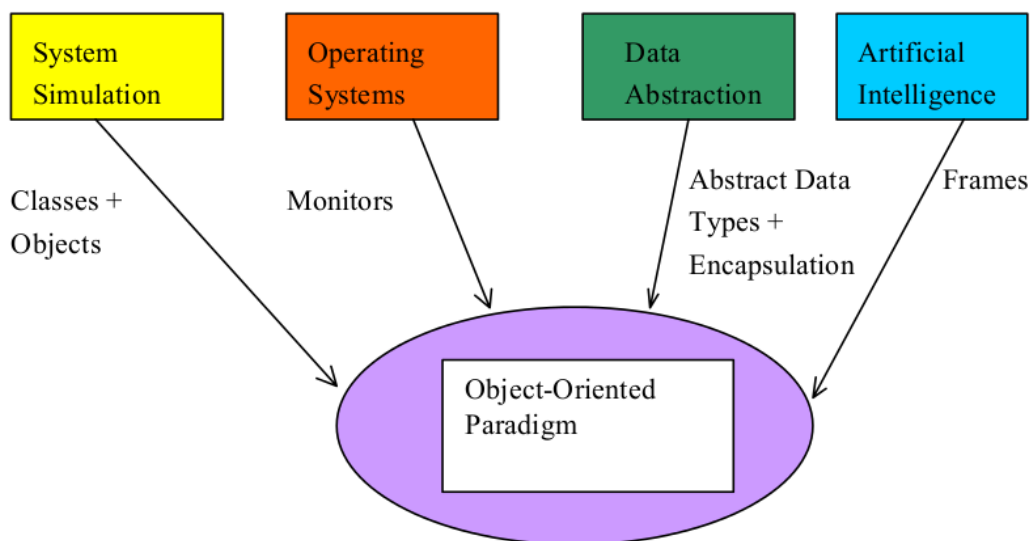


Figura 11 – A essência do Paradigma Orientado a Objetos.

Fonte: CAPRETZ (2003, pág. 1).

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

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

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

Uma nova classe (designada por subclasse derivada) pode ser derivada de outra classe (designada por superclasse) por um mecanismo chamado de herança. A classe derivada herda todas as características da classe base: a sua estrutura e comportamento (resposta a mensagens). Além disso, o mecanismo de herança é permitido mesmo sem acesso ao código-fonte da classe base. Herança dá a OO seu benefício chefe sobre outros paradigmas de programação - a reutilização de código relativamente fácil sem a necessidade de alterar o código fonte existente ([LEAVENS, 2014](#)).

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

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

Os conceitos de sobrecarga e sobrescrita são constantemente confundidos. A sobrecarga acontece em tempo de compilação, enquanto a sobrescrita ocorre em tempo de execução. A sobrecarga feita na mesma classe, enquanto são necessárias classes mães e suas filhas para o uso sobrescrita, métodos privados sobrecarregados, mas eles não podem ser sobrescritos. Isso significa que uma classe pode ter mais de um método privado

mesmo nome, mas uma classe filha não pode substituir os métodos privados sua classe mãe (SINGH, 2014a).

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

2.3.3 Paradigma Orientado a Agentes

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

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

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

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

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

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

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

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

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

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

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

WOOLDRIDGE, 1998).

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

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

A empresa Acronymics e a Alternative Energy Systems Consultants (AESC) realizaram uma pesquisa afim de criar agentes de softwares capazes de comprar e vender energia eletricidade participando do mercado eletrônico. Cada agente apresentavam um comportamento único e individual determinado pelo seu próprio modelo econômico, esta pesquisa mostrou que os agentes podem ser usados para implementar mercados e leilões eletrônicos, e que um agente pode adotar os objetivos e intenções de seu stakeholder (AGENTBUILDER, 2009a).

2.3.4 Programação Declarativa

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

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

2.3.4.1 Paradigma Funcional

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

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

A programação funcional exige que as funções sejam cidadãos de primeira classe, o que significa que elas são tratadas como quaisquer outros valores e podem ser passadas como argumentos para outras funções ou serem retornadas como um resultado de uma função. Sendo cidadãos de primeira classe também significa que é possível definir e manipular funções dentro de outras funções (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 iterativas não são possíveis, porque elas são controladas por variáveis. A repetição deve ser feita por meio de recursão, não por meio de laços (SEBESTA, 2012, pág. 555).

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

2.3.4.2 Paradigma Lógico

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

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

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

A programação lógica surgiu como um paradigma distinto nos anos 70. Ela é

diferente dos outros paradigmas porque requer que o programador declare os objetivos da computação, em vez dos algoritmos detalhados por meio dos quais esses objetivos podem ser alcançados (TUCKER; NOONAN, 2009, pág. 412).

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

2.4 Teste de Software

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

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

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

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

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



Figura 12 – Defeito x Erro x Falha.

Fonte: Neto (2005).

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

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

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

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

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

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

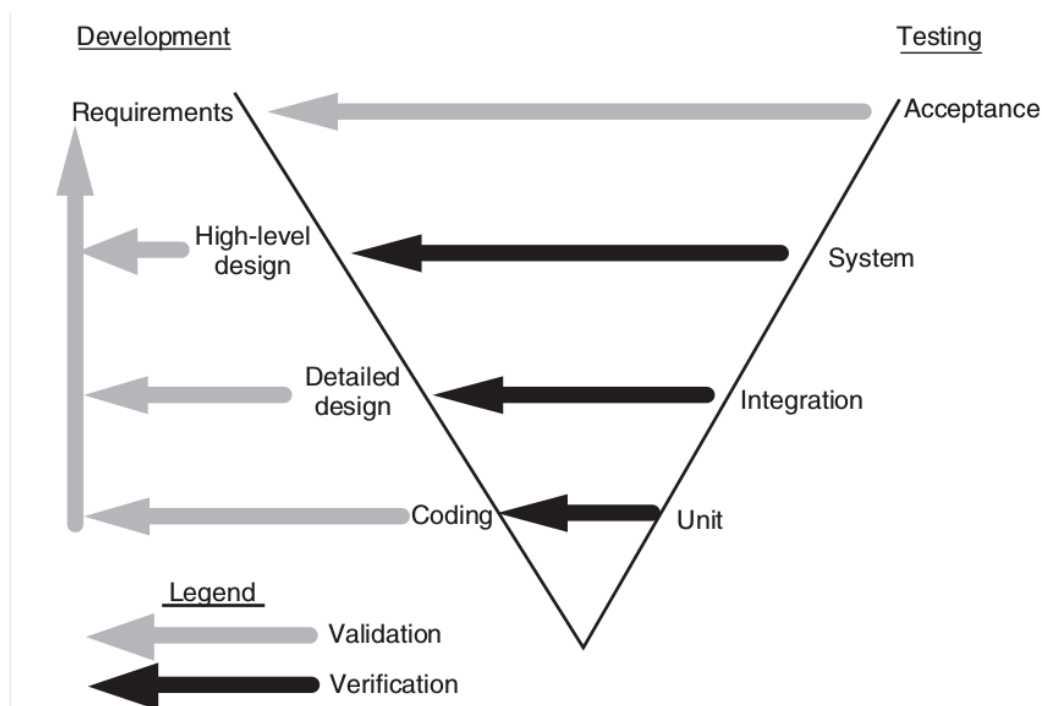


Figura 13 – Níveis de teste e seu desenvolvimento.

Fonte: Naik e TRIPATHY (2008, pág. 18).

teste de integração (NAIK; TRIPATHY, 2008, pág. 18).

Estes testes 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 formal realizado para permitir que um usuário, cliente ou outra entidade autorizada, para determinar se a aceita um sistema ou componente” (IEEE 610, 1990).

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

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

2.5 Qualidade de Software

“Qualidade é um termo que pode ter diferentes interpretações e para se estudar a qualidade de software de maneira efetiva é necessário, 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. Os fatores externos agregam bastante valor ao usuário final, mas um bom caminho para assegurar que os fatores externos terão qualidade é caprichar na construção dos fatores internos. Portanto, técnicas de qualidade interna, como análise de qualidade de código fonte, são meios para atingir a qualidade externa (BUENO; CAMPELO, 2011, pág. 4).

2.5.1 Métricas de Qualidade de Código Fonte

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

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

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

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

2.5.2 Análise Estática de Código Fonte

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

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

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

2.5.3 Ferramentas de Análise Estática

Ferramentas de análise estática de código fonte varrem o código fonte e detectam possíveis anomalias. Também pode ser usados como parte de um processo de inspeção

	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).

(SOMMERVILLE, 2011, pág. 345).

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

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

3 Metodologia de pesquisa

Segundo RODRIGUES (2007, pág. 2), metodologia de 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 metodologia de 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 a classificar uma pesquisa com base em seus objetivos em três grandes grupos: exploratórias, descritivas e explicativas (GIL, 2008, pág. 41)

“Pesquisas exploratórias têm como preocupação central identificar os fatores que determinam ou que contribuem para a ocorrência dos fenômenos. Esse é o tipo de pesquisa que mais aprofunda o conhecimento da realidade, porque explica a razão, o porquê das coisas. Por isso mesmo, é o tipo mais complexo e delicado, já que o risco de cometer erros aumenta consideravelmente” (GIL, 2008, pág. 43).

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

Segundo (BANDEIRA, 2012, pág. 2), a pesquisa experimental é um tipo de pesquisa quantitativa e visa identificar relações causais entre duas ou mais variáveis, através do método experimental. Esse método implica em três procedimentos básicos: variar a causa, controlar variáveis interferentes e medir a causa.

Considerando os objetivos de estudo desse TCC, será incorporada a pesquisa exploratória com o intuito de evidenciar os possíveis fenômenos que se repetem no Mercado de Moedas. Em relação aos procedimentos técnicos, será realizada a pesquisa quantitativa experimental para evidenciar como será solucionado o problema de pesquisa.

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 estão organizadas as atividades desse TCC.

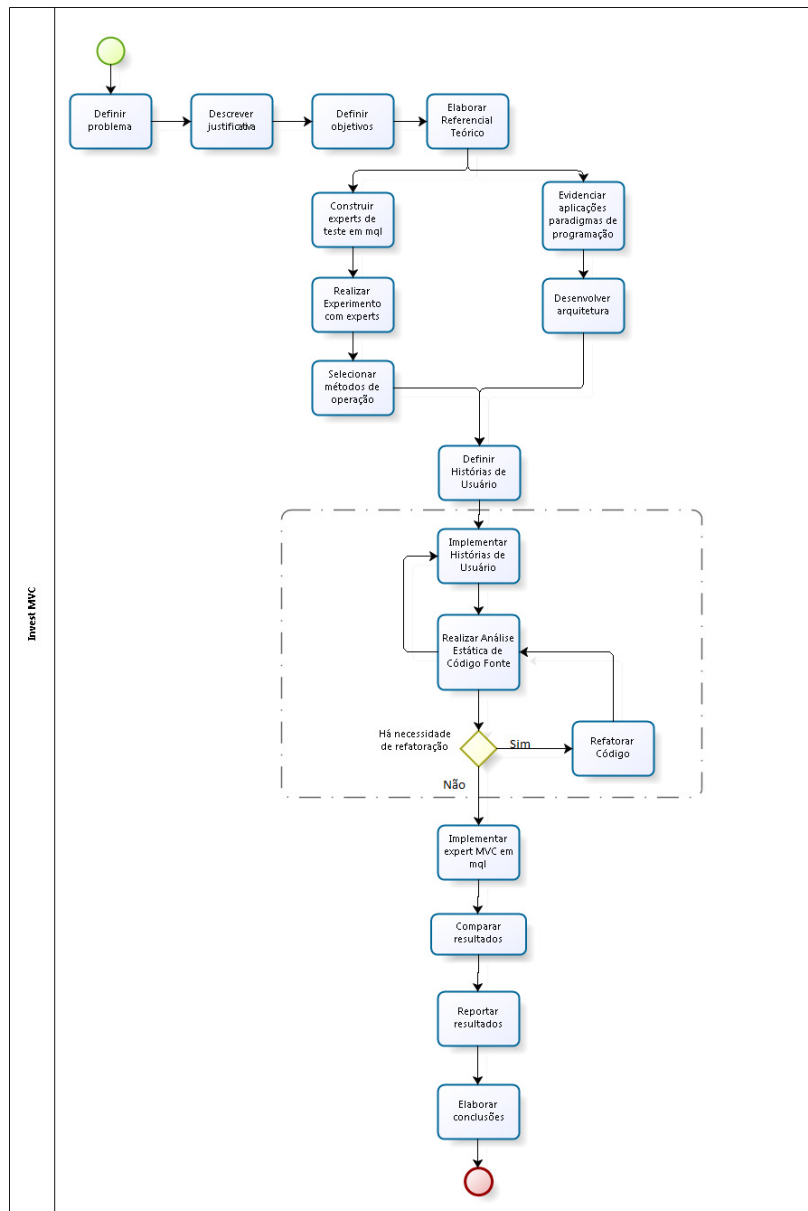


Figura 15 – Atividades da Pesquisa.

