ALEXANDRE ACÁCIO DE ANDRADE

**DESENVOLVIMENTO DE SISTEMA ESPECIALISTA COM
OPERACIONALIDADE DE APRENDIZADO PARA OPERAR EM TEMPO
REAL COM SISTEMAS INDUSTRIAIS AUTOMATIZADOS**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção do
título de Doutor em Engenharia.

São Paulo

Exemplar original:2007

Exemplar Revisado:2007

ALEXANDRE ACÁCIO DE ANDRADE

**DESENVOLVIMENTO DE SISTEMA ESPECIALISTA COM
OPERACIONALIDADE DE APRENDIZADO PARA OPERAR EM TEMPO
REAL COM SISTEMAS INDUSTRIAIS AUTOMATIZADOS**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção do
título de Doutor em Engenharia.

Área de Concentração:
Engenharia de Automação

Orientador:
Prof. Dr. Sergio Luiz Pereira

São Paulo
Exemplar original:2007
Exemplar Revisado:2007

Este exeplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 21 de Dezembro de 2007

Andrade. Alexandre Acácio

Sergio Luiz Pereira

FICHA CATALOGRÁFICA

Andrade. Alexandre Acácio

Desenvolvimento de Sistema Especialista com Operacionalidade de Aprendizado para Operar em Tempo Real com Sistemas Industriais Automatizados/ A.A. de Andrade, São Paulo, 2007.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Energia e Automação Elétricas.

1.Automação Industrial 2.Sistemas Supervisórios e Sistemas Especialistas - Interligação 3. Aprendizado de Máquina 4 – Mineração de dados. I. Universidade de São Paulo . Escola Politécnica. Departamento de Energia e Automação Elétricas II.

DEDICATÓRIA

À memória de Edile, sua presença será sempre sentida.

A Lina e Cairê por tornarem minha vida mais rica e interessante.

A minha mãe Marisa pelo amor, carinho, amizade, conselho e apoio durante toda a minha vida.

A toda minha família, especialmente a Conrado, o recém chegado que foi amorosamente aguardado.

AGRADECIMENTOS

A Deus pela graça da vida.

A meu orientador, professor e amigo Sergio Luiz Pereira, pelo imprescindível apoio.

Ao amigo Frederico Augusto de Mello Prado, por ter novamente ficado ao meu lado com seu talento.

Aos Profs. Nicolas Lachiche e Peter Flach, que mesmo distantes apoiaram significativamente esse trabalho.

À Rockwell Automation e ao Departamento de Energia e Automação Elétricas da Escola Politécnica da Universidade de São Paulo, pelos subsídios técnicos e pela infra-estrutura providenciada.

Ao Prof. Dr. Cícero Couto de Moraes, pelo permanente e indispensável apoio ao longo da jornada.

Aos colegas do convênio EPUSP/Rockwell Automation do Brasil, pela amizade, paciência e apoio.

A todas as outras pessoas que de alguma forma colaboraram nesse trabalho.

UNIVERSIDADE DE SÃO PAULO

Relatório de defesa pública de Tese do(a) Senhor(a) Alexandre Acácio de Andrade no Programa: Engenharia Elétrica, do(a) Escola Politécnica da Universidade de São Paulo.

Aos 23 dias do mês de novembro de 2007, realizou-se a Defesa da Tese do(a) Senhor(a) Alexandre Acácio de Andrade, apresentada para a obtenção do título de Doutor em Engenharia Elétrica - Área: Sistemas de Potência, intitulada:

"Desenvolvimento de sistema especialista com operacionalidade de aprendizado para operar em tempo real com sistemas industriais automatizados"

Após declarada aberta a sessão, o(a) Sr(a) Presidente passa a palavra aos examinadores para as devidas arguições que se desenvolvem nos termos regimentais. Em seguida, a Comissão Julgadora proclama o resultado:

Nome dos Participantes da Banca	Vínculo do Docente	Sigla da Unidade	Resultado
Sergio Luiz Pereira	Presidente	EP - USP	Aprovado
Selma Shin Shimizu Melnikoff	Titular	EP - USP	Aprovado
Nelson Kagan	Titular	EP - USP	Aprovado
Yaro Burian Junior	Titular	UNICAMP - Externo	Aprovado
Walter Gomes da Cunha Filho	Titular	PUC-SP - Externo	Aprovado

Resultado Final:

Aprovado

Parecer da Comissão Julgadora *

Eu, Elisabete Aparecida F da Silva Ramos _____, Técnico Acadêmico, lavrei a presente ata, que assino juntamente com os(as) Senhores(as). São Paulo, aos 23 dias do mês de novembro de 2007.

Selma Shin Shimizu Melnikoff

Yaro Burian Junior

Nelson Kagan

Walter Gomes da Cunha Filho

Sergio Luiz Pereira
Orientador(a)

* Obs: Se o candidato for reprovado por algum dos membros, o preenchimento do parecer é obrigatório.

Título homologado pela Comissão de Pós-Graduação em _____

Presidente da Comissão de Pós-Graduação

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

RESUMO

ABSTRACT

1	INTRODUÇÃO	1
1.1	SISTEMAS SUPERVISÓRIOS E AUTOMAÇÃO DE PROCESSOS	1
1.2	OBJETIVOS DESTES TRABALHOS DE PESQUISA	6
1.3	SISES - SISTEMA DE INTEGRAÇÃO DE SISTEMAS ESPECIALISTAS E SUPERVISÓRIOS	7
1.3.1	Arquitetura de hardware e arquitetura de software do SISES	10
1.3.2	Módulo de interface com sistemas supervisórios (MISS)	10
1.3.3	Módulo de Construção de Sistemas Especialistas (M.C.S.E.)	11
1.3.4	Considerações Gerais Sobre o Emprego de SS Operando com S.E.s	12
1.3.5	Características operacionais e relacionais de SE com a atividade da Engenharia do Conhecimento	13
2	TÓPICOS DE APRENDIZADO COMPUTACIONAL E DE APRENDIZADO DE MÁQUINA.....	16
2.1	INTRODUÇÃO.....	16
2.2	APRENDIZADO COMPUTACIONAL	18
2.2.1	Aprendizado-PAC - Probably Approximately Correct	20
2.3	APRENDIZADO DE MÁQUINA (A.M.)	25
2.3.1	Componentes básicos de um elemento de desempenho	26
2.3.2	Tipo de realimentação para aprendizado de máquina.....	27
2.3.3	Tipo de representação para conhecimento aprendido	28
2.3.4	Aprendizagem indutiva.....	30
2.4	DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS (KNOWLEDGE DISCOVERY IN DATABASES - KDD).....	32
2.4.1	Introdução à descoberta de conhecimento em bases de dados	32
2.4.2	O processo de aquisição de conhecimento.....	33
2.4.3	Armazéns de dados	34
2.5	MINERAÇÃO DE DADOS - DATA MINING.....	35
2.5.1	Introdução à mineração de dados.....	35
2.5.2	Principais tipos de dados de entrada na Mineração de Dados (M.D.).....	36
2.5.3	Principais tipos de dados de saída na Mineração de dados - Representação de Conhecimento.....	38
2.5.4	Principais tipos de tarefas tratadas pela mineração de dados baseadas no tipo de dado a ser minerado.....	41
2.5.5	Análise comparativa entre Aprendizado de Máquina e Mineração de dados.....	42
2.6	APRENDIZAGEM EM ÁRVORES DE DECISÃO	44
2.6.1	A estrutura das árvores de decisão	47
2.6.2	Limitações entre explicação x entendimento	47
2.7	APRENDIZAGEM POR REGRAS DE ASSOCIAÇÃO	49
2.7.1	Suporte e confiança	50

2.7.2	<i>Algoritmo A priori</i>	52
2.7.3	<i>Algoritmo Tertius</i>	56
3	ARQUITETURA DE SOFTWARE DE NEURAL SISTEMA DE INTEGRAÇÃO DE SISTEMAS ESPECIALISTAS E SUPERVISÓRIOS “NESISES”	59
3.1	MACRO-ARQUITETURA DE SOFTWARE DO NESISES	60
3.2	MÓDULO COGNITIVO META SISES	61
3.3	CONSIDERAÇÕES SOBRE O MÓDULO DECISÓRIO DO NESISES	62
4	METODOLOGIA DE TESTES E VALIDAÇÃO DO NEURAL SISTEMA DE INTEGRAÇÃO DE SISTEMAS ESPECIALISTAS E SUPERVISÓRIOS “NESISES”	65
4.1	INTRODUÇÃO.....	65
4.2	TESTES DE SIMULAÇÃO.....	66
4.2.1	<i>Testes de simulação para determinação do grau de assertividade – duas variáveis</i>	67
4.2.2	<i>Testes de simulação para determinação do grau de assertividade - três variáveis</i>	76
4.2.3	<i>Testes de simulação de processo industrial automatizado</i>	82
4.2.4	<i>Análise de rastreabilidade dos testes de simulação de processo industrial automatizado</i>	82
4.2.5	<i>Avaliação dos testes de simulação de processo industrial automatizado</i>	84
4.2.6	<i>Análise de interface dos testes de simulação de processo industrial automatizado</i>	84
4.2.7	<i>Geração de plano e desenho de teste dos testes de simulação de processo industrial automatizado</i> 86	
4.3	DEFINIÇÃO DOS CRITÉRIOS DE APROVAÇÃO DOS TESTES DE SIMULAÇÃO COM O NESISES.....	87
4.3.1	<i>Aprendizado assertivo dos testes de simulação de processo industrial automatizado</i>	87
4.4	ANÁLISE DOS RESULTADOS SEGUNDO OS CRITÉRIOS DE APROVAÇÃO DOS TESTES DE SIMULAÇÃO COM O NESISES	94
4.5	TESTES DE CAMPO COM O NESISES	95
4.5.1	<i>Objetivos dos testes de Campo com o NESISES</i>	95
4.5.2	<i>Descritivo do processo automatizado empregado para testes de campo do NESISES</i>	95
4.5.3	<i>Aprendizado assertivo dos testes de campo com o NESISES</i>	101
4.5.4	<i>Análise de rastreabilidade dos testes de campo</i>	115
4.5.5	<i>Avaliação dos testes de campo</i>	116
4.5.6	<i>Análise de interface dos testes de campo</i>	116
4.5.7	<i>Geração de plano e desenho de teste dos testes de campo</i>	117
4.6	DEFINIÇÃO DOS CRITÉRIOS DE APROVAÇÃO DOS TESTES DE CAMPO COM O NESISES	118
4.7	ANÁLISE DOS RESULTADOS SEGUNDO OS CRITÉRIOS DE APROVAÇÃO DOS TESTES DE CAMPO COM O NESISES	119
5	CONCLUSÕES E CONSIDERAÇÕES FINAIS	120
6	REFERÊNCIAS BIBLIOGRÁFICAS	123
ANEXO 1		127
METODOLOGIA DE TESTES PRELIMINARES DO ALGORITMO TERTIUS		127
ANEXO 2		135
PROGRAMA LADDER ELABORADO PARA OS TESTES DE SIMULAÇÃO		135

LISTA DE TABELAS

Tabela 1.1- Exemplos de dispositivos constituintes nos níveis de controle industrial.....	2
Tabela 2.1– Principais formas de representação de conhecimento que são utilizadas em A.M.	29
Tabela 2.2 Principais tipos de representação de conhecimento em M.D. e seus principais usos. (Ye,2003)	40
Tabela 2.3 – Sistemas comerciais utilizados para mineração de dados com regras de associação	51
Tabela 4.1 – Base de dados completa das variáveis V1, V2, V4, V5, V7 e V8 duas a duas	68
Tabela 4.2 – Primeiro conjunto simulado de variáveis 2 a 2	74
Tabela 4.3 – Segundo conjunto simulado de variáveis 2 a 2	74
Tabela 4.4 – Terceiro conjunto simulado de variáveis 2 a 2	74
Tabela 4.5 – Sintaxe das regras do NESISES.....	75
Tabela 4.6 – Regras encontradas pelo NESISES.....	75
Tabela 4.7 – Primeiro conjunto simulado de regras NESISES.....	76
Tabela 4.8 – Segundo conjunto simulado de regras NESISES.....	76
Tabela 4.9 – Terceiro conjunto simulado de regras NESISES	76
Tabela 4.10 – Quarto conjunto simulado de regras NESISES.....	77
Tabela 4.11 – Quinto conjunto simulado de regras NESISES.....	77
Tabela 4.12 – Regras encontradas pelo NESISES para o primeiro conjunto simulado.....	77
Tabela 4.13 – Regras encontradas pelo NESISES para o segundo conjunto simulado	78
Tabela 4.14 – Regras encontradas pelo NESISES para o terceiro conjunto simulado	79
Tabela 4.15 – Regras encontradas pelo NESISES para o quarto conjunto simulado	80
Tabela 4.16 – Regras encontradas pelo NESISES para o quinto conjunto simulado	81
Tabela 4.17 – Padrões simulados para o NESISES.....	91
Tabela 4.18 – Principal regra simulada para o NESISES.....	91
Tabela 4.19 – Regras encontradas pelo NESISES.....	92
Tabela 4.20 – Variáveis adquiridas pelo NESISES para análise dos dados de campo.....	102
Tabela 4.21 – Cruzamentos de variáveis utilizadas para busca de regras nos dados de campo.....	103
Tabela 4.22 – Regras encontradas pelo NESISES para todas as variáveis do sistema.....	104
Tabela 4.23 – Regras encontradas pelo NESISES para as variáveis Armazém -2 / Recepção	105
Tabela 4.24 – Regras encontradas pelo NESISES para as variáveis Armazém-4 / Recepção	106
Tabela 4.25 – Regras encontradas pelo NESISES para as variáveis Armazém -6 / Recepção	107
Tabela 4.26 – Regras encontradas pelo NESISES para as variáveis Pátio / Recepção	108
Tabela 4.27 – Regras encontradas pelo NESISES para as variáveis Pátio / Tulha.....	109
Tabela 4.28 – Regras encontradas pelo NESISES para as variáveis Tulha / Armazém 2	110
Tabela 4.29 – Regras encontradas pelo NESISES para as variáveis Tulha / Armazém 4	111
Tabela 4.30 – Regras encontradas pelo NESISES para as variáveis Tulha / Armazém 6	113

LISTA DE FIGURAS

Figura 1 - Níveis de controle industrial	3
Figura 2 - Visão parcial de um sistema supervisório de um processo siderurgico de elevada complexidade.....	5
Figura 3 - Atuação dos operadores nos sistemas automatizados	8
Figura 4 - Exemplo de atuação do SISES em um sistema supervisório inserido em um sistema de automação.....	9
Figura 5 - Comunicação do SISES com o RSView32	11
Figura 6 - Arquitetura básica dos SEs gerados no MCSE.....	12
Figura 7 - A funcionalidade de um SE ao longo do tempo.....	14
Figura 8 - Hierarquia das principais técnicas de aprendizado computacional / de máquina.....	17
Figura 9 - Funcionalidade o erro obtido a partir de uma hipótese $h(x_i)$	22
Figura 10 - Diagrama de Espaço de Hipóteses, mostrando a ϵ -ball em torno da verdadeira função f	24
Figura 11 - Principais aspectos do aprendizado de máquina	26
Figura 12 - Tipos de realimentação para aprendizado de máquina.....	27
Figura 13 - Exemplo de aprendizagem por reforço	28
Figura 14 - O processo de aquisição de conhecimento em bases de dados	34
Figura 15 - Ilustração da multidisciplinariedade da mineração de dados	36
Figura 16 - Ilustração de atributos, instâncias e conceito em M.D.	37
Figura 17 - Árvore de decisão sobre exemplo do carro esportivo	46
Figura 18 – Algoritmo para geração itemsets frequentes(pseudo código).(Agrawal 1993).....	53
Figura 19 – Função Apriori-gem, geração de itemsets candidatos (pseudo código)..(Agrawal 1993).	54
Figura 20 - Algoritmo Apriori em sua forma completa (pseudo código)..(Agrawal 1996).	55
Figura 21 – Algoritmo Tertius (pseudo código).	57
Figura 22 - Estrutura em forma de treliça do espaço de busca no algoritmo Tertius.....	58
Figura 23 - A funcionalidade do NESISES ao longo do tempo.....	60
Figura 25 - Arquitetura de software do M.C.M.S.....	62
Figura 26 - Operacionalidade dos módulos do NESISES	63
Figura 27 – Fluxograma operacional do software NESISES.....	64
Figura 28 - - Fluxograma de testes de validação do NESISES.....	65
Figura 29 - - Metodologia de testes e validação IEEE 1059 – 1993.....	66
Figura 30 - Fluxograma da rastreabilidade dos testes de simulação de processo industrial automatizado	84
Figura 31 – Interface do NESISES – módulo de mineração de dados na fase dos testes de simulação de processo industrial automatizado.....	85
Figura 32 – Plano e desenho de testes do NESISES na fase dos testes de simulação de processo industrial automatizado	86
Figura 33 – Processo de secagem de náilon	88
Figura 34 – Arquitetura básica dos testes de simulação	89
Figura 35 – Arquitetura de hardware do processo de armazenagem	97
Figura 36 – Arquitetura geral do processo de armazenagem automatizado.....	98
Figura 37 – Vista geral de tulha rodoviária automatizada	100
Figura 38 - Fluxograma da rastreabilidade dos testes de campo	116
Figura 39 - - Interface do NESISES – módulo de mineração de dados - testes de campo.....	117
Figura 40 - Plano e desenho de teste do NESISES - testes de campo	118
Figura 41 - Metodologia utilizada para testes preliminares – Tertius.....	128
Figura 42 - Utilização do algoritmo Tertius na plataforma WEKA.....	128

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina.
CAP	Controlador de Automação Programável
CCM	Centro de Controle de Motores
CLP	Controlador Lógico Programável
DLL	<i>Dynamic Link Library</i>
ERP	<i>Enterprise Resource Planning</i>
IHM	Interface Homem-Máquina
KDD	<i>Knowledge Discovery in Databases</i> - Descoberta de conhecimento em bases de dados.
MCSE	Módulo de Construção de Sistemas Especialistas.
MD	Mineração de Dados
MISS	Módulo de Interface com Sistemas Supervisórios
MRP	<i>Material Requirement Planning</i>
NESISES	Neural Sistema de Integração de Sistemas Especialistas e Supervisórios
OLAP	Online Analytical Processing (processamento analítico on-line).
PAC	Aprendizado Provavelmente Correto (<i>Probably Approximately Correct</i>).
PEA	Departamento de Energia e Automação da Escola Politécnica da USP
SDCD	Sistema Digital de Controle Distribuído
SE	Sistemas Especialistas
SISES	Sistema de Integração de Sistemas Especialistas e Supervisórios
SS	Sistemas Supervisórios
SFC	<i>Sequential Flow Chart</i>

RESUMO

Os Sistemas Supervisórios (SS) executam diversas funções vitais em um processo automatizado e também operam como interface homem-máquina. Os mesmos recebem informações de dispositivos como Controladores Lógicos Programáveis (CLP), inversores de frequência, etc, e ao mesmo tempo enviam parâmetros de controle fornecidos pelos operadores do processo aos equipamentos de controle.

Na operação de SS, a atuação correta e a experiência dos operadores humanos é portanto também vital no controle do processo automatizado. Em recente trabalho(Andrade-2001) de pesquisa na Escola Politécnica da USP foi desenvolvido um Sistema Especialista para operar em tempo real com Sistemas Supervisórios para auxiliar na tomada de decisão dos operadores do sistema. Ao longo do tempo de operação de uma planta automatizada ocorrem novas situações que passam a compor os novos cenários do sistema e também contribuem para o aumento do conhecimento e da experiência dos operadores humanos.

Assim sendo, Sistemas Especialistas constantemente devem ser atualizados com novas regras para atender às novas demandas da planta automatizada.

Este trabalho de pesquisa apresenta os resultados obtidos com o Sistema Especialista desenvolvido para operar em tempo real com Sistemas Supervisórios, como também o andamento das pesquisas no campo de aprendizado de máquina e mineração de dados com o objetivo de desenvolver e de habilitar Sistemas Especialistas que operam em tempo real com a capacidade de aprender conforme ocorrem eventos durante o funcionamento de uma planta industrial automatizada.

ABSTRACT

Supervision systems (SS) perform diverse vital functions in an automated process and also operate as a man machine interface. The SS receive informations from devices as programmable logical Controllers (PLC), frequency inverters , etc, and at the same time send parameters of control supplied by the process users to the control equipment.

In the SS operation, the correct actions and the experience of the human users are therefore vital in the control of the automated process. In a recent research work(Andrade-2001) of the USP Polytechnic School an Expert System was developed to interact in real time with a SS aiming to help in the decision-making process of the system users. During the operation time of an automated plant new situations occur, which come to compose the new system settings and also contribute to the increasing of the human users' knowledge and experience.

Thus, Expert Systems should be constantly brought up to date with these new rules for attend the new demands of the automated plant.

This research work shows the results obtained by the expert system, developed to operate in real time with the supervisory systems. It also shows the course of the researches in the fields of machine learning and data mining with the objective of developing and enabling Expert Systems that operate in real time with the capacity to learn events as they happen during the operation of an automated industrial plant.

1 INTRODUÇÃO

Este capítulo apresenta um breve relato sobre o desenvolvimento e aplicação de Sistemas Especialistas (SEs) em plantas automatizadas. Há também as limitações operacionais de SEs operando ao longo do tempo, as motivações e as justificativas do porquê da necessidade de investimentos em pesquisas na área de aprendizado de máquina para o desenvolvimento e implementação de SEs com capacidade de aprendizado

1.1 Sistemas Supervisórios e automação de processos

Sistemas Supervisórios (SS) (Gomi, 1989 p.3) são definidos como sistemas com capacidade de exercer controle sobre um dado sistema físico e verificar o seu desempenho de acordo com a ação desejada. SS são implementados computacionalmente e são dotados de uma Interface Homem - Máquina (IHM), que permite a entrada de parâmetros manualmente, além de poderem gerar relatórios e principalmente representar pictoricamente os processos de forma geral.

Os SS estão cada vez mais presentes em plantas industriais automatizadas e cada vez tomam papel de destaque em sua operação por meio da intensa interação com os operadores do processo.

Uma das formas teóricas (Webb et al.,1992) (Castrucci, Moraes, 2007) de classificar o grau de automação nos processos produtivos industriais é a pirâmide da automação industrial, composta por 5 níveis: Nível 1 – Máquina, Nível 2 – Estação, Nível 3 – Célula, Nível 4 – Central, Nível 5 – Planta.

A Tabela 1.1 apresenta alguns exemplos de dispositivos e de produtos de aplicação nos diversos níveis.

Tabela 1.1- Exemplos de dispositivos constituintes nos níveis de controle industrial

Nível	Exemplos de dispositivos
1	Botoeiras, Chaves de emergência, Sensores de temperatura, pressão, nível, umidade, opacidade, PH, movimento; medidores de vazão etc.
2	Controladores Lógicos CLP (Controlador Lógico Programável); Controladores digitais do tipo “single-loop” ; SDCCD(Sistema Digital de Controle Distribuído); Inversores de frequência e demais drivers de acionamento
3	Sistemas Supervisórios(SS), Simuladores de processos, Sistemas de operação por batelada e demais Interfaces Homem -Máquina industriais..
4	É definido pela exploração dos sistemas MRP(<i>Material Requirement Planning</i>), <i>Just-in-time</i> e o MRP II (<i>Manufacturing Resource Planning</i>).
5	Sistemas ERP(<i>Enterprise Resource Planning</i>), Gestão de Recursos Corporativos): SAP, etc.

Figura 1 ilustra os níveis de automação industrial segundo a classificação adotada.

Figura 1 - Níveis de controle industrial

Dentro da classificação adotada neste trabalho de pesquisa, os Sistemas Supervisórios (SS) operam no nível célula recebendo informações de dispositivos controladores como Controladores Lógicos Programáveis (CLP), inversores de frequência, etc. Os SS executam diversas funções vitais em um processo automatizado e também operam como Interface Homem Máquina (IHM), com operadores humanos para que os mesmos possam analisar, diagnosticar, tomar decisões e interferir no processo. Os operadores dos sistemas automatizados passam boa parte do tempo de trabalho operando os SS e conseqüentemente atuando nos processos controlados por meio dos SS. Assim sendo, os operadores normalmente permanecem em salas de supervisão e de controle e detêm a grande responsabilidade de supervisionar, alterar receitas, *set-points*, inferir conclusões,

tomar decisões conforme procedimentos pré-estabelecidos e atuar em situações emergenciais. Deve-se ressaltar que a atuação dos operadores é classificada na teoria geral da automação como um processo manual.

Atualmente as plantas industriais automatizadas podem atingir altos níveis de complexidade podendo ter em alguns casos até 15.000 pontos de comunicação (troca de dados entre sinais de campo e controladores, denominados na literatura técnica e comercial de *tags*). Sistemas de automação demasiadamente complexos dificilmente podem ser supervisionados de uma forma eficiente por um único operador (Kaszkurewicz et al.,1997). Assim sendo, é comum que as diretrizes técnicas das empresas detentoras de sistemas de automação relativamente complexos, com mais de 3000 *tags*, designem diversos operadores para que cada um monitore e controle sub-sistemas específicos da planta industrial automatizada. Como consequência os operadores acabam tornando-se especialistas em sub-áreas do processo e operam concomitantemente o SS por meio de diversas IHMs.

A dedicação, a atenção, o conhecimento, a experiência, o tempo de resposta e a assertividade dos operadores humanos são fundamentais para que todos os níveis de qualidade especificados para o processo sejam atingidos. Assim sendo, percebe-se que a interação entre os operadores e os SS que ocorre principalmente no nível célula, sendo um processo manual, está sujeito a todos os tipos de ocorrências decorrentes dos processos não automatizados. A Figura 2 mostra um SS de uma planta siderúrgica, setor de sinterização, nota-se que o SS é de elevada complexidade.

Figura 2 - Visão parcial de um sistema supervisorio de um processo siderurgico de elevada complexidade

1.2 Objetivos deste trabalho de pesquisa

O problema de pesquisa proposto neste estudo é a criação e validação de ferramenta computacional com capacidade de aprendizado de forma a diminuir a dependência do sistema de automação de seus operadores bem como fornecer subsídios a engenharia de conhecimento para a elaboração de sistemas especialistas operando com sistemas de automação industrial.

Por se tratar de um estudo exploratório, optou-se pela elaboração de metodologia para a criação de solução de software dotado de capacidade de aprendizado com um sistema de automação industrial.

A solução de software deverá:

- Aplicar tecnologias de aprendizado computacional em sistemas de automação industrial.
- Armazenar o conhecimento aprendido com o processo automatizado para auxílio a sua operação.
- Diminuir a dependência aos operadores em sistemas industriais automatizados.
- Reduzir o numero de intervenções da Engenharia do Conhecimento em Sistemas Especialistas operando com Sistemas de Automação Complexos.
- Ser testada em condições o mais próximo possível de processos industriais complexos automatizados.

Para o desenvolvimento do software contou-se com a solução estudo de Andrade (2001) em que foi elaborada solução de software para funcionamento de sistemas especialistas *on-line* com sistemas supervisórios de automação industrial, intitulado de SISES (Sistema de Integração Sistemas Especialistas e Supervisórios). O mesmo será modificado e ampliado para agregar funcionalidades de aprendizado evoluindo assim para o NESISES. (Neural Sistema de Integração Sistemas Especialistas e Supervisórios).

No capítulo um é introduzido o problema de pesquisa com um breve resumo do desenvolvimento do SISES e suas limitações.

O capítulo dois é explicitado as pesquisas bibliográficas desenvolvida sobre aprendizado de maquina, seu embasamento teórico que culmina com a escolha da mineração de dados como foco de interesse.

O capítulo três é apresentada a arquitetura de software que foi elaborado para a realização do NESISES, optou-se pelo uso do algoritmo de mineração de dados Tertius e desenvolvimento próprio com base na linguagem Dephi.

O capítulo quatro é apresentada a metodologia de testes e validação do software desenvolvido sendo que foram elaborados testes de simulação e também de campo em planta industrial automatizada.

No capítulo cinco são tecidas as considerações finais, contribuições deste trabalho e propostas para estudos futuros.

1.3 SISES - Sistema de Integração de Sistemas Especialistas e Supervisórios

A solução de software proposta por ANDRADE (2001) foi desenvolvida e testada em um ambiente de criação e de aplicação de sistemas especialistas denominado Sistema de Integração de Sistemas Especialistas e Supervisórios "SISES". O SISES foi elaborado para operar em tempo real com SS para aconselhar e também se necessário automatizar as funções executadas pelos operadores humanos.

A Figura 3 ilustra em linhas gerais a atuação do operador de um sistema industrial automatizado. Nota-se que essa atuação pode ocorrer principalmente nos três níveis mais baixos da pirâmide de automação, de acordo com o seguinte ciclo:

- O operador percebe as informações do processo no SS por meio de seus sentidos;
- O operador utiliza suas capacidades cerebrais para análise e interpretação dos dados, tomando ou não decisões;
- Tendo decidido, o operador atua no processo alterando parâmetros de operação que podem ser desde auxilio e diagnostico de manutenção (nível 1), verificações de lógicas de funcionamento (nível 2) ou principalmente operando diretamente o S.S.

Figura 3 - Atuação dos operadores nos sistemas automatizados

A Figura 4 ilustra em linhas gerais a atuação do SISES em um SS inserido em um sistema de automação.

O SISES possibilita duas formas de operação:

- Operar como um substituto total ou parcial dos operadores humanos nos processos decisórios;
- Operar em tempo real como um conselheiro dos operadores do processo.

Figura 4 - Exemplo de atuação do SISES em um sistema supervisório inserido em um sistema de automação

1.3.1 Arquitetura de hardware e arquitetura de software do SISES

O SISES (Pereira e Andrade, 2004) é uma ferramenta computacional ou um ambiente de desenvolvimento de SEs que se comunicam com SSs. O SISES é composto de dois macroblocos: Módulo de Interface com Sistemas Supervisórios (MISS) e Módulo de Construção de Sistemas Especialistas (MCSE)

1.3.2 Módulo de interface com sistemas supervisórios (MISS)

O MISS é o módulo de comunicação do Sistema Especialista (SE) com o SS. Como a maioria dos SS atuais opera com o sistema operacional Windows®, e como também o SS escolhido foi o RSView® da Rockwell Software, o SISES foi elaborado visando à compatibilidade com estes significativos produtos do mercado mundial. Entretanto, os fundamentos científicos e metodológicos do SISES podem ser aplicados e desenvolvidos para quaisquer outros ambientes operacionais e SS. O ambiente de desenvolvimento de SS escolhido para operar com o SISES neste trabalho de pesquisa possibilita diversas formas de comunicação com o Windows. Entretanto, a comunicação mais eficiente é efetuada pelas (DLLs) *Dynamic Link Library* específicas: RsvApplication.dll e RsvProject.dll. Tal fato ocorre uma vez que qualquer outra forma de comunicação com o RSView utiliza indiretamente essas mesmas DLLs, e, portanto acrescenta mais etapas na comunicação SISES com RSView (Rockwell Software, 2003).

A Figura 5 ilustra como ocorrem as diversas interações do RSView e o SISES, destacando-se a funcionalidade do MISS que está representada pelo laço de comunicação **RSView – Objetos VCL e Delphi**.

Figura 5 - Comunicação do SISES com o RSView32

1.3.3 Módulo de Construção de Sistemas Especialistas (M.C.S.E.)

O MCSE possui um ambiente onde são construídos SEs. A

Figura 6 ilustra a organização do MCSE, que é composto de três módulos:

- Base de conhecimento: é o conjunto de regras obtido pelo engenheiro do conhecimento e armazenado no arquivo do SISES;
- Editor de bases: é a ferramenta que permite a edição e alteração da Base de Conhecimento;
- Máquina de inferência: módulo do SE que efetua as deduções e conclusões sobre a base de conhecimento. No SISES a Máquina de Inferência opera empregando o algoritmo de encadeamento para trás (*backward chaining*) (Buchanan et al.,1985).

Figura 6 - Arquitetura básica dos SEs gerados no MCSE

1.3.4 Considerações Gerais Sobre o Emprego de SS Operando com S.E.s

O Sistema Especialista desenvolvido foi testado por meio de simulações e em campo por mais de 300 horas com o objetivo de ser validado como ferramenta viável de integração de Sistemas Especialistas com Sistemas Supervisórios.

Tanto nos testes de simulação quanto nos testes de campo o SE funcionou em tempo real operando juntamente com um sistema de supervisão e controle desenvolvido para um processo industrial de produção de náilon (Andrade, 2001 ,p. 90).

O SISES foi testado e validado sobre os seguintes critérios:

- *Critério de desempenho global do sistema:* o sistema como um todo deve funcionar a contento;
- *Critério de desempenho do S.S.:* O SS não deve sofrer queda de desempenho apreciável;
- *Critério de tempo de execução:* o tempo para o SE chegar a uma conclusão não pode ser de uma ordem de grandeza superior aos

tempos padronizados que os operadores humanos têm para atingir as mesmas conclusões;

- *Critério de compatibilidade:* o SISES não pode requerer tantos recursos do sistema operacional (Windows) que prejudique seu desempenho;
- *Critério de assertividade:* o SE tem de chegar a resultados corretos.

Os testes de desempenho do SISES forneceram fortes indícios que SEs podem ser empregados como poderosas ferramentas de auxílio na automação industrial, sendo inclusive muito mais rápidos que os operadores humanos para tomada de decisão.

