



UNIFEI

Universidade Federal de Itajubá

Instituto de Engenharia Elétrica

Ferramenta de Desenvolvimento e de Aplicação de Lógica Fuzzy

Área de Automação e Sistemas Elétricos Industriais

Daniele de Alcântara Barbosa



Ferramenta de Desenvolvimento e de Aplicação de Lógica Fuzzy

Orientador: Dr. Leonardo de Mello Honório
Coorientador: Dr. Luiz Edival de Souza

DISSERTAÇÃO APRESENTADA À ESCOLA FEDERAL DE
ENGENHARIA DE ITAJUBÁ PARA A OBTENÇÃO DO TÍTULO DE
MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA

ITAJUBÁ
ESTADO DE MINAS GERAIS – BRASIL
2005

Barbosa, Daniele de Alcantara.

Ferramenta de Desenvolvimento e de

Aplicação de Lógica Fuzzy / Daniele de Alcântara Barbosa

– UNIFEI – 125p

Monografia apresentada a UNIFEI para obtenção do grau de mestre em Ciências em Engenharia.

1. Fuzzy 2. Sistema 3. Simulações

I. Título: Ferramenta de Desenvolvimento e de Aplicação de Lógica Fuzzy

Aprovado em

COMISSÃO DE AVALIAÇÃO

Uma homenagem à minha filha Bruna, pela
alegria que traz a minha vida.

Agradecimentos

Aos membros e professores que fazem parte do CRTI.

Ao meu Orientador, Leonardo de Mello Honório, meus agradecimentos pela orientação na realização deste trabalho.

A todos aqueles que direta ou indiretamente colaboraram para que este projeto fosse concluído.

Resumo

A lógica Fuzzy é utilizada para automatizar atividades que compreendem situações ambíguas, tratando os problemas sob um novo paradigma. A potencialidade está na capacidade de obter conclusões e respostas baseadas no vago, no ambíguo. Ela aproxima a decisão computacional da decisão humana, tornando as máquinas mais capacitadas ao seu trabalho. Pode ser utilizada para automatizar atividades de natureza industrial, biológica ou química.

Apesar da matemática envolvida no desenvolvimento desta lógica ser conhecida na literatura, apresentar-se-á uma forma diferente de representá-la. Serão descritos a análise e projeto, baseados em objetos de uma ferramenta de desenvolvimento de lógica Fuzzy.

O desenvolvimento da aplicação foi feito em uma plataforma para sistemas distribuídos, abrangendo seu uso. A linguagem utilizada é compatível com o Framework.NET.

O sistema funciona como uma dll (Dynamic Link Library) que, assim como uma função, pode ser utilizada por qualquer aplicativo, desde que incluídas no código do aplicativo as chamadas para as mesmas.

Como vantagem a ferramenta desenvolvida apresenta recursos gráficos de configuração da lógica fuzzy, simuladores de resultados, saída e entrada de dados via planilha do Excel e o mais importante, métodos abertos para o usuário. Com esta última funcionalidade é possível a interrupção do sistema em qualquer etapa de processamento podendo-se alterar ou implementar novas soluções. Desta forma o sistema pode ser utilizado para auxiliar quaisquer aplicações de fuzzy tornando-o uma poderosa ferramenta de auxílio e desenvolvimento de soluções computacionais para sua utilização em situações de incertezas e para controle de processos.

Para demonstrar o uso da ferramenta, um jogo foi implementado. A simulação consta de um sistema responsável por definir a direção que uma nave deve seguir com o objetivo de chegar à Terra. Entretanto existem alguns obstáculos - os meteoros.

Abstract

Fuzzy logic has been used to make activities which comprise ambiguous situations totally automatic, treating the problems under a new paradigm. Its potentiality lies in the capacity of drawing conclusions and getting answers based on vague, on ambiguous parameters. It brings the computerized decision close to the human decision, making the machines more capable to perform their tasks. It can be used to make activities of industrial, biological or chemical nature more automatic.

Even though the mathematics involved in the development of this logic is known in the available literature, a different form of representing it will be presented. The analysis and project will be outlined, based on objects of a Fuzzy logic development tool.

The application developed throughout this work made use of a distributed systems platform, covering its use. The language used is a .Net Framework compliant. The system works as a DLL (Dynamic Link Library) component which, just like a function, can be used by any application, provided the calls to the component are included in the application code.

As an advantage, the tool developed presents graphical resources for configuration of the Fuzzy logic, simulation of results, input and output of data through any Excel spreadsheet and, the most important, methods open to the user. With this last functionality, it is possible to interrupt the system at any stage of processing, so new solutions can be changed or implemented. This way, the system can be used to help any fuzzy applications making it a powerful tool to help and develop computer solutions for use under uncertainty situations and for process control.

So as to demonstrate the tool flexibility, a computer game has been implemented. The simulation is a system responsible for defining the direction a spacecraft must follow so as to reach planet Earth. However, some obstacles must be circumvented, namely the meteors.

Índice

1	Introdução.....	1
1.1	Aspectos Gerais.....	1
1.2	Objetivo.....	2
1.3	Organização do Trabalho.....	3
2	Metodologia empregada.....	4
2.1	Variáveis Lingüísticas.....	5
2.2	Regras Fuzzy.....	8
2.3	Processo de Inferência.....	9
2.4	Defuzzificação.....	10
2.5	Webservice.....	11
2.6	Smart Client.....	13
3	Análise, Projeto e Implementação.....	15
3.1	A Plataforma.....	15
3.2	Orientação a Objetos.....	15
3.3	Análise e Projeto.....	17
3.3.1	Análise.....	18
3.3.2	Projeto.....	27
4	Descrição das Telas do Sistema.....	40
4.1	Telas do Sistema.....	40
4.1.1	Tela Fuzzy – Tela Principal.....	41
4.1.2	Menu File.....	43
4.1.3	Menu Fuzzy.....	44
4.1.4	Menu Help.....	46
4.1.5	Tela Category.....	47
4.1.6	Tela Variables Definition.....	48
4.1.7	Tela Rules Generation.....	50
4.1.8	Tela Rules Definition.....	51
4.1.9	Tela Add Variable.....	53
4.1.10	Tela Add Membership.....	54
4.1.11	Tela System Simulation.....	55
4.1.12	Tela Analysis.....	57
4.1.13	Tela Project Data.....	58
4.1.14	Tela Report.....	59
4.1.15	Tela Alarm Configurations.....	61
4.1.16	Tela Edit Name - Rule.....	62
4.1.17	Tela Salvar.....	63
4.1.18	Tela Abrir.....	64
5	Simulações.....	65
5.1	Iniciar Sistema.....	67
5.2	Definir Categorias.....	68
5.2.1	Deletar Categoria.....	69
5.3	Adicionar Variáveis.....	69
5.3.1	Alterar dados da variável.....	71
5.3.2	Deletar variável:.....	72
5.4	Adicior Membership's.....	72
5.4.1	Alterar Dados dos Membership's.....	73

5.4.2	Dados de cada membership:.....	74
5.4.3	Alterando nome do Membership.....	76
5.4.4	Deletar membership:.....	77
5.5	Gerar Regras.....	77
5.5.1	Entrar com cada regra.....	79
5.5.2	Deletar regras.....	82
5.5.3	Editar nome da regra.....	83
5.5.4	Alterar ou Deletar Nome de Variáveis e/ou Memberships após definição das regras.....	84
5.6	Importar dados de Entrada.....	84
5.7	Simulação do Sistema.....	86
5.8	Relatório do Sistema Fuzzy.....	89
5.9	Salvar projeto.....	91
5.10	Abrir projeto.....	92
5.11	Fechar Sistema.....	92
5.12	O Jogo: Aplicação Collision Game.....	93
5.13	Conclusões.....	105
6	Conclusões Finais.....	107
6.1	Proposta de Desenvolvimento Futuro.....	108
7	Referências Bibliográficas.....	110

Índice de Figuras

Figura 2.1. Tipos de Funções de pertinência	5
Figura 2.2 Variáveis Linguísticas	7
Figura 2.3 - Índices de Pertinência	8
Figura 2.4 - Processo de Inferência por Máximos e Mínimos.....	9
Figura 2.5 - Centro de Gravidade pelo Centroíde	10
Figura 2.6 – Topologia de Integração entre aplicativos entre redes distintas.....	13
Figura 2.7 – Topologia de Integração entre dispositivos móveis	13
Figura 3.1 - Processo de Definição de lógica Fuzzy	17
Figura 3.2 - Diagrama de Classe	19
Figura 3.3 - Classe Sistema	19
Figura 3.4 - Classe Grandeza.....	22
Figura 3.5 - Classe Membership.....	24
Figura 3.6 - Classe Regra	25
Figura 3.7 - Classe Dados.....	26
Figura 4.1 - Fuzzy System.....	41
Figura 4.2 - Menu File	43
Figura 4.3 - Menu System	44
Figura 4.4 – Menu Help.....	46
Figura 4.5- Category.....	47
Figura 4.6 - Variables Definition.....	48
Figura 4.7 - Rules Generation	50
Figura 4.8 - Rule Definition	51
Figura 4.9 - AddVariable.....	53
Figura 4.10 - Add Membership	54
Figura 4.11 - System Simulation	55
Figura 4.12 - Analysis	57
Figura 4.13 - Data.....	58
Figura 4.14 - Report	59
Figura 4.15 - Export	61
Figura 4.16 - Edit Rule	62
Figura 4.17 - Salvar	63
Figura 4.18 - Abrir.....	64
Figura 5.1 - Novo Sistema.....	67
Figura 5.2 - Tela Variables Definition.....	68
Figura 5.3 - Lista de Categorias.....	68
Figura 5.4 - Menu Adicionar Variável	69
Figura 5.5 - Adicionar Variável.....	70
Figura 5.6 - Variável Ativa.....	70
Figura 5.7 – Alterar Dados da Variável.....	71
Figura 5.8 – Menu Adicionar Membership	72
Figura 5.9 - Adicionar Membership	72
Figura 5.10 - Tela Completa.....	73
Figura 5.11 - Dados Memberships	74

Figura 5.12 - Valores Pontos Memberships	75
Figura 5.13 - Correspondência dos pontos	76
Figura 5.14 - Memberships.....	76
Figura 5.15 - Deletar Memberships.....	77
Figura 5.16 - Menu Gerar Regra	79
Figura 5.17 - Tela Rules Generation	80
Figura 5.18 - Geração das Regras.....	81
Figura 5.19 - Regras do Sistema.....	82
Figura 5.20 - Editar Nome da Regra.....	83
Figura 5.21 - Cabeçalho da Planilha.....	85
Figura 5.22 - Planilha Preenchida.....	85
Figura 5.23 - Menu Importar Dados	86
Figura 5.24 - Análise dos Dados	87
Figura 5.25 - Tela de Simulação.....	87
Figura 5.26 - Simulação	88
Figura 5.27 - Menu Report	89
Figura 5.28 - Relatório	90
Figura 5.29 - Relatório Regras	90
Figura 5.30 - Relatório Dados	91
Figura 5.31 - Relatório Memberships.....	91
Figura 5.32 - Menu Save As.....	91
Figura 5.33 - Abrir Projeto	92
Figura 5.34 - Memberships.....	95
Figura 5.35 - Algoritmo.....	96
Figura 5.36 - Primeira Simulação do Jogo	97
Figura 5.37 - Memberships 2 Simulação.....	99
Figura 5.38 - Segunda Simulação do Jogo	101
Figura 5.39 - Terceira Simulação do Jogo.....	103
Figura 5.40 - Terceira Simulação do Jogo.....	105

Índice de tabelas

Tabela 1.1 Variáveis lingüísticas.....	6
Tabela 1.2 Características da lógica fuzzy	11
Tabela 5. 1 Regras da primeira Simulação.....	77
Tabela 5.2 Regras da segunda Simulação.	100
Tabela 5.3 Regras da terceira Simulação.....	104

1 Introdução

1.1 Aspectos Gerais

Na década de 60, observando a ausência de recursos para trabalhar em situações de incertezas, Zadeh [1] propôs a base da lógica Fuzzy visando automatizar atividades que compreendessem situações ambíguas não passíveis de processamento através da tradicional lógica booleana.

Diversas áreas estão sendo beneficiadas com o uso desta lógica. Ela ganhou espaço como área de estudo, pesquisa e desenvolvimento, sendo hoje fundamental para aplicações que tenham determinados graus de incerteza. Disseminada na indústria a partir dos anos 70, atualmente é utilizada tanto em aplicações industriais, comerciais e biológicas. Dentre diversos exemplos estão produtos como máquinas de lavar, câmaras digitais, geladeiras, condicionador de ar, veículos auto guiados e robôs. Outra área que vem demonstrando grande potencial é a área de jogos de computador.

Fuzzy é uma técnica de inteligência artificial [2] baseada na Teoria dos Conjuntos, que serve para os modos de raciocínio aproximados ao invés de exatos. Seus conceitos podem ser utilizados para traduzir em termos matemáticos, informações imprecisas. É descrita em uma linguagem natural baseada em conhecimentos heurísticos. Ela possibilita que um problema que tenha certa imprecisão e incerteza de informação seja abordado de forma mais adequada.

Estes sistemas utilizam um conjunto de regras do tipo if then, formado pela união de todos os estados das grandezas utilizadas. Inicialmente as variáveis de entrada sofrem

um processo de fuzzificação, sendo que as informações são convertidas em números fuzzy, para então ocorrer a formulação e execução de uma estratégia de controle. Terminado este processo, efetua-se a inferência sobre o conjunto de regras obtendo os valores dos termos das variáveis de saída, isto é tratado na literatura como defuzzificação, que consiste em converter os dados nebulosos para valores numéricos precisos.

Este trabalho pretende fornecer uma ferramenta que facilite o uso de aplicações que necessitem trabalhar com incertezas. O sistema funciona com uma biblioteca de classes, fornecendo todas as funções necessárias.

1.2 Objetivo

O objetivo principal deste trabalho está centrado na modelagem de um sistema computacional de Lógica Fuzzy usando programação orientada a objetos.

Serão descritos a análise e o projeto, baseados em objetos de uma ferramenta de desenvolvimento de lógica Fuzzy.

A análise e o projeto orientados a objeto [3] têm se tornado paradigmas fundamentais no desenvolvimento de um software. O objetivo destas fases é mostrar o desenvolvimento do programa, desde o fluxo de informações, realizações de tarefas, ao dimensionamento. A modelagem deve ser de tal forma que permita a todos os envolvidos (analista, programador, cliente, etc.) uma compreensão única do projeto. Se usado de forma apropriada, dentre as vantagens estão o aumento da produtividade, o reuso e uma maior compreensão do sistema.

O desenvolvimento da aplicação será feito com uma plataforma para sistemas distribuídos, alcançando uma maior confiabilidade. A linguagem utilizada é o Visual Basic. NET [4]. Para a simulação a linguagem utilizada é C # [5].

Para demonstrar o uso da ferramenta, um jogo foi implementado. A simulação consta de um sistema responsável por definir a direção que uma nave espacial deve seguir tendo como objetivo chegar à Terra, onde toda a inteligência é baseada na lógica Fuzzy.

1.3 Organização do Trabalho

Este trabalho está organizado da seguinte forma: o capítulo 2 introduz alguns conceitos de Fuzzy; o capítulo 3 mostra os fundamentos da análise e do projeto orientados a objeto, com uma breve apresentação do paradigma orientado a objetos, além do desenvolvimento dos algoritmos de implementação; o capítulo 4 apresenta a descrição das telas do sistema e exemplos de como definir a entrada de dados; no capítulo 5 temos algumas simulações do uso da ferramenta e resultados; e por fim, no capítulo 6, será discutida a avaliação dos resultados.

2 Metodologia empregada

Este capítulo aborda de forma abrangente os principais conceitos da lógica Fuzzy. Será apresentado as variáveis lingüísticas, as regras, o processo de inferência utilizado, o processo de defuzzificação, além de vantagens e desvantagens no uso desta lógica.

A lógica Fuzzy pode ser utilizada para automatizar atividades de natureza industrial, biológica ou química. Sua importância na indústria vai desde o chão de fábrica, onde é aplicado o controle de processos, até o nível gerencial, sendo utilizada em datawarehouses¹ [6] e em outras ferramentas de auxílio à tomada de decisão. Cabe ressaltar que Fuzzy tem merecido lugar de destaque junto às pesquisas da área de Inteligência Artificial (I.A.).

Em oposição à lógica clássica, que trata apenas de dois valores (verdadeiro ou falso, pertencente ou não ao conjunto), no universo da lógica fuzzy são atribuídos valores reais, pertencentes ao intervalo $[0,1]$, identificando o grau de pertinência de cada elemento em um determinado conjunto.

Para exemplificarmos, considere dois conjuntos: pessoas altas e pessoas baixas. Não existe uma fronteira para determinar quanto um elemento pertence ou não aos conjuntos citados. Com Fuzzy é identificado graus de pertinência para cada elemento, de acordo com o intervalo de números reais de 0 a 1. Identificado o grau de pertinência de cada valor, é verificado quanto é possível para um dado elemento pertencer ao conjunto.

¹ datawarehouse é uma coleção de dados orientada por assuntos, integrada, variante no tempo, e não volátil, que tem por objetivo dar suporte aos processos de tomada de decisão.

Os gráficos destas funções de pertinência podem ter diferentes representações ou formas, sendo a representação mais adequada determinada pelo contexto da aplicação [1,7,8]. A Figura 2.1 apresenta cinco importantes tipos de funções de pertinência.

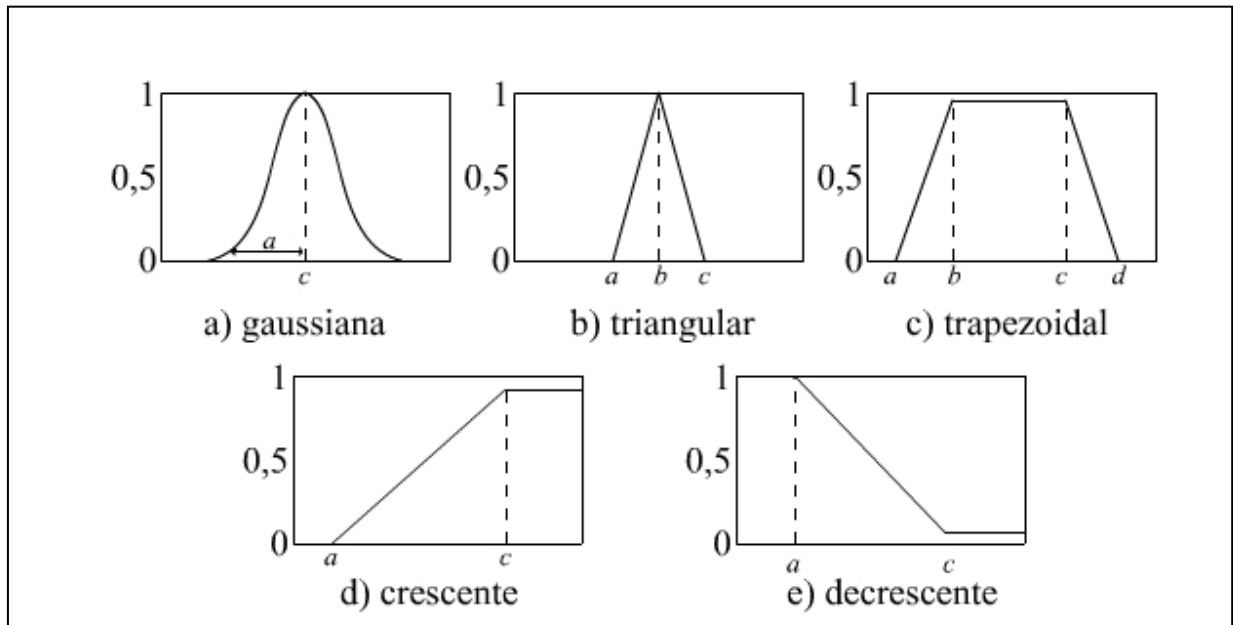


Figura 2.1. Tipos de Funções de pertinência
Fonte: Sousa, 1995.

2.1 Variáveis Lingüísticas

A lógica Fuzzy utiliza variáveis lingüísticas ao invés de numéricas. Estas variáveis modelam conhecimentos imprecisos e vagos sobre uma grandeza, cujos valores são desconhecidos. Pode ser considerada uma variável lingüística, aquela cujos valores assumidos são nomes de conjuntos fuzzy. A variável temperatura, por exemplo, pode assumir os valores alto, médio e baixo, que são descritos através de conjuntos Fuzzy, os quais são conhecidos na literatura por memberships.

Para Zadeh [1,9], uma variável lingüística é aquela cujo valor pode ser descrito:

- qualitativamente, usando uma expressão que envolva termos lingüísticos, e
- quantitativamente, usando uma correspondente função de associação.

A Tabela 1.1 mostra algumas variáveis lingüísticas com valores (memberships) frequentemente usados.

Tabela 1.1 Variáveis lingüísticas

Variáveis lingüísticas	Memberships
Idade	Novo, Adulto, Velho
Pressão	Alta, Média, Baixa
Velocidade	Baixa, Média, Alta

Na definição formal uma variável lingüística é uma quintupla $(X, T(X), U, G, M)$, onde:

X é o nome da variável.

$T(X)$ é o conjunto de nomes dos valores lingüísticos de X .

U é o universo de discurso.

G é a regra sintática para gerar os valores de X como uma composição de termos de $T(X)$, conectivos lógicos (negação, intersecção e união), modificadores e delimitadores.

M é a regra semântica que associa a cada valor gerado por G um conjunto fuzzy em U .

A regra sintática determina a maneira pela qual podem ser gerados os valores lingüísticos, que estão no conjunto de termos da variável. A regra semântica determina um procedimento computacional do significado de determinada variável lingüística [1,10].

Por exemplo, supondo-se o controle de temperatura de um ambiente através de um aparelho de ar condicionado, onde: se a temperatura estiver alta, o aparelho de ar condicionado deverá abaixar a potência para esfriar o ambiente; se a temperatura estiver média a potência deverá permanecer; e se a temperatura estiver baixa o aparelho de ar condicionado deverá elevar a temperatura.

A temperatura é classificada como: “muito baixa”, “baixa”, “média”, “alta” e “muito alta”. E o ajuste do ar condicionado em “frio”, “médio” e “quente”. Quando dizemos que a temperatura está baixa, isto não significa um valor exato, mas um intervalo. Desta forma, definimos as grandezas fuzzy através de conjuntos para representar os intervalos conforme entendemos através das variáveis.

A primeira grandeza, “temperatura”, representa a variável de entrada do sistema; e a grandeza “ajuste do ar condicionado” é utilizada para o resultado da lógica. Cada grandeza é então escalonada em memberships. Para cada valor de temperatura lido, é atribuído um valor de pertinência aos devidos memberships.

As figuras 2.2 e 2.3 representam as variáveis lingüísticas referente ao exemplo.

Na Figura 2.2 Variáveis Lingüísticas, a grandeza “Temperatura” está dividida em cinco memberships; “muito alta”, “alta”, “média”, “baixa” e “muito baixa”.

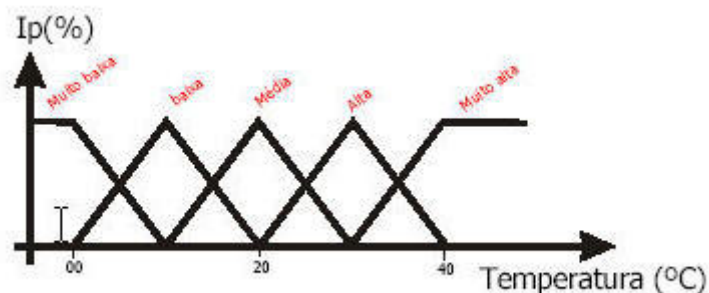


Figura 2.2 Variáveis Lingüísticas

A Figura 2.3 mostra, para um valor de temperatura, a obtenção dos índices de pertinência em relação aos memberships relacionados. Onde, dado um valor, é feita a projeção vertical deste para a aquisição dos pontos de interseção com os memberships. Uma nova projeção horizontal, feita a partir dos pontos de interseção, é traçada para a obtenção dos índices de pertinência do valor lido com esses memberships.

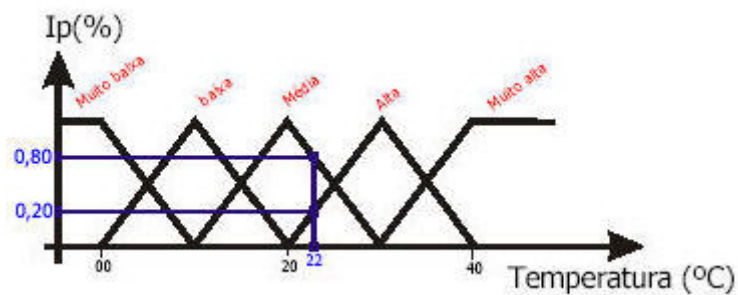


Figura 2.3 - Índices de Pertinência

A próxima etapa de um processo de lógica fuzzy é atribuir os índices de pertinência calculados a um processo de inferência, ou seja, as regras previamente determinadas, responsáveis pelo comportamento do sistema.

2.2 Regras Fuzzy

A construção de um sistema de lógica Fuzzy é baseada na idéia de incorporar conhecimentos de especialistas. Logo a estratégia de controle é representada por um conjunto de regras de decisão.

As regras são compostas por sentenças condicionais do tipo “if then“. Consiste em uma ou mais variáveis de entrada associadas a conjuntos nebulosos, e, uma ou mais variáveis de saída também associadas a conjuntos nebulosos.

As variáveis de entrada podem ser concentradas por operadores lógicos “and”.

Um exemplo pode ser claramente visualizado na frase abaixo.

“*Se a temperatura está muito alta, ligar o ar condicionado em muito frio*”.

Esta regra pode ser escrita em termos de variáveis lingüísticas da seguinte forma:

Se temperatura = alta então arcondicionado = frio.

2.3 Processo de Inferência

Construído o conjunto de regras, os índices de pertinência calculados são atribuídos a um processo de inferência para extrair a resposta final. Existem vários métodos para desenvolver o método de inferência. No entanto, para este ensaio, será usado o método de Mínimos e Máximos. Esse método trabalha com os valores de pertinência obtidos pelos processos anteriormente descritos. Nele, atribui-se ao índice da preposição um índice de pertinência igual ao mínimo dos valores de pertinência dos índices contidos na preposição. Podem existir casos em que uma função de pertinência esteja em mais de uma proposição, obtendo-se assim mais de um valor para o mesmo. Nesse caso o índice final para essa função é o máximo dos índices atribuídos ao mesmo².

A Figura 2.4 mostra o processo de inferência utilizando mínimos e máximos.

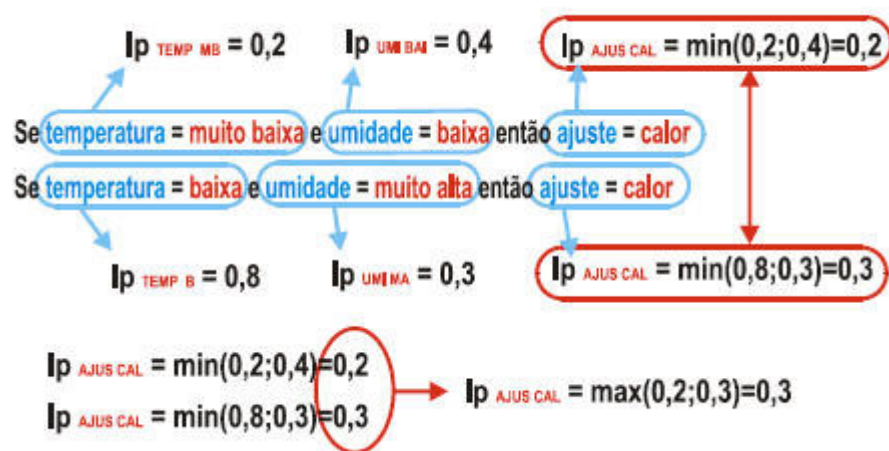


Figura 2.4 - Processo de Inferência por Máximos e Mínimos

² Para maiores informações, consultar as referências [1,7,10,11]

Observa-se na Figura 2.4, que neste processo de inferência, atribui-se ao índice da preposição um índice de pertinência igual ao mínimo dos valores de pertinência dos índices contidos na proposição.