3.2.1 Descrição dos Objetivos das Atividades de Pesquisa

A tabela 1 evidencia cada atividade da metodologia de pesquisa e o objetivo de cada atividade.

Tabela 1 – Atividades e objetivos metodologia de pesquisa

Atividade	Objetivo
Definir problema	Definir o problema de pesquisa do TCC.
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 experts em mql	Implementar expert Fibonacci.mql, Minimos-Quadrados.mql, CorrelacaoLinear.mql, Estocastico.mql, MediaMovel.mql.
Realizar experimento com experts	Definir critérios de entrada e saída para cada expert e realizar a simulação de cada expert 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 para paradigmas de programação	Evidenciar aplicações dos paradigmas: funcional, lógico, estruturado e multiagente.
Desenvolver arquitetura	Desenvolver a arquitetura da ferramenta MVC 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 código fonte da ferramenta MVC.
Implementar expert em mql	Implementar toda a lógica da ferramenta MVC em linguagem mql.
Comparar resultados monetários expert mql e ferramenta mvc	Comparar os resultados monetários do expert em mql com a ferramenta MVC durante um tempo a se determinado de operação.
Reportar resultados	Registrar os resultados da comparação do desempenho da ferramenta MVC e do expert em mql.
Elaborar conclusões	Desenvolver as conclusões do TCC.

3.3 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 em 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 sustentam qualquer implementação de controle de processos empíricos: transparência, inspeção e adaptação ([SCHWABER; SUTHERLAND, 2013](#), pág. 4).

Nesse TCC, vertentes defendidas pelo Scrum serão adaptadas e incorporadas a metodologia de pesquisa. Os produtos de trabalho serão alocados e desenvolvidos em Sprints (intervalo de tempo de 1 a 4 semanas). Durante a execução de uma Sprint, serão usadas adaptações que forem necessárias para que os artefatos e produtos de software tenham a maior qualidade possível.

4 Cronograma

A tabela 2 mostra quais são as atividades em cada Sprint e a duração da mesma. O cronograma detalhado encontra-se no apêndice A - Cronograma InvestMVC.

Tabela 2 – Cronograma simplificado

Sprint	Atividades	Data de início	Data de finalização
Sprint 1	<ol style="list-style-type: none"> 1. Construir introdução 2. Implementar Métodos Numéricos 	10/08/2014	31/08/2014
Sprint 2	<ol style="list-style-type: none"> 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
Sprint 3	<ol 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	<ol style="list-style-type: none"> 1. Realizar Experimentos Métodos de Operação 2. Revisar Referencial Teórico Contexto Financeiro 3. Desenvolver Experts em MQL4 	01/10/2014	15/10/2014
Sprint 5	<ol style="list-style-type: none"> 1. Descrever Metodologia de Pesquisa 2. Realizar Experimentos Métodos de Operação 3. Evidenciar Aplicações Paradigmas de Programação 	16/10/2014	31/10/2014
Sprint 6	<ol style="list-style-type: none"> 1. Definir Backlog de Histórias de Usuário 2. Verificar Qualidade de código fonte 	01/11/2014	10/11/2014

5 Suporte Tecnológico

5.1 Ferramentas para teste unitário e teste de integração

XUnit ¹ 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 PlUnit que respectivamente auxiliam na criação de testes em linguagem C, Java, Haskell e Prolog.

5.1.1 CUnit

Cunit ² 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.

5.1.2 JUnit

JUnit ³ 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.

5.1.3 HUnit

HUnit ⁴ 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/>>

5.1.4 PIUnit

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

5.2 Ferramenta para teste funcional: Cucumber

Cucumber ⁷ 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.

5.3 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.

5.3.1 Eclemma

Eclemma ⁸ 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.

5.3.2 HPC

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

5.4 Ferramenta de Análise Estática de Código Fonte

5.4.1 Analizo

Analizo ¹⁰ 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

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

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

⁷ <<http://cukes.info/>>

⁸ <<http://www.eclemma.org/>>

⁹ <https://www.haskell.org/haskellwiki/Haskell_program_coverage#Hpc_tools>

¹⁰ <<http://www.analizo.org/>>

métricas e possui licença GPL3.

5.4.2 Sonar

Sonar ¹¹ 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.

5.5 Ferramentas de Mercado de Moedas

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

5.5.1 MetaTrade

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

5.5.2 MetaEditor

MetaEditor ¹⁵ 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.

5.5.3 Alpari-UK

Alpari-UK ¹⁶ é uma corretora com sede oficial na Inglaterra. Possui tecnologia de negociação que inclui a plataforma MetraTrader. 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.

5.5.4 FXDD

FXDD ¹⁷ é uma corretora com sede oficial nos Estados Unidos e possui as mesmas características tecnológicas que a Alpari-UK. Inclui as tecnologias da plataforma

¹¹ <<http://www.sonarqube.org/>>

¹² <<http://www.metaquotes.net/>>

¹³ <<http://www.mql4.com/>>

¹⁴ <<http://www.mql5.com/>>

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

¹⁶ <<http://www.alpari.co.uk/>>

¹⁷ <<http://www.fxdd.com/>>

MetaTrader e intermedia as negociações através da cobrança de uma corretagem para o investidor.

6 Protocolo de Experimentação para seleção de Métodos Matemáticos

Esta seção descreve como foi percorrido o Protocolo de Experimentação para selecionar os Métodos Matemáticos que serão implementados na ferramenta InvestMVC. O Protocolo de Experimentação utilizado é uma adaptação do protocolo presente no Anexo A - Template de Protocolo Experimental.

Ao final do capítulo, o leitor será capaz de entender porque foi escolhido os métodos de Correlação de Pearson, Fibonacci e Mínimos Quadrados para serem incorporados na ferramenta InvestMVC.

6.1 Preparação para o Protocolo

A seguinte questão de pesquisa foi definida na capítulo 1: é possível desenvolver uma ferramenta multiparadigma que substitua os Experts convencionais do Mercado de Moedas?

É necessário que a ferramenta InvestMVC tenha métodos relevantes para operar no Mercado de Moedas. Logo, é deve-se selecionar os métodos para serem implementados na ferramenta através de critérios bem definidos. A seleção dos métodos, critérios para realização dos experimentos, bem como resultados serão apresentados nas seções posteriores.

6.2 Controle de Versão do Protocolo

O controle de versão do protocolo desse experimento encontra-se no github¹ da ferramenta InvestMVC.

6.3 Projeto do Protocolo de Experimentação

Nesta seção é definido as variáveis dependentes e independentes, critérios para seleção dos Métodos Matemáticos e definições para simulações dos métodos.

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

6.3.1 Variáveis dependentes e independentes

Variáveis independentes são aquelas que são manipuladas e as que variáveis dependentes são apenas medidas ou registradas. Os termos variável dependente e independente aplicam-se principalmente à pesquisa experimental, onde algumas variáveis são manipuladas (HOPPEN, 2010, pág. 13).

Para se selecionar os métodos de operação no Mercado de Moedas da ferramenta InvestMVC, foi definido as seguintes variáveis independentes e seus respectivos valores a serem manipulados:

- 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.

Também foi definido as variáveis dependentes:

- 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.

6.3.2 Critério para seleção dos Métodos Matemáticos

Os Métodos Matemáticos implementados em linguagem MQL4 que obter lucro de 10% do capital inicial serão aprovados no experimento.

6.3.3 Definições para simulação dos métodos de operação

Foi utilizado o simulador da plataforma Metatrader ² para realizar a simulação e, posteriormente, analisar os resultados e definir os métodos a serem implementados na ferramenta InvestMVC.

Através do simulador da plataforma Metatrader4, obteve-se o desempenho do expert CorrelacaoPearson.mql, MediaMovet.mql, MinimosQuadrados.mql, Estocastico.mql e Fibonacci.mql durante o período de agosto de 2012-2014.

² <<http://www.metatrader4.com/>>

6.4 Execução e Análise dos dados

Nesta seção é evidenciado os resultados das simulações dos Experts em linguagem MQL4 (produto da implementação dos Métodos Matemáticos) através de relatórios e gráficos.

6.4.1 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 B - Experts para Experimento.

6.4.2 Simulação do método de Correlação Linear

O expert CorrelacaoPearson.mql obteve o percentual de negociações com lucros de 56.86% no período agosto 2012-2013. Nesse período, o expert teve um lucro de 1981.60 USD. No período de agosto 2013-2014, 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		
Deposito 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 (35.71%)
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 agosto 2012-2013 do expert CorrelacaoPearson.mql

Fonte: Autores

Foram gerados os gráficos de simulação 2012-2013 e 2013-2014. É possível perceber que o método de Correlação de Pearson, perde dinheiro em alguns períodos, mas os ganhos são superiores as perdas.

Barras em teste	7244	Ticks modelados	10799187
Mismatched charts errors	8		
Deposito 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 agosto 2013-2014 do expert CorrelacaoPearson.mql

Fonte: Autores

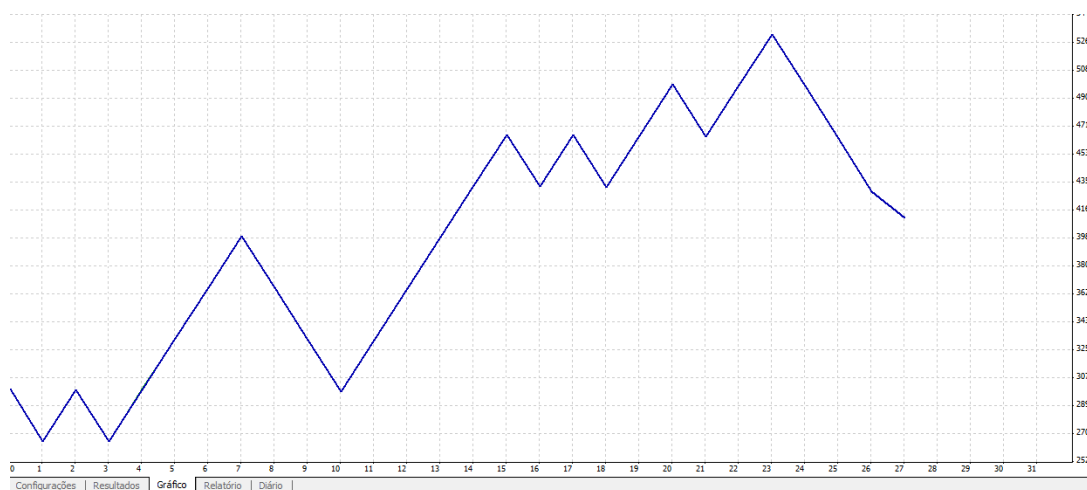


Figura 18 – Gráfico gerado pela simulação do expert CorrelacaoPearson.mql no período agosto 2012-2013

Fonte: Autores

6.4.3 Simulação do método de Mínimos Quadrados

O expert MinimosQuadrados.mql obteve o percentual de negociações com lucros de 77.88% no período agosto 2012-2013 e o lucro nesse período foi de 1341.88 USD. No período de agosto 2013-2014, 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.

O expert MinimosQuadrados.mql, teve altos e baixos nas simulações durante os dois anos (2012-2013 e 2013-2014). Mas, no desempenho geral, conforme é evidenciado nos gráficos, o expert teve um desempenho satisfatório.

