COPPE/UFRJ

FUSÃO DE DADOS MULTI-NÍVEL EM AMBIENTES DE MONITORAÇÃO CONTÍNUA DE SISTEMAS TÁTICOS NAVAIS UTILIZANDO MÚLTIPLAS LÓGICAS

José Gomes de Carvalho Júnior

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Civil, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Civil.

Orientador: Nelson Francisco Favilla Ebecken

Rio de Janeiro

Março de 2010

FUSÃO DE DADOS MULTI-NÍVEL EM AMBIENTES DE MONITORAÇÃO CONTÍNUA DE SISTEMAS TÁTICOS NAVAIS UTILIZANDO MÚLTIPLAS LÓGICAS

José Gomes de Carvalho Júnior

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA CIVIL.

Examinada por:

_____
Prof. Nelson Francisco Favilla Ebecken , D.Sc.

_____
Prof. Antônio Cesar Ferreira Guimarães, D.Sc.

_____
Prof. Álvaro Luiz Gayoso de Azeredo Coutinho, D.Sc.

_____
Prof.Alexandre Gonçalves Evsucoff, Dr.

_____
Prof. Carlos Alberto Nunes Cosenza, D.Sc.

RIO DE JANEIRO, R.J. - BRASIL
MARÇO DE 2010

Carvalho Jr., José Gomes de

Fusão de dados multi-nível em ambientes de monitoração contínua de sistemas táticos navais utilizando múltiplas lógicas / José Gomes de Carvalho Júnior – Rio de Janeiro: UFRJ/COPPE, 2010

XIII, 106p.: i1; 29,7 cm

Orientador: Nelson Francisco Favilla Ebecken

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia Civil, 2010

Referências Bibliográficas: p 100-106

1. Fusão de dados multi-sensores. 2. Inferência com múltiplas lógicas. 3. Sistemas de tempo real. I. Ebecken, Nelson Francisco Favilla. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Civil. III. Título.

Computadores não servem para nada. Eles só podem lhe dar respostas.
Picasso

Este trabalho é dedicado os meus pais, José (in memoriam) e Arlette, pelos exemplos de dedicação, trabalho e amor, bem como à minha esposa Rossani e meus filhos Vinícius, Matheus e Esther, pelo muito que me inspiram.

AGRADECIMENTOS

A Deus, pela sustentação e pelo conforto espiritual de sua contínua presença.

A minha companheira, Rossani, pelo apoio e incentivo de muitos anos.

A minha mãe, Arlette, pela dedicação e pela intercessão cotidiana junto ao Criador.

Aos meus filhos Vinícius, Matheus e Esther, pela compreensão de algumas ausências.

Ao meu orientador, Professor Nelson Ebecken, por sua competência, disponibilidade constante, simplicidade e clareza ao transmitir seus conhecimentos e pela experiência na condução do processo de confecção deste trabalho.

Aos demais professores com quem travei contacto no decorrer do curso no PEC, Prof. Alexandre Evsukof, Prof.ª. Beatriz Lima, Prof. Nelson Ebecken e Prof. Carlos Cosenza, pela dedicação e interesse que demonstraram no processo de ensino-aprendizagem.

Aos demais mestres de minha vida, inumeráveis, que com apenas uma ou com muitas pedras me ajudaram a pavimentar o caminho até aqui.

Aos também inumeráveis alunos com quem pude ter o prazer de dividir um pouco do que aprendi e com os quais igualmente aprendi inúmeras lições.

Ao amigo e parceiro de trabalho, Pablo, por sua presença motivadora, sua paciência em nossas intermináveis discussões, sua lealdade pessoal e profissional, sua fraternal divisão de tarefas e a co-autoria em alguns elementos de suporte desta tese, que servirão também como elementos de suporte em sua dissertação.

Aos colegas de trabalho, José Ricardo e Yuan, que colaboraram com pacotes, dados e explicações sobre modelos que são usados direta ou indiretamente neste trabalho.

Aos colegas de trabalho, Rubens, Rogério e Mônica, que colaboraram com discussões interessantes e motivadoras sobre nossos mútuos trabalhos e aos demais colegas no PEC e no IPqM que direta ou indiretamente colaboraram com seu apoio e estímulo.

Aos colegas de trabalho, Cmdtes. Nicolino e Da Mota, que colaboraram com suas expertises sobre o meio operativo naval.

Aos meus chefes no Instituto de Pesquisas da Marinha (IPqM) neste período de estudos, Cmdtes. Vianna Tavares, Roberto Martins, Cardoso, Marins e Andrada, pelos incentivos moral e profissional, pelo suporte logístico e pela confiança que sempre demonstraram no meu trabalho.

A Marinha do Brasil por possibilitar e incentivar o aperfeiçoamento profissional.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.).

FUSÃO DE DADOS MULTI-NÍVEL EM AMBIENTES DE MONITORAÇÃO CONTÍNUA DE SISTEMAS TÁTICOS NAVAIS UTILIZANDO MÚLTIPLAS LÓGICAS

José Gomes de Carvalho Júnior

Março / 2010

Orientador: Nelson Francisco Favilla Ebecken

Programa: Engenharia Civil

Neste trabalho é proposto um modelo de fusão de dados para sistemas de monitoração contínua do ambiente tático naval que incorpora, além da fusão de objetos identificados por diferentes sensores, a fusão com as informações do contexto que podem ser oriundas de fontes humanas, de bancos de dados de informações geográficas ou de dados recebidos por sistemas de comunicação. O modelo proposto incorpora um motor de inferências e dá suporte ao desenvolvimento de sistemas em ambientes de monitoração contínua de tempo real com regras que usam diferentes tipos de lógicas, tais como a lógica de primeira ordem, a lógica fuzzy e a lógica temporal. No modelo proposto, a abordagem declarativa expressa nas regras lógicas é compatibilizada com a abordagem procedimental presente nas arquiteturas de software orientadas a objetos, possuindo a agilidade requerida em ambientes de monitoração contínua em tempo real. O modelo de fusão de alvos é dinamicamente ajustado de acordo com a qualidade do modelo de estimação de características que o precede. A validação foi realizada com dados simulados e com dados reais oriundos de testes com navios da Marinha do Brasil.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

MULTI-LEVEL DATA FUSION FOR CONTINUOUS MONITORING NAVAL TACTICAL SYSTEMS USING MULTIPLE LOGICS

José Gomes de Carvalho Júnior

March / 2010

Advisor: Nelson Francisco Favilla Ebecken

Department: Civil Engineering

This work proposes a data fusion model for naval tactical scenarios, in which data from different sensors were integrated with data from the context in order to provide a situation awareness picture. The model includes a framework, which deals with data fusion into an integrated approach including a reasoning process that supports rules expressed in first order logic, fuzzy logic and temporal logic. Data fusion is done between targets provided by sensors and between those targets and context information provided by human reports, geographic data banks and information received by data link. The framework reconciles declarative paradigm expressed by rules with procedural paradigm used in object oriented architectures. The whole model is focused in real time monitoring systems. The target fusion model is dynamically adjusted by actual quality of tracker model in which it is based. The model was validated with a multi-level data fusion application that deals with data provided by a tactical simulator and also by actual data.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1 INTRODUÇÃO

1.1 DESCRIÇÃO DO PROBLEMA

Os sistemas táticos operativos são aplicações computacionais dedicadas que visam apresentar um panorama tático integrado aos operadores de uma determinada unidade, seja ela um navio, um helicóptero, um avião ou uma estação terrestre como um aeroporto, tanto com objetivos civis como controle do tráfego aéreo, como com objetivos militares para suporte a decisões estratégicas em um determinado cenário. Neste panorama tático encontram-se representados os elementos de interesse para o acompanhamento do deslocamento de outras unidades móveis que se encontram em uma região no entorno da unidade em questão. Essas unidades, são chamadas genericamente de acompanhamentos, são posicionadas no panorama tático com os dados obtidos a partir de diferentes sensores, tais como radares, sonares, bóias radiosônicas, alças optrônicas, receptores GPS (Global Position System), enlaces de dados captados pelos sensores de outras unidades, receptor de sinais de rádio codificado e acoplado a antenas de radar (IFF), equipamento de guerra eletrônica (MAGE), além de acompanhamentos introduzidos manualmente pelos operadores a partir de detecções visuais ou recebidos por reporte de sensores não integrados ao sistema tático.

Ocorre que os sensores são cada vez mais sofisticados, precisos, disponíveis em maior quantidade e capazes de gerar uma crescente quantidade de informação. Entretanto, muitas destas informações são redundantes ou estão correlacionadas, pois os sensores possuem áreas de interseção em seu escopo de operação. Assim, para os operadores dos sistemas táticos é cada vez mais importante que estes possuam algoritmos capazes de compilar os dados proveniente dos distintos sensores, apresentando um quadro tático consolidado de forma a facilitar a compreensão e a tomada de decisões.

Uma das características do problema é que existe um legado de sistemas táticos atualmente disponíveis no Brasil e qualquer solução proposta deve ser passível de ser acoplada aos sistemas existentes, tratando a fusão a partir das informações e dados normalmente disponíveis nos sensores e fontes de dados mais comuns a estes sistemas, sempre considerando essas informações de forma individual (que é como elas já se apresentam nesses sistemas) e procurando compô-las em um ambiente integrado que leve em consideração as limitações, a precisão, a freqüência, o escopo e a disponibilidade de cada sensor disponível.

As unidades carecem também de uma fusão das informações dos sensores com informações de mais alto nível. De fato, é necessário um sistema de apoio à decisão que integre as informações dos sensores, do contexto geo-climático e do contexto tático de forma a maximizar a eficiência das decisões tomadas em períodos exíguos de tempo e a partir de cenários que podem estar congestionados de informações. Ou seja, além da capacidade de compilação da posição e cinemática dos alvos no cenário tático, que já existe hoje, é desejável que haja um sistema que seja capaz de incorporar características que permitam reconhecer padrões de formação e manobras, associar dados dos acompanhamentos identificados com informações do contexto, tais como informações climáticas (vento, corrente, etc.), geográficas (linha de costa, profundidade, rotas de navegação) ou táticas (situação de operação emanada pelo comando da operação, por exemplo), deduzir intenções dos oponentes (prever tempos e distâncias para ataques) e sugerir ações de manobras.

Por fim, um item desejável de tais sistemas, embora possa parecer prosaico, efetivamente tem fortes razões culturais e operativas. Esse item diz respeito à possibilidade de explicação das conclusões tomadas. Em ambientes militares, é comum que os operadores de distintos níveis somente confiem em informações ou recomendações derivadas automaticamente por um sistema de suporte á decisão, se este puder oferecer razões compreensíveis a um elemento humano, de forma a permitir a validação das decisões. Dito em outras palavras, o elemento humano, no meio militar, deve ser sempre o responsável em última instância pelas decisões.

As necessidades relatadas apontam na direção do uso de sistemas em que o conhecimento possa ser expresso de forma compreensível por especialistas, que é uma característica dos sistemas baseados em conhecimento (KBS do acrônimo no idioma Inglês). Em tais sistemas o conhecimento é expresso por meio de regras (chamado de paradigma declarativo), o que atende às necessidades de justificar decisões e possibilitar validá-las. Entretanto, o uso do paradigma declarativo deve ser compatibilizado com os sistemas táticos existentes, que possuem características de sistemas de monitoração de tempo real e são projetados com arquiteturas orientadas a objetos e implantados com linguagens orientadas a objetos, tais como Java e C++.

1.2 OBJETIVOS

Este trabalho propõe, implementa e valida um *framework* para fusão de dados que foi criado com os seguintes objetivos principais:

- Tratar o problema de fusão de dados com uma abordagem que integra os distintos níveis em que tradicionalmente é dividido o problema de fusão. A hipótese é que a fusão de dados pode ser expressa como resultado de relacionamentos de interesse entre os objetos do domínio e que essas associações perpassam os níveis tradicionais em que é dividido o problema geral de fusão de dados (que são explicados na seção 2.3).
- Expressar os relacionamentos de interesse em regras de produção, de forma a tornar possível a aplicação de um mecanismo de inferências lógicas e o oferecimento de explicações sobre as decisões tomadas que sejam compreensíveis por um elemento humano.
- Permitir que o raciocínio expresso nas regras contemplasse o uso de diferentes lógicas, de forma a aumentar a expressividade do modelo de comportamento, incluindo-se o raciocínio com conhecimento impreciso e a dimensão temporal na elaboração das regras.
- Incorporar um mecanismo que realize automaticamente as inferências a partir da alteração dos dados de forma eficiente para todas as lógicas, já que a fusão é realizada em sistemas aplicativos de tempo real, onde a taxa de alteração de dados é grande.
- Facilitar a aderência dos relacionamentos de interesse com as técnicas de modelagem de software modernas que utilizam arquiteturas e linguagens orientadas a objetos.
- Validar o *framework* e o modelo de fusão propostos em um sistema tático simulado, implantando exemplos de fusão de dados nos distintos níveis propostos pelo modelo mais aceito para fusão de dados (esse modelo, o JDL, e seus níveis são explicados na seção 2.3). Essa validação cita os distintos níveis do modelo JDL apenas como referência, uma vez que o problema de fusão é tratado nesta validação de uma forma integrada, sem o compartilhamento previsto no modelo JDL.

1.3 DESCRIÇÃO DO DOCUMENTO

No capítulo 2 deste trabalho é descrito o problema da fusão de dados sob o ponto de vista que a comunidade da área o aborda. É descrito o modelo conceitual mais utilizado pelos pesquisadores para desenvolver e classificar as funções de fusão de dados. São detalhados os níveis em que se divide o modelo, bem como as funcionalidades previstas em cada nível e as técnicas e ferramentas utilizadas em cada um desses níveis.

No capítulo 3 é relatada a pesquisa realizada sobre as características dos modelos baseados em conhecimento existentes atualmente no mercado e é feita uma avaliação de suas virtudes e defeitos, particularmente no que se refere à aderência às arquiteturas orientadas a objetos e aos sistemas de monitoração em tempo real.

No capítulo 4 são então apresentados os detalhes da arquitetura do *framework* que foi proposto e implementado para dar suporte ao modelo de fusão de dados, incluindo explicações sobre as diferenças em relação aos *frameworks* existentes, os mecanismos de inferência, a sintaxe e a semântica utilizadas para as distintas lógicas empregadas e a aderência às arquiteturas orientadas a objetos e aos sistemas de tempo real.

No início do capítulo 5 é descrito o ambiente de um sistema de Comando, Controle, Comunicações, Computadores e Inteligência (C^4I) atualmente existente em navios da Marinha do Brasil e que foi usado como sistema alvo para validação do modelo de fusão e do *framework* propostos. A seguir são descritas as restrições impostas por esse sistema, que é tratado com um sistema legado e é relatada a arquitetura do ambiente de simulação desse sistema legado que foi efetivamente utilizado para validar o modelo de fusão e o *framework*. No restante do capítulo 5 são detalhados os testes de fusão planejados e executados nesse ambiente e são apresentadas as métricas utilizadas e os resultados colhidos nessa validação.

No capítulo 6 é apresentada a conclusão do trabalho e são relatados trabalhos futuros previstos.

2 FUSÃO DE DADOS

2.1 O CONCEITO DE FUSÃO DE DADOS

O termo “fusão de dados” se aplica a uma variedade de situações onde existem diferentes conjuntos de dados, produzidos por diferentes fontes, que podem ser usados para produzir inferências que não seriam possíveis ou seriam mais limitadas, caso fossem produzidas pelos dados oriundos de uma só fonte.

Os dados produzidos por essas fontes podem ser dos mais diferentes formatos, freqüência, resolução, acurácia, precisão e grandeza, além de se referirem às mais diversas aplicações. Em uma aplicação militar podem ser dados de sensores militares referentes a objetos de interesse tático; em uma aplicação médica podem ser dados oriundos de equipamentos médicos referentes a um paciente; em uma aplicação de geologia, podem ser dados referentes à composição do solo oriundos de diferentes satélites; ou ainda, sem naturalmente esgotar as possíveis aplicações, podem ser dados oriundos de diferentes máquinas em uma aplicação de controle industrial.

A fusão de dados se aplica a diferentes massas de dados, tais como informações de cinemática e formato de objetos, séries temporais de medidas, imagens de diferentes formatos e dimensões e informações simbólicas categóricas disponíveis em bancos de dados. Desta forma, as aplicações se estendem a muitas áreas do conhecimento, tais como engenharia [1], medicina [2,3], segurança [4], fusão de imagens e texto [5], entre inúmeras outras. De forma genérica, o objetivo do processo de fusão é a combinação dos dados e conhecimentos de diferentes fontes com o objetivo de maximizar a utilidade das informações, eliminando redundâncias, diminuindo a incerteza e aumentando a disponibilidade das informações.

De acordo tanto com a complexidade do problema que se objetiva resolver quanto com o nível de refinamento da informação associada aos dados, a fusão de dados pode envolver o uso de técnicas tais como processamento de sinais, estatísticas de estimação, reconhecimentos de padrões, inteligência artificial e técnicas de sistemas de informação.

2.2 FUSÃO DE DADOS EM AMBIENTES TÁTICOS MITARES

Nas aplicações militares de fusão de dados [6, 7, 8], as informações disponíveis são referentes a objetos de natureza militar, tais como aviões, helicópteros, navios, submarinos, tanques, mísseis, grupos de unidades operativas

tais como batalhões, esquadras, frotas, pelotões, etc. Os dados são gerados por sensores tais como radares, sonares, sensores de raios infravermelhos, dispositivos transdutores instalados em veículos participantes do cenário tático, dispositivos de comunicação que repassem relatos sobre alvos remotamente detectados e acompanhados, além de relatos de identificação visual humana. Neste caso, os objetos são referenciados genericamente como alvos e sobre eles pode-se realizar diferentes procedimentos, com variados objetivos, tais como reconhecimento da existência do alvo a partir de um conjunto de dados básicos oriundos de um sensor, determinação da posição do alvo, extração das características cinemáticas do alvo, classificação do grau de hostilidade do alvo, apoio à decisão de engajamento com um armamento, fusão da representação sintética de alvos oriundos de diferentes sensores, determinação da identidade do alvo, previsão das intenções de manobra de grupos de alvos a partir da fusão de informações do cenário com informações de especialistas, entre outras aplicações possíveis.

Devido ao grande escopo dos procedimentos e aplicações possíveis de serem realizadas com os dados, foram propostos alguns modelos de classificação dos sistemas segundo suas funcionalidades e os níveis de dados com que as aplicações trabalham. Esses modelos, de diferentes laboratórios e grupos de pesquisa, acabaram por ser unificados em um modelo que sofreu colaboração de diversas organizações de pesquisa e é, atualmente, em geral aceito pela literatura da área [9, 10] como padrão de referência de funcionalidade e terminologia para os sistemas de fusão de alvos. O modelo foi criado a partir da fundação, em 1986, do *Joint Directors of Laboratories Data Fusion Working Group*, que originou, logo após, a publicação do modelo. O modelo é chamado de *Joint Directors of Laboratories Data Fusion Group Model*, ou simplesmente JDL. Após a primeira versão do modelo [11], houve algumas revisões [12], que incorporaram novas funcionalidades de maior nível. O JDL é um modelo conceitual que identifica processos e técnicas, e não um modelo de processo, em que os dados fluem de um nível para outro. A abrangência do modelo não se restringe ao campo militar, e tem sido empregado na classificação de sistemas da área de processos, robótica e medicina.

2.3 O MODELO JDL

O JDL [11, 12] é um modelo conceitual para sistemas de fusão de alvos, que identifica e detalha cada processo necessário a um determinado objetivo de fusão, separando esses processos em níveis de acordo com a quantidade de informação

agregada aos dados com que o sistema daquele nível trabalha. De forma bem genérica, pode-se dizer que foram propostos os seguintes níveis de sistemas:

Nível 0 - pré processamento de dados de sensores: adaptação do interfaceamento com os sensores (ótico/magnético/elétrico) para leitura dos sinais, conversão de codificação (analógica/digital), translações de espaço (fase/freqüência/amplitude), filtragem de ruído e demais tratamentos de sinais necessários à obtenção de dados confiáveis no escopo do sensor tratado.

Nível 1 – processamento de objetos: fusão de dados oriundos de diferentes sensores com o objetivo de determinar a posição, velocidade, aceleração e rumo do alvo.

Nível 2 – análise de contexto: interpretação do contexto composto pelos alvos, o ambiente geográfico, as condições climáticas, o relacionamento dos alvos entre si e com o cenário geográfico de forma a identificar novas entidades como formaturas de alvos, padrões de movimento e ações de engajamento que estejam em curso.

Nível 3 – análise de impacto: estabelece hipóteses de evolução do cenário tático, procurando prever as conseqüências da evolução do contexto atual frente aos alvos presentes, às entidades identificadas e às ações em curso. Procura identificar as ações previsíveis dos entes envolvidos e as ações recomendáveis ao operador do sistema.

Nível 4 – refinamento do processo: monitora a evolução do processo de fusão, procurando correções de alto nível na utilização ou ponderação dos sensores envolvidos na obtenção das informações dos níveis 1 a 3. Pode determinar o sensor mais adequado para obter informações sobre um determinado alvo, alterar os parâmetros dos algoritmos de fusão de dados multisensores e detectar falhas de equipamentos.

Nível 5 – refinamento cognitivo – introduz a avaliação humana on-line de especialistas ou operadores do sistema, no sentido de orientar a interpretação do contexto e a produção de conclusões para o processo de tomada de decisão.

Os processos desses diversos níveis lidam com dados que são oriundos de sensores, de informações produzidas por processos de níveis inferiores, de informações armazenadas em bancos de dados e de dados de tutoria introduzidos pelos operadores dos sistemas. Além disso, as informações produzidas pelos sistemas são, em alguma instância, apresentadas aos operadores dos sistemas através de uma Interface Homem-Máquina (IHM).

Todos esses módulos são representados no modelo JDL, que é apresentado na figura 1.

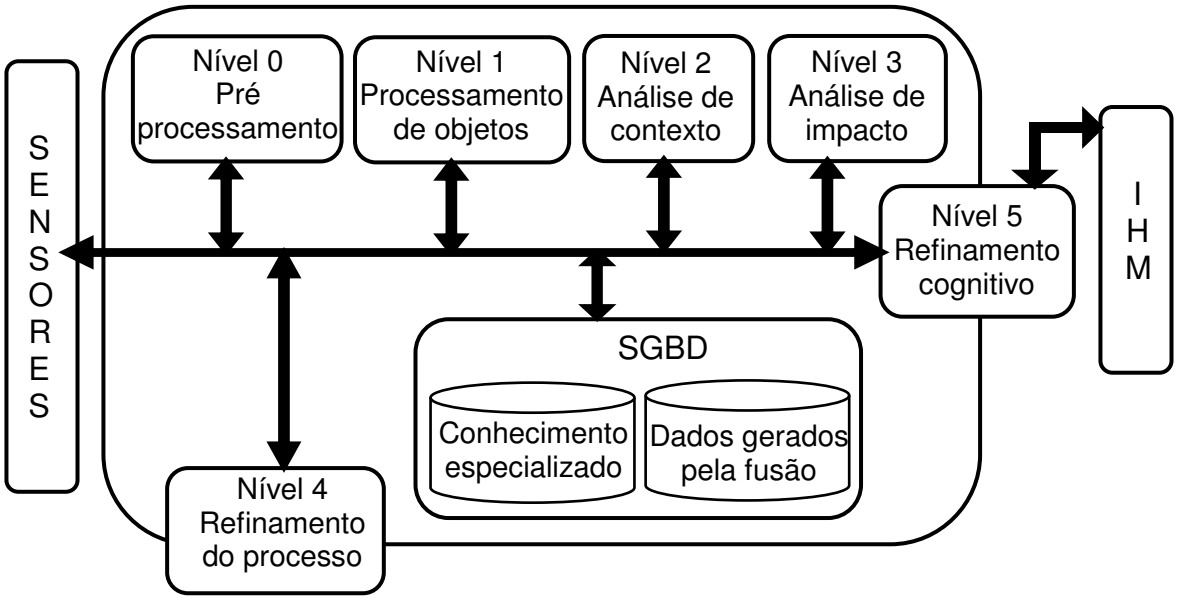


Figura 1 – Modelo JDL para sistemas de fusão de alvos

No item que se segue, são analisados de forma pormenorizada, os níveis do modelo JDL, os objetivos dos sistemas de cada nível e as técnicas empregadas.

2.4 OS NÍVEIS DO MODELO JDL

2.4.1 NÍVEL ZERO DO MODELO JDL

O nível de pré-processamento dos dados é responsável pelo interfacemento com os sensores, a digitalização dos dados, eventuais transformações entre os espaços de freqüência e de tempo e toda a transformação necessária nos dados coletados para que possam ser utilizados para serem correlacionados a possíveis alvos.

A natureza, a precisão, a qualidade e a quantidade dos dados podem ser bastante variadas, já que os sensores podem ser dos mais diferentes tipos. Mesmo para sensores de um mesmo tipo, radares por exemplo, pode haver bastante variação destes parâmetros. Como também se pode estar lidando com dados de sonares, sensores de infravermelho, dados recebidos por rádio, hodômetros, anemômetros, dados de equipamentos de *Global Positional System* (GPS) e equipamentos de

interrogação de alvos (*Interrogation Friend-Foe-Neutral* – IFFN), é necessário acessar as fontes de dados, realizar a leitura e conversão da informação para uma forma utilizável e, freqüentemente, realizar algum processamento de limpeza do sinal, conhecido como filtragem de ruído.

Em um sinal de radar, por exemplo, a filtragem visa reduzir a probabilidade de ser detectado um ruído (*clutter*) proveniente de um componente ambiental, como uma nuvem, uma onda do mar, um acidente geográfico ou qualquer outro eco que não corresponda a um alvo de real interesse. A filtragem visa conseguir detectar um alvo em um ambiente ruidoso, não estacionário, mantendo constante a probabilidade de ocorrência de falso alarme.

O sinal de vídeo radar possui um nível de ruído branco, acima do qual ocorrem os ecos relativos a alvos existentes. No caso de haver a presença de falsas detecções provenientes de retornos de ondas, nuvens e outras perturbações ambientais (que são genericamente conhecidas como *clutter*), o nível de ruído se eleva podendo causar a detecção de um alvo que não existe (falso alarme), diminuindo a relação sinal/ruído e dificultando a detecção de alvos reais. Uma técnica bastante conhecida e aplicada para este fim é o CFAR [13], que visa adaptar o limiar do sinal de vídeo acima do qual considera-se que existe um eco relativo a um alvo.

Outro objetivo, ainda no nível 0, é a extração de características do alvo, tais como a posição e o tamanho. Outras características, como rumo, velocidade, aceleração e identificação do alvo ficam a cargo do Nível 1 do Modelo JDL.

2.4.2 NÍVEL UM DO MODELO JDL

Neste nível os sistemas procuram integrar os dados oriundos de diferentes sensores, relativos a diferentes alvos em um quadro integrado que resuma as características tais como posição, velocidade, rumo, aceleração e identificação dos diversos alvos presentes no cenário tático. Esta é uma tarefa bastante complexa, sobre a qual versam a maioria dos sistemas de fusão de alvos existentes.

Neste nível, de acordo com a arquitetura adotada para a integração dos sensores, várias tarefas de monta precisam ser executadas.

Primeiramente é necessário associar as diferentes medidas (associação medida-medida), ou seja, em um ambiente dinâmico em que os objetos estão em movimento, apesar da filtragem do pré-processamento ainda pode haver a presença de medidas espúrias, sendo necessário associar as medidas realizadas em um determinado instante com aquelas realizadas no instante anterior pelo mesmo sensor.

Da capacidade de realizar corretamente esta associação das medidas aos alvos reais, depende a possibilidade de caracterização desses alvos, que passam posteriormente a serem chamados genericamente de “acompanhamentos”.

É também necessário associar medidas com acompanhamentos, isto é, a chegada de várias medidas em um determinado instante, obriga a associar cada uma delas, ou uma parte delas, a cada um dos acompanhamentos que já estejam identificados, de forma a permitir a atualização dinâmica das características dos acompanhamentos.

Além disso, é ainda necessário associar acompanhamentos com outros acompanhamentos, isto é, fundir as representações de acompanhamentos geradas individualmente pelos diversos sensores disponíveis, que são produzidas em diferentes freqüências, com diferentes precisões e com diferentes características, em um único acompanhamento de maior confiabilidade.

As funções do nível 1 podem ser particionadas em quatro grandes sub-tarefas, de acordo com [9]: alinhamento dos dados, correlação de objetos, estimativa de características e identificação do alvo. A figura 2 ilustra estas sub-tarefas do nível 1 e os parágrafos que se seguem detalham as sub-tarefas.

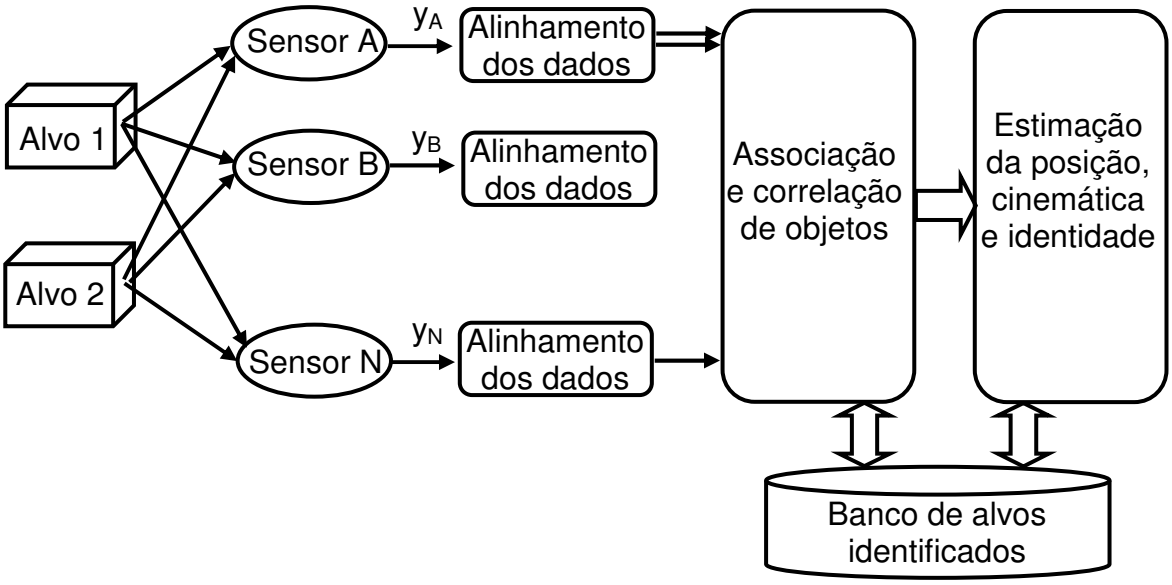


Figura 2 – Sub-tarefas do nível 1 no Modelo JDL para fusão de alvos

O alinhamento dos dados é necessário graças à característica assíncrona dos diferentes sensores que, adicionalmente, podem ainda estar baseados em localidades diversas e utilizar distintos sistemas de referência posicional. Assim, antes que se

possa fundir informações, é necessário trazê-las para um mesmo padrão espaço-temporal. Assim, as informações sobre azimute, elevação e distância produzidas por diferentes sensores, devem ser mapeadas em um sistema de coordenadas comuns aos diversos sensores. Da mesma forma, o instante em que as informações são recebidas e os instantes em que elas foram efetivamente lidas nos diferentes sensores não coincidem e todas as informações devem ser atualizadas para uma mesma base de tempo.