O último passo desse processo é o cálculo do valor de saída, tratado na literatura como defuzzificação.

2.4 Defuzzificação

Como no processo de inferência, existem diversas formas de defuzzificação [1,7,10,11]. Os processos mais comuns são: Máximo, Média dos Máximos, Centróide, Altura e Altura modificada. Para a ferramenta, o processo de defuzzificação utilizado foi o do centro de gravidade pelo centróide, onde a saída precisa é o valor no universo que corresponde ao centro de gravidade do conjunto fuzzy.

A Figura 2.5 apresenta este processo de Defuzzificação pelo Centróide.

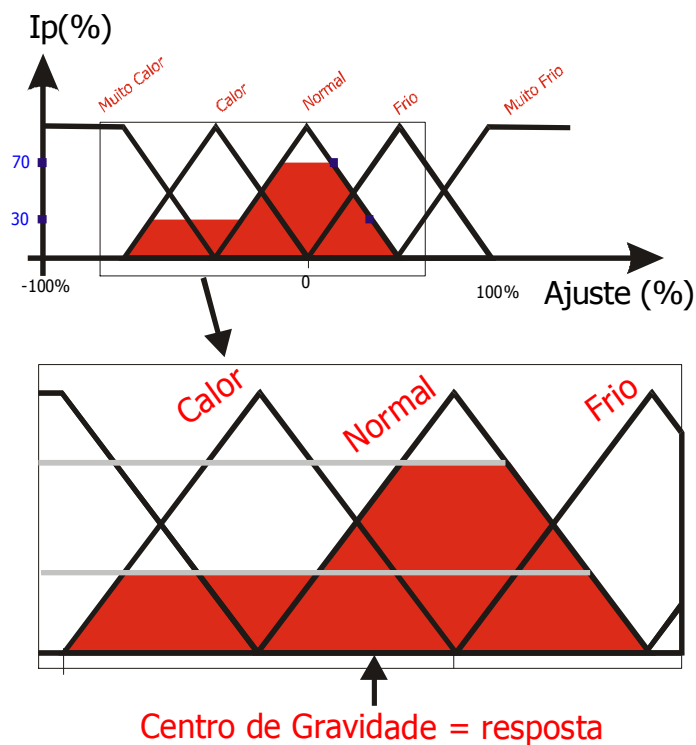


Figura 2.5 - Centro de Gravidade pelo Centróide

No exemplo mostrado acima, tem-se dois índices de pertinência associados a dois diferentes memberships. Acha-se a área sob cada membership, tendo como máximo o valor do índice de pertinência associado. A resposta da lógica, nesse caso, é o centro de gravidade da área formada pela união dos dois triângulos encontrados.

Também é interessante fazer-se claro algumas vantagens e desvantagens no uso desta lógica. A Tabela 1.2 apresenta algumas destas características.

Tabela 1.2 Características da lógica fuzzy

Características	Vantagens	Desvantagens
A Lógica Nebulosa está baseada em palavras e não em números, ou seja, os valores verdades são expressos lingüisticamente. Por exemplo: quente, muito frio, verdade, longe, perto, rápido, vagaroso, médio;	O uso de variáveis lingüísticas nos deixa mais perto do pensamento humano;	Necessitam de mais simulação e testes;
Possui vários modificadores de predicado, tais como: muito, mais ou menos, pouco, bastante, médio;	Requer poucas regras, valores e decisões;	Não aprendem facilmente;
Possui também um amplo conjunto de quantificadores, como: poucos, vários, em torno de, usualmente;	Simplifica a solução de problemas e a aquisição da base do conhecimento;	Dificuldades de estabelecer regras corretamente;
Faz uso das probabilidades lingüísticas (como, PE, provável e improvável) que são interpretados como números nebulosos e manipulados pela sua aritmética;	Mais variáveis observáveis podem ser valoradas;	Não há uma definição matemática precisa.
Manuseia todos os valores entre 0 e 1, tomando estes, como um limite apenas.	Mais fáceis de entender, manter e testar;	
	São robustos. Operam com falta de regras ou com regras defeituosas;	
	Acumulam evidências contra e a favor.	
	Proporciona um rápido protótipo dos sistemas.	

Fonte: Camargo [12]

2.5 Webservice

São componentes de software que provêm dados e serviços a outras aplicações, é uma maneira prática e eficiente de aplicativos se comunicarem, baseado no protocolo SOAP (Simple Object Access Protocol) e, portanto qualquer plataforma que interprete rotinas HTTP (Hypertext Transfer Protocol) e manipule XML pode utilizar os dados dos WEBSERVICES [13,14]. Simplificadamente pode-se dizer que um Webservice é como

um site Web sem interface para usuário, servindo aplicativos ao invés de pessoas, também é possível que ao invés de receber pedidos de navegadores e devolver páginas Web como resposta, um Webservice receba um pedido através de uma mensagem formatada em XML de um aplicativo, realize uma tarefa e devolva uma mensagem de resposta formatada em XML para o aplicativo.

Hoje é considerado um padrão do W3C como uma linguagem universal de troca de dados devido sua vantagem dos dados serem trafegados em formato XML, resultando que todos os dados possuam <tags> o que torna o processo de comunicação mais robusto, com dados consistentes e sem falhas. Outra vantagem é a possibilidade de ter uma infinidade de aplicativos conectados em rede, mesmo rodando em plataformas diferentes através de protocolos comuns e auto-descrição, que promovem um enfoque flexível e orientado a mensagens para a integração.

Um conjunto de bibliotecas de classe fornece rotinas que permite aos aplicativos ler e escrever dados no formato XML, comunicar-se via Internet, acessar bancos de dados e muito mais. Todas as bibliotecas de classe baseiam-se em uma biblioteca básica que fornece funções de gerenciamento dos tipos de dados usados com maior frequência, como strings e números, e funções de baixo nível, como acesso a arquivos.

Com as plataformas acima descritas é possível implementar integrações entre aplicativos da mesma intranet, aplicativos em redes corporativas diferentes e aplicativos de redes corporativas com dispositivos remotos, como mostram as figuras 2.6 e 2.7.

Integração entre Sistemas

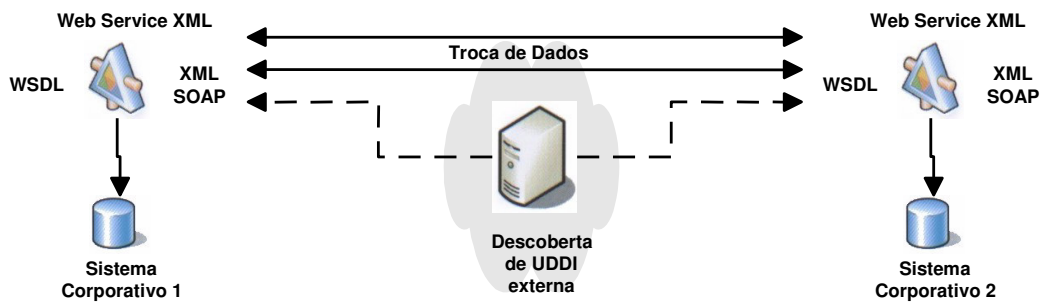


Figura 2.6 – Topologia de Integração entre aplicativos entre redes distintas

Dispositivos Móveis

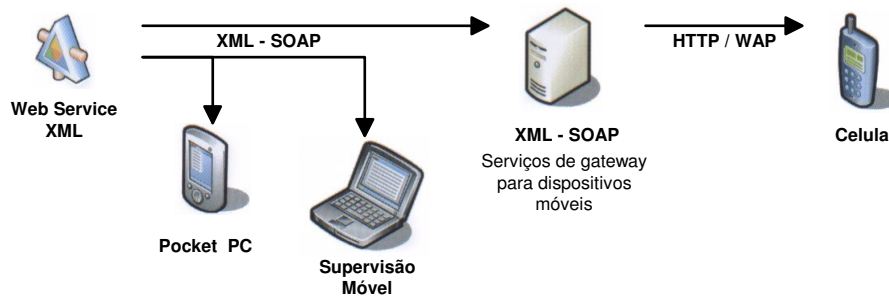


Figura 2.7 – Topologia de Integração entre dispositivos móveis

Portanto a utilização dessa tecnologia no desenvolvimento de aplicações, de acordo com as opções disponíveis possibilita configurar sistemas de forma distribuída.

2.6 Smart Client

São aplicações que utilizam Web Services XML como forma de comunicação, processamento local e pode ser distribuída e atualizada em um servidor central. São flexíveis e potentes, além de estáveis e de fácil distribuição. Dentre as características estão:

- São executadas na máquina cliente.
- Pode ser usado on-line ou off-line.

- Consome Web Services para prover grande funcionalidade e informações atualizadas para o usuário do negócio.
- Podem ser projetadas para rodar em múltiplos dispositivos, incluindo-se PDA's e telefones móveis.

3 Análise, Projeto e Implementação

Este capítulo tem a finalidade de apresentar a plataforma do desenvolvimento, a linguagem orientada a objetos, e também a análise, projeto e implementação orientados a objeto, fases fundamentais no desenvolvimento de um software. São apresentados os objetivos da ferramenta, e as contribuições que oferece.

3.1 A Plataforma

O sistema de lógica fuzzy foi feito em uma plataforma distribuída, possibilitando o desenvolvimento eficiente da aplicação para qualquer plataforma e qualquer dispositivo.

A mesma é baseada em objetos e classes. Na realidade possui diversas classes que funcionam como blocos lógicos para a construção de aplicações. O conceito de orientação a objeto é fundamental para utilização, definição e compreensão das classes.

3.2 Orientação a Objetos

O paradigma de orientação a objetos (OO) tem sido largamente empregado nas arquiteturas de desenvolvimento de software. Rumbaugh [15] define orientação a objetos como: "uma nova maneira de pensar os problemas utilizando modelos organizados a partir de conceitos do mundo real. O componente fundamental é o objeto que combina estrutura e comportamento em uma única entidade".

Dizer que um software é orientado a objetos significa que ele é estruturado como uma coleção de objetos separados que incorporam tanto a estrutura como o comportamento dos

dados. De forma simples, OO é uma técnica de programação para organizar programas, baseada em classes e objetos. Tem como objetivo principal reduzir a complexidade e aumentar a produtividade nos desenvolvimentos de software. Dentre os principais conceitos estão: classes e objetos, herança, polimorfismo e encapsulamento.

Uma classe representa uma entidade e como tal define seu comportamento e características. É um padrão a ser seguido pelo objeto, definindo como os objetos da classe se comportarão. As classes contêm variáveis e métodos, podendo herdar estes de outras classes. Objetos são instâncias de classes, que têm acesso aos métodos da mesma, e são formados por um conjunto de propriedades (variáveis) e procedimentos (métodos). Cada objeto é uma nova ocorrência ou uma instância de alguma classe. Na programação OO, manipulações de dados são através dos objetos.

O conceito de herança permite definir uma nova classe com base em alguma anterior. A nova classe herda todas as propriedades e métodos, podendo ainda programar as diferenças. É uma das responsáveis pela facilidade do reaproveitamento de código.

Polimorfismo tem a propriedade de usar o mesmo nome para métodos diferentes, implementados em diferentes níveis de hierarquia. Para cada classe tem-se um comportamento específico para o método. É responsável pela facilidade de extensão de um programa OO.

Por fim, encapsulamento funciona como uma proteção para as variáveis e métodos, onde a utilização dos objetos não depende de sua implementação interna, e sim de sua interface.

As vantagens da OO estão na alta capacidade de reutilização efetiva de código, na fácil manutenção e documentação, na maior agilidade no desenvolvimento, e na confiabilidade.

3.3 Análise e Projeto

Dentre as fases do desenvolvimento de um software, citaremos as fases de análise e projeto, do sistema Fuzzy. A fase da análise modela o problema principal (classes e objetos) e cria um modelo ideal do sistema sem levar em conta requisitos técnicos. A fase do design (projeto) expande e adapta os modelos da análise para um ambiente técnico, sendo trabalhadas em detalhes as soluções técnicas. Para Graic Larman[3], a análise enfatiza uma investigação do problema de como uma solução é definida e o projeto enfatiza uma solução lógica, ou seja, como o sistema atende aos requisitos.

- O Sistema

Um sistema fuzzy deve oferecer ao usuário a capacidade de definir as grandezas de entrada e saída que fazem parte do processo, ajustar os campos de pertinência (memberships) de cada grandeza e definir as regras (if then), que manipulem as variáveis associadas ao processo de defuzzificação.

A Figura 3.1 mostra as etapas básicas de um sistema que utiliza lógica Fuzzy.

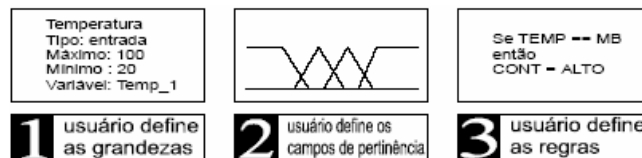


Figura 3.1 - Processo de Definição de lógica Fuzzy

Logo, para ser possível a criação de uma ferramenta de desenvolvimento de lógica fuzzy, é necessário o armazenamento dos dados envolvidos (grandezas, memberships, regras).

Os objetos do sistema são salvos no formato XML (Extensible Markup Language), utilizando-se de classes da plataforma, que permitem a serialização e deserialização de objetos neste formato. Serialização é um processo que converte um objeto de modo que ele possa ser transportado. Deserialização reconstrói o objeto em questão. Juntos estes processos permitem que dados sejam facilmente armazenados e transferidos.

Neste projeto, foi usada a tecnologia *XML serialization*. Esta tecnologia serializa somente propriedades e campos públicos não preservando a fidelidade ao tipo. Isto é útil quando se quer fornecer ou consumir dados sem restringir a aplicação que usa os dados. Cabe ressaltar que devido XML ser um padrão aberto, os dados podem ser processados independentes da plataforma.

3.3.1 Análise

A apresentação da análise será feita através do diagrama de classe usado na programação orientada a objetos. Um diagrama de classe ilustra as especificações para as classes do software de uma aplicação. É um gráfico que mostra a estrutura de um sistema.

Nesta fase todas as classes que fazem parte do domínio do problema foram modeladas e interligadas através de relacionamentos.

A Figura 3.2 apresenta o Diagrama de classe do sistema.

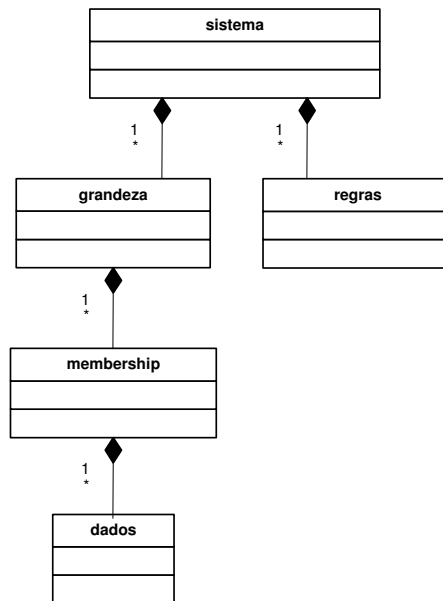


Figura 3.2 - Diagrama de Classe

A seguir, faz-se uma exposição detalhada das classes representadas na Figura 3.2 pelo diagrama de classe.

A Figura 3.3 ilustra a classe Sistema

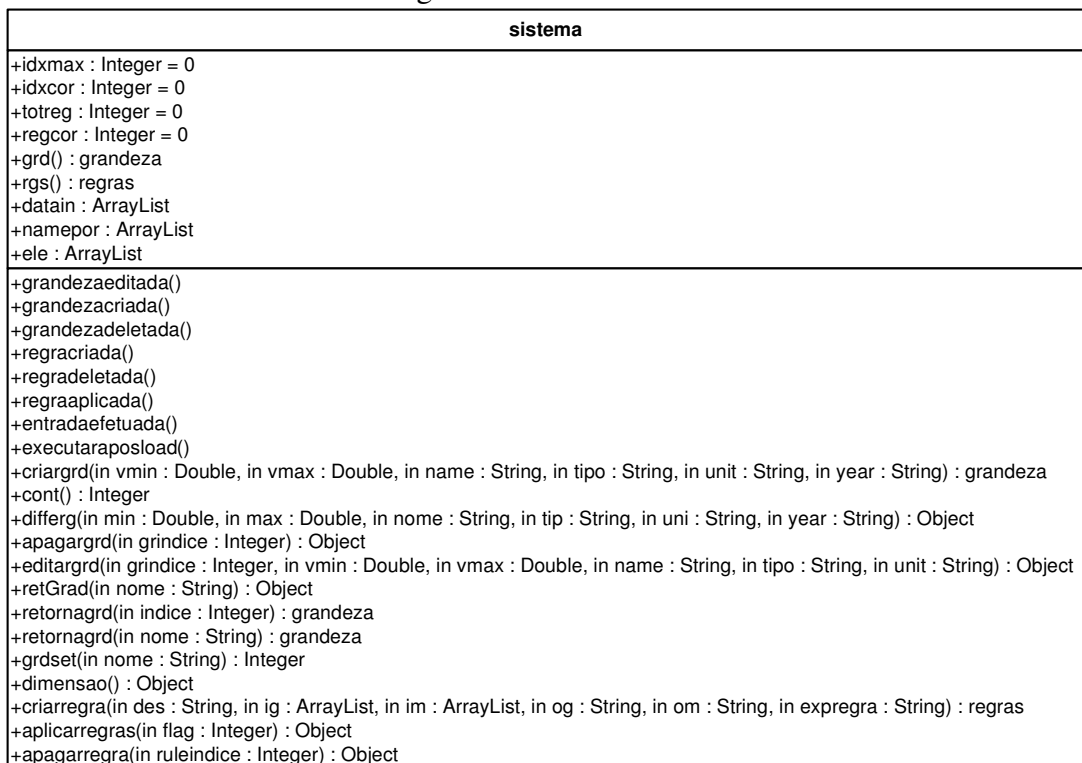


Figura 3.3 - Classe Sistema

Na descrição das classes serão abordados os dados que passaram pelo processo de serialização.

Os principais atributos e métodos da classe Sistema são:

Declarações da classe:

- ✓ **Class Sistema***
- ✓ **idxmax**: armazena o número de grandezas do sistema.
- ✓ **idxcor**: recebe índice grandeza corrente .
- ✓ **totreg**: armazena o número de regras do sistema.
- ✓ **grd()*** :Armazena dados da grandeza no arraylist grd() Tipo da variável: “Grandeza”.
- ✓ **rgs()*** :Armazena dados das regras no arraylist rgs() Tipo da variável: “Regras”

Eventos da classe:

Eventos são notificações que a classe envia e recebe para o código que a criou. O evento é associado a uma rotina que o gerencia.

- ✓ **Event grandezaeditada ()** : Ocorre quando uma grandeza é alterada.
- ✓ **Event grandezacriada ()**:Ocorre quando uma grandeza é criada.
- ✓ **Event grandezadeletada ()**:Ocorre quando uma grandeza é excluída.
- ✓ **Event regracriada ()**:Ocorre quando uma regra é criada.
- ✓ **Event regradeletada ()**:Ocorre quando uma regra é excluída.
- ✓ **Event regraaplicada ()**: Ocorre quando uma regra é aplicada.

Métodos

* passa pelo processo de serialização

- ✓ **Entradaefetuada:** chama a função aplicarregas (o processo de defuzzificação é o do centróide). Aciona o evento regraaplicada().
- ✓ **Executaraposload:** associa o evento entradaalterada com a rotina entradaefetuada.
- ✓ **Criargrd:** responsável por criar a grandeza. Recebe os dados passados pelo usuário relativos à grandeza, como valor máximo, mínimo, nome, tipo, unidade da grandeza e categoria. Os valores são atribuídos ao arraylist grd (), do tipo grandeza. Incrementa o variável idxmax (armazena o número de grandezas do sistema).Ativa o evento grandezacriada().
Tipo de retorno da função: grandeza.
- ✓ **Cont:** armazena o número total de memberships do sistema.
- ✓ **Differg:** esta função não permite a criação de duas grandezas com o mesmo nome. Compara-se a variável criada com as existentes no sistema, caso já exista alguma grandeza de mesmo nome, uma mensagem de erro é emitida, caso contrário, a função Criargrd é chamada.
- ✓ **Apagargrd:** função responsável por apagar a grandeza do referido índice.
Aciona o evento grandezadeletada ().
- ✓ **Editargrd:** edita os dados do objeto grandeza.
- ✓ **Retornargrd:** retorna a grandeza do referido índice.
- ✓ **Grdset:** retorna o índice da grandeza.
- ✓ **Dimensão:** Retorno o valor da variável idxmax (armazena número de grandezas existentes no sistema).
- ✓ **Criarregra:** Esta função é responsável por criar as regras do sistema.
Recebe os dados passados pelo usuário na definição da regra. Os valores são

atribuídos ao arraylist rgs (), do tipo regra, e então armazenados. Incrementa-se a variável totreg (variável que armazena o número de regras existentes).

- ✓ **Aplicarregras:** Dado o valor da entrada esta função verifica quais regras estão sendo utilizadas. Aplicam-se as regras. Calcula o peso da regra no membership de saída.
- ✓ **Apagarregra:** Esta função apaga a regra. Atualiza a variável totreg, (variável que armazena o número de regras).

A classe Grandeza é apresentada na Figura 3.4.

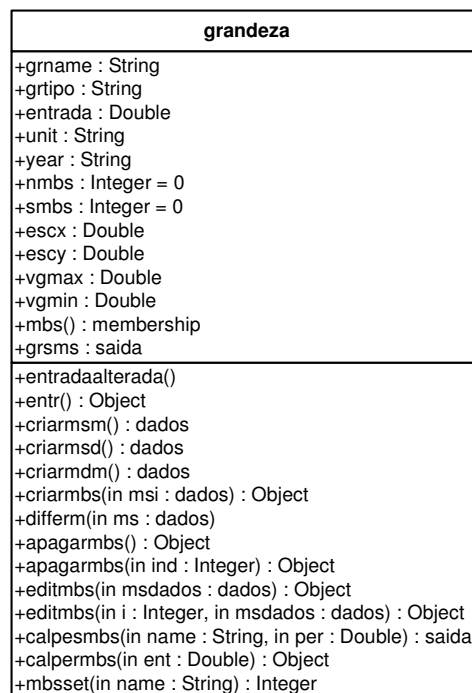


Figura 3.4 - Classe Grandeza

Os principais atributos e métodos da classe Grandeza são:

Declarações da classe:

* passa pelo processo de serialização

- ✓ **Gname**: nome da grandeza.
- ✓ **Grtipo**: tipo da grandeza (input/output).
- ✓ **Entrada**: armazena valor da entrada.
- ✓ **Unit**: unidade da grandeza.
- ✓ **Year**: categoria da grandeza.
- ✓ **Nmbs**: número de memberships.
- ✓ **Smbs**: membership selecionado.
- ✓ **Escx**: armazena valor que será usado para cálculo dos pontos dos memberships. Valor baseado na largura do componente gráfico onde são desenhados os memberships de acordo com valores Max e Min atribuídos à grandeza.
- ✓ **Escy**: armazena dados do componente gráfico.
- ✓ **Vgmax**: valor máximo atribuído à grandeza.
- ✓ **Vgmin**: valor mínimo atribuído à grandeza.
- ✓ **grsms**: variável tipo saída.
- ✓ **mbs()***: Armazena dados do membership.

Eventos:

- ✓ **Event entradaalterada ()**

Métodos

- ✓ **Entr**: Chama a função `calpermbs` atribuindo o novo valor de entrada. Valor este definido pelo usuário. Ativa o evento `entradaalterada ()`.
- ✓ **Criarmsm**: cria membership à direita.
- ✓ **Criarmsd**: cria membership triangular.
- ✓ **Criarmdm**: cria membership à esquerda.

* passa pelo processo de serialização

- ✓ **Criarmbs:** Esta função é responsável por criar o membership. Ela recebe os dados do membership que são atribuídos ao arraylist mbs, do tipo membership. Incrementa o valor nmbs.
- ✓ **Differm:** Esta função não permite a criação de memberships de mesmo nome relativos a uma determinada grandeza. Caso exista um membership de mesmo nome uma mensagem de erro é exibida, caso contrário, a função criarmbs é chamada.
- ✓ **Apagarmbs:** apaga membership.
- ✓ **Apagarmbs:** variável smbs recebe índice do membership selecionado, chama função Apagarmbs ().
- ✓ **Calpermbs:** chama função mspert(da classe membership) para cada membership de cada grandeza, passando o valor da entrada para o cálculo da pertinência.
- ✓ **Mbsset:** Retorna o número do membership referente ao nome entrado.

A classe Membership é apresentada na Figura 3.5

membership
+msper : Double
+msout : saida
+ms : dados
-msgfx : GraphicsPath
+msgraf(in escx : Double, in escy : Double, in vgmin : Double) : GraphicsPath
+mspert(in ent : Double) : Double
+mspeso(in per : Double) : saida
+mspesomax(in per : Double) : saida
+graf() : Object

Figura 3.5 - Classe Membership

Os principais atributos e métodos da classe Membership são:

Declarações da classe:

- ✓ **Msper**: pertinência do membership.
- ✓ **Msout**: variável tipo saída.
- ✓ **Ms**: variável tipo dados.
- ✓ **Msgrfx**: armazena dados do gráfico.

Métodos

- ✓ **Msgraf**: Anexa os segmentos de linha do membership (pontos x1, y1; x2, y2; x3, y3).
- ✓ **Mspert**: Calcula índice de pertinência para cada membership relacionado.
- ✓ **Mspeso**: Calcula saída (defuzzificação) pelo centróide .
- ✓ **Graf**: retorna o valor da variável msgrfx.

A classe Regra é apresentada na Figura 3.6

regra
+descricao : String
+habilitada : Boolean
+ativa : Boolean
+imbsname : ArrayList
+igrdname : ArrayList
+ombsname : String
+ogrname : String
+ogrval : Single
+ogrdpes : Single
+explicregra : String

Figura 3.6 - Classe Regra

Os principais atributos e métodos da classe Regra são:

Declarações da classe:

- ✓ **Class regras**: esta classe pode ser serializada.
- ✓ **Descricao**: armazena a nome da regra.
- ✓ **Habilitada**: verifica se a regra esta habilitada.
- ✓ **Ativa**: verifica se a regra esta ativa.

- ✓ **Imbsname** * : Armazena nomes memberships usados na geração das regras.
- ✓ **Listgdz** * : Armazena nomes grandezas usadas na geração das regras.
- ✓ **Omsname**: armazena nome membership - resposta da regra
- ✓ **Ogrdname**: armazena nome grandeza - resposta da regra.
- ✓ **Ogrdval** e **Ogrdpes**: capturam o valor e o peso da saída gerada pela regra.
- ✓ **Explicregra**: Descrição da regra.

A classe Dados é apresentada na Figura 3. 7

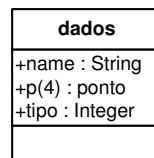


Figura 3. 7 - Classe Dados.

Os principais atributos e métodos da classe Regra são:

Declarações da classe:

- ✓ **Name**: nome do membership.
- ✓ **Structure ponto**: x e y armazenam pontos do membership.
- ✓ **Tipo**: define tipo do membership criado: 1 corresponde a left; 2, center e 3 right.
- ✓ **p(4)** * : armazena dados dos pontos do membership.

Structure Saída – usada na classe sistema e membership

- ✓ **Peso, Posx**: peso e valor da saída.

* passa pelo processo de serialização

3.3.2 Projeto

Tendo em mãos o diagrama de classe da análise do domínio do problema, é traçado como estas classes irão interagir para realizar todas as funções necessárias. Nesta fase implementa-se cada função do sistema. Modelos mais detalhados com ênfase nas soluções para armazenamento de dados, funções primordiais do sistema e interface com usuário são definidos.

O usuário será capaz de fazer a entrada dos dados das grandezas (de entrada e saída) do sistema, bem como a entrada dos campos de pertinência, referentes a cada grandeza, definindo nome, e os pontos de cada segmento de reta de cada membership.