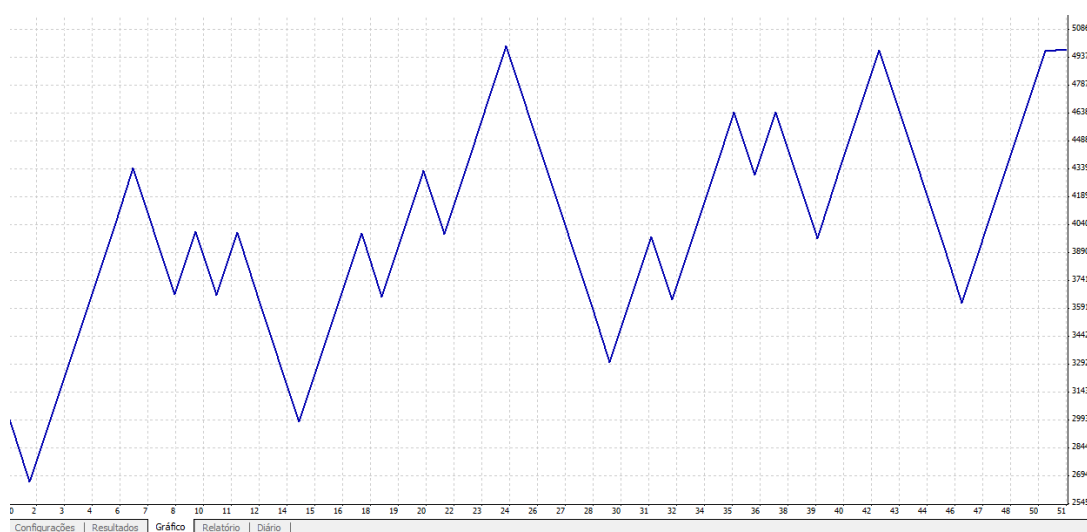


Figura 19 – Gráfico gerado pela simulação do expert CorrelacaoPearson.mql no período agosto 2013-2014

Fonte: Autores

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 agosto 2012-2013 do expert

Fonte: Autores

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 agosto 2012-2013 do expert

Fonte: Autores

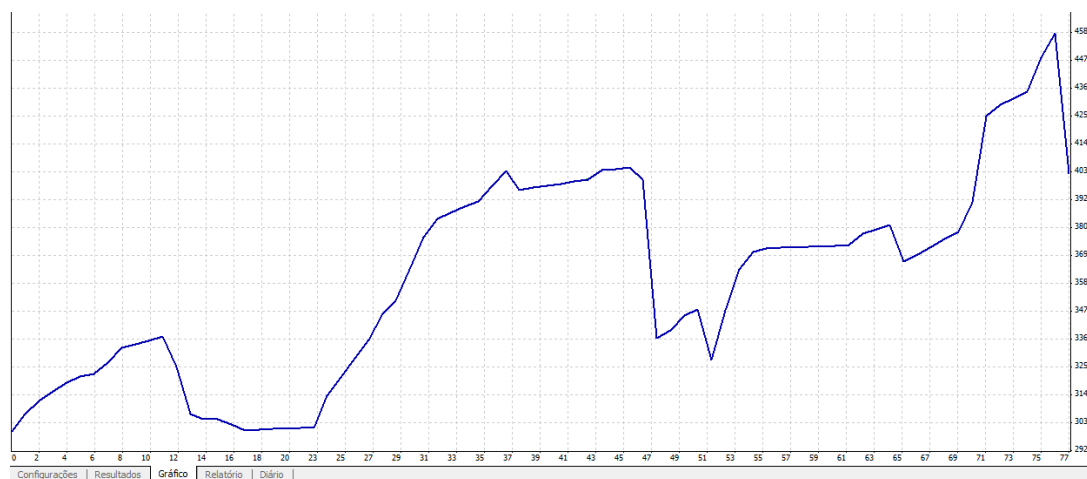


Figura 22 – Relatório de simulação no período agosto 2012-2013 do expert

Fonte: Autores

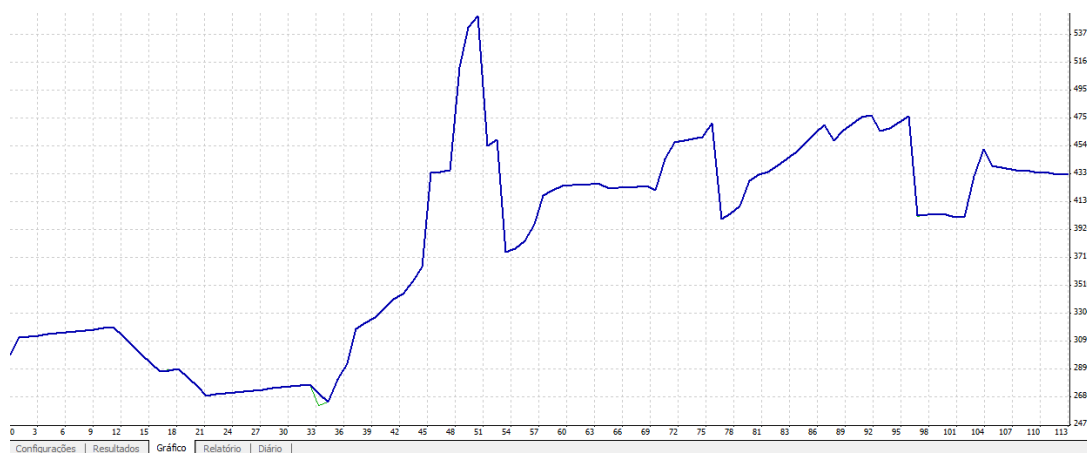


Figura 23 – Relatório de simulação no período agosto 2013-2014 do expert

Fonte: Autores

6.4.4 Simulação do método de Fibonacci

O expert Fibonacci.mql obteve o percentual de negociações com lucros de 72.73% no período agosto 2012-2013 e o lucro nesse período foi de 341.20 USD. No período de agosto 2013-2014, 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 de no período de agosto 2013-2014, o expert ter negociado mais vezes (25 contra 11).

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

É possível visualizar nos gráficos das simulações, as perdas de capital que o expert Fibonacci.mql gerou.

Barras em teste	7244	Ticks modelados	10799187
Mismatched charts errors	8		
Deposito 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	Negociações 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 agosto 2012-2013 do expert

Fonte: Autores

Barras em teste	7244	Ticks modelados	17462779
Mismatched charts errors	0		
Deposito 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	Negociações 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

Fonte: Autores

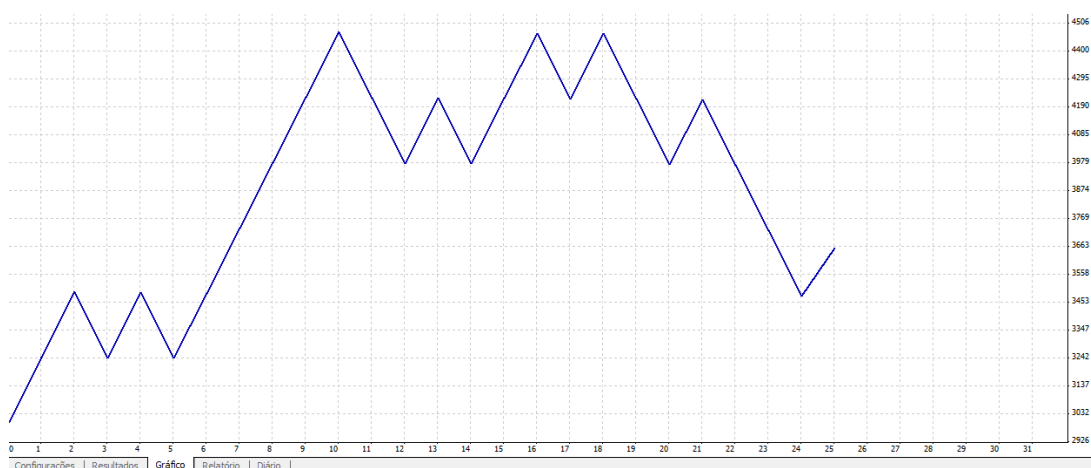


Figura 26 – Gráfico gerado pela simulação do expert Fibonacci.mql no período agosto 2012-2013

Fonte: Autores

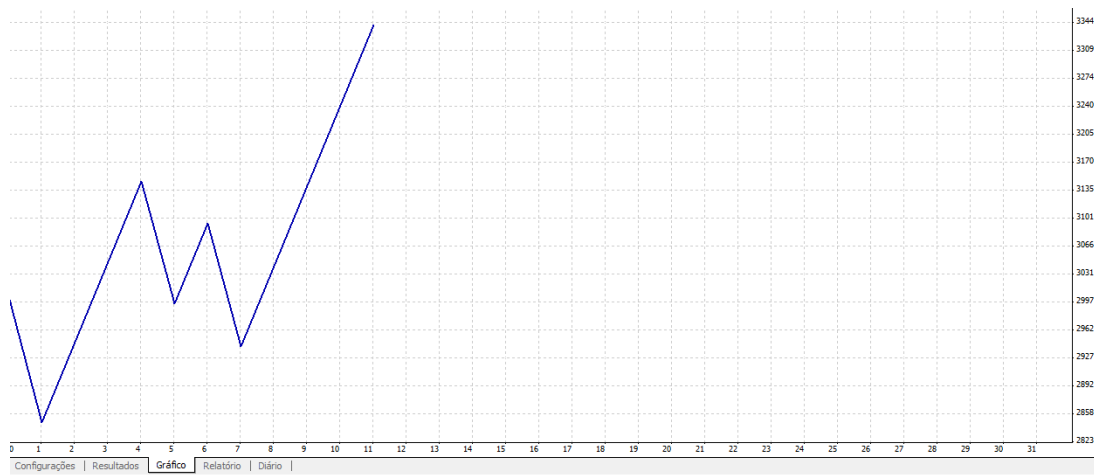


Figura 27 – Gráfico gerado pela simulação do expert Fibonacci.mql no período agosto 2013-2014

Fonte: Autores

6.4.5 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 2013-2014, 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		
Deposito 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	Negociações 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 agosto 2012-2013 do expert Estocastico.mql

Fonte: Autores

É possível visualizar nos gráficos das simulações, as perdas de capital que o expert Estocastico.mql gerou.

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	Negociações com lucro	126.40
	Média	Negociações 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 agosto 2013-2014 do expert Estocastico.mql

Fonte: Autores

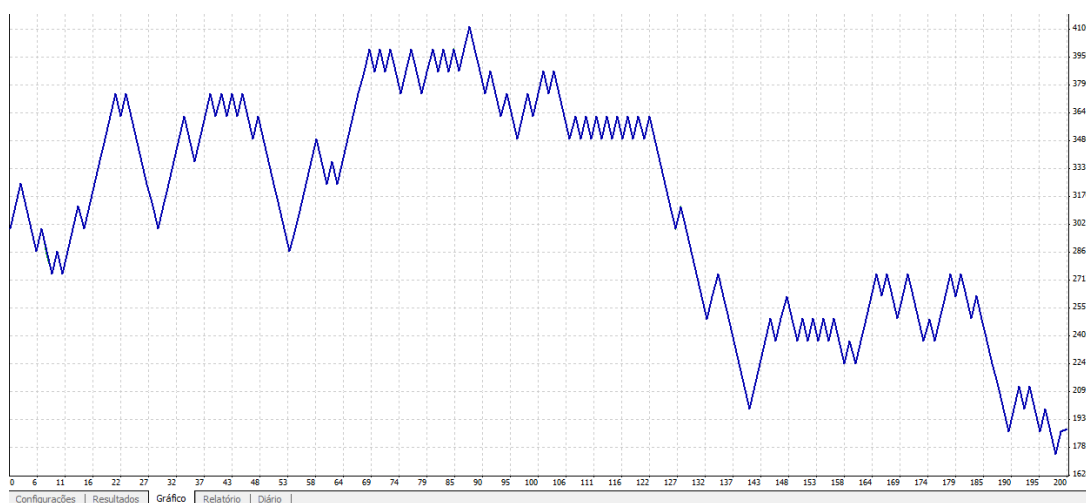


Figura 30 – Gráfico gerado pela simulação do expert Estocastico.mql no período agosto 2012-2013