A etapa de associação e correlação de objetos é necessária para que se possa associar as medidas fornecidas por cada sensor com os alvos efetivamente existentes. Um sensor pode produzir várias detecções de alvos durante um período de tempo e estes podem se referir a um único objeto real (um navio ou um avião, por exemplo). Os métodos existentes para definir essas associações procuram estabelecer alguma métrica que quantifique a distância medida-medida ou medida-acompanhamento. Há métricas de distância vetorial, métricas baseadas em coeficientes de correlação ou outras medidas probabilísticas de similaridade. A idéia é medir o grau de relacionamento entre os pares, de forma a determinar se a iésima observação do sensor A na figura 2, está relacionada à jésima observação do sensor B. Ou seja, deseja-se investigar a associação entre y_{Ai} e y_{Bj}.

Primeiramente, é necessário determinar se os sensores A e B lidam com tipos de dados diferentes e, caso necessário, fazer as transformações para determinar a proximidade física de y_{Ai} e y_{Bj}. A seguir, como as observações y_{Ai} e y_{Bj} podem ter sido feitas em tempos distintos, isto é, $t_i \neq t_j$, elas precisam ser sincronizadas, aplicando-se equações de movimento que calculem a posição em um instante subseqüente ao da observação, o que deve levar em consideração tanto as características do sensor quanto a dinâmica do objeto sob observação.

Outra abordagem possível é procurar relacionar as medidas y_{Ai} com o késimo acompanhamentos já existente, x_k. Para tanto o acompanhamento conhecido em um tempo t, ou seja, $x_k(t)$, deve ser extrapolado, com o uso de modelos dinâmicos, para a posição no tempo t_i, de forma a poder ser associado à medida y_{Ai}. A partir disso, é possível utilizar um modelo de previsão que forneça uma medida prevista para este acompanhamento, ou seja, uma y_A prevista que pode ser dada por:

$$y_{A\ \ prevista}(t_i) = g[x_k(t_i)] + r$$

Onde g é uma função que inclui várias transformações de coordenadas e r é um ruído desconhecido.

A partir daí, a medida de associação vai comparar y_A real com y_A prevista. Computada essa medida, é necessário utilizar alguma estratégia ou lógica para estabelecer se afinal duas observações se referem ao mesmo objeto. Geralmente são usadas técnicas de janelas para estabelecer fronteiras aceitáveis para a investigação. Essas janelas podem ser estabelecidas a partir de conhecimentos prévios sobre a cinemática dos objetos (velocidade máxima, por exemplo).

A próxima etapa ilustrada na figura 2 é a de estimação. Para estimação da posição e cinemática dos acompanhamentos, são utilizados algoritmos que assumem algum modelo para o ruído e para cinemática e mapeiam os acompanhamentos em um vetor de estados que inclui informações como posição e velocidade. O modelo do filtro de Kalman é o mais utilizado para este fim e fornece a estimativa para o vetor de estados mais adequada à medida observada, segundo um critério matemático proposto pelo modelo. O modelo realiza correções sucessivas na estimativa, a partir das diferenças entre o vetor de estados previsto e o vetor efetivamente observado, que é passado ao modelo pela etapa anterior. A tarefa das técnicas de estimação é determinar o valor de um vetor de estados x (posição e velocidade, por exemplo) que minimiza alguma relação de métrica entre o vetor estimado e o vetor de dados, conhecida como função objetivo. Assim, se a métrica usada for a soma do quadrado das diferenças, a medida escalar a ser minimizada é dada pela função objetivo L(x):

$$L(x) = \sum \left[y_i - y_{i\ previsto} \right]^2$$

Várias outras métricas podem ser usadas, tais como mínimos quadrados ponderados, máxima verossimilhança e funções Bayesianas. No caso do filtro de Kalman, por exemplo, a métrica é minimizar a matriz de covariâncias do erro de estimação a posteriori.

Realizar a estimativa é recalcular o vetor x que melhor se adequa à função objetivo especificada, o que significa processar as sucessivas estimativas seqüencialmente. Este é o papel desempenhado pelo filtro de Kalman e pelo filtro de Kalman estendido.

Os combinação dos processos de associação e estimação produzem dados fundidos que estimam a posição, a velocidade e outros atributos (como tamanho, por exemplo) dos acompanhamentos.

A tarefa de estimação da identificação envolve técnicas de reconhecimento de padrões. O reconhecimento de padrões pode ser feito a partir de características

extraídas dos sinais recebidos pelos sensores para um determinado acompanhamento. Os tratamentos que extraem as características principais dos sinais dos sensores são bastante dependentes da natureza do sensor, e podem incluir funções de amostragem, transformações do domínio do tempo para o da freqüência, processamento da imagem para determinação de bordas, dentre outras técnicas de processamento de sinais. Após isolar um vetor de características y, podem ser usadas técnicas de clusterização e classificação como redes neurais ou qualquer outro método estatístico, para classificar o vetor de características y em um grupo que permita identificar o acompanhamento como sendo, por exemplo, um míssil ou um navio da classe tal.

Da mesma forma que o problema de fusão de posição e de dados cinemáticos, o problema de fusão de características com o objetivo de identificação do acompanhamento também precisa lidar com o problema de associação de diferente medidas a um mesmo acompanhamento e, neste sentido, o problema de associação e correlação de objetos também afeta a identificação do acompanhamento.

O conceito de fusão de identificação pode ser conseguido através de diferentes modelos de arquitetura. Ela pode ser implementada no nível das medidas realizadas individualmente por cada sensor. Alternativamente pode ser realizada após as identificações individuais de cada sensor. Pode ainda ser realizada após a fusão posicional dos acompanhamentos oriundos dos vários sensores. De fato, a função de identificação pode ser feita separadamente ou de forma integrada com a fusão posicional.

Para realizar a fusão de identificação, as técnicas empregadas são de três grupos principais: modelos baseados em propriedades físicas dos sensores, técnicas de inferência baseadas em extração de características e modelos baseados em conhecimento simbolicamente representável.

Os modelos físicos estabelecem vetores de características como a seção reta do sinal de radar (Radar Cross-Section – RCS) ou o espectro principal de emissão do alvo para sensores de infravermelho ou ainda alguma medida que combine a freqüência e a intensidade de um sinal sonoro recebido pelo sonar, e estimam a identificação do alvo comparando as informações de um modelo de previsão (filtro de Kalman, por exemplo) com as medidas reais. As dificuldades dessa abordagem giram em torno da obtenção de um modelo correto das diferentes identificações possíveis, isto é, obter assinaturas das classes previamente, já que as condições efetivas de medidas podem variar muito em relação às condições em que as assinaturas foram

obtidas (o posicionamento dos alvos varia constantemente, os níveis de ruído são grandes e a proximidade de alvos causa interferência entre eles). Apesar das dificuldades, é possível estabelecer a identificação dos acompanhamentos com os mesmos modelos clássicos de estimação que são usados para a fusão posicional como mínimos quadrados, máxima verossimilhança, filtros de Kalman e métodos Bayesianos.

As técnicas baseadas em extração de características procuram estabelecer uma identificação a partir apenas de atributos como a forma, a temperatura, o tamanho, as emissões que possam ser características de algum tipo de objeto. Em [9] é feita uma divisão dessas técnicas em paramétricas e não paramétricas. As técnicas paramétricas pressupõem a existência de informações a priori, tal como distribuições estatísticas dos dados de identificação e podem ser técnicas de inferência Bayesiana, teoria de Dempster Shafer ou métodos de clusterização como K-means. As técnicas não paramétricas não utilizam conhecimento a priori. Podem ser usadas redes neurais, métodos de votação, figura de mérito e métodos baseados na entropia, por exemplo.

Os sistemas baseados em modelos cognitivos procuram mimetizar o comportamento humano na identificação de padrões. Utilizam-se técnicas como tabelas de decisão, sistemas especialistas ou lógica fuzzy. A abordagem proposta na aplicação de validação do modelo desta tese se encaixa nesta classificação de sistema cognitivo, como será explicado no capítulo 5.

2.4.3 NÍVEL DOIS DO MODELO JDL

Este nível utiliza as informações geradas no nível anterior para produzir inferências sobre possíveis relações entre os acompanhamentos identificados, ou entre estes e o ambiente, ou ainda a composição desses elementos no tempo ou no espaço em outros entes de maior nível hierárquico ou estratégico.

Uma das atividades deste nível é a agregação de objetos, ou seja, a composição de informações separadas fornecidas pelo nível 1 em um ente maior que faça parte do contexto. Assim, por exemplo, um avião detectado com identificação de caça inimigo, um sinal identificado de radar de direção de tiro e a existência de alvos amigos possíveis no raio de alcance do armamento supostamente portado pelo avião, sugerem a iminência de um engajamento por parte do inimigo. Podem ser detectados padrões de formaturas, de acordo com as posições relativas dos acompanhamentos ou estabelecidas cadeias de comando, de acordo com os padrões de comunicação detectados.

A segunda atividade importante neste nível é fazer a correspondência entre eventos e atividades. Algumas situações envolvem uma seqüência de atividades que podem ser interpretadas para se extrair informações sobre um comportamento futuro ou atitude de maior nível que esteja em curso ou vá ser desfechada.

Outra atividade possível é a análise de contexto, que busca a fusão de informações disponíveis sobre as condições climáticas e/ou geográficas com as informações sobre os acompanhamentos oriundos do nível anterior.

2.4.4 NÍVEL TRÊS DO MODELO JDL

Neste nível, o objetivo é fundir o cenário corrente com as projeções futuras possíveis para o contexto corrente. Isto envolve fazer previsões das possíveis movimentações do inimigo, de suas posições futuras, das suas atitudes, das conseqüências para as forças amigas de suas atitudes, das falhas que podem ocorrer nas forças amigas e das atitudes preventivas e corretivas que possam ser tomadas.

Uma tarefa possível é a identificação de oportunidades. Por exemplo, o sucesso de um ataque a um alvo inimigo pode depender das condições de revide disponíveis por parte desse inimigo. Isso pode depender da configuração corrente em termos de potencialidades (armas, raio de operação, características de sensores disponíveis) e das condições geo-climáticas envolvidas. Identificar oportunidades futuras de maior sucesso dessas ações, de acordo com as possíveis evoluções do cenário, é uma das atitudes que podem ser tomadas nesse nível.

Outra tarefa é a estimativa de conseqüências. Ainda a partir do cenário atual, que envolve tanto as forças disponíveis de ataque e defesa quanto as condições geo-climáticas, é possível estabelecer hipóteses de perdas, tanto amigas quanto inimigas, que podem ocorrer a partir da tomada de determinadas atitudes, novamente tanto amigas quanto inimigas.

Para o tipo previsto de atividades para este nível, podem ser usados modelos baseados em algoritmos de clusterização, árvores de decisão, redes neurais, sistemas especialistas e agentes inteligentes, dentre outros.

2.4.5 NÍVEL QUATRO DO MODELO JDL

Como já foi citado, este é um nível de meta processo. Ou seja, neste nível devem ser implantados processos de otimização e controle dos algoritmos dos demais níveis. Isto pode ser conseguido refinando os algoritmos (redefinindo parâmetros dos modelos, por exemplo), ou redirecionando / redefinindo o uso dos sensores em um

determinado cenário (determinando quando se deve ou não usar um sensor ou indicando sob quais condições operativas ele deve ser usado para maximizar os resultados obtidos). Assim, a tarefa deste nível é observar os sistemas como um todo, procurando sintonizar todo o aparato disponível no sentido de obter os melhores resultados dentro de uma determinada estratégia. Isto é bastante complexo pois é necessário levar em consideração materiais remotamente localizados (sensores, aviões, navios, tropas, etc), doutrinas operativas vigentes e escolhas humanas dos operadores do sistema.

Os algoritmos usados nesse nível devem ser capazes de levar em consideração tanto os aspectos locais dos modelos usados em cada nível anterior, quanto os aspectos estratégicos e doutrinários mais gerais, de forma que técnicas como agentes inteligentes, sistemas baseados em conhecimento simbólico, técnicas de otimização de parâmetros, dentre outras, podem ser apropriadas para os sistemas deste nível.

2.4.6 NÍVEL CINCO DO MODELO JDL

Este nível não existia no modelo original do JDL e foi acrescentado em 2001, após alguns pesquisadores [14] apontarem a necessidade de se introduzir um nível final de tratamento (pós-tratamento da informação) antes da apresentação das informações na interface Homem-Máquina do sistema, da mesma forma que existe um pré-tratamento no nível 0.

O argumento é que há necessidade de incluir um pós tratamento das informações produzidas pelos sistemas que explore novas características do interfaceamento com as pessoas, contemplando aspectos como sensores de movimento corporal, interpretadores de linguagem natural, uso intensivo de recursos visuais, auditivos e tácteis e adaptação dinâmica da interface homem-máquina ao viés cognitivo do operador (por exemplo, tendência de determinados operadores de valorizar mais aspectos positivos da situação tática ou de focar a atenção em determinadas áreas, que pode ser compensada pela IHM reforçando os recursos para focar a atenção em alguma área que necessite de atenção imediata).

3 SISTEMAS BASEADOS EM CONHECIMENTO

Aplicações que necessitam de técnicas que utilizem conhecimentos simbolicamente representados (KBS do acrônimo em inglês) geralmente possuem um domínio extenso e complexo, dificultando a integração do modelo de comportamento com a descrição do domínio.

As técnicas modernas de modelagem do domínio utilizam-se de linguagens que permitem descrever a hierarquia e as relações entre as classes do domínio, tal como a linguagem UML. Para descrições mais completas do domínio, que incluam restrições às relações entre as classes, como cardinalidade e transitividade, utilizam-se linguagens descritoras de ontologias (tais como OWL, OIL, etc.). Estas linguagens são particularmente úteis para aplicações nas quais o domínio é bastante aberto, como as aplicações que lidam com documentos de texto em um ambiente de rede, por exemplo.

As linguagens ontológicas em geral possuem motores que realizam inferências automaticamente a partir da hierarquia das classes e das restrições impostas às relações entre as classes. Tais inferências são realizadas a partir de regras automaticamente deduzidas a partir da modelagem e são realizadas através de alguma linguagem de consulta, que em geral podem ser vistas como extensões das linguagens de consulta a bancos de dados (SQL).

Entretanto, embora tanto a UML quanto linguagens ontológicas possuam ferramentas para descrição da hierarquia do domínio, um problema é que elas não permitem uma boa aderência entre essas técnicas e a representação do conhecimento em forma de regras. Em aplicações de domínio complexo, o raciocínio não se restringe aos objetos que possuem uma relação hierárquica. Nestes casos, é sempre útil [15], mesmo em linguagens ontológicas, possuir um mecanismo que permita realizar inferências a partir de regras de produção que contenham as regras de negócio.

Um segundo problema importante no paradigma de sistemas baseados em conhecimento é a capacidade de lidar com valores que estejam em constante mudança, que é uma característica dos sistemas de tempo real. Tais características podem ser encontradas, por exemplo, em um sistema militar de controle tático, em um sistema de controle de uma planta industrial ou em um sistema de controle e monitoração de pacientes em um ambiente hospitalar. Nestas situações, o conhecimento utilizado nas regras que expressam a dinâmica comportamental está

em constante alteração, implicando em reavaliação também constante das conclusões expressas nas regras.

Um terceiro problema refere-se à expressividade do conhecimento expresso nas regras. Um fator que contribui para melhorar a expressividade do conhecimento representado por regras e melhorar a cobertura do domínio da aplicação, diminuindo assim a quantidade de regras de representação do conhecimento, é a utilização de variáveis fuzzy nas regras. A modelagem Fuzzy permite particionar de forma mais intuitiva o domínio da aplicação e expressar de forma mais concisa o conhecimento dos especialistas. Outro fator que aumenta a expressividade, principalmente na modelagem do comportamento dinâmico dos sistemas, é o uso da dimensão temporal. Alguns comportamentos somente podem ser identificados se forem considerados os eventos relacionados ao tempo em que eles ocorreram, particularmente quando o comportamento a ser considerado é de natureza evolutiva [16]. Assim, em um sistema médico, por exemplo, a evolução do quadro clínico de um paciente para efeito de diagnóstico e de proposição de tratamentos está relacionada a uma seqüência de evidências e ocorrências temporalmente relacionadas que podem caracterizar esta ou aquela patologia específica e recomendar este ou aquele procedimento terapêutico [16]. Em sistemas de controle tático de um cenário militar também ocorrem comportamentos que podem ser detectados e contra medidas que podem ser sugeridas, que estão relacionados a seqüências temporais de eventos.

3.1 PARADIGMAS DECLARATIVO E PROCEDIMENTAL

3.1.1 SISTEMAS DE PRODUÇÃO

Sistemas de produção são uma forma de representar conhecimento declarativo e assim modelar o raciocínio humano. Nesses sistemas os conhecimentos são expressos como regras de produção que representam conhecimentos heurísticos ou especializados sobre assuntos específicos. Na primeira parte das regras aparecem as condições, também chamada parte do “se” da regra, ou corpo da regra, ou lado esquerdo da regra ou LHS (*left-hand size*). Na segunda parte das regras estão as ações, também chamada de parte do “então”, ou cabeça da regra, ou lado direito da regra ou RHS (*right-hand size*).

Um sistema de produção possui um conjunto de regras (base de conhecimentos), um conjunto de fatos (também chamado de memória de trabalho) e um algoritmo chamado motor de inferências que produz novos fatos a partir de fatos existentes.

Regras da base de conhecimentos podem ser disparadas quando suas condições são satisfeitas por fatos da memória de trabalho (esse processo é chamado unificação). As regras que estão prontas para serem disparadas formam um conjunto de conflito. Os sistemas de produção em geral estabelecem uma política para escolher a ordem em que as regras prontas serão disparadas, ou seja, uma estratégia de resolução de conflitos para determinar a próxima regra a ser disparada. As regras são então disparadas, o que pode gerar novos fatos e, portanto, gerar mais regras prontas. Esse processo continua até que não haja mais regras prontas, ou seja, até que o conjunto de conflito esteja vazio.

Essa abordagem é chamada de orientada a dados ou *forward chain*, na medida em que gera uma seqüência de fatos não necessariamente objetivados por quem disparou o processo. Outra abordagem é a orientada a objetivos ou *backward chain*, na qual as regras são examinadas a partir das conclusões, de acordo com uma lista de objetivos a serem descobertos. Essa lista de objetivos pode crescer a partir dos objetivos iniciais, caso regras de interesse possuam premissas que não podem ser avaliadas pelo desconhecimento de determinadas variáveis (que então viram os novos objetivos).

3.1.2 SISTEMAS ORIENTADOS A OBJETOS

Em outra ponta do desenvolvimento de aplicações estão os sistemas baseados nas técnicas de análise e modelagem de software mais modernas. Dentre as propostas que a engenharia de software criou, a mais largamente utilizada para aplicações de grande porte ou grande complexidade é a que envolve técnicas de análise, projeto e programação orientados a objetos, o que é genericamente chamado de Orientação a Objetos (OO). Os principais princípios que a OO propõe, são:

- Abstração - É a capacidade de criar um modelo de dados e comportamentos desejáveis para uma classe de objetos. O modelo (chamado Classe) define os dados e comportamentos que serão oferecidos como serviços a outras classes e aqueles que serão apenas para uso interno dos objetos da classe. A abstração é a visão que os demais objetos têm de um determinado objeto, separando o seu comportamento essencial de sua implementação. Os objetos são instâncias das classes que possuem as características definidas no modelo. Na nomenclatura adotada uma classe é uma abstração que contém diversos atributos e pode responder a mensagens ou métodos, que são operações que podem ser realizadas pelas instâncias da classe.

- Encapsulamento - É o processo de criar interface para os serviços providos por um objeto, permitindo que os clientes possam requisitar serviços ignorando a forma de implementação dos mesmos e, ao mesmo tempo, impedindo que as informações internas do objeto sejam acessadas ou alteradas livremente pelos clientes.
- Hierarquia – é a capacidade de estabelecer ordenações de abstrações. A relação de herança é um exemplo de hierarquia. Uma classe Empregado herda todas as Características da Classe Pessoa e acrescenta mais algumas características particulares dessa classe. A agregação e a composição também são exemplos de relações hierárquicas.
- Modularidade – é a capacidade dividir o domínio de um problema em módulos que possuem relações entre si, mas sejam coesos e tenham baixo acoplamento. Na boa modularização que é conseguida com o correto uso do encapsulamento e das relações hierárquicas é que reside o segredo de criar módulos reusáveis entre projetos.
- Polimorfismo – é a capacidade de executar serviços que possuem a mesma interface (assinatura) em classes ligadas por uma estrutura de herança, mas que possuem comportamentos distintos, especializados para cada uma das classes. A decisão sobre qual forma será selecionada é tomada em tempo de execução, quando se conhece efetivamente a classe do objeto.

3.1.3 COMPATIBILIZANDO OS PARADIGMAS

Vários modelos têm sido propostos para tratar o problema de inclusão de regras de produção em um sistema que utilize modelagem procedimental [17,18]. Algumas linguagens para especificação de ontologias, tais como OCML [19] e LOOM [20] permitem a inclusão de regras de produção (comparações pormenorizadas destas e de outras linguagens de especificação de ontologias são feitas em [17, 21]). A vantagem dessas e de outras linguagens de especificação de ontologias como Ontoligua [22], OKBC [23] e FLogic [24], é que permitem a especificação de ontologias completas. Essas ontologias incluem tanto conceitos que são comumente encontrados em linguagens de modelagem tipo UML, tais como classes, herança, polimorfismo, instanciação, tipagem forte de atributos e encapsulamento, bem como conceitos que não são encontrados em UML, como facetas de propriedades (transitividade e cardinalidade), ou ainda operações sobre conjuntos como união, interseção e complemento (uma comparação extensa sobre correspondências entre características de UML e linguagens ontológicas pode ser encontrada em [25]). Entretanto as

linguagens ontológicas são muito utilizadas em domínios abertos, mas não possuem largo emprego em aplicações embutidas em ambientes de domínio limitado e que requeiram características de tempo real, ambientes nos quais as linguagens orientadas a objetos, como C++ e Java são dominantes.

Outra abordagem para acoplar a modelagem comportamental à abordagem procedimental é feita pelos *frameworks* que permitem a especificação de regras de produção em arquiteturas orientadas a objetos implementadas em alguma linguagem OO (SmallTalk, C++ ou Java, por exemplo). Para esse tipo de ambiente por vezes é utilizado o termo *Embedded Object Oriented Production Systems* (EOOPS). A motivação dessas abordagens é juntar as vantagens dos sistemas de conhecimento declarativo com a estruturação sistemática, modular e hierárquica proporcionada pelas linguagens orientadas a objetos.

Os sistemas mais tradicionais de *shells* de sistemas especialistas, tais como Snark [26], implementam uma base de fatos e uma base de regras claramente separados, sobre os quais é possível realizar inferências típicas da Lógica Proposicional, ou seja não é possível lidar com fatos que sejam de um mesmo "tipo".

Em OPS5 [27] aparece o conceito de instancias de classes de fatos. Os fatos têm tipos que são declarados antes das regras com os respectivos campos (equivalentes a atributos). Instancias desses tipos são criadas na base de fatos por meio de comandos específicos. As regras são compostas por condições que são basicamente comparações com atributos dos fatos da base de fatos e por ações que são seqüências de alterações da base de fatos. OPS5 trabalha com o conceito de quantificadores ($\forall$ e $\exists$). Por exemplo, se um tipo pessoa possui um campo nome e existem várias instâncias de pessoa na base de fatos, então para procurar por uma pessoa de nome "José" deve ser criado um tipo objetivo que tenha também um campo nome e a regra deve comparar esses campos para que o motor de inferências ache a instância correta na base de fatos. Assim, OP5 lida com uma lógica de predicados (ou lógica de primeira ordem), em que são consideradas instâncias de fatos. A unificação de regras e fatos é feita com o algoritmo Rete [28], que é utilizado por quase todos os motores de inferência para lógica e predicados e que explicaremos em detalhes na seção 4.4.1. Entretanto, as comparações de campos são feitas apenas através de valores e não existe o conceito de referência para instâncias, como nas linguagens orientadas a objetos.

CLIPS (*C Language Integrated Production System*) [29,30] foi uma linguagem patrocinada pela NASA, que possui uma sintaxe parecida com OPS5, mas mistura

conhecimento declarativo com funções usadas para conhecimento procedimental. CLIPS pode ser usado com as linguagens C, C++ e ADA, mas também define uma linguagem OO própria, chamada COOL, que pode ser usada para definir todo o domínio de uma aplicação. É possível definir estruturas de dados (como as *structs* de C) ou classes mesmo. Os atributos de uma classe podem ser fatos e novos fatos devem ser incluídos explicitamente na base de fatos com um comando do tipo *assert* (como em OPS5). Objetos e fatos compartilham a memória de trabalho, mas apenas os objetos possuem referências (ou seja, podem ser referenciados por um nome). Entretanto as regras podem fazer referências a fatos e a objetos. CLIPS teve bastante aceitação, mas acabou desatualizado, por algumas razões. Primeiramente, propõe uma sintaxe complexa (muito mais próxima de Lisp do que de C, como se poderia esperar, devido ao nome da linguagem). O segundo problema é que os objetos do domínio em COOL são separados dos objetos do sistema, embora possa haver ligação entre eles com algum esforço de programação. Um terceiro problema é que COOL não sofreu atualizações e linguagens mais populares tais como Java se desenvolveram e dominaram o mercado OO.

NéOpus [31] é um sistema de produção integrado com *Smalltalk*. A abordagem é de integração entre a linguagem nativa e o sistema de regras de produção. A parte das condições das regras pode usar todos os operadores da linguagem e os objetos definidos na linguagem podem ser usados nas regras e serão unificados com a base de fatos. As premissas são expressões booleanas em Smalltalk que podem também invocar métodos dos objetos. NéOpus possui um pré-compilador que compila a sintaxe das regras para a linguagem *Smalltalk*. A aplicação do domínio deve notificar o motor de inferências quando um objeto for modificado externamente. Cada regra deve conter uma lista de todas as variáveis que se modificadas devem causar a reavaliação da premissa da regra. Apesar de representar um avanço em termos de integração com a abordagem OO, NéOpus sofre de alguns dos males de CLIPS. Principalmente porque *Smalltalk* sofre dos mesmos problemas de COOL e também não evoluiu como outras linguagens OO.

JESS [32] é uma *Expert System Shell* (ESS) para a plataforma Java que estende as funcionalidades e a sintaxe de CLIPS. Em JESS é possível tanto definir regras em uma sintaxe proprietária (paradigma declarativo) quanto especificar código Java a ser executado juntamente com as regras (paradigma procedimental). O motor de inferências é orientado a dados e implementa uma versão do algoritmo RETE. Assim como CLIPS, JESS define uma linguagem para a criação das aplicações, e não apenas das regras. Comandos de criação de interfaces gráficas, comunicação, ou

entrada/saída são disponibilizados nesta linguagem, enfim, toda uma aplicação pode ser desenvolvida dentro do *shell* de JESS. Desta forma, é possível construir Java *servlets*, EJBs e *applets* combinados com conhecimento declarativo expresso em regras. JESS utiliza a estratégia de interpretação de regras e a sintaxe utilizada na definição das regras não é semelhante a Java, aumentado o esforço de compreensão do usuário, tipicamente um programador Java. A eficiência de JESS é prejudicada pelo fato de ser um ambiente puramente interpretado. Além disso, a separação entre os objetos de JESS e os de Java. Existe uma característica de JESS que é distinta das ferramentas anteriores e que também é utilizada (embora de forma um pouco diferente) no *framework* que foi implementado neste trabalho e que será descrito em detalhes no capítulo 4. Esta característica diz respeito à forma de atualização de valores na base de fatos. Em JESS não existe nenhum comando explícito para a aplicação do domínio avisar ao sistema que comanda a parte declarativa quando um objeto foi modificado (como acontece com NéOpus e outras ferramentas do tipo, até mais modernas). Assim, os objetos devem ser capazes de notificar quando uma de suas propriedades foi alterada e isto pode ser utilizado para verificar a condição das regras.

Na década corrente, vários *frameworks* com abordagem EOOPS foram desenvolvidos especificamente para acoplamento com Java. Em geral esses *frameworks* seguem o padrão JSR-94 [33]. Esse padrão foi desenvolvido pela comunidade Java e adotado como a interface de aplicação em Java a ser usada como referência para o desenvolvimento de motores de inferência para regras de produção. As operações padronizadas são divididas em administrativas e executivas. As operações administrativas permitem carregar as regras de um arquivo e registrá-las em um domínio ou conjunto de regras específico. As operações executivas permitem manipular um domínio previamente carregado, inserindo, removendo ou modificando a base de conhecimento e manipulando a base de regras, ou seja, executando as regras e realizando consultas.

O padrão não define uma sintaxe para a linguagem na qual as regras serão expressas, ou seja, é deixada a cargo das implementações que adotem o modelo a tarefa de definir um formato para o arquivo e uma sintaxe para a linguagem de construção das regras. No JSR-94 são previstos dois cenários de uso para as regras: stateless e statefull. No cenário stateless, não existe uma base de conhecimento permanente e podem ser realizadas consultas à base de regras passando-se fatos que serão usados para as inferências e não serão guardados, perdendo-se logo após a consulta. No cenário *statefull*, é mantida uma base de conhecimentos permanente e

as inferências são realizadas a partir de consultas a um conjunto de regras e à base conhecimentos. Nesse cenário são oferecidos comandos para atualização da base de conhecimentos permanentes.

Algumas das mais conhecidas implementações de *frameworks* para motores de inferência de regras são comentadas a seguir.

JRuleEngine [34] é um motor de inferência para Java que utiliza o padrão JSR-94 e uma sintaxe proprietária para definir as regras. Os fatos são definidos como variáveis às quais o usuário associa um nome e a ferramenta associa um valor internamente na base de conhecimentos. As regras, que expressam lógica de primeira ordem, possuem um nome, uma descrição e uma lista de variáveis-operando que só podem ser encadeadas através do conectivo lógico "E". Podem ser usados os operadores relacionais tradicionais e mais alguns definidos pelo pacote, que expressam os conceitos de *pertencer* a um conjunto ou de *existir* na base de conhecimentos. O acoplamento entre as variáveis do domínio e os fatos da base de conhecimento deve ser feito pelo usuário, no domínio da aplicação, através de comandos oferecidos pelo pacote. Para realizar as inferências o usuário deve realizar uma consulta ao banco de regras previamente montado, passando os fatos a serem considerados (para consultas *stateless*) ou utilizando a base de fatos corrente (para consultas *statefull*).

JEOPS [35] é também um motor de inferências de código aberto que suporta regras de negócio usando lógica de primeira ordem em aplicações Java servidoras, clientes ou Servlets.

Mandarax [36] é uma biblioteca de classes Java de código aberto para inferências de lógica de primeira ordem. Diferentemente da maioria dos demais *frameworks* a estratégia de buscas é orientada a objetivos (*backward chain*). As regras são importadas no formato RuleML e em outros formatos padrão. O projeto foi dividido em dois subprojetos: PROVA e TAKE. PROVA dá suporte a construção e inferência de regras, leitura de regras de outras bases de regras, bancos de dados e semântica web. TAKE adota uma abordagem em que o motor de inferências é um interpretador, como originalmente ocorre no Mandarax.