1.3.5 Características operacionais e relacionais de SE com a atividade da Engenharia do Conhecimento

Deve-se observar que as necessidades operacionais de um sistema automatizado não são estáticas. Isto ocorre porque as plantas industriais sofrem alterações ao longo do tempo e também porque os algoritmos dos processos podem ser alterados conforme necessidades de ordem geral.

Assim sendo, o conhecimento para operar um SS de uma planta industrial automatizada é dinâmico. Os operadores dos sistemas automatizados, de forma geral, também acumulam conhecimentos ao longo do tempo de operação do sistema.

A Figura 7 ilustra como se dá ao longo do tempo a discrepância entre o conhecimento registrado em um SE dedicado para a automação industrial e o conhecimento adquirido com a operação do processo pelos operadores humanos.

Inicialmente o SE é produzido com as diversas atividades da engenharia do conhecimento, como resultado dessas atividades surge um SE oprando com o sistema de automação, porem ao longo do tempo ocorrem alterações na planta/sistema industrial que tornam o SE incompleto ou obsoleto, tornando necessária nova intervenção da engenharia de conhecimento.

Para que não haja uma grande discrepância entre o conjunto de conhecimentos registrados em um SE por meio de regras e por meio da base de

conhecimento, fazem-se necessárias, portanto, constantes atualizações após “n” macro-seqências de eventos transcorridos nas plantas automatizadas. Estas atualizações necessitam que a atividade exercida pela engenharia do conhecimento seja efetuada recorrentemente.

Assim sendo nota-se que, apesar de SEs poderem ser empregados como ferramentas muito úteis para a automação industrial, os mesmos necessitam de constantes atualizações por parte da Engenharia do conhecimento para que as Bases de conhecimento reflitam, de maneira fiel, os conhecimentos do sistema automatizado. Como a decisão de atualizar o SE pode ocorrer em qualquer determinado instante de tempo, existe a probabilidade de haver uma grande discrepância entre o SE implantado e o conhecimento necessário para lidar com a planta industrial automatizada.

Figura 7 - A funcionalidade de um SE ao longo do tempo

Este trabalho de pesquisa apresenta a base teórica, a metodologia de desenvolvimento da arquitetura de hardware, a arquitetura de software e também

dos testes de validação do **NESISES** - “Neural Sistema de Integração de Sistemas Especialistas e Supervisórios”.

O NESISES desenvolvido e apresentado neste trabalho de pesquisa executa dinamicamente e continuamente a atividade da engenharia do conhecimento empregando técnicas da ciência da inteligência artificial voltada para o aprendizado de máquina. O capítulo 3 ilustra a operacionalidade do NESISES.

2 TÓPICOS DE APRENDIZADO COMPUTACIONAL E DE APRENDIZADO DE MÁQUINA

2.1 Introdução

Este capítulo apresenta um breve resumo teórico e os principais tópicos da área de aprendizado computacional e aprendizado de máquina. A ciência do aprendizado computacional tem diversas ramificações, sendo cada uma delas objeto de pesquisas variadas. Neste trabalho de pesquisa optou-se pelo estudo de alguns ramos do aprendizado computacional considerados mais adequados à resolução dos problemas relativos à operacionalidade de Sistemas Especialistas operando em tempo real com Sistemas Supervisórios no controle de processos industriais. A Figura 8 ilustra os principais ramos do aprendizado computacional.

Os blocos tracejados na Figura 8 (Aprendizagem Bayesiana, Aprendizagem em Redes Neurais e Aprendizagem em árvores de decisão) representam as principais técnicas utilizadas na mineração de dados que será de suma importância para esse trabalho.

Figura 8 - Hierarquia das principais técnicas de aprendizado computacional / de máquina

2.2 Aprendizado computacional

Desde o começo da história da Inteligência Artificial (Turing 1950), especula-se sobre a possibilidade das máquinas computacionais terem ou não capacidade de desenvolver as habilidades de inteligência e de aprendizado. O aprendizado computacional como ramo organizado da ciência surgiu com o artigo seminal de G. Valiant (1984) onde um modelo teórico de aprendizado, suficientemente geral, foi proposto. Com o modelo de Valiant como ponto de partida, a seguinte definição é aceita: “O aprendizado computacional é a teoria que estuda uma coleção de modelos matemáticos de aprendizado de máquina, e tem entre seus objetivos o desenvolvimento de novos algoritmos para aprender a partir de dados.”(Vapnik, <http://www.learningtheory.org>).

Esse ramo da ciência da computação se aproxima de outras disciplinas com objetivos semelhantes, tal como Estatística e Aprendizado de Máquina. No entanto, seu foco principal é na eficiência computacional (do ponto de vista de formalismo matemático) e a complexidade computacional.

A título de exemplo, é mostrado em linhas gerais o modelo proposto por Valiant. Esse modelo baseia-se em três premissas fundamentais:

- *O Aprender é probabilístico (processo aleatório de geração de dados):* Uma amostra finita de dados (dos quais deve-se aprender algo) é gerada randomicamente; essa metodologia necessariamente gera erros que serão mantidos dentro de uma margem, que será quantificada durante a elaboração das hipóteses por um algoritmo de aprendizado;
- *Algoritmos eficientes:* Do ponto de vista computacional significa que seu tempo de execução tem de ser, no máximo, da mesma ordem de grandeza de um polinômio de grau igual ao tamanho da amostra que tem de ser aprendida. Por exemplo, dada uma amostra m de observações do processo aleatório, o tempo de execução de um bem sucedido algoritmo de aprendizado será limitado por um polinômio de grau m ;

- *Algoritmos gerais*: Um algoritmo deve ser capaz de aprender sob uma grande variedade de dados de entrada.

No artigo original de Valiant, o processo aleatório de geração de dados consistiu numa distribuição desconhecida ou densidade P sobre um espaço X de entradas; e uma função alvo Booleana f em X ; escolhida de uma conhecida classe F de funções alvo. A amostra finita dada por um algoritmo de aprendizado genérico em conformidade com os itens A e B, consiste em pares (x, y) , onde x é distribuído de acordo com P e $y = f(x)$. A exigência que algoritmos de aprendizado sejam gerais é atendida pelo fato que a distribuição P é arbitrária, e a classe de função F é tipicamente grande demais para permitir uma busca exaustiva (em tempo da ordem polinomial).

Durante a década de 1980 e 1990, foi obtida grande quantidade de resultados científicos sobre o tema a partir do modelo de Valiant e outros semelhantes. Muitos dos primeiros artigos sugeriam que os problemas de aprendizado simples e naturais podem ser computacionalmente difíceis por uma variedade grande de razões. Por exemplo, Pitt e Valiant (1988) mostraram que problemas de aprendizado para os quais a escolha mais óbvia para a forma da função de hipótese h , levam a um problema de dificuldade polinomial proporcional ao número de variáveis do sistema (NP). Kearns e Valiant (1994) exibiram íntimas conexões entre problemas de aprendizado e criptografia ao mostrar que vários problemas naturais, inclusive autômatos finitos de aprendizado e fórmulas booleanas, são computacionalmente difíceis de serem resolvidos independente do método usado na representação da hipótese.

As dificuldades práticas encontradas na implementação do modelo original de Valiant levaram a algumas modificações ao modelo. Uma modificação frutífera ao modelo foi suplementar à amostra casual dada ao algoritmo com um mecanismo questionador, isto é, ao invés de simplesmente receber passivamente pares (x, y) de alguma distribuição, o algoritmo agora pode solicitar ativamente a classificação de qualquer x desejado sob a função alvo f desconhecida. Com este mecanismo adicional, um número expressivo de algoritmos foi descoberto, incluindo para autômatos finitos por Angluin (2001); que era considerado computacionalmente intratável sem o mecanismo questionador, e para árvores de decisão por Kushlevitz e Mansour (1993).

Uma desvantagem do mecanismo questionador é a dificuldade de simular tal mecanismo em aplicações reais de aprendizado de máquina, onde apenas observações passivas do tipo propostas no modelo original de Valiant estão disponíveis. Uma alternativa talvez mais largamente aplicável deste modelo é a modificação chamada de aprendizado fraco ou modelo de reforço. Aqui está suposto que já se tem um algoritmo eficiente, mas com pouca capacidade de generalização a partir das hipóteses. Isto é formalizado pelo fato de que o algoritmo de aprendizagem fraca sempre tem como resposta uma hipótese cujo erro com respeito à função alvo(h) desconhecida é levemente melhor que o adivinhar aleatório. Entende-se por adivinhar aleatório a escolha totalmente randômica de uma hipótese. A meta de um algoritmo de reforço é então combinar as muitas hipóteses medíocres geradas por várias execuções do algoritmo de aprendizagem fraca numa única hipótese que é muito melhor que o adivinhar aleatório. Isto é possível por se supor que algoritmo de aprendizagem fraca será melhor que o adivinhar aleatório para muitas distribuições diferentes como entradas.

Embora considerações computacionais sejam as características distintivas primárias do aprendizado computacional, uma fração significativa do trabalho e interesse neste campo está dedicada a questões de natureza principalmente estatística e da teoria da informação. Assim sendo, caracterizações do número de observações necessárias por qualquer algoritmo para uma boa generalização foi o assunto de estudo prolongado intenso, onde destaca-se o trabalho de Vapnik(1995).

2.2.1 Aprendizado–PAC - *Probably Approximately Correct*

No artigo original de 1984, Valiant propôs o conceito de sistema de aprendizado denominado Provavelmente Aproximadamente Correto. (*Probably Approximately Correct*- PAC).

Na aplicação da teoria proposta por Valiant, o mais importante é a conexão entre os exemplos de treinamento e os exemplos de teste; a hipótese deve ser aproximadamente correta no conjunto de teste e não apenas no conjunto de treinamento.

A ciência do aprendizado computacional define duas funções, f e h e também um elemento denominado genericamente de algoritmo de aprendizagem.

A função denominada f é uma função genérica sobre qualquer domínio. A mesma é desconhecida e é o objetivo do aprendizado.

A função h é também denominada função hipótese e é a função que tenta emular o desempenho da função f . Em termos computacionais significa que h aproxima f .

A função básica do algoritmo de aprendizado é de, a partir dos resultados obtidos e conhecidos da função f , denominados de *Instâncias ou exemplos*, gerar a função h . O algoritmo de aprendizagem é treinado pelas instâncias da função f . Mais formalmente, *instância* (Russel e Norvig, 2004, p631), ou exemplo é um par $(x, f(x))$, onde x é a entrada e $f(x)$ é a saída da função $f(x)$ aplicada a x .

Os exemplos conhecidos de f são chamados de conjunto de treinamento, de onde são escolhidos exemplos que são efetivamente utilizados pelo algoritmo de aprendizagem. Este subconjunto do conjunto de treinamento é chamado de conjunto de teste.

Valiant introduziu a suposição de que tanto o conjunto de treinamento quanto o conjunto de teste devem ser randomicamente selecionados e de forma independente a partir de uma mesma população de exemplos, usando a mesma distribuição de probabilidade (para a extração). Esta suposição é conhecida como suposição estacionária (*stationarity assumption*). É muito mais concreta que as propostas que justificam indução com base na vaga e abstrata premissa que estabelece que o futuro deve ser semelhante ao passado. Sem a suposição estacionária, a teoria do aprendizado computacional não pode fazer qualquer reivindicação sobre o futuro, uma vez que não necessariamente existe conexão entre futuro e passado. A suposição estacionária fundamenta-se no pressuposto que o processo que seleciona exemplos não é viciado. O fundamental é saber qual deve ser o número de exemplos de treinamento a ser usado para que o algoritmo de aprendizado possa aprender bem um conceito. É fundamental também caracterizar o que bem ou bom significa, quando avaliando conceitos aprendidos bem como avaliando o algoritmo de aprendizado usado para aprendê-los.

Sejam :

X : conjunto de todos os possíveis exemplos, domínio da função f

D : distribuição de densidade de probabilidade a partir da qual os exemplos são extraídos

H : conjunto de possíveis hipóteses

h : uma hipótese específica de H .

m : número de exemplos no conjunto de treinamento

Assumindo-se que a função $f \in H$. Pode-se então definir o *erro* da hipótese h com relação à função f , dada uma distribuição D dos exemplos, como a probabilidade que h seja diferente de f em um exemplo.

O diagrama de blocos na Figura 9 ilustra de forma genérica o erro e pode indicar o quanto h está próxima de f . Quanto mais $e(x_i)$ tender a zero mais pode-se afirmar que o algoritmo de aprendizado atingiu seu objetivo.

Figura 9 - Funcionalidade o erro obtido a partir de uma hipótese $h(x_i)$

Não se pode expressar a precisão de uma hipótese sem levar em consideração a distribuição D , que especifica quais áreas de divergência entre a hipótese e a função f são mais importantes que outras. Uma hipótese h é chamada aproximadamente correta se:

$\text{erro}(h) \leq \epsilon$; onde ϵ é uma constante muito pequena.

Para um algoritmo de aprendizado específico, qual a probabilidade do conceito que ele aprende tenha um erro que seja limitado por ϵ ? É conveniente poder estabelecer um limite pequeno δ na probabilidade de que esse erro seja maior que ϵ , ou seja, é conveniente que :

$$p(\text{erro}(h) > \epsilon) < \delta$$

Tem-se, então, os elementos para dizer que um conceito aprendido é bom quando a probabilidade que o seu erro seja maior do que uma *precisão* ϵ , é menor do que um *grau de confiança* δ .

Diferentes graus de quão bom é o conceito correspondem a diferentes valores de ϵ e δ . Quanto menor forem ϵ e δ , melhor será o conceito aprendido.

O sistema de aprendizado recebe, então, como entrada, os parâmetros de precisão ϵ , e confiança δ , e é suposto induzir uma hipótese h tal que com probabilidade pelo menos de $1 - \delta$, o erro(h) $\leq \epsilon$. A hipótese induzida será provavelmente (dentro da confiança δ) uma boa aproximação (com erro no máximo de ϵ) do conceito real.

A abordagem do aprendizado PAC é a de mostrar que, após o exame de m exemplos, com alta probabilidade, todas as hipóteses consistentes serão aproximadamente corretas.

Como comentado em Russel (2004, p648), pode-se pensar em uma hipótese aproximadamente correta como aquela que está perto da verdadeira função f , no espaço de hipóteses estaria dentro do que é chamada de ϵ -ball em torno da função f , onde a ϵ -ball de um ponto x é o conjunto de pontos a uma distância no máximo ϵ de x . A ϵ -ball é definida dentro do espaço de todas as hipóteses H como uma superfície de separação entre as hipóteses boas e as ruins, no caso todas as hipóteses que estão na distância $\leq \epsilon$ são hipóteses boas, as demais são hipóteses ruins.

A Figura 10 mostra o conjunto de todas as hipóteses H (definidas pelo retângulo), divididas entre a ϵ -ball (definida pela circunferência) e todo o resto, que é chamado Hruim.

Figura 10 - Diagrama de Espaço de Hipóteses, mostrando a ϵ -ball em torno da verdadeira função f

Muito embora a proposta do modelo de aprendizado-PAC tenha contribuído com o importante resultado da relação entre o número de exemplos de treinamento e a confiança que se pode creditar à hipótese produzida por um algoritmo de aprendizado, esse resultado não é assim tão útil na prática, uma vez que é função do tamanho do espaço de hipóteses. Um grande número de trabalhos práticos em Aprendizado de Máquina emprega diversas técnicas experimentais para determinar se a aproximação da função induzida está correta ou não independente do tamanho do espaço de hipóteses.

Uma técnica experimental muito difundida consiste na divisão aleatória dos exemplos disponíveis para o aprendizado em dois conjuntos: conjunto de treinamento e conjunto de teste. O algoritmo de aprendizado usa o conjunto de treinamento para a indução da expressão do conceito, h . Uma vez obtida, h é testada usando o conjunto de teste, com o objetivo de obter uma estimativa de quão correta esta hipótese é.

Segundo diversos autores Russel(2004), Tanimoto(1987), Ginsberg(1993), tipicamente, $2/3$ dos exemplos inicialmente disponíveis são usados para o treinamento e $1/3$ para o teste, entretanto o tamanho do conjunto de teste deve ser escolhido visando obter um teste estatisticamente significativo da hipótese h .

Existe ainda uma noção adicional usada no aprendizado-PAC, que é a noção de *bias ou viés*.

A maioria dos algoritmos de aprendizado se restringe ao aprendizado de conceitos expressos de acordo com uma determinada sintaxe. Isso é importante porque contribui para uma restrição no tamanho do espaço de hipóteses obtido. Pode-se demonstrar que, quanto maior for o tamanho do espaço de hipóteses, mais difícil fica aprender qualquer instância particular dele. Essa forma de *bias* é conhecida como *bias do espaço de hipóteses restrito*.

Considere um algoritmo de aprendizado que, dado um conjunto de m exemplos de treinamento, induz uma função h que satisfaz o *bias do espaço de hipóteses restrito* e é consistente com os exemplos de treinamento. Tal algoritmo pode ser *pac-aprendível* ?

Se o universo em questão for finito e m for extremamente grande, a resposta é sim.

Para m bem grande, a expectativa é que muitos dos elementos significativos do universo compareçam como exemplos de treinamento, de maneira que qualquer função h que satisfizer tais exemplos é equivalente ao conceito f (Ginsberg,1993,pg136).

2.3 Aprendizado de máquina (A.M.)

Aprendizado de máquina é definido da seguinte forma: (Mitchel, 1997 p2)
 “Definição: Um programa de computador é dito aprender da experiência E com respeito a alguma classe de tarefas T e para medida de desempenho P , se seu desempenho em tarefas T , medido por P , melhora com experiência E .”

Diversos pesquisadores (Mitchell 1997; Russel 2004; Michie 1994) sugerem que um entendimento bem sucedido de como fazer computadores aprender abriria novos usos para os computadores. Um detalhado entendimento de algoritmos de processamento de informações para aprendizado de máquina pode levar a um melhor entendimento das capacidades (e incapacidades) de aprendizado do ser humano.

Alguns autores dividem (Russel, 2004, p629) o projeto de um elemento de aprendizagem em três aspectos mais importantes, como mostrado na Figura 11

Figura 11 - Principais aspectos do aprendizado de máquina

- Componentes: os *componentes* do elemento de desempenho que devem ser aprendidos;
- Realimentação: a *realimentação* que estará disponível para aprender esses componentes;
- Representação: a *representação* que será usada para os componentes.

Devido a sua importância, esses elementos serão estudados mais detalhadamente.

2.3.1 Componentes básicos de um elemento de desempenho

Existem diversas maneiras de classificar (Mitchell,1997, p5; Norvig,2004, p630) os componentes básicos de um agente ou elemento de aprendizagem.

A) Um mapeamento direto de condições no estado atual para ações.

B) Um meio para deduzir propriedades relevantes do mundo a partir da seqüência de percepções.

C) Informações sobre o modo como o mundo evolui e sobre os resultados de ações possíveis que o agente pode executar.

D) Informações de utilidade indicando a desejabilidade de estados do mundo.

E) Informações de valores de ações indicando a desejabilidade de ações.

F) Metas que descrevem classes de estados cuja realização maximiza a utilidade do agente.

2.3.2 Tipo de realimentação para aprendizado de máquina

O tipo de realimentação disponível para aprendizagem é fator fundamental na determinação da natureza do problema de aprendizagem. O campo de aprendizagem de máquina costuma distinguir três casos: aprendizagem **supervisionada**, **não-supervisionada** e **por reforço**, como representado na Figura 12:

Figura 12 - Tipos de realimentação para aprendizado de máquina

O problema da **aprendizagem supervisionada** envolve a aprendizagem de uma *função* a partir de exemplos de suas entradas e saídas, sendo esse tipo de realimentação restrito aos casos de ambientes completamente observáveis. Para ambientes parcialmente observáveis, o problema é mais difícil, porque os efeitos imediatos podem ser invisíveis.

O problema da aprendizagem **não-supervisionada**, (Weiss, 1999, p264) envolve a aprendizagem de padrões na entrada, quando não são fornecidos valores de saída específicos. O objetivo é descobrir saídas desejadas ou úteis por tentativa-e-erro ou processos de auto-organização. Esse tipo de aprendizagem é utilizado principalmente no contexto de sistemas de raciocínio probabilístico.

O problema da **aprendizagem por reforço** (Mitchell, 1997, p367) é o mais geral das três categorias. Em vez de ser informado sobre o que fazer por um instrutor, um programa de aprendizagem por reforço deve aprender a partir do reforço. Genericamente pode-se ter um programa que existe num ambiente descrito com base nos seguintes itens:

- S - Conjunto de todos os possíveis estados onde o agente pode atuar.
- A - Conjunto de todas as possíveis ações que o agente pode executar.

at - Ação que o agente executa em cada unidade de tempo

st - Estado em que o agente se encontra ao executar at .

γ - Tempo que o agente leva para executar uma ação, chamado de atraso.

A cada unidade de tempo, o agente executa uma ação em algum estado st e recebe um valor estimado de recompensa, que indica o valor imediato desta transição de estado-ação. O encadeamento de ações-recompensas produz uma seqüência si de estados, ai de ações, e de recompensas imediatas ri . A tarefa do agente é aprender uma estratégia de controle, π : S função de A, que maximize a soma esperada destas recompensas, com futuras recompensas descontadas exponencialmente por sua demora. A Figura 13 exemplifica como acontece o processo de aprendizado, neste caso a meta do programa ou agente é aprender a escolher ações que maximizem $r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \dots$, onde $0 \leq \gamma < 1$, ou seja, maximizem as recompensas ao longo do tempo, tempo esse que vai aumentando exponencialmente com as iterações do processo.

Figura 13 - Exemplo de aprendizagem por reforço

2.3.3 Tipo de representação para conhecimento aprendido

O tipo da representação das informações aprendidas também desempenha um papel crucial na determinação de como os algoritmos de aprendizagem devem funcionar. As formas mais consagradas, segundo importantes pesquisadores, (Mitchell, 1997, p22; Russel, 2004, p158; Tanimoto, 1987, pg89) de representação de conhecimento são:

- Polinômios ponderados para funções de utilidade em programas de jogos(pesos variáveis de acordo com o jogo);
- Sentenças lógicas proposicionais e de primeira ordem para todos os componentes de um agente lógico;
- Redes neurais;
- Algoritmos genéticos;
- Descrições probabilísticas como redes bayesianas para os componentes inferenciais de um agente de teoria da decisão.

A Tabela 2.1 mostra as principais formas de representação de conhecimento que são utilizadas em A.M. (Tanimoto,1987,pg130). Nota-se que qualquer que seja o método de representação, o mesmo possui limitações, sendo portanto necessário um estudo aprofundado para a correta escolha de um método mais apropriado para o tipo de conhecimento que se deseja aprender.

Tabela 2.1– Principais formas de representação de conhecimento que são utilizadas em A.M.

Método	Relações Manuseadas	Mecanismo de Inferência	Limitações Principais
Lógica Proposicional	Booleanas, tabelas da verdade.	Modus ponens, etc	Modela somente verdades booleanas, mas não as relações entre elas
Hierarquia Conceitual	“ISA”	Busca gráfica e fechamento transitivo	Limitado a uma relação
Lógica de Predicados	Qualquer predicado	Resolução & outros	Falhas em organizar conhecimento; desajeitado para controle de informações
Frames	Binário ou ternário	-	É somente uma metodologia, não um sistema de representação
Redes Semânticas	Binário ou ternário	-	Não há padrões
Restrições	Qualquer predicado	Propagação; satisfação	Não há padrões
Regras de produção	Se – então (if-then)	Regras de ativação	Desajeitado para conhecimento não-processual.
Base de dados Relacional	n-dimensional	Seleção, projeção e agrupamento	Desajeitado para controle de informação

2.3.4 Aprendizagem indutiva

A pesquisa em aprendizado automático de máquina tem investido muito no desenvolvimento de sistemas que aprendem regras de classificação a partir de exemplos pré-classificados (Batista,1997,pg18; Nicoletti,2005,pg7; Russel, 2004, p633), área identificada também com o nome de *aprendizado indutivo simbólico supervisionado*.

Tipicamente, a entrada para o sistema de aprendizado é um conjunto de *exemplos de treinamento*, cada um deles previamente classificado por um especialista humano do domínio em questão como pertencente a uma determinada classe (o que caracteriza a supervisão). O objetivo do sistema é produzir um conjunto de regras (ou qualquer outra representação classificatória, tal como expressões lógicas, árvores de decisão, etc.) que possa ser usado para prever, com precisão, a classe de novos exemplos. Nesta forma de aprendizado as classes são identificadas como sendo os conceitos.

O conjunto de regras induzidas que descrevem pacientes com arteriosclerose, por exemplo, pode ser visto como uma definição do conceito arteriosclerose; de uma forma simplista pode-se dizer, então, que o sistema que induzir tal conjunto de regras adquiriu o conceito de arteriosclerose. Em classes de exemplos como o de diagnósticos médicos, a regra de classificação pode ser vista como uma função h que associa a descrição de um paciente expressa como um conjunto de pares na forma *atributo - valor_de_atributo*, ao nome de uma classe.

Em uma situação geral de aprendizado, os exemplos de treinamento podem ser vistos como pares na forma $(x, f(x))$, onde x representa um conjunto de pares atributo-valor e $f(x)$ a classe associada a esse conjunto. O objetivo da inferência indutiva (ou indução), implementada por um algoritmo de aprendizado indutivo é: dado um conjunto de treinamento formado por pares $(x, f(x))$, obter a expressão de uma função h , que seja uma aproximação da função f . A função h é chamada de hipótese.

Como comentado em Ginsberg (1993, pg 134), na maioria dos casos o conjunto de treinamento representa apenas uma pequena amostra do espaço de todos os possíveis pares $(x, f(x))$. É razoável supor isso, uma vez que em domínios

médicos, por exemplo, existe a expectativa de que médicos aprendam como diagnosticar doenças sem que tenham visto todos os possíveis pacientes. Isso implica que, sem o uso do processo indutivo, não existe maneira de saber o valor de $f(x)$, para um valor de x que nunca tenha sido observado antes. Dizer isso é dizer que os algoritmos de aprendizado devem encontrar definições de f (no caso, funções h) que sejam generalizações dos exemplos de treinamento e não definições criadas especificamente de f , que apenas se aplicam aos exemplos observados durante o aprendizado.

Qualquer preferência por uma hipótese, além da mera consistência com os exemplos de treinamento, é chamada *bias* (que na terminologia de aprendizado significa heurística). Devido ao fato de sempre existir um número muito grande de hipóteses consistentes com o conjunto de treinamento, todo algoritmo de aprendizado exibe algum tipo de *bias* (ou viés). Um exemplo de *bias* é, a lâmina Ockham, que estipula que ao haver mais de uma hipótese plausível para explicar algum fenômeno, deve-se optar pela hipótese mais simples.

É digno de nota o fato que o aprendizado indutivo nunca pode ser dedutivamente garantido. Por exemplo, uma pessoa compra uma caixa de uvas e prova 120 unidades, sendo todas elas doces, porém isso não garante que a próxima a ser provada será doce também; mas tal fato representa um forte indício que a próxima a ser provada também será doce. Como, então, responder à pergunta:

“Como se sabe que uma hipótese h está próxima da função f , se não se sabe quem é f ?”.

As respostas a essa pergunta estão sendo obtidas via teoria de aprendizado computacional, vide item 2.1, onde Valiant propôs o conceito de sistema de aprendizado Provavelmente Aproximadamente Correto. (*Probably Approximately Correct*- PAC).

2.4 Descoberta de conhecimento em bases de dados (*knowledge discovery in databases* - KDD)

2.4.1 Introdução à descoberta de conhecimento em bases de dados

O progresso da tecnologia da informação levou nos últimos anos ao estabelecimento de grandes bases de dados em diversos ramos de atividades humanas. Grandes quantidades de dados são armazenados e mantidos a um custo decrescente, porém quantidades massivas de dados são de pouca utilidade porque os dados provêm informação limitada para tomadas de decisões. O conhecimento que está escondido na base de dados é mais importante que o dado em si e a capacidade humana de análise é limitada.

Como forma de organizar as técnicas que foram desenvolvidas para resolução das dificuldades crescentes em se lidar com bases de dados, foi cunhado o termo “Descoberta de conhecimento em bases de dados” (*knowledge discovery in databases* - KDD) (Piatetsky-Shapiro, et al., 1991; Fayyad, et al., 1996) que engloba diversas áreas do conhecimento com o objetivo final de extrair conhecimento de dados.

Segundo Fayyad (1996): “Knowledge Discovery in Databases is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data”, que em tradução livre significa “Descoberta de conhecimento em bases de dados é um processo não trivial de identificação de padrões válidos, novos, entendíveis e úteis em dados” onde:

- *Padrão* é uma sentença que descreve fatos em um subconjunto dos dados obedecendo a uma classificação;
- *O Processo* tem múltiplos passos, envolvendo: preparação de dados, pesquisa de padrões, avaliação de conhecimento e refinamento;
- *Validação* é a verificação de que os padrões descobertos são extensivos a novos dados, não utilizados na descoberta dos padrões.

2.4.2 O processo de aquisição de conhecimento

Brachman e Anand (1996) modelaram o processo de aquisição de conhecimento de maneira prática, enfatizando a natureza iterativa desse processo; são nove passos, explicitados a seguir:

- A. Desenvolver o entendimento do domínio, o conhecimento prévio necessário e a identificação dos objetivos do usuário;
- B. Criar um subconjunto-alvo de dados, onde se descobrirão os padrões;
- C. Preparação da base de dados, com remoção de ruídos, coleta-las informações necessárias para a modelagem e escolha da estratégia para tratamento de campos incompletos;
- D. Redução e Projeção de dados, encontrando os principais aspectos representativos dos dados, reduzindo a dimensão do Banco de Dados;
- E. Escolha da tarefa de *Mineração de dados*, se classificação, regressão, clusterização ou previsão;
- F. Escolha do algoritmo do *Mineração de dados*, selecionando o método apropriado a ser utilizado na descoberta dos padrões;
- G. *Mineração de dados*. A procura, propriamente dita, dos padrões, utilizando-se das ferramentas selecionadas no item F;
- H. Interpretação dos padrões obtidos, com possível retorno a um dos passos A a G;
- I. Consolidação do conhecimento adquirido, com implementação ou mesmo apenas relatório. Isto inclui verificação de consistência com o conhecimento já existente.

Este processo é iterativo, podendo haver interações (loops) entre passos. A Figura 14 ilustra como ocorrem as etapas do processo de aquisição de conhecimento em bases de dados (Fayyad (1996)).

Figura 14 - O processo de aquisição de conhecimento em bases de dados

2.4.3 Armazéns de dados

Um campo de conhecimento relacionado com o processo de aquisição de conhecimento em bases de dados é chamado de armazéns de dados (*data warehousing*), esse campo se refere à maneira como os dados são armazenados e estruturados em bases de dados que normalmente são disponibilizadas para análises *on line* e apoio a tomadas de decisões. A armazenagem de dados colabora com o processo de aquisição de conhecimento em bases de dados de duas maneiras importantes:

Limpeza dos dados: Trata-se do processo no qual os dados são, de maneira lógica, endereçados, uniformemente representados e manipulados de forma a tratar dados perdidos evitando erros e ruídos quando possível.

Acesso aos dados: Trata-se dos métodos que tornam os dados armazenados efetivamente acessíveis.

Ao executar a limpeza de dados e os tornar acessíveis, os armazéns de dados preparam o caminho para o processo de aquisição de conhecimento.

Os armazéns de dados possibilitam a análise de grandes volumes de dados, coletados dos sistemas de informações transacionais. São as chamadas séries históricas que possibilitam uma melhor análise de eventos passados para a tomada

de decisões presentes e a previsão de eventos futuros. Por definição, os dados em um armazém de dados não são voláteis, ou seja, eles não mudam, salvo quando é necessário fazer correções de dados previamente carregados. Os dados então são somente para leitura e não podem ser alterados.

Segundo Fayyad (1996), o processo de aquisição de conhecimento em bases de dados, KDD, refere-se ao processo de extrair conhecimento útil de dados, e mineração de dados refere-se a um passo particular neste processo. Devido à sua importância no contexto geral deste trabalho de pesquisa, a mineração de dados será estudada em maiores detalhes.

2.5 Mineração de dados - *Data mining*

2.5.1 Introdução à mineração de dados

Mineração de Dados (M.D.) mais conhecida como “Data mining” possui varias definições (Han,2000,p3; Witten,2005,p4; Hand, et al.2001,p22) onde se destaca: “A mineração de dados se refere à descoberta de conhecimento em bases de dados (*knowledge discovery in databases* - KDD), e é a extração automatizada ou conveniente de padrões representando conhecimento implicitamente armazenado em bases de dados grandes, armazéns de dados, e outros repositórios de informação de grande porte”.

Ou ainda (Larose,2005,p2) “Mineração de dados é o processo de descobrir novas correlações significativas, padrões e tendências peneirando grandes quantidades de dados armazenados em repositórios, usando as tecnologias de reconhecimento de padrões”.

A mineração de dados é um campo multidisciplinar, onde se dá a intersecção com outras áreas do conhecimento como: a tecnologia de base de dados, inteligência artificial, aprendizado de máquina, redes neurais, estatística, reconhecimento de padrão, sistemas baseados em conhecimento, aquisição de conhecimento, recuperação de informações, computação de alto desempenho e

visualização de dados. A Figura 15 ilustra qualitativamente o aspecto de confluência de múltiplas disciplinas que é o campo da mineração de dados.