Fonte: Autores

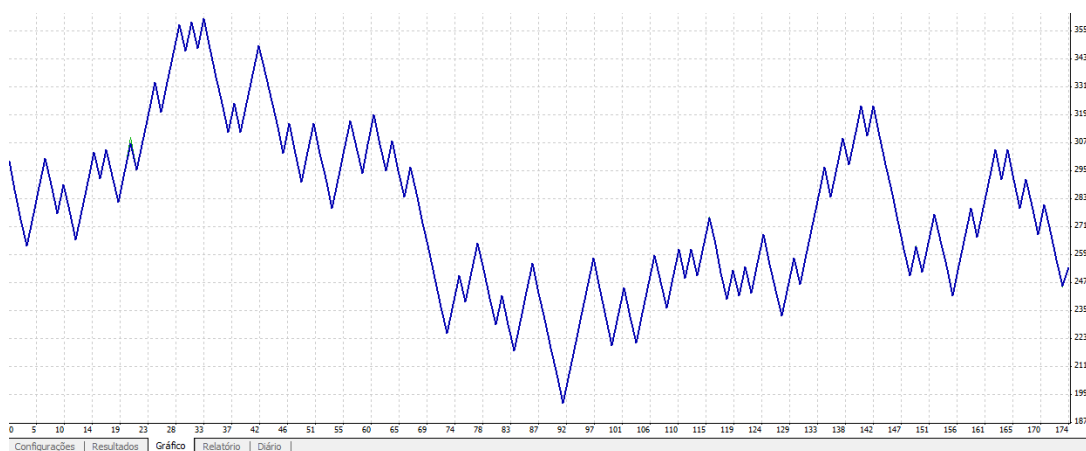


Figura 31 – Gráfico gerado pela simulação do expert Estocastico.mql no período agosto 2013-2014

Fonte: Autores

6.4.6 Simulação do método de Média Móvel

O expert MediaMovel.mql obteve o percentual de negociações com lucros de 43.55% no período agosto 2012-2013. 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 2013-2014, o percentual de negociações com lucros foi de 48.48% e o percentual de negociação com prejuízos foi de 51.82%. Foi obtido o 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		
Deposito Inicial	3000.00		
Lucro líquido total	-2987.00	Lucro Bruto	10124.18
Fator de lucro	0.77	Compensação esperada	-16.06
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** | Diário

Figura 32 – Relatório de simulação no período agosto 2012-2013 do expert MediaMovel.mql

Fonte: Autores

Barras em teste	7244	Ticks modelados	10799187
Mismatched charts errors	8		
Deposito 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	Negociações 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** | Diário

Figura 33 – Relatório de simulação no período agosto 2013-2014 do expert MediaMovel.mql

Fonte: Autores

É possível visualizar nos gráficos das simulações, as perdas de capital que o expert MediaMovel.mql gerou.

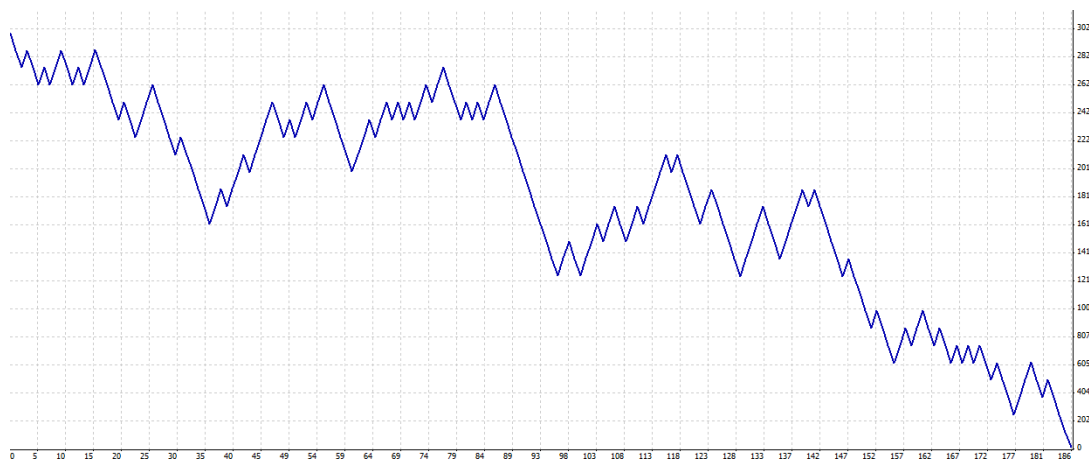


Figura 34 – Gráfico gerado pela simulação do expert MediaMovel.mql no período agosto 2012-2013

Fonte: Autores

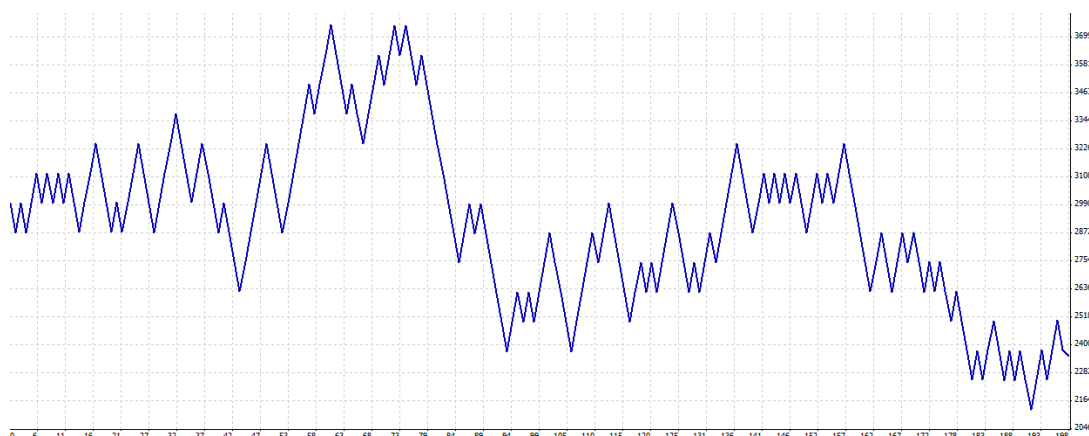


Figura 35 – Gráfico gerado pela simulação do expert MediaMovel.mql no período agosto 2013-2014

Fonte: Autores

6.5 Limitações do experimento

A linguagem MQL4 não possui nenhum suporte para teste unitário. Portanto, os experts programados para este experimento não possuem testes automatizados.

O simulador MetaTrader possui código fonte fechado. Portanto, não possível realizar nenhum tipo de adaptação do simulador através do código fonte para o experimento.

6.6 Definição métodos de operação ferramenta MVC

Os Métodos Matemáticos de Correlação Linear, Fibonacci e Mínimos Quadrados tiveram êxito nos dois anos de simulação (agosto 2012 a agosto de 2014). 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 a serem implementados na ferramenta MVC.

7 Proposta da Solução

7.1 Backlog de Histórias de Usuário ferramenta InvestMVC

A tabela 3 evidencia o Backlog da ferramenta InvestMVC e as respectivas pontuações de cada História de Usuário.

Tabela 3 – Atividades e objetivos metodologia de pesquisa

Histórias de Usuário	Pontuação
US1 - Agente Correlação Linear	3
US2 - Agente Fibonacci	3
US3 - Agente Mínimos Quadrados	3
US4 - Agente Tendência	2
US5 - Agente Gestor/Consultor	2
US6 - Criar conta de usuário	2
US7 - Acompanhar retorno financeiro	5
US8 - Criar Experts	2
US9 - Editar Experts	2
US10 - Excluir Experts	2
US11 - Ativar Expert	1
US12 - Desativar Expert	2
US13 - Método Correlação Linear em linguagem C	2
US14 - Método Fibonacci em linguagem C	2
US15 - Método Mínimos Quadrados em linguagem C	2
US16 - Método Correlação Linear em linguagem Haskell	2
US17 - Método Fibonacci em linguagem Haskell	2
US18 - Método Mínimos Quadrados em linguagem Haskell	2
US19 - Inserir na Base de Conhecimento	2
US20 - Retirar na Base de Conhecimento	2
US21 - Calcular Critério de Entrada	3
US22 - Realizar Operação no MetaTrader	1
Total de pontos	49

7.1.1 Histórias de Usuário

7.1.1.1 Histórias de Usuário Componente MultiAgente

Nas seções a seguir, são apresentadas as Histórias de Usuário com seus critérios de aceitação referentes ao componente Multiagente.

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.

7.1.1.2 Histórias de Usuário Componente Orientado a Objetos

Nas seções a seguir, são apresentadas as Histórias de Usuário com seus critérios de aceitação referentes ao componente Orientado a Objetos.

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.

7.1.1.3 Histórias de Usuário Componente Estruturado

Nas seções a seguir, são apresentadas as Histórias de Usuário com seus critérios de aceitação referentes ao componente Estruturado.

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.

7.1.1.4 Histórias de Usuário Componente Funcional

Nas seções a seguir, são apresentadas as Histórias de Usuário com seus critérios de aceitação referentes ao componente Funcional.

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.

7.1.1.5 Histórias de Usuário Componente Lógico

Nas seções a seguir, são apresentadas as Histórias de Usuário com seus critérios de aceitação referentes ao componente Lógico.

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.

7.1.1.6 História de Usuário Componente MQL

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.

7.2 Arquitetura Orientada a Componentes

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

Por usar vários paradigmas, a ferramenta InvestMVC terá vários componentes e cada componente terá seu paradigma de programação e responsabilidade bem definida, como mostrado na figura 36.

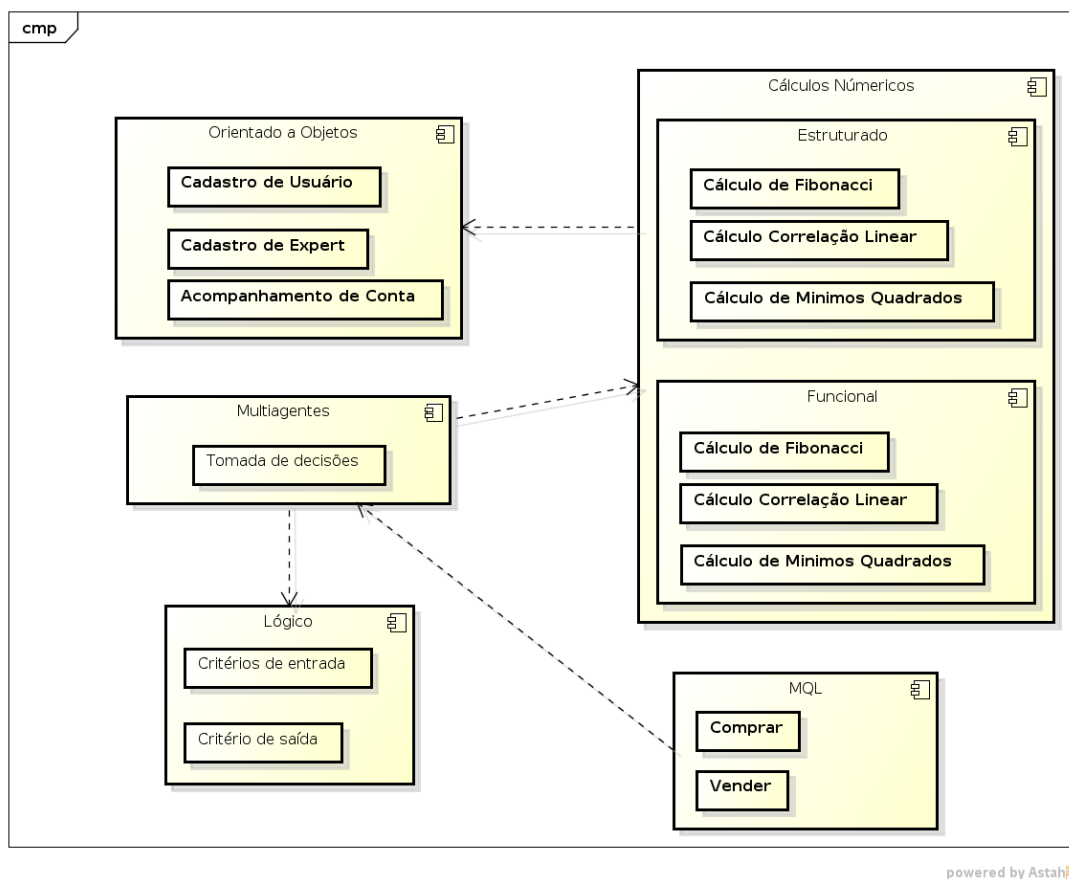


Figura 36 – Diagrama de Sequência InvestMVC

Fonte:Autores

7.2.1 Componente Orientado a Objetos

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

juntamente com o framework grails, fornece a criação de um projeto com uma arquitetura MVC definida, como demonstra a figura 37.