OpenRules [37] é um ambiente para desenvolvimento e integração de árvores de decisão e bases de regras em aplicações web. As regras são escritas em tabelas de planilhas *Excel*, *OpenOffice* ou *Google Spreadsheets* e a ferramenta funciona como um *plug-in* para o Eclipse IDE, na medida em que os autores supõem que as planilhas são um software de amplo domínio em ambientes corporativos e, portanto,

de fácil domínio pelos especialistas que elaboram as regras. O motor de inferências é orientado a dados (*forward chain*).

SweetRules [18] é um conjunto de ferramentas para escrever e interpretar regras para um ambiente Web, usando os padrões *Rule Markup Modeling Language* (RuleML) e Semantic Web Rule Language (SWRL). A lógica suportada pelo SweetRules é a Courteous Logic, incluindo um mecanismo de priorização de regras para resolver conflitos. Uma característica relevante é a capacidade de interoperabilidade com outros mecanismos de especificação de regras e ontologias, tais como *XSB Prolog*, *Jess Production Rules*, *Jena-2* da HP e *CommonRules* da IBM. O motor de inferências implementa tanto a estratégia orientada a dados quanto a orientada a objetivos.

Hammurapi [38] é um motor de inferências também compatível com o padrão JSR-94, que suporta as estratégias orientada a dados e orientada a objetivos. As regras são escritas em Java.

Drools [39] é um *framework* que incorpora um motor de inferências que utiliza uma versão adaptada do algoritmo RETE [28] da estratégia de busca orientada a dados (*forward chain*). O objetivo principal da ferramenta é oferecer uma forma mais natural de expressar regras de negócio em uma arquitetura orientada a objetos. O *framework* propõe uma linguagem de sintaxe declarativa exclusivamente para expressar as regras que é suficientemente flexível para permitir a referências à semântica do domínio do problema. Assim, tanto a premissa das regras quanto a conclusão das mesmas podem fazer referência aos objetos do domínio da aplicação. A interface do pacote oferece formas para o usuário explicitamente incorporar novos fatos à base de conhecimento, alterar esses dados ou excluí-los. As regras são expressas utilizando operações de lógica de primeira ordem. Uma extensão recente, o DROLS Fusion, trabalha também com lógica temporal, mas não com lógica fuzzy. A ferramenta também atende ao proposto no padrão JSR-94.

Apesar do padrão JSR-94 não indicar um formato específico para a linguagem de especificação das regras, vários padrões estão propostos para o armazenamento e intercâmbio de conhecimento em forma de arquivos. Muitos se baseiam em extensões de XML e alguns incorporam capacidade de expressar ontologias. Esses padrões são utilizados em maior ou menor grau pelas ferramentas anteriormente citadas. Alguns dos padrões mais conhecidos são: *Production Rule Representation* (PRR) proposto pela OMG, *Rule Markup Language* (RuleML), *Semantic Web Rule Language* (SWRL), *DARPA Agent Markup Language* (DAML) e RDF/XML.

De uma forma geral, as ferramentas comentadas nesta seção buscam o acoplamento dos paradigmas declarativo e procedimental. Como será visto em detalhes no capítulo 4, este trabalho propõe outra abordagem, na qual as regras não são vistas como algo externo ao domínio do problema e que deve ser compatibilizado com a técnica de modelagem do restante da aplicação que é desenvolvida usando orientação a objetos. De fato as regras podem ser modeladas como parte do domínio através de classes de relacionamento entre objetos, que relacionam aspectos (atributos) de objetos de uma determinada classe com atributos de objetos de outra classe. Isto pode ser implementado respeitando uma separação entre os conhecimentos declarativos e procedimentais. As classes de regras podem ter uma sintaxe simples o suficiente para não exigir conhecimento profundo sobre o restante da modelagem nem exigir uma sintaxe proprietária. A sintaxe proposta para escrita das regras é a da própria linguagem de programação (Java por exemplo), utilizando operadores lógicos que são funções oferecidas pelo *framework*. Ademais, o *framework* estende o raciocínio expresso nas regras de forma a incorporar outras lógicas, como a Lógica Fuzzy e a Lógica Temporal, como será visto no capítulo 4.

3.1.4 DIFICULDADES DA INTEGRAÇÃO DOS PARADIGMAS

O motor de inferências típico de uma aplicação executa um ciclo com as seguintes operações:

- Recebe as alterações na base de fatos, oriundas da aplicação do domínio ou da execução de regras da base de conhecimentos;
- Executa uma operação de unificação entre a base de fatos e a base de conhecimentos para verificar se existem fatos que tornam premissas de regras verdadeiras;
- Adiciona as regras prontas a serem disparadas em um conjunto de conflito; e,
- Seleciona por algum critério de prioridade a ordem de execução das regras.

Quando se usa uma linguagem OO no domínio e se pretende integrar com o sistema declarativo, surge o problema detecção da alteração de valores dos atributos dos objetos do domínio. Uma abordagem força-bruta procura a cada ciclo comparar todos os objetos com todas as regras, mas o custo é inviável para aplicações reais. Outra abordagem é a quebra do paradigma de encapsulamento para permitir que o motor de inferências observe os dados locais dos objetos, o que pode ser feito obrigando os objetos a implementarem uma interface pré-definida, mas que de qualquer forma contraria o modelo conceitual da OO. A maioria dos *frameworks*,

portanto, resolve esse problema forçando que a aplicação informe quais objetos foram modificados, de forma a incluí-los na base de fatos. Como foi relatado, JESS implementa uma forma de notificação automática por parte dos objetos, que é adotada também no *framework* deste trabalho.

Outro problema diz respeito mais diretamente à sintonia entre os dois paradigmas. Algumas vezes não é claro o que o desenvolvedor deve fazer em um paradigma e o que é melhor fazer no outro. Isto ocorre porque muitos problemas podem ser resolvidos tanto via regras de produção quanto por métodos diretos dos objetos. A forma de implementação dos sistemas influencia nessa escolha. Os sistemas criados como ambientes completos que possuem uma linguagem proprietária associada, tendem a ficar ultrapassados ou possuem menos recursos que as linguagens padrão de OO e desencorajam a utilização da estrutura de serviços oferecidos, encorajando, por outro lado a abordagem declarativa. Em sistemas que oferecem uma biblioteca de funções, a utilização da linguagem da nativa da aplicação é encorajada, aumentado a dúvida de qual abordagem utilizar para cada problema. Não existem receitas de bolo para resolver esse problema. De fato deve-se usar o bom senso e a experiência para deixar para as regras o conhecimento mais especializado, que lida com associações complexas entre entes do domínio ou que se deseja isolar de aspectos da implementação. Problemas mais característicos do domínio, tais como comportamentos internos dos objetos, interfacemento com o meio externo, acesso a serviços simples do domínio devem ser implementados pelos próprios objetos da aplicação e não através de regras. Na seção 4.2 é explicada a abordagem que foi utilizada neste framework e que trata as regras como classes de associação entre os objetos do domínio.

3.2 SISTEMAS DE TEMPO REAL

O segundo desafio importante abordado neste trabalho é a compatibilização do paradigma declarativo com os sistemas de tempo real. Em todos os sistemas que buscam compatibilizar os paradigmas declarativo e procedimental, existem dois elementos fundamentais: os conhecimentos que são expressos através de regras e a base de dados conhecidos (também chamados fatos).

O que os sistemas descritos na seção anterior realizam é a manutenção de uma base de fatos que co-existe juntamente com os dados da aplicação modelada com técnicas de OO. Como foi visto, na maioria desses *frameworks* cabe à aplicação atualizar a base de fatos usada pelo mecanismo de inferência quando um novo fato externo tiver que ser considerado ou quando houver alguma mudança em um dado já

presente na base de fatos. Naturalmente que o acionamento de regras também gera novos fatos que são automaticamente adicionados à base de fatos. Essa separação é necessária porque todas essas ferramentas implementam versões de lógica de primeira ordem e devem realizar uma operação de unificação da base de fatos com a base de regras para permitir a existência dos quantificadores da lógica de primeira ordem.

Esses sistemas geralmente possuem duas abordagens para os dados considerados nas regras, que, aliás, fazem parte do padrão JR-94: a abordagem *statefull* e a abordagem *stateless*. Na abordagem *stateless* não há base de fatos e as consultas são realizadas à base de regras após a passagem de um determinado conjunto de fatos na própria consulta. Na abordagem *statefull* existe a base de fatos, o que implica que esses sistemas mantém uma replicação de dados provavelmente já modelados na aplicação em uma estrutura paralela, que é a base de fatos utilizada para as inferências.

Um fato é um conhecimento considerado como verdadeiro, que em vários sistemas pode ser visto com um objeto com valores de seus atributos, tal como:

Carlos é um homem solteiro de 25 anos de idade.

Como foi visto, quase todos os motores de inferência aplicam uma estratégia de busca orientada a dados (*forward chain*) para encontrar as regras que possam ser ativadas com os dados existentes na base de conhecimento e gerar novos fatos (ou verdades) que possam ser acionadas à base de conhecimento (alguns também usam a estratégia orientada a objetivos ou *backward chain*). Para todos os sistemas, caso haja uma mudança de fatos na base de conhecimento, é necessário que a aplicação que modela o domínio remova o fato que não é mais verdadeiro e adicione o novo fato. Assim, caso o estado civil de Carlos passe de solteiro para casado, o fato antigo deve ser removido e o novo fato deve ser incluído na base de dados:

Carlos é um homem casado de 25 anos de idade.

Entretanto, em sistemas de aquisição e tratamento de dados de tempo real, essa abordagem pode não ser adequada. Quando se deseja realizar inferências sobre medidas que variam de forma muito intensa, a remoção e inserção de fatos pode tornar os algoritmos de busca bastante custosos. Considere, por exemplo, que o fato a ser mantido fosse:

Carlos é um homem solteiro de 25 anos de idade com 75 batimentos cardíacos por minuto

Se em um sistema de monitoração de pacientes os batimentos cardíacos de cada paciente são fornecidos por sensores que continuamente adquirem informações vitais e deseja-se considerar estas informações para realizar inferências, então haverá uma grande quantidade de novos fatos constantemente sendo removidos e inseridos.

Em outra aplicação, se for considerado um sistema de monitoração tática que lide com centenas de objetos para os quais se monitora dezenas de informações e para o qual se deseja realizar inferências com essas informações, então haverá milhares de informações em constante alteração. Considerando que a eficiência da estratégia de busca é diretamente relacionada com a quantidade de informação na base de fatos e que a freqüência de alteração destes é elevada em sistemas de monitoração de tempo real, pode se tornar bastante onerosa a aplicação de qualquer estratégia de busca de regras na base de conhecimento.

A abordagem utilizada neste trabalho (detalhada na seção 4) não inclui uma base de conhecimentos separada da estrutura de dados modelada para o software de monitoração de tempo real, de forma que não é necessário que a aplicação se preocupe em realizar alterações na base de fatos que são considerados pelas regras.

3.3 EXPRESSIVIDADE DO CONHECIMENTO DECLARATIVO

O terceiro desafio importante abordado neste trabalho é a expressividade do paradigma declarativo a ser utilizado. Os sistemas que foram vistos na seção 3.1.3 trabalham apenas com lógicas de ordem 0 ou 1. Isso limita a expressividade do conhecimento expresso nas regras, já que há diversos problemas que não podem ser resolvidos com esse tipo de lógica, tais como raciocínios que lidem com conhecimentos incertos ou imprecisos e raciocínios que dependam da evolução temporal de acontecimentos.

O *framework* proposto neste trabalho, além de regras baseadas em uma lógica de primeira ordem, também realiza inferências com lógica fuzzy e com operadores lógicos usados em lógica temporal, como será abordado em detalhes no capítulo 4. Por hora serão feitas algumas observações sobre estas distintas lógicas e sua expressividade.

3.3.1 LÓGICA DE PRIMEIRA ORDEM

A Lógica de Primeira Ordem ou Lógica de Predicados é usada na maioria dos *frameworks* vistos na seção 3.1.3 e é uma extensão da Lógica Proposicional. A Lógica Proposicional é uma linguagem que consiste de símbolos de proposições e conectivos

lógicos que permite manipular proposições que sejam verdadeiras, falsas ou desconhecidas e, a partir dessas proposições, criar um mecanismo de inferências para produzir novos fatos. Enquanto a Lógica Proposicional lida apenas com fatos atômicos, a Lógica de Primeira Ordem manipula objetos e suas relações usando os mesmos conectivos lógicos. Para exemplificar, em um ambiente de programação uma proposição simples pode ser avaliada independentemente de qualquer variação, tal como:

4 == 2 + 2

Já a Lógica de Primeira Ordem lida com expressões que, para serem avaliadas, necessitam de instâncias de valores, tais como:

alvo.ambiente == “aéreo”

Isto aumenta a expressividade da lógica, pois permite lidar com distintos objetos e as relações entre eles em um determinado modelo de mundo.

Além disso, a lógica de predicados acrescenta quantificadores que não existem na lógica proposicional. Assim, dada uma sentença φ qualquer, então ∀x φ significa que φ é verdadeiro para todo valor de x e ∃x φ significa que para algum valor de x φ é verdadeiro.

Um problema, ao lidar com Lógica de Primeira Ordem, é que ela lida com negações clássicas ou negações fortes (*hard negation*) e também com negações por falha (*soft negations*). Negações fortes são afirmativas das quais existe certeza sobre o fato de que são falsas (usualmente representa-se como ¬A). Já as negações por falha são afirmativas que não se pode provar, ou seja, são desconhecidas (geralmente representa-se como ~A).

Benjamin Grosof [40] introduziu o conceito da Lógica Cortês como uma extensão da lógica tradicional que aumenta a expressividade das regras. Para tanto, são acrescentados três recursos: rótulos para identificar as regras, uma instrução de sobrescrição para estabelecer as prioridades entre os rótulos das regras (da priorização de regras é que vem o nome Lógica Cortês) e um mecanismo de exclusão mútua entre predicados.

A precedência é introduzida para tratar possíveis conflitos. Um conflito aparece sempre que duas regras avaliadas como verdadeiras oferecem conclusões conflitantes, tais como A e ¬A. Introduzindo prioridades nas relações entre regras, a máquina de inferências sempre produz conclusões consistentes e computacionalmente tratáveis.

Existem propostas, como a de Marc Dörflinger [41] para motores de inferência em Lógica Cortês que propõem realizar um pré-processamento das regras, identificar as que possuam negações por falha e substituí-las por novas regras que somente possuam negação forte. Desta forma sempre é produzido um conjunto consistente de regras. Isto é necessário porque nestas propostas se deseja trabalhar com uma hipótese de mundo aberto.

Já a hipótese de mundo fechado e completamente especificado simplifica a confecção das regras. Isto é semelhante ao que ocorre em uma consulta a um banco de dados, onde se considera que as conclusões supõem que a informação está completamente representada no banco de dados. Por exemplo, imagine que se execute uma consulta do tipo:

Selet * from alvos where alvos.tipo == “navio”

Naturalmente espera-se que a resposta seja o conjunto de todos os navios conhecidos. A rigor, a Lógica de Primeira Ordem considera a hipótese de mundo aberto, o que não permite concluir que a resposta corresponda efetivamente a todos os navios, pois pode haver alvos para os quais não se conheça o tipo e que efetivamente sejam navios. Por esse motivo, todos os sistemas relatados na seção 3.1.3 trabalham com a hipótese de mundo fechado, o que simplifica as inferências.

3.3.2 LÓGICA FUZZY

O conjunto Fuzzy ou Nebuloso foi proposto originalmente por Zadeh [42] para modelar dados incertos encontrados na vida real. Na teoria de conjuntos clássica, um elemento pertence ou não pertence a um conjunto. Na teoria Fuzzy, um elemento possui um determinado grau pertinência a um conjunto, que é modelado por uma função Fuzzy.

Funções fuzzy podem ser usadas para dividir o domínio de uma variável (também chamado de *Universo do Discurso* da variável) em partições não necessariamente mutuamente exclusivas. Essas partições podem ser identificadas por nomes, também chamados de *Conceitos Lingüísticos*. A cada valor do domínio da variável, pode então ser atribuído um grau de pertinência a um dos conceitos lingüísticos.

A figura 3 mostra uma variável temperatura, com um universo do discurso de 100^{o} até 360^{o}, dividido em cinco partições representadas por funções Fuzzy triangulares e trapezoidais com interseções entre si. Foram atribuídos às partições os conceitos lingüísticos *Frio, Média, Quente e Muito Quente*.

μ

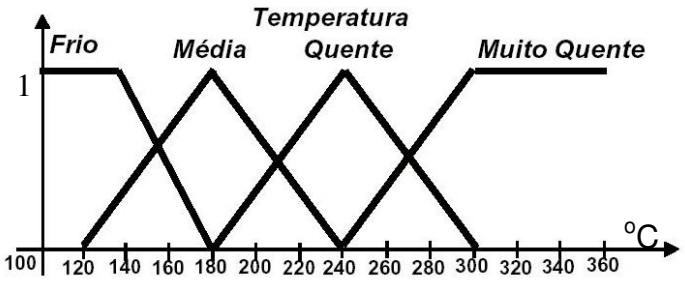


Figura 3 – Conceitos Lingüísticos modelados por funções Fuzzy

Formalmente, se x é um elemento de um domínio X, um conjunto "fuzzy" de A em X é definido por um conjunto de pares ordenados:

$A = \{(x, \mu_A(x)) / x \in X\}$

Onde:

$\mu_A(x)$ é a pertinência do elemento *x* ao conjuto *A* dado por uma função Fuzzy.

Para o exemplo da figura 3, cada temperatura *x* entre 100 e 360 graus possui um valor de pertinência $\mu_{Frio}(x)$, $\mu_{Média}(x)$, $\mu_{Quente}(x)$, $\mu_{Muito\ Quente}(x)$ dados pelas funções Fuzzy modeladas na figura 3. Naturalmente que para cada x algumas dessas pertinências são zero e algumas não são.

Existem algumas formas propostas por diferentes autores para realizar operações de união, intersecção e complemento com conjuntos Fuzzy. Todas operam com os graus de pertinência. Assim, se *A* e *B* são dois conjuntos "fuzzy" em *U* com suas funções de pertinências μ_A e μ_B , respectivamente, então as formas mais usadas para cálculo da união, da interseção e do complemento estão descritas na tabela 1.

	$\mu_{A\cup B}(x)$	$\mu_{A\cap B}(x)$	$\mu_{\neg A}(x)$
Zadeh	$max\ [\mu_A(x), \mu_B(x)]$	$min\ [\mu_A(x), \mu_B(x)]$	$1-[\mu_A(x)$
Produto	$[\mu_A(x) + \mu_B(x)] - [\mu_A(x) * \mu_B(x)]$	$\mu_A(x) * \mu_B(x)$	$1-[\mu_A(x)$
Lukasiewicz	$min\ [1, (\mu_A(x) + \mu_B(y)]$	$max\ [0, (\mu_A(x) + \mu_B(y) - 1]$	$1-[\mu_A(x)$

Tabela 1 – Fórmulas de cálculo de Operadores Fuzzy mais usadas

Se A e B são dois conjuntos Fuzzy não vazios, então uma relação Fuzzy R é um subconjunto Fuzzy de AxB. A relação Fuzzy mais comum é a relação *sup-star*:

Def. sup-star: Se *R* e *S* são relações Fuzzy em *UxV* e *VxW,* respectivamente, a composição *R* e *S* é uma relação Fuzzy definida por

$RoS = \{(u,w), sup[\mu_R (u,v) * \mu_S (v,w)]\}$,

Onde: $u \in U$, $v \in V$ e $w \in W$.

* é qualquer operador de norma triangular (geralmente *min*).

a composição *RoS*, nada mais é do que o produto cartesiano de matrizes, sendo que no lugar da adição é utilizado o operador de união Fuzzy e no lugar da multiplicação é utilizado o operador de intersecção Fuzzy.

Uma implicação fuzzy é composta por uma regra que possui a seguinte forma:

SE x é A ENTÃO y é B

onde *A* e *B* são valores lingüísticos definidos por conjuntos Fuzzy no universo de discurso *X* e *Y.* Por exemplo:

Se temperatura é alta então vazão é baixa

A implicação $x \rightarrow y$ é uma relação binária no produto do espaço *X* x *Y* que descreve a relação entre *x* e *y* e, que deve ser calculada ponto a ponto. As fórmulas de cálculo dessa operação de implicação mais utilizadas são descritas na tabela 2.

	$x \rightarrow y$
Larsen	x.y
Lukasiewicz	min(1, 1-x+y)
Mamdani	min(x,y)

Tabela 2 – Fórmulas de cálculo de implicação Fuzzy mais usadas

Um sistema Fuzzy é composto por um conjunto de regras da forma:

SE x_1 *É* P_1 *E* x_2 *É* P_2 *E ...* x_n *É* P_n *ENTÃO y É Q* (3.1)

Onde:

$\bar{x} = (x_1, x_2, ...x_n)^T \in U$ (vetor de variáveis de entrada)

$U \subset R^n$ (Universo do discurso das entradas)

$y \in W$ (variável de saída)

$W \subset R$ (Universo do discurso da saída)

$P_1..P_n$, Q são conceitos lingüísticos modelados com funções Fuzzy

Cada regra em (3.1) define $P_1 x ... x\ P_n \Rightarrow Q$ no espaço *UxW.*

Os sistemas de inferências lógicas fuzzy são usualmente chamados de sistemas de controle fuzzy, pois encontram muitas aplicações na área de controle. Eles foram primeiramente propostos por Mamdani [43]. Para utilizar um conjunto de regras fuzzy em uma aplicação de engenharia que lide com variáveis reais, deve-se criar um sistema que possa produzir as inferências a partir de regras escritas utilizando-se as variáveis lingüísticas características do domínio (segundo os especialistas). Neste sistema, as regras atuam sobre as pertinências calculadas pela fuzzificação dos valores de entrada. As regras disparadas em um determinado instante produzem resultados que devem então ser agregados para produzir um conjunto final que passa ainda por um processo de *deffuzificação* para produzir um valor escalar na saída. A figura 4 ilustra um sistema de inferências fuzzy.

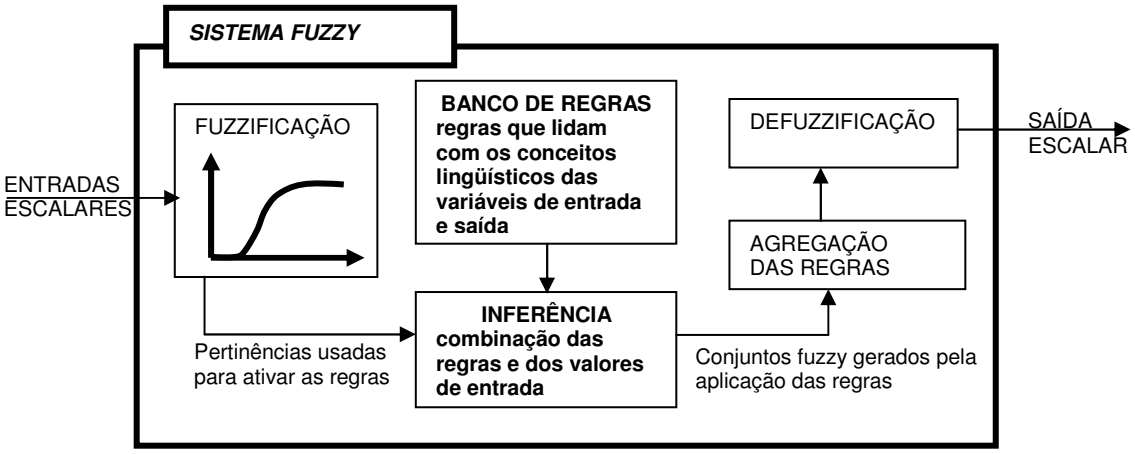


Figura 4 – Sistema típico de inferências Fuzzy

Por lidarem apenas com conceitos lingüísticos tais como *perto*, *longe*, *crítico*, *lento*, *rápido*, *grande* ou *pequeno*, as regras criadas com essa abordagem permitem uma maior expressividade para transcrever o conhecimento do senso comum.

Como as funções fuzzy que modelam os conceitos lingüísticos podem ter interseções no domínio de uma variável, um mesmo valor de entrada pode disparar regras com diferentes conceitos lingüísticos nas premissas. Por exemplo, uma velocidade que seja considerada ao mesmo tempo como “pequena” e como “média” (com diferentes graus de pertinência) poderá causar o disparo de regras diferentes em que a velocidade esteja associada nas premissas aos conceitos lingüísticos *pequena* e *média*.

A Lógica Fuzzy possui muitas aplicações e diversas formas de cálculos para os operadores lógicos, para a forma de cálculo da inferência, para a agregação de regras

que tenham sido disparadas em um determinado instante e para a fase final de defuzificação, ou seja, produção de um valor final escalar na saída. Sob o ponto de vista de aumentar a expressividade do conhecimento contido nas regras, com certeza a utilização da lógica fuzzy simplifica a compreensão do domínio e a expressão do conhecimento, na medida em que é possível utilizar os conceitos lingüísticos na elaboração dos raciocínios que compõem o saber especializado.

3.3.3 LÓGICA TEMPORAL

Algumas abordagens têm sido sugeridas para tratar o problema de inclusão do domínio do tempo no raciocínio baseado em conhecimento simbolicamente representado [44, 45, 46]. Qualquer lógica que considere o tempo como uma seqüência de estados é uma lógica temporal. Em uma lógica atemporal, como a lógica proposicional ou a lógica de primeira ordem, uma assertiva considerada verdadeira, tal como “Fulano é magro”, é invariante no tempo. Em uma lógica temporal essa assertiva pode variar no tempo em relação à avaliação como verdadeira ou falsa. Em alguns tipos de lógica temporal é possível expressar raciocínios tais como “Fulano sempre foi magro”, “Fulano eventualmente ficará magro”, “Fulano era magro *enquanto* não comia” ou “Fulano era magro *até que* passou a comer”.

De uma forma geral, as lógicas temporais procuram prover mecanismos que permitam efetuar raciocínios variantes no tempo. Para tanto existem diferentes propostas. A lógica temporal pode variar na sua linguagem, que pode ser proposicional ou de primeira-ordem e no modelo de tempo utilizado, que pode ser linear (como na Lógica de Tempo Linear – LTL – que modela o tempo como uma seqüência de estados) ou ramificado (como na *Computation Tree Logic* – CTL – que modela o tempo como uma árvore de futuras possibilidades não determinadas) e, ainda, contínuo ou discreto.

Sob o ponto de vista de aumentar a expressividade do conhecimento contido nas regras, que é o assunto desta seção, é importante que o KBS considere alguma lógica que seja capaz de modelar as relações temporais entre eventos ou estados. O framework apresentado neste trabalho implementa uma versão simplificada de algumas propostas de mecanismos de inferência que usam lógica temporal. A principal aderência é com a Lógica Temporal de Intervalos (ITL do acrônimo em inglês) originalmente desenvolvida em [47].

No capítulo 4, no qual é explicada a arquitetura do *framework*, são fornecidos maiores detalhes sobre a sintaxe a semântica e os operadores implementados. Entretanto, pode-se adiantar que a idéia básica é permitir o raciocínio temporal a partir

de eventos. Eventos são fatos com uma marca de tempo associada. A utilização de eventos que são considerados em intervalos finitos do passado simplifica a implementação sem sacrificar a expressividade. A quantidade de memória requerida para guardar eventos é minimizada, pois só são memorizados eventos que eventualmente sejam examinados nas regras e somente pelo tempo em que serão necessários, de acordo com a semântica do operador temporal envolvido. Por exemplo, se uma regra considera a ocorrência de um evento nos 10 minutos passados, somente eventos dentro desses dez minutos estarão registrados. Se outra regra considera a do mesmo evento sobre um período de 20 minutos passados, aí então somente eventos dos últimos 20 minutos estarão guardados.

4 A ARQUITETURA DO MODELO

Este trabalho propõe um ambiente em que seja possível expressar o conhecimento com regras que utilizam distintas lógicas e raciocinar automaticamente com a mudança dos valores utilizados na expressão dessas regras. Essa abordagem permite implantar (como será visto no capítulo seguinte) os níveis 1, 2 e 3 da fusão de dados em ambientes táticos navais com regras que utilizam distintas lógicas. Este ambiente é adequado á utilização em um projeto de software que tenha sido orientado a objetos ou orientado a componentes e pode-se qualificá-lo como um "*framework*" de inferência para múltiplas lógicas. Este capítulo detalha as soluções adotadas no *framework* proposto e implementado que, no restante deste trabalho, será chamado de Real Time Multiple Logic Reasoner (RT-MLR). São abordados os tópicos de como é representado o conhecimento, qual é o funcionamento do motor de inferências e qual é o funcionamento da sintaxe e da semântica das diferentes lógicas empregadas na elaboração das regras.

4.1 OS OBJETIVOS DO RT-MLR

A motivação principal dos *frameworks* que suportam conhecimento declarativo é separar o conhecimento do negócio em si em um conjunto de regras que pode ser mais facilmente mantido e depurado por especialistas do negócio. Desta forma, esses especialistas não necessitam de conhecimento profundo da metodologia ou da arquitetura que foi utilizada para implementar o restante do software aplicativo

Além de visar integrar as abordagens de programação declarativa e procedimental, o RT-MLR também visa contemplar características de aplicações de monitoração em tempo real, nas quais há uma grande freqüência de alteração de

valores e aumentar a expressividade dos conhecimentos representados nas regras. Observados pontualmente, os objetivos principais que embasam o modelo são:

- Aumentar o desacoplamento entre dados e conhecimento, na medida em que a programação declarativa permite expressar a lógica separadamente dos dados. Os dados estão nas classes do domínio e a lógica está nas regras. O desacoplamento entre o conhecimento e a arquitetura de classes do domínio pode ajudar a expressar soluções complexas em domínios complexos.
- Aumentar a escalabilidade, na medida em que novos conhecimentos podem ser facilmente agregados (pode-se incluir, excluir ou alterar uma regra sem implicar em alteração do domínio).
- Aumentar a facilidade para analisar o conhecimento estático e justificar o raciocínio dinâmico. A análise estática é facilitada na medida em que as regras são uma forma de expressar conhecimento de forma fácil, pois estão destacadas das classes de domínio. A análise dinâmica é facilitada pela possibilidade de verificação do raciocínio por meio do rastreamento das regras usadas para as inferências realizadas.
- Disparar regras automaticamente a partir da alteração significativa dos valores dos atributos das classes do domínio da aplicação que estejam sendo considerados nas regras.
- Aumentar a integração sintática e semântica, na medida em que as regras são expressas na própria linguagem em que é construído o domínio da aplicação (Java na implementação realizada para validação, embora a proposta não impeça que possa haver implementações em outras linguagens orientadas a objetos como C++, por exemplo). O desenvolvedor não necessita aprender outra sintaxe para expressar as regras, que são expressas na própria linguagem OO. Além da aceleração da curva de aprendizado, há o aumento da expressividade da conclusão das regras, na medida em que qualquer comando nativo da linguagem pode ser empregado (no caso da implementação em Java, todo o arsenal da *Java Virtual Machine* pode ser acessado pelas regras). Essa característica também elimina a necessidade de utilização de outros interpretadores ou pré-compiladores. Tanto para o domínio da aplicação quanto para o domínio do conhecimento expresso nas regras, apenas o compilador Java é utilizado para analisar a correção sintática e semântica do código gerado.