Figura 15 - Ilustração da multidisciplinariedade da mineração de dados

A meta da mineração de dados (Fayyad et al., 1996) é conseguir extrair um conhecimento de alto nível proveniente de dados de baixo nível no contexto de grandes conjuntos de bases de dados. Para um melhor entendimento dos principais conceitos envolvidos na mineração de dados, serão estudados seus três principais aspectos:

- Dados de entrada;
- Dados de saída;
- Algoritmos.

2.5.2 Principais tipos de dados de entrada na Mineração de Dados (M.D.)

O dados de entrada têm a forma de **conceitos, instâncias e atributos**.

Em M.D. o objeto primário de estudo e análise é o próprio dado e o conjunto de possíveis interpretações de um determinado conjunto de dados.

2.5.2.1 Instância

Instância são os dados que podem ser agrupados e classificados segundo algum ou alguns atributos. A instância ou até mesmo um conjunto de instâncias formam a entrada dos esquemas de M.D. que, no software desenvolvido neste

trabalho é o algoritmo de M.D. denominado Tertius (vide 2.7.3) que está na máquina de inferência.

2.5.2.2 Atributo

Atributo em M.D. é um qualitativo ou um adjetivo de um dado ou até de um conjunto de dados.

2.5.2.3 Conceito

Conceito: É o aprendizado obtido por meio da aplicação do algoritmo de M.D. no conjunto de instâncias de entradas, que neste trabalho são as regras geradas pelo algoritmo Tertius, conforme melhor explicado no item 2.7.3.

A Figura 16 ilustra genericamente o sistema aplicado neste trabalho.

Figura 16 - Ilustração de atributos, instâncias e conceito em M.D.

2.5.3 Principais tipos de dados de saída na Mineração de dados - Representação de Conhecimento

Os dados (Ye,2003) de saída em mineração de dados têm diversas formas de serem representados. Genericamente, a estrutura dos dados de saída é chamada de representação de conhecimento e são casos particulares das representações de conhecimento utilizadas em aprendizado de máquina (Vide tabela 2.1).

O tipo de dado disponível e a natureza do problema de mineração de dados tipicamente determinam qual metodologia de mineração de dados é mais apropriada. Um problema de mineração de dados pode ter como objetivo prever e classificar dados, descobrir padrões de dados (Last, et al.2004). Os dados de entrada podem não possuir nenhum conhecimento prévio (dados não etiquetados) ou podem possuir conhecimento prévio (dados etiquetados) sobre categorias de dados ou grupos na forma de registros de dados com ou sem marcadores de tempo para indicar a seqüência temporal dos registros de dados.

As principais metodologias de representação de conhecimento em mineração de dados são mostradas na Tabela 2.2, onde se destacam pela sua intensa utilização as árvores de decisão e as regras de associação.

2.5.3.1 Principais tipos de dados usados na Mineração de dados

Os tipos de dados (Ye,2003) utilizados em mineração de dados podem ser classificados em quatro tipos principais:

- **Dados etiquetados** (*labeled*), são aqueles dados onde se tem algum conhecimento prévio dos dados como, por exemplo, a separação em categorias de um grupo de consumidores de um supermercado;
- **Dados não etiquetados** (*unlabeled*) dados onde não se tem nenhum conhecimento prévio;
- **Registros de não temporais**, são dados que não ocorrem seguindo uma seqüência temporal;
- **Dados de séries temporais**, são dados que ocorrem seguindo uma seqüência temporal.

2.5.3.2 Principais tipos de problemas trabalhados pela mineração de dados

Os problemas computacionais tipicamente trabalhados (Ye,2003) pela mineração de dados podem ser classificados em três tipos principais:

Predição e classificação. Nesse caso a M.D. é utilizada para inferir conseqüências de dadas situações e também para classificá-las; particularmente a classificação é freqüentemente chamada de aprendizagem supervisionada porque o número e o tipo das classes onde o modelo de aprendizado é aplicado estão predefinidos.

Descoberta de padrões de dados, associações e estruturas. Neste caso a M.D. é utilizada para inferir padrões não evidentes em conjuntos de dados, associações entre os dados ou ainda estruturas, particularmente importante para o aprendizado não supervisionado. Alguns pesquisadores(Agrawal, et al., 1995; Faloutsos, et al., 1994), têm utilizado a M.D. com a abordagem de padrões para reconhecimento de fala.

Reconhecimento de similaridades e diferenças nos dados. Neste caso a M.D. lida somente com duas categorias de dados, dados produzidos em condições normais e dados produzidos em condições anormais; em geral o objetivo desse tipo de problema é detectar anormalidades.

É digno de nota que esses três tipos de problemas são os mais comumente utilizados, não sendo de maneira alguma a totalidade dos problemas que são tratados por meio da mineração de dados.

Tabela 2.2 Principais tipos de representação de conhecimento em M.D. e seus principais usos. (Ye,2003)

Metodologia de Mineração de Dados	Tipos de Dados				Problema computacional trabalhado pela M.D.		
	Dados etiquetados (labeled)	Dados não etiquetados (unlabeled)	Registros de dados não temporais	Dados de séries temporais	Predição e classificação	Descoberta de padrões de dados, associações e estruturas.	Reconhecimento de similaridades e diferenças nos dados
Árvores de decisão	X		X		X	X	X
Regras de associação		X	X			X	X
Redes neurais artificiais	X	X	X	X	X		X
Análise estatística		X	X		X		X
Análise Bayseana	X	X	X	X	X	X	X
Processos ocultos de Markov e padrões de mineração seqüenciais	X	X		X	X		
Modelos de predição e classificação	X		X	X	X	X	X
Métodos Psicométricos de modelamento de variáveis ocultas	X	X	X		X	X	X
Agrupamentos Escaláveis		X	X			X	X
Séries temporais similaridade e indexação.	X	X		X	X		X

2.5.4 Principais tipos de tarefas tratadas pela mineração de dados baseadas no tipo de dado a ser minerado

Nas últimas décadas ocorreram (Lu, 1997 p3; Hand, et al.2001,p47) muitos avanços em pesquisas e desenvolvimentos de mineração de dados. Diferentes considerações filosóficas em descoberta de conhecimento em bases de dados (KDD) levaram a metodologias diferentes no desenvolvimento de técnicas de KDD. Em decorrência dos tipos de dados disponíveis para serem minerados (vide tabela 2.2), tarefas de mineração de dados podem ser classificadas em sete tipos principais:

A. Regra de mineração característica. A sumarização de características gerais de um conjunto especificado pelo usuário dentro da base de dados. Por exemplo, o sintoma de uma doença específica pode ser sumarizado por um conjunto de regras características;

B. Regra de mineração discriminatória. A descoberta de características ou propriedades que distinguem um conjunto de dados chamados *classe alvo* de outro conjunto de dados chamado *classe contraste*. Por exemplo, para distinguir uma doença de outras, uma regra discriminatória sumariza os sintomas que diferenciam essa doença de outras;

C. Regra de mineração associativa.

A descoberta de associações dentro de um conjunto de objetos, matematicamente assim representado:

$$\{A_i\}^m_{i=1} \dots \{B_j\}^n_{j=1} \text{ tal. que } A_1 \wedge \dots \wedge A_m \rightarrow B_1 \wedge \dots \wedge B_n .$$

Por exemplo, em um diagnóstico médico de uma doença, pode-se descobrir que um conjunto de sintomas freqüentemente ocorra junto com outro conjunto de sintomas;

D. Regra de mineração classificatória. A categorização de dados em um conjunto conhecido de classes. Por exemplo, em um conjunto de carros com diversas características, pode-se efetuar uma classificação por suas quilometragens rodadas;

E. Agrupamentos (*Clustering*). A identificação de agrupamentos (classes ou grupos) de um conjunto de objetos baseados em seus atributos. Os objetos são

agrupados, segundo um critério tal que, dentro de um grupo, as similaridades são minimizadas e entre grupos diferentes, as similaridades são maximizadas. Por exemplo, um conjunto de doenças pode ser agrupado em muitos grupos baseados nas semelhanças de seus sintomas;

F. Regra de mineração evolucionária. A descoberta de um conjunto de regras que refletem a evolução geral de um conjunto de dados. Por exemplo, podem-se descobrir os maiores fatores que influenciam as flutuações no mercado de ações.

2.5.5 Análise comparativa entre Aprendizado de Máquina e Mineração de dados

Os métodos de aprendizado de máquina típicos (Mitchell 1997) partem do paradigma de aprendizado por exemplos. Em geral esses métodos trabalham com um conjunto de conceitos ou classes-etiquetadas de exemplos de treinamento que são examinados para derivar-se ou aprender-se uma hipótese.

As mais importantes diferenças entre os métodos de aprendizado por exemplos e os métodos de mineração de dados são:

A. Diferenças filosóficas concernentes ao problema da descrição de conceitos. Na maioria dos algoritmos do tipo aprende-por-exemplos desenvolvidos em aprendizagem de máquina, o conjunto de exemplos a ser analisado é particionado em dois conjuntos: exemplos positivos e negativos, respectivamente representando alvo e classes contrastantes. O processo de aprendizado seleciona um exemplo positivo aleatoriamente, e usa para formar uma hipótese que descreva os objetos ou a classe dos objetos. O processo de aprendizado então executa generalizações na hipótese usando os exemplos positivos restantes, e especialização usando os exemplos negativos. Em geral, a hipótese resultante cobre todos os exemplos positivos, mas nenhum dos exemplos negativos.

Uma base de dados normalmente não armazena os dados negativos explicitamente, dessa forma os exemplos negativos não podem ser usados para especialização. Isto é o motivo pelo qual os métodos de mineração de dados devem colecionar um conjunto de dados comparáveis que não estão na classe alvo

(positivo), para uso como dados negativos. A maioria dos métodos orientados à base de dados também tendem a ser baseados em generalizações. Mesmo que a maioria dos métodos orientados à base de dados forneçam a operação de especialização (*drill-down*), esta operação essencialmente é implementada por uma busca exaustiva até o encontro de um estado prévio.

B. Diferenças entre o tamanho do conjunto de treinamento. Na maioria dos métodos de aprendizado de máquina o conjunto de exemplos de treinamento é relativamente pequeno em comparação com os dados analisados por técnicas orientadas à base de dados. Caracteristicamente nos métodos de aprendizado de máquina, é mais fácil achar descrições que cubram todos os exemplos positivos sem cobrir nenhum exemplo negativo. No entanto, considerando a diversidade e quantidade enorme de dados armazenados em muitas bases de dados reais, não é desejável que para análise de tais dados sejam derivadas regras ou padrões que apenas cubram todos os exemplos positivos, mas nenhum dos negativos. Em vez disso, é desejável um conjunto de características ou regras que agrupem a maioria dos dados na classe positiva, distinguindo os exemplos positivos dos exemplos negativos. (Isto também pode ser descrito como uma distribuição de probabilidade).

C. Diferenças concernentes aos métodos de generalização utilizados. Tanto a mineração de dados quanto os métodos de aprendizado de máquina empregam remoção de atributo e generalização de atributo (também conhecida como *concept tree ascension*) como suas principais técnicas de generalização. Considerando o conjunto de exemplos de treinamento, o enfoque do aprendizado de máquina tenta generalizar a partir de cada indivíduo do conjunto de treinamento, um por um, ao passo que o enfoque orientado à base de dados executa generalizações em atributos individuais um por um.

Na estratégia de um por um (referidos aos itens do conjunto de treinamento ou *tuple*), o enfoque de aprendizado de máquina examina todos os indivíduos do conjunto de treinamento para se inferir conceitos generalizados. O algoritmo deve procurar cada nó no espaço de busca representando todos os possíveis conceitos derivados de generalização em cada exemplo de treinamento.

Por outro lado, uma aproximação de base de dados empregando uma estratégia atributo-orientada executa generalização em cada atributo ou dimensão uniformemente para todos os indivíduos no início da etapa de generalização, o que

usualmente torna o enfoque de base de dados ou mineração de dados mais eficiente do ponto de vista computacional.

D. Otimização à estrutura da base de dados. Os algoritmos que partem do enfoque da mineração de dados são otimizados sobre a estrutura da base de dados. Assim esse enfoque consegue ser mais eficiente que a maioria dos algoritmos de aprendizado de máquina que não são otimizados para se aproveitarem das funções predefinidas das bases de dados o que, em base de dados grandes, modifica significativamente o desempenho dos algoritmos.

2.6 Aprendizagem em árvores de decisão

Árvores de decisão (Agrawal *et al.*, 1996 p1; Russel, 2004, p633; Nogueira, 1999, p29) são um método baseado em divisões sucessivas de dados, a partir de valores determinados de suas variáveis, uma a cada vez. É um método que oferece grande facilidade de apresentação do conhecimento que extrai. Porém, como consequência direta da simplicidade de representação, pode ocorrer uma limitação à sua capacidade de classificação. Ao aumentar-se a complexidade do modelo, admitindo então expressões mais abrangentes, este fica mais eficaz, tornando-se entretanto, de mais difícil compreensão.

Há vários algoritmos que implementam Árvores de Decisão e indução de regras; seu uso a princípio ocorreu preponderantemente em casos de previsão, sendo hoje largamente utilizado para classificação, regressão e descrição (Agrawal *et al.*, 1996 p8).

Simplificadamente, uma árvore de decisão toma como entrada um objeto ou situação descritos por um conjunto de atributos e retorna uma decisão ou resultados da decisão. Os atributos de entrada podem ser discretos ou contínuos. Foi dado destaque as entradas discretas. O valor de saída também pode ser discreto ou contínuo; a aprendizagem de uma função de valores discretos é chamada aprendizagem de classificação; a aprendizagem de uma função contínua é chamada regressão.

Um interessante exemplo de como se realiza uma árvore de decisão é mostrado em Nogueira (1999). Exemplo: Qual o perfil dos motoristas de automóveis BMW?

Para chegar-se a estes padrões, seria realizado um processo de elicitación de conhecimento de tal forma a dividir as perguntas em subgrupos, ou seja, em classes.

Pergunta: Os motoristas de BMW gostam de carros esportivos?

Esta pergunta já agruparia as respostas em dois grupos distintos, tão mais distintos quanto mais adequada e exata a pergunta fosse.

Pode-se apresentá-la de uma forma mais eficiente: Os motoristas de BMW gostam muito de carros esportivos? Nesta nova versão atribuiu-se um valor à variável gostar, no caso muito.

Em seguida poder-se-ia fazer outra pergunta, obedecendo, às variáveis disponíveis no Banco de Dados. Por exemplo, outra pergunta abordaria a classe social dos motoristas de BMW podendo surgir uma segregação que retiraria apenas os de classe média da resposta. O que parece evidente, visto que normalmente apenas pessoas ricas e seus motoristas dirigem carros deste valor no Brasil.

As perguntas se repetiriam até que houvesse apenas um tipo de resposta para o subgrupo formado em uma dada pergunta. Caso o outro ramo, subgrupo, para aquela pergunta tenha mais de uma classificação, teriam continuidade as perguntas.

Por meio destas respostas, o Banco de Dados se classificaria e poderiam se obter regras como:

Pessoas abastadas, que gostam muito de carros esportivos, não tenham uma Ferrari, residam em uma cidade onde haja um ou mais revendedores da BMW e não haja nenhum da AUDI, têm 15% de chance de dirigir um BMW.

Facilmente se reconstruiriam as perguntas que teriam sido feitas para chegar-se a esta regra:

1. Pessoas ricas guiam BMW?
2. Pessoas que gostam muito de carros esportivos guiam BMW?
3. Pessoas que têm Ferrari guiam BMW?
4. Há motoristas de BMW onde não há revendas?
5. Há motoristas de BMW onde há revendas AUDI ?

Teriam então constituído uma árvore que teria como um de seus ramos a Figura 17.

Figura 17 - Árvore de decisão sobre exemplo do carro esportivo

Para a utilização de Árvores de Decisão, os dados devem estar organizados de forma regular, como uma série de atributos a cada dado, podendo haver valores discretos e contínuos.

Devido a esta restrição, há limitações na diversidade possível de utilizações, como ocorreria para a análise de dados descritos subjetivamente, como uma ficha de uma modelo de desfile de moda, por exemplo. Esta limitação poderia ser contornada possivelmente com a utilização de algoritmos identificadores de palavras, que poderiam gerar um arquivo adequado à utilização, ou na utilização de um formulário padronizado para a ficha médica, como ocorre em vários locais, como a agência Elite-Models em São Paulo.

Apesar da relativa tolerância à inconsistência de dados, deve-se evitar o excesso de dados incompletos ou claramente díspares. Mesmo que a ferramenta

seja robusta a algum nível de ruído (Russel, 2004, p635), sua eliminação prévia normalmente catalisa o aprendizado.

A definição das classes - para as variáveis endógenas - deve ser realizada previamente, nos casos em que houver necessidade, ou seja, quando as variáveis se apresentarem com valores contínuos. As Árvores de Decisão apresentam rendimento sensivelmente superior ao classificar níveis de padrões do que se obtém com valores contínuos.

2.6.1 A estrutura das árvores de decisão

As Árvores de Decisão têm a estrutura baseada em:

Folhas, que seriam as classificações existentes, e os Nós de Decisão, onde se encontram os testes que a árvore faz dos dados.

Essa estrutura funciona da seguinte forma: Um dado entra na árvore através da raiz, percorrendo os nós e neles sendo direcionados de forma a alcançarem as folhas a que se referem suas classes.

A partir das classificações obtidas pelas folhas, em que se enquadraram os dados, e dos nós, que foram percorridos para chegar-se a elas, geram-se as regras que determinam aquela classificação.

O exemplo dado, *Motoristas de BMW*, mostra os testes realizados (nós), correspondendo às perguntas feitas: “É maior de 18 anos?”, por exemplo, bem como as folhas, “Dirige um BMW”.

2.6.2 Limitações entre explicação x entendimento

As árvores de decisão têm grandes qualidades como classificadoras. Além dessa característica, seus modelos são de fácil compreensão para os usuários. As perguntas, ou testes, a que são submetidos os dados acabam por gerar regras inteligíveis, visto que o próprio processo de eliciação do conhecimento é extremamente simples, na verdade lógico, uma lógica binária.

Entretanto, ao exigir-se um refinamento excessivo da análise, as informações extraídas passam a se tornar mais complexas caracterizando uma relação de custo - benefício entre a quantidade de informação que uma árvore de decisão pode expressar e seu entendimento por usuários.

As árvores de decisão, como visto, são divisões sucessivas de blocos de dados. A cada nó se tenta segmentar os blocos em dois grupos, distintos pela pergunta chave a que se submeterá o bloco.

Nota-se então a importância da escolha das perguntas a serem feitas aos blocos, que visarão sempre à geração de sub-blocos, de maior pureza possível.

Para tal, faz-se clara a necessidade de uma definição da impureza (Michie, 1994,pg62). Esta função tem diversas formas. A pureza ideal seria conseguida pela obtenção de um bloco que fosse descrito com apenas um determinado arranjo de variáveis, não havendo ocorrência com qualquer variação de qualquer parâmetro, ausência de variação essa pouco provável em situações complexas características da maioria dos problemas práticos.

Em outras palavras, à medida que os blocos que chegam a um determinado nó apresentam uma classificação francamente preponderante, este nó está se aproximando da pureza, sendo a situação ideal definida como aquela em que só haja uma classificação para os elementos daquele bloco, o nó puro.

Para que se alcance a pureza, são utilizados algoritmos diversos, entre eles o CART (Breiman et al, 1984) que têm a função de segmentar os blocos, levando em consideração apenas um parâmetro por vez. Dificilmente se obterá nós puros; associa-se portanto, probabilidades de ocorrência a cada classificação. As divisões se sucedem até que as variações de impureza vão diminuindo, atingindo os níveis considerados **toleráveis** pelo especialista, se possível.

A determinação do tamanho ideal das árvores é um aspecto fundamental a ser estudado. A existência de ruídos no banco de dados pode gerar uma árvore com mais informações do que o banco de dados que ela representa. Ao criar-se um refinamento excessivo, cria-se uma maior propensão à inclusão dos ruídos, como se fossem informações. Por outro lado, pode-se ter uma árvore superajustada. Porém é grande a possibilidade de pobreza de representação ao se restringir precocemente o desenvolvimento de uma árvore.

Deve-se, por isso, avaliar o momento do término da segmentação, aquele a partir do qual a evolução da pureza fica estagnada, passando a árvore a gerar não

regras, mas exceções. Duas são as formas de evitar os dois desequilíbrios acima descritos: *forward pruning* e *backward pruning*. No primeiro faz-se o acompanhamento da evolução da impureza dos nós, da forma já descrita. No segundo se efetuam podas nas árvores obtidas, simplificando os modelos obtidos (Breiman et al, 1984).

A capacidade das regras geradas de representar os padrões procurados fica então atrelada à representatividade da amostra. Ou seja, a utilização de blocos desbalanceados em relação ao universo certamente trará este desbalanceamento para as regras que se obterá.

2.7 Aprendizagem por regras de associação

As regras de associação (Gonçalves, 2005; Agrawal *et al.*, 1996, Larose 2005) são técnicas que realizam uma chamada análise de afinidade, que é o estudo de atributos ou características que sempre estão associadas mutuamente. Os métodos mais conhecidos de implementação das regras de associação são os *market basket analysis* ou seja, análise de transações de compra; são técnicas que mostram as condições para os valores dos atributos que ocorrem freqüentemente num conjunto de dados. Esta técnica de exploração de dados é utilizada em larga gama de aplicações em áreas como a prática de negócio e pesquisas científicas. Desde a análise das preferências dos consumidores, gestão de recursos humanos até a própria história da linguagem, estas técnicas permitem aos analistas e investigadores descobrir padrões escondidos em grandes conjuntos de dados. Como exemplo de uma regra de associação que poderia ser encontrada num banco de dados de um supermercado seria o fato de que 70% dos clientes que compram o produto A, também adquirem, na mesma ocasião, o produto B.

Formalmente as regras de associação foram introduzidas em (Agrawal et al, 1993) da seguinte forma: sejam $\Gamma = \{i_1, i_2, \dots, i_m\}$ um conjunto de m itens distintos e D uma base de dados formada por um conjunto de transações, onde cada transação T é composta por um conjunto de itens (*itemset*), tal que $T \subseteq \Gamma$. Uma regra de associação é uma expressão na forma $A \Rightarrow B$ (lê-se A implica em B), onde $A \subset \Gamma$, $B \subset \Gamma$, $A \neq \emptyset$; $B \neq \emptyset$ e $A \cap B = \emptyset$. A é denominado antecedente e B denominado conseqüente da regra. Tanto o antecedente, quanto o conseqüente de uma regra de

associação podem ser formados por conjuntos contendo um ou mais itens. A quantidade de itens pertencentes a um conjunto de itens é chamada de **comprimento** do conjunto. Um conjunto de itens de comprimento k costuma ser referenciado como um *k-itemset*.

2.7.1 Suporte e confiança

Cada regra da forma $A \Rightarrow B$ possui dois atributos que determinam sua validade no conjunto de dados e também limitam a quantidade de regras extraídas. São eles o suporte e a confiança. Estes possibilitam o descarte das regras julgadas de pouco interesse, já que são menos freqüentes e confiáveis.

O suporte s (Larose, 2005) para uma particular regra de associação $A \Rightarrow B$ é a proporção de transações em D que contem ambas A e B , ou seja:

$$\text{Suporte}(A \cup B) = \frac{\text{N.de.registros.com}(A \cup B)}{\text{Numero.total.de.transações.de.D}}$$

é a porcentagem de transações onde este *itemset* aparece, este por sua vez será considerado freqüente se o seu suporte for maior ou igual a um suporte mínimo estabelecido previamente.

A confiança c para uma particular regra de associação $A \Rightarrow B$ é uma medida da acurácia da regra, definida como uma porcentagem das transações em D que contém A e também contém B , ou seja:

$$\text{Confiança} = P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{\text{N.de.transações.contendo.A.e.B}}{\text{Numero.de.transações.contendo.A}}$$

sendo que estatisticamente, a confiança é a probabilidade condicional de se encontrar B , já tendo encontrado A .

A função do Suporte é determinar a freqüência que ocorre um conjunto dentre todas as transações da Base de Dados, é a porcentagem de transações onde este conjunto aparece. Um conjunto será considerado freqüente se o seu suporte for maior ou igual a um suporte mínimo estabelecido previamente.

Uma típica abordagem para mineração de regras de associação em bases de dados consiste em encontrar todas as regras que possuam suporte e confiança maiores ou iguais, respectivamente, a um suporte mínimo (*SupMin*) e uma confiança mínima (*ConfMin*), especificados pelo usuário.

Nesta abordagem, o processo de mineração é dividido em duas etapas:

1. Determinar todos os conjuntos de itens que possuem suporte maior ou igual a *SupMin*. Estes conjuntos são chamados de **conjuntos freqüentes** (*frequent itemsets*).

2. Para cada conjunto freqüente encontrado na Etapa 1, gerar as regras de associação que possuem confiança maior ou igual a *ConfMin*.

Existem diversos algoritmos(Pivato 2006, Amo e Furtado 2005) consagrados para mineração de dados por meio de regras de associação. As implementações comerciais mais utilizadas estão representadas na Tabela 2.3. Para fins científicos e acadêmicos, os mais freqüentemente encontrados na literatura são o A priori, com sua variação dita Preditivo a priori e os algoritmos que possuem heurísticas com o objetivo de tornar a busca de regras mais eficiente; nessa classe de algoritmos se destaca o Tertius. Devido a sua importância nesse trabalho, esses algoritmos serão mais bem detalhados.

Tabela 2.3 – Sistemas comerciais utilizados para mineração de dados com regras de associação

Nome	Fabricante	Funções	Principal característica
Intelligent Miner	IBM	Algoritmos para regras de associação, classificação, regressão, padrões seqüenciais, <i>clustering</i>	Integrado com o SGBD DB2 da IBM. Grande escalabilidade dos algoritmos
MineSet	Silicon Graphics Inc.	Algoritmos para regras de associação, classificação, análise estatística.	Um robusto conjunto de ferramentas avançadas de visualização.
Clementine	Integral Solutions Ltd.	Algoritmos de regras de indução, redes neurais, classificação e ferramentas de visualização	Interface orientada ao objeto
DBMiner	DBMiner Technology Inc.	Algoritmos de regras de associação, classificação, <i>clustering</i>	<i>Data Mining</i> utilizando OLAP(Online Analytical Processing)
Genamics Expression	Genamics Developer	Algoritmos de análise de seqüências	Análise de proteínas e de seqüências de DNA

2.7.2 Algoritmo A priori

O algoritmo Apriori, proposto por Agrawal e Srikant (1993), possui três passos básicos:

- 1-) Encontra os itens individuais e seus respectivos suportes para determinar quais são freqüentes (Geração de conjuntos candidatos);
- 2-) Poda dos conjuntos candidatos;
- 3-) Contagem do Suporte.

Esses passos são executados da seguinte forma:

Inicialmente é formado o primeiro conjunto de freqüências ou itemsets. No passo subsequente, esse conjunto é utilizado para gerar novos itemsets, denominados itemsets candidatos, acrescentando um item a cada elemento do conjunto. A seleção de itemsets freqüentes a partir dos candidatos é realizada por meio do cálculo do suporte para os candidatos, desprezando aqueles com suporte abaixo do mínimo.

Esse processo é iterativo, uma vez que os itemsets freqüentes encontrados darão origem aos novos itemsets candidatos, até que não seja possível gerar novos itemsets freqüentes, pois o suporte de todos os futuros itemsets candidatos seria inferior ao suporte mínimo.

O número k de itens em um itemset é denominado tamanho do itemset e um itemset de tamanho k é representado como um k -itemset. Os itens dentro de um itemset são mantidos em ordem lexicográfica. Um k -itemset é representado por c e o conjunto de k -itemsets freqüentes e k -itemsets candidatos são representados por L_k e C_k , respectivamente.

O algoritmo para geração de itemsets freqüentes é mostrado na Figura 18. A entrada do algoritmo é constituída de um banco de dados D composto por conjunto m de itens $A=\{a_1, a_2, \dots, a_m\}$ ordenados lexicograficamente e por conjunto de n transações $T=\{t_1, t_2, \dots, t_n\}$ tal que $t_i \in T$ e $t_i \subseteq A$. Também é necessário informar o suporte mínimo sup-min .

Ao iniciar o processamento, o conjunto de itemsets contendo apenas um item é gerado para ser a origem de novos itemsets. Isso é feito contando as ocorrências de itens no banco de dados e calculando o suporte para cada item encontrado, gerando o conjunto de 1-itemset freqüente, L_1 , utilizado na primeira iteração para encontrar os itemsets candidatos C_2 .

```

begin
   $L_1 = \{1\text{-itemset freqüente}\}$ ; // Conjunto de itemsets com 1 item
  for ( $k=2; L_{k-1} \neq \emptyset ; k++$ ) do
     $C_k = \text{apriori-gen}(L_{k-1})$ ; // Novos itemsets candidatos
    forall (transação  $t \in D$ ) do
       $C_t = \text{subset}(C_k, t)$ ; // Candidatos contidos em t
      forall (Candidatos  $c \in C_t$ ) do
         $c.\text{contador}++$ ;
      end
    end
     $L_k = \{ c \in C_k \mid c.\text{contador} \geq \text{sup-min} \}$ ;
  end
  Resposta =  $\cup_k L_k$ ;
end

```

Figura 18 – Algoritmo para geração itemsets freqüentes(pseudo código).(Agrawal 1993).

O passo seguinte consiste de duas fases, executadas até não ser possível gerar novos itemsets devido ao suporte mínimo. Na primeira fase, os itemsets freqüentes L_{k-1} são usados para gerar o conjunto de itemsets candidatos C_k usando a função Apriori-gen descrita na Figura 19. Na segunda fase, o banco de dados é percorrido e para cada transação t_i , $t_i \in T$, é verificado se a transação contém itemsets candidatos. Se satisfeita a condição, o suporte do itemset candidato é incrementado. Isso é feito gerando um subconjunto C_t que possui somente k-itemsets da transação e verificando se esses k-itemsets estão presentes também em C_k . A tarefa de gerar o conjunto C_t é realizada pela função Subset. Todos os suportes para os itemsets candidatos são calculados e atualiza-se o conjunto de itemsets freqüentes L_k desprezando aqueles candidatos com suporte menor que o mínimo, pois os itemsets freqüentes L_{k-1} que não obtiverem o valor

superior ao suporte mínimo no nível $k - 1$, também não vão possuir suporte maior que o mínimo no nível k .

```

begin
  insert into Ck
  select p.item1, p.item2, ... , p.item $k-1$ , q.item $k-1$ 
  from L $k-1$  p, L $k-1$  q,
  where p.item1 = q.item1, ... , p.item  $k-2$  = q.item  $k-2$ , p.item  $k-1$  < q.item  $k-1$ ;
  /* É realizada a etapa de poda, em que todo itemset  $c \in C_k$  é removido se
  algum  $(K-1)$  subconjunto de  $c$  não pertencer a L $k-1$  */
  forall (itemset candidatoc  $\in C_k$ ) do
    forall (( $k-1$ ) – subconjuntos  $s$  do itemset candidatoc) do
      if ( $s \notin L_{k-1}$ ) then
        delete c de Ck
      end
    end
  end
end

```

Figura 19 – Função Apriori-gen, geração de itemsets candidatos (pseudo código)..(Agrawal 1993).

Na Figura 19 nota-se que a função Apriori-gen tem como argumento L_{k-1} , o conjunto de todos os $(k-1)$ -itemsets freqüentes e retorna um conjunto de todos os k -itemsets candidatos. Primeiro, no passo de associação, relaciona-se L_{k-1} com L_{k-1} para ser possível encontrar novos itemsets. Nessa etapa já é possível eliminar alguns itemsets realizando uma filtragem no conjunto C_k . A eliminação de alguns elementos é válida, pois ao se combinar os $(k-1)$ -itemsets freqüentes pode-se gerar itemsets com combinações de itens que não estão representados nas transações do banco de dados ou que não possuem suporte mínimo.

Essa situação pode ser verificada olhando os possíveis subconjuntos s constituídos de $(k-1)$ - itemsets formados a partir de um itemset candidato c , $c \in C_k$, e verificando se os itemsets de s estão entre os elementos de L_{k-1} . Assim, são eliminados todos os itemsets candidatos $c \in C_k$ tal que algum $(k-1)$ -subconjunto de c não esteja em L_{k-1} .

A função Subset do algoritmo Apriori retorna os k -itemsets candidatos que estão contidos em uma dada transação t_i . Os itemsets candidatos C_k são

armazenados em uma estrutura em árvore conhecida como *hash-tree* em que cada nó da árvore pode conter uma lista de itemsets ou uma *hash-table* (nó folha e nó intermediário, respectivamente). Partindo do nó raiz, a função encontra todos os itemsets candidatos presentes na transação t_i . Se um nó folha é atingido e o itemset encontrado está contido na transação t_i , uma referência é adicionada ao conjunto de resposta. Se um nó intermediário é atingido a partir de um item a_j , pesquisa-se em cada item após a_j em t_i . Isso é possível porque os itens estão em ordem lexicográfica. No nó raiz, todos os itens a_j em t_i são pesquisados.