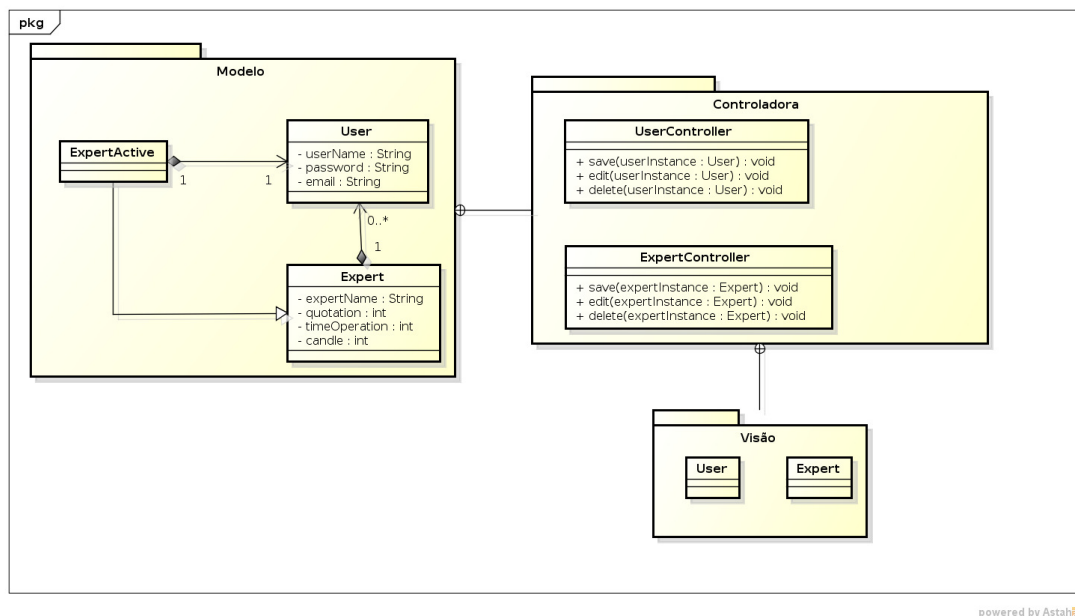


Figura 37 – Diagrama de Classe InvestMVC componente Orientado a Objetos

Fonte: Autores

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

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

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

7.2.2 Componente Cálculos Numéricos

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

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

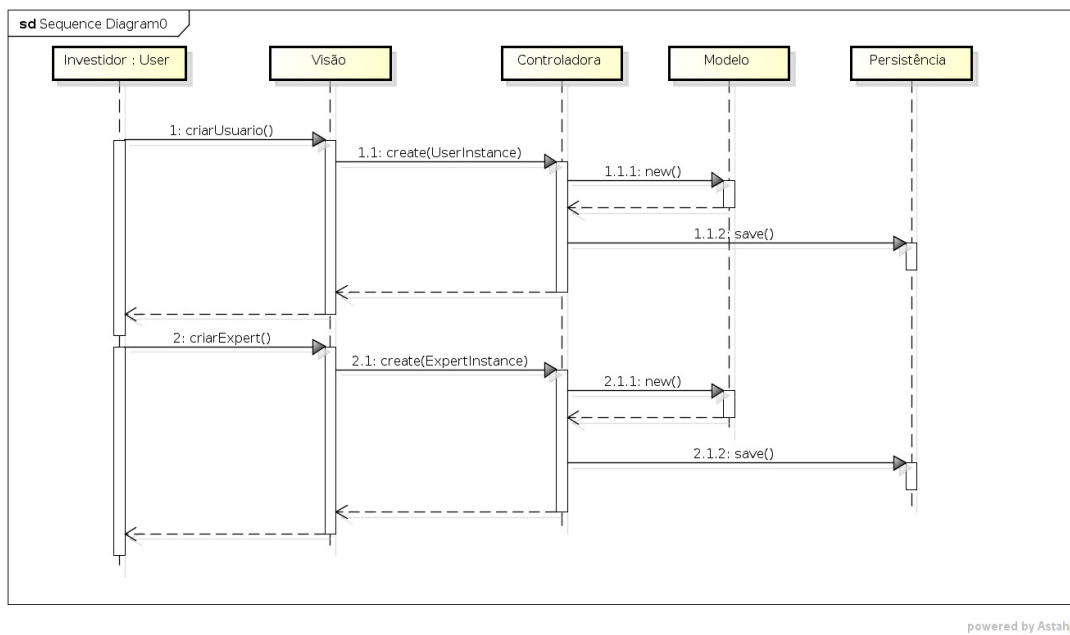


Figura 38 – Diagrama de sequência InvestMVC componente Orientado a Objetos

Fonte: Autores

7.2.2.1 Componente Funcional

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

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

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

O componente Funcional espera uma solicitação de cálculo do componente Multiagente, logo após a solicitação o componente busca na persistência (um arquivo) as cotações do mercado, com estas cotações o componente é capaz de realizar o cálculo do Método Numérico que é esperado pelo componente Multiagente, como é mostrado na figura 40.

7.2.2.2 Componente Estruturado

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

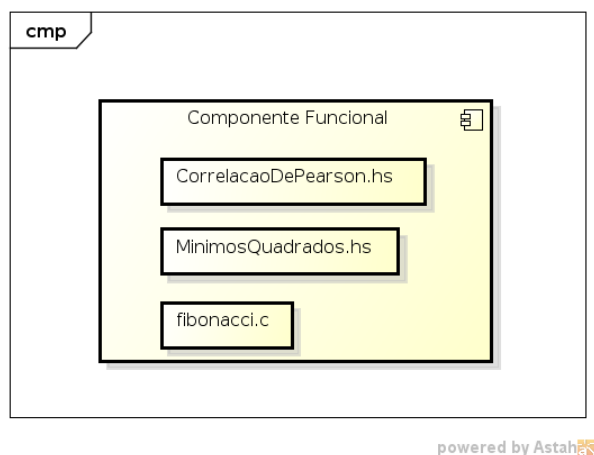


Figura 39 – Componente Funcional InvestMVC

Fonte: Autores

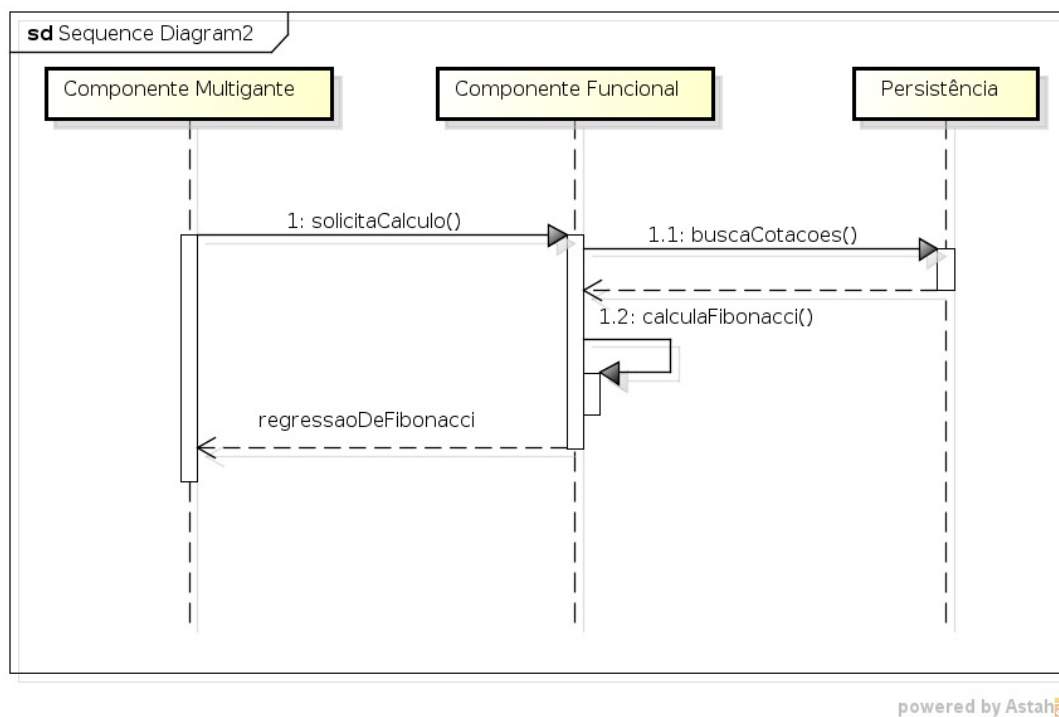


Figura 40 – Diagrama de Sequência do Componente Funcional InvestMVC

Fonte: Autores

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

A arquitetura e sequência do fluxo de dados do Componente Estruturado segue a mesma lógica do Componente Funcional, como ilustrado na figura 41.

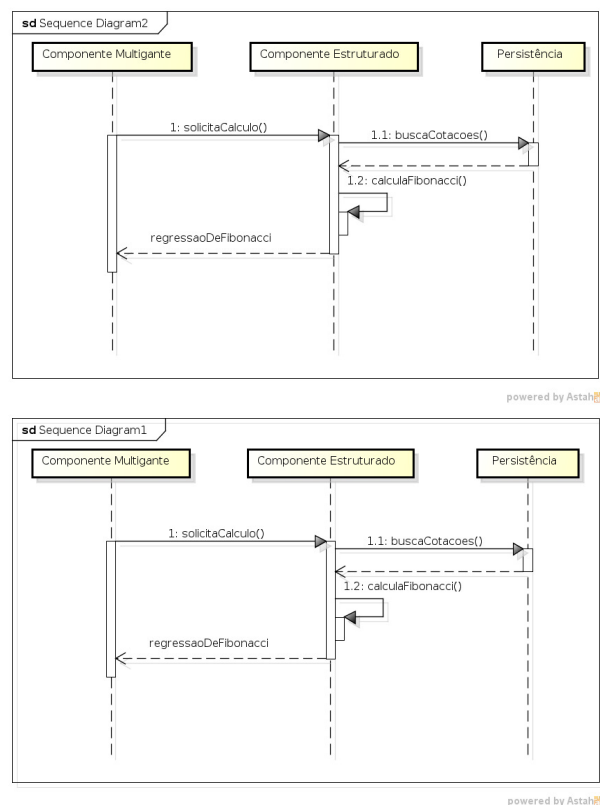


Figura 41 – Diagramas de Componentes e de Sequência do Componente Estruturado InvestMVC.

Fonte: Autores.

7.2.3 Componente Multiagentes

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

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

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

A arquitetura do Componente Multiagente está modularizado por pacotes: o pacote comportamentos será formado por comportamentos que serão usados pelos Agentes de Software, o pacote metodosNumericos será formado por Agentes que acerão o componente

Cálculos Numéricos, o pacote investidores será composto por Agentes que vão interagir com o Componente MQL e o pacote execucao iniciará a execução do SMA.

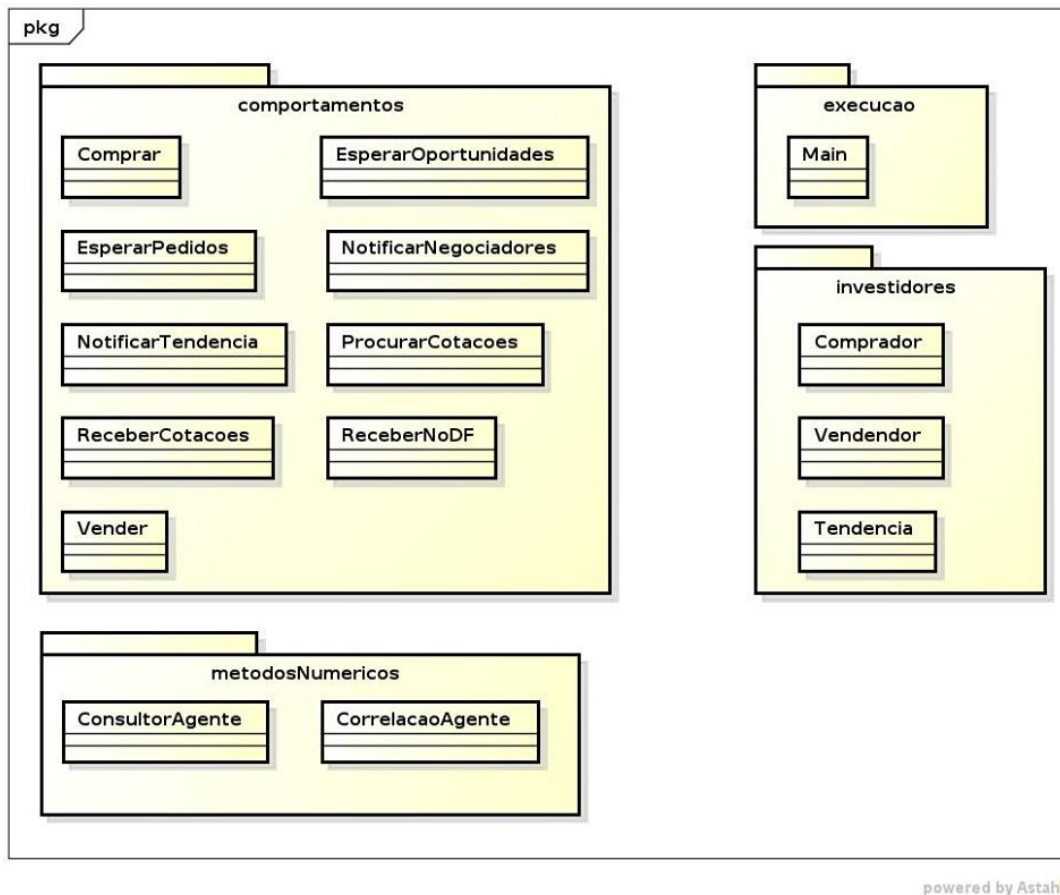


Figura 42 – Diagrama de Classe do Componente Multiagente InvestMVC

Fonte: Autores

7.2.4 Componente Lógico

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

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

A interação do Componente Lógico com o Componente Multiagentes é evidenciada na figura 43.