- Aumentar a expressividade do conhecimento, na medida em que são implementados múltiplos mecanismos de inferência que suportam regras expressam distintas lógicas, como a Lógica de Primeira Ordem, a Lógica Nebulosa (ou Fuzzy) e a Lógica Temporal.

4.2 DEFININDO ESSENCIALMENTE O MODELO

O RT-MLR é essencialmente uma proposta de acoplamento do paradigma declarativo às arquiteturas orientadas a objetos em aplicações de tempo real. A visão proposta é que as regras do paradigma declarativo são vistas como algo que interage com o domínio da aplicação. Entretanto, essa interação não é tão próxima que necessite conhecimento de métodos e seqüências de atividades do domínio da aplicação, nem é tão distante a ponto de ignorar a modelagem dos dados nas classes do domínio.

De fato, as regras são modeladas como classes de associação entre os objetos do domínio da aplicação (veja na figura 5 um exemplo para um domínio de medicina). De uma forma mais genérica, as classes de associação representam conjuntos de regras que permitem expressar algum particular aspecto do domínio da aplicação que leva em consideração a relação existente entre atributos de objetos de classes distintas, que eventualmente não guardam nenhuma relação hierárquica entre si. Essa linha de raciocínio, que tenta integrar regras à arquitetura do domínio definido em UML está presente em outros trabalhos atuais, como em [48].

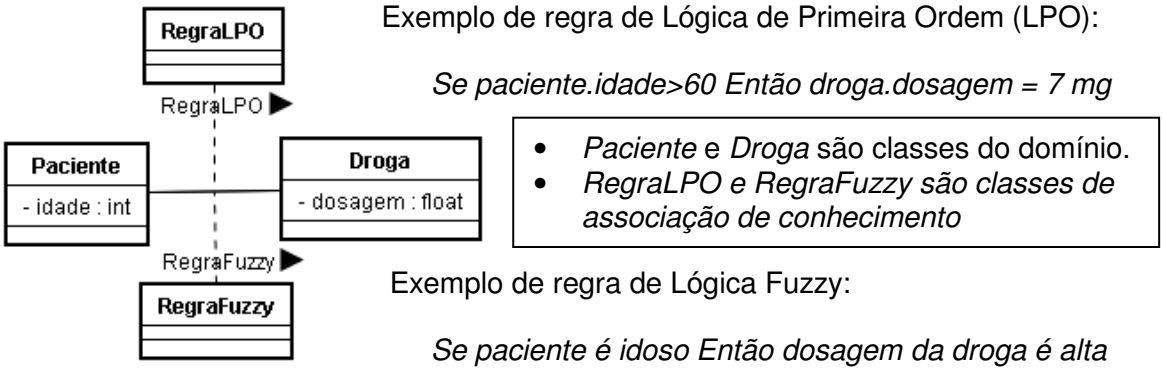


Figura 5 – Regras de associação utilizando distintas lógicas

Assim, no RT-MLR, as classes de associação acomodam o conhecimento que relaciona os dados de distintos objetos do domínio, respeitando a separação entre as abordagens procedimental e declarativa. As classes de regras têm uma construção simples que não requer grande conhecimento do restante da modelagem e nem requerem uma sintaxe proprietária para expressar o conhecimento, na medida em que

este é expresso na própria linguagem de programação em que foi programada a aplicação (na implementação efetivamente realizada do RT-MLR, em Java).

As regras utilizam operadores lógicos que dão suporte às distintas lógicas contempladas pelo motor de inferências do RT-MLR. Esses operadores, tais como os operadores lógicos booleanos da Lógica de Primeira Ordem, os operadores lógicos da Lógica Fuzzy e os operadores booleanos e modais específicos da Lógica Temporal, são oferecidos como recursos nativos do RT-MLR que são chamados no corpo das regras.

Como as regras utilizam a sintaxe da linguagem da aplicação, não é necessário nenhum esforço de aprendizado de uma linguagem específica para expressar as regras, nem é necessário oferecer um interpretador sintático e semântico adicional. Além disso, podem ser utilizados nas regras quaisquer comandos nativos da linguagem da aplicação, tanto na parte da conclusão quanto na parte da premissa das regras.

A linguagem da aplicação é aqui citada de uma forma genérica, embora a implementação realizada do RT-MLR seja para a linguagem Java. O motivo da generalidade da citação é que, por ser um framework leve, podem perfeitamente ser realizadas outras implementações do RT-MLR para outras linguagens orientadas a objetos, como C++, por exemplo.

4.3 REPRESENTAÇÃO DO CONHECIMENTO

A representação do conhecimento no RT-MLR é feita em regras que representam raciocínios expressos em diversas lógicas. A mais tradicional destas é a Lógica de Primeira Ordem.

No RT-MLR assume-se a hipótese de mundo fechado, pois a proposta é utilizar o modelo em problemas de monitoração em tempo real, nos quais as variáveis estão todas definidas, ou seja, o modelo define completamente o mundo que se deseja representar.

Um problema já explicado na seção 3.3.1 é o conflito de conclusões, que no RT-MLR é resolvido com um mecanismo de priorização obrigatória das regras. No RT-MLR existe o conceito de conjuntos de regras. Podem ser criados conjuntos distintos de regras para cada tipo de lógica. Assim, podem coexistir conjuntos para regras de lógica de primeira ordem, conjuntos de regras fuzzy e conjuntos para regras temporais. Os conjuntos podem existir concomitantemente e a inferência é realizada simultaneamente para todos os conjuntos.

Após a criação dos conjuntos, as regras devem ser explicitamente adicionadas a um dos conjuntos. No caso de um conjunto de regras de lógica de primeira ordem, a hierarquia das regras dentro do conjunto é definida pela ordem de inserção. Assim, as regras que são inseridas depois são hierarquicamente superiores às que são inseridas primeiro. O conjunto de regras de lógica de primeira ordem serve basicamente para estabelecer essa hierarquia. Em uma aplicação típica, é provável que só seja necessário criar um conjunto para todas as regras de lógica de primeira ordem. Entretanto, nada impede que seja criado mais de um conjunto de regras de Lógica de Primeira Ordem.

Já para o caso de regras fuzzy, a quantidade de conjuntos tende a ser maior. Neste caso, cada conjunto basicamente deve agregar as regras que se referem às mesmas variáveis de saída (variáveis que aparecem na parte da conclusão das regras). Mais adiante, quando for explicado o funcionamento das regras fuzzy, esta necessidade ficará mais clara.

Para o caso das regras com lógica temporal, o conjunto de regras funciona da mesma forma que para a lógica de primeira ordem. De fato, não há necessidade de criar um conjunto específico para regras de Lógica Temporal, na medida em que essas regras são de fatos regras de Lógica de Primeira Ordem que utilizam algum operador de Lógica Temporal. Assim, desde que seja criado um conjunto de regras de Lógica de Primeira Ordem, as regras que eventualmente utilizem operadores de Lógica Temporal podem ser agregadas a esse conjunto na ordem de precedência que se julgue adequada.

4.4 O MECANISMO DE INFERÊNCIA

Em sistemas típicos baseados em regras, existe um conjunto de regras, conhecido como base de conhecimentos e uma base de dados ou base de fatos, também chamada de memória de trabalho. O funcionamento básico de um motor de inferências deve permitir que fatos sejam externamente incluídos ou retirados da base de fatos. A partir de cada inclusão o mecanismo deve percorrer as regras examinando se há regras que podem ser disparadas pela inclusão do novo fato. O disparo das regras pode agregar novos fatos à base de conhecimento o que gera o disparo de mais regras em um processo recorrente. Essa estratégia de inferência, como já foi dito, é chamada de orientada a dados.

4.4.1 A INFERÊNCIA EM MOTORES TRADICIONAIS

Para lidar com o fato empiricamente observável de que existe pouca mudança nas condições testadas nas regras, a maioria das implementações de motores de inferência (OPS5, ART, CLIPS, JESS, JEOPS, DROLS e quase todos os outros já citados) utiliza o método RETE de busca orientada a dados [28]. O algoritmo RETE se baseia em duas observações:

1. Que o disparo de uma regra afeta apenas um pequeno conjunto de regras e nem todas precisam ser reexaminadas.
2. Que muitas regras possuem estruturas similares no seu lado esquerdo, não havendo necessidade de replicar as estruturas.

O algoritmo propõe então um exame prévio das regras e a montagem de uma rede de nós que representam testes de um valor (os testes de igualdade de valores) ou dois valores (os testes de junção com operadores lógicos), que são os padrões que aparecem do lado esquerdo das regras. Ao construir a rede, é possível identificar padrões iguais que apareçam nas regras e simplificar a rede. No exemplo retratado na figura 6a existem duas regras que envolvem a identificação de três padrões (que são os nós com o símbolo "?") e a combinação dos mesmos com dois operadores (que são os nós com o símbolo "+").

O exame prévio da estrutura das regras permite descobrir as redundâncias estruturais e construir uma rede de nós com a estrutura retratada na figura 6b.

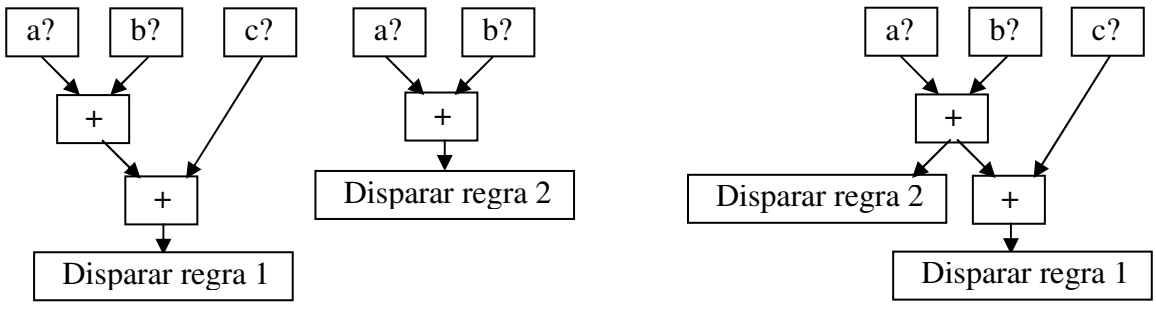


Figura 6 – Duas regras que envolvem a identificação de três padrões

Os nós marcados com "+" possuem uma memória do lado esquerdo e uma memória do lado direito e é essa memória que acelera a inferência. A cada novo fato introduzido na base de conhecimento, os motores de inferência testam o disparo dos nós superiores. Caso haja compatibilidade, a informação é enviada para baixo e vai

fluindo pela rede, testando novas compatibilidades nos nós abaixo. Caso passe por todos os testes, a regra é disparada.

Suponha que tenha havido a inclusão na base de conhecimento de um fato testado no nó “c?”. Se o fato é compatível com o valor testado, essa compatibilidade é passada ao nó duplo abaixo (marcado com “+”) e é testada contra o valor armazenado na memória do lado esquerdo desse nó. Caso haja compatibilidade a regra 1 será acionada.

Os motores que utilizam o algoritmo RETE fazem um pré-processamento das regras e constroem primeiramente essa rede. Esses motores também oferecem algum mecanismo para excluir e incluir novos fatos na base de conhecimento. Após cada inclusão na base de conhecimentos, eles percorrem os nós superiores buscando nós que combinem com o novo fato e, a partir daí, a informação flui para baixo pela rede levando ao disparo das regras quando a informação atinge as folhas da rede.

Na capacidade de memória dos nós da rede é que reside a simplificação do exame de padrões no algoritmo RETE. Supõe-se que, por exemplo, se somente o valor do lado direito foi recalculado pela investigação de um nó, então o valor do lado esquerdo não sofreu alteração e pode ser usado o valor armazenado na memória do lado esquerdo. Caso algum novo fato futuro altere somente o valor do lado esquerdo desse nó então uma nova inferência será feita usando-se agora o valor armazenado na memória do lado direito, em um esquema de análise seqüencial de cada novo fato inserido na base de fatos. Como será visto na seção que se segue, isto pode representar um problema em aplicações concorrentes de tempo real.

4.4.2 A INFERÊNCIA NO RT-MLR

Um dos objetivos do RT-MLR é adaptar o mecanismo de inferência ao processamento de tempo real de uma aplicação que possua uma modelagem e uma implementação orientada a objetos e tipicamente multi-thread. Algumas abordagens simplificadoras foram então adotadas.

4.4.2.1 SIMPLIFICANDO A BASE DE FATOS

A primeira característica básica da inferência no RT-MLR vem da constatação de que os dados a serem considerados pelas regras de negócio em uma aplicação dedicada de tempo real estão no próprio domínio da aplicação. Por exemplo, a velocidade de um alvo, a quantidade de batimentos cardíacos de um paciente ou a

pressão de uma linha de transmissão de vapor em uma planta industrial, são informações ou “fatos” que podem ser considerados em regras de produção.

Assim, no RT-MLR não há uma base de fatos (memória de trabalho) paralela à base de dados da aplicação e exclusiva para realização de inferências. Desta forma evita-se a replicação de informações já existentes no domínio da aplicação. A figura 7 ilustra o conceito:

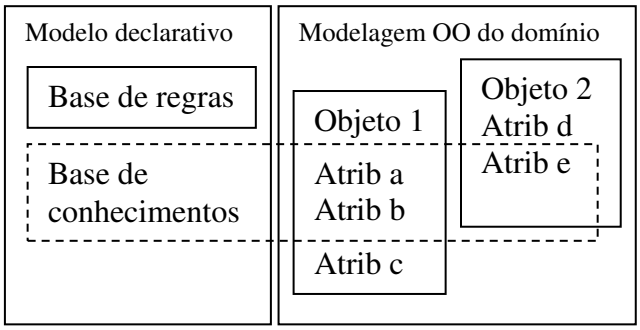


Figura 7 – Os fatos considerados nas regras do RT-MLR

A base de fatos é formada apenas por atributos que já tenham sido definidos no domínio da aplicação. A abordagem é de mundo fechado de conhecimentos, onde as regras consideram diretamente os atributos definidos no domínio da aplicação. Assim sendo, os fatos são os valores dos atributos e um novo fato é uma mudança de valor em um desses atributos. Na hipótese de mundo fechado assume-se que estes fatos existem desde sempre com algum valor, quer sejam valores numéricos, lógicos ou valores de qualquer outro tipo básico.

Para realizar as inferências as regras são então acionadas a partir de um mecanismo que será explicado na seção seguinte e atuam diretamente sobre os atributos dos objetos específicos a que foram atreladas. Isto implica que a aplicação deve criar instâncias das regras para os objetos do domínio que desejar investigar, mas por outro lado a aplicação não necessita incluir e excluir fatos na base de fatos.

A idéia básica dessa troca é a observação de que em aplicações de domínio fechado, como sistemas de monitoração, as regras são escritas por especialistas para atuar sobre entes genéricos, tais como a pressão arterial de um paciente, a velocidade de um alvo ou a temperatura de uma caldeira, mas, ao passarem a ser consideradas como parte da aplicação, podem ser explicitamente instanciadas por quem constrói os objetos do domínio, pois esses indivíduos conhecem a arquitetura do domínio e podem ver as regras como classes de associação entre objetos do domínio. Assim,

como eles têm essa visão profunda do domínio, apesar de não terem construído as regras, com certeza saberão identificar quais regras se referem a quais objetos do domínio.

Por exemplo, em uma aplicação de Sistema Tático, um padrão de projeto do tipo *Abstract Factory* provavelmente fará parte da arquitetura OO e será utilizado para criar acompanhamentos de alvos no cenário tático. Ao criar objetos da classe *Acompanhamento*, o arquiteto do padrão *Abstract Factory* pode instanciar as regras que um especialista tenha previamente criado e que se refiram a algum atributo da classe *Acompanhamento*.

A consideração da base de fatos como a própria base de dados da aplicação provoca a troca da necessidade de atualizar informações em uma base de fatos separada do domínio da aplicação (o que em um sistema de tempo real teria de ser feito muitas vezes pela aplicação, ou melhor, tantas vezes quantas haja alterações) pela necessidade de instanciar no domínio da aplicação as regras de interesse de cada objeto (o que é feito uma única vez ao instanciar o referido objeto).

Desta forma não há necessidade de unificar regras e fatos, pois as regras são objetos para os quais há distintas instâncias. A quantificação da lógica de primeira ordem fica garantida pela múltipla instanciação das regras, compatível com o princípio da reusabilidade que é possibilitado e objetivado pelo emprego das técnicas de OO.

4.4.2.2 ACIONAMENTO AUTOMÁTICO DA INFERÊNCIA

A segunda característica importante, diz respeito aos valores considerados para a inferência. Em um ambiente de tempo real em que os fatos são atributos do domínio da aplicação, as alterações destes estão sendo geradas por sensores assíncronos, concorrentes e em processos paralelizados (a aplicação freqüentemente é *multi-thread*, por exemplo). Nesse ambiente, a seqüência de fatos novos gerados pode ser muito rápida. No mecanismo do RETE explicado na seção 4.4.1, cada novo fato incluído na base de conhecimentos é filtrado pela rede até chegar aos nós folha, causando então a ativação das regras. Como foi explicado, na capacidade de memória dos nós da rede é que reside a simplificação do exame de padrões no algoritmo RETE. Se vários fatos são gerados quase ao mesmo tempo em um ambiente concorrente (multi-thead), antes que seja possível realizar as inferências para o primeiro deles, então as decisões tomadas podem não estar atualizadas com a realidade corrente. Essa característica será explicada a seguir.

Suponha, por exemplo, que um fato novo F1 seja gerado e inicie o disparo do mecanismo de inferência de um *framework* que utilize o algoritmo RETE. Suponha ainda que logo a seguir atualizações assíncronas de dados ocorram na base de dados da aplicação. Isto equivale à geração de novos fatos. Suponha, por exemplo, que foi gerado o fato F2. Este fato gerará futuras investigações da rede, mas, no momento, estará aguardando a propagação do fato F1 na rede, pois o mecanismo de inferência não é concorrente. Eventualmente o fato F2, que só será analisado mais adiante, pode ser considerado na premissa de uma regra na qual o fato F1 também é considerado. Suponha então que o fato F1 é usado do lado esquerdo de um operador e o fato F2 é usado do lado direito do operador em uma mesma regra:

If F1 and F2 then...

Quando esta regra for avaliada pela inclusão do fato F1 na base de fatos, será usado o valor memorizado para o lado direito do operador, pois o fato F2 ainda não pertence à base de fatos. A regra será avaliada usando o valor armazenado (que não é o mais atual, já que F2 já possui uma atualização disponível) e isto pode gerar uma decisão não atualizada. Naturalmente que o fato F2 será adicionado à base de fatos no futuro breve, causando a ativação da mesma regra e reconsiderando a decisão passada, mas a decisão inicial tomada pode ter sido equivocada, em virtude de se ter usado um dado armazenado na estrutura da rede que não correspondia à fotografia mais atual em um sistema de aquisição de dados em tempo real. Além disso, pode eventualmente não ser possível reconsiderar as decisões tomadas.

Por este motivo o RT-MLR não utiliza valores armazenados em memórias de trabalho paralelas. Não existe uma base de fatos paralela aos dados da aplicação nem existe uma memória em uma estrutura de rede previamente construída para otimizar as redundâncias possivelmente existentes nas regras. Para tomar a decisão mais atual possível a cada momento, o RT-MLR paga o preço de sempre ler os valores correntes da base de dados no momento em que a regra está sendo investigada. Assim, na situação anteriormente descrita, o RT-MLR dispararia a avaliação da regra pela alteração de dados correspondente ao fato F1 e, ao investigar a regra em que também constasse a verificação do valor correspondente ao fato F2, este seria lido diretamente da base de dados do domínio da aplicação. Isto garante que se este dado foi recentemente atualizado por um processo concorrente, o valor mais atual será lido e considerado na avaliação da regra.

Assim, no RT-MLR o mecanismo de acionamento das regras é diretamente vinculado à alteração do valor das variáveis do domínio, o que define uma estratégia

de acionamento que pode ser chamada de "*on line monitoring*". Desta forma, ao ser alterado o valor de alguma variável que é citada na premissa de alguma regra, esta é automaticamente investigada. Apenas as regras que envolvem essa variável são investigadas e eventualmente acionadas. Nesse sentido, o modelo aplica uma variação da estratégia orientada a dados. Não é feita uma busca por regras de interesse para os "fatos" novos e sim é mantida uma estrutura que indica diretamente as regras de interesse para determinado fato. O eventual disparo de uma ou mais regras de interesse pode, por si só, gerar a alteração de "fatos" que sejam utilizados em outras regras. Assim, uma cadeia para frente de acionamentos (e "descobertas") será gerada a partir da alteração de um dado dentro do domínio da aplicação. Por exemplo, podemos ter regras do tipo:

Se alvo é do tipo míssil e situação de operação é combate então disparar alarme de ataque aéreo em curso

Se velocidade do alvo >= 1000 nós então alvo é do tipo míssil

Para cada alteração de velocidade lida pelo sensor de cada alvo acompanhado pelo sistema de controle tático, será acionada a segunda regra e, caso esta seja disparada, a alteração do tipo causará a investigação também da primeira regra. As informações sobre a velocidade, tipo do alvo e a situação de combate podem estar em objetos distintos no domínio da aplicação. A base de fatos, portanto, reside no próprio domínio da aplicação. Ela pode ser alterada por partes do software de aplicação (por exemplos processos de leitura de sensores) ou pela parte da conclusão das regras que tenham sido acionadas (como no caso da conclusão da segunda regra). Em ambos os casos as regras adequadas serão diretamente selecionadas para serem investigadas e poderão ou não gerar novos "fatos".

4.4.2.3 OS TIPOS DE DADOS UTILIZADOS NAS REGRAS

No RT-MLR é construída uma estrutura de listas que associa valores utilizados na base de conhecimentos com as regras em que estes valores são testados. Assim, apenas as regras que envolvem esses valores são investigadas quando há alguma alteração desses valores.

Os atributos das classes de domínio que são utilizados nas classes de regras não devem ser definidos pela aplicação como atributos de tipos básicos. Eles devem ser definidos herdando de classes fornecidas pelo RT-MLR que encapsulam os tipos básicos do Java (numéricos, booleanos, etc). Ao serem mudados os valores desses

tipos nas classes do domínio é feito o disparo das regras que tenham sido agrupadas nos conjuntos de regras anteriormente definidos e que constam nas listas.

Esses tipos são chamados de “dados observáveis” (ou *ObservableData* como será descrito na próxima seção em que é explicada a sintaxe do RT-MLR). A esses tipos está acoplado um monitor para garantia de acesso mutuamente exclusivo em caso de aplicações concorrentes.

O acionamento das regras pela mudança dos dados possui ainda uma sofisticação que pode ser sintonizada pela aplicação. Essa sofisticação diz respeito ao limite mínimo de variação do valor do dado observável que se considera significativo para acionamento das regras. Assim, a aplicação, ao definir o dado observável, pode também definir um valor mínimo abaixo do qual uma variação de valor não é considerada relevante e não causa o acionamento das regras. Assim é possível sintonizar a investigação das regras em ambientes em que haja muitas variações de valores que, entretanto, sejam pouco expressivas. Caso não seja definido um valor para esse mínimo, será considerado o valor zero, isto é, qualquer alteração de valor do atributo causará a investigação das regras associadas.

4.4.3 INFERÊNCIA PARA LÓGICA DE PRIMEIRA ORDEM

Quanto às regras de lógica de primeira ordem, o RT-MLR trabalha com regras que são parecidas com a Lógica Cortês ou *Courteous Logic* proposta por Grosof [40] e já explicada na seção 3.4. Na abordagem deste trabalho é usada da proposta da Lógica Cortês apenas a priorização das regras. Entretanto, diferentemente do trabalho de Marc Dörflinger [41] onde é proposto um motor de inferências para Lógica Cortês, não são estabelecidos rótulos para as regras e precedências para os rótulos, o que, na proposta de Dörflinger, pode levar à existência de regras com a mesma precedência ou regras sem precedência (sem rótulo). Ao invés disso, nesta abordagem, as regras pertencem a um conjunto em que a precedência é explicitamente definida pelo usuário ao adicionar regras ao conjunto, ou seja, existe uma ordem obrigatória e seqüencial de prioridade para todas as regras. Na proposta de Dörflinger, quando as prioridades são iguais e as conclusões conflitantes, não é realizada nenhuma conclusão. O mecanismo de priorização obrigatória das regras resolve o problema de exclusão mútua implícita, que ocorre justamente pela existência de conclusões conflitantes, ou seja, sempre há uma conclusão, que é a da regra mais prioritária.

4.4.4 MECANISMO DE INFERÊNCIA PARA LÓGICA FUZZY

Para regras fuzzy, o mecanismo de inferência funciona de forma semelhante, mas existem algumas diferenças. O disparo de investigação das regras é executado da mesma forma, ou seja, vinculado à alteração dos valores das variáveis que aparecem nas premissas das regras.

Cada regra disparada pela alteração do valor considerado na premissa tem o seu lado esquerdo avaliado considerando-se as funções fuzzy previamente definidas e os operadores fuzzy utilizados. Para os operadores lógicos da premissa é implementada a forma sugerida por Zadeh (descrita na seção 3.3.2).

Caso a pertinência da premissa seja maior que zero, a regra é disparada, gerando uma função fuzzy de saída para cada regra. Foi implementado inicialmente o método de Mamdani para a inferência (MIN), conforme descrito na seção 3.3.2, mas outros modelos de inferência fuzzy podem facilmente ser adicionados.

Em seguida, são considerados os conjuntos previamente definidos para as variáveis de saída que aparecem na conclusão das regras. Cada conjunto associado a uma variável de saída tem as demais regras do respectivo conjunto também avaliadas, na medida em que os valores das premissas dessas regras podem não ter sofrido alteração, mas seus valores atuais podem também causar o disparo das respectivas regras e, portanto, contribuir para o cálculo do conjunto fuzzy resultante de saída.

A seguir é feita a agregação dos conjuntos fuzzy resultantes da ativação das diferentes regras acionadas (correntemente é usado o critério de MAX para agregação dos conjuntos das regras, mas teoricamente pode ser definido um critério distinto para cada conjunto de regras).

Por fim, é feita a defuzzificação do conjunto resultante da agregação das regras e é atribuído um valor escalar de saída para a variável de saída representada pelo conjunto de regras. Atualmente é usado para a defuzzificação o critério de média dos máximos, mas também pode ser definida uma forma de defuzzificação distinta para cada conjunto de regras.

4.4.5 MECANISMO DE INFERÊNCIA PARA LÓGICA TEMPORAL NO RT-MLR

O RT-MLR utiliza o conceito de Lógica Temporal de Intervalos (ITL, do acrônimo no idioma inglês), que foi originalmente definido em [47], e é usado em uma extensão de DROOLS [39] e em trabalhos correlatos [49]. Na ITL o elemento básico

usado é o evento, que pode ser visto como um fato associado a um determinado tempo. O evento é um valor lógico ou analógico com uma marca de tempo associada.

No RT-MLR a ocorrência de um evento está relacionada à mudança de valor em um dos atributos das classes do domínio que seja usado em alguma das regras e, portanto, seja definido como de uma das classes específicas fornecidas pelo RT-MLR para encapsular tipos básicos. Essa mudança de valor é armazenada internamente ao RT-MLR e torna-se uma ocorrência imutável. Ou seja, um evento é uma alteração de valor de um atributo que ocorreu em algum instante no passado cuja ocorrência está registrada e não pode ser alterada.

Naturalmente que os eventos ocorridos não são guardados eternamente. Para tanto, os eventos possuem outro tempo associado, o "intervalo de esquecimento", que é um valor de intervalo de tempo a ser considerado do instante atual para o passado, durante o qual o evento deve permanecer registrado. Eventos cuja ocorrência se deu antes do tempo atual menos o intervalo de esquecimento não são mais guardados. O intervalo de esquecimento é definido intrinsecamente pelos operadores temporais utilizados nas regras.

A partir do registro destes eventos é possível realizar inferências que incluam operações que lidam com os tempos de ocorrência. Essas operações estabelecem relações de precedência tais como antes, depois, durante, até que, etc.

O RT-MLR permite também estabelecer métricas tais como médias ou contagens de alterações efetuadas sobre um determinado período de tempo do passado. A seção seguinte descreve em detalhes a sintaxe e semântica dessas operações.

4.5 SINTAXE, SEMÂNTICA E OPERADORES LÓGICOS

Como foi previamente explicado, a sintaxe de elaboração das regras é a mesma da linguagem Java. Não há nenhuma fase de pré-processamento das regras e é utilizado apenas o compilador da própria linguagem.

Cada regra é tratada como um objeto de uma classe específica do pacote RT-MLR. Na medida em que as regras são tratadas como classes de associação entre objetos, a instanciação das regras é feita no domínio da aplicação. Ou seja, a aplicação decide quais objetos devem ser associados a quais outros com as regras previamente especificadas.

Ou seja, na abordagem do RT-MLR as regras são classes que dão origem a um conjunto de objetos específicos para associar objetos do domínio da aplicação. A

relação de associação pode ter cardinalidade um para um, um para muitos ou muitos para muitos. A cardinalidade da associação, a quantidade de objetos do domínio e o interesse de efetivamente associar alguns desses objetos é que vai definir, no domínio da aplicação, o número efetivo de instâncias da regra que serão criadas.

4.5.1 SINTAXE DA LÓGICA DE PRIMEIRA ORDEM

Suponha que se tenha uma aplicação de fusão de dados em um ambiente tático, onde existe uma classe Target e uma classe Ambient. Pode-se ter uma regra de associação de cardinalidade um (do lado da classe Ambient) para muitos (do lado da classe Target), como a abaixo descrita em linguagem natural:

If Target.type=neutral and Ambient.rain=yes Then...

Neste caso, provavelmente haverá uma instância dessa regra para cada objeto da classe *Target* que for criado e a instanciação da regra se dará a cada instanciação de *Target*.

Na mesma aplicação poder-se-ia ter uma regra de fusão que procurasse investigar se distintos objetos da aplicação se referem a um mesmo alvo real. Nesse caso, uma regra poderia relacionar objetos da classe *Target* com os demais objetos da mesma classe, em uma cardinalidade muitos para muitos. Entretanto, poderia haver interesse em definir esta regra de fusão apenas para alvos oriundos de sensores distintos. Assim, no domínio da aplicação é que se instanciariam as regras para investigar os objetos que se desejasse associar, que seriam apenas os que possuem sensores distintos, restringindo, portanto, as instâncias geradas para esta regra.

Em todos esses casos, repare que o conhecimento específico é escrito uma só vez, no momento em que se elabora a regra de negócio que é implementada pela classe de relacionamento. A cardinalidade dessa regra, entretanto, depende da cardinalidade dos objetos do domínio e da cardinalidade dos relacionamentos da regra de associação com os objetos do domínio.

Na sintaxe proposta para a premissa das regras é possível estabelecer precedências. Cada operador lógico, inclusive a inferência, é implementado como um método a ser chamado no corpo da regra. Assim, a premissa da regra é composta por chamadas aninhadas de métodos que implementam operações lógicas. O aninhamento define a precedência das operações dentro da regra.