Em um próximo passo o usuário deverá entrar com as regras, determinando o comportamento e atuação do sistema. Nesta etapa um conhecimento maior da linguagem específica do ambiente de trabalho ajuda o operador entender melhor o sistema de entrada de regras. As regras do tipo if –then são feitas escolhendo-se as grandezas de atuação com seus respectivos memberships.

Posteriormente, dar-se-á as entradas para cada grandeza, e o sistema, através de um processo de inferência, calcula automaticamente o valor de saída de acordo com as regras previamente estabelecidas. Os valores de entrada e saída podem ser visualizados no sistema.

O cálculo do valor de saída, mais conhecido como Defuzzificação foi calculado segundo o método do centróide.

O sistema gera um relatório de todos os dados do sistema, que são salvos no formato XML.

O capítulo 4 apresenta todas as telas do sistema e a interação com o usuário.

Projetos do Sistema:

Para o sistema foram desenvolvidos dois projetos. O primeiro denominado Fuzzy (biblioteca Fuzzy) contém as classes do sistema. Este funciona como uma biblioteca que possui toda a lógica necessária para implementação de quaisquer sistema que necessite usar Fuzzy (desde que contenha as referências para seu uso).

O segundo projeto denominado WFuzzy, é responsável pela interface com o usuário. Nele está presente o formulário principal do sistema, com menus direcionados às principais telas. É responsável pela criação das grandezas, criação dos memberships, simulação do sistema e geração das regras. Cabe ressaltar que o segundo projeto adicionou a referência do primeiro (biblioteca Fuzzy)

Juntos, estes dois projetos formam a ferramenta de desenvolvimento, denominada **Sistema Fuzzy**. No primeiro, é desenvolvida toda a matemática de fuzzy e no segundo toda a interface com o usuário. Usa-se esta ferramenta para entrar graficamente de forma eficaz com os dados das grandezas, memberships e regras. O capítulo 5 mostra o uso do sistema e da biblioteca com uma simulação.

A seguir uma breve descrição dos dois projetos.

- Projeto Fuzzy

Visão Geral: são criadas as classes necessárias ao sistema. Estas classes podem ser utilizadas por outros projetos. É necessário atribuir a referência aos outros projetos.

Arquitetura: O projeto foi criado utilizando a tecnologia Class Library da plataforma do desenvolvimento.

Descrição: Contém as classes necessárias para desenvolver sistemas que necessitem utilizar a lógica Fuzzy. Possibilidades: criar as grandezas, memberships e regras, definir as entradas e fazer o processo de defuzzificação.

Como mencionado, foram definidas 5 classes no sistema. No desenvolvimento foram utilizados recursos que a programação orientada a objetos oferece. Como exemplo, reuso de código. Abaixo será descrito as principais funcionalidades das classes. Os relacionamentos entre elas podem ser visto na Figura 3.2, e a descrição detalhada nas figuras Figura 3.3 a Figura 3.7.

Classe Grandeza

- ✓ Define os dados da grandeza.
- ✓ Responsável pelo armazenamento dos dados dos memberships atribuídos a cada grandeza.
- ✓ Responsável pelo cálculo da pertinência.
- ✓ Define o tipo de membership de cada grandeza (left, right, center).
- ✓ Garante que memberships da mesma grandeza tenham nomes diferentes.
- ✓ Responsável por apagar memberships da grandeza.
- ✓ Responsável por editar memberships da grandeza.

Classe Regras

- ✓ Define os dados da regra

- ✓ Responsável pelo armazenamento dos memberships e grandezas utilizadas nas regras.
- ✓ Armazena o valor e o peso da saída gerada pela regra

Classe Memberships

- ✓ Define os dados do membership.
- ✓ Responsável por desenhar os segmentos de reta dos memberships.
- ✓ Responsável pelo cálculo da pertinência de cada segmento de reta de acordo com o valor de entrada.
- ✓ Defuzzificação (Centróide).

Classe Dados

- ✓ Define os pontos de cada membership (esquerda, central e direita).

Classe Sistema

- ✓ Principal classe do sistema Fuzzy.
- ✓ Responsável pelo armazenamento: de dados das grandezas, das regras, dos dados de entrada, dentre outros.
- ✓ Verifica novos dados de entrada
- ✓ Aplica regras de acordo com o valor da entrada
- ✓ Responsável por criar a grandeza

- ✓ Controla dados da grandeza (apaga, edita, não permite a criação de grandezas de mesmo nome).
- ✓ Responsável por criar as regras e apagá-las.

Como mencionado, o sistema funciona como uma dll (Dynamic Link Library), podendo ser chamada por um ou mais aplicativos que contenham as chamadas para esta função. Para usar esta biblioteca deve-se adicionar a referência ao projeto em desenvolvimento. Declarada a variável do tipo da classe, é feito o acesso às propriedades e métodos.

Classe Sistema

Fuzzy.sistema sis ;

Campos

◆ idxmax	Representa o número total de variáveis do sistema. <i>Ex: sis.idxmax</i>
◆ idxcor	Representa o número da grandeza corrente.
◆ totreg	Representa o número total de regras no sistema.
◆ regcor	Representa o número total de regras no sistema.
◆ grd()	Arraylist que armazena dados da grandeza. Tipo da variável: "Grandeza".
◆ rgs()	Arraylist que armazena dados da regras. Tipo da variável: "Regras".
◆ datain()	Arraylist que armazena dados de entrada.
◆ namepor()	Arraylist que armazena nome do conjunto de entrada.

Eventos

⚡ grandezaeditada()	Ocorre quando se altera dados da grandeza.
⚡ grandezacrida()	Ocorre quando a grandeza é criada.
⚡ grandezadeletada()	Ocorre quando a grandeza é deletada.
⚡ regracriada()	Ocorre quando a regra é criada.
⚡ regradeletada()	Ocorre quando a regra é deletada.

 `regraaplicada()`

Ocorre quando se aplica a regra.

Métodos



`entradaefetuada`

Aciona função *Aplicarregras*. Aciona o evento *regraaplicada()*.

```
sist.entradaefetuada(  
    entradaefetuada()
```



`executaraposload`

Associa o evento *Entradaalterada* com a rotina *Entradaefetuada*.

```
sist.executaraposload(  
    executaraposload()
```



`criargrd`

Adiciona Grandeza

```
sist.criargrd(  
    criargrd(vmin As Double, vmax As Double, name As String, tipo As String, unit As String, year As String) As fuzzy.grandeza
```

- `vmin`: valor mínimo da variável.
- `vmax`: valor máximo da variável.
- `name`: nome da variável.
- `tipo`: “Input” ou “Output”.
- `unit`: unidade da grandeza.
- `year`: categoria da grandeza



`cont`

Armazena o número total de memberships do sistema.

```
sist.cont(  
    cont() As Integer
```



`differg`

Não permite a criação de grandezas de mesmo nome. Chama a função *Criargrd*.

```
sist.differg(  
    differg(min As Double, max As Double, nome As String, tip As String, uni As String, year As String) As Object
```

- `min`: valor mínimo da variável.
- `max`: valor máximo da variável.
- `nome`: nome da variável.
- `tip`: “Input” ou “Output”.
- `uni`: unidade da grandeza.
- `year`: categoria da grandeza



`apagargrd`

Apaga a grandeza do referido índice.

```
sist.apagargrd(
  apagargrd (grindice As Integer) As Object
```

- grindice: índice da grandeza. Índice iniciado em 1, este segue uma ordem crescente de acordo com a criação da grandeza.

editargrd

Edita Grandeza

```
sist.editargrd(
  editargrd (grindice As Integer, vmin As Double, vmax As Double, name As String, tipo As String, unit As String) As Object
```

- min: valor mínimo da variável.
- max: valor máximo da variável.
- nome: nome da variável.
- tip: “Input” ou “Output”.
- uni: unidade da grandeza.
- year: categoria da grandeza

retornargrd

Retorna a grandeza do referido índice.

```
sist.retornargrd(
  ▲ 1 of 2 ▼ retornargrd (indice As Integer) As fuzzy.grandeza
```

- indice: índice da grandeza.

Retorna o índice da referida grandeza.

```
sist.retornargrd(
  ▲ 2 of 2 ▼ retornargrd (nome As String) As fuzzy.grandeza
```

- nome: nome da grandeza Ex: “Temperatura”.

Atribuir valor da entrada

```
sist.retornargrd(X).entr = 10
```

- X = indice: índice da grandeza.
- X= nome: nome da grandeza Ex: “Temperatura”.

grdset

Retorna o índice da grandeza

```
sist.grdset(
  grdset (nome As String) As Integer
```

- nome: nome da grandeza.

criarregra

Adiciona Regra

sist.criarregra()

criarregra (des As String, ig As System.Collections.ArrayList, im As System.Collections.ArrayList, og As String, om As String, exprega As String) As fuzzy.regas

- des: nome da regra.
- ig: grandezas utilizadas na regra.
- im: memberships utilizados na regra.
- og: grandeza resposta da regra.
- om: membership resposta da regra.

Calcula peso e valor de saída.

aplicarregras

sist.aplicarregras()

aplicarregras (flag As Integer) As Object

- flag = 0 indica que o processo de Defuzzificação é o do Centróide.

Apagarregra

Apaga a grandeza do referido índice

sist.apagargrd()

apagargrd (grindice As Integer) As Object

- grindice = índice da grandeza.

Propriedades

sist.dimensao()

Retorna o numero de grandezas do sistema

Classe Grandeza

grd fuzzy.grandeza


Campos

Grtipo	Representa o tipo da grandeza.
Grname	Representa o nome da grandeza.
entrada	Representa o valor de entrada do sistema.
unit	Representa a unidade da grandeza.
year	Representa a categoria da grandeza.
nmbs	Representa o número de membeships
smbs	Representa o membership selecionado
escx	Armazena valor para membership
escy	Armazena valor para membership
vgmax	Representa o valor máximo atribuído a grandeza
Vgmin	Representa o valor mínimo atribuído a grandeza

 `mbs`


Arraylist que armazena dados do membership

Eventos

 `entradaalterada()`

Ocorre quando se efetua a entrada .

Métodos

 `Criarmsm`


Cria membership a direita.

```
grd.criarmsm(  
    criarmsm () As fuzzy.dados)
```

Ou

```
sist.grd(indice).criarmsm
```

- `indice`: índice da grandeza desejada.

 `Criarmsd`


Cria membership triangular.

```
grd.criarmsd(  
    criarmsd () As fuzzy.dados)
```

Ou

```
sist.grd(indice).criarmsd
```

- `indice`: índice da grandeza desejada.

 `Criarmdm`


Cria membership a esquerda.

```
grd.criarmdm(  
    criarmdm () As fuzzy.dados)
```

Ou

```
sist.grd(indice).criarmdm
```

- `indice`: índice da grandeza desejada.

 `Criarmbs`


Atribui ao arraylist `mbs` os dados do membership criado.

```
grd.criarmbs(  
    criarmbs (msi As fuzzy.dados) As Object)
```

Ou

```
sist.grd(indice).criarmbs
```

- `indice`: índice da grandeza desejada.

 `Differm`

Não permite a criação de memberships de mesmo nome, atribuídos a mesma grandeza. Chama a função `Criarmbs`.

Deve-se criar uma variável do tipo dados e atribuir um nome ao membership. Ex: `variaveldados.name = "alto"`

```
grd.differm(  
    differm (ms As fuzzy.dados))
```

Ou

```
sist.grd(indice).differm
```

- `indice`: índice da grandeza desejada.

 `Apagarmbs`

Atualiza memberships do arraylist `mbs()` após membership deletado.

```
grd.apagarmbs (
```

```
▲ 1 of 2 ▼ apagarmbs () As Object
```

Apaga membership selecionado. Chama a função *apagarmbs* .

```
grd.apagarmbs (
```


```
▲ 2 of 2 ▼ apagarmbs (ind As Integer) As Object
```

- ind: índice do membership.

Ou

```
sist.grd(indice).apagarmbs(ind)
```

- indice: índice da grandeza desejada.
- ind: índice do membership desejado.

 Calpesmbs

Calcula pertinência dos memberships. Acionado pela propriedade *Entr()*

```
calpermbs(ent as double)
```

- ent: valor da entrada.

 Mbsset

Retorna o número do membership referente ao nome entrado

```
grd.mbsset (
```

```
mbssset (name As String) As Integer
```


- name: nome do membership

Ou

```
sist.grd(indice).mbsset(nome)
```

- indice: índice da grandeza desejada.
- nome: nome do membership da grandeza.

Propriedades

 Entr ()

Atribui o valor de entrada recebido. Aciona a função *calpermbs* passando o valor da entrada.

Classe Membership

Campos

 Mspcr

Representa o índice de pertinência do membership.

 Msout

Representa a variável do tipo saída.

 Ms

Representa a variável do tipo dados.

 Msgrfx

Representa uma série de linhas e curvas conectadas. Armazena dados do gráfico.

Métodos



Representa os segmentos de linha do membership (pontos $x_1, y_1; x_2, y_2; x_3, y_3$).

Msgraf

```
grd.mbs(i).msgraf(
```

```
msgraf(escx As Double, escy As Double, vgmim As Double) As System.Drawing.Drawing2D.GraphicsPath
```

- i : índice do membership da grandeza.
- $escx$: valor que será usado para cálculo dos pontos dos memberships de acordo com dados do componente gráfico.
- $escy$: armazena dados do componente gráfico do membership.
- $Vgmim$: valor mínimo atribuído à grandeza.



Calcula pertinência dos memberships.

Mspert

Acionado pela função *calpermbs*

```
mbs(i).mspert(ent)
```

- i : unidade do membership.
- ent : valor entrada.



Calculo Defuzzificação. O calculo é feito pelo centróide.

Mspeso

Acionado pela função *Aplicaregra*.

Propriedades



Graf

Retorna dados do gráfico.

Classe Membership

Campos

Descricao	Representa o nome da regra.
Habilitada	Habilita regra ao ser criada. Variável recebe “true”.
Ativa	Representa a regra utilizada.
imbsname	Representa o arraylist de memberships utilizados na regra.
igrdname	Representa o arraylist de grandezas utilizadas na regra.
Ombsname	Representa o membership do final da regra.
Ogrdname	Representa o membership do final da regra.
Ogrdval	Representa o valor da saída gerada pela regra.
Ogrdpes	Representa o peso gerado pela regra.
Explicregra	Representa a descrição da regra.

Classe Dados

Campos

◆ Ponto	Representa a estrutura x,y usada para o membership
◆ p(4)	Representa o arraylist dos pontos do membership. Tipo: ponto.
◆ Tipo	Define o tipo do membership criado. “1” - membership esquerda . “2” - membership central. “3” - membership direita .

Estrutura Saída

◆ saída	Representa a estrutura peso, posx usado na regra Peso: valor do peso. Posx: valor da saída.
---------	---

• Projeto Wfuzzy

Visão Geral: Projeto responsável pela interface com o usuário. Cria a interface de criação das grandezas, memberships, da simulação do sistema, e geração das regras.

Arquitetura: O projeto foi criado utilizando a tecnologia windows application da plataforma utilizada.

Descrição: Contém os formulários e controles responsáveis pela interação com o usuário como o **Sistema Fuzzy**.

Principais formulários do sistema

- ✓ Sistema Fuzzy: formulário principal. Oferece menus para as funcionalidades do sistema. Importa a classe *Fuzzy.Sistema* definida no projeto 1, fornecendo assim dados da mesma. No sistema este formulário é chamado de Fuzzy.
- ✓ AddGrandezas: responsável pela criação das grandezas. Nele o usuário entra com nome da grandeza, unidade, valor máximo, mínimo, e o tipo da mesma (input/output). No sistema chamado de Add Variable.

- ✓ Grandezas: menu para criação das grandezas, criação dos membership's atribuídos às respectivas grandezas.
- ✓ Visualiza o controle *cmbsp* responsável pela alteração do nome e dados do membership. É possível também neste formulário alterar dados da grandeza, como nome valor máximo, mínimo, tipo, unidade. No sistema chamado de Variables Definition.
- ✓ Simulação: simulação do sistema. Efetua-se o valor de entrada, podendo assim observar as regras que são usadas e as que não são aplicáveis. Verifica-se o valor de saída. No sistema chamado System Simulation.
- ✓ Gerregras: permitir ao usuário visualizar todas as regras existentes no sistema, podendo ainda deletar ou editar o nome da regra desejada. Caminho para adição de novas regras. No sistema chamado Rules Generation.
- ✓ Regras: responsável pela geração das regras. No sistema chamado Rules Definition.
- ✓ AddMembership: recebe o nome do membership, atribuído pelo usuário e aciona o controle *cmbsp*. Chamado Add Membership.
- ✓ Cmbsp: tem a função de criar os memberships, responsável por suas propriedades, dados, desenho. É neste controle que o usuário altera os dados dos campos de pertinência, atribuindo novos valores. Por se tratar de um controle é adicionado no formulário Grandezas.
- ✓ Case: Podemos definir as grandezas por categorias. Esta tela é responsável pela adição das categorias. Chamada no sistema de Category.
- ✓ Repor: Relatório dos dados. Chamada Report.

As figuras e descrições das telas estão no capítulo 4.

4 Descrição do Sistema

Este tópico tem como objetivo apresentar as telas do sistema e suas funcionalidades.

4.1 Telas do Sistema

Neste tópico são apresentadas todas as telas do sistema. Nos próximos cada tela é especificada .

- ✓ Tela Fuzzy
- ✓ Tela Category
- ✓ Tela Variables Definition
- ✓ Tela Rules Generation
- ✓ Tela Rules Definition
- ✓ Tela Add Variable
- ✓ Tela Add Membership
- ✓ Tela System Simulation
- ✓ Tela Analysis
- ✓ Tela Data
- ✓ Tela Report
- ✓ Tela Export
- ✓ Tela Edit Name Rule
- ✓ Tela Save, Save As.
- ✓ Tela Open

4.1.1 Tela Fuzzy – Tela Principal

Objetivo: Tela principal do sistema. Composta de quatro menus: File, Fuzzy, Window e Help, respectivamente com seus sub-menus, disponibilizam recursos do sistema. A Figura 4 1mostra a tela Fuzzy.

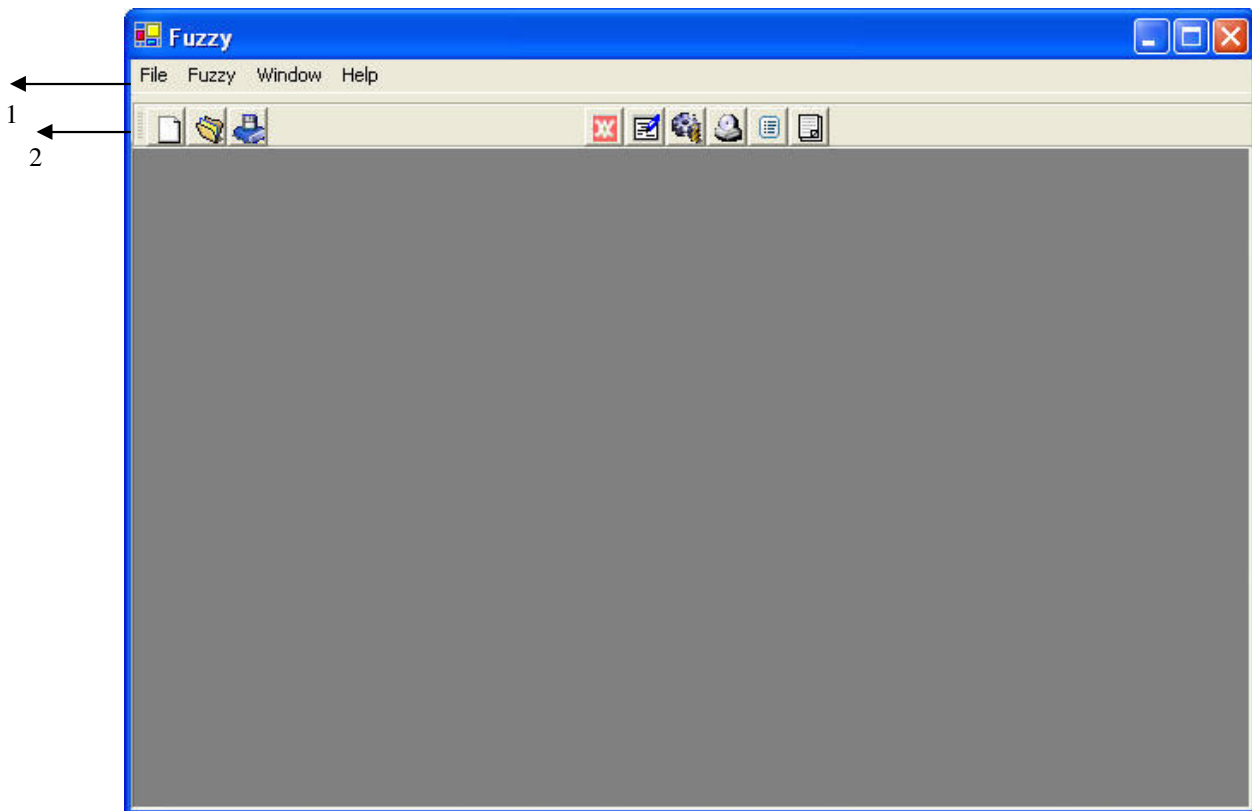


Figura 4 1 - Fuzzy System

Campos:

1- Menus:

File: Ao clicar serão exibidos sub-menus.

Fuzzy: Ao clicar serão exibidos sub-menus.

Window: Opções de formato das janelas.

Help: Ao clicar serão exibidos sub-menus.

2- Atalhos:



Botão **New** : Inicia um novo projeto.



Botão **Open:** Abre projeto



Botão **Save:** Salva o alterações do projeto corrente.



Botão **Variables Definition:** Abre a tela **Variable Definition** .



Botão **Rules Gerneration:** Abre tela **Rules Generation**



Botão **System Simulation:** Abre a tela **System Simulation**



Botão **Data Fuzzy System:** Abre a tela **Project** .



Botão **Data Analysis:** Abre a tela **Analysis** .



Botão **Report:** Abre a tela **Report** .

4.1.2 Menu File

Objetivo: Permite iniciar um novo sistema, abrir, fechar ou salvar arquivos referentes ao sistema. A Figura 4.2 apresenta o menu “File”

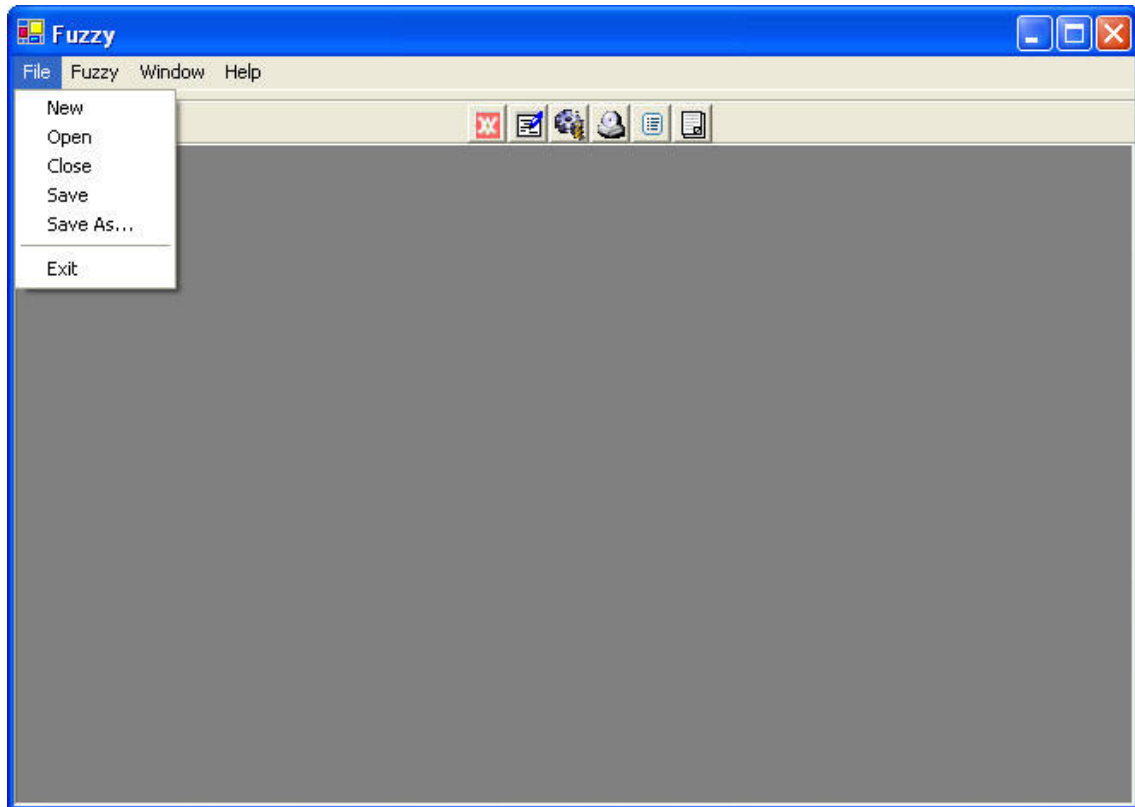


Figura 4.2 - Menu File

Campos:

New: Inicia um novo projeto.

Open: Abre um projeto. Ao clicar neste campo uma nova tela será exibida, onde é possível localizar algum projeto.

Close: Fecha o projeto corrente.

Save / Save As: Salva as alterações do projeto corrente.

Exit: Sai do programa Fuzzy.

4.1.3 Menu Fuzzy

Objetivo: Cada sub-menu leva a outra tela, possibilitando: a criação das grandezas e memberships, criação de regras, efetuar entradas, simular, visualizar e analisar dados, geração de relatório, importar , exportar dados. O menu “Fuzzy ” é apresentado Figura 4.3.

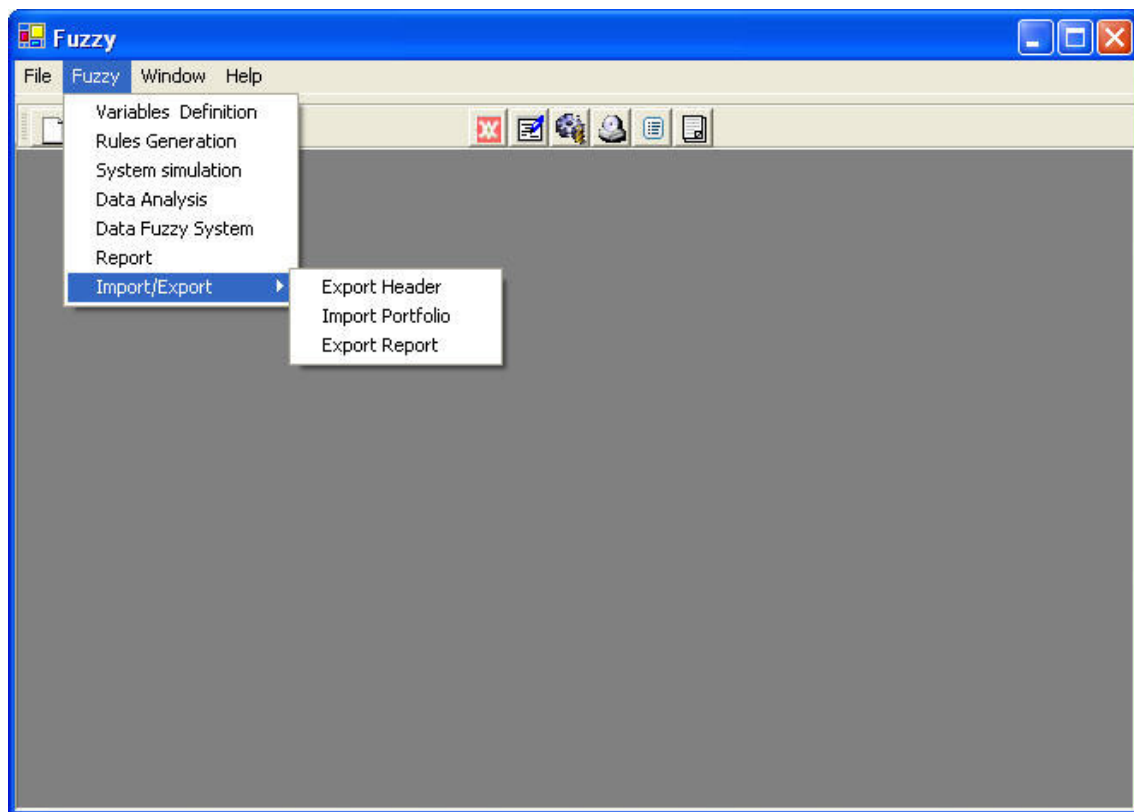


Figura 4.3 - Menu System

Campos:

Variable Definition: Abre a tela **Variable Definition**, onde são definidas as métricas, memberships do sistema.