7.2.5 Componente MQL

O componente MQL será responsável por receber a resposta do Componente Multiagentes para realizar uma compra ou venda. Também será recebido outros atributos

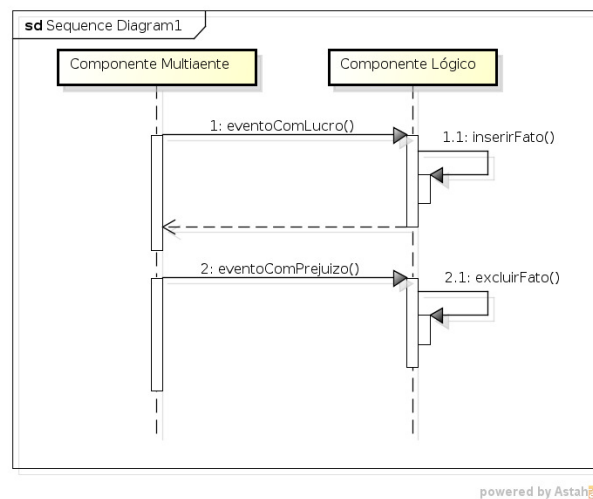


Figura 43 – Diagrama de Sequência do Componente Lógico InvestMVC.

relacionados a compra ou venda, como alavancagem, stop loss e take profit.

A interação do Componente MQL com o Componente Multiagentes é evidenciada na figura 44.

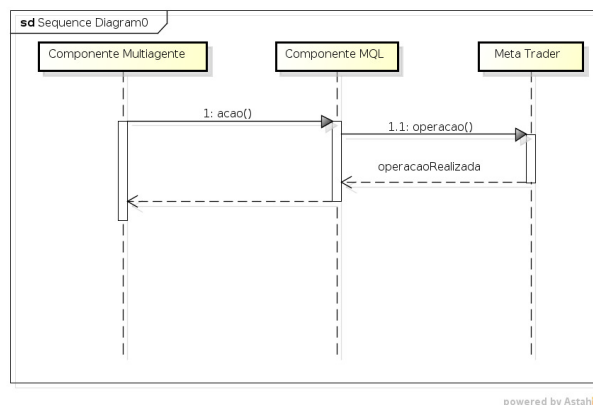


Figura 44 – Diagrama de Sequência do Componente MQL InvestMVC.

7.2.6 Fluxo de atividades da ferramenta InvestMVC

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

O componente Multiagentes vai verificar a tendência do Mercado de Moedas, por meio da plataforma MetaTrader. Sendo assim, o componente Multiagentes buscará na persistência o Expert que está ativo. Sabendo qual o Expert que foi ativado, o componente Multiagente faz a requisição de cálculos para os módulos C e Haskell. A partir desse resultado, o componente Multiagente procurará no Módulo Base de Conhecimento,

a alavancagem (quanto deve arriscar) e os valores de entrada para o método de Correção de Pearson, Fibonacci e Mínimos Quadrados. Caso todas as especificações para o módulo Multiagentes sejam obedecidas, ele informa ao componente MQL para realizar uma compra ou venda.

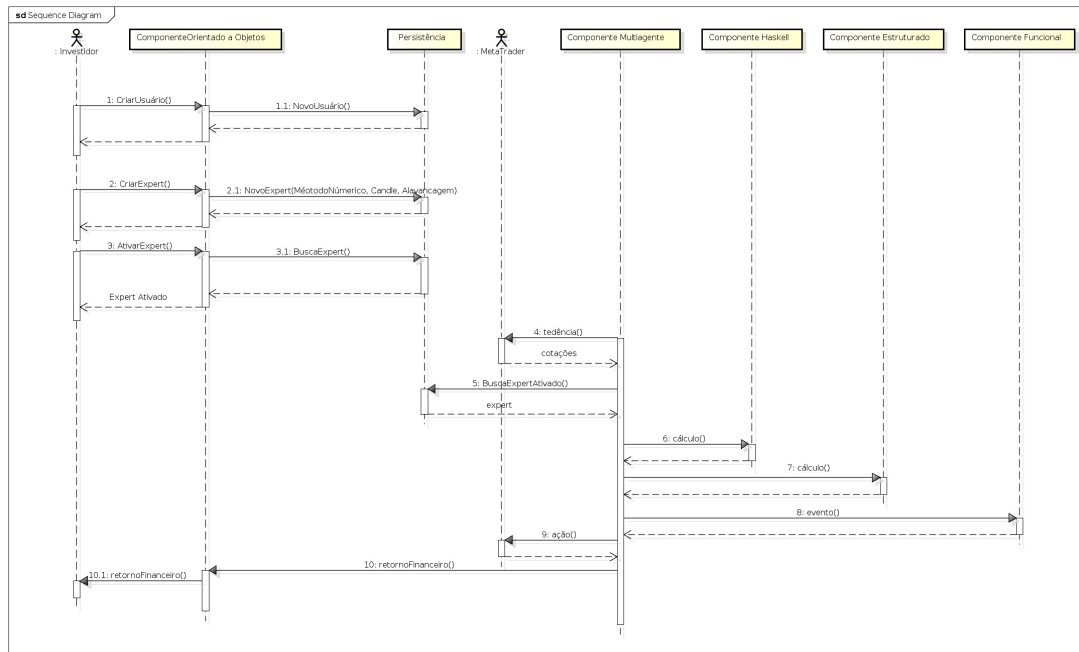


Figura 45 – Diagrama de Sequência InvestMVC

Fonte: Autores

8 Resultados

Na tabela 4 pode ser visualizado as Histórias de Usuário, a pontuação e o estado (a fazer, em progresso ou feito) de cada História. Foram implementados 12 pontos do total de 48 pontos estimados. Sendo assim, já foi implementado 25% dos pontos da ferramenta InvestMVC.

8.0.7 Resultados do Componente Orientado a Objetos

8.0.8 Resultados do Componente Funcional

8.0.9 Resultados do Componente Estruturado

Encontra-se no apêndice D, 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 46 é possível ver o resultado da suite de teste do Componente Estruturado. A cobertura de código encontra-se em 100%.

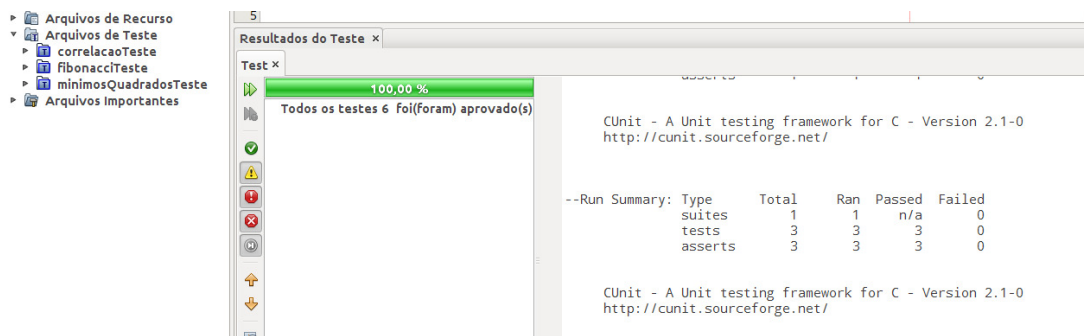


Figura 46 – Cobertura de código fonte do Componente Estruturado InvestMVC

A figura 47 revela o resultado da qualidade de código fonte do Componente Estruturado. É possível perceber que praticamente que a métrica de números de parâmetros por método/função (ANPM) ficou nível bom no pacote de Métodos Matemáticos em C. O acoplamento (ACC) do pacote Testes dos Métodos Matemáticos em C ficou em nível bom. As demais métricas ficaram em níveis excelentes. Portanto, o código fonte do Componente Estruturado da ferramenta InvestMVC ficou uma qualidade excelente em sua maior parte.

Tabela 4 – Atividades e objetivos metodologia de pesquisa

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

	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, ∞[
Métodos Matemáticos em C	0	2,67	2	0	0	0
Testes dos Métodos Matemáticos em C	2	1,66	1	0	0	3

Figura 47 – Resultado da Qualidade de Código Fonte do Componente Estruturado

8.0.10 Resultados do Componente Multiagente

Nenhum agente do Componente Multiagente foi construído por completo. Entretanto, toda a estrutura interna desse componente já está definida conforme consta na figura X.

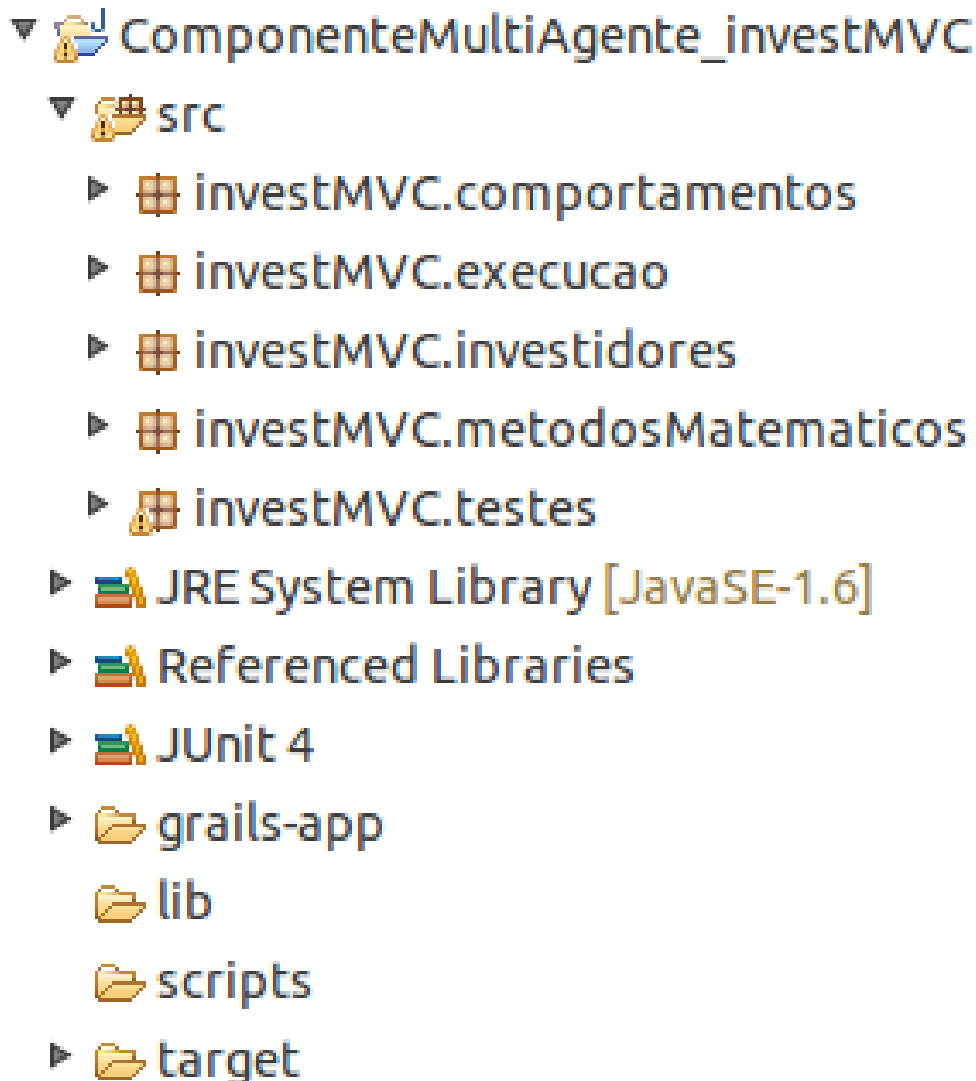


Figura 48 – Estrutura Interna Componente Multiagente

8.0.11 Resultados do Componente Lógico

8.0.12 Resultados do Componente MQL

No apêndice F - Componente MQL, é possível visualizar o código do Componente MQL que será responsável por interagir com a plataforma de negociação MetaTrader. É possível visualizar na figura 49, o Componente MQL (InvestMVC ComponenteMQL) já instalado na plataforma.