Por exemplo, se tivéssemos a regra:

If Target.velocity>100 and Command.priority=combat Then...

A sintaxe seria:

```
class myRule1 extends FirstOrderRule {
  public myRule1 (DoubleObservableData velocity,  StringObservableData priority) {
            super (new AndB (     new Ht (velocity, 100),              // premissa
                                  new EqB (priority, "combat")         // premissa
                              ),                                       //
                    new MyConclusion1() );                             // conclusão
  }
}
```

Onde:

- FirstOrderRule é a classe do RT-MLR que deve ser extendida para implementar classes que contenham regras de Lógica de Primeira Ordem.
- DoubleObservableData e StringObservableData são tipos de dados básicos definidos no RT-MLR que devem ser herdados por atributos do domínio que sejam utilizados em regras. Essas classes implementam o padrão de projeto *Observer* e encapsulam os tipos nativos do Java *double* e *String*.
- AndB, Ht e EqB são operadores lógicos definidos no RT-MLR respectivamente para o operador lógico "E", para o operador relacional "maior do que" e para o operador relacional "igual a".
- MyConclusion1 é um objeto do domínio que contém as conclusões para a regra criada e estende uma classe do RT-MLR (FirstOrderConclusion). Esse objeto pode ter quaisquer seqüências de comandos Java desejados.

Os parâmetros do construtor da regra são livres. Assim, se desejássemos aumentar a granularidade dos parâmetros passados e ao invés de passarmos a velocidade e a prioridade passássemos os objetos que contém essas informações como atributos, poderíamos implementar da mesma regra da seguinte forma:

```
class myRule1 extends FirstOrderRule {
  public myRule1 (Target myTarget, Command command)  {
    super (new AndB (  new Ht (myTarget.velocity, 100) ,
                       new EqB (command.priority, "combat")),
            new MyConclusion1() );
  }
}
```

Neste caso, os tipos de dados dos atributos nas classes do domínio da aplicação (*velocity* e *priority*) devem ser *DoubleObservableData* e *StringObservableData*, que encapsulam os tipos nativos *double* e *String*.

Esses atributos devem ser explicitamente instanciados nas classes do domínio, como no exemplo abaixo para a variável *velocity*:

```
Class Target extends ...
  DoubleObservableData velocity =
        New DoubleObservableData();
```

A partir disso, as regras podem considerar esse atributo no corpo ou premissa da regra. Qualquer alteração nos dados de *velocity* ou *priority* no domínio da aplicação acionará a investigação de uma instância destas regras.

Opcionalmente, pode-se definir um intervalo de variação para os valores de um atributo, de forma que variações de valores dentro do intervalo não causem o disparo das regras. Desta forma, a aplicação pode evitar a investigação de regras quando houver pouca significância em uma pequena variação de valores de um determinado atributo. Neste caso o atributo deveria ser declarado chamando a seguinte sobrecarga de construtor:

```
...Class Target extends ...
  DoubleObservableData velocity =
        New DoubleObservableData(0.1);
```

O parâmetro 0,1 especifica que mudanças do valor da velocidade menores que 0,1 não causarão a ativação das regras vinculadas a este atributo.

Na parte da conclusão das regras que utilizam lógica de primeira ordem podem ser utilizados livremente os comandos e bibliotecas do Java.

A regra anteriormente descrita deverá ser complementada criando uma classe com a conclusão da regra (*MyConclusion1*). Essa classe deve estender a classe *FirstOrderConclusion* e implementar o método *response*() com os comandos da conclusão. Desta forma:

```
public MyConclusion1 extends FirstOrderConclusion{
  public void response () {
      //  … quaisquer comandos Java desejados
  }
```

As regras devem ser agrupadas em conjuntos. Esses conjuntos definem a ordem de execução das regras, estabelecendo assim uma precedência entre as regras e evitando o problema do conflito de conclusões. O conjunto e suas regras são definidos como no exemplo abaixo:

```
MySet mySet = new MyFirstOrderSet ();
mySet.addRule(new myRule1); // adicionando a primeira regra
mySet.addRule(new …); // adicionando outras regras
```

A alteração de qualquer atributo envolvido na premissa de mais de uma regra do conjunto acionará seqüencialmente as regras na ordem em que estas foram adicionadas ao conjunto.

A tabela 3 apresenta um resumo sintático e semântico dos operadores de lógica de primeira ordem disponíveis:

Nome	Retorno	Significado
EqB (A,B)	True se A= B; False se A≠B	Equal
NeB (A,B)	True se A≠ B; False se A=B	Not Equal
Ht (A,B)	True se A> B; False se A≤B	Higher than
Lt (A,B)	True se A< B; False se A≥B	Lesser than
He (A,B)	True se A≥ B; False se A<B	Higher or equal
Le (A,B)	True se A≤ B; False se A>B	Lesser or equal
And (A,B)	True ou False pela tabela verdade do E	Operador lógico E
Not (A)	True se A= False; False se A=True	Operador lógico NOT
Or (A,B)	True ou False pela tabela verdade do OU	Operador lógico OU
In (A,B)	True se A∈B False se A∉B	A pertence a B. A é um elemento e B uma coleção

Tabela 3 – Operadores de Lógica de Primeira Ordem

4.5.2 REGRAS DE LÓGICA FUZZY

As regras Fuzzy apresentam uma semântica um pouco distinta. A definição de um conjunto de regras fuzzy deve incluir todas as regras para as quais se deseje um valor de saída único após a defuzzificação. Esse valor de saída provavelmente será enviado a uma variável do domínio da aplicação. Considere, por exemplo, a seguinte regra fuzzy:

If Target.velocity is big and Target.distance is near then Target.priority is high

Para criar a regra, primeiramente deve-se criar os conjuntos fuzzy “big”, “near” e “high”. A sintaxe para criá-los é a seguinte (os suportes de velocidade são em nós, os de distância são em jardas e os de prioridade em uma escala de 0 a 10):

```
TrapezeFunctionData big =
        new TrapezeFunctionData (100,200,500,1000);
TrapezeFunctionData near =
        new TrapezeFunctionData (0,0,2000,5000);
TrapezeFunctionData high =
        new TrapezeFunctionData (7,9,10,10);
```

A sintaxe para criação da regra seria a seguinte:

```
class myRule2 extends FuzzyRule {
  public myRule2 (Target myTarget) {
    super (new AndF (
                    new IsF (myTarget.getVelocity(), big),
                    new IsF (myTarget.getDist(), near) ),
             new FuzzyConclusion(high));
  }
}
```

As regras em cuja conclusão aparece a mesma variável de saída (no caso a variável target.priority) devem ser agrupadas em um mesmo conjunto. O conjunto e suas regras são definidos como no exemplo abaixo:

```
MyFuzzySet myFuzzySet = new MyFuzzySet (Target target);
myFuzzySet.addRule(new myRule2);
myFuzzySet.addRule(new …); // outras regras
```

O acionamento de qualquer regra do conjunto myFuzzySet acionará seqüencialmente as etapas de cálculo da pertinência da premissa, cálculo da função fuzzy de saída de cada regra do conjunto, agregação da função fuzzy de saída com as demais funções fuzzy de saída das regras associadas e defuzzificação do função fuzzy final para encontrar um valor escalar de saída. Esse valor final de saída é passado como parâmetro a um método do conjunto myFuzzySet. Nesse método (que é o método response(double result)) o usuário pode, da mesma forma que na conclusão de uma regra de lógica de primeira ordem, colocar os comandos Java que desejar. Tipicamente, esse valor de defuzzificação será enviado a alguma variável do domínio da aplicação que corresponde a uma variável de saída das regras fuzzy. Para o caso da variável target.priority da regra acima, ter-se-ia:

```
public MyFuzzySet extends FuzzySet {
  Target target;
  public void MyFuzzySet (Target target) {
      this. target = target };
  public void response (double result) {
     target.setPriority(result);
     //  … outros comandos Java
  }
```

A tabela 4 apresenta os operadores de Lógica Fuzzy disponíveis:

Nome	Retorno	Significado
IsF (A,B)	$\mu_B(A)$	Pertinência do valor *A* ao conceito fuzzy *B*
NotF (A)	$1-\mu(A)$	Operador lógico NOT
AndF(A,B)	$Min(\mu(A), \mu(B))$	Operador lógico E
OrF(A,B)	$Max(\mu(A), \mu(B))$	Operador lógico OU

Tabela 4 – Operadores de Lógica Fuzzy

4.5.3 REGRAS DE LÓGICA TEMPORAL

Como citado na seção 3.2.4, o RT-MLR lida com o conceito de evento, que é a mudança de valor de um atributo associada a um determinado instante no tempo. A lógica temporal no RT-MLR é uma extensão da lógica de primeira ordem. Assim, as regras que utilizam lógica de primeira ordem podem fazer referência a eventos tanto quanto fazem referência aos atributos das classes do domínio. A diferença é que os eventos não são guardados nas classes de domínio, eles fazem parte de uma memória de ocorrências que reside no RT-MLR.

Diferentemente de outras implementações (como DROOLS, por exemplo) em que a aplicação deve definir explicitamente o evento como uma entidade a ser futuramente considerada em regras, o RT-MLR define automaticamente os eventos a partir das regras que os utilizam. A idéia é que, como as regras são sempre ativadas pela alteração dos valores e estas estão vinculadas à ocorrência dos eventos, então não é necessário criar os eventos explicitamente, o RT-MLR os cria quando fizerem parte de alguma regra que contenha um operador temporal.

O RT-MLR disponibiliza um conjunto de operadores temporais. Alguns deles produzem resultados booleanos (verdadeiro ou falso) e estão relacionados a operadores lógicos ou relacionais com algum intervalo de tempo associado. Também nesta categoria de retorno encontram-se operadores que executam comparações no tempo vinculadas aos conceitos de *antes*, *durante* e *depois*. Existem também operadores que retornam valores tais como contagens ou médias de valores em intervalos determinados de tempo.

Uma vez que os operadores apareçam nas regras, os eventos de interesse são registrados pelo RT-MLR e permanecem guardados exatamente pelo tempo em que

eles são necessários para realizar o raciocínio definido na regra (o intervalo de tempo é definido por cada operador na própria regra).

Por exemplo, na seção 3.3.1 criou-se o atributo velocidade da classe alvo para poder utilizá-lo em uma regra. Caso se desejasse tratá-lo como um evento, poder-se-ia usá-lo em uma regra do tipo:

Se velocidade do alvo ultrapassou 100 nos últimos 5 minutos E o radar foi ligado mais que 3 vezes nos últimos 10 minutos Então...

A regra poderia ser implementá-la assim:

```
class myRule3 extends FirstOrderRule {
  public myRule3 (Target myTarget,   Radar myRadar)  {
      super (new AndB (
                          new ThereWasHt(myTarget.velocity, 100, 5000,0),
                          new Gt (new CountEq (myRadar.powerOn, true, 10000,0),  3)),
              new MyConclusion3() );
   }
}
```

A tabela 5 apresenta os operadores de Lógica Temporal disponíveis:

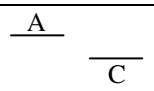
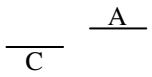
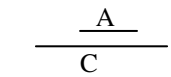
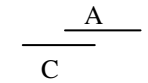
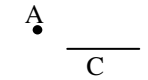
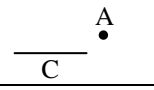
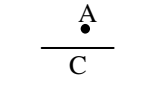
Nome	Retorno	Significado
ThereWasEq (A,B,T1,T2)	True se houve A=B no intervalo de tempo [agora–T1, agora-T2]	Equal em um intervalo do passado
ThereWasNe (A,B,T1,T2)	True se houve A≠B no intervalo de tempo [agora–T1, agora-T2]	Not Equal em um intervalo do passado
ThereWasHt (A,B,T1,T2)	True se houve A>B no intervalo de tempo [agora–T1, agora-T2]	Higher than
ThereWasLt (A,B,T1,T2)	True se houve A<B no intervalo de tempo [agora–T1, agora-T2]	Lesser than
ThereWasHe (A,B,T1,T2)	True se houve A≥B no intervalo de tempo [agora–T1, agora-T2]	Higher or equal
ThereWasLe (A,B,T1,T2)	True se houve A≤B no intervalo de tempo [agora–T1, agora-T2]	Lesser or equal
CountEq (A,B,T1,T2)	Quantidade de vezes em que houve A=B no intervalo de tempo [agora–T1, agora-T2]	Contagem de igualdade
First (A,B,T1,T2)	Tempo da primeira ocorrência de A=B no intervalo de tempo [agora–T1, agora-T2] ou zero caso não haja ocorrências.	Tempo da primeira ocorrência de um evento
Last (A,B,T1,T2)	Tempo da última ocorrência de A=B no intervalo de tempo [agora–T1, agora-T2] ou zero caso não haja ocorrências.	Tempo da última ocorrência de um evento
OcoursBefore (A,B,C,D, T1,T2)	True se há alguma ocorrência de A=B antes C=D no intervalo de tempo [agora–T1, agora-T2]	
OcoursAfter (A,B,C,D, T1,T2)	True se há alguma ocorrência de A=B depois C=D no intervalo de tempo [agora–T1, agora-T2]	
OcoursInto (A,B,C,D, T1,T2)	True se há alguma ocorrência de A=B que inicia depois e termina antes de C=D no intervalo de tempo [agora–T1, agora-T2]	
BeginsDuring (A,B,C,D, T1,T2)	True se há alguma ocorrência de A=B que inicia depois e termina depois de C=D no intervalo de tempo [agora–T1, agora-T2]	
ReachsBefore (A,B,C,D, T1,T2)	True se há alguma ocorrência de A alcançando o valor B antes de C=D no intervalo de tempo [agora–T1, agora-T2]	
ReachsAfter (A,B,C,D, T1,T2)	True se há alguma ocorrência de A alcançando o valor B depois de C=D no intervalo de tempo [agora–T1, agora-T2]	
ReachsDuring (A,B,C,D, T1,T2)	True se há alguma ocorrência de A alcançando o valor B depois de C=D no intervalo de tempo [agora–T1, agora-T2]	
Average (A,T1,T2)	Valor médio de A no intervalo de tempo [agora–T1, agora-T2]	Valor médio de variáveis numéricas

Tabela 5 – Operadores de Lógica Temporal

Com os operadores descritos na tabela anterior, é possível estabelecer raciocínios de precedência temporal entre eventos. Por exemplo, se desejasse saber se um radar foi ligado e desligado antes que outro tenha sido ligado, se poderia ter a seguinte regra:

```
class myRule4 extends FirstOrderRule {
  public myRule4 (Radar myRadar1,   Radar myRadar2)  {
       super (new OcoursBefore(myRadar1.powerOn, true,
                                        myRadar2.powerOn, true, 0, 0),
                 new MyConclusion4() );
   }
}
```

Na regra acima o tempo T1=0 representa o tempo mais antigo possível e T2=0 representa o instante atual, ou seja, o intervalo de tempo é todo aquele para o qual há registro de eventos.

5 VALIDAÇÃO DO MODELO PROPOSTO

Neste capítulo é apresentado o resultado da aplicação do modelo proposto em um problema cujo domínio é a fusão de dados em um ambiente de sistemas táticos navais. Foi escolhida a arquitetura de um sistema tático em uso em navios da Marinha do Brasil, pela facilidade de adaptação do modelo e de execução dos testes. Assim, o início deste capítulo relata o que é um sistema tático naval. Em seguida são relatadas as restrições impostas pela arquitetura do sistema existente ao uso do RT-MLR. Logo após são explicados os objetivos de fusão a serem alcançados especificamente no sistema escolhido para validação e como esses objetivos se encaixam nos níveis tradicionais de fusão de dados do modelo JDL. Na parte final do capítulo são explicitados os raciocínios lógicos aplicados para conseguir a fusão objetivada.

5.1 O CONCEITO DE SISTEMA TÁTICO

Um sistema tático, também chamado de Sistema de Comando, Controle, Comunicações, Computadores e Inteligência, abreviadamente conhecido na literatura como C^4I, é um aparato computacional que integra múltiplos computadores, sensores, equipamentos de comunicação e algoritmos. Segundo Waltz e Linnas [50], os sistemas táticos possuem várias tarefas de alto nível que podem ser classificadas da seguinte forma:

1. Sensoreamento: esta tarefa inclui a conexão do sistema com os diversos sensores disponíveis e todos os algoritmos necessários para adquirir dados, filtrá-los, identificar a existência e estimar parâmetros de objetos de interesse que se encontrem em um ambiente. Ou seja, identificar aeronaves, navios, armas (mísseis, foguetes, etc.) e sensores (radares, sonares ativos, etc.) que compõem o cenário tático.
2. Comunicação: transferência de informações entre unidades de uma força, incluindo localização e situação de acompanhamentos de interesse e dados obtidos de sensores.
3. Processamento: fusão de informações de distintos sensores com qualquer outro tipo de meta informação (bancos de dados, por exemplo), de forma a constituir um quadro tático preciso em tempo real.
4. Comando: algoritmos que facilitem a avaliação dos possíveis significados da situação tática, auxiliando as tarefas de comando.

5. Controle: tarefas que permitam o desenvolvimento e a difusão sob controle das ordens e decisões emanadas do comando.

A figura 8 ilustra o conceito do ambiente e mostra uma das telas de um sistema tipo C^4I .

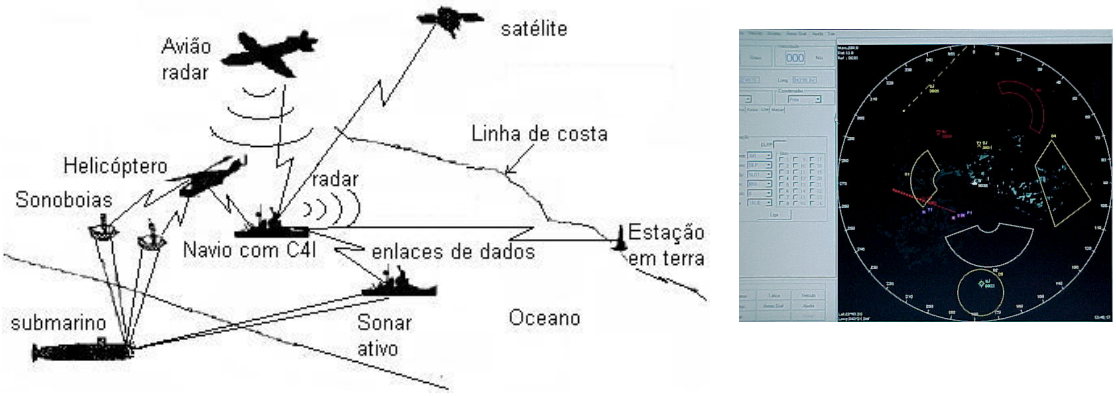


Figura 8 – Sistema C4I em um cenário tático naval

5.2 RESTRIÇÕES AO MODELO PROPOSTO NA ARQUITETURA DO SISTEMA LEGADO

A grande maioria dos sistemas de fusão de alvos desenvolvidos ou em desenvolvimento no mundo, estão no nível 1 do modelo JDL. No Brasil, particularmente na Marinha do Brasil (MB), o trabalho já realizado [51] refere-se a algoritmos de fusão medida-acompanhamento com filtro de Kalman e filtro de Kalman estendido. Os equipamentos desenvolvidos na MB utilizam esse modelo e são chamados de Extratores de Alvos Radar. Tais equipamentos fornecem como saída, a partir do sinal de entrada de um sensor radar, vários acompanhamentos que tenham sido identificados em uma determinada área definida pelo operador de algum sistema de acompanhamento tático onde o Extrator Radar esteja inserido ou em toda área coberta pelo radar, no caso em que se deseje um que o Extrator Radar realize uma identificação automática dos alvos a serem acompanhados.

Esses equipamentos estão instalados em diferentes navios, interfaceando com alguns sistemas táticos, que se pode chamar de sistemas legados. Em alguns destes sistemas táticos existe mais de um sensor radar, nestes casos é instalado um Extrator Radar para cada radar disponível.

Além dos radares, estão também disponíveis nos navios da MB dados (acompanhamentos) oriundos de outros equipamentos, tais como os recebidos pelo equipamento de comunicação de dados com outras unidades (outros navios, helicópteros ou aviões) que tenham sido detectados por sensores dessas unidades remotas. Esses acompanhamentos são chamados de acompanhamentos de link de dados ou, resumidamente, acompanhamentos link.

Em alguns casos, estão disponíveis acompanhamentos oriundos do equipamento *Interrogation Foe-Friend-Neutral* (IFFN) ou, resumidamente, acompanhamentos IFF, cuja característica principal é apresentar a identificação do acompanhamento em foco (apenas para acompanhamentos amigos e neutros).

Ainda em outros casos estão disponíveis dados de sensores sonares passivos, que estabelecem direções de onde provêem os sinais de contatos. Esses acompanhamentos são chamados de linhas de marcação, pois não identificam a posição, apenas a direção do contato estabelecido.

Também desenvolvido pela MB, existe um sistema de guerra eletrônica que identifica e classifica emissões de radares conhecidos, fornecendo informações da direção destas emissões. Essas informações são chamadas de “linhas de marcação”, na medida que informam a marcação (ou seja, a direção) em que a emissão foi percebida.

Esses sistemas legados, em vários casos podem ser alterados em relação ao sistema tático de integração das informações, mas em geral é demasiado caro ou pouco viável trocar os sensores disponíveis. O modelo a ser desenvolvido, portanto, deve respeitar as restrições objetivas do aparato tecnológico legado. Ou seja, é necessário trabalhar com um esquema de fusão que preserve, tanto quanto possível, a individualidade dos sensores sem, entretanto, perder o foco na relevância dos resultados de fusão a serem obtidos.

Assim, para aplicar o modelo proposto deve-se receber todos os contactos passados pelos diferentes sensores e tentar associá-los aos acompanhamentos já existentes. Portanto, continua existindo a fusão de informação no nível de cada sensor, que é responsável por filtrar os dados recebidos pelo sensor (fusão de nível 0), associar as medidas e fundi-las em acompanhamentos que possuem uma posição, um rumo e uma velocidade determinados (fusão de nível 1, monosensor). A arquitetura proposta é, portanto, segundo [10], do tipo 2, ilustrada na figura 9. Neste esquema, cada sensor possui seu próprio sistema de acompanhamento de alvos,

dotado de sub-sistemas para associação medida-medida e extração dos dados do acompanhamento por algoritmos do tipo Filtro de Kalman.

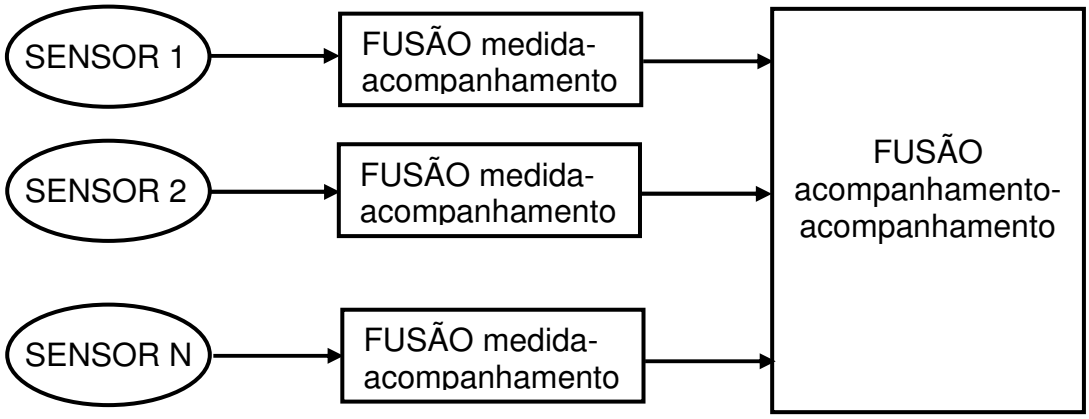


Figura 9 – Arquitetura proposta, do tipo 2a como descrita em [10]

5.3 FUSÃO DE ACOMPANHAMENTOS DE SENSORES DISTINTOS

O primeiro nível de fusão que se pode implementar com o modelo proposto é a fusão dos acompanhamentos gerados pelos sistemas de fusão individuais de cada sensor que são, nos sistemas da MB, os equipamentos chamados de Extratores Radar. Os equipamentos Extrator Radar de cada radar executam a fusão de nível zero, ou seja, fazem a filtragem do ruído, a fusão das medidas adquiridas pelo sensor com os acompanhamentos identificados e produzem um vetor de estados para cada acompanhamento com a posição e a velocidade estimadas. Esses dados são as entradas do sistema com o modelo que se está propondo, que realiza a associação acompanhamento-acompanhamento e funde os acompanhamentos individuais de cada sensor em uma informação consolidada única, ou seja, um modelo de fusão de nível 1.

Além das informações tradicionais sobre a estimativa da cinemática dos acompanhamentos, o equipamento Extrator Radar atualmente também fornece o tamanho dos acompanhamentos identificados. Esta informação também será usada no processo de fusão dos acompanhamentos de distintos radares.

Esta arquitetura de fusão de acompanhamentos de múltiplos sensores é descrita em [10] como sendo do tipo 2a, em contraposição a um outro tipo, chamado 2b. A diferença entre as duas é que na arquitetura tipo 2b é previsto um fluxo também no sentido contrário, ou seja, do sistema de fusão multisensor para o sistema de fusão

dos sensores individuais. Isto não é contemplado no modelo de arquitetura proposto pelos mesmos motivos de compatibilização com os sistemas legados, na medida em que os sistemas que estamos procurando integrar (ou seja, os equipamentos Extratores Radar) não lidam com essas informações que viriam de um nível superior porque não foram para isso projetados.

Os dados a serem usados correspondem, portanto, às informações sobre alvos que já tenham sofrido tratamento de correlação no nível de cada sensor individual, por um algoritmo adequado de fusão medida-acompanhamento, tal como um Filtro de Kalman, e que tenham redundado em representações sintéticas dos alvos (também chamados de acompanhamentos), conforme ilustrado parcialmente na Figura 10. Portanto, é suposto que os dados passados pelos sensores são confiáveis e representativos da melhor estimativa possível sobre o alvo, do ponto de vista individual do sensor.

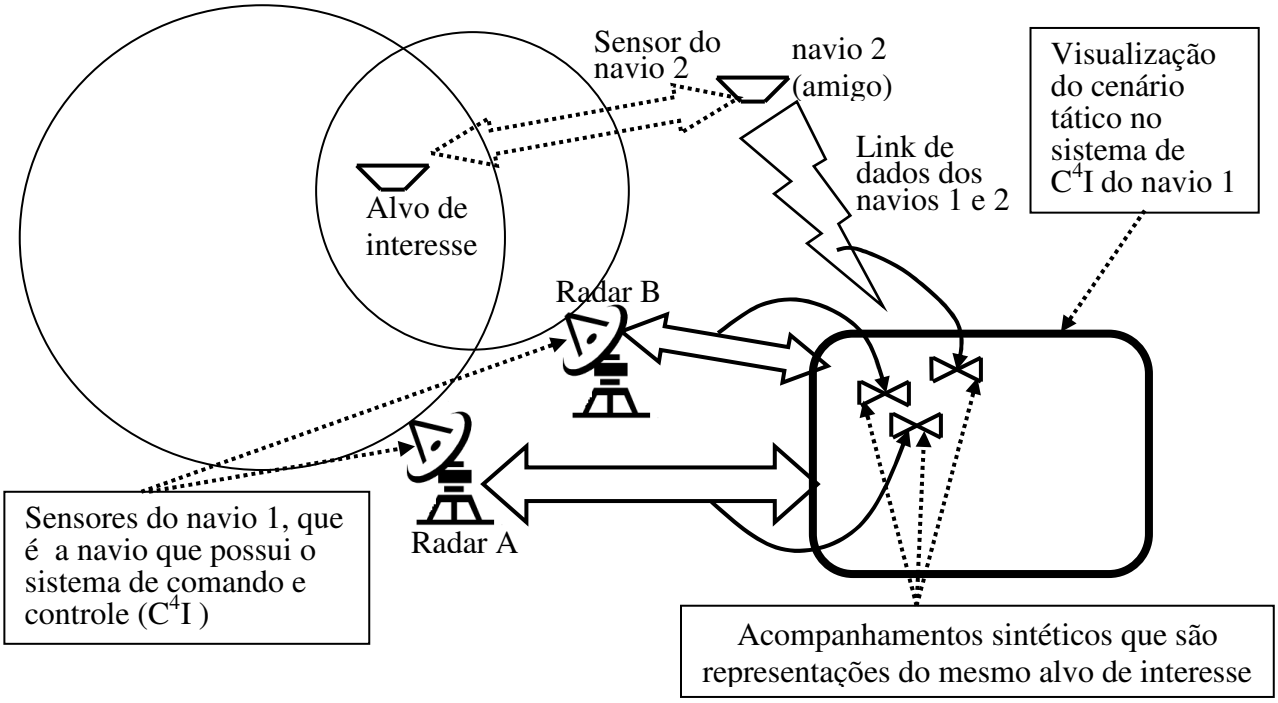


Figura 10 – Alvo, sensores e acompanhamentos em um cenário tático.

5.3.1 O EXTRATOR RADAR[1]

Como foi citado na seção anterior, o Extrator Radar é um equipamento constituído de hardware e software que recebe sinais de diversos equipamentos (entre eles o Radar propriamente dito) e, após a aplicação de alguns algoritmos de correlação e estimação, produz acompanhamentos ou "tracks" correspondentes aos alvos identificados pelo equipamento radar. As principais informações produzidas para cada acompanhamento são a posição, a velocidade e o tamanho dos acompanhamentos.

Como o processo realizado pelo Extrator radar envolve um ambiente com vários alvos, é necessário realizar um processo de associação entre as novas observações a cada varredura do radar e os acompanhamentos anteriormente criados e sendo atualmente estimados. O bloco denominado Correlação na Figura 11 é o módulo responsável por realizar esta associação. Por correlação entenda-se o processo de associação entre as medidas recebidas e um dos acompanhamentos que estejam sendo estimados pelo sistema.

Uma vez feita a correlação do acompanhamento com uma das medidas lidas, estes valores (a medida e o acompanhamento existente) são passados a um bloco de estimação, que atualmente é realizada com um Filtro de Kalman. O filtro de Kalman recebe a medida como sendo a última disponível para o acompanhamento sendo realizado e produz uma estimativa de posição e velocidade do alvo.

As informações produzidas por cada Extrator Radar são então passadas a um sistema tático que apresenta as informações recebidas de cada Extrator Radar, juntamente com outras informações de interesse como, por exemplo, o sinal de vídeo do radar digitalizado (ao que se chama de vídeo bruto radar) e outras informações sobre o contexto (como cartas náuticas, informações meteorológicas, etc.).

O filtro de Kalman também propaga a informação de estimação de posição do acompanhamento de volta para o bloco de correlação, pois esta informação é útil para definir o ponto em torno do qual será criada uma janela que limita a região onde serão procuradas as medidas que serão consideradas para a correlação do próximo ciclo.