Finalmente, é apresentado na Figura 20 o algoritmo para geração de regras proposto por Agrawal et al. (1996). O algoritmo tem como entrada o conjunto de itemsets freqüente LK e a medida confiança mínima *conf-min*. Para gerar as regras, considerando todos os itemsets freqüentes l , $l \in LK$, são encontrados subconjuntos a não vazios de l . Para cada subconjunto a , a saída é uma regra na forma $a \Rightarrow (l - a)$ tal que a proporção $\text{suporte}(l) / \text{suporte}(a)$ seja no mínimo igual a confiança mínima.

```

Begin
  forall (k-itemset freqüente  $lk$ ,  $k \geq 2$ ) do
    call genrules ( $lk, lk$ );
  end
end
/* genrules gera todas as regras válidas  $\tilde{a} \Rightarrow (lk - \tilde{a})$ , para todo  $\tilde{a} \subset am$  */
Procedure genrules ( $lk, am$ )
Begin
   $A = \{(m-1)\text{-itemset } am-1 \mid am-1 \subset am\}$ ;
  forall ( $am-1 \in A$ ) do
     $conf = \text{sup}(lk) / \text{sup}(am-1)$ ;
    if ( $conf \geq \text{conf-min}$ ) then
      OUTPUT: regra  $am-1 \Rightarrow (lk - am-1)$ , confiança =  $conf$  e suporte =
 $\text{sup}(lk)$ ;
      if ( $m-1 > 1$ ) then
        call genrules( $lk ; am-1$ )
      end
    end
  end
end
end

```

Figura 20 - Algoritmo Apriori em sua forma completa (pseudo código)..(Agrawal 1996).

2.7.3 Algoritmo Tertius

O algoritmo Tertius (Flach e Lachiche 1999) consiste em um algoritmo derivado em parte da idéia inicial do algoritmo Apriori mas, como principal diferencial, o Tertius faz uso de abordagens heurísticas com o objetivo de tornar a busca de regras mais eficiente. Tertius faz uso de um mecanismo de descoberta de lógica de primeira ordem para encontrar cláusulas lógicas na base de dados, onde são criadas todas as possíveis instâncias das cláusulas e então testadas nos dados.

Através de um algoritmo tipo “A*” (Russel, 2004, p97), é realizada uma busca no espaço de possibilidades das regras de associação. Executa a busca através de uma heurística e uma função de estimativa otimista, ou seja, que sempre estima o melhor resultado.

A heurística usada é um “ajuste” dos dados utilizando uma distribuição Qui quadrado χ^2 que fornece uma regra do tipo associação, ou seja $A \Rightarrow B (\alpha, \beta)$. Ainda segundo Flach e Lachiche (2001), o objetivo do algoritmo consistiu em aplicar um *best-first search* (no caso o A*), encontrando as k hipóteses mais confirmadas, também incluindo um refinamento de operadores não redundantes para anular buscas desnecessárias.

As regras formadas pelo Tertius são do tipo associação, ou seja $A \Rightarrow B (\alpha, \beta)$ com dois números associados, esses números representam respectivamente a probabilidade de confirmação de que a regra seja verdadeira e a frequência de contra-exemplos encontrados. Na Figura 21, pode-se observar a ilustração do algoritmo Tertius.

```

Begin
  Agenda <= {regra vazia}
  enquanto agenda não é vazia
    regra <= primeira regra da agenda
    se regra pode ser armazenada em resultado
      então adicione regra em resultado
    se regra pode ser armazenada em resultado
    se regra pode ser refinada
      então refina regra
      para cada filho
        calcula estimativa otimista e confirmação
        se filho pode ser armazenado em agenda
          então adicione filho em agenda
          ordene agenda conforme estimativa
  otimista
end

```

Figura 21 – Algoritmo Tertius (pseudo código).

Segundo [Deltour 2007], a Figura 21 representa a implementação original do *Tertius* (a que foi utilizada nessa tese); a agenda é iniciada com uma regra vazia e a cada refinamento é obedecida a seguinte ordem: adiciona-se um literal, unificam-se duas variáveis e instancia-se uma variável com uma constante do domínio. A cada regra refinada ocorre um decréscimo do valor da estimativa otimista para a regra.

Para o *Tertius*, o espaço de busca é uma treliça e existem diversos caminhos que levem a regra vazia inicial para uma hipótese. Considerando as mesmas clausas muitas vezes significa gerar todos os refinamentos (Gillmeister e Cazella 2007) diversas vezes antes que se perceba queda de eficiência na busca. O objetivo é achar uma árvore que cubra completamente a treliça. Na Figura 22 pode-se analisar um exemplo de árvore gerada pelo refinamento (arestas em negrito) considerando um subconjunto A,B,C.

Figura 22 - Estrutura em forma de treliça do espaço de busca no algoritmo Tertius.

2.5.2.1 Motivos da escolha do algoritmo Tertius

O algoritmo Tertius foi escolhido nesse trabalho por possuir literatura internacional comprobatória de suas características (Flach e Lachiche 2001, Frank e Kirkby 2007), além de ter um código fonte aberto (5000 linhas de código em C) o que aumenta a flexibilidade do software para eventuais mudanças que fossem necessárias. Esse algoritmo também possui uma implementação mais recente dentro do software de mineração de dados WEKA, o que permitiu testes comparativos dos resultados preliminares entre o Tertius dentro do NESISES e sua implementação no WEKA (vide anexo 1).

3 ARQUITETURA DE SOFTWARE DE NEURAL SISTEMA DE INTEGRAÇÃO DE SISTEMAS ESPECIALISTAS E SUPERVISÓRIOS “NESISES”

Este capítulo apresenta a proposta de hardware e software do Neural Sistema de Integração de Sistemas Especialistas e Supervisórios “NESISES”. Conforme é explanado no capítulo 1, o NESISES é um desenvolvimento científico e tecnológico do SISES.

O SISES é basicamente um sistema de interligação entre sistemas especialistas e sistemas de supervisão de sistemas automatizados, capaz de executar inferências a partir de regras estabelecidas em sua base de conhecimento pelo engenheiro do conhecimento, e por meio das inferências expor mensagens para os operadores do sistema automatizado e até mesmo tomar ações no sistema através do sistema supervisorio. O SISES, para se manter atualizado, precisa de periódicas atualizações por meio da engenharia do conhecimento. A Figura 7 ilustra a operacionalidade do SISES ao longo do tempo.

O NESISES por sua vez apresenta todas as funções e possibilidades do SISES somadas à capacidade de aprendizado com a operação da planta industrial.

A Figura 23 ilustra o modelo macro-funcional do NESISES em função dos eventos que ocorrem na planta ao longo do tempo. Nota-se que a eliciação do conhecimento, após as regras estarem estabelecidas, é feita automaticamente pelo módulo de aprendizado do NESISES que por sua vez altera as regras do S.E, não necessitando da constante intervenção do engenheiro do conhecimento.

Figura 23 - A funcionalidade do NESISES ao longo do tempo

Assim sendo, a arquitetura de software do NESISES foi desenvolvida integrando-se e ampliando a arquitetura de software do SISES.

3.1 Macro-Arquitetura de software do NESISES

A Figura 24 apresenta a Macro-Arquitetura de software do NESISES. Conforme ilustrado na Figura 24, o NESISES é composto do SISES mais o módulo denominado “Módulo Cognitivo Meta SISES”.

O Módulo Cognitivo Meta SISES (M.C.M.S.) deve ter a capacidade de atuar nos módulos reativos do SISES que são a base de conhecimento (B.C.) e a máquina de inferência. Na primeira versão do NESISES, a atuação do M.C.M.S. será exclusivamente na base de conhecimento.

Figura 24 - Macro- Arquitetura de software do NESISES

3.2 Módulo Cognitivo Meta SISES

A Figura 25 ilustra a arquitetura de software do módulo cognitivo meta SISES. Esse módulo será dividido em três sub-módulos:

- **Módulo Observatório (M.O.).** O módulo observatório deve ter a habilidade de observar as saídas do S.E. e/ou planta e registrá-las temporariamente, indexadas as alterações e condições de contorno da planta;
- **Módulo Analítico (M.A.).** O módulo analítico faz as comparações e medições entre os valores desejados e os valores obtidos ao longo do tempo. Assim sendo, o M.A. deverá receber as informações do M.O. e realizar os cálculos analíticos;
- **Módulo Atuador Editor Decisório (M.A.E.D.).** Este módulo, em função dos resultados obtidos do M.A., decide qual regra ou ponderação da base do conhecimento deverá ser alterada. O módulo atuador, por sua vez, é composto de dois módulos: o módulo decisório e o módulo editor *on-line* da base de conhecimento.

Figura 25 - Arquitetura de software do M.C.M.S.

A operacionalidade e o fluxo de informações e de atuações do NESISES é ilustrado na Figura 26. Note-se que a planta alimenta o SISES de dados *on line* por meio do MISS (Módulo de Interface Sistema Supervisório), o MISS por sua vez fornece os dados necessários para os tags do SISES e também para o módulo observatório. O módulo observatório verifica as saídas do SISES que podem estar afetando a planta, e também se comunica com o módulo analítico.

O módulo analítico por sua vez comunica-se com o módulo decisório que tem a incumbência de decidir sobre a alteração ou não de algum parâmetro de regra da base de conhecimento, sendo essa alteração de atuação automática.

3.3 Considerações sobre o módulo decisório do NESISES

Na Figura 26 o módulo decisório é o módulo que possui a função e a responsabilidade de alternar regras existentes ou mesmo incluir novas regras.

Esse módulo aparece com um detalhe em vermelho na Figura 26 porque será devidamente pesquisado e desenvolvido como trabalho de continuidade deste atual trabalho de pesquisa.

Esta decisão foi tomada devido ao fato de que modelos de julgamento de tomada de decisão são uma área promissora da I.A. Entretanto, diversas pesquisas ainda devem ser feitas para que haja a possibilidade de se tomarem decisões de alteração em tempo real em planta automatizada.

Observa-se que a concretização computacional de um módulo decisório é em princípio algo complicado a ser efetuado. Entretanto, os critérios de tomada de decisão cientificamente elaborados e empregados para esta área de conhecimento necessitam de investimento em tempo e esforços que não são o principal escopo deste trabalho de pesquisa.

Assim sendo, conforme será afirmado nas conclusões desse trabalho, o desenvolvimento dos mesmos são sugeridos como continuidade deste trabalho. Portanto, embora previsto no NESISES, a função de tomada de decisão nesta atual etapa é efetuada manualmente.

Figura 26 - Operacionalidade dos módulos do NESISES

Para a construção do módulo cognitivo meta SISES, foram utilizadas as metodologias pesquisadas no capítulo 2 com foco nas técnicas de mineração de dados, especialmente as regras de associação com o objetivo de explorar-se as

características de descoberta de padrões e associações entre os dados de registros não temporais.

O fluxograma mostrado na Figura 27 explicita como se deu a implementação computacional do NESISES.

Figura 27 – Fluxograma operacional do software NESISES

4 METODOLOGIA DE TESTES E VALIDAÇÃO DO NEURAL SISTEMA DE INTEGRAÇÃO DE SISTEMAS ESPECIALISTAS E SUPERVISÓRIOS “NESISES”

4.1 Introdução

Este capítulo apresenta a metodologia e as diversas etapas de testes empregadas para validação do Neural Sistema de Integração de Sistemas Especialistas e Supervisórios “NESISES”.

As macro etapas dos testes de validação do NESISES são mostradas na Figura 28, onde a principal divisão é entre os testes de simulação e os testes de campo.

Figura 28 - - Fluxograma de testes de validação do NESISES

Para executar cada uma das macro-etapas dos testes foi utilizada a metodologia proposta pela norma IEEE (IEEE, 1994). A IEEE 1059 – 1993 foi utilizado a adaptação de Pedrycz e Peters (2001) para as especificidades deste trabalho de pesquisa. Essa norma pode ser observada em termos gerais na Figura 29.

Figura 29 - - Metodologia de testes e validação IEEE 1059 – 1993

4.2 Testes de simulação

Com objetivo de depurar o NESISES resolvendo o máximo possível dos problemas inerentes ao desenvolvimento de um software, foram elaborados testes de simulação que buscavam encontrar padrões de dados conhecidos que foram gerados pela simulação.

Os testes de simulação foram desenvolvidos em duas etapas:

- a) Testes de simulação para determinar o grau de assertividade.
- b) Testes de simulação de operação de processo industrial automatizado.

4.2.1 Testes de simulação para determinação do grau de assertividade – duas variáveis

Nesta etapa a metodologia de validação da IEEE 1059 foi adaptada apenas para verificar o grau de assertividade.

A complexidade do algoritmo Tertius não permite de forma palatável determinar teoricamente o grau de confiança estatística da resposta. Assim sendo, este trabalho de pesquisa aplicou diversos testes para determinar aproximadamente de modo empírico o grau de confiabilidade do NESISES. Para tanto, foi gerada a tabela 4.1 de massa de dados relacionando cada uma das variáveis booleanas, duas a duas. A disposição estatística dos dados da Tabela 4.1 está representada nas tabelas 4.2, 4.3 e 4.4

A massa de dados que o modo analítico do NESISES recebeu foi originada por um programa em linguagem ladder desenvolvido especificamente para que o sistema supervisor aquisitasse os dados e os transferisse para o NESISES, que assim conseguiu os dados mostrados na Tabela 4.1.

O objetivo deste programa era o de gerar um modelo determinístico de referência para que as propriedades do NESISES pudessem ser aferidas de forma combinacional.

A Tabela 4.1 consiste em duas colunas onde estão a data e hora registrados pelo sistema e seis colunas de dados onde estão os valores aquisitados para verificação do grau de assertividade. Nota-se que, com o intuito de representar totalmente a massa de dados, essas colunas foram agrupadas de forma a caber quatro conjuntos de data/valor de variável em cada página.

Tabela 4.1 – Base de dados completa das variáveis V1, V2, V4, V5, V7 e V8 duas a duas

Date	Time	V1	V2	V4	V5	V7	V8	Date	Time	V1	V2	V4	V5	V7	V8	Date	Time	V1	V2	V4	V5	V7	V8	Date	Time	V1	V2	V4	V5	V7	V8
11/7/2007	18:18:19	0	0	0	0	0	0	11/7/2007	18:27:28	1	0	1	0	1	1	11/7/2007	18:35:30	1	0	1	1	1	0	11/7/2007	18:43:50	0	1	0	1	0	0
11/7/2007	18:19:17	1	0	0	0	0	0	11/7/2007	18:27:31	1	0	1	0	1	1	11/7/2007	18:35:30	1	0	1	0	1	0	11/7/2007	18:43:50	0	1	0	0	0	0
11/7/2007	18:19:17	1	1	1	0	1	1	11/7/2007	18:27:31	1	0	1	0	1	1	11/7/2007	18:35:39	1	0	1	0	1	0	11/7/2007	18:43:59	0	1	0	0	0	0
11/7/2007	18:19:20	1	1	1	0	1	1	11/7/2007	18:27:40	1	0	1	0	1	1	11/7/2007	18:35:39	1	0	1	1	1	0	11/7/2007	18:43:59	0	1	0	1	0	0
11/7/2007	18:19:20	1	1	1	1	1	1	11/7/2007	18:27:40	1	0	1	1	1	1	11/7/2007	18:35:48	1	0	1	1	1	0	11/7/2007	18:44:08	0	1	0	1	0	0
11/7/2007	18:19:30	1	1	1	1	1	1	11/7/2007	18:27:49	1	0	1	1	1	1	11/7/2007	18:35:48	1	0	1	0	1	0	11/7/2007	18:44:08	0	1	0	0	0	0
11/7/2007	18:19:30	1	1	1	0	1	1	11/7/2007	18:27:49	1	0	1	0	1	1	11/7/2007	18:36:00	0	0	1	0	1	0	11/7/2007	18:44:20	0	1	0	0	0	0
11/7/2007	18:19:39	1	1	1	0	1	1	11/7/2007	18:27:58	1	0	1	0	1	1	11/7/2007	18:36:00	0	1	0	1	0	1	11/7/2007	18:44:20	0	1	0	1	0	0
11/7/2007	18:19:39	1	1	1	1	1	1	11/7/2007	18:27:58	1	0	1	1	1	1	11/7/2007	18:36:09	0	1	0	1	0	1	11/7/2007	18:44:30	0	1	0	1	0	0
11/7/2007	18:19:48	1	1	1	0	1	1	11/7/2007	18:28:10	1	0	1	1	1	1	11/7/2007	18:36:09	0	1	0	0	0	1	11/7/2007	18:44:30	0	1	0	0	0	0
11/7/2007	18:19:51	1	1	1	0	1	1	11/7/2007	18:28:10	1	0	1	0	1	1	11/7/2007	18:36:19	0	1	0	0	0	1	11/7/2007	18:44:39	0	1	0	0	0	0
11/7/2007	18:19:51	1	1	1	0	1	1	11/7/2007	18:28:19	1	0	1	0	1	1	11/7/2007	18:36:19	0	1	0	1	0	1	11/7/2007	18:44:39	0	1	0	1	0	0
11/7/2007	18:20:00	1	1	1	0	1	1	11/7/2007	18:28:19	1	0	1	1	1	1	11/7/2007	18:36:31	0	1	0	1	0	1	11/7/2007	18:44:48	0	1	0	1	0	0
11/7/2007	18:20:00	1	1	1	1	1	1	11/7/2007	18:28:29	1	0	1	1	1	1	11/7/2007	18:36:31	0	1	0	0	0	1	11/7/2007	18:44:48	0	1	0	0	0	0
11/7/2007	18:20:09	1	1	1	1	1	1	11/7/2007	18:28:29	1	0	1	0	1	1	11/7/2007	18:36:40	0	1	0	0	0	1	11/7/2007	18:45:00	0	1	0	0	0	0
11/7/2007	18:20:09	1	1	1	0	1	1	11/7/2007	18:28:41	1	0	1	0	1	1	11/7/2007	18:36:40	0	1	0	1	0	1	11/7/2007	18:45:00	0	1	0	1	0	0
11/7/2007	18:20:18	1	1	1	0	1	1	11/7/2007	18:28:41	1	0	1	1	1	1	11/7/2007	18:36:49	0	1	0	1	0	1	11/7/2007	18:45:09	0	1	0	1	0	0
11/7/2007	18:20:18	1	1	1	1	1	1	11/7/2007	18:28:50	1	0	1	1	1	1	11/7/2007	18:36:49	0	1	0	0	0	1	11/7/2007	18:45:09	0	1	0	0	0	0
11/7/2007	18:20:31	1	1	1	1	1	1	11/7/2007	18:28:50	1	0	1	0	1	1	11/7/2007	18:36:58	0	1	0	0	0	1	11/7/2007	18:45:18	0	1	0	0	0	0
11/7/2007	18:20:31	1	1	1	0	1	1	11/7/2007	18:28:59	1	0	1	0	1	1	11/7/2007	18:36:58	0	1	0	1	0	1	11/7/2007	18:45:18	0	1	0	1	0	0
11/7/2007	18:20:40	1	1	1	0	1	1	11/7/2007	18:28:59	1	0	1	1	1	1	11/7/2007	18:37:10	0	1	0	1	0	1	11/7/2007	18:45:31	0	1	0	1	0	0
11/7/2007	18:20:40	1	1	1	1	1	1	11/7/2007	18:29:08	1	0	1	1	1	1	11/7/2007	18:37:10	0	1	0	0	0	1	11/7/2007	18:45:31	0	1	0	0	0	0
11/7/2007	18:20:49	1	1	1	1	1	1	11/7/2007	18:29:08	1	0	1	0	1	1	11/7/2007	18:37:19	0	1	0	0	0	1	11/7/2007	18:45:40	0	1	0	0	0	0
11/7/2007	18:20:49	1	1	1	0	1	1	11/7/2007	18:29:20	1	0	1	0	1	1	11/7/2007	18:37:19	0	1	0	1	0	1	11/7/2007	18:45:40	0	1	0	1	0	0
11/7/2007	18:20:58	1	1	1	1	1	1	11/7/2007	18:29:20	1	0	1	1	1	1	11/7/2007	18:37:29	0	1	0	1	0	1	11/7/2007	18:45:49	0	1	0	1	0	0
11/7/2007	18:21:10	1	1	1	0	1	1	11/7/2007	18:29:30	1	0	1	1	1	1	11/7/2007	18:37:29	0	1	0	0	0	1	11/7/2007	18:45:49	0	1	0	0	0	0
11/7/2007	18:21:10	1	1	1	0	1	1	11/7/2007	18:29:30	1	0	1	0	1	1	11/7/2007	18:37:41	0	1	0	1	0	1	11/7/2007	18:45:58	0	1	0	0	0	0
11/7/2007	18:21:19	1	1	1	1	1	1	11/7/2007	18:29:39	1	0	1	0	1	1	11/7/2007	18:37:41	0	1	0	1	0	1	11/7/2007	18:45:58	0	1	0	0	0	0
11/7/2007	18:21:19	1	1	1	1	1	1	11/7/2007	18:29:39	1	0	1	1	1	1	11/7/2007	18:37:50	0	1	0	1	0	1	11/7/2007	18:46:01	0	1	0	0	0	0
11/7/2007	18:21:28	1	1	1	1	1	1	11/7/2007	18:29:48	1	0	1	0	1	1	11/7/2007	18:37:50	0	1	0	0	0	1	11/7/2007	18:46:01	0	1	0	1	0	0
11/7/2007	18:21:28	1	1	1	0	1	1	11/7/2007	18:29:51	1	0	1	0	1	1	11/7/2007	18:37:59	0	1	0	0	0	1	11/7/2007	18:46:10	0	1	0	1	0	0

11/7/2007	18:21:41	1	1	1	0	1	1	11/7/2007	18:29:51	1	0	1	0	1	1	11/7/2007	18:37:59	0	1	0	1	0	1	11/7/2007	18:46:10	0	1	0	0	0	0
11/7/2007	18:21:41	1	1	1	1	1	1	11/7/2007	18:30:00	1	0	1	0	1	1	11/7/2007	18:38:08	0	1	0	1	0	1	11/7/2007	18:46:19	0	1	0	0	0	0
11/7/2007	18:21:50	1	1	1	0	1	1	11/7/2007	18:30:00	1	0	1	1	1	1	11/7/2007	18:38:08	0	1	0	0	0	1	11/7/2007	18:46:19	0	1	0	1	0	0
11/7/2007	18:21:59	1	1	1	0	1	1	11/7/2007	18:30:09	1	0	1	1	1	1	11/7/2007	18:38:20	0	1	0	0	0	1	11/7/2007	18:46:28	0	1	0	1	0	0
11/7/2007	18:21:59	1	1	1	1	1	1	11/7/2007	18:30:09	1	0	1	0	1	1	11/7/2007	18:38:20	0	1	0	1	0	1	11/7/2007	18:46:28	0	1	0	0	0	0
11/7/2007	18:22:08	1	1	1	1	1	1	11/7/2007	18:30:18	1	0	1	0	1	1	11/7/2007	18:38:30	0	1	0	1	0	1	11/7/2007	18:46:41	0	1	0	0	0	0
11/7/2007	18:22:08	1	1	1	0	1	1	11/7/2007	18:30:18	1	0	1	1	1	1	11/7/2007	18:38:30	0	1	0	0	0	1	11/7/2007	18:46:41	0	1	0	1	0	0
11/7/2007	18:22:20	1	1	1	0	1	1	11/7/2007	18:30:30	1	0	1	1	1	1	11/7/2007	18:38:39	0	1	0	0	0	1	11/7/2007	18:46:50	0	1	0	1	0	0
11/7/2007	18:22:20	1	1	1	1	1	1	11/7/2007	18:30:30	1	0	1	0	1	1	11/7/2007	18:38:39	0	1	0	1	0	1	11/7/2007	18:46:50	0	1	0	0	0	0
11/7/2007	18:22:29	1	1	1	1	1	1	11/7/2007	18:30:40	1	0	1	0	1	1	11/7/2007	18:38:51	0	1	0	1	0	1	11/7/2007	18:46:59	0	1	0	0	0	0
11/7/2007	18:22:29	1	1	1	0	1	1	11/7/2007	18:30:40	1	0	1	1	1	1	11/7/2007	18:38:51	0	1	0	0	0	1	11/7/2007	18:46:59	0	1	0	1	0	0
11/7/2007	18:22:38	1	1	1	0	1	1	11/7/2007	18:30:49	1	0	1	1	1	1	11/7/2007	18:39:00	0	1	0	0	0	1	11/7/2007	18:47:08	0	1	0	0	0	0
11/7/2007	18:22:38	1	1	1	1	1	1	11/7/2007	18:30:49	1	0	1	0	1	1	11/7/2007	18:39:00	0	1	0	1	0	1	11/7/2007	18:47:20	0	1	0	0	0	0
11/7/2007	18:22:51	1	1	1	1	1	1	11/7/2007	18:30:58	1	0	1	0	1	0	11/7/2007	18:39:09	0	1	0	1	0	1	11/7/2007	18:47:20	0	1	0	1	0	0
11/7/2007	18:22:51	1	1	1	0	1	1	11/7/2007	18:30:58	1	0	1	1	1	0	11/7/2007	18:39:09	0	1	0	0	0	1	11/7/2007	18:47:29	0	1	0	1	0	0
11/7/2007	18:23:00	1	1	1	0	1	1	11/7/2007	18:31:01	1	0	1	1	1	0	11/7/2007	18:39:18	0	1	0	0	0	1	11/7/2007	18:47:29	0	1	0	0	0	0

11/7/2007	18:24:40	1	0	1	1	1	1	11/7/2007	18:32:48	1	0	1	1	1	0	11/7/2007	18:40:59	0	1	0	1	0	0	11/7/2007	18:49:19	0	0	0	0	0	0
11/7/2007	18:24:49	1	0	1	1	1	1	11/7/2007	18:32:48	1	0	1	0	1	0	11/7/2007	18:41:11	0	1	0	1	0	0	11/7/2007	18:49:19	0	0	0	1	0	0
11/7/2007	18:24:49	1	0	1	0	1	1	11/7/2007	18:33:01	1	0	1	0	1	0	11/7/2007	18:41:11	0	1	0	0	0	0	11/7/2007	18:49:28	0	0	0	1	0	0
11/7/2007	18:24:58	1	0	1	0	1	1	11/7/2007	18:33:01	1	0	1	1	1	0	11/7/2007	18:41:21	0	1	0	0	0	0	11/7/2007	18:49:28	0	0	0	0	0	0
11/7/2007	18:24:58	1	0	1	1	1	1	11/7/2007	18:33:10	1	0	1	1	1	0	11/7/2007	18:41:21	0	1	0	1	0	0	11/7/2007	18:49:40	0	0	0	0	0	0
11/7/2007	18:25:11	1	0	1	1	1	1	11/7/2007	18:33:10	1	0	1	0	1	0	11/7/2007	18:41:30	0	1	0	1	0	0	11/7/2007	18:49:40	0	0	0	1	0	0
11/7/2007	18:25:11	1	0	1	0	1	1	11/7/2007	18:33:19	1	0	1	0	1	0	11/7/2007	18:41:30	0	1	0	0	0	0	11/7/2007	18:49:50	0	0	0	1	0	0
11/7/2007	18:25:20	1	0	1	0	1	1	11/7/2007	18:33:19	1	0	1	1	1	0	11/7/2007	18:41:39	0	1	0	0	0	0	11/7/2007	18:49:50	0	0	0	0	0	0
11/7/2007	18:25:20	1	0	1	1	1	1	11/7/2007	18:33:28	1	0	1	1	1	0	11/7/2007	18:41:39	0	1	0	1	0	0	11/7/2007	18:49:59	0	0	0	0	0	0
11/7/2007	18:25:29	1	0	1	1	1	1	11/7/2007	18:33:28	1	0	1	0	1	0	11/7/2007	18:41:48	0	1	0	1	0	0	11/7/2007	18:49:59	0	0	0	1	0	0
11/7/2007	18:25:29	1	0	1	0	1	1	11/7/2007	18:33:40	1	0	1	0	1	0	11/7/2007	18:41:48	0	1	0	0	0	0	11/7/2007	18:50:11	0	0	0	1	0	0
11/7/2007	18:25:38	1	0	1	0	1	1	11/7/2007	18:33:40	1	0	1	1	1	0	11/7/2007	18:42:00	0	1	0	0	0	0	11/7/2007	18:50:11	0	0	0	0	0	0
11/7/2007	18:25:38	1	0	1	1	1	1	11/7/2007	18:33:49	1	0	1	1	1	0	11/7/2007	18:42:00	0	1	0	1	0	0	11/7/2007	18:50:20	0	0	0	0	0	0
11/7/2007	18:25:50	1	0	1	1	1	1	11/7/2007	18:33:49	1	0	1	0	1	0	11/7/2007	18:42:09	0	1	0	1	0	0	11/7/2007	18:50:20	0	0	0	1	0	0
11/7/2007	18:25:50	1	0	1	0	1	1	11/7/2007	18:33:58	1	0	1	0	1	0	11/7/2007	18:42:09	0	1	0	0	0	0	11/7/2007	18:50:29	0	0	0	1	0	0
11/7/2007	18:25:59	1	0	1	0	1	1	11/7/2007	18:33:58	1	0	1	1	1	0	11/7/2007	18:42:18	0	1	0	0	0	0	11/7/2007	18:50:29	0	0	0	0	0	0

11/7/2007	18:19:17	1	1	1	0	1	1	11/7/2007	18:27:31	1	0	1	0	1	1	11/7/2007	18:35:39	1	0	1	0	1	0	11/7/2007	18:43:59	0	1	0	0	0	0
11/7/2007	18:19:20	1	1	1	0	1	1	11/7/2007	18:27:40	1	0	1	0	1	1	11/7/2007	18:35:39	1	0	1	1	1	0	11/7/2007	18:43:59	0	1	0	1	0	0
11/7/2007	18:19:20	1	1	1	1	1	1	11/7/2007	18:27:40	1	0	1	1	1	1	11/7/2007	18:35:48	1	0	1	1	1	0	11/7/2007	18:44:08	0	1	0	1	0	0
11/7/2007	18:19:30	1	1	1	1	1	1	11/7/2007	18:27:49	1	0	1	1	1	1	11/7/2007	18:35:48	1	0	1	0	1	0	11/7/2007	18:44:08	0	1	0	0	0	0
11/7/2007	18:19:30	1	1	1	0	1	1	11/7/2007	18:27:49	1	0	1	0	1	1	11/7/2007	18:36:00	0	0	1	0	1	0	11/7/2007	18:44:20	0	1	0	0	0	0
11/7/2007	18:19:39	1	1	1	0	1	1	11/7/2007	18:27:58	1	0	1	0	1	1	11/7/2007	18:36:00	0	1	0	1	0	1	11/7/2007	18:44:20	0	1	0	1	0	0
11/7/2007	18:19:39	1	1	1	1	1	1	11/7/2007	18:27:58	1	0	1	1	1	1	11/7/2007	18:36:09	0	1	0	1	0	1	11/7/2007	18:44:30	0	1	0	1	0	0
11/7/2007	18:19:48	1	1	1	0	1	1	11/7/2007	18:28:10	1	0	1	1	1	1	11/7/2007	18:36:09	0	1	0	0	0	1	11/7/2007	18:44:30	0	1	0	0	0	0
11/7/2007	18:19:51	1	1	1	0	1	1	11/7/2007	18:28:10	1	0	1	0	1	1	11/7/2007	18:36:19	0	1	0	0	0	1	11/7/2007	18:44:39	0	1	0	0	0	0
11/7/2007	18:19:51	1	1	1	0	1	1	11/7/2007	18:28:19	1	0	1	0	1	1	11/7/2007	18:36:19	0	1	0	1	0	1	11/7/2007	18:44:39	0	1	0	1	0	0
11/7/2007	18:20:00	1	1	1	0	1	1	11/7/2007	18:28:19	1	0	1	1	1	1	11/7/2007	18:36:31	0	1	0	1	0	1	11/7/2007	18:44:48	0	1	0	1	0	0
11/7/2007	18:20:00	1	1	1	1	1	1	11/7/2007	18:28:29	1	0	1	1	1	1	11/7/2007	18:36:31	0	1	0	0	0	1	11/7/2007	18:44:48	0	1	0	0	0	0
11/7/2007	18:20:09	1	1	1	1	1	1	11/7/2007	18:28:29	1	0	1	0	1	1	11/7/2007	18:36:40	0	1	0	0	0	1	11/7/2007	18:45:00	0	1	0	0	0	0
11/7/2007	18:20:09	1	1	1	0	1	1	11/7/2007	18:28:41	1	0	1	0	1	1	11/7/2007	18:36:40	0	1	0	1	0	1	11/7/2007	18:45:00	0	1	0	1	0	0
11/7/2007	18:20:18	1	1	1	0	1	1	11/7/2007	18:28:41	1	0	1	1	1	1	11/7/2007	18:36:49	0	1	0	1	0	1	11/7/2007	18:45:09	0	1	0	1	0	0
11/7/2007	18:20:18	1	1	1	1	1	1	11/7/2007	18:28:50	1	0	1	1	1	1	11/7/2007	18:36:49	0	1	0	0	0	1	11/7/2007	18:45:09	0	1	0	0	0	0

11/7/2007	18:22:20	1	1	1	0	1	1	11/7/2007	18:30:30	1	0	1	1	1	1	11/7/2007	18:38:39	0	1	0	0	0	1	11/7/2007	18:46:50	0	1	0	1	0	0
11/7/2007	18:22:20	1	1	1	1	1	1	11/7/2007	18:30:30	1	0	1	0	1	1	11/7/2007	18:38:39	0	1	0	1	0	1	11/7/2007	18:46:50	0	1	0	0	0	0
11/7/2007	18:22:29	1	1	1	1	1	1	11/7/2007	18:30:40	1	0	1	0	1	1	11/7/2007	18:38:51	0	1	0	1	0	1	11/7/2007	18:46:59	0	1	0	0	0	0
11/7/2007	18:22:29	1	1	1	0	1	1	11/7/2007	18:30:40	1	0	1	1	1	1	11/7/2007	18:38:51	0	1	0	0	0	1	11/7/2007	18:46:59	0	1	0	1	0	0
11/7/2007	18:22:38	1	1	1	0	1	1	11/7/2007	18:30:49	1	0	1	1	1	1	11/7/2007	18:39:00	0	1	0	0	0	1	11/7/2007	18:47:08	0	1	0	0	0	0
11/7/2007	18:22:38	1	1	1	1	1	1	11/7/2007	18:30:49	1	0	1	0	1	1	11/7/2007	18:39:00	0	1	0	1	0	1	11/7/2007	18:47:20	0	1	0	0	0	0
11/7/2007	18:22:51	1	1	1	1	1	1	11/7/2007	18:30:58	1	0	1	0	1	0	11/7/2007	18:39:09	0	1	0	1	0	1	11/7/2007	18:47:20	0	1	0	1	0	0
11/7/2007	18:22:51	1	1	1	0	1	1	11/7/2007	18:30:58	1	0	1	1	1	0	11/7/2007	18:39:09	0	1	0	0	0	1	11/7/2007	18:47:29	0	1	0	1	0	0
11/7/2007	18:23:00	1	1	1	0	1	1	11/7/2007	18:31:01	1	0	1	1	1	0	11/7/2007	18:39:18	0	1	0	0	0	1	11/7/2007	18:47:29	0	1	0	0	0	0
11/7/2007	18:23:00	1	1	1	1	1	1	11/7/2007	18:31:01	1	0	1	1	1	0	11/7/2007	18:39:18	0	1	0	1	0	1	11/7/2007	18:47:38	0	1	0	0	0	0
11/7/2007	18:23:09	1	1	1	1	1	1	11/7/2007	18:31:10	1	0	1	1	1	0	11/7/2007	18:39:30	0	1	0	1	0	1	11/7/2007	18:47:38	0	0	0	1	0	0
11/7/2007	18:23:09	1	1	1	0	1	1	11/7/2007	18:31:10	1	0	1	0	1	0	11/7/2007	18:39:30	0	1	0	0	0	1	11/7/2007	18:47:51	0	0	0	1	0	0
11/7/2007	18:23:18	1	1	1	0	1	1	11/7/2007	18:31:19	1	0	1	0	1	0	11/7/2007	18:39:39	0	1	0	0	0	1	11/7/2007	18:47:51	0	0	0	0	0	0
11/7/2007	18:23:18	1	1	1	1	1	1	11/7/2007	18:31:19	1	0	1	1	1	0	11/7/2007	18:39:39	0	1	0	1	0	1	11/7/2007	18:48:00	0	0	0	0	0	0
11/7/2007	18:23:30	1	1	1	1	1	1	11/7/2007	18:31:28	1	0	1	1	1	0	11/7/2007	18:39:49	0	1	0	1	0	1	11/7/2007	18:48:00	0	0	0	1	0	0
11/7/2007	18:23:30	1	1	1	0	1	1	11/7/2007	18:31:28	1	0	1	0	1	0	11/7/2007	18:39:49	0	1	0	0	0	1	11/7/2007	18:48:09	0	0	0	1	0	0

11/7/2007	18:25:20	1	0	1	0	1	1	11/7/2007	18:33:19	1	0	1	1	1	0	11/7/2007	18:41:39	0	1	0	0	0	0	11/7/2007	18:49:50	0	0	0	0	0	0
11/7/2007	18:25:20	1	0	1	1	1	1	11/7/2007	18:33:28	1	0	1	1	1	0	11/7/2007	18:41:39	0	1	0	1	0	0	11/7/2007	18:49:59	0	0	0	0	0	0
11/7/2007	18:25:29	1	0	1	1	1	1	11/7/2007	18:33:28	1	0	1	0	1	0	11/7/2007	18:41:48	0	1	0	1	0	0	11/7/2007	18:49:59	0	0	0	1	0	0
11/7/2007	18:25:29	1	0	1	0	1	1	11/7/2007	18:33:40	1	0	1	0	1	0	11/7/2007	18:41:48	0	1	0	0	0	0	11/7/2007	18:50:11	0	0	0	1	0	0
11/7/2007	18:25:38	1	0	1	0	1	1	11/7/2007	18:33:40	1	0	1	1	1	0	11/7/2007	18:42:00	0	1	0	0	0	0	11/7/2007	18:50:11	0	0	0	0	0	0
11/7/2007	18:25:38	1	0	1	1	1	1	11/7/2007	18:33:49	1	0	1	1	1	0	11/7/2007	18:42:00	0	1	0	1	0	0	11/7/2007	18:50:20	0	0	0	0	0	0
11/7/2007	18:25:50	1	0	1	1	1	1	11/7/2007	18:33:49	1	0	1	0	1	0	11/7/2007	18:42:09	0	1	0	1	0	0	11/7/2007	18:50:20	0	0	0	1	0	0
11/7/2007	18:25:50	1	0	1	0	1	1	11/7/2007	18:33:58	1	0	1	0	1	0	11/7/2007	18:42:09	0	1	0	0	0	0	11/7/2007	18:50:29	0	0	0	1	0	0
11/7/2007	18:25:59	1	0	1	0	1	1	11/7/2007	18:33:58	1	0	1	1	1	0	11/7/2007	18:42:18	0	1	0	0	0	0	11/7/2007	18:50:29	0	0	0	0	0	0
11/7/2007	18:25:59	1	0	1	1	1	1	11/7/2007	18:34:11	1	0	1	1	1	0	11/7/2007	18:42:18	0	1	0	1	0	0	11/7/2007	18:50:38	0	0	0	0	0	0
11/7/2007	18:26:09	1	0	1	1	1	1	11/7/2007	18:34:11	1	0	1	0	1	0	11/7/2007	18:42:31	0	1	0	1	0	0	11/7/2007	18:50:38	0	0	0	1	0	0
11/7/2007	18:26:09	1	0	1	0	1	1	11/7/2007	18:34:20	1	0	1	0	1	0	11/7/2007	18:42:31	0	1	0	0	0	0	11/7/2007	18:50:50	0	0	0	1	0	0
11/7/2007	18:26:21	1	0	1	0	1	1	11/7/2007	18:34:20	1	0	1	1	1	0	11/7/2007	18:42:40	0	1	0	0	0	0	11/7/2007	18:50:50	0	0	0	0	0	0
11/7/2007	18:26:21	1	0	1	1	1	1	11/7/2007	18:34:29	1	0	1	1	1	0	11/7/2007	18:42:40	0	1	0	1	0	0	11/7/2007	18:51:00	0	0	0	0	0	0
11/7/2007	18:26:30	1	0	1	1	1	1	11/7/2007	18:34:29	1	0	1	0	1	0	11/7/2007	18:42:49	0	1	0	1	0	0	11/7/2007	18:51:00	0	0	0	1	0	0
11/7/2007	18:26:30	1	0	1	0	1	1	11/7/2007	18:34:38	1	0	1	1	1	0	11/7/2007	18:42:49	0	1	0	0	0	0	11/7/2007	18:51:09	0	0	0	1	0	0
11/7/2007	18:26:39	1	0	1	0	1	1	11/7/2007	18:34:41	1	0	1	1	1	0	11/7/2007	18:42:58	0	1	0	0	0	0	11/7/2007	18:51:09	0	0	0	0	0	0
11/7/2007	18:26:39	1	0	1	1	1	1	11/7/2007	18:34:41	1	0	1	1	1	0	11/7/2007	18:42:58	0	1	0	1	0	0	11/7/2007	18:51:18	0	0	0	1	0	0
11/7/2007	18:26:48	1	0	1	1	1	1	11/7/2007	18:34:50	1	0	1	1	1	0	11/7/2007	18:43:10	0	1	0	1	0	0	11/7/2007	18:51:21	0	0	0	1	0	0
11/7/2007	18:26:48	1	0	1	0	1	1	11/7/2007	18:34:50	1	0	1	0	1	0	11/7/2007	18:43:10	0	1	0	0	0	0	11/7/2007	18:51:21	0	0	0	1	0	0
11/7/2007	18:27:00	1	0	1	0	1	1	11/7/2007	18:34:59	1	0	1	0	1	0	11/7/2007	18:43:19	0	1	0	0	0	0	11/7/2007	18:51:30	0	0	0	1	0	0
11/7/2007	18:27:00	1	0	1	1	1	1	11/7/2007	18:34:59	1	0	1	1	1	0	11/7/2007	18:43:19	0	1	0	1	0	0	11/7/2007	18:51:30	0	0	0	0	0	0
11/7/2007	18:27:10	1	0	1	1	1	1	11/7/2007	18:35:09	1	0	1	1	1	0	11/7/2007	18:43:29	0	1	0	1	0	0	11/7/2007	18:51:39	0	0	0	0	0	0
11/7/2007	18:27:10	1	0	1	0	1	1	11/7/2007	18:35:09	1	0	1	0	1	0	11/7/2007	18:43:29	0	1	0	0	0	0	11/7/2007	18:51:39	0	0	0	1	0	0
11/7/2007	18:27:19	1	0	1	0	1	1	11/7/2007	18:35:21	1	0	1	0	1	0	11/7/2007	18:43:41	0	1	0	0	0	0	11/7/2007	18:51:48	0	0	0	1	0	0
11/7/2007	18:27:19	1	0	1	1	1	1	11/7/2007	18:35:21	1	0	1	1	1	0	11/7/2007	18:43:41	0	1	0	1	0	0	11/7/2007	18:51:48	0	0	0	0	0	0

Conforme pode-se observar, as Tabelas 4.2, 4.3, e 4.4 resumem em duas regras cada informação relevante (programada arbitrariamente no ladder) presente na massa de dados representada na Tabela 4.1.

Tabela 4.2 – Primeiro conjunto simulado de variáveis 2 a 2

Variável V1	V2
V1 = 1	30% das vezes V2 = 1
V1 = 0	70% das vezes V2 = 1

Tabela 4.3 – Segundo conjunto simulado de variáveis 2 a 2

Variável V4	V5
V4 = 1	50% das vezes V5 = 1
V4 = 0	50% das vezes V5 = 1

Tabela 4.4 – Terceiro conjunto simulado de variáveis 2 a 2

Variável V7	V8
V7 = 1	70% das vezes V8 = 1
V7 = 0	30% das vezes V8 = 1

Após a simulação, o NESISES apresentou os seguintes resultados, identificados como regras de correlação entre as variáveis V1 e V2, V4 e V5, V7 e V8. As mesmas estão representadas na Tabela 4.6.

A sintaxe empregada pelo NESISES para representar suas regras é mostrada na Tabela 4.5.

Tabela 4.5 – Sintaxe das regras do NESISES

Símbolo	Significado
->	Indica implicância, ou seja, A implica em B.
and	É o operador lógico e
or	É o operador lógico ou

A Tabela 4.6 apresenta o resultado das regras apresentadas nas tabelas 4.2, 4.3 e 4.4. Observe-se que a Tabela 4.6 é a tabela apresentada pela interface do NESISES. A coluna à direita desta tabela foi acrescentada apenas para facilitar o entendimento.

Tabela 4.6 – Regras encontradas pelo NESISES

Referente à tabela	Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
4.2	1	/* 0.332298	0.0027857 */	V2(true) -> V1(true)
		/* 0.332298	0.0027857 */	V1(false) -> V2(false)
	2	/* 0.339633	0.0034568 */	V2(false) -> V1(false)
		/* 0.339633	0.0034568 */	V1(true) -> V2(true)
4.3	3	/* 0.009803	0.483092*/	V5(false) -> V4(true)
		/* 0.009803	0.483092*/	V4(false) -> V5(true)
	4	/* 0.009803	0.507246 */	V5(true) -> V4(false)
		/* 0.009803	0.507246 */	V4(true) -> V5(false)
4.4	5	/* 0.745586	0.014928 */	V8(false) -> V7(false)
		/* 0.745586	0.014928 */	V7(true) -> V8(true)
	6	/* 0.745584	0.015174 */	V8(true) -> V7(true)
		/* 0.745584	0.015174 */	V7(false) -> V8(false)

Observa-se que o NESISES foi capaz de identificar que a variável V2 é “33%” das vezes 1 quando a variável V1 for 1 (regras 1 e 2). O NESISES também foi capaz de identificar que a variável V4 é “50%” das vezes 1 quando a variável V5 for 1 (regras 3 e 4) e o NESISES também foi capaz de identificar que a variável V7 é “74%” das vezes 1 quando a variável V8 for 1 (regras 5 e 6). As regras encontradas e explicitadas na Tabela 4.6 diferem das regras inseridas na massa de dados no máximo em 4%. Assim sendo, pode-se concluir, de forma empírica, que a confiabilidade do NESISES é da ordem de 95% sem que haja um significativo erro nesta afirmação.

A Tabela 4.6 contém as regras empregadas como sendo entrada de dados do módulo analítico do NESISES.

4.2.2 Testes de simulação para determinação do grau de assertividade - três variáveis

No segundo teste de assertividade, a massa de dados foi aumentada em sua complexidade com as variáveis e distribuições mostradas nas tabelas 4.7, 4.8, 4.9, 4.10, 4.11, onde V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15 são variáveis também do tipo booleanas.

A tabelas 4.7, 4.8, 4.9, 4.10 e 4.11 ilustram de modo simplificado os dados porque na realidade eles eram compostos por centenas de linhas, a exemplo da Tabela 4.1. Essas tabelas apresentam as correlações das diversas variáveis (V1 a V15).

Tabela 4.7 – Primeiro conjunto simulado de regras NESISES

Variável V1	V2	V3
1	30% das vezes	70% das vezes
0	70% das vezes	30% das vezes

Tabela 4.8 – Segundo conjunto simulado de regras NESISES

Variável V4	V5	V6
1	50% das vezes	50% das vezes
0	50% das vezes	50% das vezes

Tabela 4.9 – Terceiro conjunto simulado de regras NESISES

Variável V7	V8	V9
1	70% das vezes	30% das vezes
0	30% das vezes	70% das vezes

Tabela 4.10 – Quarto conjunto simulado de regras NESISES

Variável V10	V11	V12
1	40% das vezes	40% das vezes
0	60% das vezes	60% das vezes

Tabela 4.11 – Quinto conjunto simulado de regras NESISES

Variável V13	V14	V15
1	60% das vezes	60% das vezes
0	40% das vezes	40% das vezes

Cada uma das tabelas-resumo era composta por duzentas linhas por três colunas de dados que seguiam a regra apresentada. Os dados obtidos pelo NESISES estão apresentados nas tabelas 4.11, 4.12, 4.13, 4.14, 4.15, 4.16.

Tabela 4.12 – Regras encontradas pelo NESISES para o primeiro conjunto simulado

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.462444	0.000000 */	V1(true) :- V2(false) AND V3(true).
	/* 0.462444	0.000000 */	V2(true) OR V3(false) :- V1(false)
2	/* 0.451920	0.000000 */	V1(false) :- V2(true) AND V3(false)
	/* 0.451920	0.000000 */	V2(false) OR V3(true) :- V1(true)
3	/* 0.405586	0.144928 */	V3(false) :- V1(false)
	/* 0.405586	0.144928 */	V1(true) :- V3(true)
4	/* 0.405584	0.152174 */	V3(true) :- V1(true)
	/* 0.405584	0.152174 */	V1(false) :- V3(false)
5	/* 0.394882	0.000000 */	V1(true) OR V2(true) :- V3(true)
	/* 0.394882	0.000000 */	V3(false) :- V1(false) AND V2(false)
6	/* 0.382066	0.144928 */	V2(false) :- V1(true)
	/* 0.382066	0.144928 */	V1(false) :- V2(true)
7	/* 0.382065	0.164251 */	V2(true) :- V1(false)
	/* 0.382065	0.164251 */	V1(true) :- V2(false)
8	/* 0.375638	0.000000 */	V1(true) OR V3(false) :- V2(false)
	/* 0.375638	0.000000 */	V2(true) :- V1(false) AND V3(true)
9	/* 0.374424	0.000000 */	V1(false) OR V2(false) :- V3(false)
	/* 0.374424	0.000000 */	V3(true) :- V1(true) AND V2(true)
10	/* 0.373271	0.000000 */	V1(false) OR V3(true) :- V2(true)
	/* 0.373271	0.000000 */	V2(false) :- V1(true) AND V3(false)

Observando-se a Tabela 4.7 nota-se que a regra de maior probabilidade é: quando V1 for verdadeiro, implica que V2 é falso e V3 é verdadeiro, considerando que, pela tabela 4.5, quando V1 for verdadeiro, V2 será em 70% das vezes falso e V3 será 70% das vezes verdadeiro; essa correlação é representada na Tabela 4.12 por uma probabilidade condicional que se traduz numericamente por uma multiplicação de probabilidades, ou seja, no caso $0.7 * 0.7 = 0.49$, que está apenas 2,7 % da regra encontrada (0.46244). Nota-se ainda que o percentual de contra-exemplos desta regra (0%) é extremamente baixo.

Tabela 4.13 – Regras encontradas pelo NESISES para o segundo conjunto simulado

Regra	Confirmação	Frequência de contra exemplos	Regra encontrada
1	/* 0.907938	0.021739 */	V5(true) :- V6(true)
	/* 0.907938	0.021739 */	V6(false) :- V5(false)
2	/* 0.907937	0.024155 */	V5(false) :- V6(false)
	/* 0.907937	0.024155 */	V6(true) :- V5(true)
3	/* 0.525691	0.004831 */	V4(true) OR V6(false) :- V5(false)
	/* 0.525691	0.004831 */	V5(true) :- V4(false) AND V6(true)
4	/* 0.523595	0.004831 */	V4(true) OR V5(true) :- V6(true)
	/* 0.523595	0.004831 */	V6(false) :- V4(false) AND V5(false)
5	/* 0.515134	0.009662 */	V4(true) OR V5(false) :- V6(false)
	/* 0.515134	0.009662 */	V6(true) :- V4(false) AND V5(true)
6	/* 0.512792	0.009662 */	V4(true) OR V6(true) :- V5(true)
	/* 0.512792	0.009662 */	V5(false) :- V4(false) AND V6(false)
7	/* 0.479250	0.014493 */	V4(false) OR V6(true) :- V5(true)
	/* 0.479250	0.014493 */	V5(false) :- V4(true) AND V6(false)
8	/* 0.476881	0.014493 */	V4(false) OR V5(false) :- V6(false)
	/* 0.476881	0.014493 */	V6(true) :- V4(true) AND V5(true)
9	/* 0.470691	0.016908 */	V4(false) OR V5(true) :- V6(true)
	/* 0.470691	0.016908 */	V6(false) :- V4(true) AND V5(false)
10	/* 0.468535	0.016908 */	V4(false) OR V6(false) :- V5(false)
	/* 0.468535	0.016908 */	V5(true) :- V4(true) AND V6(true)

Observando-se a Tabela 4.8, nota-se pelas regras 3 e 4 que quando V5 ou V6 são verdadeiros ou falsos, existe uma probabilidade muito próxima de 50% de V4 ser verdadeira ou falsa. Essa correlação é representada na Tabela 4.13 por uma probabilidade condicional que se traduz numericamente por uma multiplicação de probabilidades, ou seja, neste caso está apenas 2,3 % da probabilidade programada

na simulação. Nota-se ainda que o percentual de contra-exemplos desta regra (0,5%) é extremamente baixo.

Tabela 4.14 – Regras encontradas pelo NESISES para o terceiro conjunto simulado

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.458319	0.000000 */	V7(true) :- V8(true) AND V9(false)
	/* 0.458319	0.000000 */	V8(false) OR V9(true) :- V7(false)
2	/* 0.451920	0.000000 */	V7(false) :- V8(false) AND V9(true)
	/* 0.451920	0.000000 */	V8(true) OR V9(false) :- V7(true)
3	/* 0.405586	0.144928 */	V8(false) :- V7(false)
	/* 0.405586	0.144928 */	V7(true) :- V8(true)
4	/* 0.405584	0.152174 */	V8(true) :- V7(true)
	/* 0.405584	0.152174 */	V7(false) :- V8(false)
5	/* 0.394882	0.000000 */	V7(true) OR V9(true) :- V8(true)
	/* 0.394882	0.000000 */	V8(false) :- V7(false) AND V9(false)
6	/* 0.378708	0.000000 */	V7(false) OR V9(false) :- V8(false)
	/* 0.378708	0.000000 */	V8(true) :- V7(true) AND V9(true)
7	/* 0.377153	0.147343 */	V9(false) :- V7(true)
	/* 0.377153	0.147343 */	V7(false) :- V9(true)
8	/* 0.377152	0.164251 */	V9(true) :- V7(false)
	/* 0.377152	0.164251 */	V7(true) :- V9(false)
9	/* 0.374546	0.000000 */	V7(false) OR V8(true) :- V9(true)
	/* 0.374546	0.000000 */	V9(false) :- V7(true) AND V8(false)
10	/* 0.374424	0.000000 */	V7(true) OR V8(false) :- V9(false)
	/* 0.374424	0.000000 */	V9(true) :- V7(false) AND V8(true)

Observando-se a Tabela 4.9, nota-se que a regra de maior probabilidade é: quando V7 for verdadeiro, implica que V9 é falso e V8 é verdadeiro. Essa correlação é representada na Tabela 4.14 por uma probabilidade condicional que se traduz numericamente por uma multiplicação de probabilidades, ou seja, no caso $0.7 * 0.7 = 0.49$ que está apenas 3,17 % da regra encontrada (0.4583). Nota-se ainda que o percentual de contra-exemplos desta regra (0,0%) é extremamente baixo.

Tabela 4.15 – Regras encontradas pelo NESISES para o quarto conjunto simulado

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.989247	0.000000 */	V12(true) :- V11(true)
	/* 0.989247	0.000000 */	V11(false) :- V12(false)
2	/* 0.989236	0.004831 */	V12(false) :- V11(false)
	/* 0.989236	0.004831 */	V11(true) :- V12(true)
3	/* 0.623762	0.002415 */	V10(true) OR V12(false) :- V11(false)
	/* 0.623762	0.002415 */	V11(true) :- V10(false) AND V12(true)
4	/* 0.610184	0.002415 */	V10(false) OR V11(true) :- V12(true)
	/* 0.610184	0.002415 */	V12(false) :- V10(true) AND V11(false)
5	/* 0.470199	0.002415 */	V10(true) OR V11(true) :- V12(true)
	/* 0.470199	0.002415 */	V12(false) :- V10(false) AND V11(false)
6	/* 0.456784	0.002415 */	V10(false) OR V12(false) :- V11(false)
	/* 0.456784	0.002415 */	V11(true) :- V10(true) AND V12(true)
7	/* 0.179223	0.193237 */	V11(false) :- V10(true)
	/* 0.179223	0.193237 */	V10(false) :- V11(true)
8	/* 0.179221	0.217391 */	V11(true) :- V10(false)
	/* 0.179221	0.217391 */	V10(true) :- V11(false)
9	/* 0.179091	0.195652 */	V12(false) :- V10(true)
	/* 0.179091	0.195652 */	V10(false) :- V12(true)
10	/* 0.179091	0.214976 */	V12(true) :- V10(false)
	/* 0.179091	0.214976 */	V10(true) :- V12(false)

Observando-se a Tabela 4.10 nota-se pelas regras 3 e 4 que quando V11 ou V12 são verdadeiros **ou** falsos, existe uma probabilidade muito próxima de 60% de V10 ser verdadeira **ou** falsa. Essa correlação é representada na Tabela 4.16, existe um probabilidade muito próxima de 60% de V10 for verdadeira ou falsa que está apenas 2,3 % da probabilidade programada na simulação. Nota-se ainda que o percentual de contra-exemplos desta regra (0,24%) é extremamente baixo.

Tabela 4.16 – Regras encontradas pelo NESISES para o quinto conjunto simulado

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.994197	0.000000 */	V15(true) :- V14(true)
	/* 0.994197	0.000000 */	V14(false) :- V15(false)
2	/* 0.994193	0.002415 */	V15(false) :- V14(false)
	/* 0.994193	0.002415 */	V14(true) :- V15(true)
3	/* 0.640564	0.000000 */	V13(true) OR V14(true) :- V15(true)
	/* 0.640564	0.000000 */	V15(false) :- V13(false) AND V14(false)
4	/* 0.457865	0.000000 */	V13(true) OR V15(false) :- V14(false)
	/* 0.457865	0.000000 */	V14(true) :- V13(false) AND V15(true)
5	/* 0.217132	0.190821 */	V15(false) :- V13(false)
	/* 0.217132	0.190821 */	V13(true) :- V15(true)
6	/* 0.217131	0.200483 */	V15(true) :- V13(true)
	/* 0.217131	0.200483 */	V13(false) :- V15(false)
7	/* 0.212266	0.190821 */	V14(false) :- V13(false)
	/* 0.212266	0.190821 */	V13(true) :- V14(true)
8	/* 0.212263	0.202899 */	V14(true) :- V13(true)
	/* 0.212263	0.202899 */	V13(false) :- V14(false)
9	/* 0.036183	0.000000 */	V13(true) :- V14(false) AND V15(true)
	/* 0.036183	0.000000 */	V14(true) OR V15(false) :- V13(false)

Observando-se a Tabela 4.11, nota-se pelas regras 3 e 4 que quando V14 ou V15 são verdadeiros **ou** falsos, existe uma probabilidade muito próxima de 60% de V13 ser verdadeira **ou** falsa. Essa correlação é representada na Tabela 4.16, que está apenas 4 % da probabilidade programada na simulação. Nota-se ainda que o percentual de contra-exemplos desta regra (0,0%) é extremamente baixo.

Assim sendo, pode-se concluir de forma empírica que a confiabilidade do NESISES é da ordem de 95% sem que haja um significativo erro nesta afirmação. Esse nível de confiabilidade é perfeitamente aceitável para os objetivos acadêmicos deste trabalho de pesquisa e também com os níveis de percepção dos operadores humanos quando atuam em sistemas supervisórios complexos.

Portanto, o NESISES é capaz de “aprender” ou deduzir regras a partir de uma massa de dados dinâmica e complexa. É razoável supor que o grau de confiabilidade das regras aprendidas seja de pelo menos 95%, conforme provado empiricamente no item 4.2.1 deste trabalho.

Um dado relevante é que rodando com computadores atuais, dependendo do número de variáveis (tempo exponencial) e número de regras (tempo linear) o tempo de convergência do módulo de mineração de dados do NESISES pode chegar a 36h ininterruptas de processamento utilizando 100% da capacidade do processador com uma utilização crescente da memória RAM.

Este tempo não é impeditivo, uma vez que um engenheiro do conhecimento pode muitas vezes necessitar de muito mais tempo ou simplesmente não identificar as regras mais relevantes. Este tempo também tende a diminuir com a crescente evolução no quesito aumento de velocidade dos computadores atuais.

Outro fato é que a identificação, ou seja, o aprendizado, ocorre de forma paralela à operação do NESISES com um sistema especialista tradicional, desde que a etapa de mineração de dados não seja executada na mesma máquina ao mesmo tempo em que os S.S.s e o S.E. estejam operando.

4.2.3 Testes de simulação de processo industrial automatizado

Nesta etapa a metodologia de validação de testes IEEE 1059 foi aplicado integralmente em processo industrial de secagem de náilon, previamente estudado(Andrade 2001) cujas características de configuração de software e complexidade o tornaram adequado aos objetivos de teste e validação.

4.2.4 Análise de rastreabilidade dos testes de simulação de processo industrial automatizado

Rastreabilidade (ISO 8402) é definida como a habilidade de descrever a história, aplicação, processos ou eventos e localização de um produto, a uma determinada organização, por meio de registros e identificação. De um modo mais simples, rastrear é manter os registros necessários para identificar e informar os dados relativos à origem e ao destino de um produto ou processo.

Inicialmente, partiu-se de uma estrutura de dados que permitisse a rastreabilidade e foi implementado no software NESISES uma estrutura em árvore que documenta cada uma de suas tomadas de decisão. Para os testes de

simulação em si, foi executado o seguinte procedimento para garantir a rastreabilidade:

a) Os eventos que seriam implementados no simulador foram definidos e anotados antes de sua implementação em linguagem ladder interna ao simulador.

b) A aplicação desenvolvida no software de criação de S.S.s Rsview (Rockwell Software, 2007), software esse doado ao Convênio EPUSP-Rockwell Automation, foi configurada de forma a adquirir e armazenar automaticamente todos os eventos que ocorressem; esses eventos (registrados em base de dados) foram armazenados em arquivos de dados do tipo.DBF, sendo que esses arquivos seguem a seguinte forma:

b.1) Dois arquivos para cada dia de aquisição de dados, por exemplo: "2007 05 11 0000 (Tagname).DBF" e "2007 05 11 0000 (Wide).DBF". Note-se que no próprio nome do arquivo está a data em que foi gerado, além disso existem dois tipos de arquivos, cujos nomes são gerados automaticamente pelo Rsview, *Tagname* e *Wide* sendo *tagname* onde estão as variáveis que serão armazenadas e *wide* os dados que são armazenados.

b.2) Os arquivos tipo *wide* possuem estrutura interna em forma de tabela, sendo que a primeira e a segunda coluna guardam a data e hora/minuto/segundo/milésimo de segundo em que ocorreu o evento. As demais colunas são para armazenamento dos eventos na sequência determinada pelo arquivo *tagname*.

c) A cada vez que o algoritmo de mineração de dados foi rodado anotou-se sua parametrização e guardou-se o arquivo gerado que possui a extensão .RES.

d) Todos os arquivos gerados pela simulação, periodicamente, foram copiados e guardados em um computador diferente do qual estava rodando a simulação para evitar-se o risco de algum sobrescrito acidental.

O fluxograma mostrado na Figura 30 descreve a sequência de eventos que permitem a rastreabilidade dos testes de software.

Figura 30 - Fluxograma da rastreabilidade dos testes de simulação de processo industrial automatizado

4.2.5 Avaliação dos testes de simulação de processo industrial automatizado

Os dados foram avaliados em cada item citado na rastreabilidade, seguiu-se uma metodologia do tipo caixa branca (Pressman, 2001), onde cada parte do software é avaliada no sentido de ter de fazer corretamente sua especificação. Nos testes de simulação, a rotina interna do simulador gerava os eventos, os arquivos.DBF eram gerados diariamente, os dados desses arquivos eram verificados no sentido de correção, consistência, completude e acurácia.

Nessa etapa foram verificados também os resultados da etapa de mineração de dados onde os dados que estavam nos arquivos.RES foram analisados no que foi uma metodologia tipo caixa preta (Pressman, 2001), ou seja, o foco é apenas nos resultados sem entrar no mérito das etapas intermediárias.

Foram gerados 25 arquivos entre 02/04/2007 a 26/04/2007 totalizando 594h e o índice de falhas em cada um dos quesitos prescritos pela norma foi zero.

4.2.6 Análise de interface dos testes de simulação de processo industrial automatizado

Nessa etapa foram analisados os dados das diversas interfaces de cada parte do software e verificadas sua consistência, correção, completude e acurácia. Nessa etapa foi executada uma grande quantidade de modificações e inclusões no

software desenvolvido, entre as quais pode-se citar um mecanismo de busca para a localização das pastas onde os arquivos.DBF estão, o intervalo de tempo e os parâmetros do algoritmo de mineração de dados; a versão da interface utilizada nessa etapa está na Figura 31 onde se pode observar as principais características da interface.

A criação de uma interface que atendesse critérios de usabilidade (Fernandes,2007) e funcionalidade não foi o foco principal desse trabalho; entretanto, como se pode observar a Figura 31, buscou-se uma interface o mais fácil possível de ser utilizada.

Figura 31 – Interface do NESISES – módulo de mineração de dados na fase dos testes de simulação de processo industrial automatizado

4.2.7 Geração de plano e desenho de teste dos testes de simulação de processo industrial automatizado

Essas etapas que na norma IEEE 1059 – 1993 eram distintas, foram agrupadas de acordo com as necessidades desse trabalho de pesquisa.

Na etapa executada de acordo com o fluxograma mostrado na Figura 32, nota-se que o processo é iterativo.

Figura 32 – Plano e desenho de testes do NESISES na fase dos testes de simulação de processo industrial automatizado

4.3 Definição dos critérios de aprovação dos testes de simulação com o NESISES

Para se validarem os testes, tanto os da simulação quanto os de campo, foram adotados os seguintes critérios:

1. *Assertividade do aprendizado*: o NESISES tem de possuir aprendizado assertivo.
2. *Desempenho do S.S.*: o S.S. não deve sofrer queda de desempenho apreciável.
3. *Tempo de execução*: o tempo para o NESISES aprender algo do S.S. não pode ser de uma ordem de grandeza superior aos tempos padronizados que os engenheiros do conhecimento têm para atingir as mesmas conclusões.
4. *Compatibilidade*: o NESISES não pode requerer tantos recursos do sistema operacional (Windows) que prejudique seu desempenho. Ou seja, o funcionamento do NESISES não pode prejudicar o funcionamento normal do S.S. em suas tarefas.

4.3.1 Aprendizado assertivo dos testes de simulação de processo industrial automatizado

Nesta fase do trabalho, foi considerado aprendizado assertivo a capacidade do NESISES de alterar ou sugerir regras relevantes para a operação do sistema automatizado.

Para tanto, foi necessário previamente estabelecer que todo sistema automatizado obedece a um algoritmo de automação explícito determinado de forma consciente ou não pelo engenheiro de automação que programa as lógicas de controle, discreto ou dinâmico nos CLPs, SDCDs, CAPs e Sistemas Supervisórios.

Este algoritmo de controle é denominado neste trabalho de pesquisa como algoritmo de controle baseado no conhecimento explícito do sistema de automação. Entretanto, a automação de qualquer planta industrial nem sempre obtém de forma

direta todos os descritivos e interações entre os diversos equipamentos e processos da planta, por exemplo:

No processo de secagem de náilon representado na Figura 33, as válvulas denominadas V-20, V-32, V-33 e TCV-1 têm uma lógica de intertravamento associada, pois a abertura das válvulas V-20, V-32 caracteriza processo de resfriamento, enquanto a abertura das válvulas V-33 e TCV-1 caracteriza aquecimento. Se ocorrer abertura das válvulas de aquecimento e resfriamento simultaneamente, ocorrerão neste processo específico danos físicos ao sistema com eventual parada na produção. O intertravamento das válvulas está devidamente programado no CLP que controla o processo, porém existe uma relação de causa-efeito que não está programada no CLP que é a de que, ao ser ligada a refrigeração (abertura das válvulas V-20, V-32) existe um aquecimento da água de refrigeração que por sua vez causa o acionamento de um compressor adicional ao já normalmente ativo para manter a água de refrigeração fria. O acionamento do compressor adicional ocorre sempre alguns segundos após o seqüenciamento das válvulas, mas essa relação não está programada diretamente no ladder.

Figura 33 – Processo de secagem de náilon

Assim sendo, uma série de ocorrências e decorrências de eventos aleatórios ou previstos interagem de uma forma indireta e também de forma direta com o algoritmo de controle estabelecido nos dispositivos de controle de processo como os CLPs e S.S.s.

Estes eventos e conseqüências são definidos neste trabalho como algoritmo de controle implícito, que muitas vezes somente se torna conhecido ou parcialmente conhecido pelos operadores do sistema e pelo engenheiro de automação após um determinado tempo de operação. Esta é uma das razões pelas quais um correto dimensionamento do tempo de acompanhamento do sistema de automação, conhecido na área como operação assistida, é fundamental para o sucesso da automação.

O aspecto mais importante do critério de assertividade no aprendizado definido nessa etapa é a capacidade do NESISES identificar padrões de comportamento operacional do sistema de automação, ou seja, o NESISES tem de encontrar o algoritmo de controle implícito e explícito do sistema automatizado.

Assim sendo, foi elaborada uma arquitetura onde um emulador de CLP interage com um S.S. que gera a base dos dados que será analisada pelo NESISES e também interage com o S.E. rodando no NESISES. A Figura 34 ilustra essas interações.

Figura 34 – Arquitetura básica dos testes de simulação

O software emulador empregado neste trabalho foi o RSLogix Emulate 5 (Rockwell Software 1996), devido a características técnicas que permitem uma simulação de processos complexos fiéis à realidade industrial. Esse software possibilita emular CLPs da marca Allen Bradley da família 5, tendo por base os programas que serão rodados nos CLPs reais, ou seja, ele processa as linhas de

ladder (Michel, 1990) ou os blocos de um SFC (*Sequential Flow Chart*). Esse emulador, da mesma maneira que os CLPs Allen Bradley, utiliza como meio de troca de dados o software de comunicação RSLinx (Rockwell Software 2001), software que gerencia toda a troca de dados sendo, inclusive, o responsável por aquela que acontece entre os CLPs e o S.S..

O emulador de CLPs foi programado para gerar uma série de ocorrências e decorrências e o NESISES deveria identificar um padrão estatístico relevante e elaborar automaticamente regras a partir da base de dados que foram formadas pela aquisição desses padrões pelo S.S..

Os padrões iniciais são mostrados na Tabela 4.17, e cada seqüência é gerada pelo emulador de forma que todas as variáveis em questão são modificadas ao mesmo tempo, a cada seqüência. O emulador do CLP executa um algoritmo (**vide anexo 2**) de forma que as sete variáveis internas são alteradas concomitantemente.

Essas sete variáveis são registradas pelo S.S., cada uma delas sendo as “tags”:

V-20, V-32, V-33, PV-15, V-53, AP-2, V-20.

Cada uma das tags do S.S. é espelhada por uma variável NESISES denominada:

V1, V2, V3, V4, V5, V6, V7

Assim sendo, as variáveis NESISES espelham as tags do S.S. que espelham as sete variáveis controladas pelo emulador do CLP. A seqüência 1 é automaticamente alterada para a seqüência 2 e assim por diante até a seqüência 6 quando então é alterada para a seqüência 1.

Todo este ciclo é repetido em “moto perpétuo” e foi programado nos testes para levar aproximadamente 20 minutos.

Tabela 4.17 – Padrões simulados para o NESISES

Variável S.S.	V-20	V-32	V-33	PV-15	V-53	AP-2	V-70
<i>Variável NESISES</i>	<i>V1</i>	<i>V2</i>	<i>V3</i>	<i>V4</i>	<i>V5</i>	<i>V6</i>	<i>V7</i>
Seqüência 1	1	1	0	1	1	1	1
Seqüência 2	0	1	1	0	1	1	1
Seqüência 3	1	0	1	0	0	1	1
Seqüência 4	0	0	1	0	0	0	1
Seqüência 5	1	1	0	1	0	0	1
Seqüência 6	0	1	1	1	1	0	0

Observa-se que os estados alteram-se deterministicamente segundo o programa ladder do emulador. Assim sendo, estes dados são de natureza do conhecimento explícito do programador. Portanto, neste caso sabe-se a priori quais são as ocorrências e decorrências previstas e programadas.

A Tabela 4.18 é uma sub-tabela da Tabela 4.17. A mesma apresenta claramente um padrão de comportamento entre as variáveis PV-15, V-33 e V-70 presentes nas seqüências 2, 3 e 4. Pode-se observar que existe uma regra de formação deste padrão. A regra de formação é:

Toda vez que a variável PV-15 for 0, as variáveis V-33 e V-70 serão 1.

Tabela 4.18 – Principal regra simulada para o NESISES

Variável S.S.	PV-15	V-33	V-70
<i>Variável NESISES</i>	<i>V4</i>	<i>V3</i>	<i>V7</i>
Seqüência 2	0	1	1
Seqüência 3	0	1	1
Seqüência 4	0	1	1

Os padrões mostrados na Tabela 4.17 foram simulados nos testes de validação do NESISES inicialmente durante 594 horas sendo que a cada 24h era gerado um novo arquivo pelo S.S., arquivo esse que ao final de 24h possuía aproximadamente 9474 linhas por 7 colunas, onde cada linha representa uma seqüência e cada coluna uma das tags.

A Tabela 4.19 mostra as 21 regras mais prováveis encontradas na base de dados gerada pela simulação. Nota-se na primeira coluna da tabela dois números, que representam respectivamente a probabilidade de confirmação de que a regra seja verdadeira e a freqüência de contra-exemplos encontrados, ou o grau de confirmação da regra e sua freqüência de contra-exemplos. Observa-se que no contexto deste trabalho, contra-exemplo significa um dado que contraria a regra encontrada.

Na Tabela 4.19 observa-se que as regras encontram-se dispostas em ordem decrescente de grau de confirmação. A análise dessa tabela mostra que o NESISES foi capaz de deduzir assertivamente a regra explicitada na Tabela 4.18.

A sintaxe das regras é ilustrada na Tabela 4.5

Tabela 4.19 – Regras encontradas pelo NESISES

Regra	Confirmação	Freqüência de contra-exemplos	Regra encontrada
1	/* 0.992298	0.003750 */	pv_15 = 0 :- v_33 = 1 and v_70 = 1.
	/* 0.992298	0.003750 */	v_33 = 0 or v_70 = 0 :- pv_15 = 1.
2	/* 0.878195	0.003750 */	v_33 = 0 :- pv_15 = 1 and v_70 = 1.
	/* 0.878195	0.003750 */	pv_15 = 0 or v_70 = 0 :- v_33 = 1.
3	/* 0.875885	0.003750 */	v_33 = 0 or pv_15 = 1 or v_53 = 1 :- v_32 = 1.
	/* 0.875885	0.003750 */	v_32 = 0 :- v_33 = 1 and pv_15 = 0 and v_53 = 0.
4	/* 0.875717	0.003864 */	v_32 = 0 :- pv_15 = 0 and v_53 = 0.
	/* 0.875717	0.003864 */	pv_15 = 1 or v_53 = 1 :- v_32 = 1.
5	/* 0.872376	0.006137 */	v_32 = 0 :- v_33 = 1 and v_53 = 0.
	/* 0.872376	0.006137 */	v_33 = 0 or v_53 = 1 :- v_32 = 1.
6	/* 0.691482	0.000114 */	ap_2 = 1 or v_70 = 0 :- v_53 = 1.
	/* 0.691482	0.000114 */	v_53 = 0 :- ap_2 = 0 and v_70 = 1.
7	/* 0.689886	0.000114 */	ap_2 = 1 :- v_53 = 1 and v_70 = 1.
	/* 0.689886	0.000114 */	v_53 = 0 or v_70 = 0 :- ap_2 = 0.
8	/* 0.688497	0.000000 */	v_32 = 1 or v_33 = 0 :- pv_15 = 1.
	/* 0.688497	0.000000 */	pv_15 = 0 :- v_32 = 0 and v_33 = 1.
9	/* 0.687955	0.000227 */	pv_15 = 0 :- v_32 = 0.
	/* 0.687955	0.000227 */	v_32 = 1 :- pv_15 = 1.
10	/* 0.687709	0.001364 */	v_33 = 0 or v_70 = 0 :- pv_15 = 1 and v_53 = 1.
	/* 0.687709	0.001364 */	pv_15 = 0 or v_53 = 0 :- v_33 = 1 and v_70 = 1.
11	/* 0.687639	0.000114 */	pv_15 = 1 :- v_33 = 0.
	/* 0.687639	0.000114 */	v_33 = 1 :- pv_15 = 0.

12	/* 0.685516	0.001250 */	$v_{33} = 0 \text{ or } ap_2 = 0 \text{ :- } pv_{15} = 1.$
	/* 0.685516	0.001250 */	$pv_{15} = 0 \text{ :- } v_{33} = 1 \text{ and } ap_2 = 1.$
13	/* 0.684612	0.002500 */	$v_{33} = 0 \text{ or } v_{70} = 0 \text{ :- } pv_{15} = 1 \text{ and } ap_2 = 0.$
	/* 0.684612	0.002500 */	$pv_{15} = 0 \text{ or } ap_2 = 1 \text{ :- } v_{33} = 1 \text{ and } v_{70} = 1.$
14	/* 0.684013	0.002387 */	$v_{33} = 0 \text{ or } v_{53} = 1 \text{ :- } pv_{15} = 1.$
	/* 0.684013	0.002387 */	$pv_{15} = 0 \text{ :- } v_{33} = 1 \text{ and } v_{53} = 0.$
15	/* 0.683008	0.000114 */	$v_{32} = 0 \text{ or } ap_2 = 0 \text{ :- } v_{53} = 0.$
	/* 0.683008	0.000114 */	$v_{53} = 1 \text{ :- } v_{32} = 1 \text{ and } ap_2 = 1.$
16	/* 0.682072	0.003637 */	$v_{33} = 0 \text{ or } pv_{15} = 1 \text{ :- } v_{32} = 1 \text{ and } ap_2 = 0.$
	/* 0.682072	0.003637 */	$v_{32} = 0 \text{ or } ap_2 = 1 \text{ :- } v_{33} = 1 \text{ and } pv_{15} = 0.$
17	/* 0.681738	0.003750 */	$v_{32} = 0 \text{ or } ap_2 = 1 \text{ :- } pv_{15} = 0.$
	/* 0.681738	0.003750 */	$pv_{15} = 1 \text{ :- } v_{32} = 1 \text{ and } ap_2 = 0.$
18	/* 0.680947	0.166837 */	$v_{32} = 0 \text{ :- } v_{33} = 1 \text{ and } pv_{15} = 0.$
	/* 0.680947	0.166837 */	$v_{33} = 0 \text{ or } pv_{15} = 1 \text{ :- } v_{32} = 1.$
19	/* 0.680835	0.168315 */	$pv_{15} = 0 \text{ :- } v_{33} = 1.$
	/* 0.680835	0.168315 */	$v_{33} = 0 \text{ :- } pv_{15} = 1.$
20	/* 0.680821	0.166951 */	$pv_{15} = 1 \text{ :- } v_{32} = 1.$
	/* 0.680821	0.166951 */	$v_{32} = 0 \text{ :- } pv_{15} = 0.$

A análise da Tabela 4.19 permite concluir que a primeira regra encontrada representa a regra explicitada na Tabela 4.18 com aproximadamente 99% de certeza.

As demais linhas da Tabela 4.19 representam as outras 20 regras mais prováveis. Observa-se, conforme já afirmado, que cada uma das regras apresentadas é única e portanto, não existem uma ou mais regras que contradigam qualquer regra.

Assim sendo, conclui-se que houve um aprendizado assertivo dos padrões de comportamento da planta simulada pelo NESISES.

As demais regras inferidas e apresentadas na Tabela 4.19 representam as outras regras presentes no sistema. Observa-se que cada uma delas possui um grau de confirmação estatística de ocorrência de determinada regra.

4.4 Análise dos resultados segundo os critérios de aprovação dos testes de simulação com o NESISES

Após serem executados os testes, os mesmos foram analisados segundo os critérios estabelecidos no item 4.3. Os resultados dessas análises foram os seguintes:

1. *Assertividade do aprendizado:* o NESISES mostrou possuir aprendizado assertivo (vide item 4.3.1).
2. *Desempenho do S.S.:* o S.S. não sofreu queda de desempenho apreciável.
3. *Tempo de execução:* O tempo para o NESISES aprender algo do S.S. foi desde alguns minutos até 36 horas, o que não foi impeditivo já que para grandes volumes de informação, um engenheiro do conhecimento poderia levar tempos muitos maiores para efetuar conclusões.
4. *Compatibilidade:* o NESISES se mostrou totalmente compatível com o sistema operacional (Windows), sendo que a única restrição é que a etapa de mineração de dados ocorra em um computador diferente daquele em que esteja funcionamento o S.S.

4.5 Testes de campo com o NESISES

Finalizada a etapa de simulação, foram executados os testes de campo, que como os testes de simulação foram elaborados conforme a IEEE 1059 – 1993 e seus resultados são apresentados neste capítulo.

Foi escolhido um sistema industrial intensamente automatizado para a execução dos testes de campo. Esse sistema industrial é detalhado a seguir.

Os critérios de validação para os testes de campo também são mostrados neste capítulo.

4.5.1 Objetivos dos testes de Campo com o NESISES

Os principais objetivos dos testes de campo foram a validação do NESISES em condições reais de funcionamento em um ambiente industrial, cujas características inerentes e dinâmicas freqüentemente extrapolam os limites que simulações conseguem aferir.

4.5.2 Descritivo do processo automatizado empregado para testes de campo do NESISES

O processo automatizado é um terminal portuário de armazenagem e expedição de sólidos a granel do terminal marítimo do Guarujá – SP, Porto de Santos. O mesmo possui características de complexidade que o tornam interessante para fins de testes de campo do NESISES. O processo estudado consiste principalmente em: três armazéns, um pátio de armazenagem, quatro tulhas

rodoviárias e duas tulhas ferroviárias, totalizando aproximadamente 6000 pontos de automação.

A automação realizada se caracterizou pela intensa utilização de redes de comunicação, sendo que a grande maioria dos pontos de entradas e saídas comunicam-se com os CLPs através de redes de comunicação de dados, notadamente as redes ControlNet, DeviceNet e Profibus.

Na Figura 35 está representada a arquitetura de Hardware do sistema de automação, sendo omitidos os motores que estão interligados aos CCMs (Centro de Controle de Motores), sendo que são 136 motores (divididos em 7 CCMs) acionados pelo sistema de automação.

Esse processo possui como principal divisão a recepção onde se dá o recebimento de materiais por meio de um terminal marítimo, ou simplificadaamente: os equipamentos necessários à atividade de descarga de navios com conseqüente armazenamento em armazéns ou pátio. A outra principal divisão é a expedição onde se dá o envio de material dos armazéns ou pátio para caminhões ou vagões. Essa divisão se reflete na arquitetura de hardware, sendo que tanto a recepção quanto a expedição possuem CLPs independentes.

Na Figura 36 podem-se observar esquematicamente os principais equipamentos do sistema. Do ponto de vista físico, o sistema portuário como um todo tem capacidade para descarga de 3 milhões de toneladas por ano de fertilizantes e enxofre, sendo que o pátio tem capacidade de armazenagem de 85 mil toneladas.

Figura 35 – Arquitetura de hardware do processo de armazenagem

Figura 36 – Arquitetura geral do processo de armazenagem automatizado

Na expedição, os equipamentos automatizados estão concentrados em estruturas chamadas tulhas onde caminhões ou vagões ferroviários se posicionam para conseqüente carregamento. Na Figura 37 podem-se observar os principais equipamentos de uma tulha rodoviária, note-se que os caminhões ficam sobre um balanço.

As tulhas ferroviárias são análogas às rodoviárias, sendo que nestas é efetuado o carregamento de vagões em vez de caminhões.

Figura 37 – Vista geral de tulha rodoviária automatizada

4.5.3 Aprendizado assertivo dos testes de campo com o NESISES

Nesta fase do trabalho, foi considerado aprendizado assertivo a capacidade do NESISES de sugerir regras relevantes para a operação do sistema automatizado.

Para tanto, foi necessário estabelecer previamente que todo sistema automatizado obedece a um algoritmo de automação explícito determinado de forma consciente ou não pelo engenheiro de automação que programa as lógicas de controle, discreto ou dinâmico nos CLPs, SDCDs, CAPs e Sistemas Supervisórios.

Este trabalho de pesquisa elegeu como campo prático de aplicação o processo de armazenagem e expedição de sólidos a granel do terminal marítimo do Guarujá – SP, conforme citado anteriormente.

O sistema deveria, como condição mínima, aprender as lógicas simples do processo, como por exemplo o fato de que, se a esteira denominada TC-105 estiver ativa, as TC-s (Esteiras) que a alimentam deverão estar ativas também, no caso as esteiras TC-104, TC-102 e TC-101 (vide Figura 36). Além de aprender as regras simples, o sistema deveria ser capaz de aprender regras não triviais do processo.

Com o objetivo de operacionalizar os testes, foram escolhidas como variáveis para aquisição pelo RSView o status de ligado dos principais equipamentos do sistema automatizado, totalizando 58 variáveis representadas na Tabela 4.20. Nota-se que as variáveis foram divididas de acordo com as sub-divisões do processo, ou seja: Armazém 2, Armazém 4, Armazém 6, Tulhas, Recepção e Pátio. Essa divisão espelha um conhecimento prévio do processo, o que indica uma certa independência das variáveis de determinada área.

Por exemplo, o armazém 2 pode operar sem que os outros armazéns operem, e isso é válido para todos os armazéns e o pátio. Já as tulhas vão operar a maior parte do tempo em conjunto com algum armazém ou ainda recebendo produto diretamente dos navios, o que é uma circunstância rara, mas possível.

Tabela 4.20 – Variáveis adquiridas pelo NESISES para análise dos dados de campo

Área/ Variável					
Armazém 2	Armazém 4	Armazém 6	Tulhas	Recepção	Pátio
Tc_201_1	AL_406_1	AL_601_1	AL_801_1	Tc_101_1	AL_701_1
Tc_202_1	AL_407_1	AL_602_1	AL_901_1	Tc_102_1	AL_702_1
Tc_203_1	AL_408_1	AL_603_1	AL_902_1	Tc_103_1	AL_703_1
Tc_204_1	AL_409_1	AL_604_1	Tc_801_1	Tc_104_1	AL_704_1
	AL_410_1	AL_605_1	Tc_802_1	Tc_105_1	AL_705_1
	AL_412_1	AL_606_1	Tc_803_1		AL_706_1
	Tc_401_1	AL_607_1	Tc_901_1		Tc_701_1
	Tc_405_1	AL_608_1	Tc_902_1		Tc_702_1
	Tc_406_1	AL_609_1	Tc_903_1		Tc_703_1
	Tc_407_1	AL_610_1			Tc_704_1
		AL_611_1			
		AL_612_1			
		Tc_601_1			
		Tc_602_1			
		Tc_603_1			
		Tc_604_1			
		Tc_605_1			
		Tc_606_1			
		Tc_607_1			
		AL_601_1			

4.5.3.1 Análise dos dados completos

Inicialmente o sistema foi testado escolhendo-se um intervalo de tempo em que, segundo os operadores do sistema, o processo funcionou praticamente sem interrupção. O intervalo escolhido foi de 01/05/2007 a 31/05/2007, sendo que foram gerados pelo RSView 31 arquivos tipo “.DBF”, um para cada dia de aquisição.

Para tornar a análise mais relevante foram realizados testes segmentados, primeiramente com a base de dados completa e posteriormente com variáveis escolhidas de áreas afins, pois havia o conhecimento prévio de que os armazéns trabalham independentemente entre si, mas dependentemente da recepção. Os cruzamentos entre variáveis podem ser observados na Tabela 4.21 .

Tabela 4.21 – Cruzamentos de variáveis utilizadas para busca de regras nos dados de campo

	Armazém 2	Armazém 4	Armazém 6	Pátio
Recepção	X	X	X	X
Tulha	X	X	X	X

Como teste preliminar, o NESISES rodou com todas as 58 variáveis ao mesmo tempo, e as regras encontradas são mostradas na Tabela 4.22. Analisando a regra 1, ou seja, se a esteira TC_104 estiver acionada, implica que as esteiras TC_102 **ou** TC-401 estão acionadas com 97% de certeza, observando-se ainda a Figura 36. Nota-se que as duas primeiras regras relacionam esteiras da recepção (TC-102 e TC-104) com a esteira principal do armazém 4. Ainda da análise da Figura 36 percebe-se que as esteiras TC-102 e TC-104 funcionam em conjunto, sendo que a TC-102 alimenta a TC-104, mas em princípio são independentes da esteira TC-401 que é a esteira que alimenta o armazém 4.

Ao fazer o cruzamento dessas informações com os operadores do sistema de automação, os mesmos informaram que no mês de maio de 2007 por contingências operacionais na maior parte do tempo houve a utilização do armazém 4; daí as regras mais prováveis (regras 1 e 2) relacionarem a recepção com o armazém 4.

Nota-se ainda que as regras 3, 4, 5 e 6 relacionam as esteiras da recepção, - TC-101, TC-102, TC-104 e TC-105, sendo que essas esteiras trabalham em conjunto.

Conclui-se assim que o NESISES foi capaz de aprender regras coerentes do sistema real.

Tabela 4.22 – Regras encontradas pelo NESISES para todas as variáveis do sistema

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.976151	0.004942 */	TC_102_L(true) OR TC_401_L(true) -> TC_104_L(true)
	/* 0.976151	0.004942 */	TC_104_L(false) -> TC_102_L(false) AND TC_401_L(false)
2	/* 0.976029	0.000000 */	TC_104_L(false) -> TC_105_L(false) AND TC_401_L(false)
	/* 0.976029	0.000000 */	TC_105_L(true) OR TC_401_L(true) -> TC_104_L(true)
3	/* 0.969299	0.010119 */	TC_102_L(true) OR TC_105_L(true) -> TC_104_L(true)
	/* 0.969299	0.010119 */	TC_104_L(false) -> TC_102_L(false) AND TC_105_L(false)
4	/* 0.969140	0.000052 */	TC_104_L(true) -> TC_102_L(true)
	/* 0.969140	0.000052 */	TC_102_L(false) -> TC_104_L(false)
5	/* 0.969128	0.015402 */	TC_104_L(false) -> TC_102_L(false)
	/* 0.969128	0.015402 */	TC_102_L(true) -> TC_104_L(true)
6	/* 0.967766	0.000000 */	TC_101_L(false) OR TC_104_L(false) -> TC_102_L(false)
	/* 0.967766	0.000000 */	TC_102_L(true) -> TC_101_L(true) AND TC_104_L(true)
7	/* 0.967711	0.000078 */	TC_104_L(false) -> TC_401_L(false) AND TC_601_L(false)
	/* 0.967711	0.000078 */	TC_401_L(true) OR TC_601_L(true) -> TC_104_L(true)
8	/* 0.963946	0.015009 */	TC_102_L(true) -> TC_103_L(false) AND TC_104_L(true)
	/* 0.963946	0.015009 */	TC_103_L(true) OR TC_104_L(false) -> TC_102_L(false)
9	/* 0.962606	0.000078 */	TC_101_L(false) OR TC_103_L(true) -> TC_102_L(false)
	/* 0.962606	0.000078 */	TC_102_L(true) -> TC_101_L(true) AND TC_103_L(false)
10	/* 0.961109	0.010067 */	TC_102_L(true) OR TC_601_L(true) -> TC_104_L(true)
	/* 0.961109	0.010067 */	TC_104_L(false) -> TC_102_L(false) AND TC_601_L(false)

4.5.3.2 Análise dos dados Recepção / Armazém 2

Analisando as regras explicitadas na Tabela 4.23 nota-se que nenhuma delas utiliza as variáveis do armazém 2 (Tc_201_I, Tc_202_I, Tc_203_I, Tc_204_I), todas as regras encontradas são na verdade fatos sobre os equipamentos da recepção.

Por exemplo, a regra 1 mostra com 97% de certeza que se a esteira TC-104 estiver ligada, as esteiras TC-102 ou TC-105 também estarão, o que é verdadeiro, já que essas esteiras trabalham em conjunto (vide Figura 36).

Essas regras são coerentes, pois os equipamentos do armazém 2 estiveram desligados por contingências operacionais durante todo o período de análise, enquanto os equipamentos da recepção estiveram funcionando normalmente.

Tabela 4.23 – Regras encontradas pelo NESISES para as variáveis Armazém -2**/ Recepção**

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.969299	0.010119 */	TC_102_L(true) OR TC_105_L(true) -> TC_104_L(true)
	/* 0.969299	0.010119 */	TC_104_L(false) -> TC_102_L(false) AND TC_105_L(false)
2	/* 0.969140	0.000052 */	TC_104_L(true) -> TC_102_L(true)
	/* 0.969140	0.000052 */	TC_102_L(false) -> TC_104_L(false)
3	/* 0.969128	0.015402 */	TC_104_L(false) -> TC_102_L(false)
	/* 0.969128	0.015402 */	TC_102_L(true) -> TC_104_L(true)
4	/* 0.967766	0.000000 */	TC_101_L(false) OR TC_104_L(false) -> TC_102_L(false)
	/* 0.967766	0.000000 */	TC_102_L(true) -> TC_101_L(true) AND TC_104_L(true)
5	/* 0.963946	0.015009 */	TC_102_L(true) -> TC_103_L(false) AND TC_104_L(true)
	/* 0.963946	0.015009 */	TC_103_L(true) OR TC_104_L(false) -> TC_102_L(false)
6	/* 0.962606	0.000078 */	TC_101_L(false) OR TC_103_L(true) -> TC_102_L(false)
	/* 0.962606	0.000078 */	TC_102_L(true) -> TC_101_L(true) AND TC_103_L(false)
7	/* 0.934579	0.000131 */	TC_101_L(false) OR TC_103_L(true) -> TC_104_L(false)
	/* 0.934579	0.000131 */	TC_104_L(true) -> TC_101_L(true) AND TC_103_L(false)
8	/* 0.785298	0.000078 */	TC_101_L(false) OR TC_102_L(true) -> TC_103_L(false)
	/* 0.785298	0.000078 */	TC_103_L(true) -> TC_101_L(true) AND TC_102_L(false)
9	/* 0.785222	0.000131 */	TC_101_L(false) OR TC_104_L(true) -> TC_103_L(false)
	/* 0.785222	0.000131 */	TC_103_L(true) -> TC_101_L(true) AND TC_104_L(false)
10	/* 0.724620	0.000078 */	TC_101_L(false) -> TC_102_L(false) AND TC_103_L(false)
	/* 0.724620	0.000078 */	TC_102_L(true) OR TC_103_L(true) -> TC_101_L(true)

4.5.3.3 Análise dos dados Recepção / Armazém 4

Analisando as regras explicitadas na Tabela 4.24, nota-se que as três regras mais prováveis (1, 2 e 3) relacionam as esteiras da recepção com a principal esteira do armazém 4 (TC-401).

Analisando a regra 1, ou seja, se a esteira TC_104 estiver acionada, implica que as esteiras TC_102 ou TC-401 estão acionadas com 97% de certeza, observando-se ainda a Figura 36. Nota-se que as duas primeiras regras relacionam esteiras da recepção(TC-102 e TC-104) com a esteira principal(alimentadora) do armazém 4. Ainda da análise da Figura 36 percebe-se que as esteiras TC-102 e TC-104 funcionam em conjunto, sendo que a TC-102 alimenta a TC-104.

Essas regras são coerentes pois existe uma relação de dependência, por exemplo a regra 1 mostra que as esteiras TC-102, 401 e 104 funcionam em conjunto, o que corresponde aos fatos operacionais da recepção trabalhando em conjunto com o armazém 4 (vide Figura 36).

Tabela 4.24 – Regras encontradas pelo NESISES para as variáveis Armazém-4 /

Recepção

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.976151	0.004942 */	TC_102_L(true) OR TC_401_L(true) -> TC_104_L(true)
	/* 0.976151	0.004942 */	TC_104_L(false) -> TC_102_L(false) AND TC_401_L(false)
2	/* 0.976029	0.000000 */	TC_105_L(true) OR TC_401_L(true) -> TC_104_L(true)
	/* 0.976029	0.000000 */	TC_104_L(false) -> TC_105_L(false) AND TC_401_L(false)
3	/* 0.971303	0.004550 */	TC_103_L(true) OR TC_104_L(false) -> TC_102_L(false) AND TC_401_L(false)
	/* 0.971303	0.004550 */	TC_102_L(true) OR TC_401_L(true) -> TC_103_L(false) AND TC_104_L(true)
4	/* 0.969299	0.010119 */	TC_102_L(true) OR TC_105_L(true) -> TC_104_L(true)
	/* 0.969299	0.010119 */	TC_104_L(false) -> TC_102_L(false) AND TC_105_L(false)
5	/* 0.969140	0.000052 */	TC_104_L(true) -> TC_102_L(true)
	/* 0.969140	0.000052 */	TC_102_L(false) -> TC_104_L(false)
6	/* 0.969128	0.015402 */	TC_104_L(false) -> TC_102_L(false)
	/* 0.969128	0.015402 */	TC_102_L(true) -> TC_104_L(true)
7	/* 0.967766	0.000000 */	TC_101_L(false) OR TC_104_L(false) -> TC_102_L(false)
	/* 0.967766	0.000000 */	TC_102_L(true) -> TC_101_L(true) AND TC_104_L(true)
8	/* 0.963946	0.015009 */	TC_102_L(true) -> TC_103_L(false) AND TC_104_L(true)
	/* 0.963946	0.015009 */	TC_103_L(true) OR TC_104_L(false) -> TC_102_L(false)
9	/* 0.962606	0.000078 */	TC_101_L(false) OR TC_103_L(true) -> TC_102_L(false)
	/* 0.962606	0.000078 */	TC_102_L(true) -> TC_101_L(true) AND TC_103_L(false)
10	/* 0.956078	0.000026 */	TC_102_L(false) -> TC_104_L(false) AND TC_401_L(false)
	/* 0.956078	0.000026 */	TC_104_L(true) OR TC_401_L(true) -> TC_102_L(true)

4.5.3.4 Análise dos dados Recepção / Armazém 6

Analisando as regras explicitadas na Tabela 4.25 nota-se que as seis regras mais prováveis (1, 2, 3, 4, 5 e 6) relacionam as esteiras da recepção de maneira encadeada, sendo esse o arranjo necessário para alimentar a principal esteira do armazém 6 (TC-601). A partir da sétima regra a TC-601 é relacionada com as esteiras da recepção.

Essas regras são coerentes pois existe uma relação de dependência, por exemplo, a regra 1 mostra que as esteiras TC-102, 104 e 105 funcionam em conjunto, o que é necessário para alimentar o armazém 6 (vide Figura 36).

Tabela 4.25 – Regras encontradas pelo NESISES para as variáveis Armazém -6 / Recepção

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	* 0.969299	/0.010119 */	TC_102_L(true) OR TC_105_L(true) -> TC_104_L(true)
	/* 0.969299	0.010119 */	TC_104_L(false) -> TC_102_L(false) AND TC_105_L(false)
2	/* 0.969140	0.000052 */	TC_104_L(true) -> TC_102_L(true)
	/* 0.969140	0.000052 */	TC_102_L(false) -> TC_104_L(false)
3	/* 0.969128	0.015402 */	TC_104_L(false) -> TC_102_L(false)
	/* 0.969128	0.015402 */	TC_102_L(true) -> TC_104_L(true)
4	/* 0.967766	0.000000 */	TC_101_L(false) OR TC_104_L(false) -> TC_102_L(false)
	/* 0.967766	0.000000 */	TC_102_L(true) -> TC_101_L(true) AND TC_104_L(true)
5	/* 0.963946	0.015009 */	TC_102_L(true) -> TC_103_L(false) AND TC_104_L(true)
	/* 0.963946	0.015009 */	TC_103_L(true) OR TC_104_L(false) -> TC_102_L(false)
6	/* 0.962606	0.000078 */	TC_101_L(false) OR TC_103_L(true) -> TC_102_L(false)
	/* 0.962606	0.000078 */	TC_102_L(true) -> TC_101_L(true) AND TC_103_L(false)
7	/* 0.961109	0.010067 */	TC_102_L(true) OR TC_601_L(true) -> TC_104_L(true)
	/* 0.961109	0.010067 */	TC_104_L(false) -> TC_102_L(false) AND TC_601_L(false)
8	/* 0.960639	0.000758 */	TC_606_L(false) -> TC_605_L(false) AND TC_607_L(false)
	/* 0.960639	0.000758 */	TC_605_L(true) OR TC_607_L(true) -> TC_606_L(true)
9	/* 0.959683	0.000732 */	TC_606_L(false) -> AL_610_L(false) AND TC_605_L(false) AND TC_607_L(false)
	/* 0.959683	0.000732 */	AL_610_L(true) OR TC_605_L(true) OR TC_607_L(true) -> TC_606_L(true)
10	/* 0.959123	0.004105 */	AL_602_L(true) OR TC_602_L(true) OR TC_604_L(true) -> TC_603_L(true)
	/* 0.959123	0.004105 */	TC_603_L(false) -> AL_602_L(false) AND TC_602_L(false) AND TC_604_L(false)

4.5.3.5 Análise dos dados Recepção / Pátio

Analisando as regras explicitadas na Tabela 4.26 nota-se que as seis regras mais prováveis (1, 2, 3, 4, 5 e 6) relacionam as esteiras da recepção, TC-101, TC102, TC103, TC-104 e TC-105 de maneira encadeada, sendo esse o arranjo necessário para alimentar a principal esteira do pátio (TC-701). A partir da sétima regra a TC-701 é relacionada com as esteiras da recepção.

Essas regras são coerentes pois existe uma relação de dependência, por exemplo a regra 1 mostra que as esteiras TC-102, 104 e 105 funcionam em conjunto, o que corresponde aos fatos operacionais da recepção trabalhando em conjunto com o armazém (vide Figura 36).

Tabela 4.26 – Regras encontradas pelo NESISES para as variáveis Pátio /

Recepção

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.969299	0.010119 */	TC_102_L(true) OR TC_105_L(true) -> TC_104_L(true)
	/* 0.969299	0.010119 */	TC_104_L(false) -> TC_102_L(false) AND TC_105_L(false)
2	/* 0.969140	0.000052 */	TC_104_L(true) -> TC_102_L(true)
	/* 0.969140	0.000052 */	TC_102_L(false) -> TC_104_L(false)
3	/* 0.969128	0.015402 */	TC_104_L(false) -> TC_102_L(false)
	/* 0.969128	0.015402 */	TC_102_L(true) -> TC_104_L(true)
4	/* 0.967766	0.000000 */	TC_102_L(true) -> TC_101_L(true) AND TC_104_L(true)
	/* 0.967766	0.000000 */	TC_101_L(false) OR TC_104_L(false) -> TC_102_L(false)
5	/* 0.963946	0.015009 */	TC_102_L(true) -> TC_103_L(false) AND TC_104_L(true)
	/* 0.963946	0.015009 */	TC_103_L(true) OR TC_104_L(false) -> TC_102_L(false)
6	/* 0.962606	0.000078 */	TC_102_L(true) -> TC_101_L(true) AND TC_103_L(false)
	/* 0.962606	0.000078 */	TC_101_L(false) OR TC_103_L(true) -> TC_102_L(false)
7	/* 0.955252	0.015009 */	TC_102_L(true) -> TC_104_L(true) AND TC_701_L(false)
	/* 0.955252	0.015009 */	TC_104_L(false) OR TC_701_L(true) -> TC_102_L(false)
8	/* 0.954115	0.000105 */	TC_102_L(true) -> TC_101_L(true) AND TC_701_L(false)
	/* 0.954115	0.000105 */	TC_101_L(false) OR TC_701_L(true) -> TC_102_L(false)
9	/* 0.934579	0.000131 */	TC_104_L(true) -> TC_101_L(true) AND TC_103_L(false)
	/* 0.934579	0.000131 */	TC_101_L(false) OR TC_103_L(true) -> TC_104_L(false)
10	/* 0.926444	0.000157 */	TC_104_L(true) -> TC_101_L(true) AND TC_701_L(false)
	/* 0.926444	0.000157 */	TC_101_L(false) OR TC_701_L(true) -> TC_104_L(false)

4.5.3.6 Análise dos dados Pátio / Tulha

Analisando as regras explicitadas na Tabela 4.27 nota-se que as três regras mais prováveis (1, 2 e 3) relacionam os alimentadores do pátio (AL-702, AL-703 e AL-704) com as duas esteiras (TC-702 e 703) que são alimentadas por esses alimentadores. As demais regras explicitam os diversos encadeamentos do processo.

Essas regras são coerentes pois existe uma relação de dependência, por exemplo, a regra 1 mostra que os alimentadores AL-702 e 703 funcionam em

conjunto com as esteiras TC-702 e 703, o que corresponde aos fatos operacionais (vide Figura 36).

Tabela 4.27 – Regras encontradas pelo NESISES para as variáveis Pátio / Tulha

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.901795	0.032450 */	AL_702_L(true) OR AL_703_L(true) OR TC_702_L(true) -> TC_703_L(true)
	/* 0.901795	0.032450 */	TC_703_L(false) -> AL_702_L(false) AND AL_703_L(false) AND TC_702_L(false)
2	/* 0.900195	0.032320 */	TC_703_L(false) OR TC_704_L(false) -> AL_702_L(false) AND AL_703_L(false) AND TC_702_L(false)
	/* 0.900195	0.032320 */	AL_702_L(true) OR AL_703_L(true) OR TC_702_L(true) -> TC_703_L(true) AND TC_704_L(true)
3	/* 0.899262	0.032424 */	TC_703_L(false) -> AL_702_L(false) AND AL_703_L(false) AND AL_704_L(false) AND TC_702_L(false)
	/* 0.899262	0.032424 */	AL_702_L(true) OR AL_703_L(true) OR AL_704_L(true) OR TC_702_L(true) -> TC_703_L(true)
4	/* 0.887019	0.009597 */	TC_902_L(true) -> TC_903_L(true)
	/* 0.887019	0.009597 */	TC_903_L(false) -> TC_902_L(false)
5	/* 0.886876	0.009570 */	TC_901_L(true) OR TC_902_L(true) -> TC_903_L(true)
	/* 0.886876	0.009570 */	TC_903_L(false) -> TC_901_L(false) AND TC_902_L(false)
6	/* 0.885139	0.009518 */	TC_903_L(false) -> AL_901_L(false) AND TC_902_L(false)
	/* 0.885139	0.009518 */	AL_901_L(true) OR TC_902_L(true) -> TC_903_L(true)
7	/* 0.885031	0.009492 */	TC_903_L(false) -> AL_901_L(false) AND TC_901_L(false) AND TC_902_L(false)
	/* 0.885031	0.009492 */	AL_901_L(true) OR TC_901_L(true) OR TC_902_L(true) -> TC_903_L(true)
8	/* 0.881448	0.001203 */	TC_704_L(true) -> TC_703_L(true)
	/* 0.881448	0.001203 */	TC_703_L(false) -> TC_704_L(false)
9	/* 0.881228	0.007322 */	TC_901_L(true) -> TC_902_L(true)
	/* 0.881228	0.007322 */	TC_902_L(false) -> TC_901_L(false)
10	/* 0.880998	0.000366 */	TC_702_L(true) OR TC_704_L(true) -> TC_703_L(true)
	/* 0.880998	0.000366 */	TC_703_L(false) -> TC_702_L(false) AND TC_704_L(false)

4.5.3.7 Análise dos dados Tulha / Armazém 2

Analisando as regras explicitadas na Tabela 4.28 nota-se que nenhuma delas utiliza as variáveis do armazém 2 (Tc_201_I, Tc_202_I, Tc_203_I, Tc_204_I), todas as regras encontradas são na verdade fatos sobre os equipamentos da tulha.

Essas regras são coerentes pois os equipamentos do armazém 2 estiveram desligados por contingências operacionais durante todo o período de análise, enquanto os equipamentos da tulha estiveram funcionando normalmente. Analisando as regras pode-se inferir a funcionalidade dos equipamentos, como por

exemplo, pela regra 1 nota-se que as esteiras TC902 e TC903 trabalham em conjunto, o que corresponde à realidade operacional.

Tabela 4.28 – Regras encontradas pelo NESISES para as variáveis Tulha / Armazém 2

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.888766	0.000026 */	TC_902_L(false) -> TC_903_L(false)
	/* 0.888766	0.000026 */	TC_903_L(true) -> TC_902_L(true)
2	/* 0.888634	0.000000 */	TC_901_L(true) OR TC_903_L(true) -> TC_902_L(true)
	/* 0.888634	0.000000 */	TC_902_L(false) -> TC_901_L(false) AND TC_903_L(false)
3	/* 0.887019	0.009597 */	TC_902_L(true) -> TC_903_L(true)
	/* 0.887019	0.009597 */	TC_903_L(false) -> TC_902_L(false)
4	/* 0.886876	0.009570 */	TC_903_L(false) -> TC_901_L(false) AND TC_902_L(false)
	/* 0.886876	0.009570 */	TC_901_L(true) OR TC_902_L(true) -> TC_903_L(true)
5	/* 0.885139	0.009518 */	TC_903_L(false) -> AL_901_L(false) AND TC_902_L(false)
	/* 0.885139	0.009518 */	AL_901_L(true) OR TC_902_L(true) -> TC_903_L(true)
6	/* 0.885031	0.009492 */	TC_903_L(false) -> AL_901_L(false) AND TC_901_L(false) AND TC_902_L(false)
	/* 0.885031	0.009492 */	AL_901_L(true) OR TC_901_L(true) OR TC_902_L(true) -> TC_903_L(true)
7	/* 0.881228	0.007322 */	TC_902_L(false) -> TC_901_L(false)
	/* 0.881228	0.007322 */	TC_901_L(true) -> TC_902_L(true)
8	/* 0.879284	0.007269 */	TC_902_L(false) -> AL_901_L(false) AND TC_901_L(false)
	/* 0.879284	0.007269 */	AL_901_L(true) OR TC_901_L(true) -> TC_902_L(true)
9	/* 0.867492	0.016892 */	TC_903_L(false) -> TC_901_L(false)
	/* 0.867492	0.016892 */	TC_901_L(true) -> TC_903_L(true)
10	/* 0.865774	0.016761 */	TC_903_L(false) -> AL_901_L(false) AND TC_901_L(false)
	/* 0.865774	0.016761 */	AL_901_L(true) OR TC_901_L(true) -> TC_903_L(true)

4.5.3.8 Análise dos dados Tulha / Armazém 4

Analisando as regras explicitadas na Tabela 4.29 nota-se que as três regras mais prováveis (1, 2 e 3) relacionam os alimentadores do armazém (AL-406, AL-407 e AL-408) com as duas esteiras(TC-405 e 406) que são alimentadas por esses alimentadores. As demais regras explicitam os diversos encadeamentos do processo.

Essas regras são coerentes pois existe uma relação de dependência, por exemplo a regra 1 mostra que os alimentadores AL-408 e 406 funcionam em conjunto com as esteiras TC-406 e 405, o que corresponde aos fatos operacionais (vide Figura 36). Nota-se ainda que para as regras 7 a 12 são explicitadas relações entre os equipamentos das tulhas e a partir da regra 13 são explicitadas relações entre os equipamentos do armazém e os da tulha.

Tabela 4.29 – Regras encontradas pelo NESISES para as variáveis Tulha / Armazém 4

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.938296	0.023900 */	AL_408_L(true) OR TC_406_L(false) -> AL_406_L(false) AND AL_407_L(false) AND TC_405_L(false)
	/* 0.938296	0.023900 */	AL_406_L(true) OR AL_407_L(true) OR TC_405_L(true) -> AL_408_L(false) AND TC_406_L(true)
2	/* 0.928657	0.023900 */	AL_406_L(true) OR AL_407_L(true) OR AL_408_L(true) OR TC_405_L(true) -> TC_406_L(true)
	/* 0.928657	0.023900 */	TC_406_L(false) -> AL_406_L(false) AND AL_407_L(false) AND AL_408_L(false) AND TC_405_L(false)
3	/* 0.913777	0.003948 */	TC_406_L(false) -> AL_407_L(false) AND AL_408_L(false) AND TC_405_L(false) AND TC_407_L(false)
	/* 0.913777	0.003948 */	AL_407_L(true) OR AL_408_L(true) OR TC_405_L(true) OR TC_407_L(true) -> TC_406_L(true)
4	/* 0.897430	0.014329 */	AL_408_L(true) OR TC_405_L(true) OR TC_407_L(true) -> TC_406_L(true)
	/* 0.897430	0.014329 */	TC_406_L(false) -> AL_408_L(false) AND TC_405_L(false) AND TC_407_L(false)
5	/* 0.894948	0.014173 */	AL_407_L(true) OR AL_408_L(true) OR AL_410_L(true) OR TC_407_L(true) -> TC_406_L(true)
	/* 0.894948	0.014173 */	TC_406_L(false) -> AL_407_L(false) AND AL_408_L(false) AND AL_410_L(false) AND TC_407_L(false)
6	/* 0.893798	0.015402 */	AL_408_L(true) OR AL_409_L(true) OR AL_410_L(true) OR TC_407_L(true) -> TC_406_L(true)
	/* 0.893798	0.015402 */	TC_406_L(false) -> AL_408_L(false) AND AL_409_L(false) AND AL_410_L(false) AND TC_407_L(false)
7	/* 0.887019	0.009597 */	TC_902_L(true) -> TC_903_L(true)
	/* 0.887019	0.009597 */	TC_903_L(false) -> TC_902_L(false)
8	/* 0.886876	0.009570 */	TC_901_L(true) OR TC_902_L(true) -> TC_903_L(true)
	/* 0.886876	0.009570 */	TC_903_L(false) -> TC_901_L(false) AND TC_902_L(false)
9	/* 0.885139	0.009518 */	TC_903_L(false) -> AL_901_L(false) AND TC_902_L(false)
	/* 0.885139	0.009518 */	AL_901_L(true) OR TC_902_L(true) -> TC_903_L(true)
10	/* 0.885031	0.009492 */	TC_903_L(false) -> AL_901_L(false) AND TC_901_L(false) AND TC_902_L(false)

	/* 0.885031	0.009492 */	AL_901_L(true) OR TC_901_L(true) OR TC_902_L(true) -> TC_903_L(true)
11	/* 0.881228	0.007322 */	TC_901_L(true) -> TC_902_L(true)
	/* 0.881228	0.007322 */	TC_902_L(false) -> TC_901_L(false)
12	/* 0.879284	0.007269 */	TC_902_L(false) -> AL_901_L(false) AND TC_901_L(false)
	/* 0.879284	0.007269 */	AL_901_L(true) OR TC_901_L(true) -> TC_902_L(true)
13			AL_408_L(true) OR AL_410_L(true) OR TC_407_L(true) -> TC_406_L(true)
	/* 0.878906	0.024554 */	TC_406_L(false) -> AL_408_L(false) AND AL_410_L(false) AND TC_407_L(false)
	/* 0.878906	0.024554 */	TC_406_L(false) -> AL_408_L(false) AND AL_410_L(false) AND TC_407_L(false)
14	/* 0.877627	0.025599 */	AL_407_L(true) OR TC_405_L(true) OR TC_407_L(true) -> TC_406_L(true)
	/* 0.877627	0.025599 */	TC_406_L(false) -> AL_407_L(false) AND TC_405_L(false) AND TC_407_L(false)
15	/* 0.877005	0.025573 */	AL_407_L(true) OR AL_409_L(true) OR TC_405_L(true) OR TC_407_L(true) -> TC_406_L(true)
	/* 0.877005	0.025573 */	TC_406_L(false) -> AL_407_L(false) AND AL_409_L(false) AND TC_405_L(false) AND TC_407_L(false)
16	/* 0.875676	0.009544 */	AL_408_L(true) OR TC_903_L(false) -> TC_902_L(false)
	/* 0.875676	0.009544 */	TC_902_L(true) -> AL_408_L(false) AND TC_903_L(true)
17	/* 0.875537	0.009518 */	TC_901_L(true) OR TC_902_L(true) -> AL_408_L(false) AND TC_903_L(true)
	/* 0.875537	0.009518 */	AL_408_L(true) OR TC_903_L(false) -> TC_901_L(false) AND TC_902_L(false)

4.5.3.9 Análise dos dados Tulha / Armazém 6

Observando-se a Figura 36, nota-se que o armazém 6 possui um número de equipamentos maior que os armazéns 2 e 4, isso se deve ao fato do armazém 6 poder ser descarregado por dois lados enquanto os armazéns 2 e 4 são descarregados por somente um lado. Devido a essa maior complexidade, para o armazém 6 foram mineradas as 20 regras mais prováveis de acontecer.

Analisando as regras explicitadas na Tabela 4.30 nota-se que as regras traduzem as diversas relações de dependência entre as esteiras do lado direito, que são TC-605, TC-606, TC607 e seus respectivos alimentadores, que são AL-606, AL-607, AL-605, AL-608, AL-609 e AL-610 o que pode ser inferido nas regras 1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13.

Essas regras são coerentes com os fatos operacionais, sendo que o maior número de regras se referindo ao lado direito reflete a maior utilização deste lado do armazém no período analisado.

Tabela 4.30 – Regras encontradas pelo NESISES para as variáveis Tulha /**Armazém 6**

Regra	Confirmação	Frequência de contra-exemplos	Regra encontrada
1	/* 0.960639	0.000758 */	TC_606_L(false) -> TC_605_L(false) AND TC_607_L(false)
	/* 0.960639	0.000758 */	TC_605_L(true) OR TC_607_L(true) -> TC_606_L(true)
2	/* 0.959683	0.000732 */	TC_606_L(false) -> AL_610_L(false) AND TC_605_L(false) AND TC_607_L(false)
	/* 0.959683	0.000732 */	AL_610_L(true) OR TC_605_L(true) OR TC_607_L(true) -> TC_606_L(true)
3	/* 0.959123	0.004105 */	AL_602_L(true) OR TC_602_L(true) OR TC_604_L(true) -> TC_603_L(true)
	/* 0.959123	0.004105 */	TC_603_L(false) -> AL_602_L(false) AND TC_602_L(false) AND TC_604_L(false)
4	/* 0.956103	0.020161 */	TC_603_L(false) -> AL_602_L(false) AND AL_603_L(false) AND TC_602_L(false)
	/* 0.956103	0.020161 */	AL_602_L(true) OR AL_603_L(true) OR TC_602_L(true) -> TC_603_L(true)
5	/* 0.954902	0.002693 */	AL_609_L(true) OR AL_610_L(true) OR TC_607_L(true) -> TC_606_L(true)
	/* 0.954902	0.002693 */	TC_606_L(false) -> AL_609_L(false) AND AL_610_L(false) AND TC_607_L(false)
6	/* 0.947138	0.007949 */	TC_606_L(false) -> AL_610_L(false) AND TC_607_L(false)
	/* 0.947138	0.007949 */	AL_610_L(true) OR TC_607_L(true) -> TC_606_L(true)
7	/* 0.946779	0.007505 */	TC_606_L(false) -> AL_608_L(false) AND AL_610_L(false) AND TC_607_L(false)
	/* 0.946779	0.007505 */	AL_608_L(true) OR AL_610_L(true) OR TC_607_L(true) -> TC_606_L(true)
8	/* 0.935364	0.000575 */	TC_606_L(false) -> AL_605_L(false) AND TC_605_L(false) AND TC_607_L(false)
	/* 0.935364	0.000575 */	AL_605_L(true) OR TC_605_L(true) OR TC_607_L(true) -> TC_606_L(true)
9	/* 0.930851	0.026489 */	AL_607_L(true) OR AL_608_L(true) OR TC_605_L(true) -> TC_606_L(true)
	/* 0.930851	0.026489 */	TC_606_L(false) -> AL_607_L(false) AND AL_608_L(false) AND TC_605_L(false)
10	/* 0.923539	0.006929 */	TC_606_L(false) -> AL_605_L(false) AND AL_610_L(false) AND TC_607_L(false)
	/* 0.923539	0.006929 */	AL_605_L(true) OR AL_610_L(true) OR TC_607_L(true) -> TC_606_L(true)
11	/* 0.923466	0.021573 */	AL_609_L(true) OR TC_607_L(true) -> TC_606_L(true)
	/* 0.923466	0.021573 */	TC_606_L(false) -> AL_609_L(false) AND TC_607_L(false)
12	/* 0.922809	0.021285 */	TC_606_L(false) -> AL_608_L(false) AND AL_609_L(false) AND TC_607_L(false)
	/* 0.922809	0.021285 */	AL_608_L(true) OR AL_609_L(true) OR TC_607_L(true) -> TC_606_L(true)
13	/* 0.922710	0.026489 */	AL_608_L(true) OR TC_606_L(false) -> AL_607_L(false) AND TC_605_L(false)
	/* 0.922710	0.026489 */	AL_607_L(true) OR TC_605_L(true) -> AL_608_L(false) AND TC_606_L(true)
14	/* 0.922099	0.000628 */	AL_604_L(true) OR TC_605_L(true) OR TC_607_L(true) -> TC_606_L(true)
	/* 0.922099	0.000628 */	TC_606_L(false) -> AL_604_L(false) AND TC_605_L(false) AND TC_607_L(false)

15	/* 0.920819	0.023194 */	AL_602_L(true) OR AL_605_L(true) OR TC_604_L(true) -> TC_603_L(true)
	/* 0.920819	0.023194 */	TC_603_L(false) -> AL_602_L(false) AND AL_605_L(false) AND TC_604_L(false)
16	/* 0.917472	0.002693 */	AL_604_L(true) OR AL_610_L(true) OR TC_607_L(true) -> TC_606_L(true)
	/* 0.917472	0.002693 */	TC_606_L(false) -> AL_604_L(false) AND AL_610_L(false) AND TC_607_L(false)
17	/* 0.915774	0.007217 */	TC_606_L(true) -> TC_607_L(true)
	/* 0.915774	0.007217 */	TC_607_L(false) -> TC_606_L(false)
18	/* 0.915766	0.007191 */	TC_607_L(false) -> TC_605_L(false) AND TC_606_L(false)
	/* 0.915766	0.007191 */	TC_605_L(true) OR TC_606_L(true) -> TC_607_L(true)
19	/* 0.915167	0.027351 */	TC_606_L(false) -> TC_607_L(false)
	/* 0.915167	0.027351 */	TC_607_L(true) -> TC_606_L(true)
20	/* 0.915008	0.007112 */	AL_607_L(true) OR TC_606_L(true) -> TC_607_L(true)
	/* 0.915008	0.007112 */	TC_607_L(false) -> AL_607_L(false) AND TC_606_L(false)

4.5.4 Análise de rastreabilidade dos testes de campo

Os testes de campo se pautaram pelas restrições impostas por um processo real que, como premissa para autorização de ser utilizado como campo de testes, foi estipulado que de nenhuma forma o sistema poderia ser prejudicado. O procedimento executado para garantir a rastreabilidade foi:

a) Foram arbitrados os eventos que seriam configurados na aplicação do S.S. da planta industrial para serem aquisitados, esses foram anotados e implementados.

b) A aplicação desenvolvida na planta industrial elaborada no software de criação de S.S.s Rsview (Rockwell Software,2007), foi configurada de forma a aquisitar e armazenar automaticamente todos os eventos, definidos anteriormente, que ocorressem, e esses eventos (base de dados) foram armazenados em arquivos de dados do tipo .DBF sendo que esses arquivos seguem a seguinte forma:

b.1) Dois arquivos para cada dia de aquisição de dados, por exemplo: "2007 05 11 0000 (Tagname).DBF" e "2007 05 11 0000 (Wide).DBF". Note-se que no próprio nome do arquivo está a data em que foi gerado, além disso existem dois tipos de arquivos, cujos nomes são gerados automaticamente pelo Rsview, " Tagname " e " Wide " sendo "tagname" onde estão as variáveis que serão armazenadas e "wide" os dados que são armazenados.

b.2) Os arquivos tipo "wide" possuem estrutura interna em forma de tabela, sendo que a primeira e a segunda coluna guardam a data e hora/minuto/segundo/milésimo de segundo em que ocorreu o evento. As demais colunas são para armazenamento dos eventos na sequência determinada pelo arquivo "tagname".

c) A cada vez que o algoritmo de mineração de dados foi rodado anotou-se sua parametrização e guardou-se o arquivo gerado que possui a extensão “.RES”. Também foi gerado um arquivo de relatório sobre a resposta cuja extensão é “.REP”, que também foi guardado.

d) Todos os arquivos gerados pela execução do NESISES, periodicamente, foram copiados e guardados em um computador diferente do que estava rodando o NESISES para evitar-se o risco de algum sobrescrito acidental.

O fluxograma mostrado na Figura 38 descreve a seqüência de eventos que permitem a rastreabilidade dos testes de software.

Figura 38 - Fluxograma da rastreabilidade dos testes de campo

4.5.5 Avaliação dos testes de campo

Os dados foram avaliados em cada item citado na rastreabilidade (4.3.3). Devido às particularidades dos testes de campo não houve um acompanhamento diário da geração dos arquivos ".DBF", e após um período de tempo de 45 dias foi realizada nova visita técnica à planta industrial, onde se pôde checar a geração dos arquivos e também sua correção, consistência, completude e acurácia.

4.5.6 Análise de interface dos testes de campo

Nessa etapa foram analisados os dados das diversas interfaces de cada parte do software e verificadas sua consistência, correção, completude e acurácia. Nessa etapa foi executada uma grande quantidade de modificações e inclusões no software desenvolvido dentre as quais pode-se citar um mecanismo de escolha de variáveis para serem analisadas e sistema de relatórios para facilitar a análise dos

resultados. A versão da interface utilizada nessa etapa está na Figura 39, onde se pode observar as principais características da interface.

Nota-se o mecanismo de escolha de variáveis e os botões de Gravar Projeto, Ler Projeto e Gerar Relatório que foram necessários para auxiliar a rastreabilidade dos resultados.

Figura 39 - - Interface do NESISES – módulo de mineração de dados - testes de campo

4.5.7 Geração de plano e desenho de teste dos testes de campo

Essas etapas que na norma IEEE 1059 – 1993 eram duas etapas distintas, foram agrupadas de acordo com as necessidades desse trabalho de pesquisa.

Essa etapa foi executada de acordo com o fluxograma mostrado na Figura 40, nota-se que o processo é interativo.

Figura 40 - Plano e desenho de teste do NESISES - testes de campo

4.6 Definição dos critérios de aprovação dos testes de campo com o NESISES

Para se validarem os testes, tanto os da simulação quanto os de campo, foram adotados os seguintes critérios:

1. *Assertividade do aprendizado:* o NESISES tem de possuir aprendizado assertivo.
2. *Desempenho do S.S.:* o S.S. não deve sofrer queda de desempenho apreciável.
3. *Tempo de execução:* o tempo para o NESISES aprender algo do S.S. não pode ser de uma ordem de grandeza superior aos tempos padronizados que os engenheiros do conhecimento têm para atingir as mesmas conclusões.
4. *Compatibilidade:* o NESISES não pode requerer tantos recursos do sistema operacional (Windows) que prejudique seu desempenho, ou

seja, o funcionamento do NESISES não pode prejudicar o funcionamento normal do S.S. em suas tarefas.

4.7 Análise dos resultados segundo os critérios de aprovação dos testes de campo com o NESISES

Após serem executados os testes, os mesmos foram analisados segundo os critérios estabelecidos no item 4.6. Os resultados dessas análises foram os seguintes:

1. *Assertividade do aprendizado:* o NESISES mostrou possuir aprendizado assertivo (vide item 4.5.3).
2. *Desempenho do S.S.:* o S.S. não sofreu queda de desempenho apreciável.
3. *Tempo de execução:* O tempo para o NESISES aprender algo do S.S. foi desde alguns minutos até 36 horas, o que não foi impeditivo já que para grandes volumes de informação um engenheiro do conhecimento poderia levar tempos muito maiores para chegar a uma conclusão.
4. *Compatibilidade:* o NESISES se mostrou totalmente compatível com o sistema operacional (Windows), sendo que a única restrição é que a etapa de mineração de dados ocorra em um computador diferente daquele em que esteja funcionando o S.S.

5 CONCLUSÕES E CONSIDERAÇÕES FINAIS

Com os SS cada vez mais presentes em plantas industriais automatizadas faz-se necessária observar a intensa interação com os operadores do processo, classificados na teoria geral de automação como um processo manual. Eles permanecem em salas de supervisão e de controle e detêm a grande responsabilidade de supervisionar, alterar receitas, *set-points*, inferir conclusões, tomar decisões conforme procedimentos pré-estabelecidos e atuar em situações emergenciais em plantas industriais complexas.

Esta presença de suma importância no processo de automação motivou a realização deste estudo, no questionamento da possibilidade da minimização da participação do operador no sistema de automação a partir de um sistema de aprendizado.

Para o desenvolvimento deste estudo foi necessária a realização de ampla pesquisa em inteligência artificial para se verificarem as possibilidades de sistemas com capacidade de aprendizado. Diversos tópicos de I.A. foram estudados, destacando-se seus fundamentos como o aprendizado PAC, o aprendizado de máquina e principalmente a mineração de dados, que acabou por ser a linha mestra do desenvolvimento que se seguiu.

Ao se aprofundar o estudo da M.D., constatou-se que, devido às suas características de descoberta de conhecimento em grandes bases de dados, a M.D. seria de extrema utilidade para os objetivos deste trabalho de pesquisa, que eram aprender com a experiência do sistema de automação. A experiência então pode ser armazenada em bases de dados e posteriormente utilizada para elucidar o funcionamento de determinado sistema ou ainda propor regras de funcionamento cujos pesos podem ser modificados ao longo do tempo.

O próximo passo foi o desenvolvimento de uma ferramenta computacional que pudesse interligar um sistema supervisório de automação industrial com um sistema especialista e ainda tivesse capacidade de aprendizado, unindo-se, dessa forma, a realidade dinâmica do sistema de automação (planta) com o conhecimento

acumulado no S.E., de forma a capacitar o sistema resultante a tomar inferências sobre o sistema e eventualmente propor regras novas.

Para esse desenvolvimento foi necessário estudar-se a fundo as ferramentas computacionais existentes para executar M.D. ou mesmo partir-se diretamente do conceito de algum algoritmo de M.D. Após pesquisa, foi encontrada uma implementação em software do algoritmo Tertius que atendia às necessidades de realização da M.D. do tipo regras de associação e era flexível o bastante para ser incorporado ao SISES. Optou-se, então, pelo desenvolvimento de software próprio, o qual denominou-se NESISES, que foi o SISES original melhorado e com um módulo de M.D. encarregado de aprender.

O software NESISES foi testado em diversos testes de simulação, sofrendo aprimoramentos constantes, e após os testes de simulação, o mesmo foi testado em condições reais de uma planta industrial automatizada, onde então foram aferidas suas capacidades de aprendizado. O NESISES foi satisfatório em todos os critérios de testes.

Durante a realização deste estudo, outros questionamentos foram apontados que requerem futuros aprimoramentos, destacam-se:

- Implementação do módulo decisório do NESISES baseado em novas pesquisas sobre inteligência artificial;
- Pesquisa em novas técnicas de I.A. para melhorar a abrangência e assertividade do NESISES. Particularmente algoritmos de mineração de dados mais eficientes que o Tertius;
- Pesquisa em novas técnicas de I.A. para desenvolver metodologias e ferramentas de construção do NESISES e S.E;
- A inclusão do fator tempo nas heurísticas de forma a melhor ponderar uma eventual ação do NESISES;
- Melhoria da Interface com o usuário tornando a operação ainda mais amigável.

Assim sendo, este trabalho de pesquisa atingiu os seus objetivos iniciais e contribuiu para a diminuição da dependência dos sistemas de automação a seus operadores pelo uso de sistemas especialistas atuando em tempo real com sistemas de automação industrial.

O trabalho apresenta satisfatória utilização de técnicas de mineração de dados aplicadas a automação industrial, aproximando assim distintas áreas do conhecimento como tecnologia da informação, inteligência artificial e automação industrial.

6 REFERÊNCIAS BIBLIOGRÁFICAS

Agrawal R., Mannila, H., Srikant, R., Toivonen, H., Verkano, I. **Fast Discovery of Association Rules.**: 1996 ,Advances in Knowledge Discovery and mineracao de dados, MIT Press, pp. 307-329.

Agrawal R., T. Imielinski e R. Srikant, **Mining Association Rules between Sets of Items in Large Databases**, *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, Washington, Estados Unidos, 1993, 207–216.

Amo S. , Furtado D. A. **First- Order Temporal Patten Mining with Regular Expression Constraints** XX Simpósio Brasileiro de Banco de Dados - 2005 - Uberlândia, MG.

Andrade A. A. . **Emprego de sistemas especialistas em sistemas supervisórios de automação industrial** 2001, 154 f. Dissertação (Mestrado em Engenharia Elétrica) Escola Politécnica da Universidade de São Paulo, 2001.

Angluin D. **Learning regular sets from queries and counterexamples.** Information and Computation, 1987. 75-87

Angluin D., Abe N., Kharon R., Zeugmann T., **Algorithmic Learning Theory**, 12th International Conference 2001, volume 2225 of Lecture Notes in Computer Science. Springer,2001.

Batista, **G.H.A.P.A . Um Ambiente de Avaliação de Algoritmos de Aprendizado de Máquina Utilizando Exemplos**, São Carlos, USP-São Carlos, 1997.

Brachman R. , Anand T. **The process of knowledge Discovery in Databases: A human centered approach in AKDDM**, The AAAI/MIT Press, pp.37-58, 1996.

Breiman, L. Friedman, J. H. Olshen, R. A., Stone, C. J. **Classification and Decision Trees** , 1984, 1ª ed. Monterrey, California, USA, Wadsworth & Brooks.