Rules Generation: Abre tela **Rules Generation**, onde são exibidas as regras existentes. Caminho para adição de novas regras.

System Simulation: Abre a tela **System Simulation** .

Data Analysis: Abre a tela **Analysis**, permitindo simulação e análise de dados.

Data Fuzzy System: Abre a tela **Project Data**, exibe status do sistema.

Report: Abre a tela **Report**, responsável pelo relatório geral do sistema.

Import/Export:

- **Export Header:** Exporta o cabeçalho das variáveis definidas no sistema para uma planilha que deverá ser preenchida pelo usuário.
- **Import Portfolio:** Importa dados de uma planilha Excel, ao clicar neste sub-menu abre-se a tela Abrir para escolha da planilha. Estes dados servirão como entrada para as variáveis do sistema. A planilha deve seguir um formato padrão.

4.1.4 Menu Help

Objetivo: Contem a chamada para os tópicos de ajuda do sistema. O menu “Help” é apresentado na Figura 4.4.

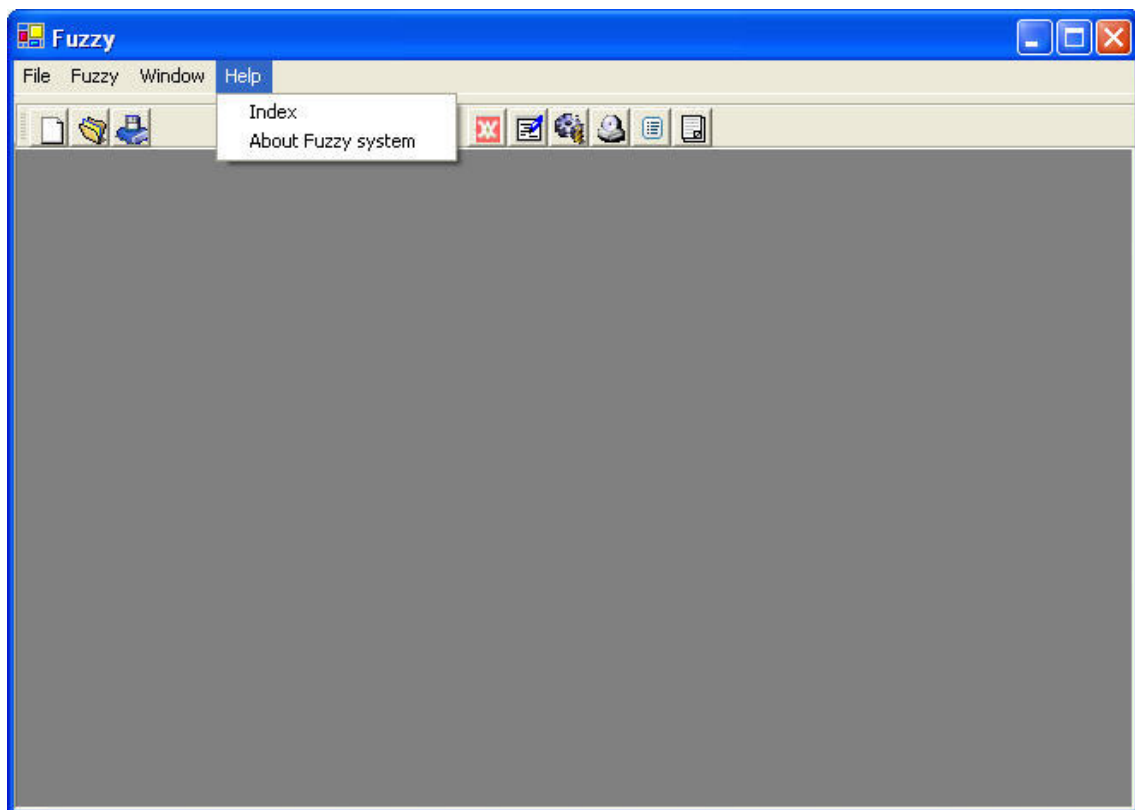


Figura 4.4 – Menu Help

Campos:

Index: help do sistema.

About Fuzzy System: tela sobre o sistema.

4.1.5 Tela Category

Visão Simplificada

Objetivo: Definir as categorias do sistema, como por exemplo, categorias ano1, ano2, etc. A Figura 4.5 apresenta a tela Category.

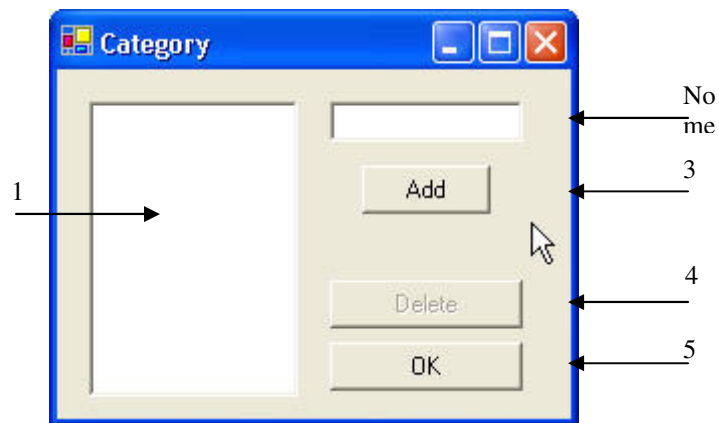


Figura 4.5- Category

Campos:

- 1- Lista as categorias existentes.
- 2- Campo para entrada da categoria.
- 3- **Botão Add:** Insere na lista (campo1) a categoria.
- 4- **Botão Delete:** Deleta a categoria selecionada no campo 1.
- 5- **Botão OK:** Insere lista de categoria no sistema. Fecha a tela.

4.1.6 Tela Variables Definition

Visão Simplificada

Objetivo: Criar variável (métrica) e memberships, alterar dados dos campos de pertinência (membership's), alterar dados da grandeza. A Figura 4.6 apresenta a tela Variables Definition

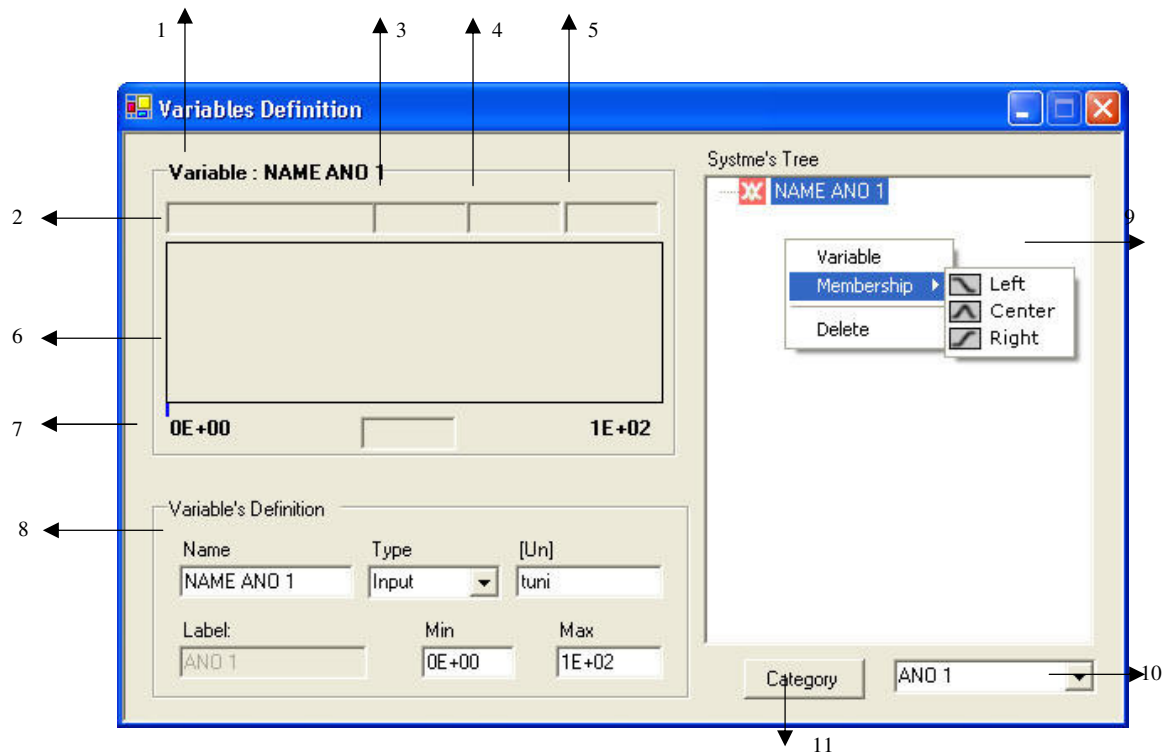


Figura 4.6 - Variables Definition

Campos:

1-Variable Name: Nome da variável ativa é exibido.

2- Exibe Nome do membership ativo.

3-4-5 - Exibe dados do membership selecionado.

6- Exibe os memberships atribuídos à variável (métrica).

7- Exibe valor mínimo e máximo da grandeza. (Valores definidos ao se criar a grandeza).

8- Variables Definition

Name: Exibe o nome atribuído à respectiva grandeza (variável).

Type: Exibe o tipo da grandeza (Input, Output).

[Un]: unidade da grandeza.

Label: Categoria da variável selecionada.

Min: Valor mínimo estipulado que a variável pode assumir.

Max: Valor máximo estipulado que a variável pode assumir.

9- System's Tree

Variable: Cria a grandeza. Ao clicar uma nova tela será exibida. Nela atribui-se o nome, tipo, valores mínimo e máximo da respectiva variável.

Membership: Criar Membership. Escolhe-se a posição (left, center, right).

Tela **Add Membership** será aberta, nesta tela atribui-se um nome para o membership.

Delete: Deleta a grandeza, deleta membership.

10- Lista as Categorias existentes

11- Botão Category : Abre a tela **Case** , permita inserir ou deletar categorias.

4.1.7 Tela Rules Generation

Visão Simplificada

Objetivo: Exibir as regras existentes, deletar regra, caminho para adição de novas regras. A Figura 4.7 apresenta a respectiva tela.

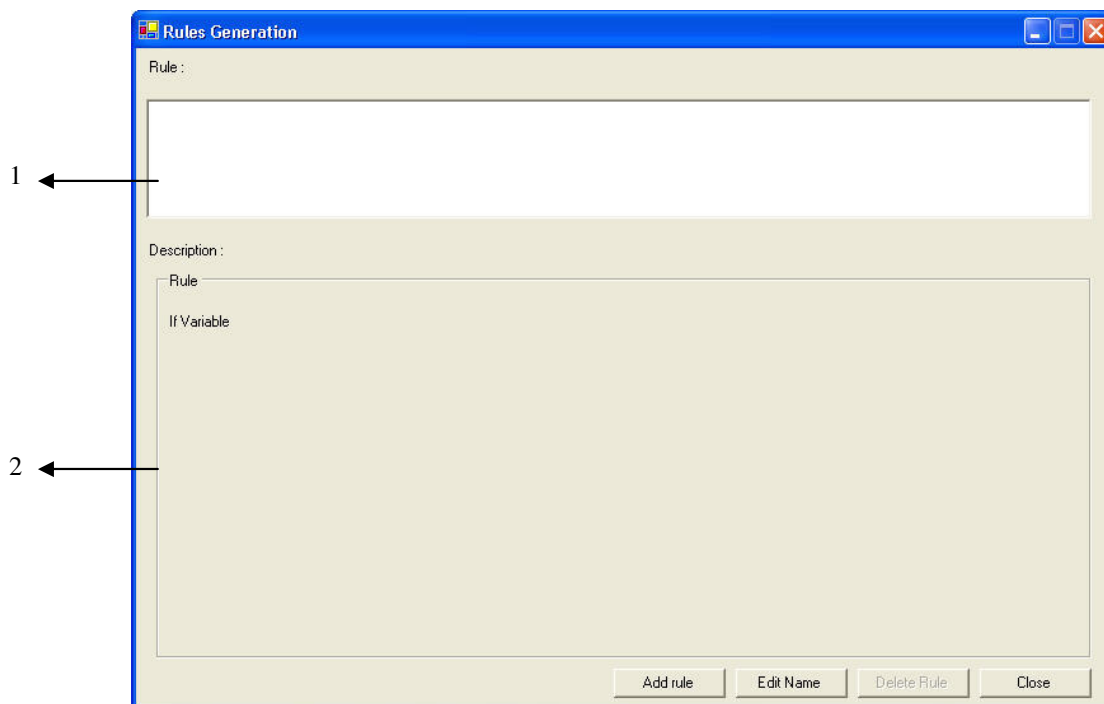


Figura 4.7 - Rules Generation

Campos:

- 1- Rule:** Exibe as regras existentes com seus respectivos nomes.
- 2- Description :** Exibe descrição da regra selecionada.

Botões:

Add Rule: Abre a tela **Rules Definition** para adição de novas regras.

Edit Name: Edita nome da regra selecionada .

Delete Rule: Deleta a regra selecionada.

4.1.8 Tela Rules Definition

Visão Simplificada

Objetivo: Definir as regras do sistema. A Figura 4.8 mostra a tela Rules Definition.

The screenshot shows a window titled "Rules Definition" with a blue title bar. It contains the following elements:

- Rule Index:** A text input field containing the number "1".
- Rule Name:** An empty text input field.
- Rule Description:** A large empty text area.
- Rules:** A section containing two side-by-side text input fields:
 - If input value of Variable (...)=** (Field 1)
 - Membership (...)** (Field 2)Below these fields are four buttons: AND, THEN, ENTER, and OK.
- Rule Context:** A section containing a text input field labeled "If Variable" (Field 3).
- Back:** A button at the bottom right of the window.

Arrows labeled 1, 2, and 3 point to the respective input fields.

Figura 4.8 - Rule Definition

Campos:

Rule Index: Índice da regra; indexada por uma ordem específica.

Rule Name: Nome da regra.

Rule Description: Descrição da regra.

Rules:

1- (If input value of variable): Exibe os nomes das variáveis atribuídas às respectivas grandezas.

2- (membership): Exibe os memberships da grandeza selecionada.

3- (Rule Contex): Exibe a regra à medida que é formulada.

Botões:

AND: Operador de condição para definição das regras. Se houver mais de uma condição deve-se acionar o operador.

THEN: Operador de condição para definição das regras.

ENTER: Conclui a geração da regra.

OK: Salva a regra gerada. Fecha a tela.

BACK: volta ao estado anterior da descrição da regra.

4.1.9 Tela Add Variable

Visão Simplificada

Objetivo: Adicionar grandeza. A Figura 4. 9 mostra a tela.

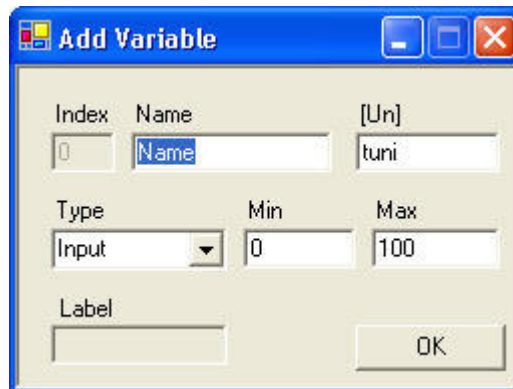


Figura 4. 9 - AddVariable

Campos:

Index: Índice da variável; indexada por uma ordem específica.

Name: Nome da variável atribuída à respectiva grandeza.

[Un]: Unidade da grandeza.

Type: Define tipo da grandeza (Input/ Output).

Min: Estipula o valor mínimo que a variável pode assumir.

Max: Estipula o valor máximo que a variável pode assumir.

Label : Nome da categoria que vai ser concatenado ao nome da variável.

Botão OK: Adição da variável.

4.1.10 Tela Add Membership

Visão Simplificada

Objetivo: Adicionar membership. A Figura 4.10 apresenta a tela Add Membership.

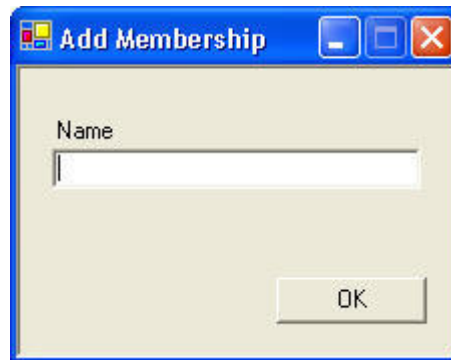


Figura 4.10 - Add Membership

Campos:

Name: Atribuir nome ao membership.

Botão OK: Cria o membership.

4.1.11 Tela System Simulation

Visão Simplificada

Objetivo: Efetuar valor de entrada, verificar valor de saída (defuzzificação). A Figura 4.11 apresenta a tela System Simulation.

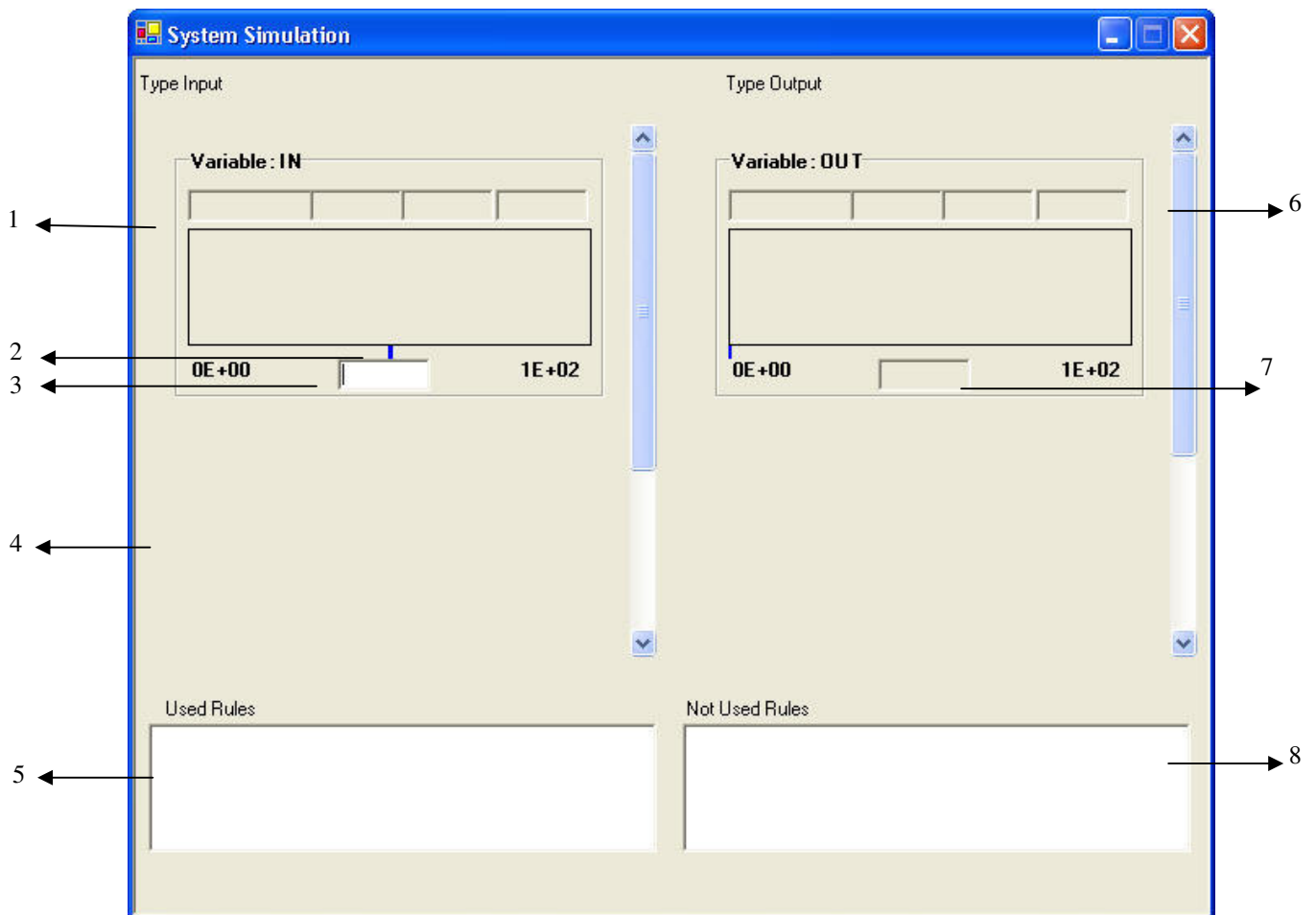


Figura 4.11 - System Simulation

Campos:

- 1- Exibe dados da variável (com seus membership's) de entrada.
- 2- Define o valor da entrada. Pode ser feita pelo usuário.
- 3- Exibe valor Entrada.

4- Exibe outras grandezas (com seus membership's) de entrada.

(idem 1-2-3.).

5- Exibe as regras que estão sendo usadas.

6- Exibe dados da variável (com seus membership's) de saída.

7- Exibe o valor de saída (defuzzificação).

4.1.12 Tela Analysis

Visão Simplificada

Objetivo: Exibir (entrar com) valores de entrada, verificar valores de saída (defuzzificação). A Figura 4 12apresenta a tela Analysis.

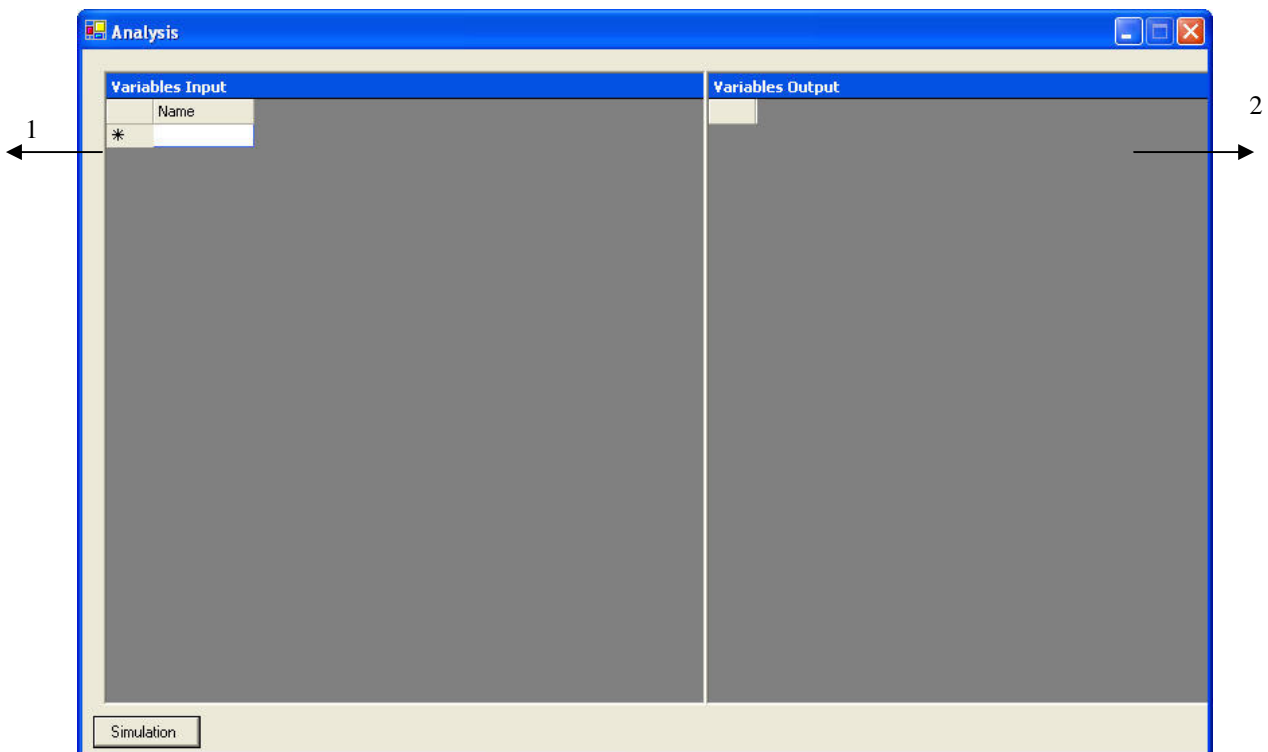


Figura 4 12 - Analysis

Campos:

1- Exibe todas as variáveis de entrada, com seus valores de entrada.

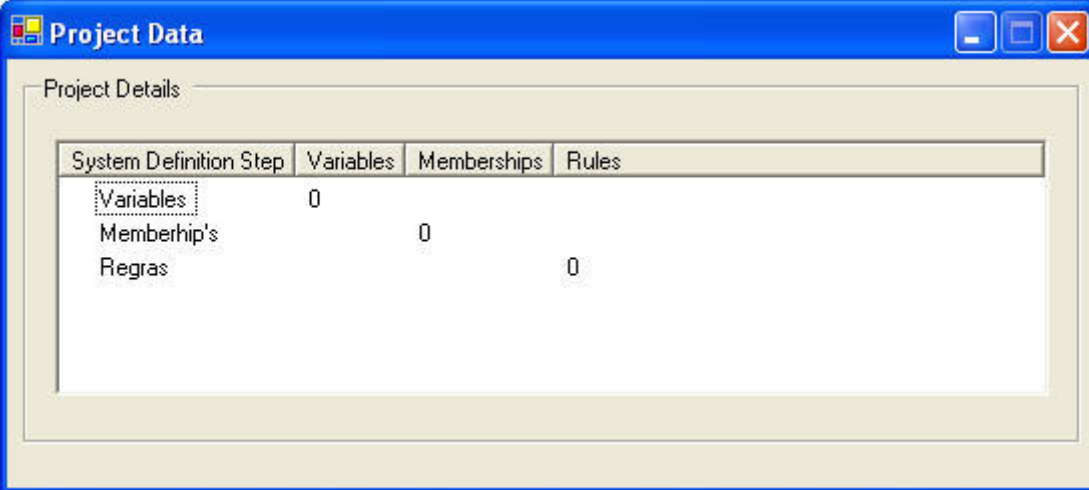
2- Exibe todas as variáveis de saída, com os resultados da análise.

Botão Simulation : Simula nova análise (não salva nova análise).

4.1.13 Tela Project Data

Visão Simplificada

Objetivo: Exibir Status do sistema, mostrando numero de variáveis, memberships e regras. A Figura 4.13 mostra a tela Project Data.



System Definition Step	Variables	Memberships	Rules
Variables	0		
Memberhip's		0	
Regras			0

Figura 4.13 - Data

Campos:

1- Variables / Memberships /Regras Exibe total de variáveis, memberships e regras do sistema.

4.1.14 Tela Report

Visão Simplificada

Objetivo: Exibir relatório do sistema, como variáveis com seus memberships, regras, dados da simulação, e visualização dos membeships.

A Figura 4.14 mostra a tela.

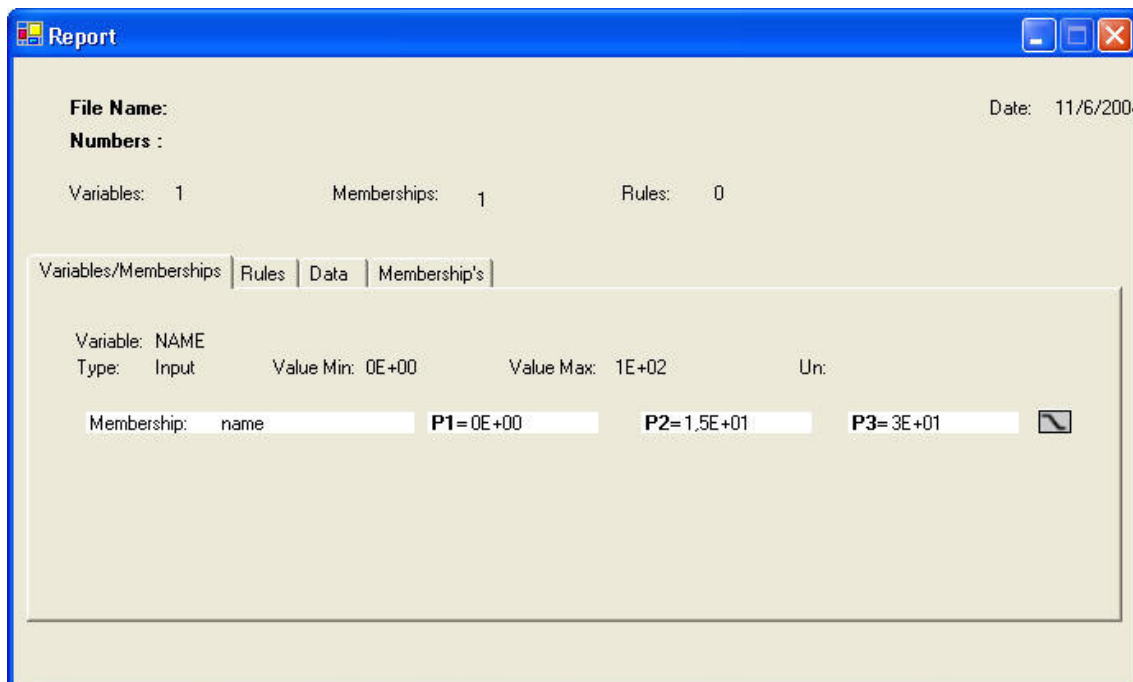


Figura 4.14 - Report

Campos:

File name: Exibe nome do projeto corrente.

Date: Data de geração do relatório.




Numbers:

Variables: Número de variáveis do projeto.

Memberships: Número de memberships do projeto.

Regras: Número de regras do projeto

Abas:

Variables/Memberships: Exibe dados da variável (nome, valor máximo, valor mínimo, tipo, unidade)/ Exibe dados do membership (nome, valores), e desenho do tipo (left , center , right ).

Rules: Exibe descrição das regras.

Data: Exibe valores entrada e resultado do sistema.

Memberships: Exibem de forma gráfica todos os memberships.

4.1.15 Tela Alarm Configurations

Visão Simplificada

Objetivo: Gerar relatório geral/Exportar dados para o Excel. A Figura 4.15 apresenta a tela Alarm Configurations.

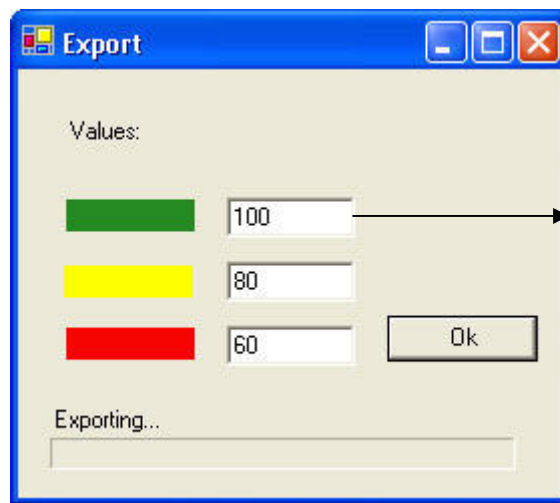


Figura 4.15 - Export

Campos:

- 1- Valores máximos para a aceitação (verde, amarelo, vermelho). Estes valores podem ser alterados pelo usuário.
- 2- **Botão OK:** Exporta dados do sistema para o Excel.

4.1.16 Tela Edit Name - Rule

Visão Simplificada

Objetivo: Editar Nome da Regra. A Figura 4.16 mostra a tela Edit Name.



Figura 4.16 - Edit Rule

Campos:

New name: Atribuir novo nome a regra selecionada.

Botões:

OK: Confirma alteração.

Cancel: Cancela operação.

4.1.17 Tela Salvar

Visão Simplificada

Objetivo: Salvar o projeto. A Figura 4.17 exibe a tela Salvar.

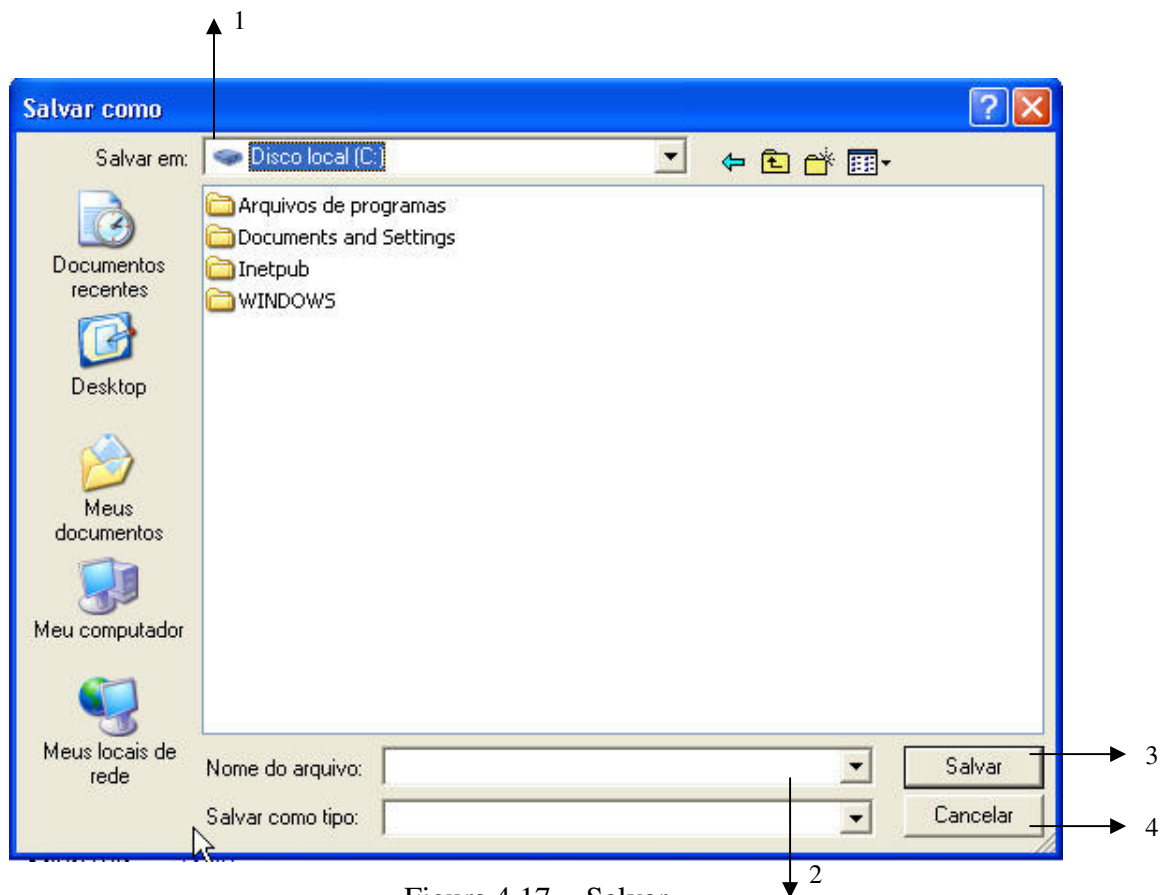


Figura 4.17 - Salvar

Campos:

- 1- Caminho para salvar o projeto.
- 2- **Nome do arquivo:** Nome dado ao arquivo.

Botões:

- Salvar:** Salva o projeto
- Cancelar:** Cancela a operação

4.1.18 Tela Abrir

Visão Simplificada

Objetivo: Abrir um projeto existente. A

Figura 4.18 exibe a tela Abrir.

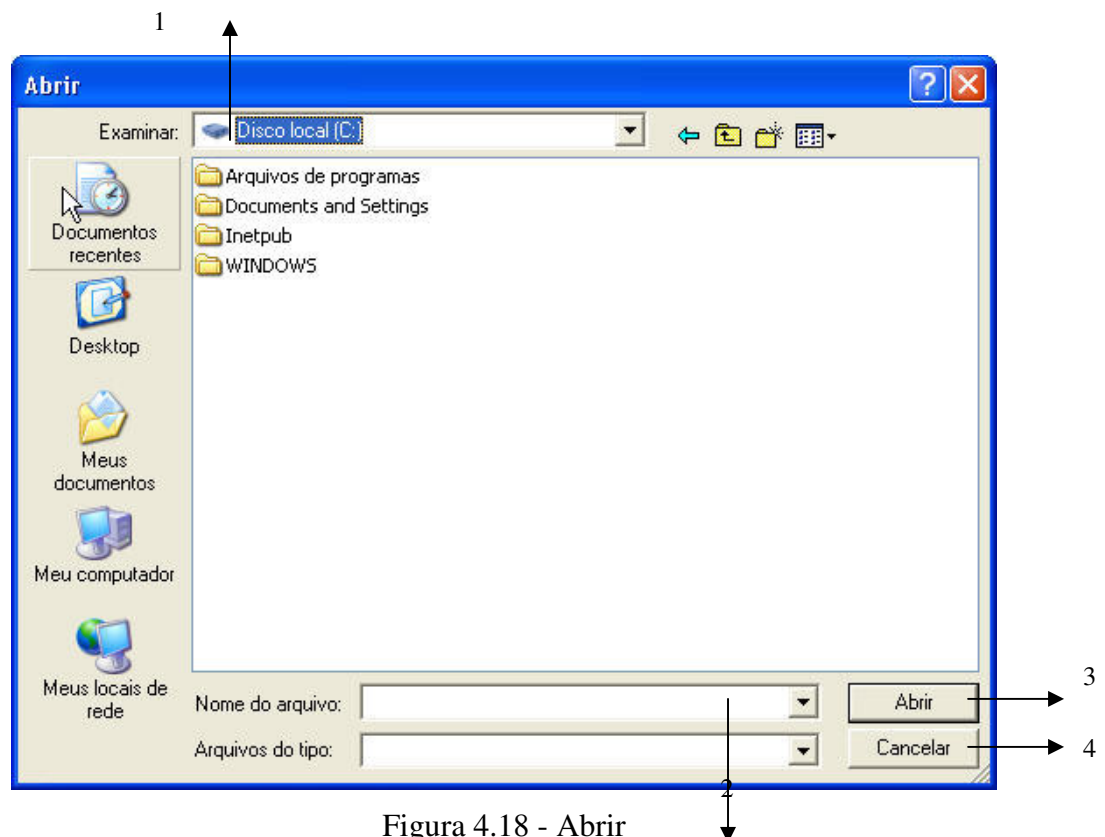


Figura 4.18 - Abrir

Campos:

- 1- Escolha do diretório onde se encontra o arquivo.
- 2- Exibe nome do arquivo.

Botões:

Abrir: Abre o projeto selecionado.

Cancelar: Cancela a operação.

5 Simulações

Este tópico tem como objetivo mostrar o uso da ferramenta, o uso da biblioteca Fuzzy e exibir resultados através de simulações.

Nos capítulos anteriores apresentou-se a ferramenta para o uso da lógica Fuzzy. Os métodos de fuzzificação, defuzzificação foram exibidos, também a biblioteca gerada para o uso da lógica; além da interface do sistema. Neste capítulo é apresentada uma aplicação para o uso da ferramenta e os resultados obtidos.

A aplicação foi desenvolvida, através de um programa escrito em linguagem C# (Sharp). Nela utilizou-se a biblioteca Fuzzy (descrita no projeto 1, capítulo 3) aproveitando as funções que disponibiliza. O exemplo é um jogo de nave espacial em 2D cuja inteligência é toda baseada em lógica Fuzzy, denominado Collision Game.

Foram realizadas três simulações. Cada uma possui um conjunto de regras, e grandezas com diferentes funções de pertinência. A representação gráfica através do Sistema Fuzzy é apresentada na primeira simulação, também são exibidos os recursos da ferramenta.

Modelagem do sistema Fuzzy

O sistema que controla a inteligência artificial do jogo é composto por quatro variáveis de entrada e uma de saída. As variáveis de entrada dividem-se em duas categorias. Categoria “MET” e “ALVO”. A variável de saída está em uma terceira categoria, denominada “OUT”.

As variáveis da categoria “MET” são: Direção_met e Distância_met. A primeira indica a direção do meteoro, a segunda indica a distância com a nave.. As variáveis da categoria “ALVO” são: Direção_alvo e Distancia_alvo, por fim da categoria “OUT” tem-se a variável Saida_out, responsável pela resposta do sistema.

Como mencionado, é no sistema Fuzzy que entramos com as grandezas, memberships e regras de controle. Empregamos este sistema para entrar graficamente de forma eficaz com os dados necessários. Este, gera um arquivo de dados com as variáveis e regras.

O arquivo é importado pela aplicação Collision Game, a qual emprega a biblioteca fuzzy utilizando suas funções para controlar o jogo.

A seguir apresentar-se-á os passos que devem ser realizados no sistema Fuzzy. Também serão ilustradas todas as possibilidades do Sistema.

- Passos que devem ser realizados no Sistema Fuzzy:
 1. Criar novo sistema.
 2. Definir categoria.
 3. Definir variáveis de entrada, saída, determinando valor máximo, mínimo, tipo e unidade destas grandezas.
 4. Definir memberships de cada variável.
 5. Definir regras de atuação.

Dados do Exemplo para primeira simulação.

Os nomes das categorias (MET, ALVO, OUT) aparecem concatenados às das variáveis para melhor compreensão.

Entrada de dados:

- ✓ Grandezas de atuação (variáveis do tipo input e output). Valores Máximos e Mínimos de cada variável.
- ✓ Membership's de cada grandeza.
- ✓ Regras de atuação.

5.1 Iniciar Sistema

- ✓ Abrir o Sistema Fuzzy

Passos

Para iniciar um novo projeto

- ✓ Na tela principal (Fuzzy) clique: menu **File** / New. Conforme ilustra Figura 5.1.

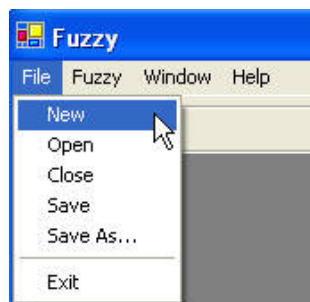


Figura 5.1 - Novo Sistema

A tela “Variables Definition”, será exibida, como ilustra a Figura 5.2.

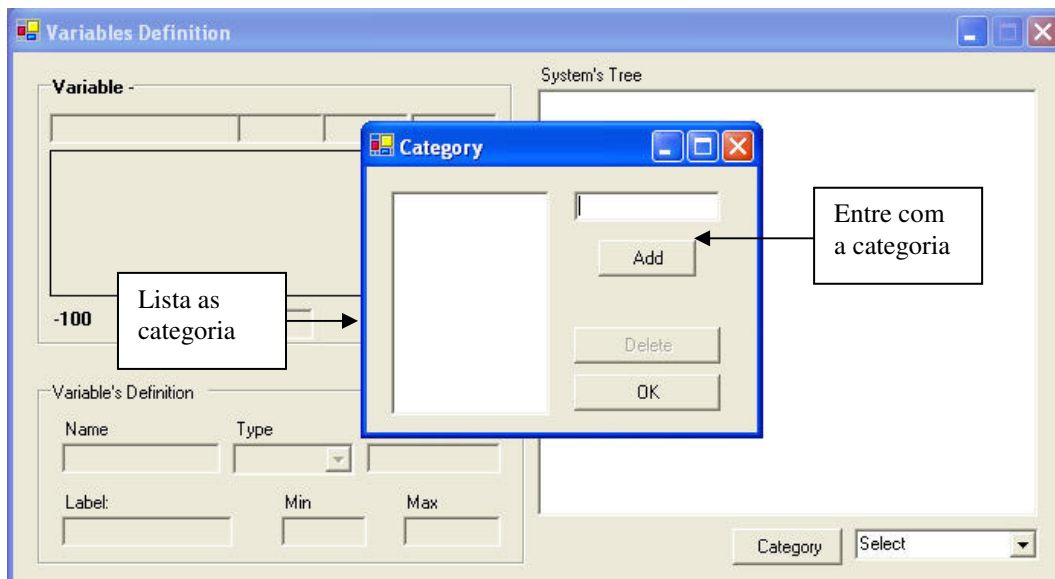


Figura 5.2 - Tela Variables Definition

5.2 Definir Categorias

- ✓ Digite o nome da categoria no campo indicado na Figura 5.2.
- ✓ Clique no botão **Add**.
- ✓ Repita para mais categorias.
- ✓ Ao término, clique no botão **OK**.

A figura Figura 5.3 exibe a lista de Categorias.

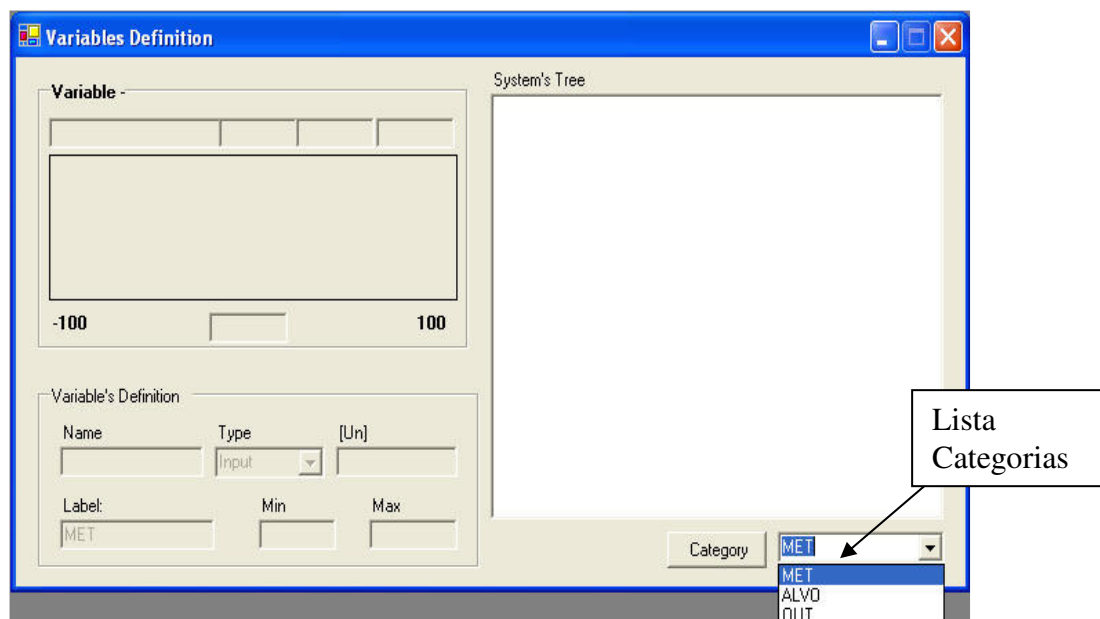


Figura 5.3 - Lista de Categorias

5.2.1 Deletar Categoria

- ✓ Na tela **Variables Definition** (Figura 5.2), Clique no botão “**Category**”.
- ✓ A tela **Category** será exibida.
- ✓ Clique no botão **Delete**.

5.3 Adicionar Variáveis

A cada variável o valor máximo e mínimo, unidade, tipo devem ser adicionados.

Passos

- ✓ Escolha no Campo **Category** a categoria da variável.
- ✓ Com o botão direito do mouse clique sobre a área branca. O menu será exibido como ilustra a Figura 5.4. Para adicionar uma variável clique no menu em **Variable**.

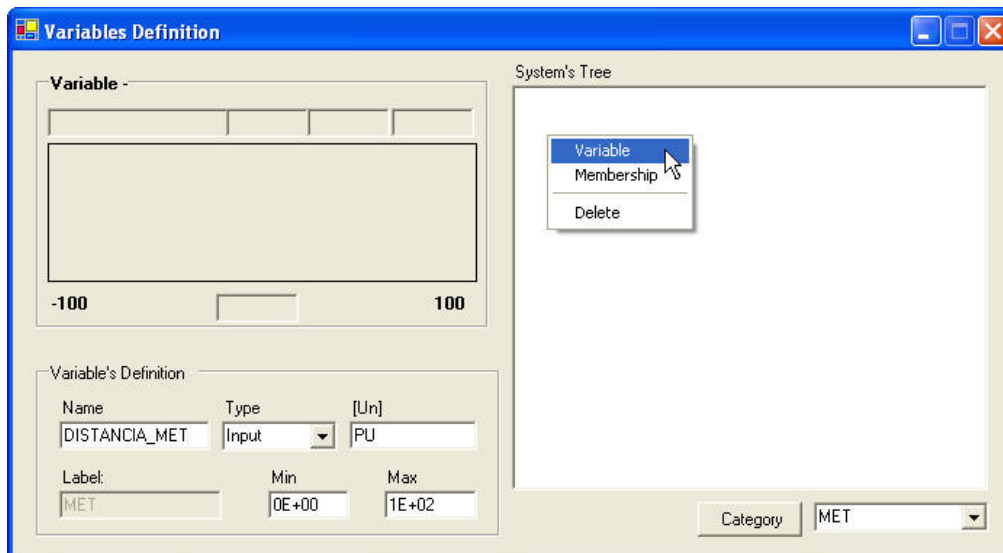


Figura 5.4 - Menu Adicionar Variável

Figura 5.5A Figura 5.5 apresenta a tela **Add Variable**.

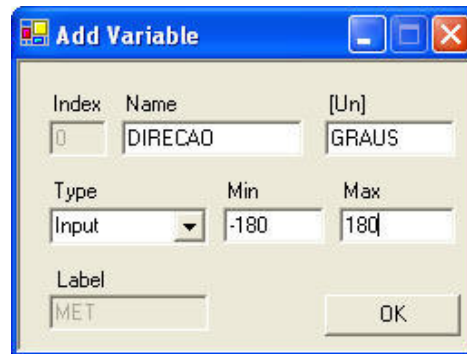


Figura 5.5 - Adicionar Variável

- ✓ Preencha os campos para a variável.
- ✓ Clique **Ok**.

Os mesmos passos devem ser seguidos para as outras variáveis da categoria.

Próximos passos:

- ✓ Altere a Categoria para “ALVO”.
- ✓ Insira as variáveis da categoria.

Os passos devem ser repetidos para a categoria “OUT”.

Na Figura 5.6, a variável “DIRECAO_MET” encontra-se em destaque na cor vermelho isto significa que ela está ativa e que todos os dados exibidos na tela são referentes a ela.

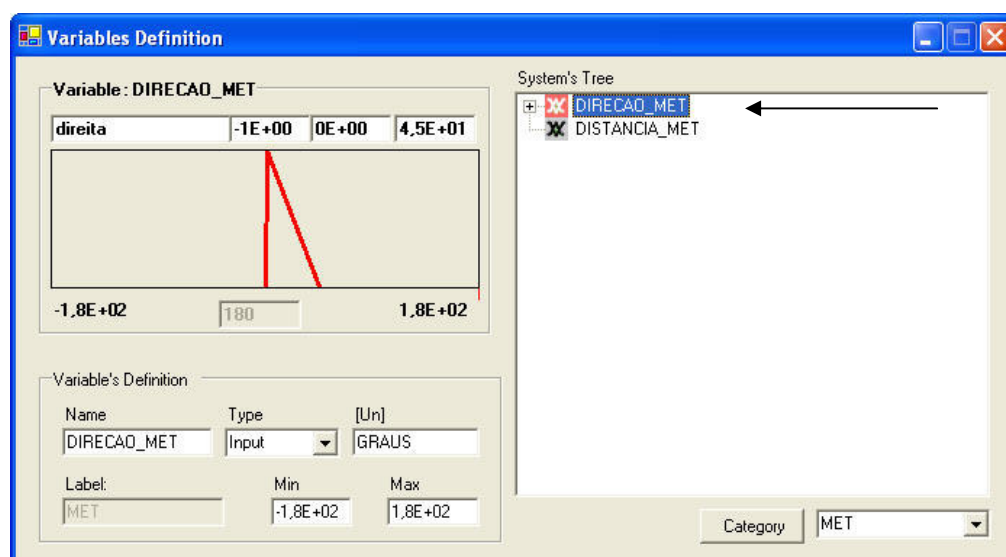


Figura 5.6 - Variável Ativa

Para ativar qualquer variável basta clicar na mesma.

5.3.1 Alterar dados da variável

Os dados da variável são exibidos na tela **Variables Definition** conforme ilustra a Figura 5.7.

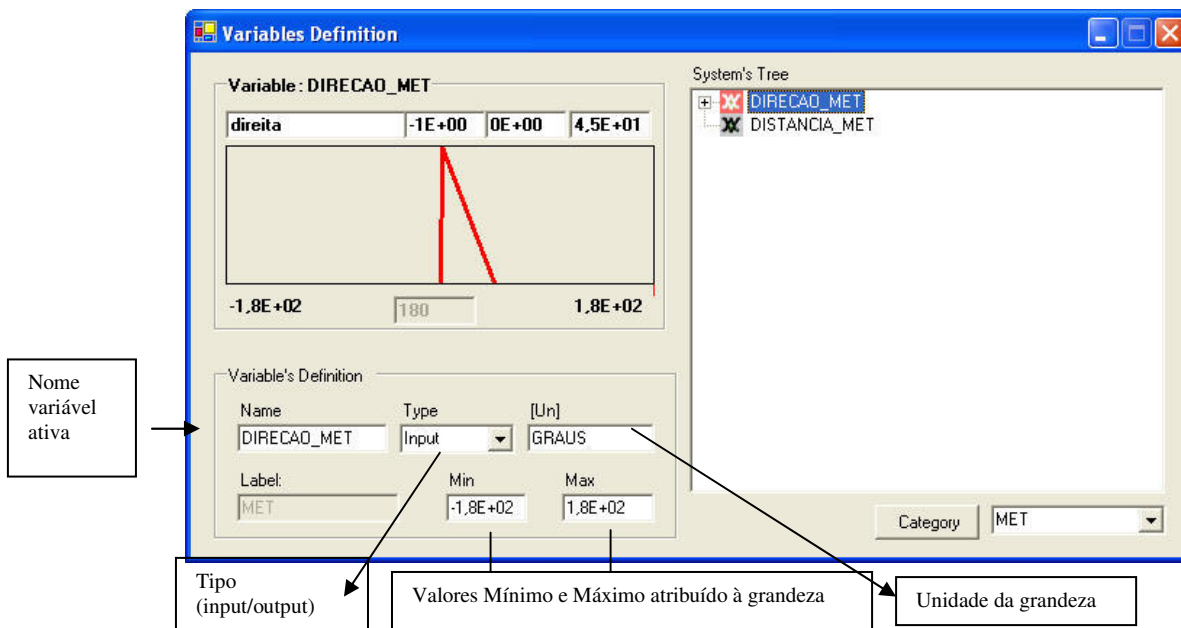


Figura 5.7 – Alterar Dados da Variável

Para quaisquer alterações:

- ✓ Clique sobre o campo que deseja alterar (Por exemplo, no campo **Name**).
- ✓ Entre com novo valor.
- ✓ Pressione “**Enter**” no teclado, após cada campo alterado.

A alteração só será efetivada se a tecla “Enter” for pressionada.

5.3.2 Deletar variável:

- ✓ Clique sobre o nome da variável ativando-a.
- ✓ Clique com o botão direito do mouse para exibição do menu.
- ✓ Clique **Delete**.

A variável e todos os memberships pertencentes a ela serão excluídos.

5.4 Adicior Membership's

- ✓ Clique inicialmente na variável que deseja adicionar os memberships. Ela ficará ativa.
- ✓ Clique com o botão direito do mouse sobre a área branca para exibição do menu;
- ✓ No menu, clique em **Membership**.
- ✓ Escolha posição para o membership (**Left**, **Center** ou **Right**).

A Figura 5.8 ilustra a adição de memberships.



Figura 5.8 – Menu Adicionar Membership

A Tela **Add Membership** é exibida conforme Figura 5.9.



Figura 5.9 - Adicionar Membership

- ✓ Entre com o nome do membership;
- ✓ Clique **Ok**.

Mesmos passos para restantes dos dados.

A Figura 5.10 apresenta a tela com todos os campos preenchidos para a categoria “MET”.

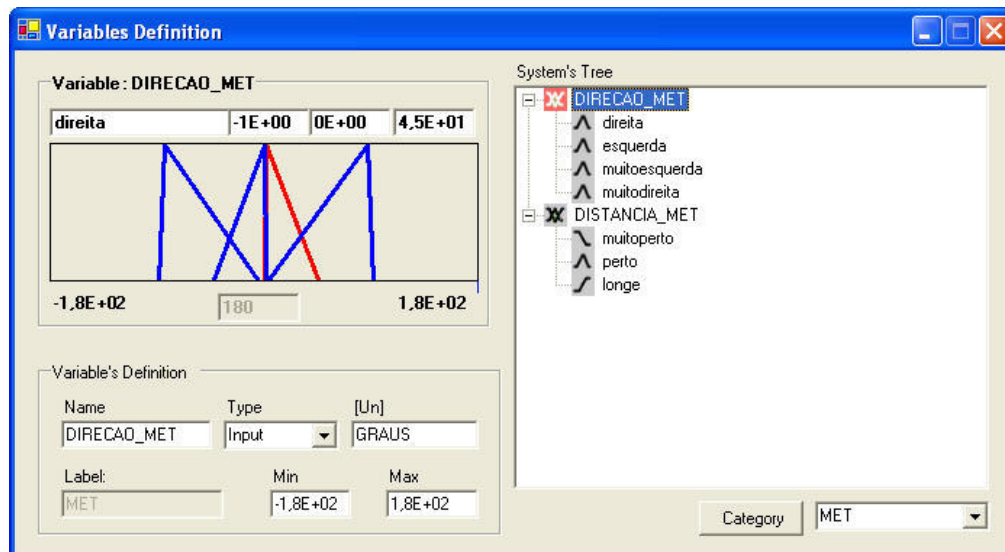


Figura 5.10 - Tela Completa

Observe na que a variável **DIRECAO_MET** esta ativa, portanto todos os dados exibidos na tela pertencem a esta grandeza.

5.4.1 Alterar Dados dos Membership's

A Figura 5.11 apresenta a tela Variables Definition com enfoque nos dados dos memberships.

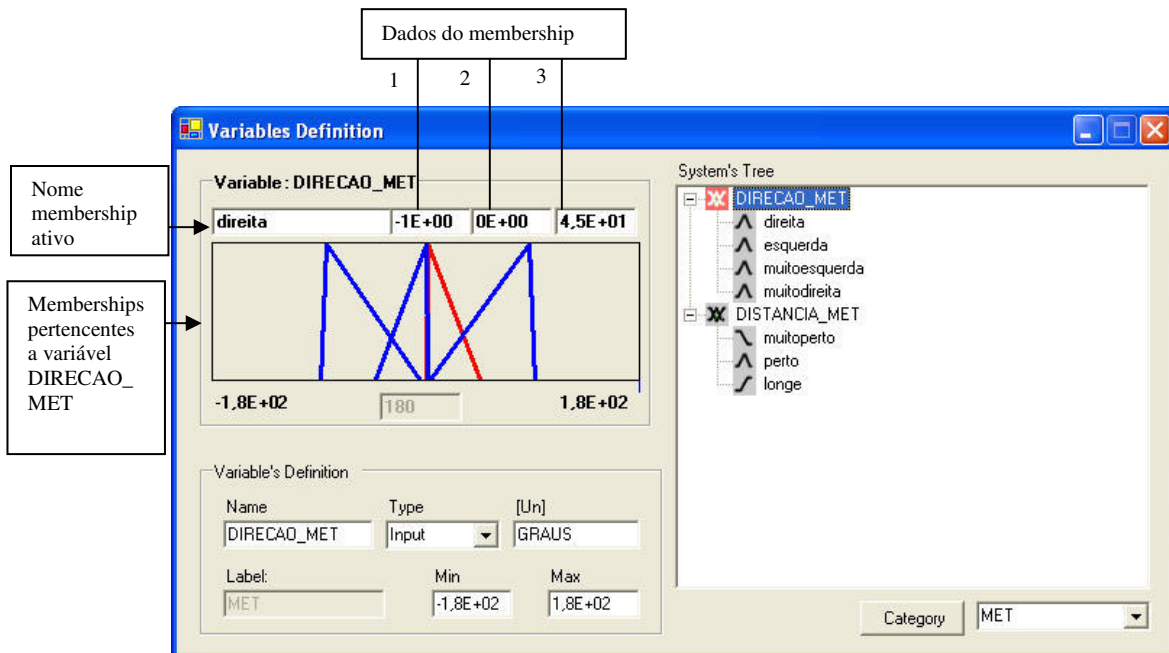


Figura 5.11 - Dados Memberships

5.4.2 Dados de cada membership:

A entrada dos campos de pertinências (memberships) é feita sob a forma de segmentos de reta, para efetivar os valores deve-se ajustar as réguas conforme desejado, criando novo segmentos.

Passos:

- ✓ Clique na variável desejada. Os memberships da variável são exibidos.
- ✓ Clique sobre a reta do membership para selecioná-lo. O membership selecionado fica em destaque (Figura 5.11: membership direita).
- ✓ Ajuste as réguas para os valores desejados.

Esta alteração pode ser realizada das seguintes formas

1- Pelos segmentos de reta dos memberships

1.1 - Alterando todos os pontos simultaneamente:

Clique no centro da reta. Com o botão do mouse pressionado ajuste para os valores desejados.

1.2 - Para alterar valores dos pontos:

Clique na extremidade da reta desejada, com o botão do mouse pressionado; movimente o ponto, alterando os dados somente daquele ponto.

2- Entrando com valores exatos

No exemplo entraremos com valores exatos do membership. A Figura 5.12 apresenta os campos que contem os valores dos pontos do membership.






Figura 5.12 - Valores Pontos Memberships

Passos para entrar com os valores:

- ✓ Selecione o membership . Pode selecioná-lo clicando no nome do membership, por exemplo, “**direita**” no campo “**System Tree**” ou no segmento de reta.

Entre diretamente com os valores nos campos 1,2 e 3 indicados na Figura 5.12. Tecler “**Enter**” no teclado após cada campo alterado, caso contrário o valor anterior é estabelecido.

Os valores aparecem em notação científica para melhor visualização.

A Figura 5.13 apresenta a correspondência dos pontos para os Memberships left , center  e right :

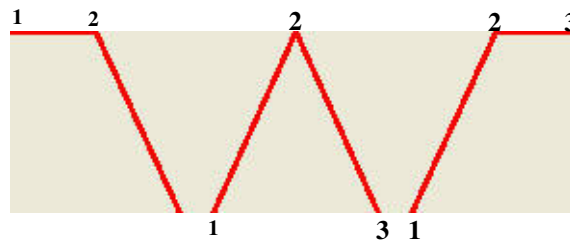


Figura 5.13 - Correspondência dos pontos

5.4.3 Alterando nome do Membership

Passos:

- ✓ Clique no campo onde está o nome do membership conforme ilustra a Figura 5.14.
- ✓ Altere o nome,
- ✓ Tecele “**Enter**”.

A Figura 5.14 ilustra os valores para o exemplo: Variável DIRECAO_MET Membership **direita**.

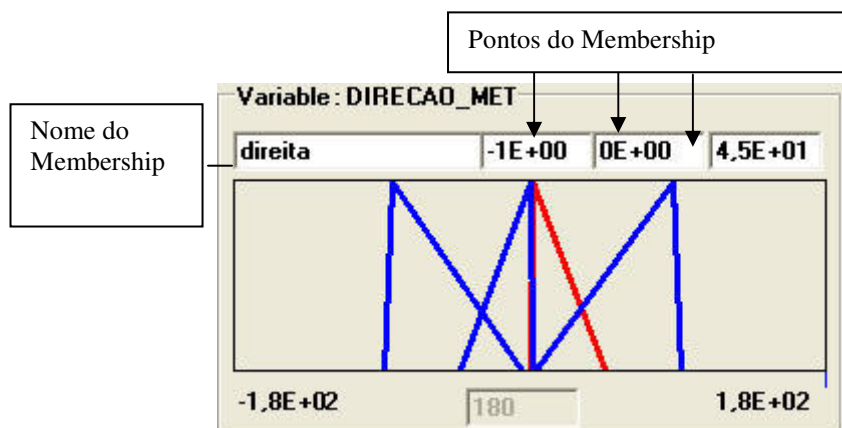


Figura 5.14 - Memberships

5.4.4 Deletar membership:

Passos:

- ✓ Clique sobre nome do membership no campo **System Tree**, conforme mostra a Figura 5.15.
- ✓ Clique com o botão direito do mouse para exibição do menu.
- ✓ Clique **Delete**.

A Figura 5.15 apresenta a exclusão do membership “direita”.

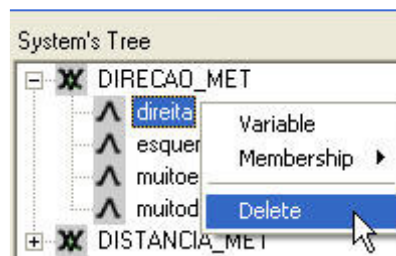


Figura 5.15 - Deletar Memberships

5.5 Gerar Regras

As regras determinam o comportamento da atuação do sistema. São criadas escolhendo-se a grandezas de atuação com seus respectivos memberships, que aparecem presentes para facilitar a entrada por parte do operador.

A Tabela 5. 1 apresenta as regras para a primeira Simulação.

Tabela 5. 1 Regras da primeira Simulação.

○ METEORO A ESQUERDA 1						
If	DIRECAO_MET	=	muitoesquerda	AND	DISTANCIA_MET	= perto
THEN	SAIDA_OUT	=	direita			
○ METEORO A ESQUERDA 2						
If	DIRECAO_MET	=	muitoesquerda	AND	DISTANCIA_MET	= muitoperto
THEN	SAIDA_OUT	=	muitodireita			

○ METEORO A ESQUERDA 3					
If	DIRECAO_MET	=	esquerda	AND	DISTANCIA_MET = perto
THEN	SAIDA_OUT	=	direita		
○ METEORO A ESQUERDA 4					
If	DIRECAO_MET	=	esquerda	AND	DISTANCIA_MET = muitoperto
THEN	SAIDA_OUT	=	muitodireita		
○ METEORO A DIREITA 1					
If	DIRECAO_MET	=	muitodireita	AND	DISTANCIA_MET = muitoperto
THEN	SAIDA_OUT	=	muitoesquerda		
○ METEORO A DIREITA 2					
If	DIRECAO_MET	=	muitodireita	AND	DISTANCIA_MET = perto
THEN	SAIDA_OUT	=	esquerda		
○ METEORO A DIREITA 3					
If	DIRECAO_MET	=	direita	AND	DISTANCIA_MET = muitoperto
THEN	SAIDA_OUT	=	muitoesquerda		
○ METEORO A DIREITA 4					
If	DIRECAO_MET	=	direita	AND	DISTANCIA_MET = perto
THEN	SAIDA_OUT	=	esquerda		
○ DISTANCIA ALVO PERTO 1					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = muitoesquerda
THEN	SAIDA_OUT	=	muitoesquerda		
○ DISTANCIA ALVO PERTO 2					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = esquerda
THEN	SAIDA_OUT	=	esquerda		
○ DISTANCIA ALVO PERTO 3					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = frente
THEN	SAIDA_OUT	=	frente		
○ DISTANCIA ALVO PERTO 4					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = direita
THEN	SAIDA_OUT	=	direita		
○ DISTANCIA ALVO PERTO 5					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = muitodireita
THEN	SAIDA_OUT	=	muitodireita		
○ DISTANCIA ALVO LONGE 1					
If	DISTANCIA_ALVO	=	longe	AND	DIRECAO_ALVO = muitoesquerda
THEN	SAIDA_OUT	=	esquerda		
○ DISTANCIA ALVO LONGE 2					
If	DISTANCIA_ALVO	=	longe	AND	DIRECAO_ALVO = esquerda
THEN	SAIDA_OUT	=	esquerda		
○ DISTANCIA ALVO LONGE 3					
If	DISTANCIA_ALVO	=	longe	AND	DIRECAO_ALVO = frente
THEN	SAIDA_OUT	=	frente		
○					

○ DISTANCIA ALVO LONGE 4					
If	DISTANCIA_ALVO	=	longe	AND	DIRECAO_ALVO = direita
THEN	SAIDA_OUT	=	direita		
○ Regra					
If	DISTANCIA_ALVO	=	longe	AND	DIRECAO_ALVO = muitodireita
THEN	SAIDA_OUT	=	direita		

5.5.1 Entrar com cada regra

Considerando a regra:

If DIRECAO_MET = muitoesquerda **AND** DISTANCIA_MET = perto
THEN SAIDA_OUT = direita

Passos:

Abra a tela **Rules Generation**

- ✓ Clique no menu **Fuzzy / Rules Generation** . Figura 5.16 ilustra a tela.



Figura 5.16 - Menu Gerar Regra

Tela **Rules Generation** é exibida. Figura 5.17.

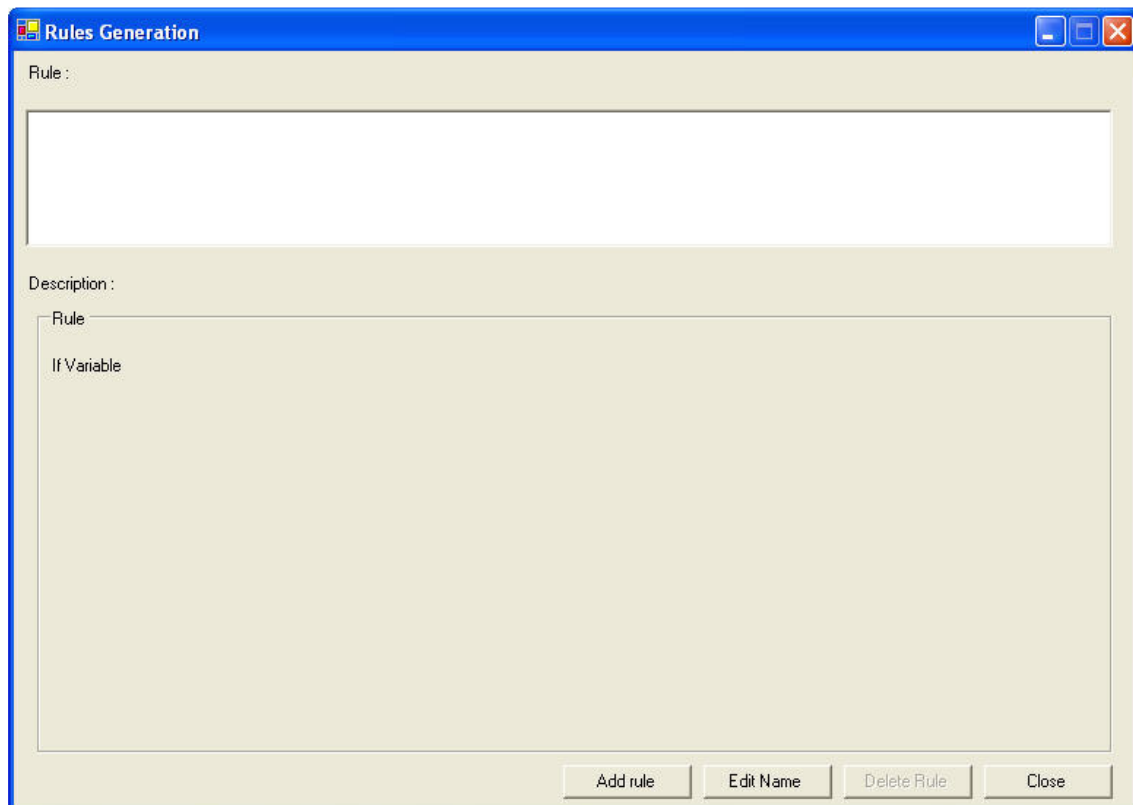


Figura 5.17 - Tela Rules Generation

Nesta, todas as regras do sistema podem ser visualizadas.

Para adicionar uma regra:

- ✓ Clique no botão **Add Rule**.

Tela **Rules Generation** é exibida conforme ilustra a Figura 5.18.

Nesta tela as grandezas de atuação e seus respectivos memberships estão presentes para facilitar a geração das regras por parte do operador.

Para dar entrada a uma regra, é necessário escolher uma das grandezas de entrada (**DIRECAO_MET**, por exemplo) no sistema da lógica Fuzzy com sua respectiva condição (membership **muitoesquerda**, por exemplo). Se houver mais de uma condição para haver a atuação deve-se acionar o botão “AND” e repetir o processo.

O segundo passo é seleccionar a grandeza e sua respectiva condição (**SAIDA_OUT** e **direita**, no exemplo) que farão a atuação, ou seja, serão responsáveis pela resposta do sistema.

A Figura 5.18 ilustra a construção da regra.

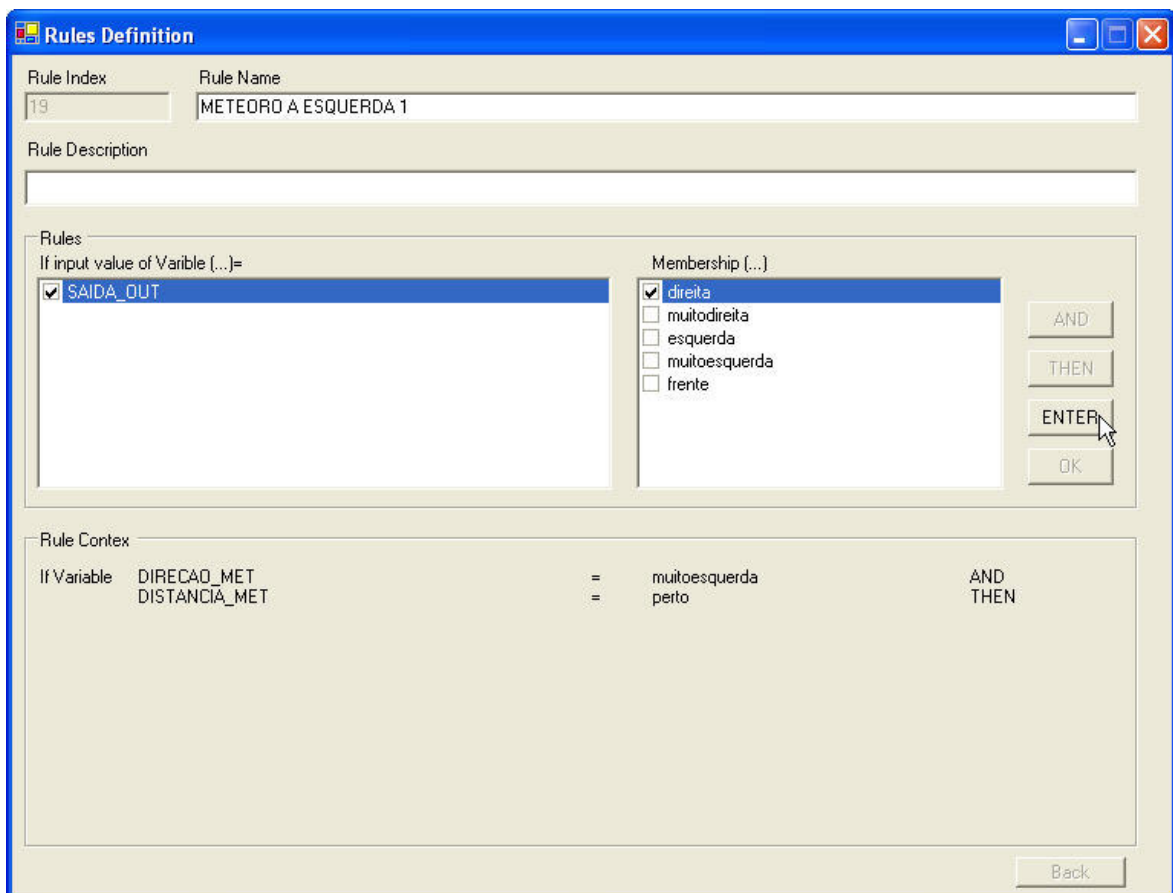


Figura 5.18 - Geração das Regras

Passos:

- ✓ Selecione a variável desejada. (Por exemplo, **DIRECAO_MET**).
- Ao seleccionar a variável, seus respectivos memberships são exibidos.
- ✓ Selecione o membership desejado. (Por exemplo, **muitoesquerda**).
- ✓ Clique no botão **AND**.

- ✓ Selecione outra grandeza. (**DISTANCIA_MET**).
- ✓ Selecione o membership. (**perto**).

Com as condições já estabelecidas (**DIRECAO_MET = muitoesquerda = AND DISTANCIA_MET = perto**).

- ✓ Clique no botão **THEN**.
- ✓ Clique na grandeza (**SAIDA_OUT**) e membership (**direita**).
- ✓ Clique no botão **ENTER** para concluir a regra.
- ✓ Clique no Botão **OK** para salvar a regra.

Mesmos passos para adição das demais regras.

A tela **Rules Generation** ilustrada na Figura 5.19, exibe todas as regras.

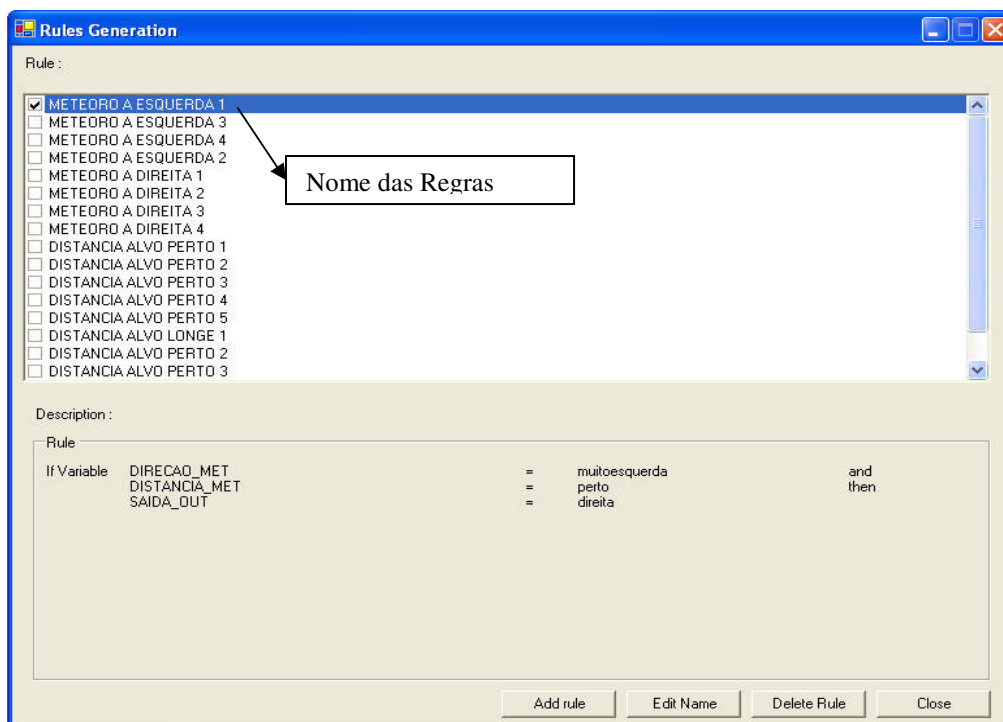


Figura 5.19 - Regras do Sistema

5.5.2 Deletar regras

Passos

- ✓ Selecione o nome da regra.

- ✓ Clique no botão **Delete Rule**.

Tela para confirmação é exibida. Clique **OK**.

5.5.3 Editar nome da regra

Passos

- ✓ Selecione o nome da regra.
- ✓ Clique no botão **Edit Name**.

A Figura 5.20 apresenta a tela de geração das regras, com a tela “Edit Rule” para edição do nome da regra.

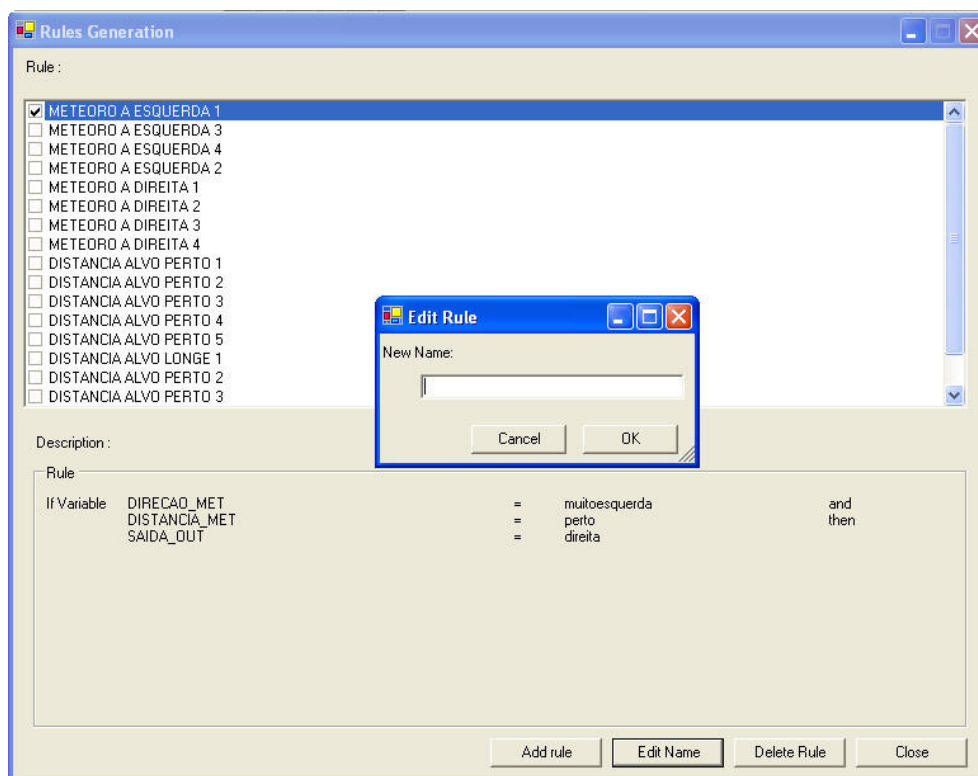


Figura 5.20 - Editar Nome da Regra

Passos para edição

- ✓ Entre com o novo nome da Regra.
- ✓ Clique no botão **OK** para confirmar, ou **Cancel** para cancelar operação.

5.5.4 Alterar ou Deletar Nome de Variáveis e/ou Memberships após definição das regras.

1. Alterar nome de variável ou membership:

Caso o usuário altere o nome de variáveis ou memberships após a definição das regras, estas são atualizadas automaticamente.

2. Deletar nome de variável ou membership:

Regras que contenham a variável ou membership são deletadas automaticamente.

5.6 Importar dados de Entrada

Os valores de entrada podem ser importados de uma planilha. Cabe ressaltar que aqui será mostrada somente uma possibilidade do sistema.

Passos:

1. Exportar Cabeçalho (Export Header) responsável pelo formato que deverá ter a planilha.
2. Importar Dados de entrada da planilha.

- **Exportar Cabeçalho - Formato da planilha**

O sistema gera automaticamente o formato padrão da planilha que deverá ser preenchida com os dados de entrada.

Os dados são exportados para uma planilha Excel contendo os nomes das variáveis, facilitando a entrada dos dados.

Passos:

- ✓ Clique no menu **Fuzzy/ImportExport/Export Header**
- ✓ Tela **Salvar como** é exibida.
- ✓ Salve a planilha.

A planilha Excel é gerada conforme ilustra a Figura 5.21.

Campo 1 – Usuário deve Informar o número de linha de dados					
▲ A	B	C	D	E	
Número	Grandezas				
Nome	DIRECAO_MET	DISTANCIA_MET	DIRECAO_ALVO	DISTANCIA_ALVO	

Figura 5.21 - Cabeçalho da Planilha

A 1ª coluna da planilha: Campo “Número”: corresponde ao número de linhas de dados que terá a planilha. Campo “Nome”: nome dado a cada linha de dados da planilha.

Somente serão importados para o sistema variáveis que tenham o mesmo nome declarado no Sistema Fuzzy; portanto os nomes das variáveis não devem ser alteradas.

Passos para preencher a planilha:

- ✓ Usuário deverá entrar com o número de linhas de linhas que conterà os valores de entrada.
- ✓ Usuário deverá entrar com restante dos dados.

A Figura 5.22 ilustra a planilha preenchida.

	A	B	C	D	E
1	3	Grandezas			
2	Nome	DIRECAO_MET	DISTANCIA_MET	DIRECAO_ALVO	DISTANCIA_ALVO
3	Entr 1	50	10	50	50
4	Entr 2	100	20	10	20
5	Entr 3	80	30	10	10

Figura 5.22 - Planilha Preenchida

Para correta importação dos dados o formato da planilha não pode ser alterado.

- **Importar dados de entrada**

- ✓ No menu Fuzzy, Clique **Import/Export/Import Portfolios**

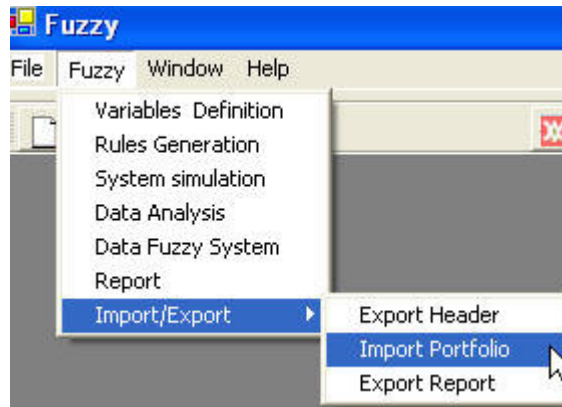


Figura 5.23 - Menu Importar Dados

Os dados são então exibidos na tela **Analysis** do Sistema.

5.7 Simulação do Sistema

Outros valores poderão ser inseridos pelo usuário. Simulações podem ser realizadas nas seguintes telas: Tela **Analysis** ou Tela **System Simulation**.

- **Tela Analysis**

- ✓ Na tela principal (Fuzzy) clique: menu **System/Data Analysis**.
- ✓ Na tabela de dados, entre com o valor desejado.
- ✓ Clique no botão **Simulation**.

A Figura 5.24 apresenta a tela Analysis

Analysis						
Variables Input					Variables Output	
	nome	DIRECAO_MET	DISTANCIA_ME	DIRECAO_ALVO	DISTANCIA_ALVO	SAIDA_OUT
	Ent1	-71	65	20	72	5,3433772755
	Ent2	-169	100	-22	48,5	-4,721252796
▶	Ent3	127	81	7,29	2,7	0,1685423186

Figura 5.24 - Análise dos Dados

- **Tela System Simulation**

- ✓ Na tela principal (Fuzzy) clique: menu **System/ System Simulation**.
- ✓ A tela **System Simulation** é exibida conforme ilustra a Figura 5.25.

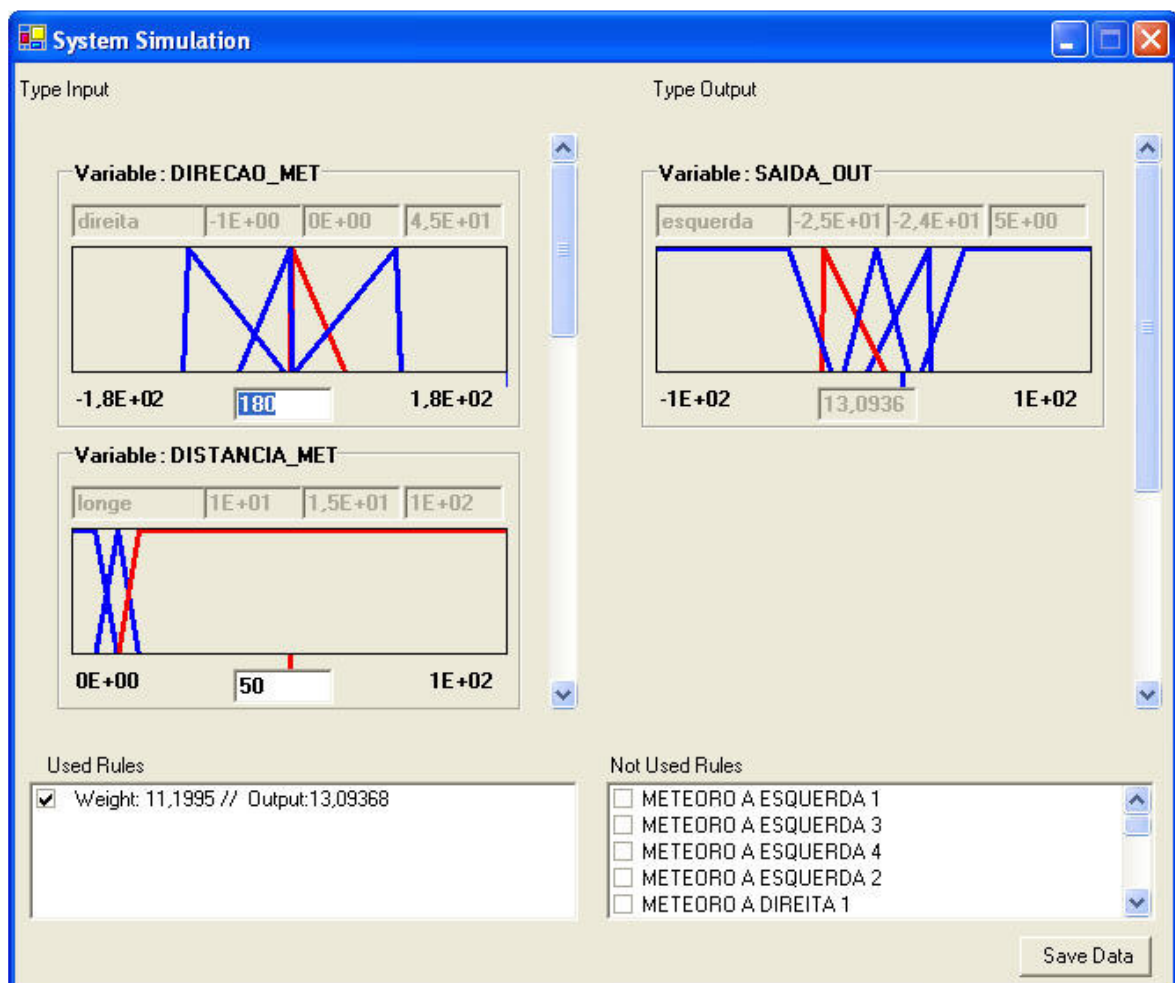


Figura 5.25 - Tela de Simulação

Observe que os campos de pertinência de cada variável são exibidos.

- Do lado esquerdo as variáveis de entrada: DIRECAO_MET, DISTANCIA_MET, DIRECAO_ALVO, etc;
- Do lado direito a variável de saída: SAIDA_OUT.

O próximo passo é determinar o valor da entrada.

De acordo com o valor da entrada são exibidas as regras aplicáveis, e as não aplicáveis. A

Figura 5.26 ilustra o processo de simulação.

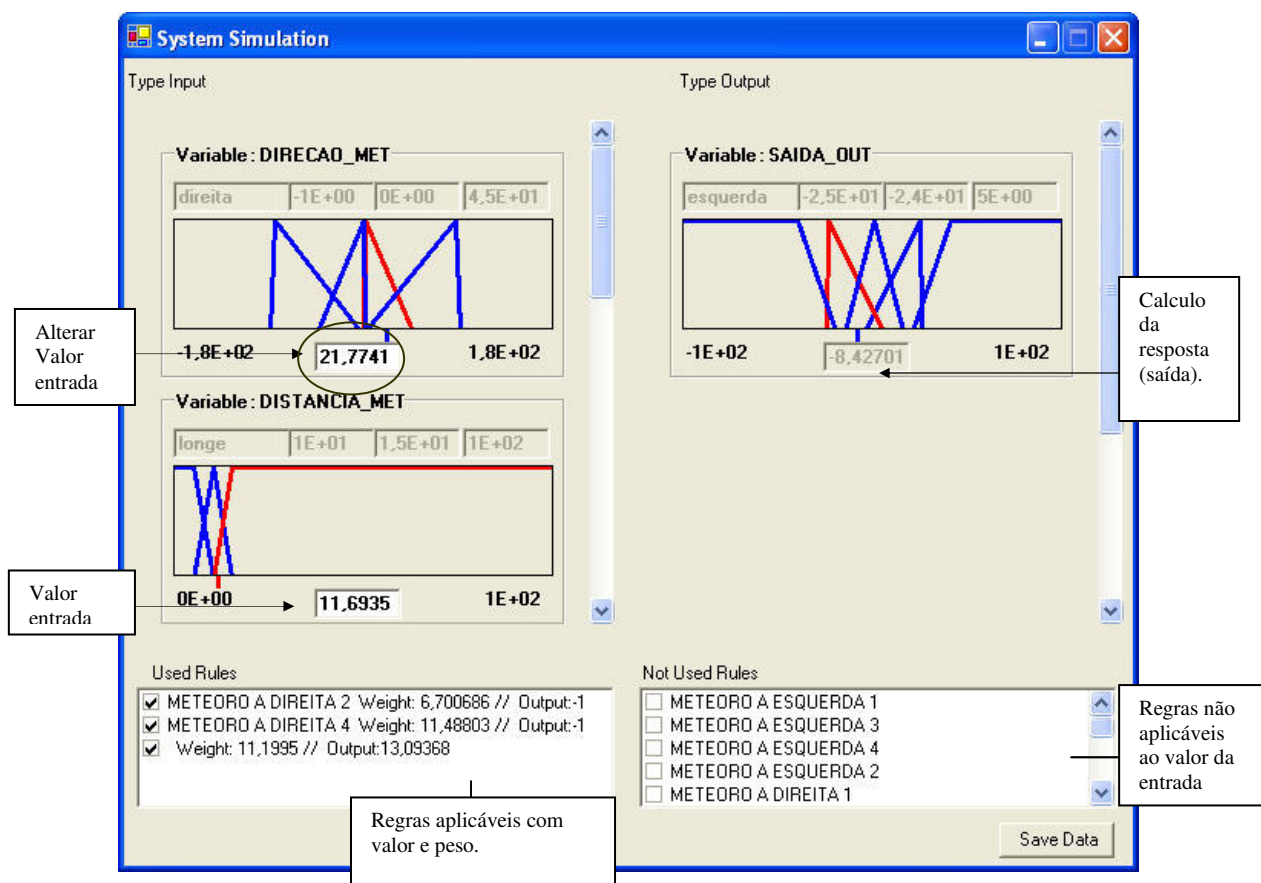


Figura 5.26 - Simulação

Passos:

Determinando o valor da entrada

- ✓ Clique na posição indicada na Figura 5.26 por um círculo.
- ✓ Entre com o valor.
- ✓ Tecele **Enter**.

Mesmos passos para as outras variáveis.

Observe que a Figura 5.26 exibe as regras utilizadas, com o peso e o valor de saída.

5.8 Relatório do Sistema Fuzzy

Este relatório é somente de visualização.

- ✓ Menu **Fuzzy** , Clique **Report**. Figura 5.27



Figura 5.27 - Menu Report

Tela **Report** é exibida conforme ilustra a Figura 5.28.

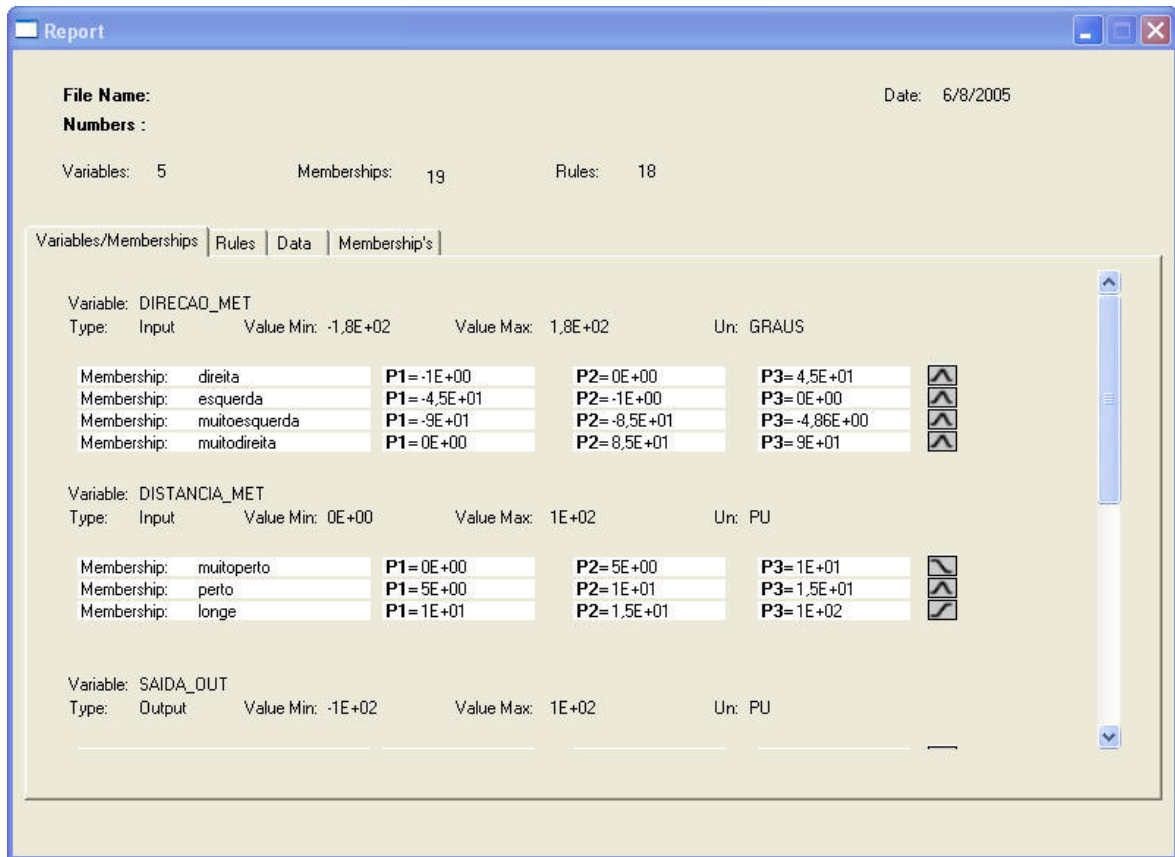


Figura 5.28 - Relatório

Esta tela exibe dados das variáveis/memberships, das regras, dados da simulação e gráfico dos memberships.

O relatório de Regras do Sistema é ilustrado na Figura 5.29.

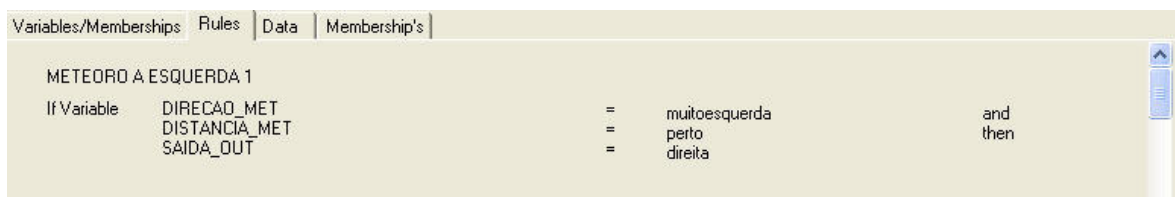


Figura 5.29 - Relatório Regras

O relatório da Simulação do Sistema é ilustrado na Figura 5.30

Simulation Input/Output					
nome	DIRECAO_ME	DISTANCIA_M	DIRECAO_ALV	DISTANC	SAIDA_OUT
Ent1	-71	65	20	72	5,343377255
Ent2	-169	100	-22	48,5	-4,721252796
Ent3	127	81	7,29	2,4	0,1685423186
*					*

Figura 5.30 - Relatório Dados

O relatório das funções de pertinência do Sistema é ilustrado na Figura 5.31.

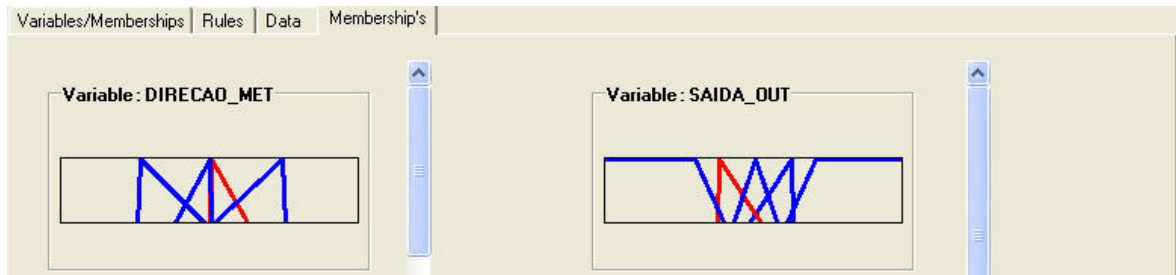


Figura 5.31 - Relatório Memberships

5.9 Salvar projeto

- ✓ Na tela principal apresentada na Figura 5.32, clique: menu **File/Save As...**

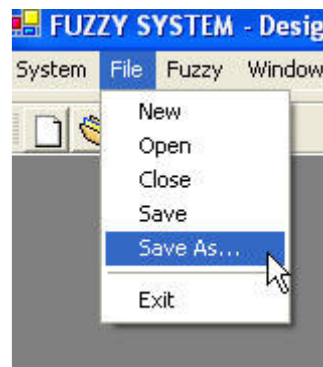


Figura 5.32 - Menu Save As

A tela **Salvar Como** é exibida.

- ✓ Escolha o nome e diretório para o projeto.
- ✓ Clique botão **Salvar**.

- ✓ O arquivo é salvo com a extensão “.Fuzzy”.

O arquivo armazena todas as variáveis, memberships, e regras de atuação. A aplicação Collision Game importa este arquivo.

5.10 Abrir projeto

- ✓ Menu **File/Open** ou o atalho mostrado na Figura 5.33. .

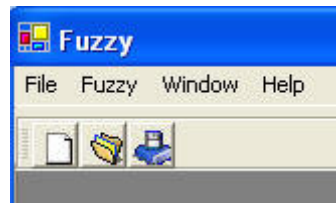


Figura 5.33 - Abrir Projeto

- ✓ Tela Abrir é exibida. Localize o arquivo com a extensão “.Fuzzy”.

5.11 Fechar Sistema

- ✓ Na tela principal (Fuzzy) clique: menu **File**, Clique **Exit**.

Notas Finais:

Variáveis da mesma categoria não poderão ter o mesmo nome.

Membership's de uma mesma variável não poderão ter o mesmo nome.

(Caso o usuário entre com nomes que já existam no sistema uma mensagem de erro será exibida).

5.12 O Jogo: Aplicação Collision Game

A finalidade deste jogo é aplicar a biblioteca contendo a matemática da lógica Fuzzy.

Para o desenvolvimento da aplicação foram empregados:

- A ferramenta de desenvolvimento da lógica Fuzzy (Sistema Fuzzy) apresentada nos tópicos anteriores.
- A biblioteca Fuzzy.
- A linguagem de programação C#.

Descrição do jogo

O jogo é construído por uma nave espacial, cujo objetivo é chegar à Terra. Entretanto existem os meteoros como obstáculos. Para chegar ao seu objetivo, a nave deve desviar destes. Trata-se de controlar a posição da nave de modo que ela possa atingir seu objetivo. Se a nave colidir com algum obstáculo, gera uma explosão. Caracteriza-se como o fim do jogo, quando a nave atinge a Terra.

A posição inicial da nave é sempre a mesma. A velocidade tem um valor inicial. Para acelerar ou frear, as setas do teclado devem ser pressionadas. (direita: acelerar, esquerda: frear, seta para cima: virar à esquerda, para baixo: virar à direita). Entretanto, para as simulações a nave orienta-se segundo regras Fuzzy.

A direção que a nave espacial deve seguir é calculada de acordo com a distância e posição em relação à Terra e aos meteoros. A nave desvia dos meteoros de acordo com a direção dos mesmos e; distância deles com a nave, virando no sentido contrário à direção dos meteoros. O peso é dado pela distância, quanto mais perto, mais ela vira. Como tem o

objetivo chegar á Terra verifica-se então o alvo e vira de acordo com a posição que esta em relação a ele.

O sistema é composto de quatro classes, três responsáveis pela aplicação em si, e a última responsável pelo método de defuzzificação, que será usada para o cálculo do valor de saída. Esta é gerada pelos valores obtidos nas conclusões das várias regras, que são agregados em uma única ação de controle através de uma média ponderada, como mostra a equação abaixo. Os pesos são valores que mostram o grau de compatibilidade do valor.

$$M = \frac{(\sum (peso * valor_i))}{\sum peso}$$

Dados da primeira simulação

Os valores das variáveis, memberships e as regras de controle foram exibidos nos tópicos anteriores para demonstrar o uso da Ferramenta Fuzzy. As regras podem ser visualizados na Tabela 5.1.

- Número de meteoros: 4.
- Condição dos meteoros: sem movimento, inseridos randomicamente.
- Número de variáveis de entrada: 4
- Número de variáveis de saída: 1
- Número de memberships: 19
- Número de regras: 18

A figura 5.34 exibe os campos de pertinência das grandezas da primeira simulação.

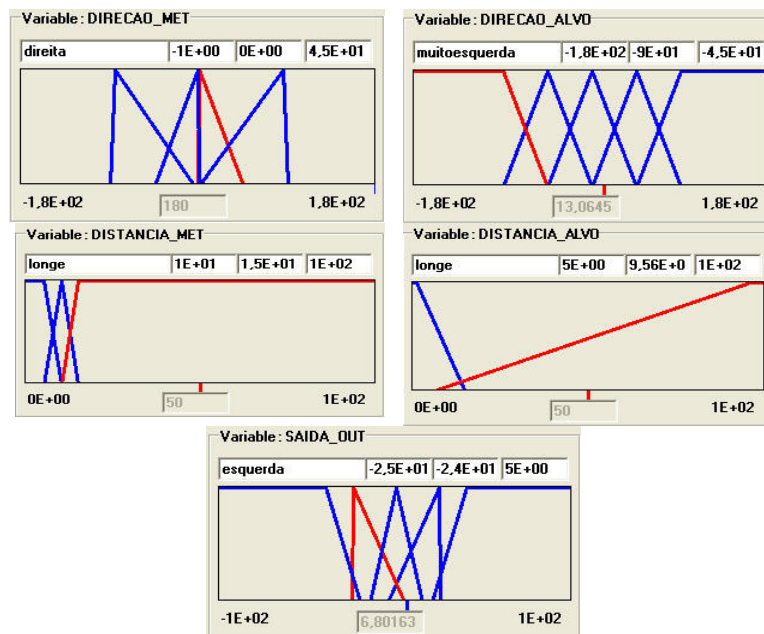


Figura 5.34 - Memberships

Nas definições dos valores máximos e mínimos de cada variável de entrada observa-se que a variável Direção_Met e Direção_Alvo podem variar no intervalo de -180 a 180 graus; para Distancia_Met e Distancia_Alvo será considerada a variação em porcentagem de 0 a 100 (importância que se dá à distancia). Por último a variável de saída Saída_out, entre -100 e 100.

As funções de pertinência da variável Direção_Met (*direita*, *esquerda*, *muitodireita* e *muitoesquerda*) e Direcao_Alvo (*frente*, *direita*, *esquerda*, *muitodireita*, *muitoesquerda*) representam o ângulo da nave com os meteoros e com o alvo, respectivamente. Das variáveis Distancia_Met (*muitoperto*, *perto*, *longe*) e Distancia_Alvo (*muitoperto*, *longe*) representam a distância da nave com os meteoros e com o alvo. De acordo com o valor de entrada, são calculadas as pertinências para cada membership; estes índices são atribuídos a um processo de inferência de acordo com as regras determinadas, expressando a trajetória da nave.

Na primeira simulação um conjunto de 18 regras descreve o comportamento da nave. As oito primeiras são responsáveis pelo controle da nave com o meteoro, as demais da nave com o alvo.

As regras foram especificadas visando cobrir toda a extensão do problema, ou seja, desviar de todos os meteoros verificando a direção e distância em relação à nave e atingir o alvo, também de acordo com a direção e distância em relação à nave. A estratégia de controle é dada pela distância. Quanto mais perto do meteoro, mais a nave vira.

Uso da biblioteca Fuzzy no sistema

A classe Saída, parte do sistema Collision Game, calcula o valor de saída, valor responsável pela direção que o míssil vai seguir.

A Figura 5.35 apresenta o algoritmo para aplicar as regras. A biblioteca Fuzzy foi utilizada para cálculo dos valores de pertinência de acordo com as entradas; a classe Saída para cálculo de Defuzzificação.

```

public bool AplicaRegras()
{
    saidafuzzy.Clear();
    float final = 0f;
    float kte = 4f;

    try
    {
        float valor, peso;
        //fuzzy em relacao a meteoros
        for(int i = 0 ; i < nmeteoros ; i ++)
        {
            sis.retornagr("DIRECAO_ALVO").entr = AlvoAngulo ;
            sis.retornagr("DISTANCIA_ALVO").entr = AlvoDistancia / kte;
            sis.retornagr("DIRECAO_MET").entr = valorAngulo[i];
            sis.retornagr("DISTANCIA_MET").entr = valorDistancia[i] / kte;
            sis.aplicarregras(1);
            valor = (float)sis.retornagr("SAIDA_OUT").grsms.posx;
            peso = (float)sis.retornagr("SAIDA_OUT").grsms.peso;
            saidafuzzy.Add(new Saida(valor,peso));
        }
        final = saidafuzzy.MediaPonderada();
    }
    catch
    {
        final = 0f;
    }
    final = (float)(final * Math.PI / 180);
    Nv1.direcao += final;
    return true;
}

```

Figura 5.35 - Algoritmo

No algoritmo a variável “sis” representa a biblioteca fuzzy. Para atribuir os valores de entrada a função *retornagr* desta biblioteca foi utilizada. Como parâmetro para a função é passado o nome das grandezas criadas no Sistema Fuzzy.

sis.retornagr("DIRECAO_ALVO").entr = AlvoAngulo

Como mencionado os valores de entrada foram calculados de acordo com os ângulos formados entre a nave e os meteoros, e nave com Terra considerando a direção da nave; também com a distância da nave com os meteoros e com o alvo. A medida que são assimilados pelo sistema a nave vai se deslocando respeitando as regras de controle.

A Figura 5.36 apresenta o movimento da nave.

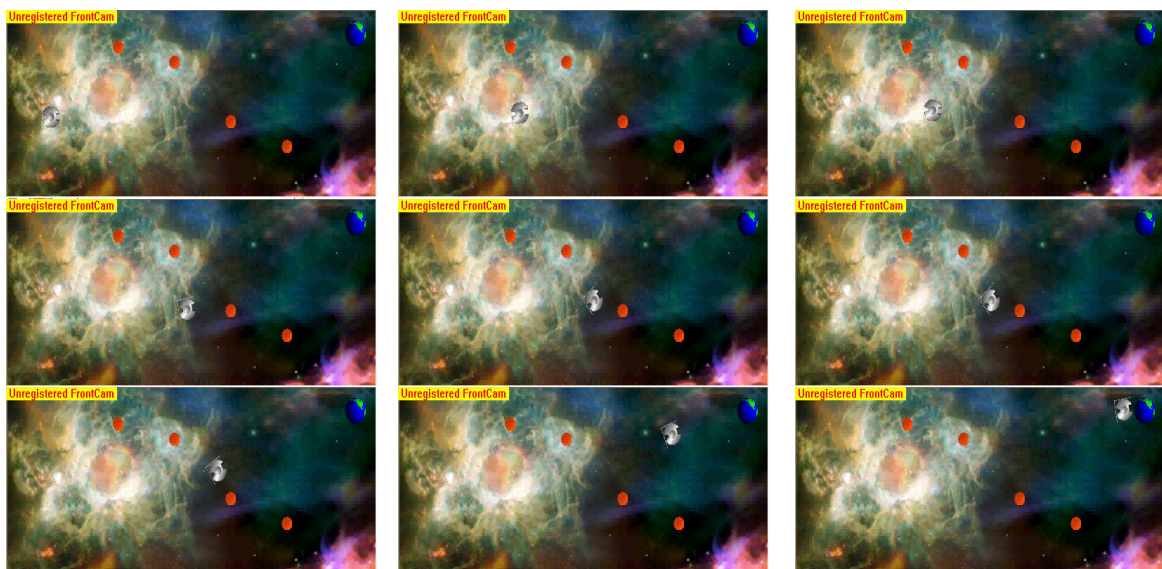


Figura 5.36 - Primeira Simulação do Jogo

Como pode ser visualizado, a nave desvia dos meteoros e chega à Terra.

Para um número fixo de meteoros a resposta foi regular. Se alterado o número, há casos de colisões. Consequentemente o sistema atual não responde satisfatoriamente as ocorrências.

Isto se deve ao fato do conjunto de regras, de acordo com os memberships atribuídos às grandezas, não estar completo. Ou seja, as regras não cobrem toda a extensão necessária. O maior problema foi visualizado no desvio da nave com o meteoro.

Para a distância da nave com os meteoros foram criadas três funções de pertinência (muito perto, perto e longe). Quando a nave se encontrava em uma distância muito pequena com o meteoro, não desviava a contento, já que no processo, o sistema, com um número maior de regras acabava calculando um valor de desvio não suficiente para a nave. Podemos dizer que um dos maiores problemas estava no número de memberships atribuídos à grandeza “distancia”, já que no processo de inferência, com memberships em mais de uma regra, a saída não era satisfatória, gerando colisões.

Para corrigir o problema os valores dos memberships relacionados à direção dos meteoros foram alterados e os memberships relacionados à distância, reduzidos.

Foi observado também que esta simulação contou com excesso de parâmetros para o “alvo” e conseqüentemente em muitas regras. Cabe ressaltar que o bom desempenho do sistema depende das regras e funções de pertinência. E definir um conjunto de regras e funções de pertinência já na primeira análise não é trivial.

Dados da segunda simulação

Nesta simulação há um número menor de memberships, algumas funções de pertinência foram ajustadas de forma empírica de acordo com o sistema analisado.

Para corrigir o problema da primeira solução, os memberships da variável *direção_alvo*, *distancia_met* e *saída_out* foram reduzidos, afetando diretamente no cálculo dos índices de pertinência. Com isto o valor de saída acaba sendo maior, corrigindo, por exemplo, o problema de desvio da nave com o meteoro. Os memberships da variável

direção_met tiveram a amplitude alterada, o que influencia diretamente no valor de saída de acordo com a entrada.

Visando melhorar a relação entrada-saída fez-se um ajuste na base de regras. Nesta simulação os meteoros se movem.

- Número de meteoros: 5.
- Condição dos meteoros: inseridos randomicamente e com movimento.
- Número de variáveis de entrada: 4
- Número de variáveis de saída: 1
- Número de memberships: 14
- Número de regras: 11

A figura 5.37 exibe os campos de pertinência das grandezas da segunda simulação.

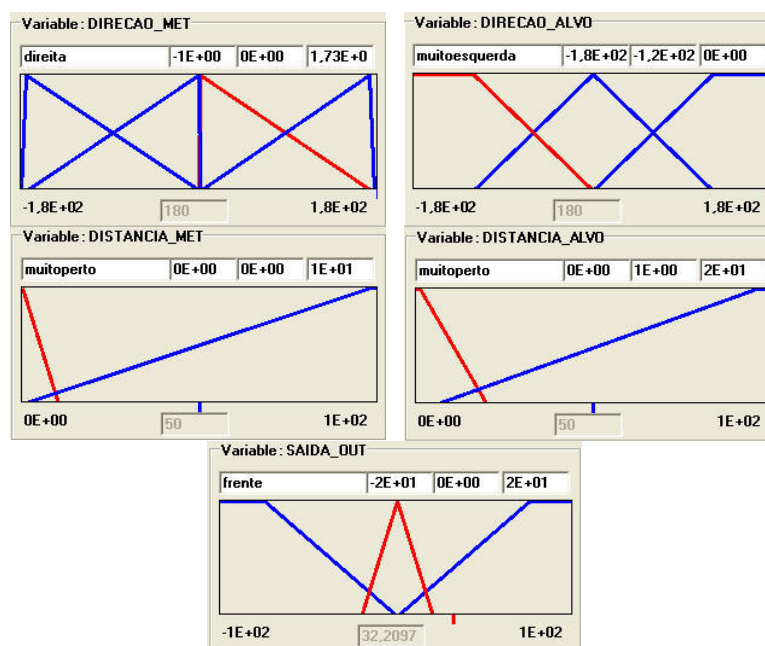


Figura 5.37 - Memberships 2 Simulação

Os valores de discurso (valores máximos e mínimos) das variáveis são os mesmos da primeira simulação. Como mencionado as funções de pertinência foram alteradas, este

ajuste envolveu um processo de observação. Para a grandeza *direção_alvo* o número foi reduzido para três funções (*frente*, *muitodireita* e *muitoesquerda*). Com isto o número de regras também foi alterado; foram removidas as regras desnecessárias.

Nesta simulação um conjunto de 11 regras é responsável pelo comportamento da nave. Para a distância com o meteoro, as regras de 1 a 4 ponderaram a condição “muito perto” ; quando esta distância é considerada, verifica-se direção do meteoro e de acordo com este, o desvio da nave é maior ou menor. A quinta regra verifica se a distância com o meteoro é considerada longe, caso seja a nave segue em frente. As demais verificam a direção e distância relativas ao alvo.

A Tabela 5.2 apresenta as regras para a segunda Simulação.

Tabela 5.2 Regras da segunda Simulação.

○ METEORO PERTO 1					
If	DISTANCIA_MET	=	muitoperto	AND	DIRECAO_MET = direita
THEN	SAIDA_OUT	=	muitoesquerda		
○ METEORO PERTO 2					
If	DISTANCIA_MET	=	muitoperto	AND	DIRECAO_MET = Esquerda
THEN	SAIDA_OUT	=	muitodireita		
○ METEORO PERTO 3					
If	DISTANCIA_MET	=	muitoperto	AND	DIRECAO_MET = muitoesquerda
THEN	SAIDA_OUT	=	frente		
○ METEORO PERTO 4					
If	DISTANCIA_MET	=	muitoperto	AND	DIRECAO_MET = muitodireita
THEN	SAIDA_OUT	=	frente		
○ METEORO LONGE 1					
If	DISTANCIA_MET	=	longe		
THEN	SAIDA_OUT	=	frente		
○ ALVO PERTO 1					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = frente
THEN	SAIDA_OUT	=	frente		

○ ALVO PERTO 2					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = muitodireita
THEN	SAIDA_OUT	=	muitodireita		
○ ALVO PERTO 3					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = muitoesquerda
THEN	SAIDA_OUT	=	muitoesquerda		
○ ALVO LONGE 1					
If	DISTANCIA_ALVO	=	longe	AND	DIRECAO_ALVO = frente
THEN	SAIDA_OUT	=	frente		
○ ALVO LONGE 2					
If	DISTANCIA_ALVO	=	longe	AND	DIRECAO_ALVO = muitodireita
THEN	SAIDA_OUT	=	muitodireita		
○ ALVO LONGE 3					
If	DISTANCIA_ALVO	=	longe	AND	DIRECAO_ALVO = muitoesquerda
THEN	SAIDA_OUT	=	muitoesquerda		

A Figura 5.38 apresenta o movimento da nave.

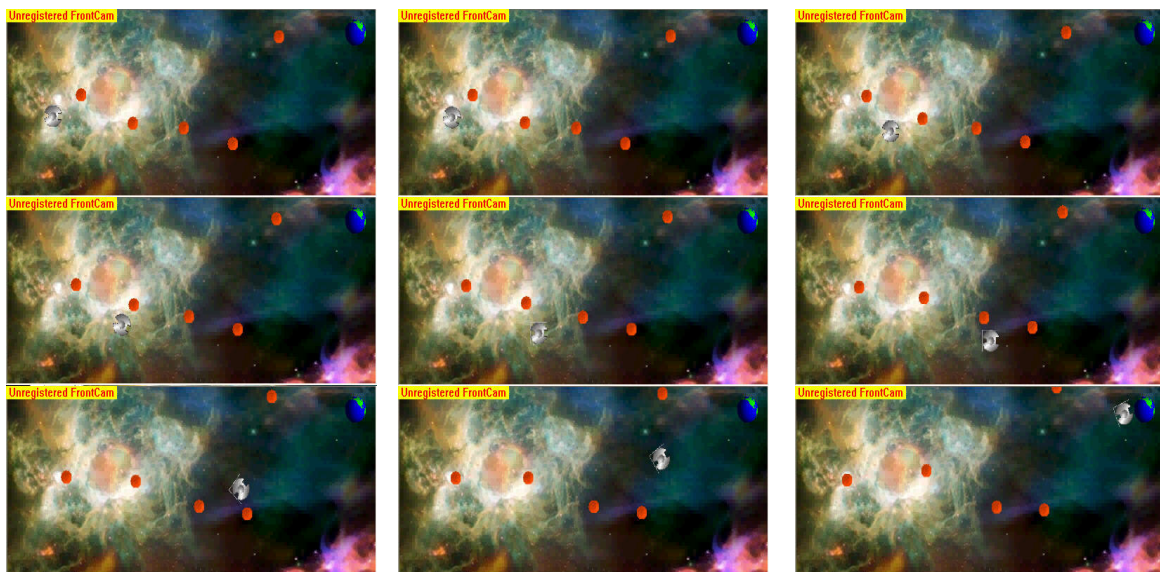


Figura 5.38 - Segunda Simulação do Jogo

O valor final da saída, responsável pela direção que a nave deve seguir, foi multiplicado por uma constante com o objetivo de alterar a sensibilidade no desvio que a nave deva fazer. A sensibilidade dos valores de entrada e saída podem ser alterados de forma a assegurar que o desvio com os meteoros e a trajetória da nave seja a melhor possível.

Multiplicar estes valores significa alterar todos os memberships de forma linear sem a necessidade de reescrevê-los. Por exemplo, multiplicar o valor de saída por uma constante, significa alterar o universo de discurso (entende-se como valores máximo e mínimo da variável e conjunto de memberships) das grandezas na mesma proporção. Pode ser considerado um ajuste de ganho, onde se aumenta ou diminui a amplitude do valor de saída.

É possível, por exemplo, determinar quão perto a nave pode chegar do meteoro antes de desviar, multiplicando o valor da distância por uma constante. Cabe ressaltar que se elevar o valor da distância aumenta-se o quão forte a nave vira para a Terra; em contrapartida a nave pode virar para a Terra não considerando a distância do meteoro.

A vantagem está na possibilidade de alterar o sistema do jogo, como tamanho dos meteoros, da nave somente alterando os pesos. Com isto o sistema é mais fácil de manipular já que independe da distância do universo.

Para esta simulação a nave atinge seu objetivo, aproximando da saída desejada. Se alterado o número de meteoros o sistema também funciona corretamente.

Dados terceira simulação

Para simplificar o conjunto de regras foi identificado visualmente o deslocamento da nave de acordo com a segunda simulação. Verificou-se as regras que estavam influenciando na resposta de forma a melhorar o sistema. A sensibilidade das funções de pertinência foram alteradas, o conjunto de regras otimizado. Os meteoros também estão em movimento.

- Número de meteoros: 8.
- Condição dos meteoros: inseridos randomicamente, tem movimento.

- Número de variáveis de entrada: 4
- Número de variáveis de saída: 1
- Número de memberships: 12
- Número de regras: 7

A figura 5.39 exibe os campos de pertinência das grandezas da segunda simulação.

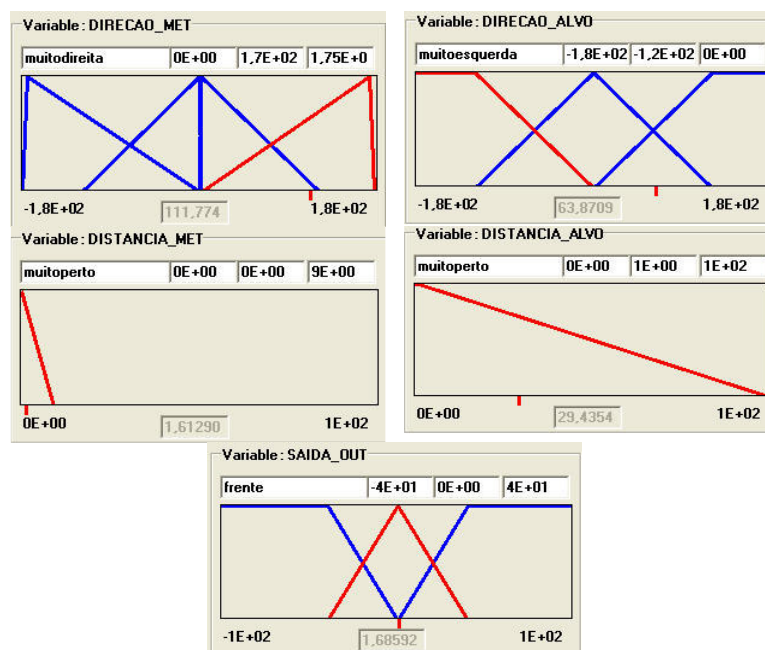


Figura 5.39 - Terceira Simulação do Jogo

Os memberships atribuídos as variáveis distancia_alvo e distancia_alvo foram alterados. Para a Distancia_Met somente o membership “muitoperto” foi considerado, o membership “longe”, visto na segunda simulação foi excluído. A região descoberta não é considerada representativa, já que de acordo com a estratégia do jogo é necessário tomar alguma medida de controle somente quando a nave se encontra próxima ao meteoro. Para a Distancia_Alvo somente o membership “muitoperto” foi considerado (Figura 5.39). Vale

ressaltar que quanto mais perto, maior será o índice de pertinência influenciando diretamente no cálculo do valor de saída.

Com estas funções de pertinência o jogo atinge seu objetivo de forma satisfatória. O número de regras foi reduzido já que somente as distâncias consideradas perto foram ponderadas no sistema. Verificou-se que memberships a mais, como na simulação dois, são considerados desnecessários já que regras da terceira simulação atendem ao sistema. Apesar de não cobrir todas as possibilidades o sistema respondeu perfeitamente.

A Tabela 5.3 apresenta as regras para a terceira Simulação.

Tabela 5.3 Regras da terceira Simulação.

○ METEORO PERTO 1					
If	DISTANCIA_MET	=	muitoperto	AND	DIRECAO_MET = direita
THEN	SAIDA_OUT	=	muitoesquerda		
○ METEORO PERTO 2					
If	DISTANCIA_MET	=	muitoperto	AND	DIRECAO_MET = Esquerda
THEN	SAIDA_OUT	=	muitodireita		
○ METEORO PERTO 3					
If	DISTANCIA_MET	=	muitoperto	AND	DIRECAO_MET = muitoesquerda
THEN	SAIDA_OUT	=	frente		
○ METEORO PERTO 4					
If	DISTANCIA_MET	=	muitoperto	AND	DIRECAO_MET = muitodireita
THEN	SAIDA_OUT	=	frente		
○ ALVO PERTO 1					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = frente
THEN	SAIDA_OUT	=	frente		
○ ALVO PERTO 2					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = muitodireita
THEN	SAIDA_OUT	=	muitodireita		
○ ALVO PERTO 3					
If	DISTANCIA_ALVO	=	muitoperto	AND	DIRECAO_ALVO = muitoesquerda
THEN	SAIDA_OUT	=	muitoesquerda		

O conjunto destas sete regras é responsável pelo comportamento da nave. As 4 primeiras controlam a nave de acordo com distância e direção em relação ao meteoro, as outras verificam a direção e distância relativas ao alvo. A primeira regra, por exemplo,

verifica se o meteoro esta muito perto e se a direção é à direita, neste caso a nave desvia muito à esquerda.

A Figura 5.40 apresenta o movimento da nave.

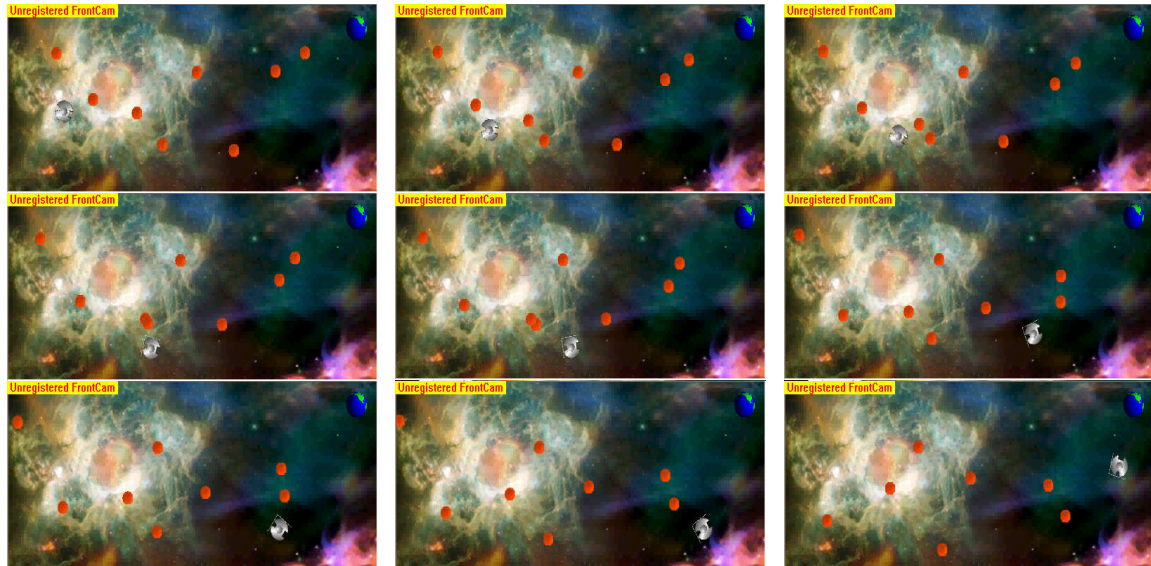


Figura 5.40 - Terceira Simulação do Jogo

A nave desviou corretamente dos meteoros e atingiu o alvo. O conjunto reduzido de regras cobriu toda a extensão do problema e a resposta foi satisfatória.

Cabe ressaltar que como na simulação anterior o valor de saída foi multiplicado por uma constante, alterando assim os universos das grandezas, também as distâncias foram multiplicadas por outra constante, fazendo com que a nave desvie do meteoro quanto estiver bem próximo a ele. O resultado além de atender o objetivo da nave chegar à Terra teve uma excelente trajetória .

5.13 Conclusões

Um conjunto de regras é dito completo se ele responder satisfatoriamente a todas as ocorrências possíveis do problema. Porém para o conjunto de regras ser considerado ideal, depende da correta adequação dos campos de pertinência atribuídos às variáveis.

O excesso de funções de pertinência atribuídas às grandezas pode ser responsável pelo acréscimo das regras levando a inconsistência ou redundância, diminuindo a eficácia da verificação e prejudicando o controle do sistema. Desta forma deve-se verificar a possibilidade de ajustar dos parâmetros, para então reduzir o número de regras.

Visto que o desempenho na primeira simulação não foi considerado satisfatório um novo conjunto de funções de pertinência e regras foram analisadas de forma a atender o esperado. As avaliações foram feitas de acordo com a resposta, no caso, na trajetória da nave e no desvio com os meteoros. O objetivo foi aumentar a eficácia do jogo reduzindo a complexidade do sistema.

Em cada simulação foram verificados os campos de pertinência das grandezas e as regras que estavam sendo utilizadas objetivando uma melhor performance do sistema.

A primeira simulação contou com um número superior de regras, funcionou bem para um número reduzido de meteoros. Porém houve casos de colisões para mais meteoros. O sistema não atendeu ao esperado, já que os desvios com os meteoros não estavam a contento. Verificou-se a necessidade de alterar a base dos dados

A segunda simulação, já com um número inferior de regras obteve o resultado esperado, porém ainda não foi considerado ideal. Para evitar redundância reduziu-se ainda mais o número de regras para a terceira simulação. Esta, com um número maior de obstáculos, teve resultado satisfatório tanto para atingir o objetivo, tanto na trajetória da nave; isto se deve ao melhor ajuste dos campos de pertinência, a alteração da sensibilidade dos valores das distâncias fazendo a nave desviar quando mais próximas aos meteoros; a alteração no valor da saída da nave fazendo com que a trajetória da nave fosse regular e atingisse o objetivo de forma satisfatória.

6 Conclusões Finais

Neste capítulo serão descritas as conclusões referentes à ferramenta e seu emprego. Finalmente serão abordadas sugestões para o aperfeiçoamento da aplicação e recomendações para trabalhos futuros.

O objetivo principal deste trabalho foi o desenvolvimento de um software para aplicações com lógica Fuzzy que tivesse como principal característica a nova forma de representar a matemática através da orientação a objetos.

A proposta foi empregar a ferramenta para auxiliar o uso de Fuzzy em situações de incerteza. No software uma interface gráfica foi desenvolvida a fim de facilitar sua utilização. Esta permite ao usuário gerar grandezas, memberships e regras de controle, de forma ágil e eficaz facilitando o processo.

Apesar da interface ser essencial, esta não é fundamental, já que qualquer aplicação pode se utilizar de bibliotecas geradas pelo sistema. Nela estão todas as funções necessárias para o desenvolvimento de um sistema de lógica Fuzzy. O sistema é independentemente da plataforma uma vez que os dados necessários são armazenados utilizando XML.

Como o sistema foi desenvolvido sob um framework para aplicações distribuídas, é possível utiliza-la como Smartclient ou desenvolver WebServices para seu acesso, deixando-a assim uma ferramenta de uso distribuído pela internet, acessível inclusive a dispositivos móveis.

Um exemplo do uso da ferramenta foi demonstrado no capítulo 5. O objetivo foi criar um sistema de IA para um jogo de nave espacial. A versatilidade e flexibilidade de Fuzzy fazem dela uma excelente ferramenta para o proposto. A aplicação utilizou o sistema Fuzzy para entrar graficamente de forma eficaz com os dados, e também as funções disponibilizadas pela biblioteca para efetuar as entradas.

Foram apresentadas algumas simulações para o comportamento do jogo de acordo as regras de controle e funções de pertinência atribuídas a cada grandeza. O jogo atingiu seu objetivo de forma satisfatória.

A ferramenta segue as etapas de desenvolvimento de um sistema de lógica Fuzzy, sendo confiável para seu propósito. O uso do paradigma orientado a objetos para representar a matemática envolvida, tornou o desenvolvimento mais rápido, compreensível, aproveitando os benefícios que este modelo proporciona.

6.1 Proposta de Desenvolvimento Futuro

Para dar continuidade a este projeto são apresentadas as seguintes sugestões para trabalhos futuros:

Na ferramenta

- Implementar outros tipos de funções de pertinência.
- Adicionar à classe outros métodos de Defuzzificação.
- Desenvolver WebServices de acesso aos dados, podendo a configuração do sistema ser feita de forma distribuída.
- Criar ferramenta de análise de consistência de regras.

- Implementar módulo de DCOM ((Distributed Component Object Model) para acesso distribuído dedicado.
- Implementar módulo, semelhante ao implementado em [16], para a geração da lógica fuzzy em código fonte para microcontroladores diversos.
- Criar módulo para extração de regras a partir de base de dados (*data mining*³).

Em Aplicações Acadêmicas

- Desenvolvimento de uma série de experimentos virtuais (pêndulo invertido, estacionar veículos, controle de temperatura, etc) para a aplicação da lógica fuzzy com realização de simulações com diferentes funções de pertinência e conjuntos de regras.
- Aplicação do sistema como ferramenta de controle no projeto manufatura virtual.
- Aplicação do jogo desenvolvido para guiar um veículo telecontrolado por uma pista de obstáculos utilizando fuzzy e visão computacional.
- Aplicação do sistema em um ambiente de autômatos distribuídos, onde um conjunto de regras centrais possam ser acessadas e alteradas pelos agentes individuais para a realização de tarefas em grupo.

Desta forma, o trabalho desenvolvido deixa em aberto uma série de implementações de curto prazo, auxiliando na pesquisa, ensino e extensão.

³ Data Mining é o processo para explorar grandes quantidades de dados, pode ser considerado uma ferramenta de gerenciamento de informação que deve revelar estruturas de conhecimento, que possam guiar decisões em condições de certeza limitada.

Referências Bibliográficas

- [1] L.A. Zadeh, "Fuzzy sets," *Info. & Ctl.*, Vol. 8, 1965, pp. 338-353.
- [2] Russell. S., Norving. P. , "Artificial Intelligence: A Modern Approach", Earson,1995.
- [3] LARMAN, C. "Utilizando UML e Padrões -Uma Introdução à Análise e Projeto Orientados a Objetos", Bookman, 2000.
- [4] Luzes, A. M. "Visual BASIC.NET - Aplicações Avançadas", Érica, 2001.
- [5] Lima, Edwin. "C# .net - Guia do Desenvolvedor", Campus, 2004.
- [6] Inmon, W.H., "Building the Data Warehouse", John Wiley & Sons Inc. ,USA, 1992.
- [7] Kaufmann, A. (1975), "Theory of Fuzzy Subsets", Academic Press, Inc.
- [8] KLIR G.J., YUAN B. "Fuzzy Sets and Fuzzy Logic: Theory and Applications"
- [9] Zadeh, Lofti A. [1975]. "The concept of a linguistic variable and its application to approximate reasoning, I, II, and III." : *Information Sciences* Vol. 8, pp. 199-249, pp. 301-357, Vol.9, pp. 43-80.
- [10] J.F. Baldwin, "Fuzzy logic and fuzzy reasoning," in *Fuzzy Reasoning and Its Applications*, E.H. Mamdani and B.R. Gaines, (eds.), London: Academic Press, 1981.
- [11] L.A. Zadeh, "Making computers think like people," *I.E.E.E. Spectrum*, 8/1984, pp.26-32.
- [12] CAMARGOS FL," **Lógica Nebulosa: uma abordagem filosófica e aplicada**, UFSC, Florianópolis SC.2002.
- [13] DEITEL, Harvey M. *Web Services: A Technical Introduction*, Ed. Prentice Hall PTR, 2002.
- [14] JAMSA, Kris. *.NET Web Services Solutions*, Ed. Sybex Inc, 2003

- [15] Rumbaugh, J.; Blaha, M.; Premerlani, W. et al. (1994). “Modelagem e Projetos Baseados em Objetos”. Campus, Rio de Janeiro, RJ.
- [16] Honório. L.M, Compilador Fuzzy. 2000. Dissertação de Mestrado – Instituto de Engenharia Elétrica – Departamento de Eletrônica, Universidade Federal de Itajubá, MG.
- [17] Belchior. A. D, Um modelo fuzzy para Avaliação da Qualidade de Software. 1997. Tese de Doutorado - Departamento de Engenharia de Sistemas e Computação (COPPE), Universidade Federal do Rio de Janeiro, RJ.
- [18] TANSCHKEIT, Ricardo, Sistemas Fuzzy. Disponível em: < www.ica.ele.puc-rio.br/cursos/download/LN-Sistemas%20Fuzzy.pdf >. Acesso em: 22/08/2004.
- [19] PEDRYCZ, W. GOMIDE, F. An introduction to fuzzy sets: analysis and design. Bradford. Imprensa Cambridge, 1998. 465p.
- [20] FuzzyTech Tutorial – Online Book – www.fuzzytech.com
- [21] S. Haack, "Do we need fuzzy logic?" Int. Jnl. of Man-Mach. Stud., Vol. 11, 1979, pp.437
- [22] Brule, James F. (1985), “Fuzzy Systems – A Tutorial” , Pacific Northwest National