[1] Esta seção foi adaptada de Oliveira [51]

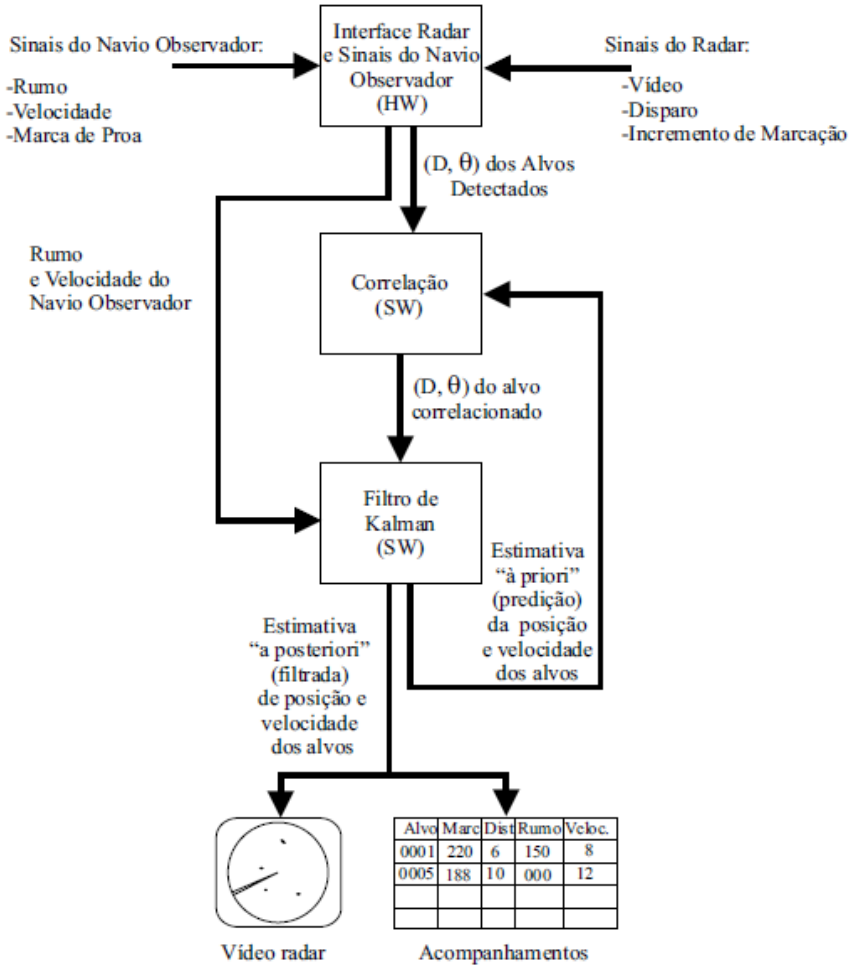


Figura 11 – Diagrama de blocos do Extrator Radar (por Oliveira[51])

5.3.2 A ESTIMAÇÃO DO FILTRO DE KALMAN[2]

Em 1960, R.E.Kalman, em seu trabalho "New Approach to Linear Filtering and Prediction Problems" lançou a base de um modelo, que ficou conhecido como Filtro de Kalman e que até hoje é uma referência na área, embora muitas alterações, extensões e composições tenham sido propostas e sejam atualmente adotadas. A idéia é considerar um sistema dinâmico e discreto no tempo, que é perturbado por um ruído Gaussiano, governado pela equação de processo:

$$s(n) = A(n)s(n-1) + \eta(n) \qquad (5.1)$$

Onde:

$s(n)$ é o vetor de estados $\in \Re^n$

[2] Esta seção foi adaptada de Oliveira [51]

$A(n)$ é a matriz de transição de estados $\in \Re^{n \times n}$

$\eta(n)$ é o vetor do ruído de processo $\in \Re^{n}$

O ruído é assumido como tendo distribuição Gaussiana e considerando E[·] como o valor esperado de [·], tem-se

$$\mathrm{E}[\eta(n)] = 0$$

$$\mathrm{E}[\eta(n)\eta^T(n)] = Q \qquad (5.2)$$

$$\mathrm{E}[\eta(k)\eta^T(j)] = 0 \quad \forall k \neq j$$

Onde Q é a matriz de covariâncias do ruído de processo.

As medidas do sistema, são modeladas como :

$$z(n) = H(n)s(n) + w(n) \qquad (5.3)$$

Onde:

$z(n)$ é o vetor de medidas (observações) $\in \Re^{n}$

$H(n)$ é a matriz de medidas $\in \Re^{p \times n}$

$w(n)$ é o vetor do ruído de medida $\in \Re^{p}$

Assume-se que os elementos de $w(n)$ têm distribuição Gaussiana e que:

$$\mathrm{E}[w(n)] = 0$$

$$\mathrm{E}[w(n)w^T(n)] = R \qquad (5.4)$$

$$\mathrm{E}[w(k)w^T(j)] = 0 \quad \forall k \neq j$$

$$\mathrm{E}[\eta(j)w^T(k)] = 0$$

Onde R é a matriz de covariâncias do ruído de medida.

Considerando-se Z^k como o conjunto das k medidas de z feitas, define-se $\hat{s}(n \mid k) = E[s(n \mid Z^k)]$ como a *n-ésima* estimativa do vetor de estados *s* do sistema, baseada nas *k* medidas feitas.

Com n-1 medidas de z e aplicando-se E[·] em (5.1), tem-se:

$$\hat{s}(n \mid n-1) = A(n)\hat{s}(n-1 \mid n-1) \qquad (5.5)$$

Onde $\hat{s}(n \mid n-1)$ é a estimativa *a priori* do vetor de estado s no instante n e $\hat{s}(n-1 \mid n-1)$ é a estimativa *a posteriori* ou corrigida do vetor s, calculada no instante anterior.

Aplicando-se E[·] em (5.3), tem-se:

$$\hat{z}(n \mid n-1) = H(n)\hat{s}(n \mid n-1) \quad (5.6)$$

Que é a estimativa *a priori* do vetor de observações z no instante n, baseada nas n-1 observações anteriores.

Ao obter-se a *n-ésima* medida de z atualiza-se a estimativa $\hat{s}(n \mid n-1)$ para obter $\hat{s}(n \mid n)$ como uma combinação linear da estimativa a priori $\hat{s}(n \mid n-1)$ com a diferença ponderada entre a medida presente z(n) e a estimativa *a priori* $\hat{z}(n \mid n-1)$.

$$\hat{s}(n \mid n) = \hat{s}(n \mid n-1) + K(n)[z(n) - H(n)\hat{s}(n \mid n-1)] \quad (5.7)$$

Onde o fator de ponderação $K(n)$ é chamado de Ganho de Kalman.

O objetivo é determinar o $K(n)$ que otimizará a equação (5.7), ou seja, obterá a melhor estimativa *a posterior*i.

Define-se o erro de estimação *a posteriori* como:

$$e(n \mid n) = s(n) - \hat{s}(n \mid n) \quad (5.8)$$

Define-se a matriz de covariâncias do erro de estimação *a posteriori* como:

$$M(n \mid n) = E[e(n \mid n)e^T(n \mid n)] \quad (5.9)$$

Define-se o erro de estimação *a priori* como:

$$e(n \mid n-1) = s(n) - \hat{s}(n \mid n-1) \quad (5.10)$$

E a matriz de covariâncias do erro de estimação *a priori* como sendo

$$M(n \mid n-1) = E[e(n \mid n-1)e^T(n \mid n-1)] \quad (5.11)$$

Substituindo (5.7) em (5.8) e o resultado em (5.9) chega-se a uma função objetivo a ser minimizada para produzir a melhor estimativa a posterior dada por (5.7). Para fazer a minimização é necessária alguma manipulação algébrica (detalhada em Oliveira [51]), que leva a:

$$K(n) = M(n \mid n-1)H^T(n)\left[\mathrm{R(n)} + \mathrm{H(n)M(n \mid n\text{ - }1)H^T}(n)\right]^{-1} \quad (5.12)$$

Para calcular $K(n)$ é necessário calcular $M(n \mid n-1)$. Substituindo (5.10) em (5.11) e após mais manipulações algébricas (também detalhadas em Oliveira [51]), chega-se a:

$$M(n \mid n-1) = A(n)M(n-1 \mid n-1)\,\mathrm{A}^{\mathrm{T}}(n) + Q \qquad (5.13)$$

E também a:

$$M(n \mid n) = \left[I - K(n)H(n)\right]M(n \mid n-1) \qquad (5.14)$$

O filtro de Kalman é um algoritmo recursivo que calcula seqüencialmente:

1) Predição do Estado (eq. 5.5)

2) Predição da Matriz de Covariâncias do Erro de Estimação (eq. 5.11)

3) Matriz de ganho de Kalman (eq. 5.12)

4) Correção da estimativa (eq. 5.7)

5) Correção da Matriz de Covariâncias do Erro de Estimação (eq. 5.14)

Como o algoritmo é recursivo, nos dois primeiros passos são estipulados valores arbitrários para as variáveis de estado e para a matriz de covariância dos erros de estimação.

5.3.3 AMBIENTE EMPREGADO PARA OS TESTES

Um pressuposto da utilização do sistema proposto nos sistemas táticos legados, como foi mencionado no início deste capítulo, é que fossem usados os valores de saída produzidos pelos equipamentos Extrator Radar atualmente utilizados nos sistemas táticos existentes na MB, inclusive com os protocolos de comunicação existentes.

Na arquitetura dos Sistemas Táticos existente nos navios da MB, existe uma Estação de Fusão de dados, que se acoplará ao sistema existente, gerando, no mesmo padrão de comunicação existente, as novas informações que impliquem em fusão de dados. No nível 1 de fusão, que é o caso da fusão de acompanhamentos de distintos sensores, a Estação de Fusão gerará acompanhamentos fundidos a partir de acompanhamento recebidos das estações de Extrator Radar. Um resumo dessa arquitetura está na figura 12.

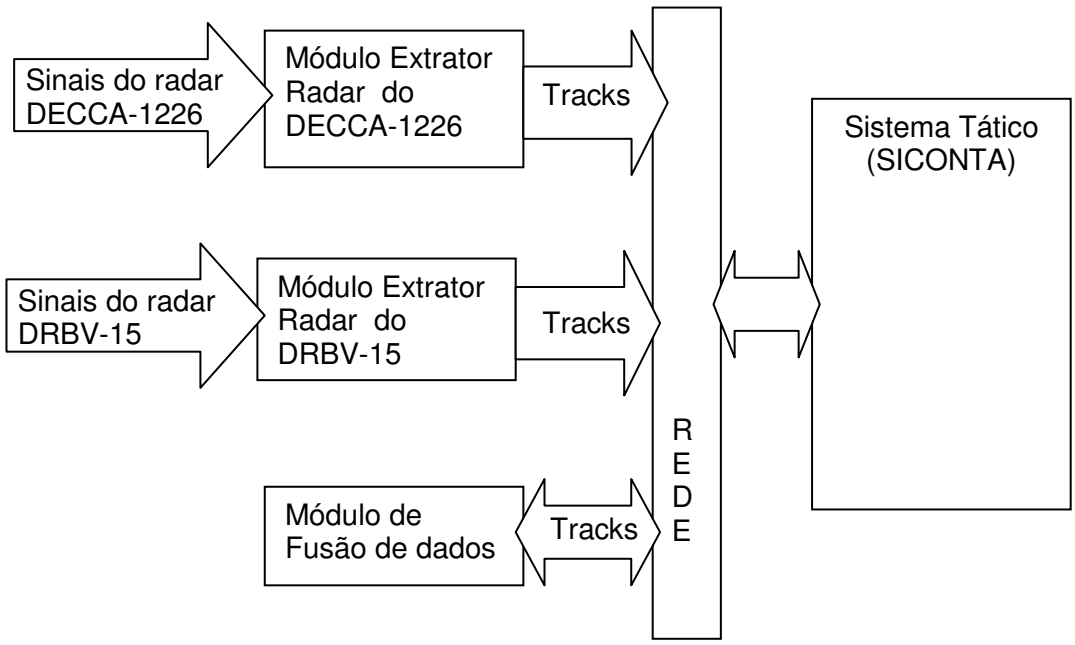


Figura 12 – O módulo de fusão agregado ao sistema legado

Para realizar os testes dentro do espírito da arquitetura proposta, mas podendo gerar alvos sintéticos com maior grau de complexidade de movimento, foi então gerado um ambiente de testes com a arquitetura descrita na figura 13.

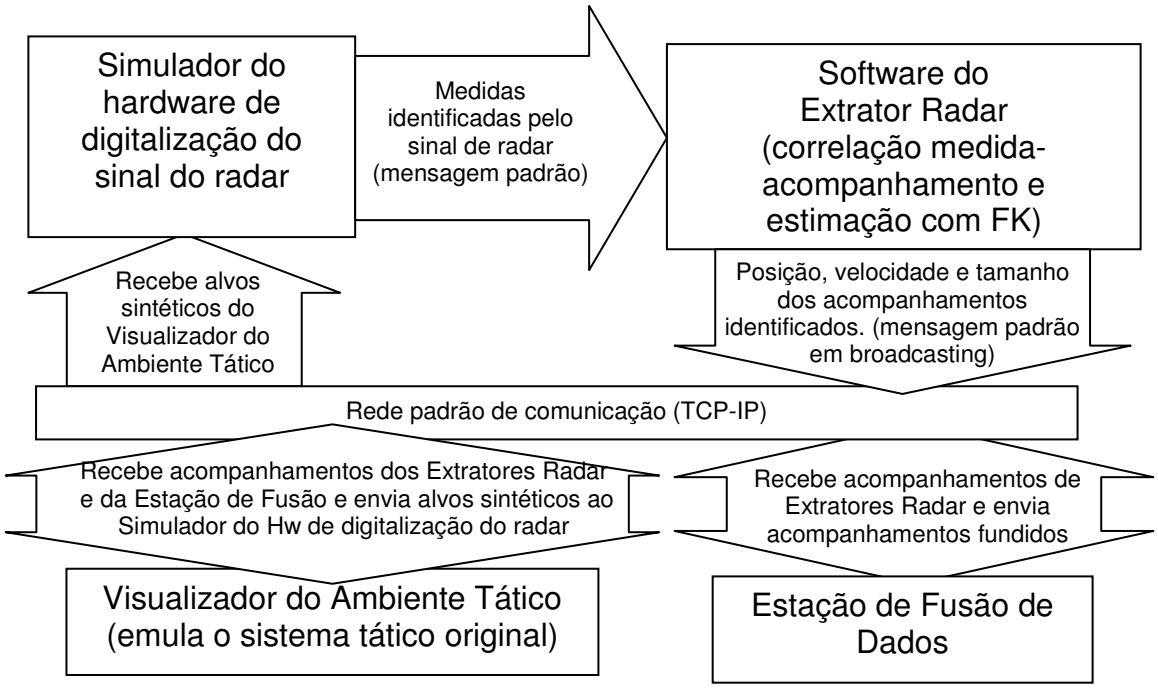


Figura 13 – Arquitetura utilizada para os testes dos modelos de fusão

Nesta arquitetura de testes, os alvos são gerados pelo operador artificialmente no Visualizador, causando a exibição de uma marca da posição do alvo, chamada de "*dot*". A posição do *dot* criado é acrescida de um ruído com distribuição de probabilidade Gaussiana escolhido pelo operador para cada sensor. As informações sobre a posição do alvo (já acrescida do ruído Gaussiano) são passadas ao simulador do hardware que as repassa ao módulo de estimação com o filtro de Kalman (Extrator Radar), que é exatamente o mesmo utilizado no sistema real. O Extrator Radar, por sua vez, cria o acompanhamento com uma posição, uma velocidade e um tamanho estimados e repassa essas informações ao módulo de Visualização (que exibe os acompanhamentos próximos ao *dot* gerado). O Extrator Radar também passa as informações sobre o acompanhamento ao módulo de Fusão, que funde os acompanhamentos de acordo com hipóteses explicitadas nas regras. Caso os acompanhamentos de diferentes módulos Extrator Radar sejam fundidos, o módulo de fusão cria e passa ao Visualizador um novo acompanhamento fundido.

Neste modelo, o framework RT-MLR e a aplicação com as regras específicas que compõem o domínio da aplicação, estão na Estação de Fusão de Dados. A estação do Extrator Radar é basicamente a mesma do sistema tático original (à exceção do fato do hardware ser simulado) e a estação de visualização emula o Sistema Tático original permitindo apenas a visualização dos objetos de interesse e a criação de alvos sintéticos desejados.

5.3.4 APLICANDO O MODELO PARA A FUSÃO DE NÍVEL 1

Dentro do que o modelo JDL classifica como fusão de nível 1, aplicou-se o RT-MLR para realizar a fusão das representações sintéticas dos alvos fornecidas por distintos sensores. Esta fusão é necessária nos casos em que diferentes sensores possuem áreas de cobertura com interseções (como é o caso de alguns radares) ou mesmo são de natureza totalmente sobreposta, como é o caso das informações recebidas pelo enlace de dados, que se sobrepõem a informações de radares, sonar e IFF.

A idéia básica é detectar a correlação das informações entre tuplas de acompanhamentos sintéticos gerados por diferentes fontes, de forma a identificar aqueles que se referem a um mesmo alvo físico real e que, dada à sobreposição de detecção existente entre os sensores, ocasionam o aparecimento de mais de uma representação sintética para o mesmo alvo real. Após a constatação da correlação, deve ser sugerida ao operador uma fusão entre os acompanhamentos identificados como representativos de um mesmo alvo.

Para realizar a associação entre acompanhamentos de sensores distintos que representem um mesmo alvo, deve-se estabelecer uma medida de similaridade entre os acompanhamentos. A teoria dos conjuntos fuzzy [42] propõe que os elementos (objetos, variáveis) pertencem, com diferentes graus de pertinência, a diferentes conjuntos fuzzy. O grau de pertinência de cada valor de uma variável a um determinado conjunto é dado por uma função de pertinência que determina o grau de associação entre o elemento e um determinado conjunto. Desta forma, o grau de pertinência pode ser visto como uma medida de correlação entre um elemento e um conjunto.

Alguns outros sistemas propostos utilizam também modelos fuzzy para fusão de dados com algum propósito semelhante ao pretendido por esta proposta, utilizando estratégias semelhantes.

Em [52] um modelo de fusão é adotado para pré-processar e classificar em um determinado grau de risco acompanhamentos oriundos de diversos sensores, incluindo informações ambientais como vento e tamanho das ondas. O sistema possui defuzzificação pelo máximo das regras acionadas e produz como saída o grau de risco de cada acompanhamento que é então fundido com dados oriundos de outros modelos, em um sistema GIS geo-referenciado.

Em [53] é proposta uma topologia de referência, centrada em cada acompanhamento, que divide o espaço próximo ao acompanhamento em anéis e setores de interesse. Baseado nessa divisão espacial é proposto um sistema fuzzy que lida com as características da densidade de acompanhamentos nos setores oriundos de diferentes acompanhamentos, no intuito de identificar erros de alinhamento (não randômicos) nos sensores envolvidos, e categorizar os acompanhamentos como pertencentes aos mesmos alvos.

Em [54 e 55] os autores propõem um modelo neuro-fuzzy (fuzzy ARTMAP) para detectar anomalias no comportamento de alvos, que possam ser indicativas de situações de risco. O ponto de destaque é a rapidez e continuidade do aprendizado, necessárias as aplicações de tempo real e a possibilidade de adaptação do algoritmo, através de intervenção do operador, às situações de baixa quantidade de dados para sintonia do modelo, podendo realizar dinamicamente a postergação no ajuste dos parâmetros.

Em [56] um modelo fuzzy foi desenvolvido para estimar as trajetórias e o formato de alvos a partir de regiões de imagens extraídas de uma imagem de vídeo. Um sistema neuro fuzzy foi então utilizado para extrair as regras

Em [57 e 58] é proposto um modelo baseado em lógica fuzzy para fundir as informações de acompanhamentos que pertençam ao mesmo alvo. Entretanto, no modelo de fuzzificação são usados suportes para as funções fuzzy obtidos de forma *ad hoc* a partir da análise dos casos de uso.

Em [59] é também proposto um modelo fuzzy para fusão de acompanhamentos em que se usou a precisão dos sensores envolvidos para determinar o suporte das funções.

O problema nas propostas de [57, 58 e 59] é que o limiar de comparação utilizado para a hipótese de fusão é arbitrário e não são dadas explicações sobre sua origem ou determinação.

Em [60] é proposto outro modelo fuzzy para fusão de acompanhamentos em que se usou um modelo de agrupamento fuzzy means (FCM) para calcular uma pertinência que serve como base de comparação das regras.

Distintamente destes trabalhos citados, neste exemplo de aplicação desenvolvido com o RT-MLR não será usada a precisão do sensor, mas uma medida da qualidade da estimativa que está constantemente sendo feita pelo filtro de Kalman.

A proposta é ajustar dinamicamente o limiar de comparação da hipótese de fusão de acordo com a matriz de covariâncias de erros de medidas que é informada pelo modelo de estimação (o filtro de Kalman) aplicado a cada sensor. Ou seja, usa-se o valor calculado de $M(n \mid n)$ definido na equação 5.14 para extrair o desvio padrão de cada atributo estimado.

A idéia básica é que a base de comparação da medida de correlação entre os acompanhamentos pode variar conforme o desempenho do Filtro de Kalman, expresso pela Matriz de Covariâncias do Erro de Estimação que está sendo calculada pelo filtro (eq. 5.14).

Na aplicação do Módulo de Fusão os acompanhamentos são objetos criados e atualizados conforme as informações são recebidas dos Extratores Radar. Esses objetos são criados de forma a funcionar com o *framework* RT-MLR. Assim, os atributos (tais como posição, velocidade e tamanho) dos acompanhamentos, são dos tipos compatíveis com o *framework* e são usados nas regras de fusão que empregam funções fuzzy. Essas regras utilizadas para a fusão são automaticamente acionadas quando há alteração de qualquer valor ligado a um atributo de acompanhamento.

Cada novo acompanhamento criado por um módulo de Extrator Radar é mantido dentro do Módulo de Fusão em uma lista de acompanhamentos específica para aquele radar. Cada vez que um novo acompanhamento é recebido, a aplicação de fusão instancia um conjunto de regras específicas para esse novo alvo. Cada objeto desse conjunto contém, portanto, uma regra que relaciona especificamente o acompanhamento criado com todos os demais acompanhamentos existentes nas listas dos radares (exceto a lista do próprio Extrator Radar que criou o acompanhamento). Ou seja, a aplicação determina que sejam futuramente investigadas as hipóteses de fusão apenas de acompanhamentos gerados por Extratores Radar distintos. Isso é feito porque se assume que cada Extrator Radar já realizou o trabalho de associação e estimação ao nível do sensor e que, portanto, acompanhamentos distintos informados pelo Extrator Radar referem-se efetivamente a alvos distintos identificados pelo conjunto Associador/Estimador. Assim, não faz sentido tentar associar acompanhamentos distintos do mesmo sensor.

5.3.4.1 MODELAGEM PARA FUSÃO DE ACOMPANHAMENTOS

Antes de testar a hipótese de fusão, é necessário lembrar que os dados recebidos dos diversos sensores são assíncronos. Isto ocorre porque cada radar tem uma particular freqüência de rotação e os dados de cada alvo são enviados cada vez que o feixe radar ilumina o alvo, o que ocorre de acordo com a freqüência de rotação da antena de cada radar.

As regras são acionadas quando há alteração do valor de um atributo de algum acompanhamento. Se este acompanhamento for chamado de *A,* então serão disparadas todas as instâncias de regras que associam *A* aos demais acompanhamentos dos demais sensores. Chame-se de *B* um desses acompanhamentos oriundo de outro sensor. Então, os atributos de *B* devem ser alinhados no tempo antes da investigação da hipótese de fusão. Isto é feito pela chamada de um método de alinhamento existente no objeto *B* da classe do domínio. O alinhamento é feito a partir de uma base de tempo global que é mantida no sistema por mensagens específicas de sincronização de tempo. Utilizando o tempo global e o tempo de recebimento da última atualização do acompanhamento *B*, é feita uma extrapolação da posição com a velocidade corrente. Essa extrapolação não causa a atualização do atributo de posição no objeto B, servindo apenas como referência de comparação mais atual para a regra que testa a hipótese de fusão.

A regra de associação utiliza os atributos alinhados no tempo para investigar a correlação dos acompanhamentos e a hipótese de fusão. Os seguintes atributos de cada acompanhamento são considerados: posição, velocidade, rumo e tamanho.

A medida de correlação é dada pela pertinência de uma função fuzzy. Todas as variáveis de entrada serão mapeadas em conjuntos representativos da diferença em módulo das grandezas em análise, para cada dois acompanhamentos do sistema oriundos de sensores diferentes (após o alinhamento temporal). O objetivo é que os acompanhamentos sejam tomados dois a dois e que o sistema determine se estes dois acompanhamentos se referem ao mesmo alvo.

Se P_A designa a posição do acompanhamento A, oriundo de um sensor e P_B designa a posição do acompanhamento B devidamente alinhada para o mesmo instante de tempo, então

$$\Delta_p = | P_A - P_B| \qquad (5.15)$$

Designa a distância em módulo da posição informada por cada Extrator Radar respectivamente para os acompanhamentos A e B. Observe-se que essa distância é calculada para cada uma das coordenada cartesianas x e y em que se decompõe a posição, ou seja, são calculados Δ_{px} e Δ_{py}.

A mesma diferença é calculada para os demais atributos de interesse, desta forma:

$\Delta_V = | V_A - V_B|$ é a diferença de velocidade em módulo (5.16)

$\Delta_R = | R_A - R_B|$ é a diferença de rumo em módulo (5.17)

$\Delta_T = | T_A - T_B|$ é a diferença de tamanho em módulo (5.18)

Os conjuntos Fuzzy de interesse são modelados como proposto na Figura 14 para cada uma das características de entrada Δ.

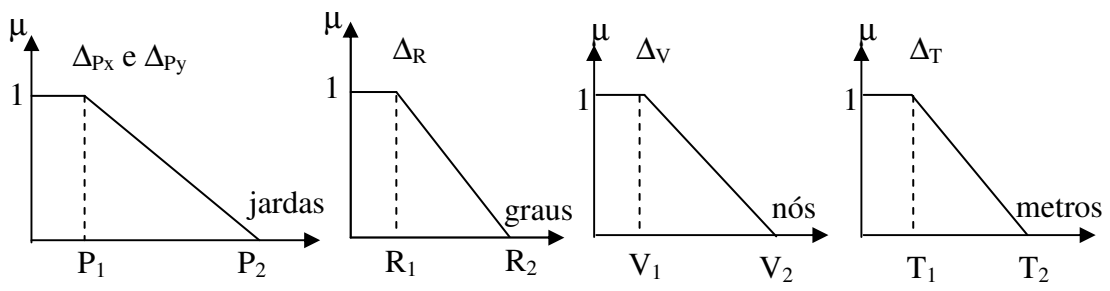


Figura 14 – Conjuntos Fuzzy das variáveis de entrada

Os suportes das funções da figura 14 são definidos a partir da precisão dos equipamentos envolvidos. O patamar próximo a zero que fornece μ=1 tem a largura da precisão do equipamento, ou seja, P_1, R_1, V_1 e T_1 dependem da precisão do equipamento. Os valores que determinam o final do suporte da funções Fuzzy são também escolhidos com base na precisão como 3 a 4 vezes maiores que o valor da precisão (3 vezes para medidas diretas como a posição, que tende a ser mais precisa e 4 vezes para medidas somente estimadas como a velocidade e o rumo.

Os valores de pertinência obtidos são medidas de similaridade entre os acompanhamentos. Esses valores farão parte da regra de associação que testa a hipótese de fusão.

Entretanto, antes de se aplicar a regra de fusão, é necessário calcular o tamanho do alvo. Isto porque o tamanho não é diretamente fornecido pelo Extrator Radar. O que este fornece é um contador de quantas vezes o alvo foi iluminado por sucessivos feixes de emissão do sinal de radar durante uma varredura da antena. Este valor está relacionado à seção reta radar do alvo, que por sua vez depende da capacidade de reflexão do material empregado, à geometria do alvo e ao seu tamanho. Não existe uma forma específica de calcular a influência destes parâmetros, mesmo para alvos com materiais e geometria conhecidos. O que existem são tabelas empíricas, levantadas a partir de experimentos e que relacionam a seção reta radar com alvos típicos [61]. A partir delas é possível perceber que a seção reta radar, e conseqüentemente o valor do contador de emissões com retorno, estão fortemente relacionados com o tamanho do alvo e a distância deste ao radar.

A relação pode se percebida pela observação da figura 15. Nela pode-se perceber que alvos com o mesmo tamanho serão iluminados pelo feixe do sinal de radar menos vezes à medida que estejam mais distantes da fonte de emissão.

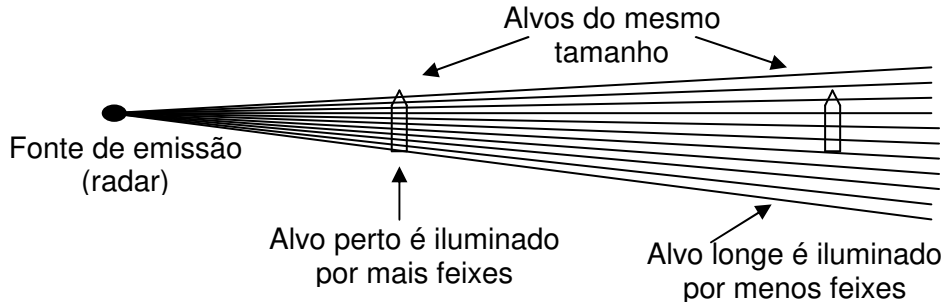
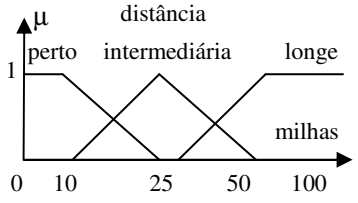
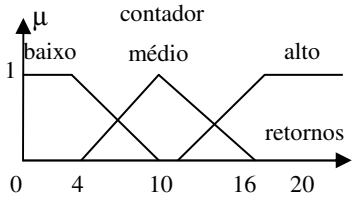


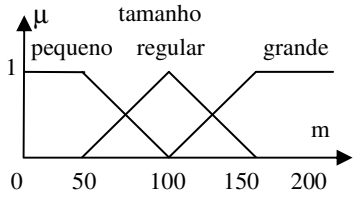
Figura 15 – Variação da quantidade de retornos do sinal de radar para alvos de mesmo tamanho em distâncias diferentes

Assim, para que se tenha uma estimativa do tamanho do alvo pode-se utilizar uma conjunto de regras fuzzy para calcular o tamanho do alvo a partir das variáveis de entrada de *número de retornos* de feixes de emissão do radar e *distância* dos alvos à fonte de emissão, ambas fornecidas pelo Extrator Radar.

As regras e os conjuntos fuzzy empregados para este cálculo estão ilustrados na figura 16.







c \ d	perto	interm.	longe
Bx	Pq	Pq	Rg
Md	Rg	Rg	Gr
Al	Rg	Gr	Gr

Figura 16 – Conjuntos e regras fuzzy para cálculo do tamanho

Ao ser recebido um valor de contador de retornos de emissão do feixe de radar e o valor correspondente à distância do acompanhamento, algumas das 9 regras fuzzy do conjunto da figura 16 serão acionadas, a defuzzificação será procedida e o valor final do tamanho do alvo será calculado.