Figura 49 – Componente MQL instalado na plataforma MetaTrader

9 Considerações Finais

Cada Paradigma de Programação presente na ferramenta InvestMVC teve seu papel bem definido através da arquitetura guiada por componentes. Também foi definido que cada componente terão seus testes unitários específicos levando em consideração que cada um é implementado em uma linguagem de programação diferente. O Paradigma Multiagente e Estruturado terão análise estática de código fonte para externalizar a qualidade de código fonte desses paradigmas. A princípio não foi definido análise estática de código para os componentes dos outros paradigmas de programação por causa da ausência de ferramentas que suportem as linguagens de programação desses paradigmas.

Através do Protocolo de Experimentação, os Métodos Matemáticos de Correlação de Pearson, Fibonacci e Mínimos Quadrados foram selecionados para serem implementados na ferramenta InvestMVC. Portanto, espera-se que esses métodos tenham sucesso na ferramenta.

Conforme explanado no capítulo de resultados, a ferramenta InvestMVC encontra-se 25% finalizada, pois 12 pontos foram concluídos dos 48 pontos definidos no backlog. Qualquer História de Usuário, só é considerada como pronta quando é realizado todos os testes unitários pertinentes. Acredita-se que o percentual de 25% em breve será aumentado, pois existem várias Histórias de Usuário que estão sendo testadas para serem consideradas finalizadas.

Referências

ADVFN. Technical analysis. 2013. Disponível em: <<http://advfn.com/educacional/analise-tecnica>>. Citado na página 36.

AGENTBUILDER. Auction agents for the electric power industry. 2009. Disponível em: <<http://www.agentbuilder.com/Documentation/EPRI/index.html>>. Citado na página 45.

AGENTBUILDER. Mercado forex: Série alertas. 2009. Disponível em: <<http://www.cvm.gov.br/port/Alertas/mercadoForex.pdf>>. Citado 2 vezes nas páginas 45 e 86.

ALBUQUERQUE, A. A. *Alavancagem Financeira e Investimento*. Monografia (Especialização) — Faculdade de Administração, Universidade de São Paulo, São Paulo, 2013. Citado na página 30.

ALMEIDA, R. J. de A. Automatizando a criação de uma base de conhecimento em prolog para gerenciar os acessos a um site. 2010. Disponível em: <<http://www.vivaolinux.com.br/artigo/Automatizando-a-criacao-de-uma-base-de-conhecimento-em-Prolog-para-gerenciar-os-acessos-a-um>>. Citado 2 vezes nas páginas 47 e 87.

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 35.

AMBLER, S. *Uma Visão Realística da Reutilização em Orientação a Objetos*. [S.l.], 1998. Citado na página 43.

APT, K. R. *From Logic programming to Prolog*. 1. ed. [S.l.], 1996. Citado na página 45.

BANDEIRA, M. *Tipos de Pesquisa*. Monografia (Artigo) — Faculdade de Psicologia, FUNREI, Minas Gerais, 2012. Citado na página 53.

BEIZER, B. *Software Testing Techniques*. 2. ed. [S.l.], 1990. Citado na página 47.

BELCHIOR, G. P. *Oficina de Metodologia Científica: Elaboração de Projetos de Pesquisa*. [S.l.], 2012. Citado na página 29.

BRADSHAW, J. M. *Software Agents*. [S.l.], 1997. Citado na página 43.

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 na página 50.

BROOKSHEAR, J. G. *Ciência da Computação: Uma Visão Abrangente*. 3. ed. [S.l.], 2003. Citado na página 40.

BRUNI, A. L.; FAMÁ, R. *Gestão de custos e formação de preços*. São Paulo, 2011. Citado na página 30.

- 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 50 e 51.
- CAPRETZ, L. F. *A Brief History of the Object-Oriented Approach*. [S.l.], 2003. Citado na página 41.
- CHEESMAN, J.; DANIELS, J. *UML Components*. [S.l.], 2001. Citado na página 82.
- CHESS, B.; WEST, J. *Secure Programming with Static Analysis*. [S.l.], 2007. Citado na página 51.
- 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 45.
- 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 30 e 32.
- 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 44.
- CVM. Mercado forex: Série alertas. 2009. Disponível em: <<http://www.cvm.gov.br/port/Alertas/mercadoForex.pdf>>. Citado 3 vezes nas páginas 25, 29 e 30.
- 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 30.
- DEBASTINI, C. A. *Análise técnica de ações: identificando oportunidades de compra e venda*. 1. ed. [S.l.], 2008. Citado 2 vezes nas páginas 30 e 31.
- DEVORE, J. L. *Probabilidade e Estatística para Engenharia e Ciências*. 6. ed. [S.l.], 2006. Citado na página 33.
- DIAS, M. A. P. *Administração de materiais: uma abordagem logística*. 2. ed. [S.l.], 1985. Citado na página 32.
- EASYFOREX. Leveraged forex trading: What is leverage in forex trading? 2014. Disponível em: <<http://www.easy-forex.com/int/leveragedtrading/>>. Citado na página 30.
- FERREIRA, A. B. de H. *Novo Dicionário da Língua Portuguesa*. 2. ed. [S.l.], 2008. Citado na página 38.
- FERREIRA, D. F. *Estatística básica*. 2. ed. [S.l.], 2009. Citado na página 33.
- 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 51 e 52.

- FORCON. Metodologias para monografia. 2014. Disponível em: <<http://www.fazermonografia.com.br/metodologia-para-monografia>>. Citado 2 vezes nas páginas 53 e 54.
- FXCM. Forex basic. 2011. Disponível em: <<http://www.fxcm.com/forex-basics/>>. Citado na página 29.
- GAGLIARDI, J. D. *Fundamentos de Matemática*. Monografia (Monografia) — Universidade Federal de Campinas; São Paulo, 2013. Citado na página 35.
- GIL, A. C. *Como elaborar projetos de pesquisa*. 4. ed. [S.l.], 2008. Citado na página 53.
- GIRARDI, R. *Engenharia de Software baseada em Agentes*. [S.l.], 2004. Citado 2 vezes nas páginas 43 e 44.
- HOOGLÉ. Functional programming. 2013. Disponível em: <http://www.haskell.org/haskellwiki/Functional_programming>. Citado 2 vezes nas páginas 46 e 84.
- HOPPEN, C. *Metodologia Científica: Pesquisa Experimental*. Monografia (Graduação) — Universidade Federal do Paraná, 2010. Citado na página 64.
- IEEE 610. *IEEE Standard Glossary of Software Engineering Terminology*. [S.l.], 1990. Citado 4 vezes nas páginas 47, 48, 49 e 50.
- INVESTFOREX. Análise técnica para mercado forex. 2014. Disponível em: <<http://www.investforex.pt/aprender-forex/analise-tecnica>>. Citado 2 vezes nas páginas 37 e 38.
- 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 32.
- JENNINGS, N. R.; SYCARA, K.; WOOLDRIDGE, M. *A Roadmap of Agent Research and Development*. [S.l.], 1998. Citado na página 45.
- JENNINGS, N. R.; WOOLDRIDGE, M. *Intelligent agents: theory and practice*. [S.l.], 1995. Citado na página 45.
- JUNGTHON, G.; GOULART, C. M. *Paradigmas de Programação*. Monografia (Monografia) — Faculdade de Informática de Taquara, Rio Grande do Sul, 2009. Citado na página 84.
- KONIS, K. Mathematical methods for quantitative finance. 2014. Disponível em: <<https://www.coursera.org/course/mathematicalmethods>>. Citado na página 32.
- 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 83.
- 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 42.
- LEWIS, W. E. *Software Testing and Continuous Quality Improvement*. 3. ed. [S.l.], 2009. Citado na página 50.

- 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 33 e 35.
- LOPES, F. D. *Caderno didático: estatística geral*. [S.l.], 2005. Citado na página 34.
- MARKET. About what is forex. 2011. Disponível em: <<http://www.markets.com/pt/education/forex-education/what-is-forex.html>>. Citado na página 29.
- MATSURA, E. *Comprar ou Vender? Como investir na Bolsa Utilizando Análise Gráfica*. 7. ed. [S.l.], 2006. Citado 2 vezes nas páginas 30 e 31.
- MEIRELLES, P. *Monitoramento de métricas de código-fonte em projetos de Software Livre*. Monografia (Tese de Doutorado) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2013. Citado 2 vezes nas páginas 51 e 52.
- 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 51.
- 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 52.
- MYERS, G. J. *The art of Software Testing*. 2. ed. [S.l.], 2004. Citado 3 vezes nas páginas 47, 48 e 50.
- NAIK, K.; TRIPATHY, P. *SOFTWARE TESTING AND QUALITY ASSURANCE Theory and Practice*. 2. ed. [S.l.], 2008. Citado 2 vezes nas páginas 48 e 49.
- 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 2 vezes nas páginas 47 e 48.
- 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 38.
- ODELL, J.; GIORGINI, P.; MÜLLER, J. P. *Agent-Oriented Software Engineering V*. [S.l.], 2005. Citado na página 44.
- PAQUET, J.; MOKHOV, S. *Comparative Studies of Programming Languages*. [S.l.], 2010. Citado 2 vezes nas páginas 39 e 46.
- PIPONI, D. Eleven reasons to use haskell as a mathematician. 2006. Disponível em: <<http://blog.sigfpe.com/2006/01/eleven-reasons-to-use-haskell-as.html>>. Citado 2 vezes nas páginas 46 e 84.
- REGRA, C. M. *Tese de Mestrado em Estatística Computacional*. Monografia (Monografia) — Universidade Aberta, 2010. Citado 2 vezes nas páginas 34 e 35.
- RILEY, F.; HOBSON, M. P.; BENCE, S. J. *Mathematical Methods for Physics and Engineering: A Comprehensive Guide*. [S.l.], 2011. Citado na página 32.

- 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 40.
- ROBOFOREX. Indicators used in forex market. 2013. Disponível em: <<http://www.roboforex.pt/beginner/start/technical-analysis>>. Citado na página 38.
- ROCHA, R. A. *Algumas Evidências Computacionais da Infinitude dos Números Primos de Fibonacci*. Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) — Universidade Federal do Rio Grande do Norte, Natal, 2008. Citado 2 vezes nas páginas 35 e 36.
- RODRIGUES, W. C. *Metodologia Científica*. [S.l.], 2007. Citado na página 53.
- RUSSEL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3. ed. [S.l.], 1995. Citado na página 43.
- SCHUMACHER, M. *Objective Coordination in Multi-Agent System Engineering: design and implementation*. 1. ed. [S.l.], 2001. Citado na página 43.
- 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 56.
- SEBESTA, R. W. *Concepts of programming languages*. 10. ed. [S.l.], 2012. Citado 3 vezes nas páginas 39, 40 e 46.
- 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 43.
- SINGH, C. Method overloading in java with examples. 2014. Disponível em: <<http://beginnersbook.com/2013/05/method-overloading/>>. Citado na página 42.
- SOMMERVILLE, I. *Engenharia de Software*. 9. ed. [S.l.], 2011. Citado 2 vezes nas páginas 51 e 52.
- SPILLNER, A.; LINZ tilo; SCHAEFER, H. *Software Testing Foundations*. 4. ed. [S.l.], 2014. Citado na página 50.
- SPIVEY, M. *An introduction to logic programming through Prolog*. 1. ed. [S.l.], 1996. Citado na página 46.
- STEFANOV, S. *Object-Oriented JavaScript*. [S.l.], 2008. Citado na página 41.
- 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 49.
- THOMPSON, S. *Haskell: The Craft of Functional Programming*. 2. ed. [S.l.], 1999. Citado na página 45.
- TUCKER, A. B.; NOONAN, R. E. *Linguagens de programação : Princípios e paradigmas*. 2. ed. [S.l.], 2009. Citado 4 vezes nas páginas 40, 41, 46 e 47.

- VENNERS, B. Polymorphism and interfaces. 1996. Disponível em: <<<http://www.artima.com/objectsandjava/webuscript/PolymorphismInterfaces1.html>>>. Citado na página 42.
- VIALI, L. M. *Estatística Básica*. Monografia (Monografia) — Instituto de Matemática da Universidade Federal do Rio Grande do Sul, 2009. Citado na página 34.
- VUOLO, J. H. *Fundamentos da Teoria de Erros*. 2. ed. [S.l.], 1996. Citado na página 32.
- WEGNER, P. *Concepts and paradigms of object-oriented programming*. 1. ed. [S.l.], 1990. Citado na página 41.
- WEISFELD, M. A. *The Object-Oriented Thought Process*. 3. ed. [S.l.], 2009. Citado na página 42.
- WILLIAMS, L. *White-Box Testing*. [S.l.], 2006. Citado 3 vezes nas páginas 48, 49 e 50.
- WOLFRAM. Built with mathematica technology: Movingaverage. 2012. Disponível em: <<http://mathworld.wolfram.com/MovingAverage.html>>. Citado 2 vezes nas páginas 37 e 38.
- WOOLDRIDGE, M. *Teamwork in Multi-agent systems: A Formal Approach*. 1. ed. [S.l.], 2010. Citado 2 vezes nas páginas 43 e 44.

10 Glossário

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

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

Cotação:

CSV:

GPL3: Licença para Software Livre.

LGPL3: Sistema operacional.

MQL4: Linguagem em paradigma estruturado que permite implementar experts.

MQL5: Linguagem em orientado a objetos que permite implementar experts.

Stop loss:

Take profit:

Tendência: revela a direção em que o mercado está indo.

Apêndices

APÊNDICE A – Cronograma InvestMVC

ScrumMe (beta)
Project:
TCC Invest MVC²

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

Sprint: Sprint1

Construir Introdução

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

Implementar Métodos Numéricos

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

Sprint: Sprint2

Construir Referencial Teórico Paradigmas de Programação

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

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

Construir Referencial Teórico de Contexto Financeiro

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

Revisar Referencial Teórico Métodos Numéricos

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

Sprint: Sprint4

Realizar Experimentos Métodos de Operação

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

Realizar Experimento Método de Mínimos Quadrados	Cleiton				
Evidenciar Aplicações Paradigmas de Programação					
Evidenciar aplicações paradigma MultiAgente	Vanessa				
Evidenciar aplicações paradigma Funcional	Vanessa				
Evidenciar aplicações paradigma Estruturado	Vanessa				
Evidenciar aplicações paradigma Lógico	Vanessa				
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 MediaMovel	Vanessa	10/28/2014			
Desenvolver Expert MinimosQuadrados	Cleiton	10/28/2014			
Desenvolver Expert CorrelacaoLinear	Cleiton	10/28/2014			
Desenvolver Expert Estocastico	Vanessa	10/28/2014			
Desenvolver Expert Fibonacci	Cleiton	10/28/2014			
Sprint: Sprint5					

Descrever Metodologia de Pesquisa

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

Realizar Experimentos Métodos de Operação

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

Evidenciar Aplicações Paradigmas de Programação

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

Sprint: Sprint6

Verificar Qualidade de Código

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

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

APÊNDICE B – Experts para Experimento

Experts em linguagem MQL4

CorrelacaoPearson.mql

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

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

```

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

```

```

63.         "CORRELAÇÃO LINEAR 55 ", correlacao_pearson(55) , "\n",
64.         "CORRELAÇÃO LINEAR 34 ", correlacao_pearson(34) , "\n",
65.         "CORRELAÇÃO LINEAR 21 ", correlacao_pearson(21) , "\n",
66.         "CORRELAÇÃO LINEAR 13 ", correlacao_pearson(13) , "\n",
67.         ""
68.     );
69.     return(0);
70. }
71. double correlacao_pearson(int tempoCorrelacao){
72.     int c =0;
73.     double soma_ordenadas = 0;
74.     double soma_abscissas = 0;
75.     double soma_ordenadas_quadrado = 0;
76.     double soma_abscissas_quadrado = 0;
77.     double numero_abscissa;
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_abscissa = NormalizeDouble(Open[c],5);
84.         numero_ordenada =NormalizeDouble(Close[c],5);
85.         soma_abscissas = soma_abscissas + numero_abscissa;
86.         soma_abscissas_quadrado = (soma_abscissas_quadrado) +
            (numero_abscissa)*(numero_abscissa);
87.         soma_ordenadas = soma_ordenadas + numero_ordenada;
88.         soma_ordenadas_quadrado = (soma_ordenadas_quadrado) +
            (numero_ordenada)*(numero_ordenada);
89.         soma_X_vezes_Y = soma_X_vezes_Y + (numero_ordenada*numero_abscissa);
90.     }
91.
92.     numerador
    =((tempoCorrelacao*soma_X_vezes_Y)-((soma_abscissas)*(soma_ordenadas)));

```



```

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

```

Estocastico.mql

```

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

```

```

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

```

```

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

```

Fibonacci.mql

```

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

```

```

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

```

```

43.         }
44.     }
45.     Comment(
46.         "Margem da Conta = ", AccountMargin() ,"\n",
47.         "Ordens em lucro = ", ponto_positivo ,"\n",
48.         "Ordens em prejuizo = ", ponto_negativo ,"\n",
49.         "Suporte = ", suporte() ,"\n",
50.         "Resistencia = ", resistencia() ,"\n",
51.         "Retracao Fibo = ", retracao_fibonacci() ,"\n",
52.         ""
53.     );
54.     return(0);
55. }

56. double estadoMercado(int tempoCorrelacao){
57.     double soma_ordenadas = 0, soma_abcissas = 0;
58.     double numero_abcissa, numero_ordenada;
59.
60.     for(int c=0; c<tempoCorrelacao; c++){
61.         numero_abcissa = NormalizeDouble(Open[c],5);
62.         numero_ordenada =NormalizeDouble(Close[c],5);
63.         soma_abcissas = soma_abcissas + numero_abcissa;
64.         soma_ordenadas = soma_ordenadas + numero_ordenada;
65.     }
66.     return ( soma_abcissas - soma_ordenadas);
67. }

68. double retracao_fibonacci(){
69.     double retracao = ( resistencia() - suporte())*FATOR_RETRACAO;
70.
71.     return (retracao);
72. }

```

MediaMovel.mql

```

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

```

```

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

```

```

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

```

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

MinimosQuadrados.mql

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



```

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

```

```

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

```

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

ResistenciaSuporte.mql

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


APÊNDICE C – Componente Multiagente

APÊNDICE D – Componente Funcional

CorrelacaoDePearson.hs

```
1. module CorrelacaoDePearson (calculaMedia, vetorX, vetorY, varianciaPearson,
   correlacaoDePearson) where
2.
3. calculaMedia [] = 0
4. calculaMedia (cabeca:calda) = sum (cabeca:calda) / fromIntegral(length (cabeca:calda))
5.
6. vetorX [] = []
7. vetorX (cabeca : calda) = [cabeca] ++ vetorX ( drop 1 calda)
8.
9. vetorY [] = []
10. vetorY (cabeca : calda) = [(head calda)] ++ vetorY ( drop 1 calda)
11.
12. covarianciaPearson [] [] xMedio yMedio = 0
13. covarianciaPearson (cabecaX : caldax) (cabecaY : calday) xMedio yMedio = (cabecaX -
   xMedio) * (cabecaY - yMedio) +(covarianciaPearson caldax calday xMedio yMedio)
14.
15. varianciaPearson [] media = 0
16. varianciaPearson (cabeca : calda) media = (cabeca - media) ** 2 + (varianciaPearson
   calda media)
17.
18. correlacaoDePearson (cabeca:calda) = do
19.     let x = vetorX (cabeca:calda)
20.     let y = vetorY (cabeca:calda)
21.
22.     let mediaX = calculaMedia x
23.     let mediaY = calculaMedia y
24.
25.     let covariancia = covarianciaPearson x y mediaX mediaY
26.     let varianciaX = varianciaPearson x mediaX
27.     let varianciaY = varianciaPearson y mediaY
28.
29.     let varianciaXY = varianciaY * varianciaX
30.     let raizVarianciaXY = sqrt (varianciaXY)
```



```
31.  
32.         return (covariancia / raizVarianciaXY)
```

testeCorrelacaoDePearson.hs

```
1. import Test.HUnit  
2. import CorrelacaoDePearson  
3.  
4. testaCalculaMedia :: Test  
5. testaCalculaMedia = TestCase (assertEqual "Para vetor [3,3,3]" 3 (calculaMedia [3,3,3]))  
6.  
7. testaVetorX :: Test  
8. testaVetorX = TestCase (assertEqual "Para vetor [1,2,3,4]" [1,3] (vetorX [1,2,3,4]))  
9.  
10. testaVetorY :: Test  
11. testaVetorY = TestCase (assertEqual "Para vetor [1,2,3,4]" [2,4] (vetorY [1,2,3,4]))  
12.  
13. suiteDeTeste :: Test  
14. suiteDeTeste = TestList[ TestLabel "Media do Vetor" testaCalculaMedia, TestLabel  
    "Posições Impares do vetor" testaVetorX, TestLabel "Posições Pares do vetor" testaVetorY]  
15.  
16. main :: IO Counts  
17. main = runTestTT suiteDeTeste
```


APÊNDICE E – 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))*
55.     ((tempoCorrelacao*somaOrdenadasQuadrado)-(somaOrdenadas*somaOrdenadas));
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 =
        %lf\n", calculoSuporte(13), calculoResistencia(13));
11.     printf("Regressao De Fibonacci = %lf\n", calculoRegressaoFibonacci(0.23, 13));
12.     return 0;
13. }
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 = calculoSujeito(quantidadeVelas) -
        calculoResistencia(quantidadeVelas);
```



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

testeFibonnaci

```
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. 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 calculoSuporte(int quantidadeVelas);
26.
27. void testCalculoSuporte() {
28.     int quantidadeVelas = 13;
29.     CU_ASSERT_DOUBLE_EQUAL(136.996, calculoSuporte(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",
        testCalculoRegressaoFibonacci)) ||
45.         (NULL == CU_add_test(pSuite, "testCalculoResistencia",
        testCalculoResistencia)) ||
46.         (NULL == CU_add_test(pSuite, "testCalculoSuporte", testCalculoSuporte))) {
47.         CU_cleanup_registry();
48.         return CU_get_error();
49.     }
50.     /* Run all tests using the CUnit Basic interface */
51.     CU_basic_set_mode(CU_BRM_VERBOSE);
52.     CU_basic_run_tests();
53.     CU_cleanup_registry();
54.     return CU_get_error();
55. }

```

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_y/quantidadeVelas);
26.        denominador = y[i]*(x[i] - soma_x/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 <CUnit/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, calculoResistencia(quantidadeVelas), 0.001 );
27. }
28.
29. double calculoSujeito(int quantidadeVelas);
30.
31. void testCalculoSujeito() {
32.     int quantidadeVelas = 13;
33.
34.     CU_ASSERT_DOUBLE_EQUAL(136.996, calculoSujeito(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",
testCalculoResistencia)) ||
54.         (NULL == CU_add_test(pSuite, "testCalculoSuporte", testCalculoSuporte))) {
55.         CU_cleanup_registry();
56.         return CU_get_error();
57.     }
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 F – Componente MQL

Código Componente MQL

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



```

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

```


Anexos

ANEXO A – Template de Protocolo Experimental

Template for a Laboratory Experiment Protocol

1. Change Record

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

2. Background

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

3. Design

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

4. Data Preparation and Collection

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

- c) define how the data will be stored.

5. Analysis

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

6. Study Limitations

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

7. Reporting

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

8. Schedule

Give time estimates for all of the major steps