Buchanan B.G. , Shortliffe E. H. **Rule-Based Expert Systems**. Addison-Wesley, 1985. 737p.

Deltour A.; **Tertius extension to Weka**. Disponível em: www.cs.bris.ac.uk/Research/MachineLearning/Tertius/. Em 01/05/2007.

Castrucci P.L., Moraes C.C; **Engenharia de automação industrial**. LTC 2ed -2007, p358.

Faloutsos, C., M. Ranganathan, and Y. Manolopoulos, "**Fast Subsequence Matching in Time-Series Databases**," *Proceeding of the ACM SIGMOD*, Minneapolis, Minn., pp.419-429, May 1994.

Fayyad U., Piatetsky-Shapiro G., Smyth P. , **From Data mining to Knowledge discovery in Databases** , American Association for Artificial Intelligence, 1996.

Kirkby R. , Frank E. **WEKA Explorer User Guide for Version 3-5-5**, January 26, 2007 Disponível: <http://opensource.cqu.edu.au/oswins/datamining/weka/ExplorerGuide-3.5.5.pdf>

Flach Peter A. , Lachiche Nicolas **Confirmation-Guided Discovery of First-Order Rules with Tertius**, Machine Learning, vol 42, n 1/2, pgs 61--95, 2001.

Flach, P. A. & Lachiche, N. (1999b). **The Tertius system**. <http://www.cs.bris.ac.uk/Research/MachineLearning/Tertius/>.

Flach Peter A. , Lachiche Nicolas **IBC A first-order representation for knowledge discovery and Bayesian classification on relational data** , Proceedings of the 9th International Workshop on Inductive Logic Programming, p.92-103, June 24-27, 1999.

Ginsberg, M.; **Essentials of Artificial Intelligence**. Morgan Kaufmann Publishers, Inc., 1993.

Gillmeister, P. R. G. ; Cazella, S. C. . **Uma análise comparativa de algoritmos de regras de associação: minerando dados da indústria automotiva**. In: Escola Regional de Banco de Dados, 2007, Caxias do Sul. Escola Regional de Banco de Dados, 2007.

Gomi E. S. **Inteligência Artificial e Programação em Lógica e suas Aplicações nos Sistemas de Supervisão e Controle de Sistemas de Potência** 1989, 132 f. Dissertação (Mestrado em Engenharia Elétrica) Escola Politécnica da Universidade de São Paulo, 1989.

Gonçalves, E. C. . **Regras de Associação e suas Medidas de Interesse Objetivas e Subjetivas**. INFOCOMP - Revista de Ciência da Computação, Universidade Federal de Lavras, v. 4, p. 26 - 35, 01 mar. 2005.

Hand D., Mannila H., Smyth P., **Principles of Data mining** , 2001, Massachusetts Institute of Technology

The Institute of Electrical and Electronics Engineers, Inc. IEEE Std 1059-1993 **IEEE Guide for Software Verification and Validation Plans** ; 1994 ISBN 0-7381-0410-8, SS16923

Jiawei Han , Micheline Kamber **Data mining: Concepts and Techniques**, 2000 , Morgan Kaufmann Publishers.

Kaszakurewicz E ; Bhaya A. ; Ebecken N. F. F. **A Fault Detection and Diagnosis Module for Oil Production Plants in Offshore Platforms** Elsevier Science Expert Systems with applications VI. 12 No 2 pp 189 – 194 ; 1997.

Kearns M. J. **The Computational Complexity of Machine Learning**, 1999, Massachusetts Institute of Technology

Kearns M. J. Valiant, L.G.; **Cryptographic Limitations on Learning Boolean Formulae and Finite Automata**, 1989, Annual ACM Symposium on Theory of Computing Proceedings of the twenty-first annual ACM symposium on Theory of computing

Larose, Daniel T. **Discovering knowledge in data : an introduction to Data mining** -2005 , John Wiley & Sons, 2003, Lawrence Erlbaum Associates, Inc.,

Lu,Yijun , **Concept Hierarchy in Data mining: Specification Generation and Implementation** , 1997 ,SIMON FRASER UNIVERSITY CANADA, Thesis of the requirements for the degree of Master of Science.

M. Last, A. Kandel and H. Bunke **Data mining in time series databases**, World Scientific Publishing Co. 2004 Series in Machine Perception and Artificial Intelligence (Vol. 57)

Michie, D ; Spiegelhalter, D.J. ; Taylor, C.C. **Machine Learning, Neural and Statistical Classification**, Cambridge , 1994, 223p.

Mitchell T.M., **Machine Learning** , McGraw-Hill, 1997, 52p.

Mitchell T.M., **Does Machine Learning Really Work?**, American Association for Artificial Intelligence, 1997.

Nicoletti M. C. , Santos F. O. , **O Modelo de Aprendizado-pac** Universidade Federal de São Carlos (UFSCar) 2005,

Nogueira C. F., **Inteligência Artificial Quebrando Paradigmas na Análise de Empresas** 1999, Editora papel virtual.

Pedrycz W., Peters J. E. **Engenharia de Software**, 2001, Editora Campus Ltda

Pereira S.L. ; Andrade A. A. . **Ambiente de Desenvolvimento de Sistema Especialista para Operar em Tempo Real com Sistema Supervisório de Automação Industrial**, Anais do Congresso CBA 2004 - XV Congresso Brasileiro de Automação Gramado RS , 2004

Piatetsky-Shapiro, G. and W. J. Frawley, **Knowledge Discovery in Databases**, The AAAI/MIT Press, pp.1-27, 1991.

Pitt L, Valiant, L.G. . **Computational Limitations on Learning from Examples** 1988 Journal of the Association for Computing Machinery. Vol. 35. No. 4. October 1988, pp. 965-984

.Pivato M. A. . **Mineração de regras de associação em dados georreferenciados** 2006, 87 f. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) ICMC-USP São Carlos, 2006.

Pressman R. S. . **Software Engineering A practitioner' s approach** . Mc Graw Hill 5ed -2001, p437.

Rockwell Software Inc.. **RSView 32 – User's Guide** -USA, 2003.
Disponível: http://literature.rockwellautomation.com/idc/groups/public/documents/webassets/external_info/hcst?dID=47301&webSize=5671

Rockwell Software Inc.. **RSLogix Emulate 5 – User's Guide** -USA, 1996. 112p.
Disponível: <http://www.ab.com/manuals/swrsi/9399WEGR.pdf>

Rockwell Software Inc.. **RSLinx – User's Guide** Milwaukee -USA, 2001. 63p.
Disponível: <http://www.ab.com/manuals/swrsi/9398LINXTD.pdf>

Russell S. Norvig P, **Artificial Intelligence a Modern Approach**. Prentice Hall 2ed -2004, p384.

Tanimoto, S. **The elements of artificial intelligence. An Introduction Using LISP** , 1987 - Computer Science Press, Inc.

Turing A.M.. **Computing machinery and Intelligence**. Mind a Quartely Review of Psychology and Philosophy, v.LIX N236, October, 1950.

Valiant, L.G.; **A theory of the learnable**. *Communications of the ACM* 27(11), 1984, pp 1134-1142.

Vapnik, Vladimir. **The Nature of Statistical Learning Theory**, Springer, Berlin 1995

Webb J.; Greshock K. **Industrial Control Eletronics**, Maxwell Macmillan International Editions, 1992. 593p.

Weiss G., **Multiagent Systems A Modern Approach to Distributed Modern Approach to Artificial Intelligence**, The MIT Press Pg iv 1999 Massachusetts Institute of Technology.

Witten, I. H. Eibe F., **Data mining : practical machine learning tools and techniques** ,. 2005 ; 2nd ed. Morgan Kaufmann series in data management systems.

Anexo 1

Metodologia de testes preliminares do algoritmo Tertius

Os testes preliminares foram feitos da seguinte maneira:

A) Em um sistema supervisor de processo industrial de secagem de nylon, arbitram-se seis variáveis do tipo digital para amostra.

B) Em planilha do tipo .CSV inseriu-se manualmente 100 linhas com valores aleatórios das seis variáveis com exceção de três, as quais foram forçadas a terem uma relação causal, no caso as três representavam válvulas e a relação foi: *Se XV-20 e XV32 estão abertas então XV-70 estará fechada.*

C) Na interface do WEKA, converteu-se o arquivo .CSV (Que pode ser aberto no Microsoft Excel) para o formato .ARFF específico do WEKA.

D) Rodou-se a base de dados no WEKA e obteve-se o padrão que foi forçado com seus respectivos graus de significância.

E) Converteu-se o arquivo .CSV original nos dois arquivos necessários ao Tertius o **.FCT** que possui os dados propriamente ditos (linhas da planilha CSV) e o **.PRD** que possui as declarações de variáveis (Primeira linha da planilha CSV)

F) Comparou-se os resultados obtidos e verificou-se a robustez do software.

A metodologia descrita nos itens de A a F está representada na figura 27.

Figura 41 - Metodologia utilizada para testes preliminares – Tertius

Os resultados preliminares encontrados por meio do software WEKA, bem como as configurações utilizadas pelo Tertius, encontram-se representados na figura 28.

Figura 42 - Utilização do algoritmo Tertius na plataforma WEKA.

Pode-se observar os dados inseridos nos testes preliminares na tabela A.1.

Nota-se que as seis variáveis digitais inicialmente escolhidas são formatadas com os nomes v1, v2, v3,v4,v5,v6 e que no arquivo tipo .fct, próprio do Tertius,é separado por “%!” e cada item inserido tem seu índice em ordem crescente.

Tabela A. 1– Dados de entrada utilizados para testes preliminares algoritmo

Tertius (Arquivo tipo .fct)

v1(i1,false).	v1(i14,true).	v1(i27,false).	v1(i40,true).	v1(i53,false).	v1(i66,false).	v1(i79,false).	v1(i92,false).
v2(i1,false).	v2(i14,true).	v2(i27,false).	v2(i40,true).	v2(i53,false).	v2(i66,false).	v2(i79,false).	v2(i92,false).
v3(i1,false).	v3(i14,true).	v3(i27,false).	v3(i40,true).	v3(i53,false).	v3(i66,false).	v3(i79,false).	v3(i92,false).
v4(i1,false).	v4(i14,true).	v4(i27,false).	v4(i40,false).	v4(i53,false).	v4(i66,false).	v4(i79,false).	v4(i92,false).
v5(i1,true).	v5(i14,false).	v5(i27,true).	v5(i40,false).	v5(i53,true).	v5(i66,false).	v5(i79,false).	v5(i92,false).
v6(i1,false).	v6(i14,false).	v6(i27,false).	v6(i40,true).	v6(i53,false).	v6(i66,true).	v6(i79,false).	v6(i92,false).
%!	%!	%!	%!	%!	%!	%!	%!
v1(i2,true).	v1(i15,true).	v1(i28,false).	v1(i41,false).	v1(i54,true).	v1(i67,false).	v1(i80,false).	v1(i93,false).
v2(i2,false).	v2(i15,true).	v2(i28,false).	v2(i41,false).	v2(i54,true).	v2(i67,false).	v2(i80,false).	v2(i93,false).
v3(i2,false).	v3(i15,true).	v3(i28,false).	v3(i41,false).	v3(i54,true).	v3(i67,false).	v3(i80,false).	v3(i93,false).
v4(i2,false).	v4(i15,false).	v4(i28,true).	v4(i41,false).	v4(i54,true).	v4(i67,false).	v4(i80,false).	v4(i93,false).
v5(i2,true).	v5(i15,true).	v5(i28,true).	v5(i41,true).	v5(i54,false).	v5(i67,true).	v5(i80,true).	v5(i93,false).
v6(i2,false).	v6(i15,false).	v6(i28,false).	v6(i41,true).	v6(i54,false).	v6(i67,true).	v6(i80,false).	v6(i93,false).
%!	%!	%!	%!	%!	%!	%!	%!
v1(i3,false).	v1(i16,false).	v1(i29,true).	v1(i42,false).	v1(i55,false).	v1(i68,true).	v1(i81,false).	v1(i94,false).
v2(i3,true).	v2(i16,true).	v2(i29,true).	v2(i42,false).	v2(i55,false).	v2(i68,true).	v2(i81,false).	v2(i94,false).
v3(i3,false).	v3(i16,false).	v3(i29,true).	v3(i42,false).	v3(i55,false).	v3(i68,true).	v3(i81,false).	v3(i94,false).
v4(i3,false).	v4(i16,false).	v4(i29,false).	v4(i42,true).	v4(i55,true).	v4(i68,false).	v4(i81,true).	v4(i94,false).
v5(i3,true).	v5(i16,true).	v5(i29,true).	v5(i42,true).	v5(i55,false).	v5(i68,false).	v5(i81,false).	v5(i94,false).
v6(i3,false).	v6(i16,false).	v6(i29,true).	v6(i42,true).	v6(i55,true).	v6(i68,false).	v6(i81,false).	v6(i94,false).
%!	%!	%!	%!	%!	%!	%!	%!
v1(i4,false).	v1(i17,false).	v1(i30,false).	v1(i43,false).	v1(i56,false).	v1(i69,false).	v1(i82,false).	v1(i95,false).
v2(i4,false).	v2(i17,true).	v2(i30,false).	v2(i43,false).	v2(i56,false).	v2(i69,false).	v2(i82,false).	v2(i95,false).
v3(i4,false).	v3(i17,false).	v3(i30,false).	v3(i43,false).	v3(i56,false).	v3(i69,false).	v3(i82,false).	v3(i95,false).
v4(i4,false).	v4(i17,false).	v4(i30,false).	v4(i43,false).	v4(i56,true).	v4(i69,false).	v4(i82,true).	v4(i95,false).
v5(i4,false).	v5(i17,true).	v5(i30,false).	v5(i43,true).	v5(i56,false).	v5(i69,false).	v5(i82,false).	v5(i95,true).
v6(i4,true).	v6(i17,false).	v6(i30,false).	v6(i43,false).	v6(i56,false).	v6(i69,false).	v6(i82,true).	v6(i95,false).
%!	%!	%!	%!	%!	%!	%!	%!
v1(i5,false).	v1(i18,false).	v1(i31,true).	v1(i44,true).	v1(i57,false).	v1(i70,false).	v1(i83,false).	v1(i96,false).
v2(i5,false).	v2(i18,true).	v2(i31,false).	v2(i44,true).	v2(i57,false).	v2(i70,false).	v2(i83,false).	v2(i96,false).
v3(i5,false).	v3(i18,false).	v3(i31,false).	v3(i44,true).	v3(i57,false).	v3(i70,false).	v3(i83,false).	v3(i96,false).
v4(i5,false).	v4(i18,false).	v4(i31,false).	v4(i44,false).	v4(i57,false).	v4(i70,false).	v4(i83,true).	v4(i96,true).

v5(i5,false).	v5(i18,false).	v5(i31,false).	v5(i44,false).	v5(i57,false).	v5(i70,false).	v5(i83,false).	v5(i96,false).
v6(i5,true).	v6(i18,true).	v6(i31,false).	v6(i44,false).	v6(i57,false).	v6(i70,false).	v6(i83,false).	v6(i96,false).
%!	%!	%!	%!	%!		%!	%!
v1(i6,true).	v1(i19,false).	v1(i32,true).	v1(i45,true).	v1(i58,false).	v1(i71,false).	v1(i84,true).	v1(i97,true).
v2(i6,false).	v2(i19,false).	v2(i32,false).	v2(i45,true).	v2(i58,false).	v2(i71,false).	v2(i84,true).	v2(i97,true).
v3(i6,false).	v3(i19,false).	v3(i32,false).	v3(i45,true).	v3(i58,false).	v3(i71,false).	v3(i84,true).	v3(i97,true).
v4(i6,false).	v4(i19,false).	v4(i32,false).	v4(i45,false).	v4(i58,false).	v4(i71,false).	v4(i84,false).	v4(i97,true).
v5(i6,false).	v5(i19,false).	v5(i32,false).	v5(i45,false).	v5(i58,false).	v5(i71,false).	v5(i84,false).	v5(i97,false).
v6(i6,true).	v6(i19,false).	v6(i32,false).	v6(i45,false).	v6(i58,false).	v6(i71,false).	v6(i84,false).	v6(i97,true).
%!	%!	%!	%!	%!	%!	%!	%!
v1(i7,true).	v1(i20,false).	v1(i33,true).	v1(i46,false).	v1(i59,true).	v1(i72,false).	v1(i85,false).	v1(i98,false).
v2(i7,true).	v2(i20,false).	v2(i33,false).	v2(i46,false).	v2(i59,true).	v2(i72,false).	v2(i85,false).	v2(i98,false).
v3(i7,true).	v3(i20,false).	v3(i33,false).	v3(i46,false).	v3(i59,true).	v3(i72,false).	v3(i85,false).	v3(i98,false).
v4(i7,true).	v4(i20,false).	v4(i33,false).	v4(i46,false).	v4(i59,false).	v4(i72,false).	v4(i85,false).	v4(i98,true).
v5(i7,false).	v5(i20,false).	v5(i33,false).	v5(i46,false).	v5(i59,false).	v5(i72,true).	v5(i85,false).	v5(i98,false).
v6(i7,false).	v6(i20,false).	v6(i33,false).	v6(i46,false).	v6(i59,true).	v6(i72,false).	v6(i85,false).	v6(i98,false).
%!	%!	%!	%!	%!	%!	%!	%!
v1(i8,false).	v1(i21,false).	v1(i34,false).	v1(i47,true).	v1(i60,false).	v1(i73,true).	v1(i86,false).	v1(i99,true).
v2(i8,false).	v2(i21,false).	v2(i34,false).	v2(i47,true).	v2(i60,false).	v2(i73,true).	v2(i86,false).	v2(i99,true).
v3(i8,false).	v3(i21,false).	v3(i34,false).	v3(i47,true).	v3(i60,false).	v3(i73,true).	v3(i86,false).	v3(i99,true).
v4(i8,true).	v4(i21,true).	v4(i34,false).	v4(i47,false).	v4(i60,false).	v4(i73,true).	v4(i86,false).	v4(i99,false).
v5(i8,true).	v5(i21,false).	v5(i34,true).	v5(i47,false).	v5(i60,true).	v5(i73,false).	v5(i86,false).	v5(i99,false).
v6(i8,false).	v6(i21,false).	v6(i34,false).	v6(i47,true).	v6(i60,true).	v6(i73,false).	v6(i86,true).	v6(i99,false).
%!	%!	%!	%!		%!	%!	
v1(i9,false).	v1(i22,true).	v1(i35,true).	v1(i48,false).	v1(i61,false).	v1(i74,false).	v1(i87,false).	
v2(i9,false).	v2(i22,true).	v2(i35,true).	v2(i48,false).	v2(i61,false).	v2(i74,false).	v2(i87,false).	
v3(i9,false).	v3(i22,true).	v3(i35,true).	v3(i48,false).	v3(i61,false).	v3(i74,false).	v3(i87,false).	
v4(i9,true).	v4(i22,true).	v4(i35,true).	v4(i48,false).	v4(i61,true).	v4(i74,true).	v4(i87,false).	
v5(i9,false).	v5(i22,true).	v5(i35,false).	v5(i48,true).	v5(i61,true).	v5(i74,false).	v5(i87,true).	
v6(i9,false).	v6(i22,true).	v6(i35,false).	v6(i48,true).	v6(i61,true).	v6(i74,true).	v6(i87,true).	
%!	%!	%!	%!	%!	%!	%!	
v1(i10,false).	v1(i23,true).	v1(i36,false).	v1(i49,false).	v1(i62,true).	v1(i75,false).	v1(i88,false).	
v2(i10,false).	v2(i23,false).	v2(i36,false).	v2(i49,false).	v2(i62,true).	v2(i75,false).	v2(i88,false).	
v3(i10,true).	v3(i23,false).	v3(i36,false).	v3(i49,false).	v3(i62,true).	v3(i75,false).	v3(i88,false).	
v4(i10,false).	v4(i23,true).	v4(i36,true).	v4(i49,false).	v4(i62,false).	v4(i75,false).	v4(i88,true).	
v5(i10,false).	v5(i23,false).	v5(i36,false).	v5(i49,false).	v5(i62,true).	v5(i75,true).	v5(i88,true).	
v6(i10,false).	v6(i23,true).	v6(i36,true).	v6(i49,false).	v6(i62,false).	v6(i75,true).	v6(i88,true).	
%!	%!	%!	%!	%!	%!	%!	
v1(i11,true).	v1(i24,true).	v1(i37,false).	v1(i50,true).	v1(i63,false).	v1(i76,false).	v1(i89,false).	
v2(i11,false).	v2(i24,false).	v2(i37,false).	v2(i50,true).	v2(i63,false).	v2(i76,false).	v2(i89,false).	
v3(i11,false).	v3(i24,false).	v3(i37,false).	v3(i50,true).	v3(i63,false).	v3(i76,false).	v3(i89,false).	
v4(i11,false).	v4(i24,false).	v4(i37,true).	v4(i50,false).	v4(i63,false).	v4(i76,false).	v4(i89,false).	

v5(i11,false).	v5(i24,false).	v5(i37,false).	v5(i50,false).	v5(i63,false).	v5(i76,false).	v5(i89,true).
v6(i11,true).	v6(i24,true).	v6(i37,false).	v6(i50,false).	v6(i63,false).	v6(i76,false).	v6(i89,false).
%!	%!	%!	%!	%!	%!	%!
v1(i12,true).	v1(i25,true).	v1(i38,false).	v1(i51,false).	v1(i64,false).	v1(i77,false).	v1(i90,false).
v2(i12,false).	v2(i25,false).	v2(i38,false).	v2(i51,false).	v2(i64,false).	v2(i77,false).	v2(i90,false).
v3(i12,false).	v3(i25,false).	v3(i38,false).	v3(i51,false).	v3(i64,false).	v3(i77,false).	v3(i90,false).
v4(i12,false).	v4(i25,false).	v4(i38,false).	v4(i51,false).	v4(i64,false).	v4(i77,false).	v4(i90,false).
v5(i12,false).	v5(i25,false).	v5(i38,false).	v5(i51,false).	v5(i64,false).	v5(i77,false).	v5(i90,true).
v6(i12,false).	v6(i25,false).	v6(i38,false).	v6(i51,false).	v6(i64,false).	v6(i77,false).	v6(i90,true).
%!	%!	%!	%!	%!	%!	%!
v1(i13,false).	v1(i26,false).	v1(i39,false).	v1(i52,false).	v1(i65,false).	v1(i78,false).	v1(i91,false).
v2(i13,true).	v2(i26,false).	v2(i39,false).	v2(i52,false).	v2(i65,false).	v2(i78,false).	v2(i91,false).
v3(i13,false).	v3(i26,false).	v3(i39,false).	v3(i52,false).	v3(i65,false).	v3(i78,false).	v3(i91,false).
v4(i13,false).	v4(i26,false).	v4(i39,false).	v4(i52,false).	v4(i65,false).	v4(i78,false).	v4(i91,false).
v5(i13,false).	v5(i26,false).	v5(i39,false).	v5(i52,false).	v5(i65,false).	v5(i78,false).	v5(i91,false).
v6(i13,false).	v6(i26,false).	v6(i39,false).	v6(i52,false).	v6(i65,false).	v6(i78,false).	v6(i91,false).

Tão importante quanto o arquivo tipo **.fct**, em que estão os dados que serão analisados, é o arquivo tipo **.prd** onde estão declaradas as variáveis que serão utilizadas, devido a falhas na documentação disponível sobre a implementação do algoritmo. Para chegar-se à correta declaração das variáveis digitais foi necessário um contato estreito com os senhores Flach e Lachiche cujo apoio foi decisivo nessa fase do trabalho, e as variáveis foram declaradas da seguinte forma:

```

v1 2 id #boolean * cwa
v2 2 id #boolean * cwa
v3 2 id #boolean * cwa
v4 2 id #boolean * cwa
v5 2 id #boolean * cwa
v6 2 id #boolean * cwa

```

Onde: **v1** é a variável ou no caso predicado.

2 é a “arity”, ou seja, o número de argumentos que a função pode receber.

id #boolean é o tipo de argumento no caso binário.

* **cwa** mostra que para esse tipo de argumento vale a definição do tipo “closed world assumption” ou suposição de mundo fechado.

A correspondência entre as variáveis utilizadas no WEKA e no Tertius original está na tabela A. 2.

Tabela A. 2– Correspondência entre variáveis inseridas no Tertius original e no Tertius WEKA

Tertius WEKA	Tertius original
K34_1_V_32_OP	V1
K34_1_V_20_OP	V2
K34_1_V_70_OP	V3
K34_1_V_33_OP	V4
K34_1_V_53_OP	V5
K34_1_PV_15_OP	V6

A tabela A.3 mostra as trinta regras mais prováveis encontradas com a implementação original do Tertius e a inserida no WEKA

Tabela A. 3– Comparação entre as regras encontradas no Tertius original e no Tertius WEKA

Regra encontrada no Tertius original	Regra encontrada no Tertius WEKA
/* 0.642956 0.000000 */v1(A,false); v2(A,false) :- v3(A,false).	1. /* 0,642956 0,000000 */ K34_1_V_70_OP = falso ==> K34_1_V_20_OP = falso or K34_1_V_32_OP = falso
/* 0.616305 0.000000 */v1(A,false); v3(A,false) :- v2(A,false).	2. /* 0,616305 0,000000 */ K34_1_V_20_OP = falso ==> K34_1_V_70_OP = falso or K34_1_V_32_OP = falso
/* 0.600192 0.010101 */v3(A,false) :- v2(A,false).	3. /* 0,600192 0,010101 */ K34_1_V_20_OP = falso ==> K34_1_V_70_OP = falso
v2(A,false); v3(A,false) :- v1(A,false).	4. /* 0,583219 0,000000 */ K34_1_V_32_OP = falso ==> K34_1_V_20_OP = falso or K34_1_V_70_OP = falso
/* 0.580181 0.050505 */v3(A,true) :- v2(A,true).	5. /* 0,580181 0,050505 */ K34_1_V_70_OP = falso ==> K34_1_V_20_OP = falso
/* 0.564611 0.010101 */v3(A,false) :- v1(A,false).	6. /* 0,564611 0,010101 */ K34_1_V_32_OP = falso ==> K34_1_V_70_OP = falso
/* 0.531547 0.101010 */v3(A,true) :- v1(A,true).	7. /* 0,531547 0,101010 */ K34_1_V_70_OP = falso ==> K34_1_V_32_OP = falso
/* 0.501083 0.020202 */v2(A,false); v5(A,true) :- v3(A,false).	8. /* 0,501083 0,020202 */ K34_1_V_70_OP = falso ==> K34_1_V_20_OP = falso or K34_1_V_53_OP = verdadeiro
/* 0.498361 0.050505 */v2(A,false) :- v1(A,false).	9. /* 0,498361 0,050505 */ K34_1_V_32_OP = falso ==> K34_1_V_20_OP = falso
/* 0.485248 0.101010 */v2(A,true) :- v1(A,true).	10. /* 0,485248 0,101010 */ K34_1_V_20_OP = falso ==>

	K34_1_V_32_OP = falso
/* 0.484127 0.020202 */v2(A,false) :- v3(A,false), v5(A,false).	11. /* 0,445814 0,020202 */ K34_1_V_32_OP = falso ==> K34_1_V_20_OP = falso or K34_1_V_53_OP = verdadeiro
/* 0.435409 0.060606 */v1(A,false) :- v3(A,false), v6(A,false).	12. /* 0,435409 0,060606 */ K34_1_V_70_OP = falso and K34_1_PV_15_OP = falso ==> K34_1_V_32_OP = falso
/* 0.417156 0.010101 */v3(A,true); v5(A,true); v6(A,true) :- v2(A,true).	13. /* 0,417156 0,010101 */ K34_1_V_70_OP = falso and K34_1_V_53_OP = falso and K34_1_PV_15_OP = falso ==> K34_1_V_20_OP = falso
/* 0.416190 0.090909 */v1(A,false); v5(A,true) :- v3(A,false).	14. /* 0,416190 0,090909 */ K34_1_V_70_OP = falso ==> K34_1_V_53_OP = verdadeiro or K34_1_V_32_OP = falso
/* 0.414807 0.040404 */v2(A,false) :- v3(A,false), v6(A,false).	15. /* 0,414807 0,040404 */ K34_1_V_70_OP = falso and K34_1_PV_15_OP = falso ==> K34_1_V_20_OP = falso
/* 0.402260 0.060606 */v1(A,false) :- v2(A,false), v6(A,false).	16. /* 0,402260 0,060606 */ K34_1_V_20_OP = falso and K34_1_PV_15_OP = falso ==> K34_1_V_32_OP = falso
/* 0.382613 0.010101 */v2(A,false) :- v1(A,false), v5(A,false), v6(A,false).	17. /* 0,382613 0,010101 */ K34_1_V_53_OP = falso and K34_1_PV_15_OP = falso and K34_1_V_32_OP = falso ==> K34_1_V_20_OP = falso
/* 0.378716 0.090909 */v1(A,false); v5(A,true) :- v2(A,false).	18. /* 0,378716 0,090909 */ K34_1_V_20_OP = falso ==> K34_1_V_53_OP = verdadeiro or K34_1_V_32_OP = falso
/* 0.371489 0.050505 */v3(A,true); v6(A,true) :- v1(A,true), v5(A,false).	19. /* 0,371489 0,050505 */ K34_1_V_70_OP = falso and K34_1_PV_15_OP = falso ==> K34_1_V_53_OP = verdadeiro or K34_1_V_32_OP = falso
/* 0.369711 0.040404 */v1(A,true); v6(A,true) :- v2(A,true).	20. /* 0,369711 0,040404 */ K34_1_PV_15_OP = falso and K34_1_V_32_OP = falso ==> K34_1_V_20_OP = falso
/* 0.366614 0.090909 */v1(A,false) :- v3(A,false), v4(A,false).	21. /* 0,366614 0,090909 */ K34_1_V_70_OP = falso and K34_1_V_33_OP = falso ==> K34_1_V_32_OP = falso
/* 0.343981 0.050505 */v2(A,true); v6(A,true) :- v1(A,true), v5(A,false).	22. /* 0,343981 0,050505 */ K34_1_V_20_OP = falso and K34_1_PV_15_OP = falso ==> K34_1_V_53_OP = verdadeiro or K34_1_V_32_OP = falso
/* 0.322571 0.010101 */v1(A,false) :- v3(A,false), v5(A,true).	23. /* 0,322571 0,010101 */ K34_1_V_70_OP = falso and K34_1_V_53_OP = verdadeiro ==> K34_1_V_32_OP = falso
/* 0.321241 0.090909 */v1(A,false) :- v2(A,false), v4(A,false).	24. /* 0,321241 0,090909 */ K34_1_V_20_OP = falso and K34_1_V_33_OP = falso ==> K34_1_V_32_OP = falso
/* 0.311535 0.000000 */v2(A,false); v4(A,false) :- v3(A,false).	25. /* 0,311535 0,000000 */ K34_1_V_70_OP = falso ==> K34_1_V_20_OP = falso or K34_1_V_33_OP = falso
/* 0.301140 0.000000 */v2(A,true) :- v3(A,true), v4(A,true).	26. /* 0,301140 0,000000 */ K34_1_V_20_OP = falso ==> K34_1_V_70_OP = falso or K34_1_V_33_OP = falso
/* 0.289379 0.010101 */v1(A,false) :- v2(A,false), v5(A,true).	27. /* 0,289379 0,010101 */ K34_1_V_20_OP = falso and K34_1_V_53_OP = verdadeiro ==> K34_1_V_32_OP = falso
/* 0.289161 0.080808 */v3(A,true); v4(A,true) :- v1(A,true), v5(A,false).	28. /* 0,289161 0,080808 */ K34_1_V_70_OP = falso and K34_1_V_33_OP = falso ==> K34_1_V_53_OP = verdadeiro or K34_1_V_32_OP = falso
/* 0.287988 0.000000 */v1(A,true) :- v3(A,true), v4(A,true).	29. /* 0,287988 0,000000 */ K34_1_V_32_OP = falso ==> K34_1_V_20_OP = falso or K34_1_V_33_OP = falso
/* 0.287988 0.000000 */v1(A,true) :- v2(A,true), v4(A,true).	30. /* 0,287988 0,000000 */ K34_1_V_32_OP = falso ==> K34_1_V_70_OP = falso or K34_1_V_33_OP = falso

Observando os resultados explicitados na tabela 4.3 nota-se que os resultados são idênticos na interface WEKA e no Tertius original, menos de diferenças sintáticas como por exemplo, no Tertius original, o operador lógico “and” é representado por “,” e o operador lógico “or” é representado por “;” enquanto que

no WEKA os operadores “or” e “and” são explicitados diretamente. Notou-se que o Tertius original tende a duplicar regras que são equivalentes do ponto de vista lógico, como por exemplo:

```
/* 0.580181 0.050505 */ v3(A,true) :- v2(A,true). E /* 0.580181 0.050505 */
v2(A,false) :- v3(A,false). Que no WEKA fica /* 0,580181 0,050505 */
K34_1_V_70_OP = falso ==> K34_1_V_20_OP = falso
```

Devido à robustez que apresentou nos testes preliminares, o Tertius foi utilizado no restante desse trabalho.

ANEXO 2

Programa ladder elaborado para os testes de simulação

LAD 11 - SIMULACAO2 --- Total Rungs in File = 9

LAD 11 - SIMULACAO2 --- Total Rungs in File = 9

LAD 11 - SIMULACAO2 --- Total Rungs in File = 9

LAD 11 - SIMULACAO2 --- Total Rungs in File = 9

LAD 11 - SIMULACAO2 --- Total Rungs in File = 9