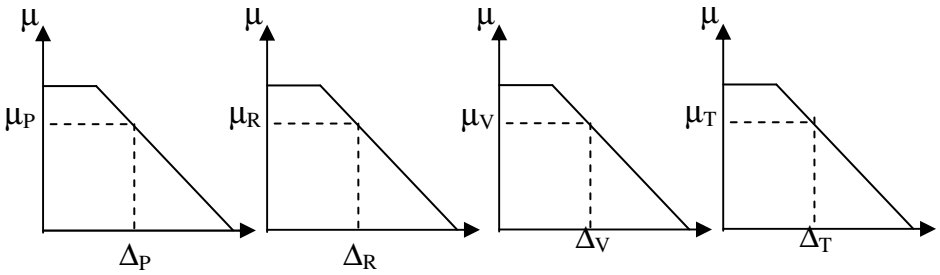
De posse do tamanho dos alvos pode-se entrar com esse valor na função fuzzy mais à direita na figura 14 e obter a medida de similaridade do tamanho dos dois acompanhamentos.

Finalmente, de posse de todas as medidas de similaridade para as distintas características dos acompanhamentos, pode-se aplicá-las na regra de associação que testa a hipótese de fusão (figura 17).

Esta é uma regra de lógica de primeira ordem que utiliza valores fuzzy e compara as medidas de similaridade encontradas para cada característica com um limiar fornecido pela estimativa do filtro de Kalman.

A regra avalia uma hipótese binária h, da seguinte forma:

$$h = \begin{cases} 0, & \text{se os dois acompanhamentos não são o mesmo alvo} \\ 1, & \text{se os dois acompanhamentos são o mesmo alvo} \end{cases}$$



Onde:
Δ_P é o módulo da diferença da posição que, na prática, é dividida em Δ_{Px} e Δ_{Py}

Figura 17 – Conjuntos fuzzy usados para testar a hipótese de fusão

A regra que testa a hipótese h é definida como:

Se $\mu_{Px} > \mu(\sigma_{Px})$ *E* $\mu_{Py} > \mu(\sigma_{Py})$ *E* $\mu_R > \mu(\sigma_R)$ *E* $\mu_V > \mu(\sigma_V)$ *E* $\mu_T > \mu(\sigma_T)$

ENTÃO h = 1 (5.19)

Os diferentes valores de σ utilizados para estabelecer os limiares de comparação de cada atributo são os desvios padrão variantes no tempo e calculados a partir da Matriz de Covariâncias do Erro de Estimação informada pelo filtro de Kalman (equação 5.14)

No caso da regra de fusão indicar que dois acompanhamentos são o mesmo alvo, um acompanhamento fundido é gerado para o sistema de visualização, com a estimativa fornecida pelo Extrator Radar com a maior precisão. Esta opção foi escolhida porque trabalhos anteriores [60, 62] mostram que para sensores com diferentes precisões em geral a fusão de características resulta em uma estimativa pior do que a fornecida pelo sensor de maior precisão.

No caso de teste foram empregados dados de dois sensores. Para o caso de haver mais de dois sensores, a hipótese de fusão seria testada para os sensores dois a dois. Os acompanhamentos fundidos teriam então a hipótese de fusão testada entre eles, de forma a gerar novas fusões (por exemplo três sensores detectando o mesmo alvo) antes de gerar o acompanhamento fundido final para o sistema de visualização.

5.3.4.2 RESULTADOS DOS TESTES DE FUSÃO NÍVEL 1

Para testar a regra descrita na seção anterior foram elaborados três cenários de teste com a arquitetura descrita na figura 13. Os testes elaborados procuram

contemplar um bom balanceamento entre casos em que a hipótese é verdadeira e casos em que a hipótese é falsa. Adicionalmente procurou-se criar cenários em que ocorressem situações em que a classificação fosse mais fácil em alguns instantes e mais difícil em outros. Naturalmente também foi feita a variação dos erros de processo e de medida para os distintos sensores e foram tabulados os resultados em tabelas de contingência. A tabela de contingência é uma matriz de confusão para o caso de haver apenas duas classes, que é exatamente a situação proposta, na medida em que a hipótese binária avaliada pela regra conduz à decisão de fundir os acompanhamentos ou não fundi-los.

No meio naval são utilizados diversos sistemas de coordenadas. As coordenadas geográficas (latitude e longitude) são as mais usadas. Internamente, os sistemas táticos em geral utilizam um sistema de coordenadas cartesianas, chamado de coordenadas de mundo. Esse sistema é utilizado para referenciar os objetos relativamente no cenário tático e passar informações por enlace de dados a outras plataformas (nesse caso a origem do sistema de coordenadas é definida secretamente entre as unidades que se comunicam para dificultar a identificação). Adicionalmente, também é utilizado um sistema de coordenadas polares (marcação e distância) com uma referência e uma orientação distintas daquele que normalmente é utilizado no meio de engenharia (figura 18). As referências de rumo dos objetos, da corrente e dos ventos, são fornecidas nesse sistema de coordenadas polares. Neste trabalho considera-se a posição do próprio navio que possui os radares como sendo a origem dos sistemas de coordenadas de mundo e de coordenadas polares.

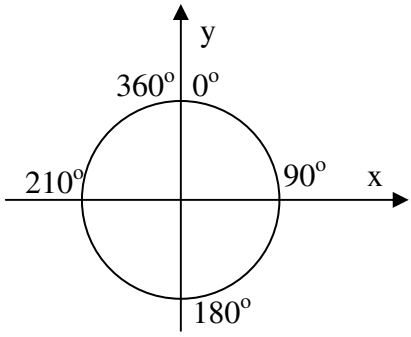


Figura 18 – Sistema de coordenadas de mundo e polares no meio naval

O primeiro cenário avaliado é composto de dois alvos, percorrendo rotas que se cruzam, sendo que o cruzamento dos alvos nas rotas foi realizado a uma curta distância e possuindo o mesmo módulo de velocidade (30 nós, que é a velocidade de

um alvo naval rápido). Foram realizados dois testes com este cenário, variando-se o ruído associado aos alvos, de acordo com a tabela 6.

O teste se iniciou com os alvos a uma distância de uma milha, um navegando no rumo 210º e o outro no rumo 135º. O teste se encerrou quando as rotas dos alvos haviam de cruzado e estes haviam percorrido posteriormente uma distância semelhante à distância inicial percorrida antes do cruzamento. A figura 19 ilustra o caminho percorrido pelos alvos.

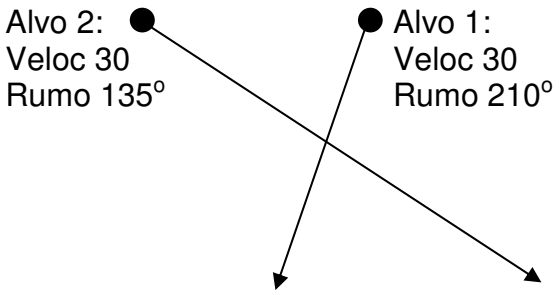


Figura 19 – Rota dos alvos no teste 1

A tabela 6 ilustra as condições iniciais do teste.

		ALVO 1	ALVO 2
Posição	X	3,32	4,32
	Y	4,55	4,55
Velocidade		30 nós	30 nós
Rumo		210º	135º
Tamanho		100 m	100 m

Tabela 6 – Posição, rumo, velocidade e tamanho dos alvos no cenário 1

No teste 1 foi utilizado um ruído Gaussiano menor para os dois sensores (embora distinto entre eles) e no teste 2 o desvio padrão do ruído foi triplicado (tabela 7).

	DRBV 1510		DECCA canal de superfície	
	TESTE 1	TESTE 2	TESTE 1	TESTE 2
Desvio padrão do ruído associado	4 m	12m	8 m	24m

Tabela 7 – Ruído associado à posição dos alvos no cenário 1

Quando o operador cria na Estação de Visualização os dois acompanhamentos, eles são enviados às duas estações de Extrator Radar (radar DRBV 1510 e radar DECCA). Cada Extrator Radar cria dois acompanhamentos individuais e os repassa à Estação de Fusão. Assim, foram criados quatro acompanhamentos distintos na Estação de Fusão. Cada acompanhamento criado gerou uma regra de teste da hipótese de fusão, existindo, portanto, quatro instâncias da mesma regra. As instâncias da regra de fusão são acionadas a cada instante em que há a atualização de um parâmetro dos acompanhamentos por cada Extrator Radar.

Como resultado esperado, deseja-se que a arquitetura de fusão identifique os acompanhamentos gerados pelos Extratores Radar que dizem respeito ao mesmo alvo e funda-os. Deseja-se também que o mesmo não ocorra para os acompanhamentos que não são relativos aos mesmos alvos. Ou seja, no total duas regras devem confirmar a hipótese de fusão e duas devem negá-la.

A tabela de contingência utilizada para os testes possui o aspecto indicado na tabela 8.

	Fundidos	Não fundidos
Fundir	V_P	F_N
Não fundir	F_P	V_N

Tabela 8 – Tabela de contingência para as hipóteses de fusão e não fusão

Com os valores de V_P, F_N, F_P e V_N são então calculados os índices de avaliação dos resultados da hipótese h descritos na tabela 9.

Erro total $erro(h) = \frac{F_P + F_N}{n}$	Confiabilidade positiva $pconf(h) = \frac{V_P}{V_P + F_P}$	Suporte $\sup(h) = \frac{V_P}{n}$	Sensitividade $sens(h) = \frac{V_P}{V_P + F_N}$
Precisão total $prec(h) = \frac{V_P + V_N}{n}$	Confiabilidade negativa $nconf\ (h) = \frac{V_N}{V_N + F_N}$	Cobertura $cob(h) = \frac{V_P + F_P}{n}$	Especificidade $esp(h) = \frac{V_N}{F_P + V_N}$

Tabela 9 – Métricas de avaliação da hipótese de fusão h

A tabela 10 aponta o resultado dos testes 1 e 2 para o cenário 1. A tabela expressa a quantidade de decisões instantâneas tomadas pela avaliação das quatro instâncias das regras durante o curso da corrida. O Extrator do radar DRBV 1510 atualiza os dados de acordo com a rotação de sua antena, a cada 2,5 segundos. O Extrator do radar Decca o faz a cada 2 segundos.

	Fundidos	Não fundidos
Fundir	624	6
Não fundir	0	628

erro(h)	*pconf(h)*	*sup(h)*	*sens(h)*
0,004	1	0,496	0,990
prec(h)	*nconf(h)*	*cob(h)*	*esp(n)*
0,995	0,991	0,496	1

(a) teste 1 (ruído menor)

	Fundidos	Não fundidos
Fundir	808	72
Não fundir	2	878

erro(h)	*pconf(h)*	*sup(h)*	*sens(h)*
0,042	0,998	0,459	0,918
prec(h)	*nconf(h)*	*cob(h)*	*esp(n)*
0,958	0,924	0,460	0,998

(b) teste 2 (ruído maior)

Tabela 10 – Resultados do teste 1 (a) e do teste 2 (b) do cenário 1

Os baixos valores de erro e os altos valores de precisão mostram que o modelo está fundindo os acompanhamentos que deveriam ser fundidos. O alto valor de confiabilidade positiva indica que dos alvos que foram fundidos quase todos deveriam realmente ter sido. O alto valor de confiabilidade negativa indica que dos alvos que não foram fundidos quase todos não deveriam realmente ter sido. Combinando a alta confiabilidade positiva com a cobertura próxima de 0,5 se tem um indicativo de que a amostra está bem balanceada entre acompanhamentos a serem fundidos e acompanhamentos que não devem ser fundidos. Os valores de suporte próximos aos de cobertura indicam que quase todos os acompanhamentos que deveriam ser fundidos, efetivamente foram. Valores altos para sensitividade indicam que quase todos os acompanhamentos que deveriam ser fundidos realmente o foram. A especificidade alta indica que quase todos os acompanhamentos que não deveriam ser fundidos efetivamente não o foram.

O segundo cenário avaliado é composto de dois alvos, percorrendo rotas que possuem uma sobreposição em termos de direção e sentido, sendo que o cruzamento dos alvos nas rotas foi realizado porque um dos alvos possui velocidade bem maior que o segundo. A situação poderia caracterizar um helicóptero voando a baixa altitude e passando por cima de uma embarcação, ambos com o mesmo rumo. O helicóptero voa a 50 nós de velocidade e a embarcação está a 20 nós. Para dificultar o teste, ambos possuem o mesmo tamanho pequeno. Foram também realizados dois testes com este cenário, variando-se o ruído associado aos alvos, de acordo com a tabela 6.

O teste se iniciou com os alvos a uma distância de 0,5 milhas, ambos navegando no rumo 0º. O teste se encerrou quando as rotas dos alvos haviam de cruzado e estes haviam percorrido posteriormente uma distância semelhante à distância inicial percorrida antes do cruzamento. A figura 20 ilustra o caminho percorrido pelos alvos.

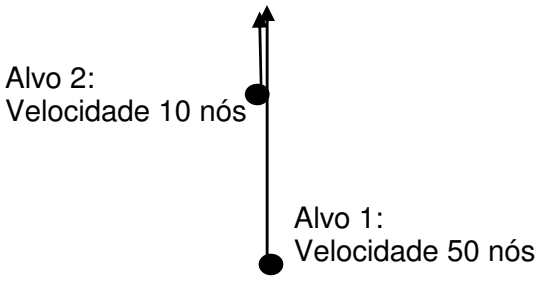


Figura 20 – Rota dos alvos no cenário 2

A tabela 12 ilustra as condições iniciais do teste.

		ALVO 1	ALVO 2
Posição	X	3,44	3,44
	Y	2,23	2,73
Velocidade		50 nós	10 nós
Rumo		0º	0º
Tamanho		20 m	25 m

Tabela 12 – Posição, rumo, velocidade e tamanho dos alvos no cenário 2

No teste 3 foi utilizado um ruído Gaussiano menor para os dois sensores e no teste 4 o desvio padrão do ruído foi aumentado para ambos os sensores (tabela 13).

	DRBV 1510		DECCA canal de superfície	
	TESTE 3	TESTE 4	TESTE 3	TESTE 4
Desvio padrão do ruído associado	4 m	20m	4m	20m

Tabela 13 – Ruído associado à posição dos alvos no cenário 2

A tabela 14 aponta o resultado dos testes 3 e 4 para o cenário 2. A tabela expressa a quantidade de decisões instantâneas tomadas pela avaliação das quatro instâncias das regras durante o curso da corrida.

	Fundidos	Não fundidos
Fundir	390	34
Não fundir	0	424

erro(h)	*pconf(h)*	*sup(h)*	*sens(h)*
0,019	1	0,481	0,962
prec(h)	*nconf(h)*	*cob(h)*	*esp(n)*
0,981	0,963	0,481	1

(a) teste 3

	Fundidos	Não fundidos
Fundir	1314	52
Não fundir	0	1366

erro(h)	*pconf(h)*	*sup(h)*	*sens(h)*
0,040	1	0,456	0,920
prec(h)	*nconf(h)*	*cob(h)*	*esp(n)*
0,956	0,926	0,456	1

(b) teste 4

Tabela 14 – Resultados do teste 3 (a) e do teste 4 (b) do cenário 2

O terceiro cenário avaliado é composto de dois alvos manobrando em uma rota circular, percorrendo rotas que se cruzam. A situação poderia caracterizar um cenário mais complexo na medida em que é esperada uma variação maior para o rumo dos acompanhamentos manobrantes estimado pelo Filtro de Kalman. Foram também realizados dois testes com este cenário, variando-se o ruído associado aos alvos, de acordo com a tabela 10.

O teste se iniciou com os alvos a uma distância de 0,5 milhas, ambos navegando no rumo 0^o, mas um guinando para bombordo e o outro guinando para boreste. O teste se encerrou quando as rotas dos alvos haviam de cruzado e estes haviam percorrido posteriormente uma distância semelhante à distância inicial percorrida antes do cruzamento. A figura 21 ilustra o caminho percorrido pelos alvos.

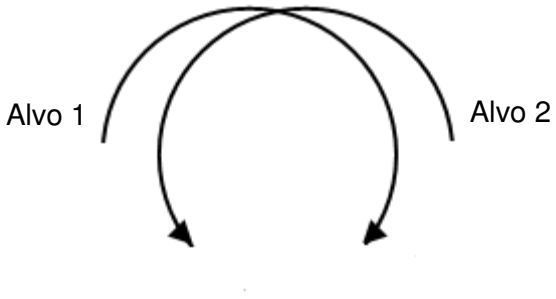


Figura 21 – Rota dos alvos no cenário 3

A tabela 15 ilustra as condições iniciais do teste.

		ALVO 1	ALVO 2
Posição	X	2,30	2,80
	Y	1,25	1,25
Velocidade		30 nós	30 nós
Rumo		0^o	0^o
Taxa de guinada		100^o/min	100^o/min
Guinada por		Boreste	Bombordo
Tamanho		100 m	100 m

Tabela 15 – Posição, velocid., guinada e tamanho dos alvos no cenário 3

No teste 5 foi utilizado um ruído Gaussiano menor para os dois sensores e no teste 6 o desvio padrão do ruído foi aumentado para ambos os sensores (tabela 16).

	DRBV 1510		DECCA canal de superfície	
	TESTE 5	TESTE 6	TESTE 5	TESTE 6
Desvio padrão do ruído associado	4m	20m	4m	20m

Tabela 16 – Ruído associado à posição dos alvos no cenário 3

A tabela 17 aponta o resultado dos testes 5 e 6 para o cenário 3. A tabela expressa a quantidade de decisões instantâneas tomadas pela avaliação das quatro instâncias das regras durante o curso da corrida.

	Fundidos	Não fundidos
Fundir	788	6
Não fundir	0	793

erro(h)	*pconf(h)*	*sup(h)*	*sens(h)*
0,004	1	0,497	0,992
prec(h)	*nconf(h)*	*cob(h)*	*esp(n)*
0,996	0,992	0,497	1

(a) teste 5

	Fundidos	Não fundidos
Fundir	830	37
Não fundir	0	844

erro(h)	*pconf(h)*	*sup(h)*	*sens(h)*
0,022	1	0,485	0,957
prec(h)	*nconf(h)*	*cob(h)*	*esp(n)*
0,978	0,958	0,485	1

(b) teste 6

Tabela 17 – Resultados do teste 5 (a) e do teste 6 (b) do cenário 3

5.3.4.3 TESTANDO COM DADOS REAIS

Para o modelo de fusão de acompanhamentos anteriormente descrito, foram também realizados testes com dados reais. Os dados utilizados são o resultado de gravações realizadas na costa brasileira. Os dados se referem a um equipamento Extrator Radar fabricado pelo Instituto de Pesquisas da Marinha (IPqM) nos moldes do anteriormente descrito na seção 5.3.1, ligado a um radar de navegação Scanter 1000. A esses dados adicionaram-se os dados gravados pelo sistema para o radar de direção de tiro (RTN30X) do referido navio. O radar de direção de tiro é muito mais preciso que o radar Scanter 1000, na medida em que não executa uma varredura circular, ficando sempre apontado diretamente para o alvo focado. Por este motivo, além de ter muito mais precisão, também tem uma freqüência de geração de informações muito maior.

Os dados são de testes de sistemas reais e por serem sigilosos não podem ser divulgados, entretanto pode-se dizer que correspondem ao acompanhamento de um navio navegando por vinte minutos entre 10 e 20 nós, durante um percurso aproximado de 7,5 milhas com uma guinada de 90^o no meio do percurso. Durante o mesmo teste, também foram gravados pelo Extrator Radar dados um outro navio, navegando a uma distância de 5 milhas do navio alvo e em um curso de afastamento praticamente ortogonal em relação a este. A existência desse navio no teste permitiu o equilíbrio de casos com hipótese de fusão positiva e negativa.

Os dados gravados durante o teste não incluem a variância do erro de estimação calculada pelo filtro de Kalman do Extrator Radar do Scanter 1000, pois esses dados não são normalmente fornecidos para o sistema de controle tático. Além disso, o radar de direção de tiro não está conectado a um equipamento Extrator Radar do IPqM, pois tem seu próprio algoritmo proprietário de rastreamento dos alvos. Desta forma, não é possível, para esses dados reais gravados, calcular o limite de comparação que leva em consideração a performance da estimação feita pelo filtro de Kalman (eq. 5.14). Como é um caso real, também não se sabe quais são o erro de medida e o erro de processo associados.

Para ser possível aplicar o modelo nestas condições, consideraram-se as informações do radar de direção de tiro como referência da posição real do acompanhamento (pois este radar é efetivamente muito preciso). A partir disso, calculou-se externamente para a massa de dados disponível a variância do erro entre a estimativa fornecida pelo Extrator Radar do Scanter 1000 e as medidas fornecidas pelo RTN30X. Os desvios padrão para a posição, a velocidade e o rumo foram então usados como entrada no modelo para calcular os limiares de comparação ($\mu(\sigma_{Px})$, $\mu(\sigma_{Py})$, $\mu(\sigma_R)$ e $\mu(\sigma_V)$) utilizados na regra (5.19) que avalia a hipótese de fusão.

Como também não há gravação sobre a quantidade de retornos do sinal de radar, também não é possível utilizar o modelo fuzzy de cálculo do tamanho do alvo e esse item foi retirado da regra que avalia a hipótese de fusão.

Os dados estão em arquivos texto, com a informação de tempo de aquisição adicionada ao restante dos dados estimados sobre os alvos. Para utilizar essa informação foi feita uma adaptação no esquema de simulação utilizado para os demais testes e ilustrado na figura 13. Foi implementado um módulo que lê o arquivo com os dados e gera mensagens seqüenciais que foram enviadas para o módulo Estação de Fusão, como se fossem adquiridos por módulos de Extrator Radar independentes. Assim, foram dispensados neste teste os módulos do Extrator Radar, do Simulador do Hardware e do Visualizador.

A tabela 18 ilustra o resultado do teste.

	Fundidos	Não fundidos
Fundir	556	13
Não fundir	0	549

erro(h)	*pconf(h)*	*sup(h)*	*sens(h)*
0,001	1	0,503	0,995
prec(h)	*nconf(h)*	*cob(h)*	*esp(n)*
0,978	0,977	0,503	1

Tabela 18 – Resultados do teste com dados reais

Os casos em que os dados do mesmo alvo não foram fundidos ocorreram no momento da guinada, em que houve uma grande variação no rumo estimado e no início da corrida quando houve a presença de algum *clutter*, o que ocasionou a geração pelo Extrator Radar de alguns pontos de *outliers* antes da estabilização da estimação.

5.3.5 APLICANDO O MODELO PARA A FUSÃO DE NÍVEL 2

Dentro do que o modelo JDL classifica como fusão de nível 2, aplicamos o RT-MLR para realizar a fusão de acompanhamentos com identificação desconhecida com informações do cenário geográfico. A idéia é considerar uma característica do cenário geográfico e associá-la a comportamentos dos acompanhamentos.

Em cenários táticos navais os acompanhamentos possuem distintas classificações. Uma delas diz respeito a uma classificação segmentada quanto à hostilidade do acompanhamento. Os acompanhamentos podem ser classificados dentro dessa segmentação de hostilidade como amigos, inimigos ou neutros. São exemplos de acompanhamentos neutros navios mercantes de nacionalidades não envolvidas em uma situação de conflito, navios-hospital, navios de pesquisa oceanográfica e outras categorias de embarcações.

Na maioria das vezes as embarcações possuem um sistema de emissão de identificação (IFF) que permite identificá-los. Entretanto, muitas vezes essa informação pode não estar disponível ou a embarcação ou aeronave pode não estar respondendo propositalmente. Nestes casos, a classificação quanto ao segmento de hostilidade é considerada como “desconhecida” e deve ser alvo de investigação que esclareça a natureza real da hostilidade do acompanhamento.

Assim, a proposta inicial neste nível é criar uma métrica da hostilidade para os acompanhamentos que se baseie em conhecimento de especialistas em operações navais militares e que considere fatores tais como o comportamento dos acompanhamentos e suas relações entre si e entre estes e entidades do cenário geográfico. O produto final dessas inferências é uma medida de *grau de hostilidade* dos acompanhamentos. Este conceito não existe atualmente na Marinha do Brasil, de forma que a proposta é criar uma escala de 0 a 100, em que quanto maior o valor maior é o grau de hostilidade avaliado segundo os critérios elaborados pelos especialistas.

Diferentemente das regras anteriormente criadas para a fusão de acompanhamentos de distintos sensores, as regras que determinam o *grau de hostilidade* dos acompanhamentos são de natureza totalmente Fuzzy. Assim, foi criado um conjunto de regras Fuzzy cuja variável de saída é o *grau de hostilidade* e algumas regras que modelam relações dos acompanhamentos com um elemento do cenário tático ou apenas com características identificadas dos acompanhamentos. A idéia é que todos os acompanhamentos considerados desconhecidos possuam alguma avaliação de grau de hostilidade. Se as características do acompanhamento não puderem ser identificadas como características de um comportamento potencialmente perigoso ou potencialmente tranqüilizador, então a classificação deve resultar em um valor característico de um grau médio de hostilidade.

O elemento do cenário geográfico utilizado para elaborar um exemplo com foi a rota de navegação. Essa escolha foi realizada porque a prática tem mostrado que a maioria do tráfego classificado como desconhecido é constituído por embarcações comerciais, chamados de navios cargueiros, que muitas vezes não respondem ao IFF por este não se encontrar disponível. Esses navios quase sempre percorrem rotas bem conhecidas de navegação comercial. A rota comercial naval não é um corredor com largura determinada, como uma via aérea, mas é um objeto que pode ser desenhado sobre uma carta náutica como uma seqüência de retas ligando diferentes regiões do planeta. A partir dessa linha central, há uma liberdade do cargueiro em seguir a certa distância por bombordo ou boreste, dentro do que poderia ser identificado como um corredor de navegação provável de embarcações comerciais.

Além disso, navios cargueiros possuem um comportamento conhecido em termos de velocidade e são embarcações de grande porte. Assim sendo, pode ser criada uma regra que identifique comportamentos compatíveis com os típicos de um navio cargueiro, de forma a classificá-los como de baixo *grau de hostilidade*.

Por outro lado, acompanhamentos desconhecidos que estejam em grande velocidade e em uma rota que seja convergente com a posição do navio que possui o sistema de C^4I, devem ser classificados como de alta hostilidade. assim, pode ser criada uma regra para identificar esse tipo de comportamento também.

O sistema pode crescer de forma a acrescentar outras variáveis de entrada necessárias para modelar outros comportamentos que possam ser identificados como característicos de um certo tipo de hostilidade. Esses comportamentos podem ser modelados por futuras regras especialistas a serem adicionadas ao sistema e que expressem esses conhecimentos.

5.3.5.1 A MODELAGEM FUZZY

Especificamente para calcular o *grau de hostilidade* em função do comportamento do acompanhamento em relação às rotas comerciais, foram modelados os conjuntos fuzzy que aparecem na figura 22.

Foram modeladas as seguintes variáveis: o ângulo entre o rumo do acompanhamento e a reta central da rota de navegação (chamado de eixo da rota), a proximidade da posição do acompanhamento com a reta central da rota e a velocidade do acompanhamento.

Os suportes dos conjuntos fuzzy da figura 22 foram definidos por especialistas em operações navais.

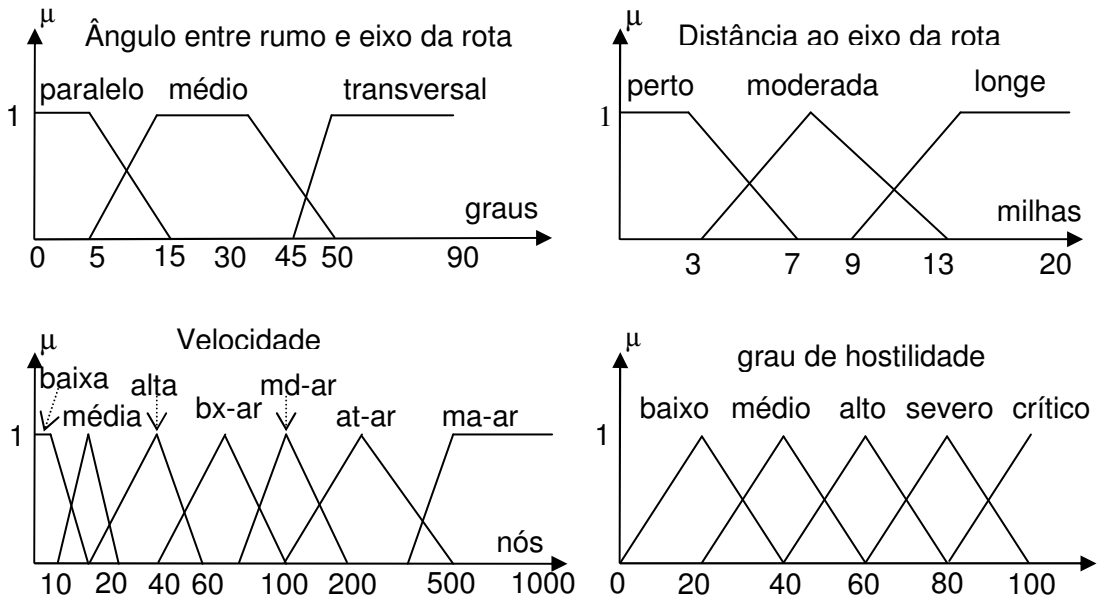


Figura 22 – Conjuntos fuzzy utilizados para cálculo do grau de hostilidade

Assim, uma regra fuzzy que reflete o grau de hostilidade baixo representado por navios grandes que naveguem em velocidade média e em rotas compatíveis com uma rota comercial, pode ser assim elaborada:

SE classificação-hostil É desconhecida E ângulo entre rumo e eixo da rota É paralelo E distância ao eixo da rota É perto E velocidade É média E tamanho É grande ENTÃO grau de hostilidade É baixo

Outras regras podem ser criadas para considerar outros aspectos do cenário e também concluir algo acerca do risco. Como exemplo foi criada uma regra que associa um acompanhamento aéreo que esteja em alta velocidade e em rota de aproximação com o navio onde está instalado o sistema, como um acompanhamento de grau de hostilidade severo. A regra é a seguinte:

SE classificação-hostil É desconhecido E ângulo entre rumo do alvo e direção do PPN É paralelo E distância ao PPN É perto E (velocidade É ma-ar OU velocidade É at-ar) E tamanho É pequeno ENTÃO grau de hostilidade É severo

Acompanhamentos que eventualmente não apresentam nenhum comportamento conhecido para determinar um grau de hostilidade podem ser modelados com regras *default* que calculem o grau de hostilidade como médio. Assim, foi criada uma regra desse tipo com a seguinte sintaxe:

SE classificação-hostil É desconhecido E velocidade É média E distância é longe ENTÃO grau de hostilidade É médio

Todas as regras ativadas em um determinado instante para o acompanhamento serão consideradas e será calculado um risco final para cada acompanhamento que reflete a combinação dos conhecimentos especialistas modelados.

5.3.5.2 RESULTADOS DOS TESTES DE FUSÃO NÍVEL 2

Foi criado um cenário de teste com três acompanhamentos de tipos, tamanhos e comportamentos distintos para testar o modelo de cálculo do *grau de hostilidade* descrito na seção anterior. Todos possuem uma classificação de hostilidade desconhecida.

O primeiro é um acompanhamento de tamanho grande, navegando a 20 nós em um rumo e uma posição próximos aos de uma rota de navegação comercial (tipicamente um cargueiro).

O segundo é um acompanhamento de tamanho pequeno, velocidade alta e que voa em rumo de interceptação em relação ao navio que contém o sistema (tipicamente um avião de ataque).

O terceiro é um acompanhamento de tamanho pequeno navegando em um local aleatório no cenário tático e uma velocidade média (poderia ser um pesqueiro).

A figura 23 ilustra o cenário em que os acompanhamentos foram criados, a posição da rota de navegação e a posição do navio que contém o sistema dentro do cenário.

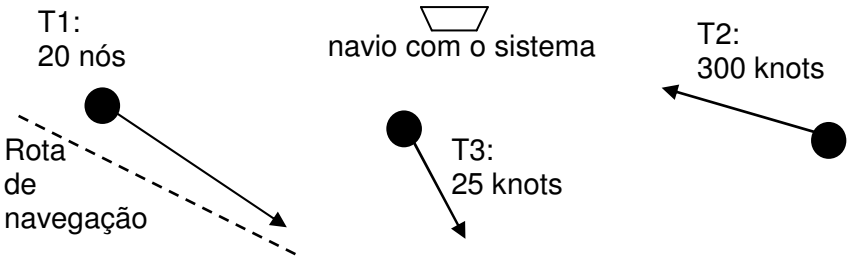


Figura 23 – Cenário com três alvos para cálculo do grau de hostilidade

Foram usados ruídos Gaussianos distintos para a posição do *dot* radar passado a cada um dos Extratores Radar. O desvio padrão do ruído Gaussiano gerado foi de $\sigma = 5$ metros para um radar e $\sigma = 20$ metros para o outro radar.

O *grau de hostilidade* foi automaticamente calculado pelas regras cada vez que ocorria a atualização de algum parâmetro de um acompanhamento gerado, durante o período de 5 minutos em que transcorreu o teste. A média e o desvio padrão dos valores de *grau de hostilidade* que foram apurados nesse período para cada acompanhamento estão descritos na tabela 18

	Desvio padrão do ruído de posição	*grau de hostilidade*	
		Média	Desvio padrão
Acompanhamento 1	**σ = 5**	19,5	1,3
	σ = 20	20,1	2,1
Acompanhamento 2	**σ = 5**	80	0
	σ = 20	80	0
Acompanhamento 3	**σ = 5**	40	0
	σ = 5	40	0

Tabela 18 – Resultados do teste para cálculo do grau de hostilidade

Os resultados obtidos mostram que o grau de hostilidade foi continuamente calculado para cada acompanhamento. Os acompanhamentos 2 e 3 não sofreram variação no cálculo do grau de hostilidade embora os atributos de posição e velocidade tenham sofrido variação em função da adição de ruído. Isto reflete o fato de que para esses acompanhamentos apenas uma regra foi acionada e a defuzzificação pela média dos máximos sempre considerou apenas um conjunto de saída e o valor final foi então o valor de máximo desse conjunto. Já para o caso do acompanhamento 1, duas regras foram acionadas (a regra que modela o comportamento dos cargueiros e a regra default). Assim, como o grau de pertinência da premissa variou em função da variação da cinemática dos acompanhamentos, então também a média resultante da defuzzificação variou o grau de hostilidade final.

5.3.6 APLICANDO O MODELO PARA A FUSÃO DE NÍVEL 3

Para criar um exemplo que possa ser classificado dentro do modelo JDL como fusão de nível 3, aplicamos o RT-MLR para realizar uma previsão de situação de engajamento futuro com uma ameaça. Engajamento é a situação que ocorre quando há um ataque direto entre duas unidades. O engajamento só pode ocorrer efetivamente quando as unidades envolvidas estão dentro do alcance das armas. Um engajamento ativo pode ocorrer quando se ataca uma unidade e um engajamento passivo ocorre quando se é atacado por uma unidade.

Para acompanhamentos suspeitos é útil realizar previsões, fornecer relatórios e emitir alarmes caso algum comportamento mais indicativo de ameaça iminente seja detectado. Acompanhamentos inimigos que mudem de rumo mais de uma vez em situações de combate e apresentem ou tenham apresentado no passado recente emissões eletromagnéticas compatíveis com radares militares podem ser encaradas como potenciais ameaças que devem ser cuidadosamente monitoradas. O que se pretende é estabelecer uma previsão de tempo de quando poderá ocorrer um engajamento futuro com uma determinada unidade suspeita. Entretanto, deseja-se que o sistema identifique automaticamente um determinado comportamento de risco de uma unidade inimiga.

Para detectar o comportamento suspeito serão usados operadores de lógica temporal. Uma regra que considere a quantidade de guinadas que o acompanhamento tenha realizado no passado e o fato de terem sido detectadas em algum momento no passado emissões de radares militares, pode ter o seguinte formato (em uma sintaxe mais simples que a efetivamente utilizada, apenas para efeito de facilitar a compreensão):

```
SE
    (ThereWasEqual(acompanhamento.emitindoRadarMil,true,900,0) E
     (Count(acompanhamento.alterandoRumo,true,900,0)>=2) E
     acompanhamento.velocidade > 5)
ENTÃO
     acompanhamento.setAlcance(
                        calculaAlcance(acompanhamento.getClasse));
     exibe(TempoParaCirculoAlcance(me,acompanhamento));
```

Onde:

- O operador temporal `ThereWasEqual` retorna *true* se o atributo booleano `milRadarEmissionDetected` houver recebido `true` nos 15 últimos minutos (900 segundos), mesmo que este atributo tenha outro valor no momento.
- O operador temporal `Count` retorna a quantidade de vezes em que foi atribuído o valor `true` para o atributo booleano `changingCourse` .
- A parte do operando que testa o valor da velocidade foi incluída apenas para forçar a execução periódica da regra, uma vez que a velocidade é estimada a cada volta da antena pelo Extrator Radar

As operações de lógica temporal foram combinadas com meta-informações, que são informações fornecidas manualmente pelo operador do sistema ou lidas de um banco de dados a partir da informação específica sobre a classe de navio ou de aeronave em que foi classificado o acompanhamento (por exemplo, fornecida a classe do navio, uma consulta ao banco de dados pode informar as armas que essa classe de navio possui e o alcance dessas armas). Assim, a informação sobre a *classe* do acompanhamento é considerada como meta-informação na medida em que seja introduzida pelo operador a partir de informativos de alto nível recebidos do comando da Força ou de outra fonte amiga ou ainda por deduções internas à unidade. Entretanto, a classificação também pode ser gerada dentro do próprio sistema, caso seja implementado algum modelo de classificação (que naturalmente pode ser feito usando o RT-MLR). Como este modelo de classificação não está proposto, vamos considerar a classe do acompanhamento como meta-informação. A partir disso se poderia então determinar o círculo de alcance de fogo da unidade, o que envolveria uma consulta a um banco de dados, e finalmente realizar um cálculo de tempo necessário para atingir esse círculo.

5.3.6.1 RESULTADOS DOS TESTES DE FUSÃO NÍVEL 3

Para este teste foi criado um acompanhamento com velocidade de 30 nós navegando a uma distância de 50 milhas do navio com o sistema (PPN). Esse acompanhamento foi então guinado duas vezes, a primeira vez em 20 graus e a segunda em 30 graus, ambas na direção do PPN. O evento correspondente à detecção de emissão de radar militar foi artificialmente introduzido pela interface do Visualizador, já que o simulador não inclui um módulo de simulação do MAGE, que é o equipamento que detecta e classifica esse tipo de emissão. Após a ocorrência dos três eventos, a regra especificada começou a ser disparada e o cálculo do tempo para entrada no círculo de alcance começou a ser periodicamente calculado, mostrando que a regra que estava anteriormente sendo investigada para cada alteração de velocidade, mas não estava sendo disparada, passou a ser disparada após a ocorrência dos eventos citados.

6 CONCLUSÕES

6.1 CONCLUSÕES

A principal contribuição do trabalho foi apresentar uma forma integrada de fundir informações contextuais com um modelo que permite o raciocínio baseado em múltiplas lógicas e que suporta processamento concorrente típico de aplicações de tempo real. Foi apresentada uma forma integrada de tratar a fusão de dados em aplicações de monitoração de um panorama tático naval em que os diversos níveis de fusão foram modelados conceitualmente como regras de associação entre os objetos do domínio que suportam conhecimento expresso com múltiplas lógicas. Para dar suporte à integração dos paradigmas declarativo e procedimental (orientado a objetos) foi implementado um framework que permite a construção de regras na própria linguagem do domínio da aplicação (Java) e possui um mecanismo de ativação automática das regras (esquema chamado de "*on line monitoring*") a partir de alterações significativas nos valores dos atributos considerados nas premissas das regras (o nível de significância da alteração é configurável pelo usuário). O framework suporta regras que expressam conhecimento em lógica de primeira ordem, lógica fuzzy e lógica temporal.

Para validação do modelo a fusão de dados foi implantada em um sistema simulado, construído com o propósito de agregar alguns componentes de software já existentes e que realizam fusão de dados de baixo nível, de forma a satisfazer requisitos dos sistemas táticos legados. Esses requisitos demandam que a fusão de alvos acompanhados seja feita após o tratamento de baixo nível, de forma que foram aproveitados os módulos de software já existentes que fazem o tratamento dos dados no nível de cada sensor (filtragem de ruídos com filtro CFAR e filtragem de estimativas da cinemática do alvo com filtro de Kalman). Com os dados provenientes desses equipamentos, foi então proposta uma primeira fusão que é a de acompanhamentos distintos do mesmo alvo. Essa fusão foi realizada com dois conjuntos de regras. Um primeiro conjunto de regras (totalmente fuzzy) infere o tamanho dos alvos em função do número de retornos de disparos do radar e da distância do alvo ao sensor. Esta informação e as demais informações sobre a cinemática dos acompanhamentos são então usadas como entradas de um segundo processo que envolve a fuzzificação das características disponíveis dos acompanhamentos e o teste da hipótese de fusão através de uma regra de lógica de primeira ordem (como outros modelos já propuseram anteriormente), mas que inova ao estabelecer um limiar de aceitação da hipótese de fusão que está associado à matriz de covariâncias dos erros de

estimação, calculada na etapa anterior pelo filtro de Kalman. Os testes mostraram que o modelo apresentou boa resposta na fusão de acompanhamentos de radares distintos, quer com dados simulados com diferentes níveis de ruído de processo e de medida introduzidos, quer com os dados reais de acompanhamentos observados por sensores existentes em navios da Marinha do Brasil a partir de corridas de testes registradas anteriormente.

Foi também elaborado um caso de teste do que seria considerado um exemplo de nível 2 de fusão pelo modelo conceitual de fusão de dados (JDL). Entretanto, como foi dito anteriormente, a divisão conceitual em níveis não têm nenhuma relevância na ferramenta utilizada e este caso de teste foi implementado com o mesmo princípio de regras de produção usado no caso anterior. As regras e as inferências produzidas neste caso são inteiramente fuzzy e procuram inferir a hostilidade dos acompanhamentos desconhecidos a partir do seu comportamento e de sua relação com outros entes do contexto geográfico como as rotas de navegação. A escala de classificação de hostilidade é uma proposta, no sentido em que não existe nenhuma norma que regulamente um grau de hostilidade, mas os dados dos testes mostraram que é possível capturar o conceito de comportamento hostil com conhecimentos especializados e em uma linguagem facilmente compreensível por um especialista e traduzi-lo em uma medida escalar que se mostrou coerente. Naturalmente que muitos mais conhecimentos e situações deveriam ser expressos em outras regras, estabelecendo relações com outros entes do contexto, mas o método é totalmente escalável e o acréscimo de conhecimento trará o refinamento da informação final fornecida sobre a hostilidade dos acompanhamentos.

Um terceiro modelo de fusão foi testado no ambiente simulado, envolvendo o cálculo de tempo para que um navio amigo entre no círculo de fogo de um acompanhamento inimigo. A resposta produzida é apenas um cálculo geométrico, mas a decisão de iniciar automaticamente o cálculo para determinados acompanhamentos leva em consideração aspectos do comportamento passado desses acompanhamentos. Assim, o modelo demanda operações lógicas com métricas de eventos ocorridos no passado e torna-se um caso de uso adequado para testar operadores temporais introduzidos como parte do conjunto de lógicas suportadas pelo modelo. Os testes mostraram que as regras conseguiram capturar o comportamento modelado e calcular o tempo para o evento futuro previsto.

6.2 TRABALHOS FUTUROS

6.2.1 INTRODUZIR O CÁLCULO DE RISCO

Apesar de não ter sido implementado, é possível determinar um grau de *risco* de cada acompanhamento a partir do cálculo do *grau de hostilidade*. Risco é um assunto muito discutido e em geral possui diversas abordagens. Muito do trabalho e da literatura sobre risco envolve o gerenciamento de risco [63] com objetivo de apoio à decisão. As áreas de aplicação vão dos sistemas de manutenção ao planejamento de atividades e investimentos.

Quanto ao risco no meio militar brasileiro existem alguns trabalhos que estão relacionados ao risco de atividades [64], segurança orgânica [65] e ações operativas a serem executadas [66]. Especificamente quanto ao risco que representa um inimigo em um ambiente tático não encontramos normas aplicáveis ou trabalhos relacionados.

No dicionário Michaelis risco é definido como “Possibilidade de perigo, incerto, mas previsível, que ameaça de dano uma pessoa ou uma coisa”.

Em [67] risco é definido como “probabilidade de que agentes, que são ameaças, explorem vulnerabilidades, expondo ativos a perdas de confidencialidade, integridade e disponibilidade, causando impactos aos negócios”.

Apesar desta definição estar relacionada ao ambiente de negócios, a idéia de relacionar ameaças e vulnerabilidades no cálculo do risco parece aplicável à situação em tela. O *grau de hostilidade* anteriormente calculado é uma métrica de quanto o acompanhamento é hostil baseado em seu comportamento e em características que possam relacioná-lo com outros elementos do contexto. Entretanto, o fato do acompanhamento ser hostil em maior ou menor grau deve ser combinado com a potencialidade de ameaça que ele representa para deduzirmos o risco efetivo para a integridade de uma unidade amiga. A potencialidade da ameaça está relacionada com as armas que ele possa dispor e também com as defesas com que se possa contar. Também seria necessário considerar a situação operativa corrente, que é emanada pelo comando da força e é classificada como branca, amarela ou vermelha. Essa classificação representa o grau de prontidão e de beligerância que se pode esperar em um cenário tático. Assim sendo, em tese, poder-se-ia criar um conjunto de regras fuzzy que levasse em consideração esses cinco aspectos, ou seja, o *grau de hostilidade* demonstrado pelo acompanhamento (e já calculado pelo sistema), a *potencialidade* das armas de defesa e de ataque do *inimigo* e do *objeto a defender* e a

classificação da *situação operativa* atual para produzir uma variável de saída que representasse um cálculo de risco.

6.2.2 MELHORAR A INTEGRAÇÃO COM O CONTEXTO

Uma das características mais importantes e uma das principais motivações deste trabalho é a possibilidade de relacionar informações dos sensores com informações do contexto geográfico, ambiental e tático. No capítulo onde foi relatada a validação do modelo, isso foi exemplificado no cálculo do grau de hostilidade, quando se relacionou as características dos acompanhamentos oriundos do Extrator Radar com as rotas de navegação. Entretanto, o exemplo é pobre em relação ao potencial de informações disponíveis sobre o contexto geográfico. Para explorar esse potencial de forma adequada, é necessário trabalhar no sentido de integrar o aplicativo tático com um sistema de informações geográficas (GIS). Escolher um GIS de código aberto onde seja possível acessar os objetos, extrair características e relacioná-las com entidades nativas do sistema tático, certamente agregará valor à tarefa de interpretação do contexto.

6.2.3 EXPANDIR A QUANTIDADE DE PARADIGMAS

Uma das contribuições do trabalho é possibilitar o uso de diferentes formas de raciocínio de uma forma integrada através de regras que contemplam lógicas distintas. Um progresso relevante nesse sentido seria incorporar a possibilidade de executar inferência Bayesiana no RT-MLR. Existem alguns pesquisadores [68, 69] que têm linhas de pesquisa nesse sentido, embora trabalhem prioritariamente com linguagens tipo OWL.

6.2.4 RACIOCÍNIO AUTOMÁTICO SOBRE O DOMÍNIO

A inferência automática a partir da alteração dos valores considerados nas regras é um dos aspectos principais deste trabalho. Entretanto, a inferência é realizada a partir apenas das regras elaboradas por especialistas. A adoção de uma base estrutural que se apoiasse em uma forma mais sistemática de descrever a hierarquia e os relacionamentos, incluindo restrições aos relacionamentos como cardinalidade e transitividade, certamente agregaria capacidade de realizar inferências a partir das relações hierárquicas, principalmente no que se refere à capacidade de classificação. Provavelmente a teoria mais adequada para dar suporte a essa abordagem seja a Description Logic (DL), que é suportada pela principal linguagem para descrição de ontologias, que é a OWL. Entretanto, OWL é uma linguagem

planejada para domínios abertos e para adequar a DL ao desenvolvimento de aplicações de tempo real deveriam ser pesquisados ou desenvolvidos módulos que permitissem um bom acoplamento entre DL e aplicações de tempo real. Alguns trabalhos nesse sentido [70, 71] têm sido recentemente publicados.

REFERÊNCIAS

[1] Ye, X., Veeramachaneni, K., Yan,Y., et al. “Unsupervised Learning and Fusion for Failure Detection in Wind Turbines”, *12th International Conference on Information Fusion*, pp. 1497-1503, Seattle, USA, July 2009.

[2] Tavard, F., Simon, A., Leclercq C., et al. “Data Fusion of Left Ventricle Electro-Anatomic Mapping and Multislice Computerized Tomography for Cardiac Resynchronisation Therapy Optimization”. *Proceedings of IEEE Computers in Cardiology*; pp. 613–6; 2009

[3] Q. Zhang, W. Tang, L. Lai, W. Sun, e K. Wong, “Medical diagnostic image data fusion based on wavelet transformation and self-organising features mapping neural networks”, *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, Shanghai, China: , pp. 2708-2712, *2004.*

[4] Srinivas, N. , Veeramachaneni, K., Osadciw, L.A., “Fusing Correlated Data from Multiple Classifiers for Improved Biometric Verification,” *12th International Conference on Information Fusion*, pp. 1504-1511, Seattle, USA, July 2009.

[5] Païs, G., Deloule, F., Beauchêne, D., “Animated Movie Activity Characterization by Image and Text Information Fusion” *12th International Conference on Information Fusion*, pp. 1068-1075, Seattle, USA, July 2009.

[6] Kouemou, G., Neumann, C., Opitz,F., **“***Fusion Technologies for Radar Target Classification Using Dempster-Shafer Rules in Littoral Environment*”, pp. 217-223. *12th International Conference on Information Fusion*, pp. 1068-1075, Seattle, USA, July 2009

[7] Mohammad, E., Abdalla, O., Garfield, M. et al. “Target Tracking in Multi-Static Active Sonar Systems Using Dynamic Programming and Hough Transform*”*, pp. 62-69. *12th International Conference on Information Fusion*, pp. 1068-1075, Seattle, USA, July 2009

[8] Smith, D., Singh, S., “Approaches to Multisensor Data Fusion in Target Tracking: A Survey”, *IEEE Transactions On Knowledge And Data Engineering*, Vol. 18, NO. 12, December 2006.

[9] Hall, D. L., S. A. McMullen, *Mathematical Techniques in Multisensor Data Fusion*, Artech House Publitions, 2004.

[10] Bar-Shalom, Y., Blair, W. D., *Multitarget-Multisensor Tracking: Applications and Advances*". 2000

[11] Kessler, O., *Functional Description of the Data Fusion Process*, Warmininster, PA: Office of Naval Technology, Naval Air Development Center, 1992

[12] Steinberg, A., C. Bowman, and E.E.White Jr., *Revisions to the JDL Data Fusion Model*, 3rd NATO/IRIS conferation, Quebec City, Canada, 1998

[13] Messali, Z. and F. Soltani, "Performance of Distributed CFAR Processors In Pearson Distributed Clutter" *IEEE Transactions on Aerospace and Electronic Systems*, v. 36, pp 1377 - 1386, Oct 2000

[14] Hall, M.J., S.A. Hall and T. Tate, "Removing the HCI Bottleeneck: How the Human Computer Interface (HCI) Affects the Performance of Data Fusion Systems", *In Proc. of 2000 MSS National Symposium on Sensor and Data Fusion*, San Diego, CA, pp 89-104, June 2000.

[15] Rosati, R., "On combining description logic ontologies and nonrecursive datalog rules" *In Proc. of the 2nd Int. Conf. on Web Reasoning and Rule Systems (RR 2008)*, 2008

[16] J.M. Juarez, M. Campos, J. Palma, e R. Marin, "Computing context-dependent temporal diagnosis in complex domains," *Expert Syst. Appl.*, vol. 35, 2008, pp. 991-1010. 2008.

[17] Corcho, Ó., Gómez-Pérez, A., "A Roadmap to Ontology Specification Languages", *In Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, October 02 – 06. 2000

[18] Bhansali, S., Grosof, B., "Extending the SweetDeal Approach for E-Procurement using SweetRules and RuleML", *In Proceedings of International Conference on Rules and Rule Markup Languages* (RuleML-2005). 2005

[19] Motta, E., *Reusable Components for Knowledge Modelling*. IOS Press. Amsterdam. 1999.

[20] MacGregor, R., *Inside the LOOM classifier*. In SIGART bulletin. #2(3):70-76. June, 1991.

[21] Ahmad, M., Colomb, R., “Managing ontologies: a comparative study of ontology servers”. *In Proceedings of Eighteenth Conference on Australasian Database* - Volume 63 - Ballarat, Victoria, Australia. 2007.

[22] Farquhar, A. et al. “The Ontolingua Server: A Tool for Collaborative Ontology Construction”. *In Proceedings of KAW96*. Banff, Canada, 1996.

[23] Chaudhri, V., et al., *Open Knowledge Base Connectivity 2.0.3*, Disponível em: <www.ai.sri.com/~okbc/okbc-2-0-3.pdf>, acessado em 13/12/2009.

[24] Kifer, M., Lausen, G., and Wu, J., “Logical Foundations of Object-Oriented and Frame-Based Languages”. Journal of the ACM. V. 42, 4 (Jul. 1995), pp 741-843 1995.

[25] Baclawski, K. et al. “Extending the UML for Ontology Development”, *International Journal of Software and Systems Modeling*, 1(2): 142-156. 2002.

[26] Vialatte, M., *Description et implémentation du moteur d’inférence Snark*. Tese de Doutorado, Université Paris VI. Paris: Laforia, 1985.

[27] Brownston, L., *Programming Expert Systems in OPS5*. Addison-Wesley, 1985

[28] Forgy, C. “Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem”, *Artificial Intelligence Journal*, V. 19, pp 17–37, 1982

[29] Giarratano, J., *CLIPS User’s Guide*, disponível em: < **clips**rules.sourceforge.net/documentation/v630/ug.pdf > acessado em 08/12/2009.

[30] Lopez F., *Nasa Clips Rule-Based Language*, disponível em: <http://www.siliconvalleyone.com/clips.htm>, acessado em 09/12/2009.

[31] Pachet, F., Représentation de *connaissances* par objets et règles: le système NéOpus. Tese de Doutorado, Université Paris VI. Paris: Laforia. 1992

[32] Friedman-Hill, E., *JESS, the Java Expert System Shell*, disponível em: <http://herzberg.ca.sandia.gov/jess>, consultado em 10/12/2009

[33] Mahmoud, Q., *Getting Started With the Java Rule Engine API (JSR 94): Toward Rule-Based Applications,* disponível em: < http://java.sun.com/developer/technicalArticles/J2SE/JavaRule.html> July 26, 2005. consultado em 10/12/2009

[34] *JRuleEngine User Manual* , disponível em <http://jruleengine.sourceforge.net/usermanual.html> (versão de Abril de 2008), acessado em 25/01/2010

[35] Filho, C. S., Ramalho, G. 2000. "JEOPS - The Java Embedded Object Production System", In *Proceedings of the international Joint Conference, 7th Ibero-American Conference on Ai: Advances in Artificial intelligence* (November 19 - 22, 2000).

[36] Dietrich, J., *The Mandarax Project*, disponível em: <http://mandarax.sourceforge.net>, acessado em 12/12/2009

[37] disponível em: <http://openrules.com>, acessado em 12/12/2009

[38] disponível em: <http://www.hammurapi.biz> acessado em 12/12/2009

[39] Rupp, N. A., "The Logic From The Bottom Line: An Introduction to The Drools Project" , disponível em www.theserverside.com/articles/article.tss?l=Drools, 2004, acessado em 14/12/2009

[40] Grosof, B. N., *Courteous logic programs: Prioritized conflict handling for rules*, Technical report, IBM T.J. Watson Research Center, IBM Research Report RC 20836. December 1997.

[41] Dörflinger, M., *Interpreting Courteous Logic Programs*, Diploma Thesis, Department of Informatics, University of Zurich. 2005

[42] Zadeh, L.A., "Fuzzy Sets", *Information and Control*, vol 8, pp 338-353, 1965

[43] Mamdani, E.H., "Applications of fuzzy algorithms for simple dynamic plant*", Proc. IEEE 121 (12)* 1585-1588, 1974

[44] Pani AK, Bhattacharjee GP. "Temporal representation and reasoning in artificial intelligence: a review", *Math Computing Model*;34(1-2):55-80, 2001

[45] Palma, J., Juarez, J. M., Campos, M., Marin, R., "Fuzzy theory approach for temporal model-based diagnosis: An application to medical domains", *Arificial Intelligence in Medicine*, v.38 n.2, p.197-218, October, 2006

[46] Juarez, J. M., Campos, M., Palma, J., et al., "Computing context-dependent temporal diagnosis in complex domains", *Expert Systems Applications* 35, 3 , pp 991-1010, October. 2008.

[47] Allen, J. F., “An interval-based representation of temporal knowledge”. In *Proceedings of the 7th international Joint Conference on Artificial intelligence* - Volume 1 ,Vancouver, BC, Canada, August 24 - 28, 1981

[48] Raj, A., Prabhakar, T. V., and Hendryx, “S. Transformation of SBVR business design to UML models*”*. In *Proceedings of the 1st Conference on India Software Engineering Conference* (Hyderabad, India, February 19 - 22, 2008).

[49] Bennett, B., Galton, A. P., “A unifying semantics for time and events”, *Artificial Intelligence*, Volume 153, Issues 1-2, Logical Formalizations and Commonsense Reasoning, Pages 13-48, March 2004

[50] Waltz, E. and Linnas, J., *Multi sensor Data Fusion*, Artech House, Norwood, MA. (1990)

[51] Oliveira, J. R. P., *Acompanhamento de Alvos Radar Utilizando Filtragem de Kalman e Vetor de Estados com Dimensão Variável*, Dissertação de Mestrado, PEE - COPPE – UFRJ, 2005.

[52] Grasso, R., Giannecchini, S. “Geo-spatial Tactical Decision Aid systems: fuzzy logic for supporting decision making”, *IEEE 9th Conference on Information Fusion*, pp 1-8, Florence, Italy, July 2006

[53] Shi, Y., Wang,Y., Shan,X., “A Novel Fuzzy Pattern Recognition Data Association Method for Biased Sensor Data”, *IEEE 9th Conference on Information Fusion*, Florence, Italy ,July 2006

[54] Bomberger, N. A., Rhodes, B. J., Seibert, M., Waxman, A. M., “Associative Learning of Vessel Motion Patterns for Maritime Situation Awareness”, *IEEE 9th Conference on Information Fusion*, July 2006

[55] Rhodes, B.J., Bomberger, N.A., Seibert, M.C., Waxman, A.M., “Maritime situation monitoring and awareness using learning mechanisms”, In *Proceedings of IEEE MILCOM 2005 Military Communications Conference,* Atlantic City, NJ, USA, 2005.

[56] García, J., Molina, J. M., Besada, J. A., Portillo, J. I., “A MultiTarget Tracking Video System based on Fuzzy and Neuro-Fuzzy Techniques”, *EURASIP Journal on Applied Signal Processing, special issue on Advances in Intelligent Vision Systems: Methods and Applications*. Nº 14, pp 2341-2358, 2005

[57] Tummala, M., Midwood, S. A., Glenn, I. N., “Multi Sensor Data Fusion Using Fuzzy-Association Techniques”, *Proceedings of the 40th Midwest Symposium on Circuits and Systems*, 1997. Volume 2, Issue , 3-6 Aug. 1997

[58] Midwood, Sean A., *A Computationally Efficient and Cost Effective Multisensor Data Fusion Algorithm for the United States Coast Guard’s Vessel Traffic Services System*, Master’s Thesis, Naval Postgraduate School, Monterey, CA, September 1997.

[59] Oliveira, C. A, Belderrain, M. C, “Aplicação de Lógica Fuzzy para fusão de dados de acompanhamentos de plataformas no cenário maritime”, *Revista Pesquisa Naval*, Nº 21, pp 49-57, 2008

[60] Aziz, A. M., “Fuzzy track-to-track association and fusion approach in distributed multisensor-multitarget multiple-attribute environment”, *Signal Processing* Nº 87, pp 1474-1492, 2007

[61] Skolnik, Merril I, *Introduction to Radar Systems*, McGrawnHill, second edition, 1980

[62] Vermaak, J., Godsill,S.J.,Perez,P., “Monte Carlo Filtering for multi-target tracking and data association”, *IEEE Trans. Aerospace Electronic System* 41 (1) – Jan 2005, 309-332

[63] Vaughan , E. J. *Risk management*. New York: John Willey & Sons, 1997

[64] *Instruções Reguladoras Sobre Análise de Riscos para ambientes de Tecnologia da Informação do Exército Brasileiro* (IRRISC) Departamento de Ciência e Tecnologia IR 13-10 Brasília. 2007

[65] Gomes, P.R.S, *A análise de risco nas OM operacionais do Exército Brasileiro e suas contribuições para a Segurança Orgânica*, Dissertação mestrado e Ciências Militares, Escola de Comando e Estado-Maior do Exército, 2008.

[66] *Gerenciamento de Riscos Aplicado às Atividades Militares*. 1. BRASIL. Exército. Comando de Operações Terrestres CI 32/2: ed. Brasília. 2005

[67] Sêmola, M., *Gestão de Segurança da Informação*. Rio de Janeiro, Campus, 2003.

[68] Costa, P.C.G. et al., “A Multi-Disciplinary Approach to High Level Fusion in Predictive Situational Awareness”, *12th International Conference on Information Fusion Seattle*, WA, USA, July 6-9, 2009

[69] Laskey, K.B. , “MEBN: A Language for First-Order Bayesian Knowledge Bases”, *Artificial Intelligence*, 172(2-3), pp. 140-178, 2008.

[70] Baader, F, et al, “A Novel Architecture for Situation Awareness Systems, Automated Reasoning with Analytic Tableaux and Related Methods”, *18th International Conference, TABLEAUX 2009*, Oslo, Norway, July 6-10, 2009.

[71] Franz Baader,F, Sertkaya, B. “Usability Issues in Description Logic Knowledge Base Completion”, *Proceedings of the 7th International Conference on Formal Concept Analysis*, (ICFCA 2009), volume 5548 of *Lecture Notes in Artificial Intelligence*, pp 1–21. Springer Verlag, 2009.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo