UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA MECÂNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

NAZARENO DE OLIVEIRA PACHECO

SISTEMA DE APOIO À SOLUÇÃO DE NÃO-CONFORMIDADES: UM ESTUDO DE CASO NA EXTRUSÃO DE ALUMÍNIO

Tese submetida ao Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal de Santa Catarina para a obtenção do Grau de Doutor em Engenharia Mecânica
Orientador: Prof. João Carlos Espíndola Ferreira, Ph.D.
Coorientador: Prof. Walter Luís Mikos, Dr.

Florianópolis

2014

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Pacheco, Nazareno de Oliveira
SISTEMA DE APOIO À SOLUÇÃO DE NÃO-CONFORMIDADES: UM ESTUDO DE CASO NA EXTRUSÃO DE ALUMÍNIO / Nazareno de Oliveira Pacheco ; orientador, João Carlos Espíndola Ferreira ; coorientador, Walter Luís Mikos. - Florianópolis, SC, 2014.
168 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia Mecânica.

Inclui referências

1. Engenharia Mecânica. 2. Raciocínio Baseado em Casos. 3. Sistemas Multiagentes. 4. Ontologias. 5. Extrusão de alumínio. I. Ferreira, João Carlos Espíndola. II. Mikos, Walter Luís. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia Mecânica. IV. Título.

NAZARENO DE OLIVEIRA PACHECO

SISTEMA DE APOIO À SOLUÇÃO DE NÃO-CONFORMIDADES: UM ESTUDO DE CASO NA EXTRUSÃO DE ALUMÍNIO

Esta Tese foi julgada adequada para obtenção do Título de Doutor em Engenharia Mecânica e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Mecânica

Florianópolis, 29 de setembro de 2014.

_____

Prof. Armando Albertazzi Gonçalves Jr, Dr.
Coordenador do Curso

Banca Examinadora:

_____	_____
Prof. João Carlos Espíndola Ferreira, Ph.D. - Orientador Universidade Federal de Santa Catarina	Prof. Walter Luís Mikos, Dr. - Coorientador Universidade Tecnológica Federal do Paraná
_____	_____
Prof. Fernando Antônio Forcellini, Dr. Universidade Federal de Santa Catarina	Prof. Osiris Canciglieri Jr, PhD. Pontifícia Universidade Católica do Paraná
_____	_____
Prof. Gilmar Ferreira Batalha, Dr. Universidade de São Paulo	Prof. Carlos Augusto Silva de Oliveira, Dr. Universidade Federal de Santa Catarina

Prof. Armando Albertazzi Gonçalves Jr., Dr.
Coordenador do Curso

Banca Examinadora:

Prof. João Carlos Espíndola Ferreira, Ph.D. - Orientador
Universidade Federal de Santa Catarina

Prof. Walter Luís Mikos, Dr. – Coorientador
Universidade Tecnológica Federal do Paraná

Prof. Fernando Antonio Forcellini, Dr.
Universidade Federal de Santa Catarina

Prof. Gilmar Ferreira Batalha, Dr.
Universidade de São Paulo

Prof. Carlos Augusto Silva de Oliveira, Dr.
Universidade Federal de Santa Catarina

Prof. Osiris Canciglieri Junior, Ph.D.
Pontifícia Universidade Católica do Paraná

Prof. Gustavo Daniel Donatelli, Dr.
Universidade Federal de Santa Catarina

Prof. Fernando Humel Lafratta, Dr.
Universidade do Estado de Santa Catarina

Este trabalho está dedicado a Deus quem sempre tem sido fiel para comigo, a minha esposa Elisa pelo apoio e suporte, minhas filhas Sara e Beatriz pelos momentos de ausência,

ao meu avô Antônio Nazareno, meus pais Seloé e Nazarita pelo seu amor incondicional, aos meus irmãos Juliano e Gabriel e a minha sogra Ionice.

AGRADECIMENTOS

A meu orientador, o professor João Carlos Espíndola Ferreira, pela sua colaboração e paciência no desenvolvimento do presente trabalho.

A meu coorientador, o professor Walter Luís Mikos, pela colaboração no desenvolvimento do presente trabalho.

Ao Fábio Schuartz, pela ajuda e colaboração no desenvolvimento do presente trabalho.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro.

Aos meus colegas de laboratório Julio Ticona e Rafael Alberto López, com os quais trabalhei e compartilhei o laboratório.

A meu amigo Roberto Silvio Ubertino Rosso Jr.

Aos amigos da Igreja Adventista do Sétimo Dia, por terem me dado sua amizade e carinho.

RESUMO

Esta tese investiga o uso da abordagem de sistemas multiagentes (SMA) buscando compartilhar e recuperar conhecimentos decorrentes da solução de problemas prévios de não-conformidades e da aplicação do método de análise de modos de falha e efeitos em processos de manufatura (PFMEA) e raciocínio baseado em casos. Neste sentido, propõe-se um modelo de multiagentes em apoio à solução de problemas de não-conformidades em processos de fabricação, com o intuito de superar não somente as dificuldades relacionadas à natureza do conhecimento, mas também quanto à sua distribuição. A distribuição adotada no modelo considera tanto o aspecto geográfico das fontes quanto à fragmentação relacionada aos processos existentes na cadeia produtiva. Nesta ótica, são considerados agentes computacionais cujo comportamento inclui o uso de raciocínio baseado em casos e métodos de recuperação baseado em ontologias. Por fim, um protótipo computacional foi desenvolvido para permitir a verificação e a validação do modelo proposto, sendo que as bases de conhecimento manipuladas pelo sistema são instanciadas com conhecimentos no domínio do processo de extrusão de alumínio obtidos a partir da literatura e de pesquisas de campo em uma empresa que fabrica peças por extrusão de alumínio, com ênfase na liga 60xx.

Palavras-chave: Não-conformidades. Extrusão de Alumínio. Aquisição e Análise de dados. Conhecimento Distribuído. Sistemas Multiagentes. Raciocínio Baseado em Casos. Ontologias.

ABSTRACT

This thesis investigates the use of the multi-agent systems (MAS) approach seeking to share and retrieve knowledge from previous solutions of nonconformance problems and the application of the method of failure modes and effects analysis in manufacturing processes (PFMEA) and case-based reasoning (CBR). In this sense, it is proposed a MAS-based model to support the solution of nonconformance problems in manufacturing processes in order to overcome the difficulties related to both the nature of knowledge and on its distribution. The distribution adopted in the model considers both the geographical aspect of the sources and the fragmentation related to existing processes in the production chain. From this perspective, agents were developed whose behavior includes the use of case-based reasoning and retrieval methods based on ontologies. Finally, a software prototype was developed to allow the verification and validation of the proposed model, and the foundations of knowledge manipulated by the system are instantiated with knowledge in the field of the aluminum extrusion process obtained from the literature and from a company that manufactures parts via aluminum extrusion, with emphasis on the 60xx alloy.

Keywords: Nonconformances. Aluminum Extrusion. Data Acquisition and Analysis. Distributed Knowledge. Multiagent systems. Case-based Reasoning. Ontologies.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACC - *Agent Communication Channel*
ACL - *Agent Communication Language*
ADK - *Agent Development Kit*
AID - *Directory of agent identifiers*
AMS - *Agent Management System*
AUML - *Agent Unified Modeling Language*
CBR - *Case-Based Reasoning*
CDE - *Data Capture System*
CEP - Controle Estatístico do Processo
CEQ - Controle Estatístico da Qualidade
CO-ODE - *Collaborative Open Ontology Development Environment*
CSELT - *Centro de Studi e Laboratori Telecomunicazioni*
DCS - *Data Capture System*
DF - *Directory Facilitator*
DIG - *Description Logics Implementation Group*
FACT - *Fast Classification of Terminologies*
FIPA - *Foundation for Intelligent Physical Agents*
FMEA - Análise de Modos de Falhas e seus Efeitos
FMECA - *Failure Mode Effects and Criticality Analysis*
GAIA - Grupo de Aplicações de Inteligência Artificial da Universidade Complutense de Madri
GUI - *Graphical User Interface*
IA – Inteligência Artificial
IDEF0 - *Integrated DEFiniton Methods*
ISO - *International Organization for Standardization*
JADE - *Java Agent DEvelopment Framework*
JAS – *Java Agent Services*
JDBC - *Java Database Connectivity*
JVM - *Java Virtual Machine*
KBSI – *Knowledge Based Systems Inc.*
KM - *Knowledge Management*
LGPL - *Lesser General Public License*
LISP - *list-processing language*
MTP - *Message Transport Protocol*
MTS - *Message Transport System*
NASA - *National Aeronautics and Space Administration*
nRQL - *new Racer Query Language*
OWL - *Web Ontology Language*

OWL-DL - *Web Ontology Language – Description Logic*
PFMEA - Análise de Modos de Falhas e seus Efeitos em Processos de Manufatura
PSL - *Process Specification Language*
RacerPro - *Renamed ABox and Concept Expression Reasoner*
RBC - Raciocínio Baseado em Casos
RDF - *Resource Description Framework*
RMI - *Remote Method Invocation*
RQLD - *Query Language for RDF*
SADT - *Structured Analysis and Design Technique*
SMA – Sistemas Multiagentes
TILAB - *Telecom Itália LABoratori*
W3C - *World Wide Web Consortium*
XML - *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Devido à globalização e ao aumento da concorrência, cada vez mais a indústria de manufatura está em busca de produzir produtos de alta qualidade de maneira eficiente com baixo custo, qualidade essa que tem que satisfazer as expectativas e necessidades de clientes cada vez mais exigentes. Para atender tal exigência, a indústria de manufatura teve a necessidade de buscar cada vez maior flexibilidade para responder de maneira rápida ao mercado. Essa estratégia orientada para o cliente exige uma infraestrutura de gestão integrada de informações, integrando-se completamente as necessidades do negócio com o processo de manufatura (HENDRICKS et al., 2007).

Apesar dos avanços tecnológicos para tal integração, a indústria de manufatura continua a passar por uma significativa transformação na sua organização, com o aumento dos custos das matérias-primas, e as flutuações de preços de produtos. Para lidar com essas variações, mudanças estruturais têm sido realizadas nas empresas de manufatura para manter a fidelidade do cliente, por meio da busca por maior eficiência, eficácia e flexibilidade nos seus processos (FILIPOV e CHRISTOVA, 2008).

Contudo, essas mudanças no modo de organização e, por extensão, na forma de atuar no mercado, resultam em uma mudança na forma de competição, deixando de ser somente entre empresas individuais, passando para uma competição entre redes (CARRIE, 2000).

No tocante à melhoria da qualidade dos produtos, um aspecto fundamental diz respeito ao tratamento das não-conformidades ao longo do ciclo de vida do produto pois, mesmo consistentes aplicações de medidas orientadas para a garantia da qualidade não podem eliminar, completamente, a possibilidade de que erros ou não-conformidades venham a ocorrer durante os processos de produção. Tais problemas diminuem a funcionalidade, a qualidade e a confiabilidade dos produtos, afetando negativamente a competitividade das empresas devido ao aumento dos custos (FÖRSTER et al., 1996; PFEIFER, 1997).

Estudos feitos em empresas alemãs com sistemas de manufatura tradicionais dos setores metal-mecânico e químico mostraram que durante a produção, em média, 60% das não-conformidades ocorridas são recorrentes, isto é, já ocorreram do mesmo modo ou de modo semelhante no passado e consumiram, em média, 10% dos recursos de pessoal e máquinas (PFEIFER, 1997; KLAMMA, 2000).

Porém, o conhecimento originado durante o processo de investigação das causas das não-conformidades, que inclui as medidas

introduzidas para evitá-las bem como os resultados de sua eficácia, frequentemente não é retido ou, em geral, é de difícil recuperação, prejudicando a aprendizagem organizacional a partir do tratamento das não-conformidades. Este é seguramente um pré-requisito para o processo de melhoria contínua da qualidade dos produtos e processos, ao mesmo tempo em que contribui para reduzir os custos (PFEIFER et al., 1998; PFEIFER et al. 2000; LARI, 2003).

Neste sentido, a literatura revela que os sistemas multiagentes (SMA) constituídos por múltiplos elementos computacionais interativos denominados agentes, os quais são dotados de capacidade de comunicação e ação autônoma, e que podem atuar de modo cooperativo visando atingir seus objetivos de projeto (WOOLDRIDGE, 2002; SULTAN et al., 2014), representam uma abordagem aplicável e eficiente para o desenvolvimento de ambientes para a gestão do conhecimento distribuído.

Adicionalmente, esta tecnologia tem sido considerada uma importante e reconhecida abordagem de Inteligência Artificial (IA) distribuída para o desenvolvimento da próxima geração de sistemas de projeto e manufatura inteligente. Diversas pesquisas têm sido realizadas neste âmbito aplicando-se sistemas multiagentes, as quais abrangem as seguintes áreas: engenharia simultânea, projeto colaborativo, integração entre empresas de manufatura distribuída, gestão da cadeia de suprimentos, planejamento e controle da produção, programação da produção, movimentação de materiais e sistemas de manufatura holônica (SHEN et al., 2001 e 2005; PAOLUCCI e SACILE, 2005).

Por outro lado, na última década, o Raciocínio Baseado em Casos (RBC) (do termo em inglês *Case-Based Reasoning*) evoluiu de uma área de pesquisa específica da IA para um campo de interesse amplo no desenvolvimento de sistemas de gestão do conhecimento. O RBC pode ser entendido como um procedimento para solucionar problemas em que a ideia fundamental é que, em determinados domínios, os problemas a serem resolvidos tendem a ser recorrentes, repetindo-se com pequenas variações em relação à sua versão inicial e, deste modo, soluções prévias podem ser reaplicadas também com pequenas modificações (KOLODNER, 1993, AAMODT e PLAZA, 1994; WATSON, 2003; von WANGENHEIM e von WANGENHEIM, 2003).

Nesse contexto, a contribuição e o ineditismo esperados para este trabalho de tese consistem, fundamentalmente, em abordar o processo de melhoria da qualidade industrial, em especial, como apoiar as situações de tratamento das não-conformidades em ambientes de manufatura distribuída e ao longo do ciclo de vida do produto, por meio de um sistema

de multiagentes em conjunto com o raciocínio baseado em casos para constituir um ambiente de gestão do conhecimento organizacional relativo à qualidade.

1.1 PROBLEMA DE PESQUISA

A solução de problemas de não-conformidades em processos de manufatura permanece ainda um desafio do ponto de vista acadêmico, pois, basicamente, as soluções de tais problemas envolvem atividades intensas em conhecimento e baseadas fortemente em experiências as quais, em casos complexos, podem extrapolar o conhecimento e a experiência dos técnicos, tecnólogos e engenheiros de uma única empresa integrada.

Do ponto de vista metodológico, a identificação dos elementos chave que contribuem para a eficácia da solução de problemas de não-conformidades não é uma tarefa comum. A norma internacional NBR ISO 9004:2000 (Sistemas de gestão da qualidade ‑ Diretrizes para melhorias de desempenho) destaca dois aspectos diferentes para esta questão.

O primeiro refere-se às não-conformidades que já aconteceram concretamente e, portanto, necessitam de ações corretivas para evitar a sua recorrência. E o segundo refere-se às não-conformidades potenciais que, desta forma, necessitam de ações preventivas que evitem a sua ocorrência.

Para o segundo aspecto, a norma estabelece que uma atenção especial deve ser dispensada aos seguintes itens: à definição de não-conformidades potenciais e de suas causas, avaliação da necessidade de ações para evitar a ocorrência da não-conformidade, definição e implementação de ações necessárias, registros de resultados das ações executadas e análise crítica de ações preventivas executadas.

O mesmo estudo revelou que o conhecimento produzido durante o processo de investigação das causas das não-conformidades, sobre as medidas introduzidas para evitá-las, bem como os resultados de sua eficácia, frequentemente não eram armazenados de forma apropriada. Isto dificulta a recuperação e reuso deste conhecimento, prejudicando, assim, a aprendizagem organizacional a partir do tratamento das não-conformidades (PFEIFER et al., 1998; PFEIFER et al. 2000; KLAMMA, 2000).

Por sua vez, em relação à segunda perspectiva, é importante observar que estes elementos indicados pela norma podem ser

determinados mediante a aplicação do método de Análise de Modos de Falha e seus Efeitos - FMEA[1] (STAMATIS, 2003). Este é um importante método preventivo para a garantia da qualidade, no qual diversos especialistas no domínio são envolvidos em um processo de investigação sistemática de todas as causas e efeitos relacionados a todos os possíveis modos de falha de um sistema.

Este processo de investigação ocorre, ainda, nas fases iniciais de desenvolvimento do produto e permite, desta forma, planejar e priorizar as ações com o objetivo de melhorar o produto ou o processo, considerando os respectivos níveis de severidade e probabilidades de ocorrência e detecção (STAMATIS, 2003).

No entanto, a literatura revela que estes valiosos conhecimentos sobre os produtos e processos de produção empregados na manufatura representam um grande desafio para seu compartilhamento e reuso no contexto de sistemas inteligentes de recuperação de conhecimento (DITTMANN et al., 2004). Isto porque, em geral, o conhecimento decorrente da aplicação deste método de análise não é completamente organizado do ponto de vista semântico para ser manipulado pelos sistemas inteligentes. Em outras palavras, o significado do conhecimento produzido depende de interpretação dos especialistas envolvidos e podem diferir da interpretação de outros especialistas (DITTMANN et al., 2004).

Assim, tendo em vista o cenário exposto acima, este trabalho de tese propõe a seguinte questão norteadora do trabalho de pesquisa:

Como representar e compartilhar os conhecimentos produzidos durante os processos de solução de problemas de não-conformidades, bem como aqueles decorrentes da aplicação do método de Análise de Modos de Falha e seus Efeitos em Processos de Manufatura, de modo a apoiar a solução de novos problemas de não-conformidades?

Dentro deste aspecto, as características essenciais dos ambientes de manufatura caracterizados pelas novas estruturas organizacionais sugerem o uso de novas abordagens baseadas na disponibilidade de tecnologia de informação e comunicação (HENDERSON-SELLERS e GIORGINI, 2005). Dentre as preocupações que deve-se ter em tais ambientes incluem-se: onde sistemas heterogêneos devem interagir e onde as fronteiras organizacionais e geográficas devem ser expandidas; onde é necessário operar, de forma eficiente, e dentro circunstâncias com rápidas mudanças de requisitos e com dramático aumento da quantidade de informações disponíveis; e, ainda, operar com segurança suficiente

[1] Tradução do original em inglês *Failure Mode and Effects Analysis*.

para proteger dados pessoais e outros ativos de conhecimento dos inúmeros envolvidos.

Nesta nova conjuntura, em particular, a necessidade de algum grau de autonomia que permita aos sistemas responder dinamicamente às mudanças nas circunstâncias do ambiente enquanto busca atingir seus objetivos primordiais de projeto, é vista por muitos pesquisadores como um aspecto fundamental a ser considerado nos novos modelos.

Neste sentido, a abordagem orientada a agentes computacionais ou sistemas multiagentes vem se tornando, ao longo da última década, uma importante alternativa para responder de forma concreta a estes desafios, bem como contornar a complexidade dos sistemas (LUCK et al., 2004; PECHOUCEK; THOMPSON, 2006 e MARCKI et al. 2014). Na literatura, os sistemas multiagente destacam-se ainda como tecnologias adequadas ao desenvolvimento de ambientes para a gestão do conhecimento (SHEN et al., 2001; van ELST et al., 2004).

Por outro lado, com relação às formas de representação de conhecimento e métodos de recuperação que podem ser empregados pelos agentes de informações, destaca-se a técnica de raciocínio baseado em casos (RBC). Esta técnica evoluiu, na última década, de uma área de pesquisa isolada e específica da Inteligência Artificial (IA) para um campo de interesse amplo no desenvolvimento de ferramentas para de gestão do conhecimento.

Dentre os estudos da área de RBC está o trabalho de Kolodner (1993), que estabelece que um sistema de raciocínio baseado em casos resolve problemas mediante a reutilização do conhecimento e experiências recuperadas de uma situação-problema anterior porém similar (representada por casos), a partir de uma base de casos. Se necessário esta solução recuperada ou a estratégia de solução do problema pode ser adaptada por meio do conhecimento geral do domínio. Além disso, mediante a atualização da base de casos, a solução adaptada torna-se disponível para novos problemas em um processo cíclico e integrado de solução de problemas, que aprende continuamente a partir das experiências.

Portanto, dentro desta perspectiva, o raciocínio baseado em casos pode ser usado como uma abordagem científica válida para o problema de recuperação e reuso do conhecimento produzido durante o processo de investigação das causas das não-conformidades, sobre as medidas introduzidas para evitá-las, bem como os resultados de sua eficácia.

Por outro lado, também nos últimos anos, pesquisas sobre o uso de ontologias como forma de representação de conhecimentos têm sido essenciais em muitas aplicações, dentre as quais pode-se citar: sistemas

de gestão do conhecimento, integração inteligente de informações, acesso baseado em semântica para a Internet e sistemas multiagentes (ABDULLAH et al., 2006 e VALIENTE et al., 2012).

Em relação a este tema, em particular, Mendes et al. (2013) e Yan et al (2014) sugerem que o uso de ontologias e métodos de recuperação baseados nestas ontologias podem representar uma alternativa inovadora para representar e compartilhar o conhecimento decorrente da aplicação do método de Análise de Modos de Falha e seus Efeitos, em especial visando superar as barreiras semânticas associadas ao método como apresentado pela literatura.

Dentro deste contexto, a hipótese de trabalho considerada nesta pesquisa envolve a adoção da abordagem de agentes de informação como abstrações capazes de lidar com a complexidade relacionada às diferentes fontes de conhecimento. Neste sentido, estas fontes de conhecimento dizem respeito não somente aos diferentes processos de manufatura e organizações envolvidas na cadeia produtiva, mas também às diferentes perspectivas de investigação e de análise dos problemas de não-conformidades.

1.2 OBJETIVOS DO TRABALHO DE TESE

1.2.1 Objetivo Geral

Desenvolver um sistema para coleta, análise e apresentação de resultados referentes a processos de manufatura, e com base nestes dados o sistema, em um curto espaço de tempo, apresenta para o usuário uma orientação sobre as possíveis causas da não-conformidade introduzida pelo usuário, e possíveis soluções para o problema.

1.2.2 Objetivos Específicos

- Desenvolver o sistema de apoio à tomada de decisões, e avaliá-lo em nível de chão de fábrica;
- Avaliar o impacto gerado nos colaboradores a partir da utilização do sistema.

1.3 DELIMITAÇÃO DO TRABALHO

É importante destacar que, embora a etapa de modelagem conceitual da organização multiagentes prevista nesta tese possa ser conduzida em um alto nível de abstração e, portanto, livre de um domínio de aplicação em particular, as etapas de especificação, implementação e verificação e validação do modelo sugerido, respectivamente, exigem um nível mais baixo de abstração com a escolha de pelo menos um domínio de aplicação. Isto é, deve ser escolhido pelo menos um processo de manufatura específico, a partir do qual as bases de conhecimento possam ser conceitualizadas e instanciadas, bem como preenchidas com conhecimento válido neste domínio para permitir as necessárias etapas de verificação e validação.

Assim, nesta tese, foi escolhido o processo de extrusão de alumínio, onde foram considerados dois aspectos principais: a abrangência do processo em termos de sua cadeia produtiva e a natureza da atividade de solução de problemas de não-conformidades.

1.4 METODOLOGIA

De acordo com Cervo e Bervian (2002), uma pesquisa científica pode ser classificada a partir de quatro dimensões: quanto à natureza, quanto à forma de abordagem do problema, quanto aos objetivos científicos e quanto aos procedimentos técnicos.

Quanto à natureza, esta tese se enquadra como uma pesquisa aplicada, segundo a linha de Gil (2010), pois os conhecimentos gerados são aplicáveis a problemas de cunho eminentemente prático.

Quanto à forma de abordagem do problema, esta tese tem um viés qualitativo, ainda na linha de pensamento de Gil (2010), pois ela se concentra no processo de modelagem e em seu significado, e as discussões são baseadas na análise dos resultados da construção do modelo.

Em relação ao objetivo, a pesquisa empreendida é exploratória, de acordo com a definição de Gil (2010), pois a pesquisa busca explorar o uso da abstração de agentes computacionais em apoio à solução de problemas de não-conformidades a partir de diferentes fontes de conhecimento.

Quanto aos procedimentos técnicos, conforme Gil (2010), esta pesquisa caracteriza-se, inicialmente, como uma pesquisa bibliográfica,

mas envolve, adicionalmente, uma pesquisa de campo para corroborar e complementar as lacunas teóricas identificadas na literatura.

1.5 ESTRUTURA DA TESE

Com o intuito de alcançar os objetivos propostos neste trabalho de tese, e para ter uma sequência lógica, a mesma está dividida em oito capítulos.

O capítulo 1 introduz o tema desta pesquisa, são apresentados os objetivos e as justificativas do trabalho, os tipos de pesquisas, e como o presente trabalho se enquadra nessa classificação, e finalmente é apresentada a estrutura da tese.

O capítulo 2 apresenta a revisão da literatura relativa aos conceitos que fundamentam esta pesquisa.

O capítulo 3 descreve os recursos de software utilizados para a implementação do modelo sugerido.

No capítulo 4 é desenvolvimento do modelo conceitual e especificações do projeto.

O capítulo 5 trata acerca da implementação do modelo baseado em RBC, enquanto o capítulo 6 descreve a implementação da base de conhecimento.

O capítulo 7 trata do processo de validação e verificação do modelo sugerido.

Finalmente, o capítulo 8 apresenta as conclusões e as recomendações para trabalhos futuros.

2 REVISÃO DA LITERATURA

O propósito deste capítulo consiste em explorar os aspectos da gestão do conhecimento e das técnicas para a construção de uma memória corporativa em apoio aos processos de melhoria da qualidade industrial. Com esta finalidade, apresenta-se uma revisão do atual status, que é a análise dos principais conceitos relacionados à gestão da qualidade em ambientes de manufatura distribuída, sistemas multiagentes aplicados à gestão do conhecimento, ontologias como suporte ao compartilhamento do conhecimento, técnica de raciocínio baseado em casos (RBC), e o processo de extrusão de alumínio. Este capítulo é dividido em cinco partes.

A primeira parte apresenta, inicialmente, uma análise dos conceitos relativos à gestão do conhecimento organizacional a partir da perspectiva da gestão da qualidade em ambientes de manufatura distribuída. Em especial, são apresentadas considerações sobre a natureza do conhecimento referente à qualidade, bem como a noção de memória corporativa para a melhoria contínua da qualidade.

A segunda parte concentra-se nos aspectos fundamentais da tecnologia de agentes e sistemas multiagentes.

A terceira parte apresenta definições sobre ontologias, bem como o software utilizado para efetuar a modelagem do sistema.

Na quarta parte são apresentados os fundamentos da técnica de raciocínio baseado em casos (RBC), com ênfase na apresentação de um quadro conceitual e na descrição do ciclo de RBC.

Por fim, na quinta parte são apresentados aspectos da modelagem do conhecimento associados às ações corretivas decorrentes das atividades de tratamento de não-conformidades no processo de extrusão de alumínio, obtidos através da literatura e de relatórios de não-conformidades fornecidos por uma empresa que fabrica diversos produtos mediante o processo de extrusão de alumínio.

2.1 Gestão do conhecimento em ambiente de manufatura

Os conceitos e as experiências de gestão da qualidade ocupam, inegavelmente, um espaço importante na literatura da área de manufatura e de negócios. Em parte, devido ao reconhecimento da comunidade acadêmica e industrial de que a qualidade é um fator crítico para a competitividade. Este reconhecimento pode ser observado claramente nas contribuições e trabalhos da área, tais como em Taguchi (1980), Deming

(1982), Crosby (1984), Ishikawa (1985), Juran (1988), Feingenbaum (1991) e Pfeifer (1996).

Assim, do ponto de vista histórico, o conceito de qualidade evoluiu de maneira bastante elevada no último século, partindo da ideia de que a qualidade poderia ser alcançada essencialmente por meio de técnicas de inspeção orientadas ao produto. A qualidade passa pelo conceito de controle, onde abordagens sistemáticas buscavam não somente detectar, mas estabelecer um tratamento consistente para os problemas de qualidade, concentrando-se no desempenho dos processos de manufatura com o apoio de diferentes métodos estatísticos e padrões de qualidade (SLACK et al., 2009).

No âmbito desta linha de pensamento, a literatura revela mais recentemente as abordagens voltadas à garantia da qualidade, as quais ampliaram o escopo do controle da qualidade convencional, revelando a importância de outras funções da organização para a qualidade, além das próprias operações de produção. Esta abordagem adotava, de forma inovadora, técnicas de planejamento da qualidade mais sofisticadas, análises de custos da qualidade, métodos de solução de problemas sistemáticos e outras técnicas estatísticas como, por exemplo, projeto de experimentos (SLACK et al., 2009).

Já na década de 1980, por sua vez, foram propostos modelos de referência para a preparação de sistemas de gestão de qualidade de acordo com as normas e padrões internacionalmente aceitos, em contraposição aos sistemas *ad hoc*. Entre estes, pode-se citar os modelos apresentados nas normas da série ISO 9000 (NBR ISO, 2000; ISO, 2005).

Inclui-se aqui a abordagem de gestão da qualidade total[2], que traduz um modo de agir e pensar a produção de forma orientada ao cliente, envolvendo todas as funções da organização, além dos fornecedores e clientes. Neste sentido, esta abordagem adotava estratégias consistentes de qualidade e, em especial, envolvendo e motivando as pessoas na busca de melhorias contínuas (SLACK et al., 2009; FNQ[3], 2006).

[2]Do termo em inglês, TQM – *Total Quality Management*.

[3]Conceitos Fundamentais da Excelência em Gestão da Fundação Nacional da Qualidade.

2.1.1 Modelagem de conhecimentos decorrentes do tratamento de não-conformidades por meio de ações preventivas e corretivas

É importante apresentar os diferentes termos e conceitos encontrados na literatura referente ao tratamento de não-conformidades. De acordo com a norma ISO 9000 (2000), que estabelece os fundamentos e a linguagem própria para os sistemas de gestão da qualidade passíveis de certificação internacional, uma não-conformidade é definida "como o não atendimento a um requisito", que por sua vez é definido como "uma necessidade ou expectativa que é expressa, geralmente, de forma implícita ou obrigatória". De maneira geral, um qualificador pode ser usado para distinguir um tipo específico de requisito como, por exemplo, requisito de produto, requisito da gestão da qualidade, requisito do cliente. Adicionalmente, a mesma norma define um defeito "como o não atendimento a um requisito relacionado a um uso pretendido ou especificado" do produto.

Por sua vez, uma ação corretiva é definida pela norma ISO 9000 (2000) como uma ação para eliminar a causa de uma não-conformidade identificada ou outra situação indesejada. De forma geral, uma não-conformidade pode estar relacionada a um produto, a um processo de produção, ou ao sistema de gestão da qualidade.

Uma ação corretiva, em geral, envolve: a análise crítica da não-conformidade, a determinação das causas da não-conformidade, a avaliação da necessidade de ações para assegurar que aquelas não-conformidades não ocorrerão novamente, determinação e implementação das ações necessárias, registro dos resultados de ações executadas e a análise crítica das ações corretivas executadas.

Uma ação preventiva é definida como uma ação para eliminar a causa de uma não-conformidade potencial, cujo propósito é prevenir a ocorrência desta não-conformidade, enquanto a ação corretiva diz respeito a prevenir a repetição de uma não-conformidade.

Do ponto de vista de informação, uma não-conformidade pode ser caracterizada por meio dos seguintes elementos (PFEIFER, 1997):

(a) Descrição da não-conformidade estabelecida pelos atributos que envolvem as circunstâncias nas quais a não-conformidade ocorreu, sua importância e urgência;
(b) Descrição dos parâmetros que determinam a sua atribuição a um dado componente, subproduto ou produto;

(c) Descrição dos elementos do processo de análise da não-conformidade, isto é, os sintomas coletados, as possíveis e as reais causas, as possíveis e reais ações tomadas no caso;

Por outro lado, as informações e o conhecimento produzidos ao longo do processo de investigação das causas das não-conformidades, sobre as ações corretivas ou preventivas introduzidas, bem como os efeitos de sua eficácia, frequentemente não são retidos, ou são de complicada recuperação, prejudicando a aprendizagem organizacional a partir das lições aprendidas (PFEIFER et al., 1998; PFEIFER et al. 2000; LARI, 2003).

Do ponto de vista da gestão do conhecimento, a definição de lições aprendidas que é aceita de maneira geral pela comunidade envolvida foi proposta por Secchi et al. (1999), como sendo um elemento de conhecimento ou o entendimento obtido por meio da experiência. Tal elemento pode ser positivo como o sucesso de um teste ou missão, ou negativo como um acidente ou uma não-conformidade (WEBER et al., 2001 e 2002).

Neste sentido, uma lição deve ser significativa no que diz respeito ao seu impacto real ou potencial nas operações; deve ser válida de forma realista e tecnicamente correta; deve ser aplicável em termos de identificar um projeto específico, um processo, ou decisões que reduzem ou eliminam o potencial para falhas e acidentes, ou reforçar um resultado positivo.

Assim, um sistema de lições aprendidas representa uma iniciativa estruturada para gestão do conhecimento sobre um repositório de conhecimento constituído por lições aprendidas. A literatura relata o uso de sistemas de lições aprendidas em organizações governamentais, em particular nos Estados Unidos: no departamento de defesa em aplicações militares; no departamento de energia visando a prevenção de acidentes, e também nas agências espaciais norte-americana, europeia e japonesa devido aos altos custos envolvidos com as falhas de uma missão (WEBER et al., 2002).

Do ponto de vista formal, uma lição aprendida é composta por um conjunto de características classificadas em dois grupos: os elementos indexadores e os elementos reutilizáveis.

Os elementos indexadores, que permitem a recuperação baseada na aplicabilidade da lição aprendida, incluem: uma tarefa aplicável, cujo propósito é descrever em termos de uma atividade final (ação, decisão ou processo) à qual a lição é aplicável; as pré-condições que distinguem o estado particular que determina quando esta lição é aplicável.

Os elementos reutilizáveis englobam, em essência: uma sugestão de lição aprendida em função dos elementos indexadores e descrevem o que foi aprendido mediante a experiência que deveria ser repetida ou evitada; um argumento lógico (do inglês, *rationale*) que fornece ao potencial usuário uma justificativa que particulariza como esta lição foi aprendida, e pode envolver três itens: um tipo identificador (*type*) que determina a origem da lição (falha, sucesso ou recomendação), uma descrição do que aconteceu (*what*) e um resumo das causas (*why*).

Não obstante a ampla utilização de sistemas de lições aprendidas, um dos desafios no campo de pesquisa reside na dificuldade de transmitir lições do repositório de lições aprendidas para os usuários potenciais.

Do ponto de vista da representação formal das lições aprendidas e considerando a limitação reconhecida na literatura, uma possível alternativa consiste em considerá-la como a representação de um caso, usando-se a técnica de raciocínio baseado em casos (RBC). De forma simplificada, a representação de um caso consiste em dois componentes: o problema que descreve o estado do mundo quando o caso ocorreu, e a solução que descreve como resolver o problema (WATSON, 2003). A partir da ótica do raciocínio ou inferência, a descrição do problema é usada para indexar e gerenciar a tarefa de recuperação em função das medidas de similaridade consideradas, enquanto a solução prévia é reutilizada para resolver o novo problema.

Desta forma, pode-se estabelecer uma relação entre os elementos indexadores da representação da lição aprendida e o problema no contexto do raciocínio baseado em casos, e entre os elementos de reutilização e a solução, respectivamente. Permite-se assim o uso da técnica de raciocínio baseado em casos para representar e manipular os elementos das lições aprendidas.

Nesta linha, estudos foram realizados em empresas alemãs com o propósito de desenvolver formas de representação orientada a objetos para não-conformidades e métodos computacionais baseado em bancos de dados relacionais. Estas representações tinham por objetivo facilitar as atividades relacionadas ao processo de tratamento das não-conformidades e suas variantes por meio de um sistema de gestão de informações integrado ao processo de trabalho de uma organização (PFEIFER, 1997).

Por sua vez, Dhafr et al. (2006) discorrem sobre a importância e utilidade de um modelo do ciclo de vida de uma não-conformidade dentro de um processo padrão de registro e análise de não-conformidades. O propósito desse modelo é, em essência, auxiliar os envolvidos com um conjunto de estados mediante o qual uma não-conformidade ocorre, onde os estados visam auxiliar a elaboração dos relatos de não-conformidades.

Neste sentido, o ciclo de vida da não-conformidade ilustra a ordem temporal dos vários estados de uma não-conformidade, passando pelo momento quando o primeiro caso é reportado até quando o caso é solucionado, como mostra a Figura 2.1

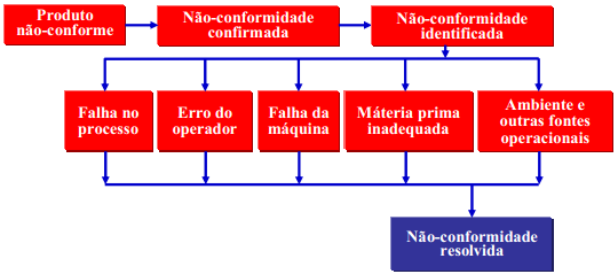

Figura 2-1 - Ciclo de uma não-conformidade
Fonte: DHAFR et al. (2006).

2.2 Conceitos essenciais do método FMEA

A Análise de Modos de Falha e seus Efeitos (FMEA) é um importante método analítico e formal de engenharia da qualidade, o qual visa, de maneira preventiva, ainda nas fases iniciais do projeto, analisar todos os possíveis modos de falha de um sistema, produto ou processo, as possíveis causas associadas a cada um destes modos de falha, bem como seus efeitos (WIRTH et al., 1996; STAMATIS, 2003; BLUVBAND, et al., 2004).

Por consequência, a partir dos resultados desta análise sistemática, os projetistas destes sistemas, produtos ou processos podem rever seus projetos com o objetivo de propor ações para: eliminar ou reduzir, em grande medida, a probabilidade de ocorrência destes modos de falha ou, ainda, aumentar a probabilidade de detecção do modo de falha associado a uma determinada causa (WIRTH et al., 1996; STAMATIS, 2003; BLUVBAND, et al., 2004).

Este método foi desenvolvido em meados da década de1960, no contexto dos projetos aeroespaciais norte-americanos, em particular, no projeto Apollo da NASA (*National Aeronautics and Space Administration*).

Posteriormente, o FMEA foi adotado pela indústria em geral tanto em aplicações civis quanto militares, e é amplamente discutido e fundamentado em publicações como normas internacionais e referenciais

de setores industriais específicos, como por exemplo: IEC 60812:2006 - *Analysis techniques for system reliability: Procedure for failure mode and effect analysis* (FMEA); SAE J-1739:2002 - *Potential Failure Mode and Effects Analysis* e AIAG *Reference Manual: Failure Modes and Effects Analysis* (2001).

Neste contexto, a norma IEC 60812 (2006) define o FMEA como um procedimento sistemático de análise de um sistema que visa identificar todos os possíveis modos de falha, suas causas e respectivos efeitos sobre o desempenho do sistema.

É importante notar que o desempenho citado pode se referir a: (a) a uma montagem subsequente; ou (b) sobre o sistema como um todo; ou (c) sobre um processo específico. Nesta definição, o termo sistema é usado como uma representação do hardware e do software (com suas interações) ou de um processo.

A norma IEC 60812 (2006) sugere o uso do método preferencialmente nas fases iniciais do ciclo de desenvolvimento, pois nesta fase os custos de remoção ou mitigação dos modos de falha são em geral mais reduzidos. Assim, esta análise deve ser iniciada tanto mais cedo quanto possível, bastando o sistema estar suficientemente definido para ser representado na forma de um diagrama de blocos, onde o desempenho de seus elementos possa ser definido e avaliado.

Cumpre observar que na literatura da área, a letra ou símbolo "C" adicionada ao termo FMEA denota que a análise do modo de falha produz também uma análise da criticidade (*criticality*). Esta extensão lógica do método é definida na literatura como FMECA (*Failure Mode Effects and Criticality Analysis*). Neste método, cada possível modo de falha é analisado de forma a determinar tanto as causas e efeitos sobre o desempenho de um dado sistema quanto à classificação dos modos de falha, considerando que uma medida de criticidade quantifica a severidade e probabilidade de ocorrência deste modo de falha (IEC, 2006).

Entretanto, a literatura apresenta aplicações do próprio método FMEA, nas quais são agregados três componentes voltados para o estabelecimento de prioridade nas ações: a severidade que expressa a gravidade dos efeitos de um dado modo de falha, a probabilidade de ocorrência de uma causa específica deste modo de falha, e a probabilidade de detecção associada à capacidade de controle empregado no sistema em detectar a ocorrência desta causa específica (STAMATIS, 2003; AIAG, 2006, SAE, 2002).

Stamatis (2003) revela que o método FMEA pode ser classificado em quatro tipos principais:

- FMEA de sistemas (*System FMEA*), usado para analisar sistemas e subsistemas nos estágios iniciais de projeto e concepção do sistema. Este tipo de FMEA concentra-se nos possíveis modos de falha relacionados às deficiências do sistema com relação às funções previstas para o sistema, bem como para as interações do sistema e seus elementos;
- FMEA de projeto (*Design FMEA*), usado para análise de produtos antes de sua liberação para a manufatura. Este tipo concentra-se nos possíveis modos de falha relacionados a deficiências no projeto do produto em relação às funcionalidades do produto ou de seus componentes;
- FMEA de processo (*Process FMEA*), usado para analisar os processos de manufatura e montagem concentrando-se nos modos de falha decorrentes de deficiências nos processos em relação às funções especificadas para os processos ou operações;
- FMEA de serviço (*Service FMEA*), usado para analisar serviços antes que estes afetem o cliente. Este tipo concentra-se nos modos de falha associados às tarefas e processos de trabalho.

Na atualidade, este método é amplamente usado nas áreas de manutenção e manufatura, e sua aplicação é indicada, e em alguns casos exigida em contratos, como um elemento essencial para os sistemas de gestão da qualidade, dentre as quais tem-se:

- ISO 9004:2000 - Sistemas de gestão da qualidade: Diretrizes para melhoria de desempenho;
- ISO 9001:2000 – Sistemas de gestão da qualidade: Requisitos;
- ISO TS 16949:2002 – Sistemas de gestão da qualidade: Requisitos particulares para a aplicação da norma ISO 9001:2000 para a produção automotiva e serviços relevantes e componentes de organizações fornecedoras (especificação técnica);

É importante notar que as etapas recomendadas estão em consonância àquelas descritas nos métodos de análise e solução de problemas apresentados na literatura. Todavia, deve-se observar, em especial, que esta referência normativa agrega as etapas à recomendação inovadora de consultas baseadas em sistemas de lições aprendidas, como mostram as Figuras 2.2 e 2.3.

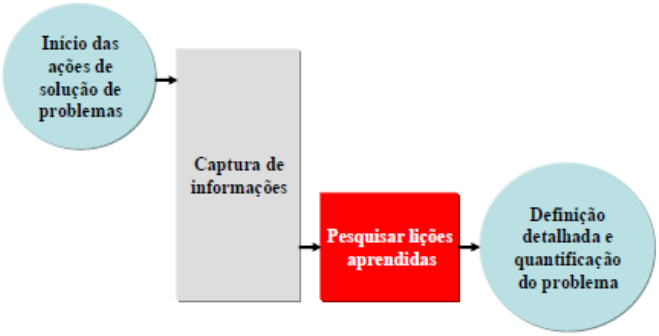

Figura 2-2 - Etapa de identificação do problema.
Fonte: AIAG, (2006).

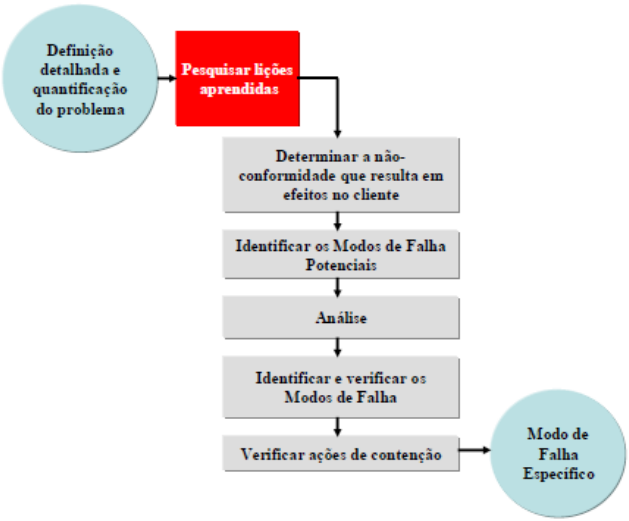

Figura 2-3 - Etapa de análise de modos de falha.
Fonte: AIAG, (2006).

A Figura 2.4 apresenta as etapas de aplicação e desenvolvimento de uma análise de modos de falha e seus efeitos para processos de manufatura e montagem (SAE, 2002; AIAG, 2006).

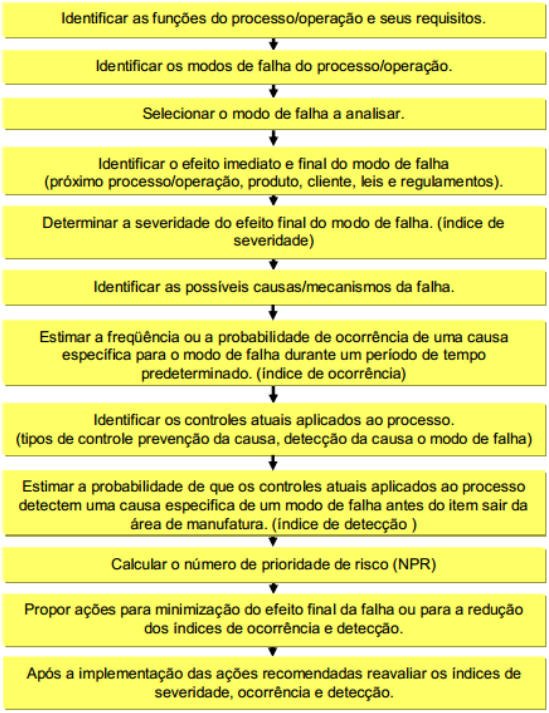

Figura 2-4 - Fluxograma das etapas do FMEA de processo
FONTE: AIAG, (2006).

É importante ressaltar que a aplicação do método FMEA é precedida por uma decomposição hierárquica do processo de manufatura em seus elementos básicos (TEOH e CASE, 2004a e 2004b), e a norma IEC (2006) recomenda o uso de diagramas de blocos para ilustrar esta decomposição. Esta recomendação é feita porque a análise deve iniciar com os elementos do nível mais baixo e, dentro desta ótica, o efeito de uma falha em um nível mais baixo pode tornar-se a causa de um modo de falha em um nível mais alto. Assim, a análise deve ser feita de forma *bottom-up* até que o efeito final sobre o sistema seja identificado.

A Figura 2.4 mostra que a primeira etapa consiste em determinar a função pela qual o processo deve responder (ou uma operação de acordo com a decomposição adotada). A determinação desta função é essencial para o desenvolvimento do método, e baseia-se em uma descrição prévia do propósito e dos objetivos do processo ou operação.

Estes objetivos devem ser derivados diretamente das especificações e requisitos de projeto do processo atual, as quais podem ser expressas na forma de um diagrama de fluxo que identifica sequencialmente o fluxo de operações e interações com pessoas, máquinas, equipamentos e ferramentas (IEC, 2006).

O modo de falha potencial representa uma descrição física da maneira pela qual potencialmente o processo pode falhar em atender as especificações e requisitos para os quais foi projetado. E, de acordo com a norma SAE J-1739:2002, ele representa a descrição de uma não-conformidade em uma operação específica, podendo ser a causa associada a um modo de falha em uma operação subsequente dependendo da complexidade do processo em questão.

A premissa adotada na aplicação do método é que uma falha pode ocorrer, mas não necessariamente vai ocorrer. Neste sentido, os especialistas devem ser capazes de responder as seguintes questões: (a) como um processo pode falhar em atender as suas especificações? (b) independentemente das especificações, o que deve ser considerado como indesejado pelo cliente (operação ou processo subsequente, assistência técnica ou cliente final)?

A norma SAE J-1739:2002 revela que uma comparação com processos similares e uma revisão de reclamações de clientes relacionadas a componentes similares é recomendada como ponto de partida.

O efeito potencial da falha é a consequência da falha sobre o cliente (próximo processo, operação, produto, cliente e leis e regulamentos), e descreve os efeitos da falha em termos do que o cliente pode notar ou experimentar.

A aplicação do método PFMEA, em geral, analisa os efeitos de falha considerando isoladamente cada modo de falha, como mostra o Diagrama de Ishikawa mostrado na Figura 2.5. Todavia, Pickard et al. (2005) apresentam um procedimento que permite a consideração de modos de falha múltiplos mantendo as características do método FMEA.

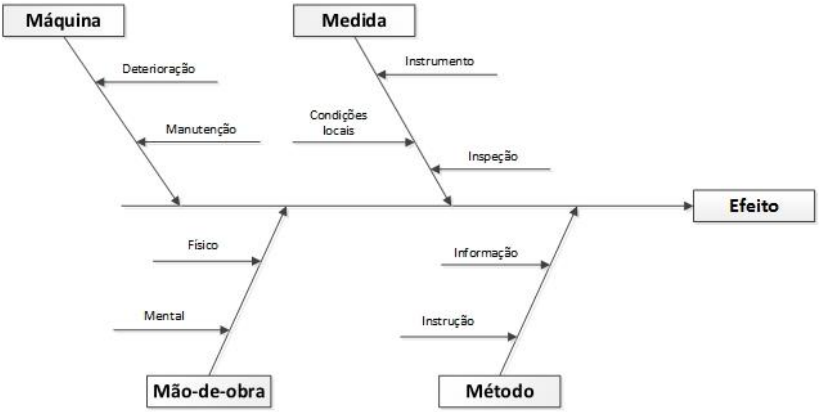

Figura 2-5 – Diagrama de Ishikawa para formalização do PFMEA
FONTE: Criado pelo autor.

Os controles atuais empregados nos processos são descrições dos métodos destinados à prevenção da ocorrência da causa da falha ou à detecção da ocorrência do modo de falha quando considerado isoladamente. Estes métodos envolvem o uso de dispositivos como: dispositivos com sensores automáticos, instrumentação de controle ou outras indicações como alarmes sonoros ou visuais, bem como gráficos de controle estatístico do processo ou processos de amostragem.

Entretanto, quando as causas específicas de um modo de falha ou o próprio modo não são detectáveis pelos métodos disponíveis e, portanto, o processo continua a operar, a análise deve estender-se para determinar os efeitos sobre o próximo nível, ou mesmo avaliar se este modo de falha pode combinar-se com outros modos, produzindo efeitos ainda mais severos.

Dentro desta perspectiva, a detecção é um índice que expressa a probabilidade de que os atuais métodos de controle irão detectar uma causa específica ou o modo de falha antes que os produtos não-conformes sejam liberados. A determinação do índice de detecção assume que a falha ocorre devido a uma dada causa, e estima a capacidade dos métodos de detecção baseando-se em tabelas de classificação.

Cumpre observar que uma profunda revisão do método FMEA está além da intenção desta tese, cujo objetivo é apresentar os conceitos essenciais do método visando permitir um entendimento da abordagem proposta no trabalho.

2.2.1 Desafios conceituais à representação do conhecimento no domínio de FMEA

Devido à sua relevância, o método de Análise de Modos de Falha e seus Efeitos (FMEA) tem sido discutido profundamente na literatura ao longo dos últimos quarenta anos, envolvendo aspectos como metodologias e procedimentos de aplicação, além de inúmeros relatos de aplicação nas mais diversas áreas.

Não obstante esta grande produção técnica e científica, destaca-se uma característica comum, isto é, que o FMEA, quando conduzido de forma apropriada, resulta em um conjunto profundo de informações sobre os sistemas, produtos e processos de uma organização. Portanto, constitui-se em uma fonte valiosa de informações e conhecimento que pode proporcionar suporte técnico à detecção antecipada de pontos fracos em um projeto, redução dos custos ao longo do ciclo de vida do produto e menores níveis de modificações durante a fase de produção (WIRTH et al., 1996; STAMATIS, 2003; TEOH e CASE, 2004a e 2004b; DITTMANN et al., 2004; MENDES et al., 2013).

Tradicionalmente, os elementos de uma análise FMEA são adquiridos a partir da visão de especialistas de diferentes áreas de atuação, tais como: projeto, planejamento de processos, controle e garantia da qualidade, serviços pós-venda e atendimento ao cliente, entre outros. Esses elementos são posteriormente registrados em planilhas ou bancos de dados, normalmente na forma de linguagem natural. Assim, este valioso conhecimento obtido, normalmente com um elevado custo, dificilmente pode ser reutilizado, pois os componentes, funções e modos de falhas, em geral, não são organizados semanticamente, e seu significado dependerá, necessariamente, da interpretação humana. Além disto, a grande quantidade informações e conhecimentos decorrentes de análises FMEA já realizadas torna a tarefa de compartilhamento e reutilização imprecisa e improdutiva (DITTMANN et al., 2004; TEOH e CASE, 2004a, 2004b).

Neste cenário, Dittmann et al. (2004) e Yan et al (2014) propõem o uso de ontologia como uma alternativa para modelar e tratar o conhecimento decorrente do FMEA de forma a superar esta dificuldade. O conceito de ontologia usado neste contexto tem suas raízes na área de pesquisa da inteligência artificial, e uma das definições mais conhecidas e aceitas do termo revela que uma ontologia é a especificação explícita de uma conceitualização (GRUBER, 1993).

É importante observar que esta definição, em geral, está relacionada com o compartilhamento de conhecimentos no âmbito de

sistemas baseados em conhecimento e envolve a descrição de forma concisa, comum e declarativa de todos os conceitos, relações e regras existentes em um dado domínio de problema (DITTMANN et al., 2004).

Adicionalmente, uma das principais vantagens das ontologias consiste em permitir o compartilhamento do conhecimento de um dado domínio, mediante sua implementação usando-se linguagens de representação formais. Estas linguagens baseiam-se geralmente em lógica, as quais possibilitam inferências sobre o conhecimento representado, bem como facilitam o processo de reutilização do conhecimento por meio de sistemas computacionais (DITTMANN et al., 2004) e (YAN et al., 2014).

2.3 Tecnologia de agentes em sistemas de produção

Na literatura, as aplicações potenciais de sistemas baseados em agentes, em geral, podem ser agrupadas em três amplas categorias (LUCK et al., 2004):

- Agentes assistentes pessoais que são devotados à captura de informações ou à execução de ações em nome de usuário humano na internet;
- Sistemas multiagentes em apoio à decisão;
- Sistemas de simulação multiagentes, em que o sistema multiagentes é usado como um modelo para simular algum domínio do mundo real.

Na área de manufatura, um conjunto de novos requisitos a serem considerados durante o projeto de sistemas de manufatura da próxima geração tem motivado pesquisas visando o desenvolvimento de aplicações com a tecnologia de agentes, tais como:

Integração empresarial: com a intenção de apoiar a concorrência global e rápida resposta ao mercado consumidor, empresas de manufatura individuais ou coletivas deverão buscar a integração no que diz respeito aos seus sistemas de gestão, e extensível aos seus parceiros comerciais via redes de comunicação;

Organizações distribuídas: para uma eficaz integração empresarial através de organizações distribuídas, serão necessários sistemas baseados em conhecimento apropriados para a gestão da demanda;

Ambientes heterogêneos: os novos sistemas de manufatura deverão incluir hardware e software heterogêneos tanto nos ambientes informacionais como de manufatura;

Interoperabilidade: ambientes informacionais heterogêneos podem usar diferentes linguagens de programação, representar dados em diferentes linguagens de representação e modelos e, ainda, operar em diferentes plataformas de sistemas operacionais. Todavia, os subsistemas e componentes, em tais ambientes, devem ser capazes de operar conjuntamente e de interagir entre si;

Arquitetura dinâmica e aberta: admitirá aos sistemas a possibilidade de integrar novos subsistemas (software, hardware ou dispositivos de manufatura) ou mesmo removê-los sem interromper ou reinicializar o ambiente de trabalho;

Cooperação: as empresas de manufatura terão que cooperar plenamente com seus fornecedores, parceiros comerciais e clientes para suprir materiais, componentes, comercialização de produtos finais, entre outros;

Integrar profissionais humanos com software e hardware: pessoas e computadores necessitam ser integrados para trabalhar coletivamente em vários estágios do desenvolvimento de produtos, bem como ao longo de todo o ciclo de vida do produto, proporcionando rápido acesso aos conhecimentos e informações necessárias;

Agilidade: significativa atenção deve ser dispensada à redução de tempo de ciclo de um produto de forma a permitir uma rápida resposta ao consumidor. A manufatura ágil é caracterizada pela habilidade em adaptar-se a ambientes com contínuas e inesperadas mudanças, por meio da rápida reconfiguração das plantas industriais, bem como pela interação com outros sistemas heterogêneos de outros parceiros comerciais;

Escalabilidade: significa que recursos podem ser adicionados à organização quando necessário;

Tolerância a falhas: os novos sistemas de manufatura deveriam ser tolerantes a falhas tanto no nível do sistema quanto no nível dos subsistemas, assim como detectar a falha do sistema e restaurar a condição normal do sistema em qualquer nível, e minimizar o impacto da falha sobre o ambiente de trabalho.

Atualmente, os sistemas de manufatura representam, incontestavelmente, um dos mais prósperos campos para a aplicação da tecnologia de agentes, em grande parte devido à capacidade intrínseca desta tecnologia em dar suporte à integração de sistemas de informação de manufatura (PAOLUCCI e SACILE, 2005). Neste sentido, a integração pode se dar em relação a uma dimensão vertical, isto é,

permitindo a integração entre diferentes fábricas e processos de negócios, bem como em relação a uma dimensão horizontal, permitindo gerenciar o fluxo interno de informações levando em consideração o aspecto de relacionamento com os clientes e da gestão da cadeia de suprimentos, representada de forma esquemática na Figura 2.6.

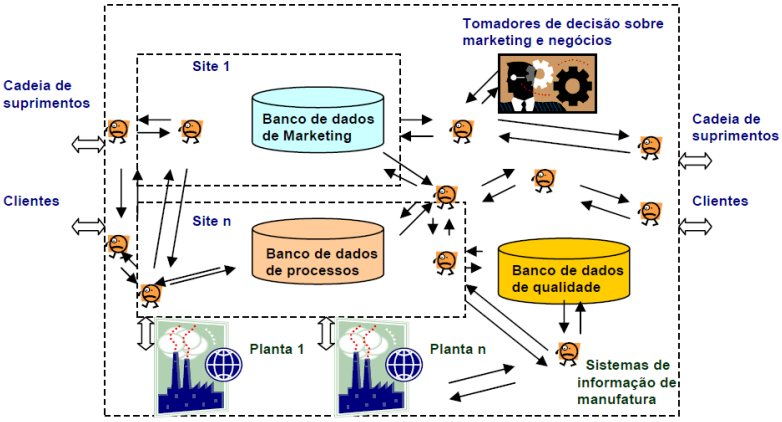

Figura 2-6 - Integração do sistema de informações de manufatura
Fonte: PAOLUCCI e SACILE (2005).

No âmbito deste conjunto, a principal capacidade requerida para este tipo de sistema e suas respectivas bases de dados é a integração dos dados de projeto, passando pelas ordens de produção e tarefas de sequenciamento. Além disso, ele deve calcular as datas de fornecimentos, alocação de recursos e os prazos de trabalho dos diversos times nos vários sites e ao longo do ciclo de vida de produto, neste caso do projeto à entrega do produto.

No que diz respeito ao desempenho requerido pelo processo de recuperação de informação de manufatura, devido às extensões geográficas e aos variados desdobramentos das modernas organizações de produção, percebe-se que um sistema de informações centralizado não representa mais uma alternativa apropriada de solução.

Neste contexto, os principais conceitos relacionados aos sistemas de informação de manufatura envolvem o uso do potencial da *Internet* e das *Intranets*, como uma infraestrutura de comunicação para a distribuição e compartilhamento de informações entre e dentro das modernas organizações de produção. Neste sentido, o acesso a informações relacionadas à manufatura em tempo real permite o

desenvolvimento e o uso de novos métodos para monitoramento e controle de processos de manufatura de forma remota (SHEN et al., 2005).

2.4 Ontologias como suporte à gestão do conhecimento

De acordo com Gunendran e Young (2006), a utilização de ontologias para compartilhamento de conhecimento é uma abordagem bastante atrativa atualmente e, neste sentido, revelam que ontologias são amplamente usadas em diferentes áreas de pesquisa, tais como: engenharia do conhecimento, inteligência artificial, aplicações de gestão do conhecimento, processamento de linguagem natural, comércio eletrônico, integração inteligente de conhecimento, integração e projeto de banco de dados, recuperação de informações e *web* semântica.

O trabalho de Gunendran e Young (2006) reporta a abrangência e importância das aplicações de ontologias como suporte à gestão do conhecimento e informações em consonância aos requisitos da engenharia de manufatura.

Mendes et al (2013), por sua vez, argumentam que a engenharia ontológica pode ser considerada como uma metodologia que trata essencialmente da conceitualização do conhecimento do mundo real por meio dos conceitos e restrições semânticas a esses conceitos conjuntamente com sofisticadas teorias e tecnologias. Assim, a engenharia ontológica pesquisa formas e meios para facilitar o compartilhamento e reuso de conhecimentos de informações, sendo que neste trabalho tais informações são relativas à engenharia de manufatura.

2.4.1 Definições de ontologia

Em primeiro lugar, é importante perceber que a ontologia é objeto de pesquisa em diferentes áreas da ciência, o que explica, em parte, porque diferentes definições de ontologia são apresentadas na literatura.

Neste cenário, considera-se que a área multidisciplinar de pesquisa em sistemas de informação tomou emprestado o termo ontologia da filosofia clássica e o reinterpretou para uma forma mais adequada à área (ZÚÑIGA, 2001). Portanto, é importante frisar que as definições apresentadas nesta subseção derivam particularmente da área da ciência da computação, em especial, da subárea de representação do

conhecimento e inteligência artificial, pois as definições de ontologia no campo da filosofia estão além do escopo desta tese.

Zúñiga (2001) propõe uma definição adequada à área de representação de conhecimento, mas buscando reconciliá-la com os objetivos das ontologias filosóficas. Assim, neste trabalho adota-se a definição na qual uma ontologia é "uma teoria axiomática que se torna explícita mediante o uso de uma linguagem formal específica, e ainda sendo projetada para pelo menos uma aplicação prática e específica". Por consequência, ela descreve a estrutura de um domínio específico de objetos respondendo por um sentido pretendido de um vocabulário ou protocolos que são empregados pelos agentes de um domínio sob investigação.

Todavia, cumpre observar que na literatura não há uma definição formal universalmente aceita, embora a ISO (*International Organization for Standardization*) apresente uma definição para ontologia no texto da norma ISO 18629-1:2004 - *Industrial automation systems and integration - Process specification language: Part 1: Overview and basic principles*, onde uma ontologia é "um léxico de terminologia especializada em conjunto com alguma forma de especificação do significado dos termos no léxico".

A norma ISO 18629-1 (2004) busca fornecer uma visão geral das diferentes séries de partes da norma, em que é definida a linguagem de especificação de processo (*Process Specification Language* – PSL). A linguagem PSL visa identificar e definir formalmente a estrutura dos conceitos semânticos intrínsecos à captura e troca de informações de processo relacionadas à manufatura de produtos discretos.

A mesma norma observa que uma ontologia é projetada para representar conceitos primitivos adequados à descrição dos processos básicos de manufatura, engenharia e de negócios. E que o foco da ontologia não é somente nos termos, mas também em seu significado, pois os termos somente podem ser compartilhados se houver concordância sobre os seus significados.

Neste sentido, a norma revela que é a semântica intencional de um termo que está sendo compartilhada e não simplesmente o termo em si. Portanto, qualquer termo usado sem uma definição explícita é uma fonte de possível ambiguidade e confusão e, portanto, o desafio para uma ontologia, segundo a norma, reside em fornecer uma caracterização rigorosa do processo de informação, bem como uma expressão precisa das propriedades lógicas básicas da informação.

2.5 Sistemas de Raciocínio Baseado em Casos aplicados à gestão do conhecimento

Na última década a técnica de Raciocínio Baseado em Casos (RBC) evoluiu de uma área de pesquisa específica da Inteligência Artificial (IA) para um campo de interesse amplo no desenvolvimento de Sistemas de Gestão do Conhecimento[4] (WATSON, 2003).

Atualmente, os sistemas de raciocínio baseado em casos são aplicados a vários tipos de problemas e em diferentes domínios. Entretanto, a literatura da área revela um conjunto de características particulares para os tipos de problemas e domínios que determinam a melhor aplicabilidade da técnica de raciocínio baseado em casos (WATSON, 2003; WANGENHEIM e WANGENHEIM, 2003), dentre as quais tem-se as seguintes situações:

- Domínio de aplicação em que não dispõe-se de modelos gerais de conhecimento estabelecidos;
- Onde existem exceções e novos casos disponíveis;
- Os casos, em geral, são recorrentes;
- Casos prévios relevantes podem ser obtidos.

Shiu e Pal (2004) observam que os principais aspectos relacionados à aplicabilidade da técnica de raciocínio baseado em casos referem-se a:

- Redução do esforço relativo à tarefa de aquisição de conhecimento, pois a técnica não requer o estabelecimento de um modelo geral de conhecimento sobre o domínio, ou mesmo organizar um conjunto de regras de especialistas humanos.
- Evitar a repetição de erros já cometidos no passado, pois a técnica tem por finalidade conservar o conhecimento sobre as falhas, bem como sucessos e, ainda, as razões para a ocorrência.
- Permitir a flexibilidade na modelagem do conhecimento, pois os sistemas baseados em modelos causais, devido à sua rigidez na formulação e modelagem do problema, dificultam a solução de problemas que se encontram na fronteira de seu conhecimento ou escopo. Em contraste, o RBC usa as experiências do passado como domínio do conhecimento e pode, com frequência, oferecer soluções razoáveis por meio de uma adaptação apropriada.
- Em situações onde o conhecimento sobre o domínio é insuficiente para construir um modelo causal ou preparar um conjunto de

[4]Do inglês, *Knowledge Management Systems.*

heurísticas, em particular para domínios não plenamente entendidos, definidos ou modelados, um sistema de RBC pode ser desenvolvido a partir de um pequeno número de casos (pilotos). Assim, não é necessário um entendimento profundo da teoria subjacente ao domínio de conhecimento.

- A possibilidade de prever o possível sucesso de uma dada solução gerada em função das informações armazenadas junto ao caso prévio (nível de sucesso) e das diferenças entre o contexto prévio e o contexto atual de aplicação.
- A capacidade de aprender ao longo do tempo, à medida que o sistema RBC é usado em novas situações, novas soluções são geradas e testadas no mundo real, e o seu nível de sucesso é determinado. Assim, novos casos podem ser adicionados à base de casos, auxiliando novas soluções de modo cada vez mais refinado.
- A solução obtida a partir de um caso anterior pode ser diretamente aplicada ao problema considerado, ou modificada de acordo com as diferenças entre os dois casos. O processo de identificação de casos similares, adaptação de soluções e aprendizagem a partir de experiências, pode ser conduzido e apoiado por conhecimentos genéricos sobre o domínio por meio de modelos de conhecimento profundo, superficial ou compilado ou ser baseado em uma aparente similaridade.

Para ilustrar estes aspectos, o ciclo de raciocínio baseado em casos pode ser apresentado por meio de um modelo dinâmico representando uma visão orientada ao processo, cujo objetivo é enfatizar a ideia de ciclo em etapas sequenciais. Permite-se assim uma visão global e externa do processo, no qual são identificados os subprocessos, suas interdependências e produtos. Neste sentido, Aamodt e Plaza (1994) sugerem um modelo composto dos quatro processos seguintes (4RE's)[5], como mostra a Figura 2.7:

- Recuperação do caso ou casos mais similares;
- Reutilização da informação ou conhecimento do caso para resolver o problema;
- Revisão da solução proposta;
- Retenção de partes desta experiência com mais probabilidade de serem úteis na solução de futuros problemas.

[5]Em referência ao mnemônico da primeira sílaba dos subprocessos (isto é, "RE").

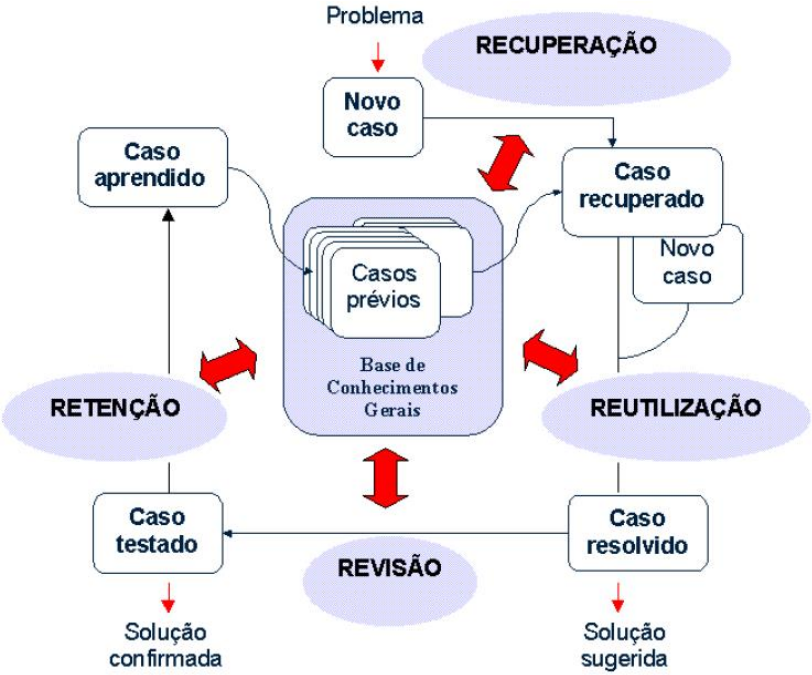

Figura 2-7 - Ciclo do Raciocínio Baseado em Casos.
Fonte: AAMODT e PLAZA (1994).

Neste modelo dinâmico, uma descrição inicial de uma situação problema define um novo caso, o qual deverá ser usado pelo processo de recuperação para pesquisar e identificar um caso similar em uma coleção de casos prévios. O caso recuperado é combinado, então, com o novo caso, por meio do processo de reutilização, tornando-se assim um caso resolvido, isto é, uma solução proposta para o problema inicial.

Posteriormente, o processo de revisão deverá testar a solução proposta no ambiente do mundo real, de modo a avaliar a sua pertinência e ajustar a solução quando esta falhar.

No final, o processo de retenção deve armazenar a informação útil para futuras reutilizações, e a base de casos anteriores deve ser atualizada com o novo caso aprendido.

Neste modelo, a base de conhecimento geral indicada normalmente contribui para o desenvolvimento do ciclo RBC. Esta contribuição pode ser representada em um extremo como sendo insignificante e, em outro extremo, uma contribuição essencial em função do método de RBC.

Assim, do ponto de vista da gestão do conhecimento, a técnica de raciocínio baseado em casos pode ser aplicada, de forma apropriada, na construção de uma memória corporativa considerando a sua adequação aos requisitos de aquisição, análise, preservação e uso do conhecimento (WATSON, 2003).

2.5.1 Tarefa de Recuperação de Casos

O objetivo da recuperação de casos é encontrar um caso ou um pequeno conjunto de casos na base de casos que contenha uma solução útil para o problema. Por exemplo, dada a descrição de uma não-conformidade, como por exemplo uma bolha no processo de extrusão, um sistema RBC deverá ser capaz de recuperar um caso descrevendo uma ou mais soluções apropriadas ao problema (por exemplo, *trocar disco*). Para realizar essa recuperação, é necessário combinar a descrição da não-conformidade atual com os problemas armazenados na base de casos.

2.5.2 Tarefa de Reutilização de Casos

Em essência, a tarefa de reutilização de um caso recuperado da base de casos durante a tarefa de recuperação consiste em transferir ou adaptar o conhecimento real contido no caso (descrição da solução) para o novo caso. Neste sentido, esta tarefa pode ser subdividida em duas subtarefas: copiar e adaptar a solução considerando as diferenças entre o novo caso e o caso recuperado, bem como identificando qual parte da solução pode ser transferida para o novo caso.

A solução originada está disponível para ser aplicada no ambiente do mundo real ou submetida a um teste com um especialista humano, ou ainda aplicada em um programa de simulação de modo a avaliar a solução originada. Neste contexto, se a solução originada não for adequada, surge uma oportunidade para a correção da falha e, consequentemente, a possibilidade da aprendizagem sustentada do sistema de RBC.

2.5.3 Tarefa de Revisão de Casos

A primeira etapa do processo de revisão envolve a subtarefa de avaliação, que em geral é fora do sistema de raciocínio baseado em casos, cujo resultado decorre da aplicação da solução no ambiente do mundo real. Estes resultados, inclusive, podem não ser obtidos de forma imediata, dependendo do domínio de aplicação (AAMODT &PLAZA, 1994).

A segunda etapa concentra-se em reparar a falha detectada, que pode envolver a identificação de parte da solução que contém a falha e a recuperação ou a geração de uma explicação coerente para a sua ocorrência. Assim, é incluída, internamente, uma memória de falha que é usada na tarefa de reutilização para predizer possíveis deficiências de planos de solução. Desta forma, a detecção de erros é usada na fase de adaptação como uma ação preventiva, pois tais erros podem ser previstos, tratados e evitados.

2.5.4 Tarefa de Retenção de Novos Casos

A tarefa de retenção de novos casos tem por objetivo incorporar, de forma contínua e apoiada, os conhecimentos úteis revelados a partir do processo de solução de um novo caso.

O processo de aprendizagem pode ocorrer a partir do sucesso ou do fracasso de uma solução originada, tendo início com as saídas das subtarefas de revisão de casos: avaliação e reparo de solução. Este processo envolve a seleção das informações da descrição do novo caso e sua respectiva solução validada, que devem ser retidas, bem como estabelecer a forma de retenção, isto é, a forma de indexação do novo caso, e como integrar este novo caso na estrutura da memória de casos.

Assim, os sistemas baseados em conhecimento, em particular os sistemas de raciocínio baseado em casos, precisam ser ricos em conhecimento, pois os métodos de inferência, apesar de engenhosos e cientificamente validados, não são perfeitos.

2.6 Mercado do Alumínio

O alumínio é produzido comercialmente há cerca de 150 anos e, nesse curto período, sua indústria se expandiu e está presente em seis regiões geográficas - África, América do Norte, América Latina, Ásia, Europa e Oceania. No total, são 46 países que produziram, em 2006, aproximadamente 34 milhões de toneladas de alumínio primário, conforme dados do *World Metal Statistics*. O Brasil é o sétimo maior produtor mundial de alumínio primário, precedido pela China, Rússia, Canadá, Austrália, Estados Unidos e Índia.

A demonstração da importância da indústria brasileira do alumínio no cenário mundial está na sua participação no mercado global. O Brasil, além da terceira maior jazida de bauxita do planeta, é o quarto maior produtor de alumina, e ocupa a quinta colocação na exportação de alumínio primário/ligas.

No mercado interno, a maior parte do alumínio e seus produtos são aplicados nos segmentos de embalagens e transportes. Na sequência, vem os segmentos de eletricidade, construção civil, bens de consumo, máquinas e equipamentos e outros, com um faturamento de R$ 14,7 bilhões em 2010. A produção de semimanufaturados de alumínio no Brasil está concentrada na região sudeste do Brasil. Minas Gerais, São Paulo e Rio de Janeiro abrigam empresas produtoras de chapas, folhas, extrudados e cabos. A indústria também está presente nos estados do Pará, Maranhão, Ceará, Pernambuco, Bahia, Mato Grosso, Paraná, Santa Catarina e Rio Grande do Sul, que também possuem unidades de produção (ABAL, 2014).

Não obstante todos os recursos científicos e tecnológicos empregados, a probabilidade de ocorrência de não-conformidades ao longo do ciclo de vida não pode ser totalmente eliminada. Em geral, não-conformidades surgem durante o processo de extrusão em decorrência da complexa inter-relação entre o produto e a ferramenta, entre o tarugo e processo, as quais são de difícil modelagem analítica e, frequentemente, tornam a tarefa de descobrir a fonte da não-conformidade e sua respectiva solução demorada e dispendiosa.

Na prática, durante o processamento, mesmo quando a prensa é adequadamente ajustada (*setup*), verificações periódicas da qualidade da peça são necessárias como uma medida de garantia da qualidade, tendo em vista as inúmeras variáveis envolvidas no complexo processo de extrusão de alumínio.

Em relação às não-conformidades que ocorrem no processo de extrusão de alumínio, relatórios de não-conformidades de uma empresa que aplica o processo de extrusão de alumínio apresentam uma classificação que será explorada no tocante aos modos de falhas, tais como: bolhas, arrancamento, abaulamento, *coring*, faixa, angularidade, rebarba, trepidação, falta de retidão (banana *shape*), ondas, ponta de flecha (*chevron*), são as não-conformidades exploradas neste trabalho de tese.

Nesta perspectiva, o estudo de caso previsto nesta tese deverá explorar a associação destes modos de falha, os quais serão usados em conjunto com os valores dos parâmetros (denominados “descritores”), como descritores de problemas no contexto da técnica de raciocínio baseado em casos. Desta forma, este conjunto de descritores deve servir como elemento comum para a recuperação de casos por similaridade nas diferentes bases, articulando os conhecimentos relativos às experiências prévias no tratamento de não-conformidades, bem como à aplicação do

método de Análise dos Modos de Falha e Efeitos (FMEA) nas ligas da série 6000.

Como o foco do trabalho é o método pelo qual são solucionados os problemas de não-conformidades, para o leitor aprofundar-se sobre o processo de extrusão de alumínio recomenda-se consultar as referências FILHO et al. (2011) e SAHA (2000).

2.7 Trabalhos anteriores que abordam o tema proposto

Na busca por publicações que tenham a mesma ênfase ou similaridade, uma delas corresponde ao trabalho de METAXIOTIS et al. (2002), que desenvolveram sistemas computacionais para a programação da produção, enquanto o artigo de Wong e Monaco (1995) buscou apresentar uma revisão sobre sistemas especialistas aplicados a negócios, tendo analisado publicações entre 1977 e 1993 em diversos temas, que incluem: auditoria, finanças, recursos humanos, sistemas de informação, *marketing*/distribuição, estratégia de gerenciamento e produção/operação envolvendo sistemas especialistas.

Observa-se que desde 1977 têm sido desenvolvidos sistemas computacionais para o suporte à manufatura, e um breve resumo de alguns desses trabalhos é apresentado abaixo.

Nargarajan e Abdulkareem (1990) desenvolveram o DCS (*Data Capture System*), que monitora o desempenho de máquinas em uma fábrica. A partir de cada máquina, que pode corresponder a qualquer tipo de processo de fabricação, o DCS obtém dados que descrevem o estado da máquina. Os dados obtidos são então analisados para possíveis melhorias na produtividade. O DCS possui um módulo chamado *Expert Data Capture System* (CDE), o qual é composto por um conjunto de regras em diferentes bases de conhecimento referentes a diversos processos, e respectivos mecanismos de inferência que são executados de maneira a solucionar um objetivo específico. Eles contribuem, de forma coletiva, para sugerir melhorias no gerenciamento do chão de fábrica.

Brandt et al. (2008) desenvolveram um sistema para o processo de extrusão de borracha que utiliza a técnica de mineração de dados, mais conhecida como *data mining*, que é o processo de explorar grandes quantidades de dados à procura de padrões consistentes, como regras de associação ou sequências temporais, para detectar relacionamentos sistemáticos entre variáveis, detectando assim novos subconjuntos de dados. Este é um tópico recente em ciência da computação, em que são utilizadas várias técnicas que incluem estatística, recuperação de informação, inteligência artificial e reconhecimento de padrões. Este

sistema não tem a capacidade de verificação e validação do conhecimento automático, ou seja, o metaconhecimento (que é o conhecimento sobre o conhecimento). Apesar da estrutura apresentada, há a necessidade de um especialista humano para efetuar a verificação e a validação.

Butdee et al. (2009) descrevem um sistema para o planejamento do projeto de ferramentas para extrusão de alumínio baseado em redes neurais, sistema este que buscou diminuir o tempo de projeto de uma nova ferramenta, além de corrigir problemas em projetos evitando refugos na produção de perfis decorrentes de uma ferramenta mal projetada. Segundo BUTDEE et al. (2009), o sistema foi testado e obteve sucesso, resultando na redução do tempo de planejamento e projeto de ferramentas.

He et al. (2012) desenvolveram um sistema para a simulação de perfis complexos utilizando a técnica de elementos finitos, concluindo por meio de experimentos que o sistema pode prever de maneira eficaz a tendência de deformação do processo de extrusão, para que se consiga fabricar um molde que, quando colocado em produção, permita um escoamento equilibrado do alumínio, diminuindo assim a ocorrência de defeitos decorrentes do escoamento desequilibrado entre o tarugo extrudado e a matriz.

Mikos et al. (2011a) desenvolveram um sistema distribuído para a determinação rápida de causas de não-conformidades e respectivas soluções para a moldagem por injeção de termoplásticos utilizando o método de raciocínio baseado em casos. Eles concluíram por meio de experimentos que o sistema pode fornecer suporte de maneira eficaz diminuindo assim a quantidade de defeitos.

Mikos et al. (2011b) desenvolveram um sistema de distribuição e compartilhamento de conhecimento e reutilização no domínio PFMEA usando a abordagem de recuperação de conhecimento baseado em ontologias. Um protótipo foi implementado com base no *Java Agent DEvelopment Framework* (JADE). Na arquitetura proposta, as diferentes bases de conhecimento podem ser distribuídas através da Intranet/Internet.

Tendo em vista as publicações encontradas sobre a aplicação de sistemas computacionais para apoio à manufatura, e mais especificamente voltado ao processo de extrusão, pode-se afirmar que esta proposta de trabalho, além de inovadora, é viável e de promissora aceitação na indústria de manufatura, pois, além de proporcionar suporte à tomada de decisões no chão de fábrica contendo equipamentos para a extrusão de alumínio, os problemas podem ser solucionados rapidamente, utilizando duas maneiras de persistência de informações: banco de dados e ontologias.

2.8 Conclusões do capítulo

Este capítulo apresentou a revisão da literatura relevante para esta pesquisa. Neste sentido, este capítulo foi dividido em sete subseções principais: evolução e desafios conceituais a gestão da qualidade, desafios à qualidade em processos de extrusão de alumínio direta, tecnologia de agentes usados em aplicações na área industrial e de gestão do conhecimento, técnica de inteligência artificial raciocínio baseado em casos e o uso de ontologias como suporte à gestão do conhecimento.

Incluiu-se neste capítulo a evolução e os desafios conceituais a gestão da qualidade, bem como a qualidade em processos de extrusão de alumínio direta no contexto do compartilhamento de conhecimento, buscando-se ressaltar a importância e a estrutura deste compartilhamento para a análise e solução de problemas de não-conformidades em processos de manufatura em estruturas organizacionais.

Por sua vez, a tecnologia de agentes, técnicas de raciocínio baseado em casos e ontologias foram apresentadas considerando os seus potenciais para o suporte eficaz no que diz respeito ao compartilhamento de conhecimento decorrente dos processos de análise e solução dos problemas de não-conformidades.

3 RECURSOS DE SOFTWARE PARA A IMPLEMENTAÇÃO DO SISTEMA PROPOSTO

O propósito deste capítulo é apresentar as questões relacionadas ao processo de escolha dos recursos de software destinados à implementação do modelo baseado em agentes, considerando as especificações de projeto do modelo estabelecidas no Capítulo 4.

Em termos metodológicos, a escolha do recurso de software destinado à implementação do modelo baseado em agentes representa, inegavelmente, um aspecto fundamental da implementação, pois a escolha de um recurso apropriado, em essência, implica em reduzir de forma significativa não somente os esforços de programação como também a abrangência dos componentes de software que necessitaram de verificação.

Neste sentido, o processo de escolha concentra-se em recursos de software que incorporem modelos de programação de agentes e ambientes de execução pré-implementados, verificados e com ampla disseminação na comunidade acadêmica e industrial.

A mesma linha de pensamento é válida em relação aos recursos de software destinados à implementação dos serviços dos agentes de recursos de conhecimento. Em particular, para as tarefas e métodos de raciocínio baseado em casos (RBC) e em relação ao sistema de raciocínio e recuperação de conhecimento, este último responsável por acessar e manipular as bases de conhecimento na forma de ontologias. Deve-se destacar que ambos os recursos devem ser integrados ao modelo de tarefas dos agentes previstos na especificação de projeto do modelo em questão.

Adicionalmente, cumpre ressaltar a necessidade de um ambiente gráfico para a edição e manutenção da estrutura da base de conhecimento ontológica com suporte para a linguagem OWL-DL (*Web Ontology Language – Description Logic*) que será adotada para a codificação desta base. Este ambiente, em especial, deve permitir a integração com sistemas de raciocínio externos, cuja função é verificar a consistência da ontologia em relação a possíveis erros ou contradições lógicas ainda nas fases de modelagem.

Dentro desta perspectiva, são apresentados a seguir o processo e os critérios adotados na escolha destes recursos, bem como um exame das características fundamentais destes recursos, visando destacar, em especial, aquelas que os tornam adequados à implementação do modelo.

Recursos de software compreendem os pacotes de software, bem como as licenças.

3.1 RECURSOS DE SOFTWARE PARA A TECNOLOGIA MULTIAGENTES

Os recursos de software para o desenvolvimento de sistemas baseados em agentes podem ser divididos em três categorias de acordo com Rainer et al. (2005): as plataformas de agentes, os ambientes de desenvolvimento e os arcabouços.

Uma plataforma de agentes é projetada como um conjunto de componentes de software do tipo *middleware*, os quais dão suporte ao desenvolvimento de aplicações multiagentes. Neste contexto, uma plataforma de agentes fornece ainda um ambiente de execução no qual um conjunto de agentes pode, ativamente, existir e cooperar buscando atingir os seus objetivos de projeto. Neste sentido, tal plataforma provê todos os serviços básicos necessários para esta finalidade, como ilustra a Figura 3.1, para plataformas em conformidade às especificações FIPA 2000 - *Foundation for Intelligent Physical Agents* (FIPA, 2002a), atualmente responsável pela disseminação da tecnologia de agentes e interoperabilidade de seus padrões com outras tecnologias.

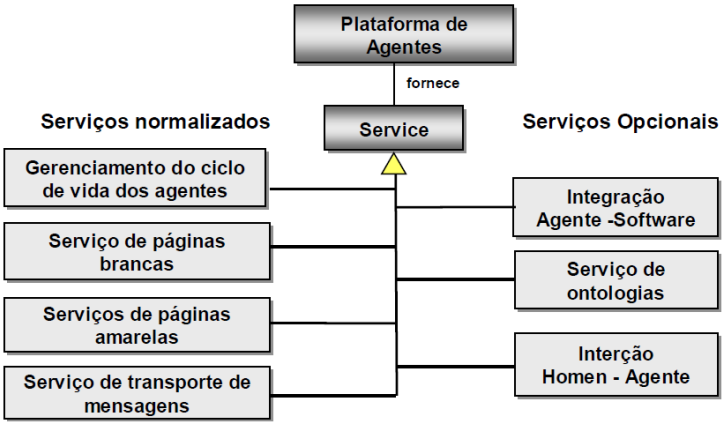

Figura 3-1 - Modelo de serviços fornecidos por uma plataforma de agentes FIPA 2000.
Fonte: FIPA (2000).

Entretanto, é importante ressaltar que, de acordo com as especificações FIPA (2000), o conceito de serviço é definido em termos de um conjunto de mecanismos funcionais.

São componentes de software que visam permitir a interoperação entre aplicações através da rede, apesar das diferenças entre os protocolos de comunicação, arquiteturas de sistemas, sistemas operacionais, bancos de dados e outros serviços disponíveis (RYMER, 1996). São utilizados para prover o ambiente de execução dos agentes e que podem servir como base para a interoperação de sistemas. Portanto, a implementação concreta destes serviços é obrigatória para as plataformas declaradas em conformidade a estas especificações.

Do ponto de vista conceitual, os serviços normalizados envolvem: (a) o serviço de gerenciamento do ciclo de vida dos agentes, que responde pela criação, remoção de agentes e a migração destes entre plataformas; (b) o serviço de páginas brancas, que permite a um agente encontrar outros agentes capazes de prover um dado serviço; (c) o serviço de páginas amarelas, que representa a lista de serviços oferecidos por um determinado agente; (d) o serviço de transporte de mensagens, que permite a interação entre agentes por meio da entrega de mensagens que são trocadas de forma assíncrona entre os agentes dentro da mesma plataforma ou em plataformas distintas.

Outros serviços opcionais fazem parte do modelo, como o serviço de ontologias, que compreende um vocabulário de símbolos aos quais pode-se associar um significado. Este modelo, em particular, refere-se aos objetos e relações entre estes objetos em um dado domínio de aplicação, podendo ser compartilhados por uma comunidade de agentes (FIPA, 2002a).

Por outro lado, um ambiente de desenvolvimento, em geral, visa dar suporte a todas as fases da engenharia de sistemas baseados em agentes, compreendendo: o levantamento de requisitos de engenharia, projeto do sistema, desenvolvimento e sua implementação. Em especial, um arcabouço para sistemas baseados em agentes fornece um modelo de programação de alto nível que compreende uma estrutura básica de componentes de software genéricos, os quais podem ser especializados ou estendidos para criar novas aplicações (RAINER et al., 2005).

3.1.1 Processo de escolha do recurso de software para a tecnologia multiagentes

Neste trabalho de tese, a escolha do recurso de software para o desenvolvimento do modelo deve considerar os recursos, cuja plataforma de agentes esteja em conformidade às especificações FIPA 2000 - *Foundation for Intelligent Physical Agents* (FIPA, 2002a).

Neste cenário, o processo de escolha considerou o trabalho de Leszczyna (2004) como ponto de partida. Nesse trabalho o autor empreendeu uma avaliação rigorosa dos recursos de software voltados a tecnologias multivalentes, cujas plataformas estão em conformidade às especificações FIPA 2000 (FIPA, 2002a), a saber: ADK - Agent *Development Kit* (TRYLLIAN, 2000), *April Agent Platform* (DALE e KNOTTENBELT, 2002), FIPA-OS (EMORPHIA, 2002), *Grasshopper* (BÄUMER et al., 1999), JACK - *Development Environment* JDE (AOS, 2006), JADE - *Java Agent DEvelopment Framework* (BELLIFEMINE et al., 2012), JAS – *Java Agent Services API* (JAVA AGENT SERVICES, 2002) e Zeus (NWANA et al., 1999).

Leszczyna (2004) propõe uma série de questões que devem ser respondidas em relação aos recursos avaliados. Estas perguntas, na verdade, buscam identificar se estes recursos são mantidos, se continuam a ser desenvolvidos pelo grupo de pesquisa original, e se este grupo continua ativo cientificamente, bem como procura revelar o grau de utilização do recurso pela comunidade científica da área.

Nesta mesma perspectiva, Oliveira (2003) apresenta uma comparação dos recursos JADE (BELLIFEMINE et al., 2012), Satélite (JEON et al., 2000), *Infosleuth* (NODINE et al., 2000), *Retsina* (SYCARA et al., 2001), *IBMAglets* (IBM, 2002), OAA - *Open Agent Architecture* (MARTIN et al., 1999), JACK - *Development Environment JDE* (AOS, 2006), FIPA-OS (EMORPHIA, 2002), Zeus (NWANA et al., 1999) e *AgentBuilder Lite* (AGENTBUILDER LITE, 2004). Contudo, Oliveira (2003) concentra-se em critérios de avaliação mais concretos e relacionados especialmente ao modelo de programação e ao ambiente de execução empregado nas plataformas, tais como: a capacidade de integração da plataforma com outros paradigmas de programação, as ferramentas de monitoramento e *debbugging* disponíveis, a qualidade da documentação, o suporte técnico disponível, a qualidade da interface homem/máquina, a curva de aprendizagem, bem como o número de usuários na comunidade.

Adicionalmente, Rainer et al. (2005) apresentam uma discussão sobre as ferramentas JADE (BELLIFEMINE et al., 2012), eXAT (STEFANO e SANTORO, 2003) e A-Globe (ŠIŠLÁK et al., 2005), bem como suas aplicações pela comunidade acadêmica e industrial.

Nesses trabalhos apresentados na literatura relevante da área, destacam-se claramente os recursos JADE (BELLIFEMINE et al., 2012) e JACK (AOS, 2006) em relação aos diversos critérios e perspectivas analisados. Todavia, o JACK é um recurso de software comercial

desenvolvido pela empresa Agent Oriented Software Group (2006) e, deste modo, não atende ao escopo definido no Capítulo 1.

Portanto, o recurso JADE, atualmente na versão 3.4.1 (BELLIFEMINE et al., 2012), é a escolha mais adequada à implementação do modelo proposto neste trabalho de tese, pois, o arcabouço JADE é distribuído e recebe suporte técnico do TILAB (Telecom Itália LABoratori) como software livre sob os termos LGPL (*Lesser General Public License*[6]).

Além disso, a partir do ano de 2003 o gerenciamento do projeto JADE passou a incluir também as seguintes empresas: Motorola, Whitestein Technologies AG, Profactor GmbH e France Telecom R&D.

3.1.2 Características do arcabouço JADE essenciais à implementação do modelo

O recurso de software adotado para a implementação da tecnologia de multivalentes, denominado JADE[7] (*Java Agent DEvelopment Framework*), é um arcabouço de software desenvolvido totalmente em linguagem Java (SUN DEVELOPER NETWORK, 2012), cuja plataforma de agentes está em conformidade às especificações da FIPA 2000 - Foundation for Intelligent Physical Agents (FIPA, 2002a)

Atualmente, a linguagem Java (SUN DEVELOPER NETWORK, 2012), desenvolvida pela Sun Microsystems, vem sendo usada amplamente em aplicações do tipo *middleware*, devido às suas características singulares, tais como: sua portabilidade, isto é, independência em relação ao sistema operacional; capacidade de processamento múltiplo (*multithreading*); e capacidade de operar em ambientes distribuídos, heterogêneos e em rede (DEITEL, 2010; HORSTMANN e CORNELL, 2001).

De acordo com Bellifemine et al. (2012), o principal objetivo do arcabouço JADE é facilitar e simplificar o desenvolvimento de sistemas multivalentes assegurando um padrão de interoperabilidade com outros sistemas multivalentes. Neste sentido, o recurso JADE contém funções pré-implementadas que compreendem os aspectos de um sistema multivalente que são independentes de um tipo particular de aplicação, e não envolvem diretamente as particularidades internas dos agentes.

---

[6] http://www.opensource.org/licenses/lgpl-license.php

[7] O arcabouço JADE foi desenvolvido em cooperação pelo CSELT (*Centro de Studi e Laboratori Telecomunicazioni*), do grupo Telecom Itália e pelo Grupo de Engenharia da Computação da Universidade de Parma no ano de 2001 (BELLIFEMINE et al., 2012).

A plataforma de agentes implementada no arcabouço JADE foi desenvolvida em conformidade à arquitetura de referência para plataformas de agentes estabelecidas nas especificações FIPA 2000 (FIPA, 2002a), como mostrado na Figura 3.2.

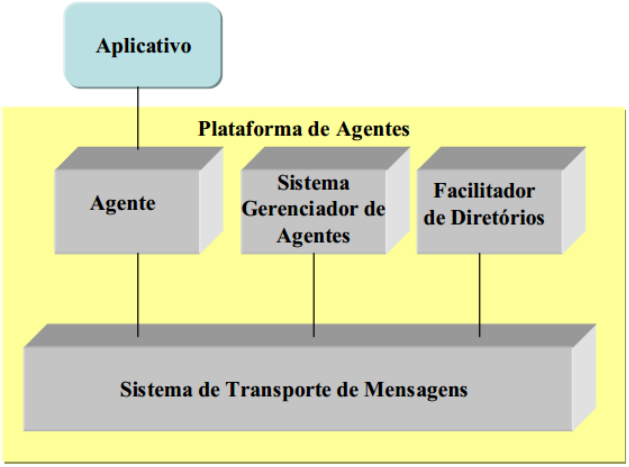

Figura 3-2 - Arquitetura de referência da plataforma de agentes FIPA 2000.
Fonte: BELLIFEMINE et al. 2012.

Esta arquitetura de referência compreende o Sistema Gerenciador de Agentes ou (*AMS - Agent Management System*) que, no arcabouço JADE, foi implementado concretamente como o agente responsável pelo gerenciamento da operação da plataforma de agentes, e suas funcionalidades principais envolvem: criar, excluir, gerenciar o ciclo de vida dos agentes.

Adicionalmente, este agente pode dar suporte à migração de agentes para e a partir de outras plataformas, denominados agentes móveis. O agente AMS mantém, ainda, um índice físico (*directory of agent identifiers* - AID) de todos os agentes residentes na plataforma, que é constituído por um nome globalmente único que obedece à estrutura <*localname*>@<*hostname*>:<*port number of the JADE RMI registry*>/JADE.

Por sua vez, o facilitador de diretório (DF- *Directory Facilitator*) no arcabouço foi implementado como o agente que fornece o serviço de "páginas amarelas" para os agentes residentes na plataforma, pelo qual um agente encontra outro que é capaz de fornecer um dado serviço requerido.

O sistema de transporte de mensagens (MTS - *Message Transport System*), disponibilizado pelo Canal de Comunicação de Agentes (ACC - *Agent Communication Channel*), é o componente de software que controla toda a troca de mensagens dentro da plataforma, bem como para e a partir de outras plataformas. Esses componentes são automaticamente iniciados quando a plataforma de agentes é executada em um *host*.

Neste cenário, uma das principais características da plataforma de agentes do arcabouço JADE que a tornam particularmente adequada à implementação do modelo proposto neste trabalho de tese, reside no fato de que seu ambiente de execução de agentes pode estar distribuído entre vários servidores (*hosts*). Portanto, esses *hosts* podem estar geograficamente separados, e cada um deles executando apenas uma máquina virtual Java (JVM - *Java Virtual Machine*).

Do ponto de vista da computação, no arcabouço JADE os agentes são implementados como *threads*[8] da linguagem Java, contidos dentro de uma instância de execução (*runtime instance*) do ambiente de execução JADE (*runtime environment*). Esta instância é denominada, na terminologia JADE, de *container*, o qual pode conter vários agentes fornecendo todos os serviços de suporte necessário para a execução destes agentes.

Por consequência, uma plataforma na terminologia JADE é composta por um *container* principal (ou *main container*) que deve estar sempre ativo, e de todos os outros *containers* (ou *non-main*) que se registram nesta plataforma ao se tornarem ativos em outros servidores distribuídos (*hosts*). Deste modo, uma plataforma integra todos os hosts envolvidos atuando como um verdadeiro elo de ligação que provê um completo ambiente de execução para o conjunto de agentes.

A Figura 3.3 apresenta a estrutura da plataforma de agentes do arcabouço JADE, distribuída entre vários servidores. Nesta Figura, o *container* principal (*Jade Main Container*) está localizado no *host* 1 que representa o *container* no qual se encontram o sistema gerenciador de agentes (AMS - *Agent Management System*), o agente facilitador de diretório (DF- *Directory Facilitator*) e o Registro RMI (*Remote Method Invocation Registry*).

[8] Um *thread* (linha de execução) é um fluxo único de controle sequencial dentro de um programa escrito em linguagem Java (SUN, 2012)

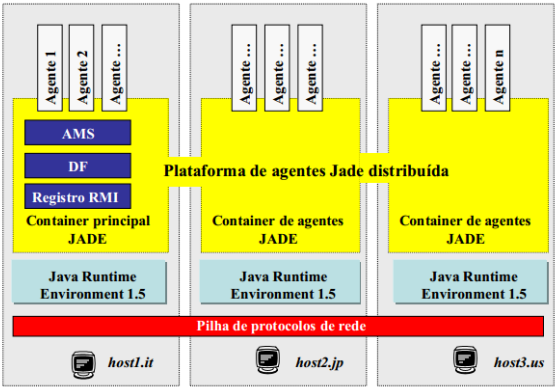

Figura 3-3 - Plataforma de agentes JADE distribuída entre vários *containers*.
Fonte: BELLIFEMINE et al. (2012).

Esse registro RMI é um servidor de nomes que a linguagem Java utiliza para registrar e recuperar referências a objetos por meio do seu nome. Isto é, o registro RMI é o meio que o JADE usa em Java para manter as referências aos outros *containers* de agentes que se conectam à plataforma.

Portanto, a plataforma JADE, em essência, representa uma abstração significativa para a complexidade e a diversidade de características envolvidas em uma aplicação desta natureza para seus desenvolvedores, tais como diferenças entre hardwares, sistemas operacionais, tipos de redes ou máquinas virtuais Java, bem como da separação física dos *hosts*.

Em suma, a plataforma JADE inclui: (a) um ambiente de execução (*runtime environment*) onde os agentes de software podem ser executados, o qual deve estar ativo em um dado servidor (*host*) antes que um ou mais agentes possam ser executados neste servidor; (b) uma ferramenta de interface gráfica que permite administrar e monitorar as atividades de execução dos agentes, (c) uma biblioteca de classes escritas em linguagem Java permitindo aos desenvolvedores usá-las diretamente ou especializá-las para desenvolver seus próprios agentes.

3.1.3 Forma de implementação do modelo de tarefas dos agentes no arcabouço JADE

A abstração do modelo de tarefas dos agentes no arcabouço JADE, denominada "comportamento"[9], é outra característica fundamental da plataforma de agentes em JADE, que a torna, também, adequada para a implementação do modelo formal proposto neste trabalho de tese.

Esta abstração baseada na noção de comportamentos busca, inicialmente, modelar arquiteturas internas reativas para os agentes. Além disso, ela possibilita a integração com outros softwares externos como, por exemplo, integrar no comportamento de um agente JADE outros sistemas computacionais capazes de acessar e processar bases de conhecimento específicas de maneira a potencializar a arquitetura interna destes agentes (FIGUEIRA e RAMALHO, 2000).

A Figura 3.4 apresenta a arquitetura interna genérica de um agente no arcabouço proposto. Nesta figura pode-se observar as várias tarefas do agente representadas por meio de uma coleção de comportamentos, cuja sequência de execução é ordenada por um escalonador interno da classe *Agent* (BELLIFEMINE et al., 2012). Este escalonador gerencia, de forma automática e transparente ao desenvolvedor, a sequência de execução dos comportamentos mediante um algoritmo denominado circular não-preemptivo (do termo em inglês *Round-Robin*).

Este algoritmo assegura que um comportamento é executado por vez e por um determinado tempo. O fato de ser não-preemptivo significa que não existe prioridade entre estes comportamentos, os quais são executados automaticamente na ordem em que foram posicionados pelo desenvolvedor na fila de comportamentos a serem executados (BELLIFEMINE et al., 2012).

A Figura 3.4 mostra também que o agente possui uma fila privativa de mensagens escritas na Linguagem de Comunicações de Agentes sugerida pela FIPA (2002b) FIPA-ACL (FIPA – *Agent Communication Linguagem*), que pode decidir quando e quais mensagens recebidas serão lidas.

[9] do termo em inglês *Behaviour*

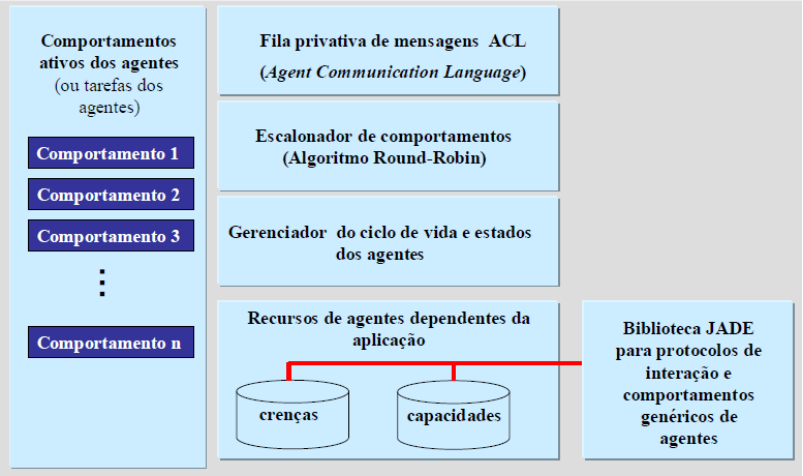

Figura 3-4 - Arquitetura interna genérica de um agente no arcabouço JADE
Fonte: BELLIFEMINE et al., (2012).

Em especial, a arquitetura interna do agente dispõe de um gerenciador de ciclo de vida próprio que possibilita ao agente a propriedade de autonomia, pois mediante este gerenciador o agente pode controlar completamente o seu *thread* de execução. Um agente genérico em JADE é dotado, ainda, de estruturas que são dependentes de cada aplicação em particular que permitem armazenar recursos como crenças (*beliefs*) e capacidades.

Neste modelo de programação, um comportamento (*behaviour*) é uma classe abstrata do arcabouço que pode ser especializada para modelar uma tarefa específica. Para isso, o arcabouço possui um conjunto de classes de comportamentos pré-implementadas para uso pelo desenvolvedor (BELLIFEMINE et al., 2012).

É importante ressaltar que os dois métodos principais da classe Behaviour são *action()* e *done()*. O método *action()* deve ser implementado pelo desenvolvedor para realizar as tarefas ou ações projetadas para o agente em questão por meio deste comportamento, enquanto o método *done()* é usado pelo escalonador de comportamentos da classe *Agent* para saber se o comportamento foi executado e finalizado ou não.

Um agente deve ser capaz de executar várias tarefas de maneira concorrente em resposta a eventos externos e, para tornar o gerenciamento eficiente, cada agente no arcabouço JADE é implementado como um

thread de execução, enquanto todas as suas tarefas modeladas podem ser implementadas como objetos da classe *Behaviour*.

Portanto, o desenvolvedor que desejar implementar um modelo de tarefa específico para o agente deve definir uma ou mais subclasses *Behaviour*, instanciá-las e adicionar os objetos desta classe na lista de comportamentos do agente. Neste sentido, a classe *Agent* inclui dois métodos: *addBehaviour* e *removeBehaviour*, os quais permitem gerenciar a fila de tarefas de um agente específico (BELLIFEMINE et al., 2012).

3.1.4 Modelo de comunicação entre agentes

A comunicação entre agentes tem um papel fundamental em uma organização multiagente, pois cada agente é projetado para desempenhar um conjunto de tarefas específicas buscando cooperar com outros agentes com a finalidade de atingir os objetivos globais do projeto. Os sistemas baseados em agentes, usualmente, adotam uma arquitetura de comunicação ponto a ponto (do inglês *peer-to-peer*), onde cada agente representa um processo computacional sendo executado em um nó de uma rede. Deste modo, os agentes devem ser capazes de comunicar-se com outros agentes visando fornecer de maneira adequada seus serviços aos demais agentes residentes em uma ou mais plataformas. Neste contexto, a comunicação entre agentes ocorre através do envio de mensagens individuais de um agente para outro mediante a passagem assíncrona de mensagens.

O modelo de referência para o transporte de mensagens entre agentes implementado no arcabouço JADE segue as especificações da FIPA (2002b, 2002c) e compreende dois níveis:

- O nível do Protocolo de Transporte de Mensagens (MTP – *Message Transport Protocol*), usado para executar a transferência física de mensagens entre agentes;
- O Serviço de Transporte de Mensagens (MTS – *Message Transport Service*) para todos os agentes registrados, cuja função é dar suporte computacional para o transporte das mensagens escritas de acordo com a Linguagem de Comunicação de Agentes (FIPA ACL – *Agent Communication Language*), observando-se que este transporte pode ocorrer entre agentes em uma mesma plataforma ou entre agentes de diferentes plataformas (FIPA, 2002b).

De uma forma abstrata, uma mensagem é composta de duas partes: o envelope da mensagem (*message envelope*) que contém as informações

necessárias ao serviço de transporte, e a própria mensagem (FIPA, 2002c).

A Figura 3.5 apresenta o modelo de comunicação instanciado no arcabouço JADE, que executa, de forma transparente ao desenvolvedor, as trocas de mensagens entre os agentes.

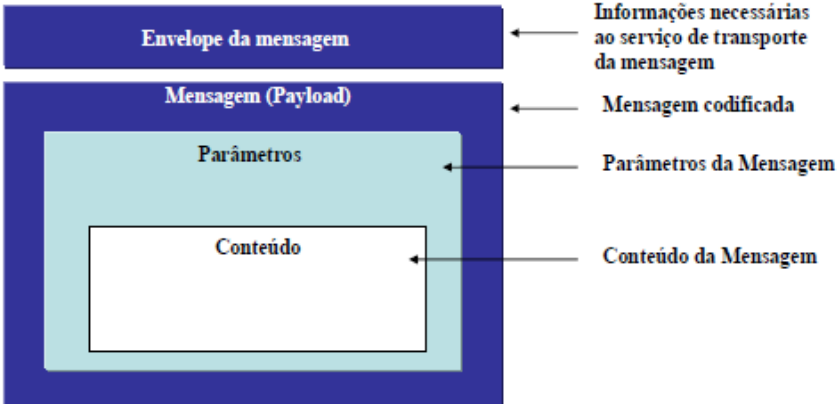

Figura 3-5 - Modelo de comunicação de acordo com as especificações FIPA-ACL
Fonte: FIPA (2002b)

3.1.5 Síntese das características essenciais do arcabouço JADE

Em síntese, o arcabouço JADE atende às especificações da FIPA (2002a, 2002b, 2002c) e oferece ao desenvolvedor um elenco de funcionalidades capazes de reduzir os esforços de programação e verificação. Além disso, ele permite ao desenvolvedor estender as classes básicas do arcabouço já verificadas, e concentrar-se unicamente na implementação do modelo de tarefas projetado para a sociedade de agentes em questão.

Dentre as principais características do arcabouço JADE relacionadas à implementação do modelo formal proposto neste trabalho destacam-se:

- Uma plataforma de agentes distribuída que permite, entre outros, que agentes possam ser executados em diferentes servidores (*hosts*), os quais podem estar separados geograficamente, embora conectados por meio da Chamada de Métodos Remotos (RMI -

Remote Method Invocation[10]) através de uma rede. Esta funcionalidade permite a implementação do protótipo de laboratório do modelo formal usando-se a abstração de *containers* como *hosts* distribuídos geograficamente.

- A abstração do modelo de tarefas dos agentes no arcabouço JADE denominada como comportamento (*behaviour*), que apresenta, entre outras vantagens, a possibilidade de integração com outros recursos de software como, por exemplo, sistemas de raciocínio capazes de acessar e processar bases de conhecimento específicas de maneira a potencializar a arquitetura interna destes agentes.
- O arcabouço JADE dispõe de métodos especiais na classe *ACLMessage* que permitem a transmissão de mensagens, cujo conteúdo pode ser um conjunto de caracteres (*strings*) e, em especial, objetos Java do tipo *java.io.Serializable*.

3.2 RECURSO DE SOFTWARE PARA MÉTODOS DE RBC

A literatura revela diferentes tipos de ferramentas voltadas ao desenvolvimento de aplicações de raciocínio baseado em casos (RBC), que podem ser classificadas em três grupos de acordo com Jaczynski e Trousse (1998): ferramentas denominadas de CBR *shells*[11], Interfaces de Programas Aplicativos RBC[12], e arcabouços (*frameworks*) para RBC.

As ferramentas denominadas CBR *shells* permitem a um usuário, em geral não programador, a geração de aplicações de raciocínio baseado em casos por meio de uma interface gráfica sofisticada. Nesta interface, os parâmetros necessários ao desenvolvimento da aplicação podem ser definidos de modo interativo pelo usuário como, por exemplo, especificar os descritores dos casos relativos ao domínio de conhecimento, bem como o vetor de pesos usados para o cálculo da medida de similaridade usada para recuperar casos. Entretanto, usualmente, estas ferramentas não permitem modificações ou integração de novos componentes de software (JACZYNSKI E TROUSSE, 1998).

As Interfaces de Programas Aplicativos RBC fornecem um conjunto de funções para gerenciar os algoritmos de RBC, e são projetadas para serem usadas por programadores permitindo uma customização limitada mediante a respectiva linguagem de programação,

[10] O mecanismo RMI ou Chamada de Métodos Remotos permite o acesso a objetos em máquinas diferentes (HORSTMANN e CORNELL, 2001).
[11] Programa que controla a interação do usuário da ferramenta com o núcleo (*kernel*) do sistema CBR.
[12] Do termo em inglês CBR API's - *Application Programming Interfaces*.

visando adicionar, por exemplo, novas medidas de similaridade ou técnicas de adaptação. Entretanto, o propósito destas interfaces não é oferecer componentes de software genéricos ou de código aberto, mas somente customizar as entradas e saídas do sistema (JACZYNSKI e TROUSSE, 1998).

Por sua vez, um arcabouço para métodos de raciocínio baseado em casos, segundo Fayad et al. (1999), consiste de uma aplicação de software incompleta e reutilizável que pode ser especializada para produzir aplicações customizadas.

Dentro desta perspectiva, apresenta-se nas próximas subseções o processo de escolha da ferramenta responsável pela implementação dos métodos de raciocínio baseado em casos, bem como descrever as características principais da ferramenta selecionada para o presente trabalho.

3.2.1 Processo de escolha da ferramenta para implementação de RBC

Considerando a natureza das ferramentas reveladas por Jaczynski e Trousse (1998), bem como as especificações de projeto estabelecidas, a escolha deve concentrar-se, prioritariamente, nos arcabouços para raciocínio baseado em casos. Neste sentido, o arcabouço em questão deve ser implementado em linguagem Java, de modo a manter-se compatível com a ferramenta previamente adotada para a implementação da tecnologia de agentes.

Neste contexto, na literatura foram identificados dois arcabouços que atendem a estes requisitos: (a) o arcabouço IUCBRF – Indiana *University Case-Based Reasoning Framework* (BOGAERTS e LEAKE, 2005) desenvolvido pelo Departamento de Ciência da Computação da Universidade de Indiana e licenciado através da Licença de Software Aberto[13]; (b) o arcabouço jCOLIBRI (BELLO-TOMÁS et al., 2004) desenvolvido pelo Grupo de Aplicações de Inteligência Artificial da Universidade Complutense de Madri (Espanha) e licenciado através da Licença Pública Geral[14].

Dentre os critérios adotados no processo de escolha do arcabouço para a implementação das tarefas e métodos de RBC destacam-se: a

[13] Do termo em inglês *Open* Software *License* regulamentado em http://www.opensource.org/licenses/osl- 3.0.php

[14] Do termo em inglês GNU *Lesser General Public License* http://www.opensource.org/licenses/lgpl-license.php

manutenção e novos desenvolvimentos do arcabouço pelo grupo de pesquisa, a linguagem e modelo de programação, qualidade da documentação e tutoriais, suporte técnico e curva de aprendizagem necessária à instanciação concreta do arcabouço.

Os arcabouços em questão adotam a abordagem de orientação a objetos e independência do domínio, e permitem, além da reutilização de código, a extensão de classes abstratas disponíveis. Adicionalmente, estes arcabouços dispõem de suporte técnico adequado e são mantidos atualizados pelo grupo de pesquisa original.

Embora, de acordo com estes critérios, ambos os arcabouços possam ser considerados viáveis para a implementação do modelo formal proposto neste trabalho de tese, o arcabouço jCOLIBRI foi escolhido devido aos seguintes aspectos: qualidade dos tutoriais e a curva de aprendizagem necessária à instanciação concreta do arcabouço, tendo em vista ainda que:

- O arcabouço jCOLIBRI foi construído a partir de um modelo explícito de tarefas RBC (CBR *Tasks*): recuperação, reutilização, revisão e retenção de casos, como descritas por Aamodt e Plaza (1994), e inclui também métodos RBC voltados à decomposição das tarefas RBC em subtarefas e métodos de resolução das tarefas e subtarefas. Este modelo constitui uma arquitetura de alto nível para o arcabouço cujo objetivo é tornar o processo de instanciação concreta do arcabouço mais consistente e simples.
- O jCOLIBRI dispõe, adicionalmente, de um conjunto de ferramentas e interfaces gráficas para a configuração de uma aplicação a partir do modelo de tarefas e métodos RBC. Estas ferramentas têm o propósito de guiar o processo de instanciação do arcabouço através destas interfaces, reduzindo significativamente os esforços de instanciação e o tempo de aprendizagem necessário para gerar novas aplicações.

3.2.2 Características do arcabouço jCOLIBRI essenciais à implementação do modelo

O arcabouço jCOLIBRI é uma evolução do sistema denominado COLIBRI desenvolvido originalmente por Díaz-Agudo (2002) e implementado em linguagem LISP, projetado originalmente para dar suporte ao desenvolvimento de aplicações de RBC. A principal contribuição do trabalho de Díaz-Agudo (2002) é a abordagem de reutilização, não somente dos componentes de software, mas também do

conhecimento relacionado ao próprio domínio de raciocínio baseado em casos a partir de uma ontologia denominada CBROnto, a qual representa os conceitos e relações neste domínio.

A ideia que fundamenta a ontologia denominada CBROnto é estabelecer uma linguagem comum para definir os elementos que compõem um sistema de RBC e permitir a construção genérica de métodos de raciocínio baseado em casos.

Nesta perspectiva, no jCOLIBRI a ontologia RBC não é representada somente como um recurso de conhecimento isolado, mas estabelece uma forma de correspondência direta que permite que os conceitos relacionados ao domínio de RBC possam ser mapeados ou relacionados diretamente às classes abstratas escritas em linguagem Java ou interfaces gráficas disponíveis no arcabouço (RECIO-GARCÍA et al., 2005 e RECIO-GARCÍA et al., 2014). Dentre os conceitos relacionados ao domínio de RBC tem-se: Casos, Base de Casos, Descrição do Caso, Solução do Caso, Função de Similaridade Global, Função de Similaridade Local, Busca (*Query*), Tarefas RBC e Métodos RBC.

Adicionalmente, a ontologia CBROnto permite compartilhar os diversos métodos de RBC para solução de problemas (PSMs – *Problem Solving Methods*), os quais representam as estratégias genéricas e reutilizáveis voltadas para a resolução das tarefas de RBC necessárias para atingir os objetivos da aplicação (RECIO-GARCÍA et al., 2005 e RECIO-GARCÍA et al., 2014). Cumpre observar que estas estratégias ou Métodos para Solução de Problemas podem ser selecionados e configurados a partir de uma biblioteca de métodos reutilizáveis disponível no arcabouço jCOLIBRI (PSML – *Problem Solving Methods Library*).

De acordo com Aamodt e Plaza (1994), o ciclo principal de raciocínio pode ser decomposto em quatro tarefas de RBC, a saber: recuperar os casos mais similares (*retrieve*), reutilizá-los para resolver o problema em questão (*reuse*), revisar a solução proposta (*revise*) e aprender com a experiência (*retain*). Dentro desta ótica, o jCOLIBRI foi construído a partir da ideia básica de separar tarefas e métodos de RBC. As tarefas RBC como apresentadas por Aamodt e Plaza (1994) indicam os objetivos que o sistema deve alcançar e, portanto, guiam a execução da aplicação. Por sua vez, os métodos são divididos em dois tipos: os métodos de decomposição, responsáveis pela decomposição de uma tarefa particular em subtarefas, e os métodos de resolução adotados diretamente na resolução de uma tarefa.

Neste sentido, o arcabouço jCOLIBRI modela o ciclo RBC por meio da tarefa RBC (*CBR Task*) e propõe um Método RBC associado

(*CBR Method*) que a decompõe em quatro subtarefas: *CBR Retrieve Task*, *CBR Reuse Task*, *CBR Revise Task*, e *CBR Retain Task* e suas respectivas subtarefas, como mostra a Figura 3.6.

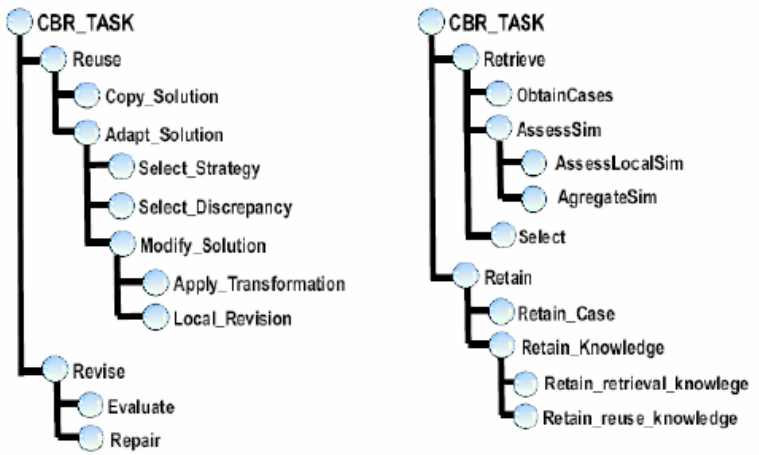

Figura 3-6 - Estrutura de tarefas na ontologia CBROnto do arcabouço jCOLIBRI. Fonte: Recio-García (2008).

Portanto, uma aplicação de RBC pode ser representada como uma estrutura em forma de uma árvore, cujos nós representam as tarefas e subtarefas de RBC a serem resolvidas. Deste modo, executar uma aplicação consiste em resolver estas tarefas executando-se os métodos RBC correspondentes por meio da estrutura de classes do arcabouço escritas em linguagem Java que são mapeadas automaticamente pela ontologia *CBROnto*.

A partir desta perspectiva, a Figura 3.7 apresenta a arquitetura geral do arcabouço jCOLIBRI, na qual se percebe que o arcabouço contém, adicionalmente, uma camada de interface com clientes remotos e outras plataformas Java, bem como uma camada para acesso a bases de conhecimento por meio de conectores de software pré-programados.

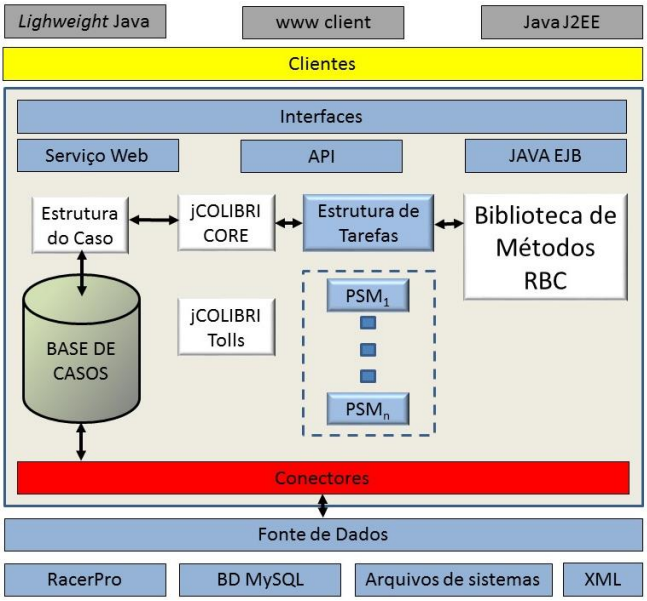

Figura 3-7 - Arquitetura geral do arcabouço jCOLIBRI.
Fonte: Recio-García (2008).

Nesta arquitetura, os conectores representam a primeira camada do arcabouço sobre os meios de persistência física, e são responsáveis por acessar e recuperar casos a partir destes meios físicos, como mostra a Figura 3.8. O uso do conceito de conectores é uma das características essenciais do arcabouço, que permite ao desenvolvedor uma grande flexibilidade em relação à escolha do meio de persistência.

Na versão jCOLIBRI 1.1.0 (07/07/2006) estão disponíveis quatro diferentes tipos de conectores: conectores para arquivos em formato texto (*plain text*), conectores para arquivos de sistema armazenados no formato XML; conectores JDBC (*Java Database Connectivity*) que torna possível o emprego do jCOLIBRI com a maioria de bancos de dados disponíveis no mercado, entre eles o MySQL (MYSQL, 2012); conector RACER que permite o acesso a bases de caso representadas com o formalismo de lógica de descrições e implementadas no sistema RACER (RACER SYSTEM, 2012).

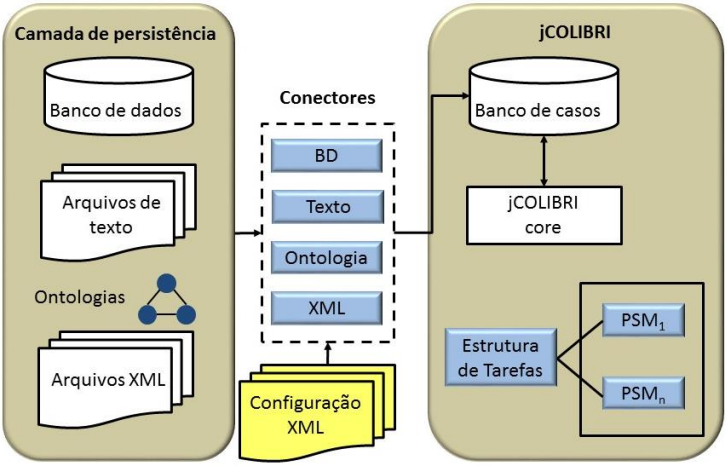

Figura 3-8 - Arquitetura de conectores disponíveis no arcabouço jCOLIBRI.
Fonte: Recio-García (2008).

Do ponto de vista do modelo de programação, o arcabouço jCOLIBRI é organizado em termos de pacotes escritos em linguagem Java (*Java Packages)* divididos em três categorias: *jCOLIBRI - Application Core* (Núcleo da Aplicação), *jCOLIBRI – API Applications Programming Interfaces* (Interfaces de Programas Aplicativos) e *jCOLIBRI - Development Packages* (Pacotes de desenvolvimento).

A primeira categoria (Núcleo de Aplicação) contém as classes devotadas à execução das tarefas e métodos referentes ao ciclo principal de RBC, classes estas que podem ser usadas sem a necessidade de modificações e constituem a essência do *jCOLIBRI kernel*. A segunda categoria (Interfaces de Programas Aplicativos) auxilia a implementação das interfaces gráficas com o usuário. A terceira categoria (Pacotes de Desenvolvimento) compreende as classes do arcabouço usadas para criar novas aplicações RBC por meio da instanciação destas classes.

Uma revisão profunda das bases conceituais do arcabouço jCOLIBRI e descrições dos diagramas de classes escritas em linguagem Java, diagramas de sequência e sobre o processo de instanciação estão além do escopo desta subseção. Entretanto, estes assuntos podem ser encontrados nas publicações de Bello-Tomás et al. (2004), Recio-Garcia et al. (2005 e 2014), bem como em tutoriais e em extensa documentação técnica disponibilizada pelo Grupo de Aplicações de Inteligência Artificial da Universidade Complutense de Madri (GAIA, 2012).

3.2.3 Síntese das características essenciais do arcabouço jCOLIBRI

O arcabouço jCOLIBRI foi projetado visando dar suporte a um amplo espectro de tipos de aplicações de raciocínio baseado em casos, a partir da abstração do modelo de tarefas e métodos descritos anteriormente. As características das aplicações de RBC podem ser bastante diversas, e alguns exemplos destas características seguem abaixo:

- Tipos de representação para os casos (pares atributo-valor, textos, representações orientadas a objetos, representações semiestruturadas);
- Estruturas para a organização da base de casos (plana, hierárquica, baseada em aprendizagem de máquina);
- Meios de persistência física para a base de casos (MySQL, RACER, XML)
- Combinações de tarefas e métodos RBC que podem ser selecionados de maneira flexível, permitindo a customização das aplicações (aplicações que usam somente as tarefas e subtarefas de recuperação até aplicações completas);

No tocante ao uso de arcabouços, uma das questões mais relevantes diz respeito ao tempo de aprendizagem necessário para saber como usá-los. Neste contexto, o jCOLIBRI dispõe de uma ferramenta para a configuração semiautomática das tarefas e métodos RBC por meio de uma interface gráfica (GUI - *Graphical User Interface*) de modo a guiar o processo de instanciação do arcabouço. A configuração de sistemas de RBC usando esta interface consiste nos seguintes passos:

- Definir a estrutura do caso, a fonte dos casos e a organização da base de casos;
- Enquanto a aplicação estiver em construção, o desenvolvedor deve selecionar sequencialmente as tarefas necessárias para cumprir o objetivo específico da aplicação, e para cada uma das tarefas ou subtarefas ele deve selecionar e configurar os métodos responsáveis pela sua resolução.
- Uma vez que a aplicação esteja completa, pode-se gerar um código completo em linguagem Java para disponibilizar a aplicação. Adicionalmente, pode-se gerar uma classe em linguagem Java que permite a sua execução dentro de outras aplicações.

Estas são, seguramente, as funcionalidades que tornam o arcabouço jCOLIBRI particularmente adequado à implementação do modelo proposto neste trabalho de tese.

3.3 SISTEMA DE RACIOCÍNIO E RECUPERAÇÃO DE CONHECIMENTO

A linguagem OWL-DL prevista para a implementação das bases de conhecimento ontológicas definidas no Capítulo 4, é uma linguagem para ontologias desenvolvida pelo W3C (PATEL-SCHNEIDER et al. 2004). Embora desenvolvida inicialmente com o objetivo de atender os requisitos decorrentes da pesquisa no campo da Web Semântica (HORROCKS et al., 2003), a OWL tornou-se rapidamente de fato uma linguagem padrão para o desenvolvimento de ontologias em geral (GARDINER et al., 2006).

O estabelecimento da linguagem padrão OWL tem induzido o desenvolvimento e a adaptação de uma ampla gama de ferramentas e serviços, incluindo os sistemas de raciocínio e recuperação de conhecimento mediante o uso de linguagem de consultas, bem como os editores de ontologias (GARDINER et al., 2006).

Neste contexto, apresenta-se a seguir o processo de escolha do sistema de raciocínio e de recuperação de conhecimento, além das características essenciais do sistema escolhido.

3.3.1 Processo de escolha dos sistemas de raciocínio e recuperação para bases de conhecimento ontológicas

Atualmente, estão disponíveis diversos sistemas de raciocínio automático para lógica de descrições, os quais dispõem de serviços de raciocínio para ontologias implementadas em linguagem OWL-DL, entre os quais se pode citar: o sistema Pellet (SIRIN e PARSIA, 2004), o sistema FaCT ++ (*Fast Classification of Terminologies*) (TSARKOV e HORROCKS, 2006) e o Sistema RacerPro (*Renamed ABox and Concept Expression Reasoner*) (HAARSLEV e MÖLLER, 2001).

Neste contexto, o processo de escolha fundamentou-se nos trabalhos de Liebig (2006) e Gardiner et al. (2006). Liebig (2006) apresenta uma análise comparativa dos principais sistemas de raciocínio disponíveis, abordando os seguintes aspectos: conformidade em relação à linguagem OWL-DL, tipos serviços de raciocínio disponíveis, verificação da eficiência e funcionamento correto dos algoritmos usados nestes serviços de raciocínio e interfaces para acesso ao sistema.

Neste sentido, Gardiner et al. (2006) buscam testar, em especial: (a) o funcionamento correto do algoritmo de raciocínio mediante a comparação do resultado dos serviços em questão efetuados por um raciocinador com aqueles obtidos por outros raciocinadores; (b) o desempenho dos raciocinadores ao realizar o serviço de classificação da taxonomia.

O sistema Pellet (SIRIN e PARSIA, 2004) é um sistema de raciocínio automático implementado em linguagem Java e baseado no algoritmo tableau (*tableaux algorithms*) (BAADER e SATTLER, 2001), desenvolvido para lógica de descrições expressivas. O sistema Pellet foi originalmente desenvolvido pela Universidade de Maryland no âmbito do projeto Mindswap (*Maryland Information and Network Dynamics Lab Semantic Web Agents Project*) (MINDSWAP, 2003), cujo propósito era desenvolver sistemas de raciocínio para serviços *web* (*Web Service*).

O sistema Pellet suporta os serviços padrão de raciocínio para lógica de descrições OWL-DL, bem como a realização de um subconjunto de consultas conjuntivas não otimizadas (*ABox queries*) de acordo com a sintaxe da linguagem de consultas RQLD[15] (*Query Language for RDF*). Este sistema pode ser acessado através da Interface DIG[16] (*Description Logics Implementation Group*).

Os serviços padrão de raciocínio disponíveis no sistema Pellet compreendem: a verificação da consistência da ontologia, a capacidade de satisfazer os conceitos (satisfatibilidade), a classificação de taxonomias e a realização conceitos. Entretanto, cumpre observar que atualmente o sistema Pellet é uma ferramenta comercial.

Por sua vez, o sistema FaCT++ (TSARKOV E HORROCKS, 2004) é uma nova implementação do sistema FaCT (HORROCKS, 1998) desenvolvido Grupo de Informática Médica da Universidade de Manchester, originalmente, escrito em linguagem Common Lisp.

Atualmente, o sistema FaCT++ usa o mesmo algoritmo otimizado do sistema FaCT (*Fast Classification of Terminologies*), mas com uma arquitetura interna diferente, além de contar com uma nova implementação em linguagem C++. O sistema suporta os serviços padrão de raciocínio e pode ser usado como um componente de software isolado (*standalone*) com acesso através da Interface DIG. Entretanto, até o ano

[15] Linguagem de consulta para RDQL - A *Query Language for RDF* é uma especificação do W3 Consortium (2004) http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/

[16] Interface padronizada no formato XML desenvolvida pelo grupo DL Implementation Group (DIG) e permite a interface com sistemas de Lógica de Descrições (BECHHOFER et al., 2003).

de 2006 o sistema FaCT++ não dispunha de suporte para qualquer tipo de linguagem de consulta para a recuperação de conhecimento.

Por fim, o sistema RacerPro (*Renamed ABox and Concept Expression Reasoner Professional*) é uma nova versão do sistema Racer desenvolvido por Haarslev e Möller (2001). O sistema RacerPro é um sistema de representação de conhecimento que fornece suporte para serviços padrão de raciocínio automático, bem como possibilita a realização de consultas conjuntivas visando a recuperação de conhecimento.

Com relação aos serviços de raciocínio, o sistema RacerPro usa o algoritmo tableau (*tableaux algorithms*[17]) para a linguagem de lógica de descrições ALCQHIR+, também conhecida como SHIQ (BAADER e NUTT, 2003), bem como incorpora todas as técnicas de otimização empregadas pelo sistema FaCT (TSARKOV E HORROCKS, 2004).

Do ponto de vista das consultas conjuntivas, o sistema RacerPro dispõe de uma máquina de inferência própria para o processamento de consultas (*query processing engine*) que adota a linguagem de consultas denominada de nRQL (*new Racer Query Language*) e permite entre outras funcionalidades o processamento de múltiplas consultas de modo assíncrono.

Neste cenário, o sistema RacerPro destaca-se devido à ampla gama de serviços de raciocínio disponíveis, bem como pela expressividade da linguagem de consultas nRQL voltada à recuperação de conhecimento.

É importante destacar que os sistemas Racer e RacerPro continuam sendo desenvolvidos por Ralf Möller e Volker Haarslev da Universidade de Tecnologia de Hamburgo e Universidade de Concórdia do Canadá respectivamente. Entretanto, cumpre observar que o RacerPro é, atualmente, um sistema com suporte comercial distribuído pela *Racer Systems GmbH & Co*. KG (RACER SYSTEM, 2007). Todavia, devido à sua origem acadêmica, a empresa disponibiliza uma licença especial sem custos para aplicações não comerciais e voltadas à pesquisa e desenvolvimento científico.

[17] (BAADER e SATTLER, 2001)

3.3.2 Características do sistema RacerPro essenciais à implementação do modelo

O sistema RacerPro, o qual foi adotado neste trabalho para a implementação das bases de conhecimento ontológicas que devem ser acessadas e processadas pelos agentes de recursos de conhecimento PFMEA, dispõe dos seguintes serviços de raciocínio: (a) para o componente *TBox* (ou definição de classes *OWL-DL*): satisfatibilidade, subclassificação, equivalência e disjunção de conceitos (classes); (b) para o componente *ABox* (instâncias das classes OWL-DL): verificação da consistência de um *ABox* em relação a um dado *TBox* considerando-se possíveis erros ou contradições na modelagem das instâncias e conceitos, verificação de instâncias, retorno e realização.

Neste sentido, é importante destacar que o sistema pode processar bases de conhecimento implementadas na linguagem OWL Lite e OWL DL (PATEL-SCHNEIDER et al. 2004), fornecendo os seguintes serviços adicionais: verificação da consistência de ontologias OWL e descrições RDF, identificação de relações implícitas em subclasses OWL induzidas durante a declaração, identificação de sinônimos de recursos (nomes de classes e instâncias).

Por sua vez, a linguagem de consultas adotada pelo sistema é denominada de nRQL (*new Racer Query Language*), e pode ser usada para consultas em: RacerPro ABoxes e TBox, documentos RDF e OWL. Neste sentido, a abordagem semântica da linguagem permite a construção de consultas sobre classes (*Description Logic - concepts*) e propriedades (*Description Logic - roles*), bem como a construção de consultas complexas especificadas mediante o uso de operadores lógicos em termos de álgebra relacional envolvendo operadores de intersecção, união, negação e projeção.

Esta linguagem permite também que as variáveis de consulta (*query variables*) sejam associadas às instâncias do componente *ABox* (instâncias OWL) que satisfazem a consulta, embora as próprias instâncias do *ABox* possam ser usadas diretamente como variáveis nas consultas. Adicionalmente, consultas complexas para o componente TBox podem ser especificadas para identificar certos padrões de relacionamento entre subclasses e superclasses em uma dada taxonomia de um *RacerPro TBox* ou base de conhecimento implementada em OWL-DL. O RacerPro utiliza o algoritmo de tableaux para inferir e construir a árvore, que é expandida até que a validade seja estabelecida ou atinja um ponto onde não seja possível prosseguir com o processo de expansão.

A gama de serviços de raciocínio e de recuperação de conhecimento disponíveis no sistema RacerPro pode também ser usada como um servidor de serviços que permite o acesso por meio de uma interface de comunicação padrão TCP/IP. Esta funcionalidade permite ao desenvolvedor uma grande flexibilidade e liberdade para a implementação de aplicações específicas baseadas nas Interfaces de Programação de Aplicativos, inclusive para aplicações em linguagem Java.

Estas são, seguramente, as funcionalidades que tornam o sistema RacerPro particularmente adequado para o desenvolvimento do protótipo do modelo formal escolhido neste trabalho de tese. A Figura 3.9 mostra a interface RacerPRo pronta para execução em conjunto com o Protègè e os agentes PFMEA.

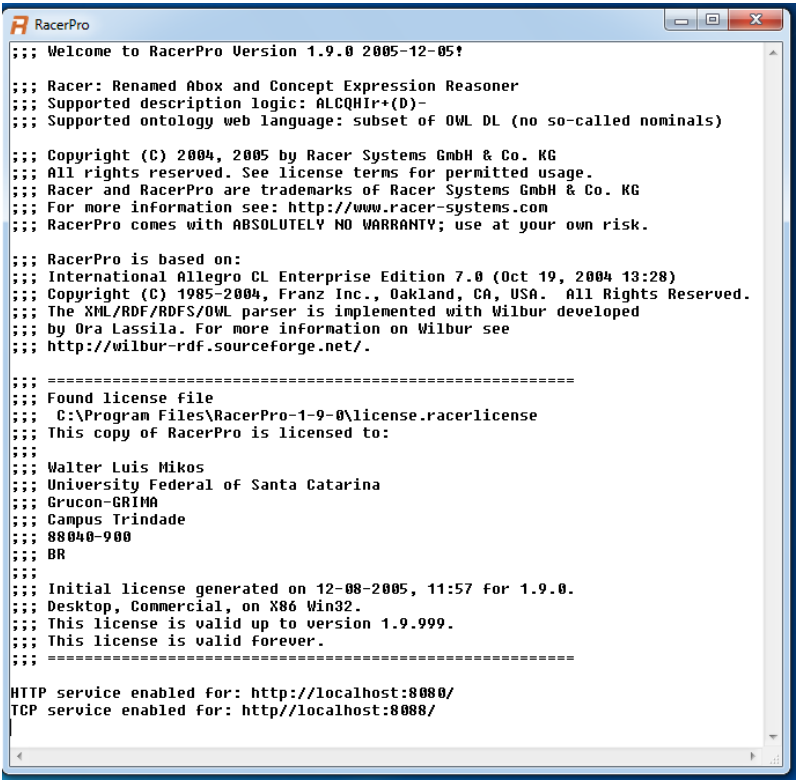

Figura 3-9 - Interface RacerPRO.
Fonte: Criado pelo autor

3.4 EDITOR GRÁFICO PARA A IMPLEMENTAÇÃO DA ONTOLOGIA

Em primeiro lugar, o editor responsável pela implementação computacional da base de conhecimento ontológica, deve ser capaz de suportar a linguagem padrão para ontologias OWL-DL. Este editor deve ainda permitir a edição e a visualização da estrutura lógica de classes (*Description Logic - concepts*) e propriedades (*Description Logic - roles*) e, em especial, permitir a execução de serviços de inferência por meio de raciocinadores (*reasoners*) externos responsáveis pela verificação da consistência da ontologia.

3.4.1 Processo de escolha do editor para implementação da ontologia

Neste contexto, o processo de seleção do editor partiu de um estudo comparativo apresentado por Su e Ilebrekke (2002) sobre as seis ferramentas mais relevantes relatadas na literatura, a saber: Ontolingua (FARQUHAR et al., 1996), WebOnto (DOMINGUE et al., 2000), WebOde (ARPÍREZ et al., 2001), Protégé-2000 (GROSSO et al., 1999 e Yan, W., 2014), OntoEdit (ONTOPRISE, 2003) e OilEd (BECHHOFER et al., 2001).

No estudo de Su e Ilebrekke (2002) empreendeu-se uma análise profunda dos editores a partir de seis tipos de qualidade: física, empírica, sintática, semântica, semântica percebida e pragmática. Entretanto, cumpre destacar que este estudo não tem o propósito de estabelecer uma hierarquia entre os editores, mas fornecer subsídios para uma escolha consistente.

Adicionalmente, Jakkilinki et al. (2004) apresentam uma comparação entre os editores Ontoligua (FARQUHAR et al., 1996), Protégé-2000 (GROSSO et al., 1999) e OntoEdit (ONTOPRISE, 2003). Jakkilinki et al. (2004) apresentam uma análise comparativa de modo a selecionar o editor mais apropriado a partir de conjuntos de critérios de avaliação, a saber: critérios relacionados à usabilidade do editor, e critérios relacionados aos aspectos ontológicos propostos por Corcho et al. (2003) e Duineveld et al. (2000). Os aspectos de usabilidade envolvem: a interface homem/máquina, possibilidade de instalação do software localmente ou a necessidade de usar um servidor remoto, estabilidade do software, disponibilidade da licença e suporte técnico.

Por sua vez, os aspectos ontológicos tratam das características da ontologia, e envolvem a capacidade do editor em:

- Permitir o uso do princípio de herança múltipla, que é a capacidade de uma classe derivada ter mais de uma classe base, cuja finalidade é permitir que novas classes herdem características das várias classes base (não relacionadas).
- Permitir uma decomposição exaustiva representada pela possibilidade de todas as instâncias de uma classe serem também instâncias de uma subclasse.
- Permitir a decomposição disjunta, isto é, se dois conceitos são disjuntos isto significa que não existem instâncias comuns.

Nestes estudos destacam-se o editor OilED e Protégé-2000 respectivamente, os quais a partir de 2003 foram incorporados no âmbito do projeto CO-ODE (*Collaborative Open Ontology Development Environment*) (CO-ODE, 2011), cujo objetivo é integrar ambos os editores em uma plataforma única denominada Protégé-OWL, a qual une três paradigmas: *frames*[18], lógica de descrições e RDF[19], a partir das funcionalidades *plug-and-play* existentes no Protégé-2000 visando proporcionar acesso a uma futura gama de outras ferramentas em código aberto.

3.4.2 Características do editor Protégé-OWL essenciais à implementação do modelo

No contexto deste trabalho de tese foi escolhido o editor Protégé-OWL (versão 3.4.7), o qual consiste de um ambiente interativo para a construção de ontologias, que oferece uma interface gráfica para a edição e manutenção de ontologias, e ainda suporta a implementação de ontologias de acordo com a linguagem OWL-DL. Além disso, o editor Protégé-OWL permite a integração direta com o sistema de raciocínio RacerPro para a verificação da consistência da ontologia e classificação de taxonomias. A Figura 3.10 mostra as definições das classes OWL-DL para a ontologia PFMEA referente à extrusão de alumínio, enquanto a

[18]Da teoria de *frames* proposta por Minsky (1974), onde um *frame* é uma descrição de um objeto complexo que compreende um nome e um grupo de *slots*, que consistem de um conjunto de atributos de valores particulares denominados de facetas.

[19]Do termo em inglês *Resource Description Framework*, que representa uma família de especificações do W3 Consortium (W3C, 2004) disponível em http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

Figura 3.11 mostra a representação gráfica da taxonomia de classes OWL-DL.

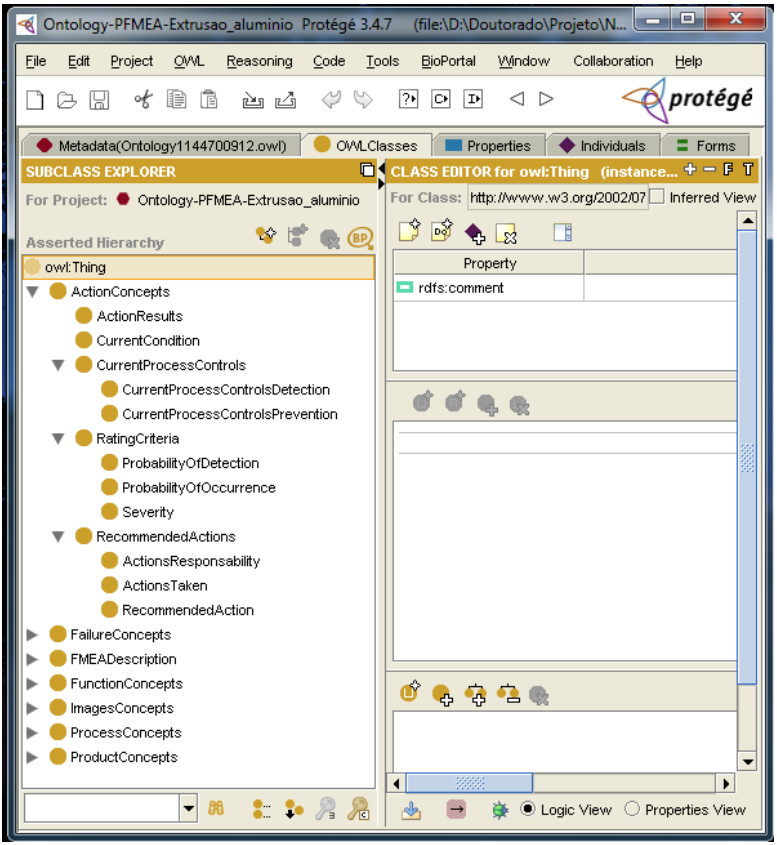

Figura 3-10 - Definições de classes OWL-DL para a ontologia PFMEA – Extrusão de Alumínio.

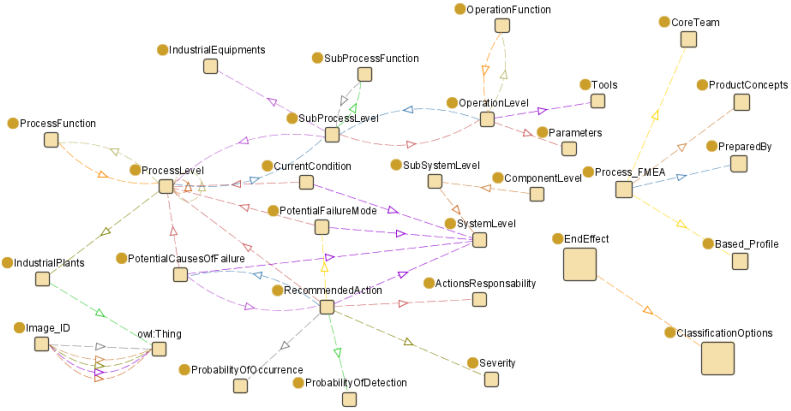

Figura 3-11 - Representação gráfica da taxonomia de classes OWL-DL da ontologia PFMEA.

3.5 Conclusões do capítulo

Este capítulo apresentou o processo de escolha dos recursos de software destinados à implementação do modelo baseado em agentes, considerando critérios que se fundamentam na literatura relevante da área, bem como as especificações de projeto do modelo detalhadas no capítulo 4.

Por outro lado, a descrição dos recursos de software selecionados concentrou-se, particularmente, nas características que os tornam especialmente adequados às necessidades e requisitos do modelo formal. No entanto, estudos comparativos detalhados dos recursos disponíveis, bem como a descrição detalhada destes recursos estão além da finalidade deste trabalho, e tais conhecimentos podem ser encontrados nas referências bibliográficas indicadas ao longo do texto.

Neste cenário, selecionou-se, em primeiro lugar, o arcabouço JADE (*Java Agent DEvelopment Framework*) desenvolvido em conformidade às especificações FIPA 2000 - *Foundation for Intelligent Physical Agents*, e amplamente utilizado para a implementação da tecnologia de agentes tanto em aplicações de natureza acadêmica quanto industrial.

É importante destacar que, de uma perspectiva metodológica, a escolha deste arcabouço visa não somente a redução dos esforços dedicados à implementação dos agentes mediante a extensão das classes abstratas em linguagem Java disponíveis, mas, em especial, reduzir os

esforços de verificação das funcionalidades pré-definidas e já implementadas. Isto porque, estas funcionalidades são genéricas e independem de uma aplicação em questão. Entre estas funcionalidades pode-se citar a arquitetura da plataforma que envolve os serviços de transporte de mensagens, páginas amarelas, páginas brancas e gerenciamento do ciclo de vida dos agentes e, em particular, as funcionalidades que possibilitam um ambiente de execução distribuído em uma rede.

Nesta mesma linha, estão o arcabouço jCOLIBRI e o sistema de raciocínio e recuperação de conhecimento RacerPro. Portanto, neste contexto, os procedimentos de verificação e validação do modelo podem concentrar-se especificamente nos aspectos dependentes da aplicação, em especial aqueles relacionados ao modelo de tarefas dos agentes.

4 DESENVOLVIMENTO DO MODELO CONCEITUAL E ESPECIFICAÇÕES DE PROJETO

Este capítulo descreve a pesquisa realizada pelo autor no tocante ao desenvolvimento do modelo conceitual da organização multiagente sugerida e a sua respectiva representação em termos de uma especificação de projeto. Dentro desta perspectiva, o capítulo é estruturado de acordo com as etapas previstas na metodologia GAIA proposta por Wooldridge et al. (2002) e complementada por Zambonelli et al. (2003) e Cernuzzi e Zambonelli (2005).

4.1 DESENVOLVIMENTO DO MODELO CONCEITUAL

A análise conceitual prevista na primeira etapa da metodologia GAIA (WOOLDRIDGE et al., 2000, ZAMBONELLI et al., 2003; CERNUZZI e ZAMBONELLI, 2005) busca capturar a essência da organização multiagente e sua estrutura. Como mencionado no capítulo anterior, no contexto desta metodologia uma organização multiagente é vista como uma coleção de papéis (*roles*) que se mantêm em determinados relacionamentos de uns para com os outros, e que participam de padrões sistemáticos e institucionalizados de interação com outros papéis.

Portanto, busca-se modelar a organização em termos de um conjunto de papéis que são desempenhados no âmbito da organização multiagente. Como resultado da etapa de análise o modelo conceitual da organização compreende dois modelos particulares: um modelo de papéis e um modelo de interação que identifica de que forma estes papéis interagem entre si.

A metodologia conduz o desenvolvedor a pensar a organização multiagente como um processo de projeto de uma organização humana tal como uma empresa, a qual, tipicamente, possui papéis como: presidente, vice-presidente, etc. Por consequência, a concretização da empresa ocorre com a instanciação destes papéis, isto é, quando indivíduos desempenham o papel do presidente, vice-presidente e assim por diante, embora não seja excluída a possibilidade de um indivíduo vir a assumir mais de um papel. A Figura 4.1 mostra a hierarquia de conceitos usados pela metodologia para modelar uma organização multiagente em termos de um modelo de papéis.

Figura 4-1 - Conceitos da etapa de análise da metodologia GAIA.
Fonte: WOOLDRIDGE et al., (2000).

Conceitualmente um papel pode ser caracterizado por meio de quatro tipos de atributos: permissões, responsabilidades, atividades e protocolos. As permissões ou direitos associados a cada papel estão relacionados ao tipo e à quantidade de informações que podem ser explorados à medida que um papel é desempenhado. Estas permissões são definidas a partir de dois aspectos principais: o primeiro identifica os recursos que podem ser legitimamente usados durante a execução do papel e, neste caso, os recursos referem-se a informações e conhecimento que os agentes podem ter acesso.

O segundo aspecto define os limites dentro dos quais o papel executor pode operar em relação aos recursos, isto é, para realizar um papel um agente tipicamente necessita acessar determinados recursos. Entretanto, alguns podem gerar informações e conhecimentos, outros podem acessar os recursos, mas sem modificá-los, enquanto outros podem necessitar modificar estes recursos.

As responsabilidades de um papel definem a sua funcionalidade, e estas responsabilidades podem ser divididas em duas categorias; propriedades de execução e propriedades de segurança. As propriedades de execução definem o que deve ocorrer, e especificam o ciclo de vida do papel, enquanto as propriedades de segurança estabelecem as condições de segurança que precisam ser mantidas para evitar situações indesejadas do ponto de vista da organização.

As atividades representam as ações que o papel deverá desempenhar, sem, no entanto, interagir com outros papéis da

organização. O protocolo, por sua vez, define a forma pela qual um papel interage com os outros papéis.

Em uma organização multiagente existem dependências inevitáveis e relacionamentos entre os diferentes papéis. De fato, tais interações são elementos chave do modo de funcionamento desta organização. Portanto, é necessário capturar e representar estas interações na fase de análise. Na metodologia GAIA tais conexões entre papéis são representadas por meio do modelo de interações, que consiste em um conjunto de definições de protocolos, um para cada tipo de inter-relação dos papéis. Este modelo pode ser entendido como um padrão formal de interações organizacionais e representa uma sequência particular de interações.

O foco do modelo de interações está no propósito da interação, em vez da ordem precisa de troca de mensagens, pois, tipicamente, uma única definição de protocolo pode gerar diversas trocas de mensagens no contexto do modelo computacional (*run time*).

Um protocolo é composto por seis atributos: propósito, iniciador, respondedor, entradas, saídas e processamento. O propósito representa a natureza da interação (e.g. requisição de informação) que é iniciada pelo papel iniciador, o qual utiliza recursos de informações ou conhecimentos denominados de entradas durante a execução da interação. Em seguida, durante o curso da interação, o outro papel envolvido, chamado de "respondedor", utiliza também recursos de informações ou conhecimentos para responder a requisição denominada de saídas e, por fim, o processamento descreve a ação do iniciador durante a interação.

Por fim, o processo de análise conceitual pode ser sintetizado nos seguintes estágios (WOOLDRIDGE et al., 2000):

(a) Identificar os papéis no âmbito da organização multiagente:
Resultado: A partir das definições de requisitos da organização multiagente é estabelecido um modelo de papéis protótipo caracterizado por uma descrição preliminar dos papéis.

(b) Para cada papel, identificar e documentar os protocolos associados, observando-se que os protocolos são os padrões de interações que ocorrem na organização entre os vários papéis:
Resultado: Um modelo de interações, o qual captura o padrão recorrente de interações entre os papéis.

(c) Usar os protocolos como base para refinar o modelo de papéis:
Resultado: Um modelo de papéis refinado, o qual documenta os papéis chave que ocorrem na organização multiagente,

suas permissões, responsabilidades e atividades, bem como os protocolos nos quais tomam parte.

4.1.1 Definição dos requisitos do modelo

Considerando o escopo e os objetivos deste trabalho de tese a definição dos requisitos do modelo é apresentada no diagrama IDEF0[20] (KBSI, 2011) da Figura 4.2.

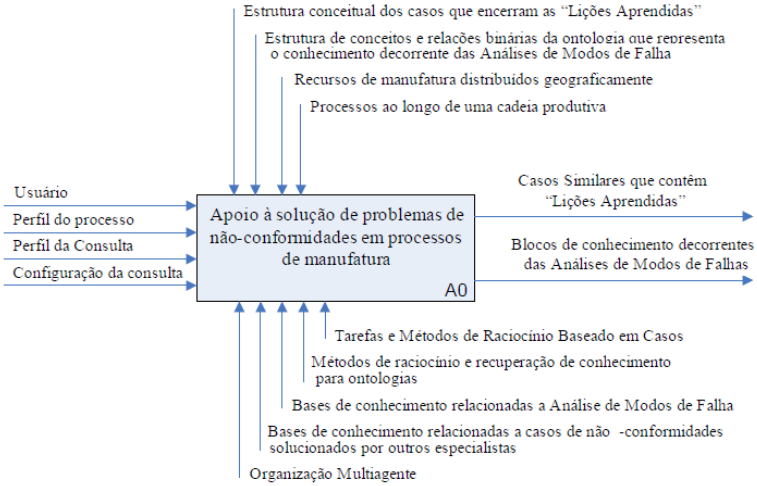

Figura 4-2 - Diagrama IDEF0 – Nível A0 para a função de apoio proposta.
Fonte: Criado pelo autor.

4.1.2 Modelo de agentes

A Figura 4.3 apresenta o modelo de agentes proposto para a organização multiagente a partir da fase de análise da metodologia GAIA. Esta figura ilustra o mapeamento dos papéis (lado direito) em relação às classes de agentes (lado esquerdo).

[20] Do termo em inglês ***I**ntegrated **DEF**initon Methods*, o IDEF0 é um método de modelagem baseado representações de diagramas desenvolvido a partir da Técnica de Análise e Projetos Estruturados (*Structured Analysis and Design Technique-SADT*) e, atualmente, mantido pela empresa *Knowledge Based Systems, Inc* (KBSI).

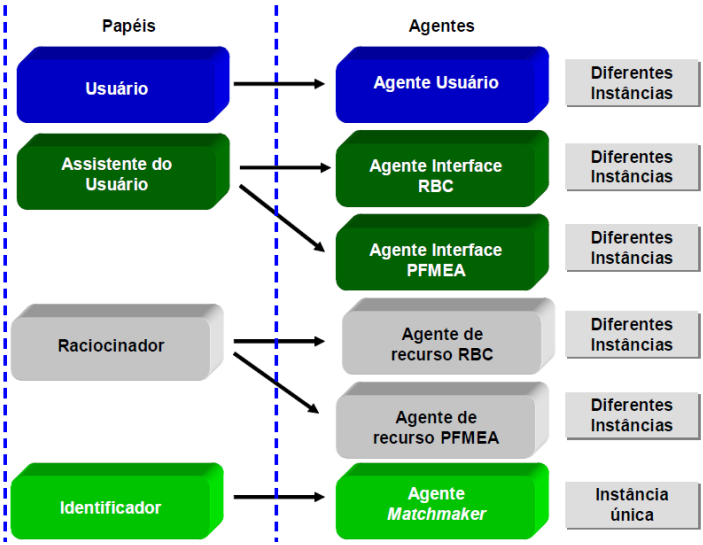

Figura 4-3 - Modelo de agentes proposto.
Fonte: Mikos, 2009.

Em primeiro lugar, é importante observar que o papel do raciocinador desdobra-se em duas classes distintas de agentes. A primeira classe de agentes corresponde à perspectiva de apoio à solução de problemas de não-conformidades baseada em cenários de recuperação de conhecimentos a partir da solução de casos de não-conformidades similares ocorridos no passado (Agentes de recurso RBC). E a segunda é relacionada à perspectiva de apoio baseada em cenários de recuperação de conhecimentos produzidos a partir da aplicação do método preventivo (Agentes de recurso PFMEA).

Como mostra a figura 4.3, tais agentes podem ser instanciados tantas vezes quanto necessário, de forma que cada agente, em particular, possa estar localizado em uma das diferentes organizações que colaboram no ambiente de manufatura distribuída (diferentes *hosts* na *Intranet/Internet*), bem como conter informações relacionadas a um processo específico ao longo da cadeia produtiva. Desta forma, caracteriza-se a ótica de distribuição tanto geográfica como organizacional.

Na mesma linha de pensamento, o papel do assistente do usuário desdobra-se em duas classes distintas, uma para cada perspectiva de apoio

à solução de não-conformidades, as quais podem ser instanciadas tantas vezes quantas forem os usuários requisitando apoio.

Por outro lado, o papel do identificador corresponde a somente um agente, denominado Agente *Matchmaker*[21], o qual centraliza todos os registros de serviços e localizações dos demais agentes e deve ser mantido sempre ativo na organização.

4.1.3 Modelo de serviços

Os principais modelos de serviços previstos na organização multiagente estão associados aos agentes de recursos RBC e agentes de recursos PFMEA. Estes modelos, segundo Wooldridge et al. (2000), decorrem do modelo conceitual e envolvem os protocolos, atividades e propriedades de execução e segurança identificadas no modelo do papel correspondente ao agente em questão.

Em primeiro lugar, o serviço de raciocínio e recuperação de conhecimentos próprios dos agentes de recursos RBC, fundamenta-se nas tarefas e métodos de raciocínio baseado em casos apresentados e discutidos na literatura por Kolodner (1993), Aamodt e Plaza (1994), Watson (2003) e von Wangenheim e von Wangenheim (2003).

Por sua vez, o serviço de raciocínio e recuperação de conhecimento próprio dos agentes PFMEA fundamenta-se nos algoritmos de inferência para linguagens baseadas em lógica, em particular, pelo algoritmo tableaux (BAADER e SATTLER, 2001). Estes algoritmos, em geral, são capazes de processar bases de conhecimento formalizadas a partir de ontologias e codificadas por linguagens baseadas em lógicas (BAADER e SATTLER, 2001; HAARSLEV e MÖLLER, 2001; TSARKOV e HORROCKS, 2005).

É importante destacar que a descrição do algoritmo tableaux está além do escopo desta tese, todavia tal descrição pode ser encontrada nas referências citadas.

4.1.4 Modelo de afinidades

O modelo de afinidades é representado nas Figuras 4.4 e 4.5, e ilustra a comunicação prevista entre os agentes propostos pelo modelo.

[21] Do termo da área de ciência da computação (em inglês *Match*), relaciona-se à noção de verificar a repetição ou igualdade entre conjuntos de dados. Contudo, a terminologia da área de sistemas multiagentes refere-se à verificação e combinação de padrões de serviços disponibilizados pelos diferentes agentes da sociedade.

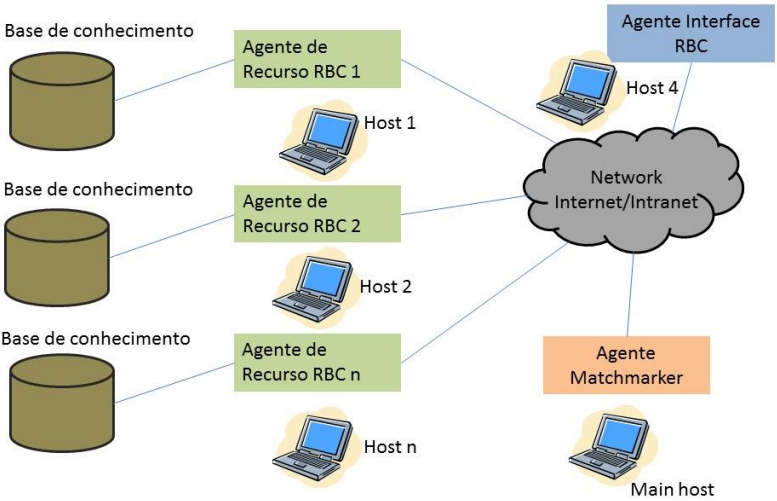

Figura 4-4 - Modelo de afinidades RBC.
Fonte: Criado pelo autor

Neste sentido, pode-se perceber que as instâncias do agente de interface devem agir em nome do usuário na organização multiagente proposta. Além disso, estes agentes deverão comunicar-se diretamente com as instâncias dos agentes de recursos de conhecimento ativos na organização e devidamente identificados pelo agente *matchmaker*.

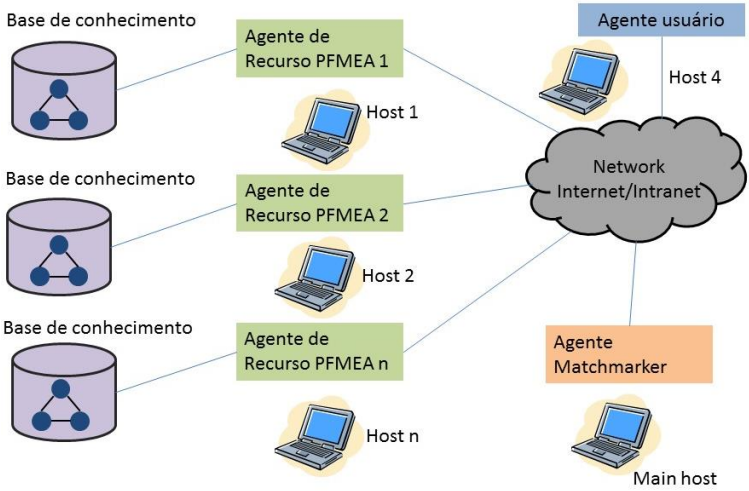

Figura 4-5 - Modelo de afinidade PFMEA.
Fonte: Criado pelo autor.

Destaca-se que estas figuras ilustram três instâncias de cada uma das classes de agente de recurso, os quais podem estar localizados em diferentes hosts distribuídos geograficamente, bem como representando processos específicos dentro da cadeia produtiva.

Adicionalmente, percebe-se que cada agente de recurso de conhecimento deverá acessar e manipular a sua própria base de conhecimento usando para isto os métodos de raciocínio previstos para o comportamento do agente.

4.2 Estrutura conceitual para um caso de não-conformidade

A estrutura conceitual de um caso de não-conformidade representa o centro da base do conhecimento que será acessada e manipulada pelos agentes de recursos de conhecimento CBR previstos.

Dependendo da área de aplicação, um caso pode ter diferentes conteúdos e representações, mas a estrutura mínima que um caso deve ter é a descrição do problema e sua solução. A representação dos casos pode ser feita de diferentes maneiras, dentre as quais tem-se: redes semânticas, quadros, objetos e árvores. Qualquer que seja a representação utilizada, também deve-se representar informações sobre a condição da não-conformidade e ações corretivas.

A estrutura proposta envolve os seguintes elementos: os descritores da não-conformidade, os descritores da solução sugerida, e os descritores dos resultados obtidos após a implementação da solução. Na Figura 4.3 é mostrada a estrutura usada neste trabalho para a representação de um caso, e os elementos que compõem a estrutura de casos são descritos nas próximas seções.

Figura 4-6 - Estrutura usada neste trabalho para um caso de não-conformidade.
Fonte: Criado pelo autor.

4.2.1 Descrição da não-conformidade

Este campo especifica um código de identificação único, que inclui a descrição da não-conformidade (tipo, localização, data da ocorrência, entre outros), e algumas informações adicionais como o usuário que identificou/realizou a tarefa de correção no processo de extrusão. O código de identificação de cada caso é essencial, o qual está relacionado com a fase de recuperação, e também com o conceito de indexação para acelerar a busca de situações semelhantes no banco de dados.

4.2.2 Descrição da solução

A solução para o domínio de aplicação considerado neste trabalho corresponde às sugestões de possíveis ações a serem efetuadas no processo de extrusão de alumínio, tendo em vista o diagnóstico da não-conformidade (Wangenheim et al., 2003). A identificação da não-conformidade é muito importante porque permite decidir quais ações devem ser tomadas para evitar ou minimizar as consequências da referida não-conformidade. Na solução também é comum adicionar a causa do problema.

4.2.3 Resultados

O campo de resultados contém uma avaliação do diagnóstico sugerido pelo sistema CBR, isto é, se ele foi correto, e se a ação de correção foi eficaz ou não. Este campo contém os dados sobre o que ocorreu durante a execução da solução sugerida, e se essa solução foi bem sucedida ou não. Com esta informação o especialista pode antecipar os problemas potenciais e prever as consequências de uma ação.

A avaliação do resultado de um processo pode levar algum tempo, mas se ela for positiva ela pode ser armazenada e reutilizada para novas situações.

4.2.4 Exemplo da estrutura conceitual de um caso

Neste cenário, a estrutura conceitual de um caso de não-conformidade representa o centro da base do conhecimento que será acessada e manipulada pelos agentes de recursos de conhecimento RBC. Esta estrutura envolve três elementos principais: os descritores da não-conformidade, os descritores da solução sugerida, e os descritores dos resultados obtidos após a implementação da ação, como mostrado nas Tabelas 4.1, 4.2 e 4.3.

Tabela 4.1 - Exemplo de descritores para a não-conformidade "bolha"

Descrição da não-conformidade	Bolha
Método de controle para detectar	Análise visual
Localização da não-conformidade	Extremidades do perfil
Quando ocorreu a não-conformidade	Durante o processo de produção

Tabela 4.2 - Exemplo de descritores da solução sugerida para uma "bolha"

Equipe responsável	Time A
Causa raiz	Lubrificação do tarugo
Ação corretiva	Verificação do procedimento de lubrificação
Causa potencial de falha	Excesso de lubrificante
Passos para resolver falha	Diminuir a quantidade de lubrificante utilizado

Tabela 4.3 - Exemplo de descritores dos resultados para uma "bolha"

O resultado em si	Diminuir a quantidade de lubrificante utilizado
Análise crítica do resultado	Testes foram realizados e criou-se um novo procedimento de lubrificação
Ações preventivas sugeridas	Criação de um procedimento de lubrificação a fim de evitar excesso/falta de lubrificante

É importante observar que a estrutura conceitual para o caso de não-conformidade foi proposta com o objetivo de torná-lo independente do domínio de aplicação, isto é, independente do processo de fabricação. A fim de fazer esta estrutura aderente a um determinado processo de fabricação, é necessário realizar a especialização do primeiro elemento da estrutura (ou seja, descritores para o problema/situação da não-conformidade), tendo em conta as características do processo de fabricação considerado.

4.2.5 Cálculo da similaridade

O sistema CBR implementado pelo *framework jCOLIBRI* realiza o cálculo da similaridade entre um novo caso e os casos armazenados na base de casos usando o algoritmo do vizinho mais próximo (Kolodner, 1993), conforme mostrado na Equação (1). Verifica-se cada descritor de maneira individual e seu grau de relevância, representado pelo peso. Cada descritor é representado por um "*string*" que é construído com base nos dados introduzidos pelo usuário na interface de entrada de dados.

$$sim(Q,C) = \frac{\sum_{i=1}^{n} sim(Q_i, C_i) * w_i}{\sum_{i=1}^{n} w_i} \quad (1)$$

onde:

Q é o novo caso, *C* é o caso já existente na base de casos, *i* o índice do descritor, w_i é o peso que representa a importância de cada descritor no cálculo de similaridade. $sim(Q_i,C_i)$ é a similaridade entre os valores dos descritores, que é calculado pela Equação (2).

$$sim(Q_i, C_i) = 1 - \left(\frac{|Q_i - C_i|}{l_i}\right) \quad (2)$$

onde:

Q_i é o valor do descritor *i* no novo caso, C_i é o valor do descritor *i* no caso presente na base de casos, l_i é o intervalo dos valores pré-determinados do descritor *i*.

O cálculo da similaridade considera a média obtida pelo somatório de cada descritor, e o valor da similaridade é utilizado pelo agente para recuperar os casos mais similares na base de casos. O sistema retorna os três casos mais similares aos descritores de não-conformidades.

4.2.6 Exemplo de cálculo de similaridade

Analisando as Tabelas 4.4 e 4.5 observa-se a relação entre o caso novo (consulta) e os casos 1 e 5 (recuperados na base de dados) e como a similaridade é calculada.

Tabela 4.4 - Comparação do caso novo e do caso 1 recuperado.
Fonte: Criado pelo autor.

Caso novo	*Peso*	*Similaridade*	*Caso 1*
Bolhas	1.0	10	Bolhas
Análise visual	1.0	10	Análise visual
Próximo da emenda	1.0	10	Próximo da emenda
Durante o *start-up*	1.0	10	Durante o *start-up*

Similaridade = ((1*10) +(1*10) + (1*10) + (1*10)) / (10+10+10+10)
Similaridade = 1.0

Tabela 4.5 - Comparação do caso novo e do caso 5 recuperado.
Fonte: Criado pelo autor.

Caso novo	*Peso*	*Similaridade*	*Caso 5*
Bolhas	1.0	10	Bolhas
Análise visual	1.0	10	Análise visual
Próximo da emenda	1.0	9.5	Extremo do perfil
Durante o *start-up*	1.0	9.5	Depois da parada de produção

Similaridade = ((1*10) +(1*10) + (1*9.5) + (1*9.5)) / (10+10+10+10)
Similaridade = 0.97

4.3 Conclusões do capítulo

Este capítulo apresentou o processo de desenvolvimento do modelo conceitual: definições dos requisitos do modelo, estrutura conceitual para os casos de não-conformidades, as similaridades calculadas para obtenção das soluções mais próximas à não-conformidade requisitada. As definições dos requisitos de modelo estão ilustradas no diagrama IDEF0 (Figura 4.2), onde pode-se observar as funções do sistema protótipo.

Por sua vez, a estrutura conceitual trata da composição de um caso de não-conformidade e, neste sentido, foi estruturado como descrição da não-conformidade, descrição da solução e resultados. É importante destacar que os cálculos de similaridade local e global buscam e comparam as informações solicitadas referentes às consultas de acordo com a composição de um caso de não-conformidade, e extraindo das bases os três casos mais similares.

5 IMPLEMENTAÇÃO DO MODELO BASEADO EM AGENTES PROPOSTO

Este capítulo apresenta os detalhes do desenvolvimento formal do comportamento de cada uma das classes ou famílias de agentes propostos, a saber: agentes de interface, agentes de recursos de conhecimento e o agente *Matchmaker*. Neste contexto, um comportamento é especificado pelo seu modelo de tarefas.

Para estabelecer uma coerência lógica na estrutura do texto deste capítulo, os detalhes de desenvolvimento são apresentados na seguinte ordem: (a) os agentes de interface para recuperação de casos (ou agente de interface RBC) e os agentes de recursos de conhecimento para Raciocínio Baseado em Casos (ou agentes de recursos RBC); (b) o agente de interface para recuperação de conhecimento decorrente da aplicação do método de Análises de Modo de Falha e Efeitos em processos de manufatura PFMEA (ou agentes de interface PFMEA) e os agentes de recursos de conhecimento que contêm os agentes de Análise de Modos de Falha e Efeitos (ou agentes de recursos PFMEA); (c) o agente *Matchmaker*.

É importante lembrar que os agentes propostos podem estar distribuídos em diferentes organizações de manufatura (ou, em termos computacionais, distribuídos entre diferentes *hosts* em uma rede de computadores). Além disso, estes agentes podem referir-se a processos manufatura específicos, caracterizando, portanto, uma distribuição não somente geográfica, mas também no âmbito de processos e organizações ao longo de uma cadeia de produtiva.

Deve-se destacar ainda que estes agentes compõem uma organização de natureza dinâmica, pois os agentes da família de agentes de interface e de recursos podem, eventualmente, deixar a sociedade, ou mesmo novos agentes podem ser incluídos a qualquer tempo em função de aspectos organizacionais da cadeia produtiva (como por exemplo, com a inclusão de novos processos ou novas organizações).

Em termos metodológicos, cumpre destacar que ao longo do desenvolvimento formal do modelo de tarefas dos agentes busca-se indicar as classes dos arcabouços que foram estendidas especialmente para a construção do protótipo, bem como os seus métodos, de modo que novas e mais refinadas implementações possam ser reproduzidas. No entanto, está além do escopo deste capítulo apresentar os detalhes das classes originais, os quais podem ser obtidos nas referências bibliográficas citadas ao longo do texto.

5.1 ASPECTOS ESSENCIAIS DA IMPLEMENTAÇÃO DOS AGENTES PROPOSTOS

Inicialmente, do ponto de vista computacional, cumpre ressaltar que, no âmbito da plataforma JADE, a implementação e execução das tarefas de cada um dos agentes é controlada de acordo com um conjunto de etapas programadas pela plataforma JADE. Neste contexto, a primeira etapa ocorre quando um dado agente é inicializado, que acarreta a execução do método construtor do agente denominado Agent da classe *jade.core.Agent*, e o agente recebe a sua identificação global única de acordo com as especificações FIPA (2002a).

Na segunda etapa ocorre o seu registro no Sistema Gerenciador de Agentes (AMS - *Agent Management System*), o qual é responsável pela manutenção do índice físico (*Agent Identifiers Directory* - AID) de todos os agentes residentes na plataforma e, em seguida, o agente é colocado no estado ativo.

Na terceira etapa é executado o método denominado *setup*, responsável pela execução das tarefas principais do agente. Deve-se destacar que este método representa o momento no qual as tarefas específicas de um agente, previstas para o seu comportamento, são realmente executadas.

Portanto, o desenvolvedor da aplicação deve implementar computacionalmente este método que, entre outras funções, deve realizar o registro da descrição do agente em questão e dos seus serviços junto ao Agente Facilitador de Diretório (DF - *Directory Facilitator*) se necessário. Além disso, este método deve adicionar pelo menos um comportamento na fila de comportamentos a serem executados pelo agente por meio do método denominado *addBehaviour*, disponível no arcabouço JADE.

5.2 AGENTE DE INTERFACE PARA RBC

De acordo com as especificações de projeto do modelo proposto, o agente de interface para recuperação casos (ou agente de interface RBC) tem por escopo interagir com os seguintes itens: (a) com entidades internas que se encontram ativas na plataforma, em particular com outros agentes de Raciocínio Baseado em Casos e com o agente *Matchmaker*, (b) com os outros usuários.

Com esta finalidade, o agente de interface RBC deve agir em nome dos usuários no âmbito da organização multiagente, e realizar, de maneira transparente ao usuário, a comunicação com os outros agentes envolvidos, visando completar, de modo cooperativo, as tarefas necessárias para raciocínio e recuperação de conhecimento no domínio em questão em apoio ao tratamento de não-conformidades.

Dentro desta perspectiva, dois aspectos são essenciais ao desenvolvimento do agente de interface: em primeiro lugar, o modelo de tarefas que expressa o comportamento do agente e seu respectivo desenvolvimento computacional e, em segundo lugar, a estratégia de configuração das consultas, bem como a apresentação dos respectivos resultados por meio de blocos de conhecimento mediante uma interface gráfica acessível ao usuário.

5.2.1 Desenvolvimento do comportamento principal do agente de interface

Em termos metodológicos, a Figura 5.1 apresenta o diagrama de sequência AUML[22] que modela a interação entre o usuário, o agente de interface RBC e um agente de RBC ativo na plataforma, indicando a abrangência do comportamento implementado pela extensão da classe original.

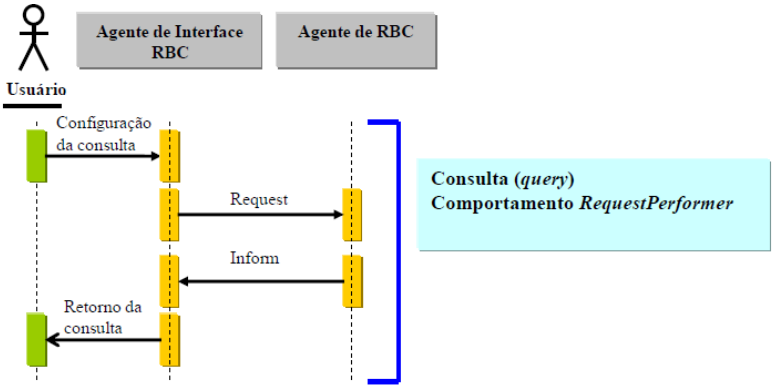

Figura 5-1 - Diagrama de sequência AUML e comportamentos do agente de interface. Fonte: Criado pelo autor.

[22] Do termo em inglês *Agent Unified Modeling Language* (AUML).

Este comportamento está de acordo com as especificações de projeto do modelo, pois os agentes foram projetados para permitir consultas (*queries*) em que o agente de interface deve comunicar-se com os agentes de recursos ativos de modo a recuperar um conjunto casos mais similares por meio de métodos e tarefas RBC.

O comportamento *RequestPerformer* é expresso pelo seu modelo de tarefas, como mostra o diagrama da Figura 5.2. Neste diagrama, a primeira tarefa implementa os métodos responsáveis por guiar e obter as consultas configuradas por um usuário. Esta tarefa, em particular, adota uma interface gráfica desenvolvida na forma de um Java *Applet*. Uma consulta consiste em estabelecer os descritores que representam um novo caso de não-conformidade sob análise e os seus referentes pesos, os quais podem ser atribuídos de forma personalizada pelo usuário em função de suas percepções e necessidades de recuperação.

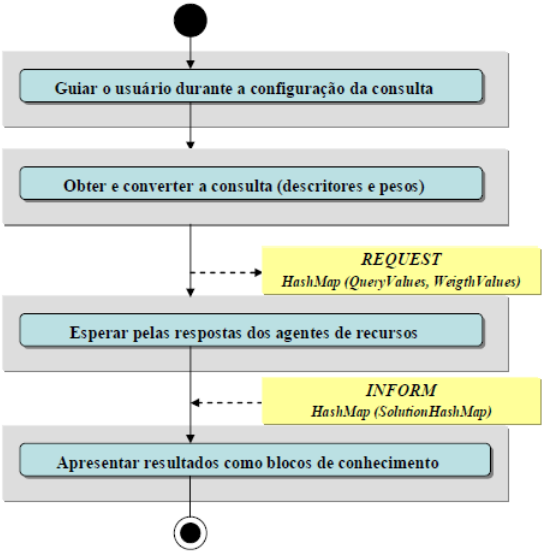

Figura 5-2 - Modelo de tarefas do comportamento principal dos agentes de interface RBC.
Fonte: Criado pelo autor.

Neste protótipo estes descritores estão relacionados, em particular, a processos de extrusão de alumínio, definidos na estrutura conceitual do caso de não-conformidades apresentada no capítulo 4.

Em relação à estrutura conceitual do caso de não-conformidade, destaca-se que o mesmo envolve quatro descritores, que são: descrição da

não-conformidade, método de controle para detecção, localização da não-conformidade e quando ocorreu a não-conformidade.

O comportamento principal prevê a implementação de métodos responsáveis por converter estes descritores e seus respectivos pesos para um formato conveniente, de modo a permitir a sua inclusão no conteúdo de uma mensagem FIPA ACL dotada do ato comunicativo tipo *Request*, cujo objetivo é estabelecer a interação social entre os agentes em questão. Neste protótipo o formato escolhido corresponde a tabelas ou mapas *hash*[23] na forma de objetos Java tipo *HashMap* mediante a instanciação da classe *java.util.HashMap*.

Deste modo, o agente de interface pode movimentar os agentes de recursos identificados na pesquisa realizada inicialmente junto ao agente *Matchmaker* (ou agente DF) para requisitar a execução do serviço de recuperação de conhecimento, tendo como referência o conteúdo da mensagem FIPA ACL. A Figura 5.3 resume a ação de comunicação.

Emissor:	Agente de interface para recuperação de casos
Receptor:	Agente de raciocínio baseado em casos
Performativa:	Request
Serviço:	Raciocínio baseado em casos
Conteúdo:	Objetos Java tipo *HashMap* (*QueryValues* e *WeightValues*).

Figura 5-3 - Síntese da comunicação entre o agente de interface e os agentes de recursos.
Fonte: Criado pelo autor.

Então, o agente de interface entra em modo de espera de acordo com o previsto em seu comportamento (Figura 5.2), esperando por mensagens dotadas na ação de comunicação *Inform* enviadas pelos agentes de recursos RBC ativos. Estas mensagens têm o seguinte conteúdo: a medida de similaridade global entre os descritores do caso sob consulta e os descritores do caso similar recuperado; os valores dos quatro descritores armazenados com o este caso e usados para o cálculo

[23] Do termo em inglês *hashing*, é uma estrutura de dados especial que armazena chaves de pesquisa (*hash*) e valores na forma de objetos Java. A biblioteca Java dispõe da implementação da classe de propósito geral para mapas denominada *HashMap* usada nesta aplicação.

da similaridade global, observando-se que o formato adotado foi um vetor de *HashMaps* denominado de *solutionHashMap*.

Por fim, o comportamento do agente prevê a conversão dos objetos *HashMap* e a apresentação destes de forma compreensível ao usuário por meio de blocos de conhecimento disponibilizados na interface gráfica.

5.2.2 Janelas gráficas do agente e estratégia de configuração de consultas

A interface gráfica associada ao agente foi desenvolvida na forma de um *Java Applet*, de modo a ser compatível com os navegadores atuais. Esta interface compreende dois conjuntos de janelas gráficas distintas: o primeiro conjunto é destinado à configuração da consulta, e o segundo é voltado para a apresentação dos respectivos resultados da consulta, como mostra a Figura 5.4.

A estratégia de configuração adotada para a consulta permite a entrada não somente dos descritores conhecidos ou considerados relevantes pelo usuário, como também ajustar livremente os valores dos pesos (*weight*) que, em essência, expressam a importância de cada um destes descritores em função das percepções e necessidades de um usuário em particular, como ilustra a Figura 5.4. Esta estratégia envolve a seleção dos valores dos descritores de um novo caso de não-conformidade diretamente a partir de caixas tipo *ComboBox* disponíveis para os tipos de dados "símbolos não-ordenados" ou a entrada de dados numéricos ou *strings* em caixas próprias via teclado, como mostra a Figura 5.4.

A configuração a partir de caixas tipo *ComboBox* busca garantir a consistência terminológica dos descritores não numéricos. Isto porque as bases de conhecimento envolvem a manipulação de casos de solução de não-conformidade propostos por diferentes especialistas e, portanto, as opções no *ComboBox* devem ser incluídas a partir de termos amplamente usados no domínio em questão, bem como apoiados na literatura da área.

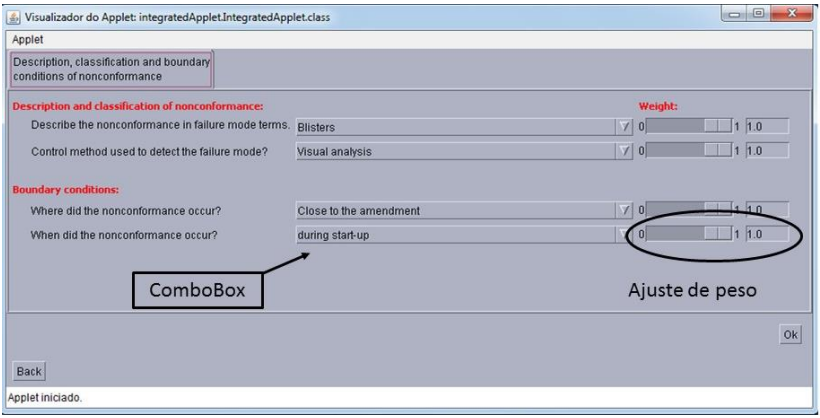

Figura 5-4 - Janela para configuração dos descritores.
Fonte: Criado pelo autor.

Por sua vez, o segundo conjunto de janelas gráficas, mostrado na Figura 5.5, foi projetado para apresentar os casos mais similares encontrados pelos agentes de recursos nas diferentes bases de conhecimento, os quais são decorrentes dos serviços de raciocínio e recuperação realizados.

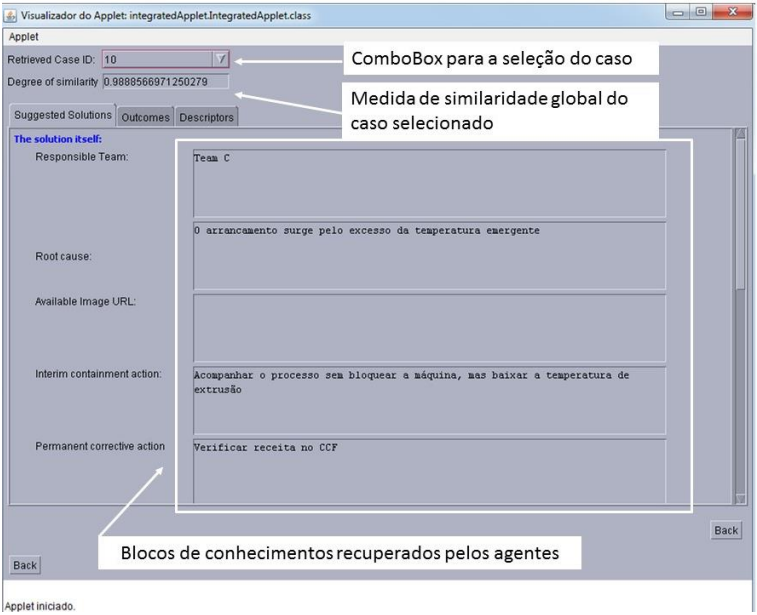

Figura 5-5 - Janela de apresentação dos casos mais similares recuperados.
Fonte: Criado pelo autor

A Figura 5.5 mostra que o conjunto é composto por três abas diferentes, as duas primeiras sendo destinadas aos atributos das soluções sugeridas e aos resultados que englobam as lições aprendidas por outros especialistas em casos similares de solução de problemas de não-conformidades. A última aba destina-se aos valores de todos os descritores armazenados junto aos casos recuperados, os quais foram usados para comparação de indexadores.

Para visualizar os casos mais similares, o usuário deve selecionar um caso particular por meio da caixa tipo *ComboBox* que apresenta a lista de identificadores dos casos recuperados. Deste modo, então, disponibiliza-se o grau de similaridade entre os descritores deste caso e os descritores informados pelo usuário, além dos blocos de conhecimento mencionados acima.

Portanto, a tática de configuração permite ao usuário a criação de uma consulta totalmente personalizada a partir de descritores de entrada validados de forma clara.

Além disso, a estratégia implementada permite a configuração de consultas mesmo que muitos dos descritores de entradas não sejam completados (ou por não serem conhecidos ou mesmo por não serem considerados relevantes por um usuário para uma dada situação), pois a estes descritores podem ser associados pesos distintos visando ajustar o cálculo da medida de similaridade global.

5.3 AGENTES DE INTERFACE PARA PFMEA

De acordo com as especificações de concepção do modelo, o agente de interface para recuperação de conhecimento decorrentes da aplicação do método PFMEA (ou agentes de interface PFMEA) tem por finalidade agir em nome dos usuários no contexto da plataforma, atuando de forma autônoma e transparente ao usuário, visando completar, de modo cooperativo, as tarefas necessárias para a recuperação de conhecimento no domínio em questão a partir de bases de conhecimento ontológicas PFMEA propostas neste trabalho de tese.

Neste cenário, destacam-se dois aspectos nesta subseção: o modelo de tarefas do agente e seu respectivo desenvolvimento computacional, e a estratégia de configuração das consultas e apresentação dos respectivos resultados.

5.3.1 Desenvolvimento computacional do comportamento do agente

O modelo de programação adotado no desenvolvimento deste agente segue o mesmo modelo apresentado para o agente de interface RBC. Neste sentido, a tarefa inicial implementada pelo método *setup* do agente de interface é destinada a pesquisar junto ao agente *Matchmaker* (ou agente DF) a descrição de todos os agentes de recursos PFMEA que dispõem do serviço de raciocínio e recuperação de conhecimento que se encontram ativos na plataforma. Isto foi feito visando conseguir os seus identificadores para futuras comunicações ponto a ponto, observando que esta atividade foi implementada a partir do método *search* da classe *jade. domain.DFService* disponível no arcabouço JADE.

5.3.2 Desenvolvimento computacional dos comportamentos principais do agente

O agente de interface PFMEA tem dois comportamentos principais denominados *updateQueryComplement* e *RequestPerformer*, que foram

implementados pela extensão da classe *jade.core.behaviours.OneShotBhaviour* que modela um comportamento atômico que deve ser executado uma única vez (BELLIFEMINE et al., 2012).

Este modelo de comportamento, em particular, está em concordância com as especificações de projeto, pois os agentes foram projetados para permitir uma consulta de dois estágios, onde o agente de interface deve recuperar um conjunto de instâncias iniciais que serão usados pelo usuário para configurar a consulta final. Por exemplo, quando um usuário busca recuperar todos os potenciais modos de falha e as respectivas causas identificadas pelo método PFMEA em relação a uma função de uma operação específica de manufatura, o agente de interface no primeiro estágio deve requisitar a recuperação de todas as funções de operações modeladas nas bases de conhecimento dos agentes de recursos e apresentá-las ao usuário para que este possa especificar a consulta final em um segundo estágio.

Em termos metodológicos, a Figura 5.6 apresenta o diagrama de sequência AUML, que modela esta interação entre o usuário, o agente de interface PFMEA e um agente de recurso PFMEA ativos na plataforma, indicando a abrangência dos comportamentos implementados pela extensão da classe original.

Na Figura 5.6 o comportamento *updateQueryComplement* implementa os métodos responsáveis pela obtenção das instâncias requisitadas pelo usuário no primeiro estágio da consulta, e esta obtenção deve acontecer de forma automática e transparente ao usuário mediante a troca de mensagens diretas entre o agente de interface e os agentes de recursos PFMEA ativos na plataforma identificados junto ao agente *Matchmaker* (ou agente DF). Este comportamento foi desenvolvido a partir de duas atividades principais, como mostra a Figura 5.7.

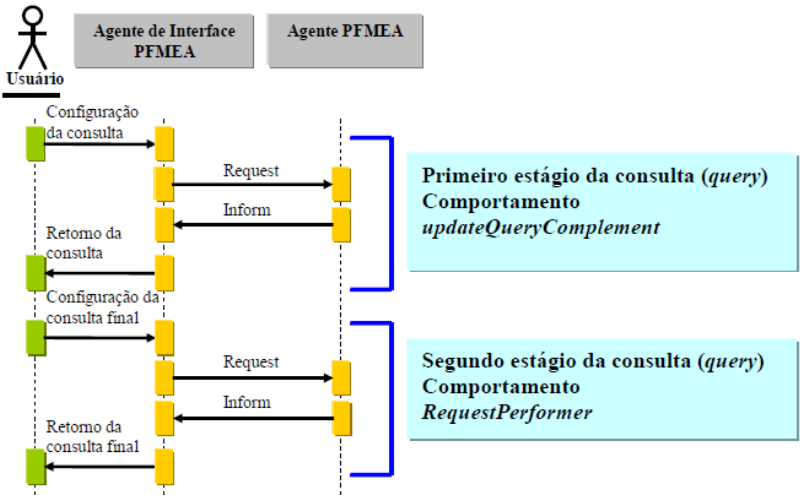

Figura 5-6 - Diagrama de sequência AUML e comportamentos do agente de interface.
Fonte: Criado pelo autor.

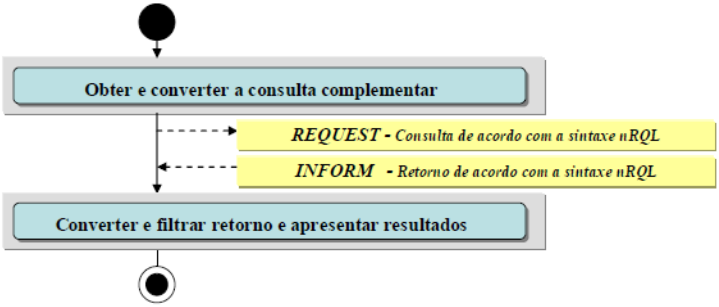

Figura 5-7 - Comportamento *updateQueryComplement* do agente de interface.
Fonte: Criado pelo autor.

A primeira atividade implementa os métodos responsáveis por guiar o processo de configuração da consulta a ser realizada pelo usuário assegurando a consistência semântica da consulta, bem como são implementados os métodos responsáveis por converter esta consulta para um formato que permita seu compartilhamento entre os agentes e transmiti-la como conteúdo da uma mensagem FIPA ACL com o ato comunicativo *Request* diretamente aos agentes de recursos PFMEA.

A segunda atividade implementa os métodos responsáveis por receber a mensagem FIPA ACL com o ato comunicativo *Inform*, em cujo conteúdo encontram-se as instâncias requisitadas e os métodos para tratar a mensagem a apresentá-la de forma compreensível ao usuário para que este possa realizar a consulta.

A janela gráfica do usuário apresentada na Figura 5.8 é parte fundamental da estratégia de consulta implementada pela primeira atividade, a qual se baseia na configuração dinâmica da consulta a partir de uma frase em linguagem natural, neste caso na língua inglesa, onde o usuário pode delimitar a extensão semântica verbal, determinando o objeto direto e os complementos desta frase.

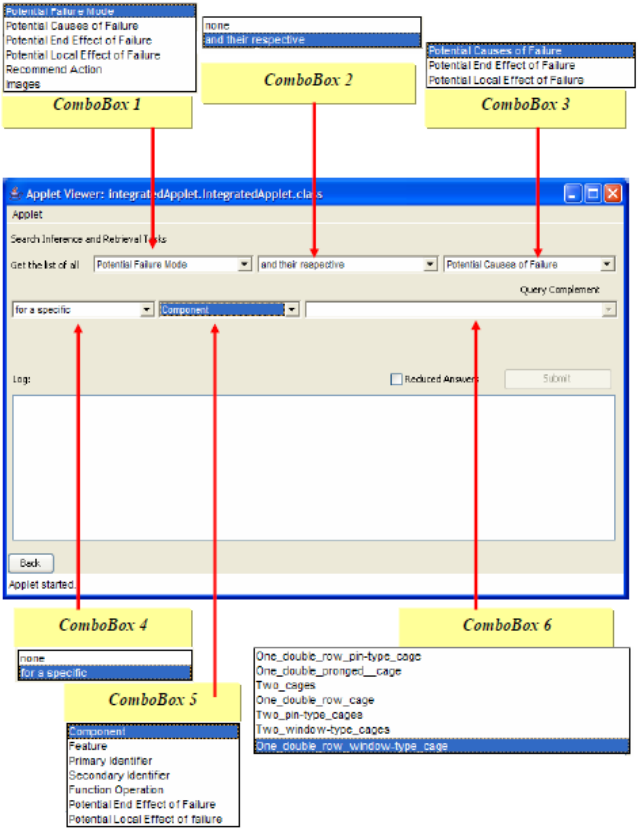

Figura 5-8 - Janela gráfica para consulta PFMEA.
Fonte: Mikos, 2009.

5.4 Conclusões do capítulo

Este capítulo apresentou o processo de implementação do sistema com seus recursos de raciocínio baseado em casos, PFMEA e multiagentes, considerando critérios que se fundamentam na literatura relevante da área, bem como as especificações de projeto do modelo detalhadas no capítulo 4.

Para o desenvolvimento de agentes, o arcabouço JADE (*Java Agent DEvelopment Framework*) desenvolvido em conformidade às especificações FIPA 2000 - *Foundation for Intelligent Physical Agents*, e amplamente utilizado para a implementação da tecnologia de agentes tanto em aplicações de natureza acadêmica quanto industrial, como funcionalidades pode-se citar a arquitetura da plataforma que envolve os serviços de transporte de mensagens, páginas amarelas, páginas brancas e gerenciamento do ciclo de vida dos agentes, e em particular as funcionalidades que possibilitam a execução do sistema em *hosts* distribuídos.

Nesta mesma direção, estão o arcabouço *jCOLIBRI* e o sistema de raciocínio e recuperação de conhecimento RacerPro. Assim sendo, nesta situação, os processos de verificação e validação do modelo podem concentrar-se especificamente nos aspectos dependentes da aplicação, em especial, aqueles relacionados ao modelo de tarefas dos agentes.

6 IMPLEMENTAÇÃO DAS BASES DE CONHECIMENTO DOS AGENTES DE RECURSOS DE CONHECIMENTO

Este capítulo tem por finalidade expor e descrever os aspectos relacionados à forma de implementação das bases de conhecimento dos agentes de recursos de conhecimento.

A etapa de implementação fundamenta-se nos trabalhos de Buchanan et al. (1983), Schreiber (1992), van Heijist et al. (1997), Fernández-Lopéz et al.(1999) e Garcia et al. (2005), cuja revisão sugere que esta etapa é parte de um processo de aquisição de conhecimento mais amplo, e que ocorre subsequentemente às etapas de identificação das fontes de conhecimento, sendo que a conceituação e a formalização já foram apresentadas e discutidas no Capítulo 4.

Neste contexto, Kim e Gil (2007) revelam que a transferência do conhecimento relacionado a soluções de problemas complexos produzidos por especialistas humanos para os sistemas computacionais tem se demonstrado uma tarefa bem desafiadora, e ao longo das duas últimas décadas diversas abordagens têm sido propostas para a aquisição interativa de conhecimento.

O foco da aquisição do conhecimento neste trabalho concentra-se na coleta de conhecimentos factuais tais como: (a) casos de soluções de problemas de não-conformidade, e (b) conceitos, relações e instâncias relacionadas a uma ontologia que representa o conhecimento no domínio de análise de modos de falha e efeitos. São consideradas nesta tese as abordagens propostas por Ericksson et al.(1995), Fernández-Lopéz et al. (1999), Denny (2011) e Sure et al.(2002).

Assim, em termos metodológicos, o problema de implementação de bases de conhecimento envolve uma função de transformação de uma descrição no nível do conhecimento (*knowledge-level description*) em uma descrição no nível simbólico (*symbol level description*), de modo a permitir seu compartilhamento e reuso entre pessoas e agentes computacionais.

Deve-se observar que, nesta tese, as bases de conhecimento dos agentes de recursos foram preenchidas ou instanciadas com conhecimento fundamentado em casos de testes validados por especialistas no domínio e obtidos a partir de duas fontes distintas: a primeira baseada em experiências relatadas na literatura relevante e a outra baseada em um trabalho de campo realizado em uma empresa multinacional situada no sul do Brasil que é responsável pela fabricação por extrusão de perfis de alumínio para fabricação de caixilhos residenciais e comerciais para o mercado de construção civil. Além disso, esta empresa produz perfis e

peças acabadas para o mercado industrial, que inclui os segmentos de autopeças, bens de consumo, indústria elétrica, transportes, máquinas e equipamentos.

6.1 IMPLEMENTAÇÃO DAS BASES DE CONHECIMENTO DOS AGENTES RBC

Na literatura, em geral, as experiências relacionadas à solução de problemas de não-conformidades em processos de manufatura são relatadas na forma de casos, os quais juntam lições a serem aprendidas envolvendo: a descrição do estado do processo quando ocorreu a não-conformidade, uma solução sugerida, e as expectativas em relação ao estado do processo quando da adoção desta solução sugerida.

A estratégia adotada na etapa de implementação das bases de conhecimento dos agentes de recursos RBC é caracterizada pela função de transformação de uma descrição no nível do conhecimento, como mostra o diagrama IDEF0 em nível macro na Figura 6.1.

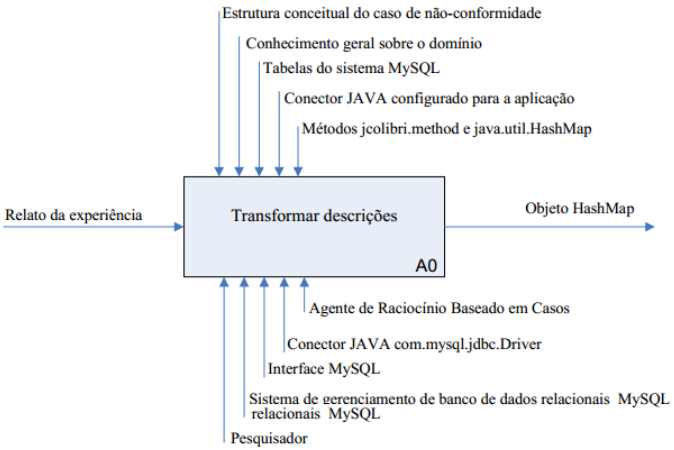

Figura 6-1 - Diagrama IDEF0: A0 – Função de transformação das descrições.
Fonte: Mikos, 2009.

Esta função tem por objetivo transformar as experiências relatadas na forma de linguagem natural em uma descrição no nível simbólico capaz de ser manipulada pelos métodos implementados como comportamentos dos agentes.

Esta estratégia tem como fundamento uma cadeia de mapeamentos unívocos, os quais têm início com os elementos factuais contidos nos relatos das experiências de soluções de problemas de não-conformidades, como mostra o desdobramento do diagrama IDEF0 da Figura 6.2.

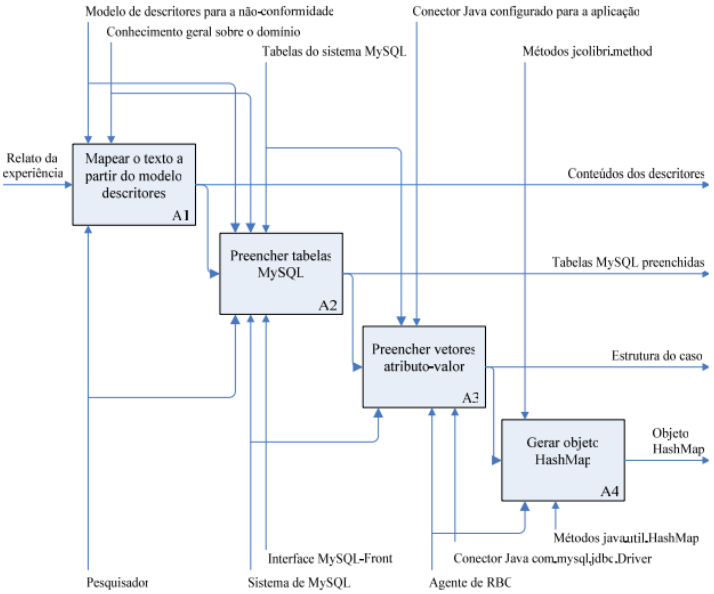

Figura 6-2 - Diagrama IDEF0 com o desdobramento da função transformação.
Fonte: Mikos, 2009.

É importante destacar que a estratégia adotada nesta etapa usa a técnica de aquisição de conhecimento tipo *top-down* baseada em modelos apresentada por van Heijst et al. (1997), na qual a estrutura conceitual do caso de não-conformidade pode ser usada para guiar o processo de aquisição do conhecimento relevante, indicando quais elementos necessitam ser obtidos das fontes disponíveis.

A Figura 6.3 ilustra a execução da atividade A2 ("*Fill out MySQL tables*"), que tem como entrada o conteúdo dos descritores resultantes da atividade A1. Tais descritores são transferidos diretamente para suas colunas correspondentes na tabela do sistema de gerenciamento de banco de dados MySQL.

A Figura 6.3 mostra a atividade A3 ("*Fill out AttributeValue vectors*"), que é executada automaticamente quando o agente está ativo na sociedade dos agentes mediante o conector *com.mysql.jdbc.Driver*

previamente configurado na especialização do *framework* jCOLIBRI. Este conector é responsável pela dinâmica do mapeamento dos parâmetros dos descritores pré-definidos, de acordo a estrutura conceitual do caso e as respectivas colunas das tabelas no MySQL.

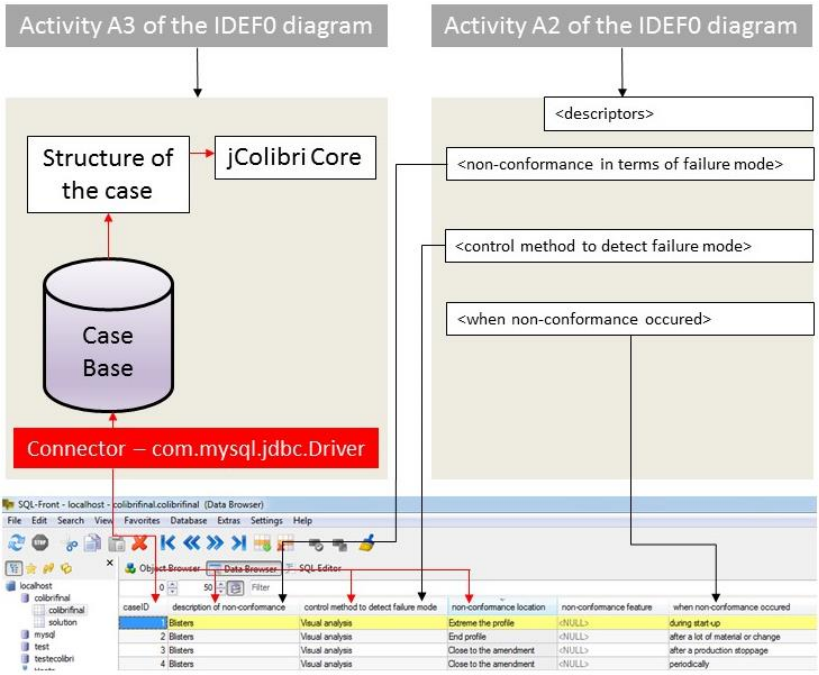

Figura 6-3 - Detalhes das atividades A2 e A3 da transformação de descrições de funções.
Fonte: Criado pelo Autor.

A atividade A4 ("*Generate HashMap object*"), mostrada na Figura 6.3, é executada automaticamente pelos agentes de recursos CBR que estão ativos na sociedade de agentes mediante os métodos implementados nos comportamentos dos agentes. Estes objetos *HashMap* são responsáveis pelas interações sociais entre os agentes CBR e o agente solicitante.

6.1.1 Implementação do componente terminológico TBox

Em primeiro lugar, foi implementado o componente terminológico TBox (*terminological component*), ou definições de classes OWL-DL. Como descrito anteriormente, este componente representa o conhecimento sobre as características dos conceitos da ontologia PFMEA (*intensional knowledge*), os quais são independentes do domínio de aplicação. Estes conceitos compreendem, por sua vez, um conjunto de axiomas terminológicos que são usados para definir os conceitos mais gerais a partir de outros conceitos e papéis primitivos, como descrito em Baader e Nutt. (2003).

O componente TBox foi implementado a partir das definições de classes OWL-DL onde, segundo Horridge et al. (2004), cada classe pode ser interpretada como um conjunto que contém indivíduos, sendo descrita de modo formal, isto é, estabelecendo-se precisamente por meio da álgebra relacional quais condições um indivíduo deve satisfazer para ser membro desta classe. Estas classes OWL-DL são organizadas dentro de uma hierarquia de superclasses e subclasses denominada de taxonomia, onde as subclasses representam uma especialização da superclasse, como mostra a Figura 6.4.

A interface do editor de ontologias PROTÉGÉ OWL-DL, mostrada na Figura 6.4, ilustra em particular a aba OWLClasses, na qual se observa a taxonomia de classes construída a partir da superclasse OWL:Thing (*subclass explorer*) e as definições das respectivas subclasses considerando para que os sete eixos formalizados na ontologia PFMEA a saber: produto (*ProductConcepts*), processo (*ProcessConcepts*), funções (*FunctionConcepts*), falhas (*FailureConcepts*), ações (*ActionsConcepts*), descrição (*FMEADescription*) e imagens (*ImagesConcepts*).

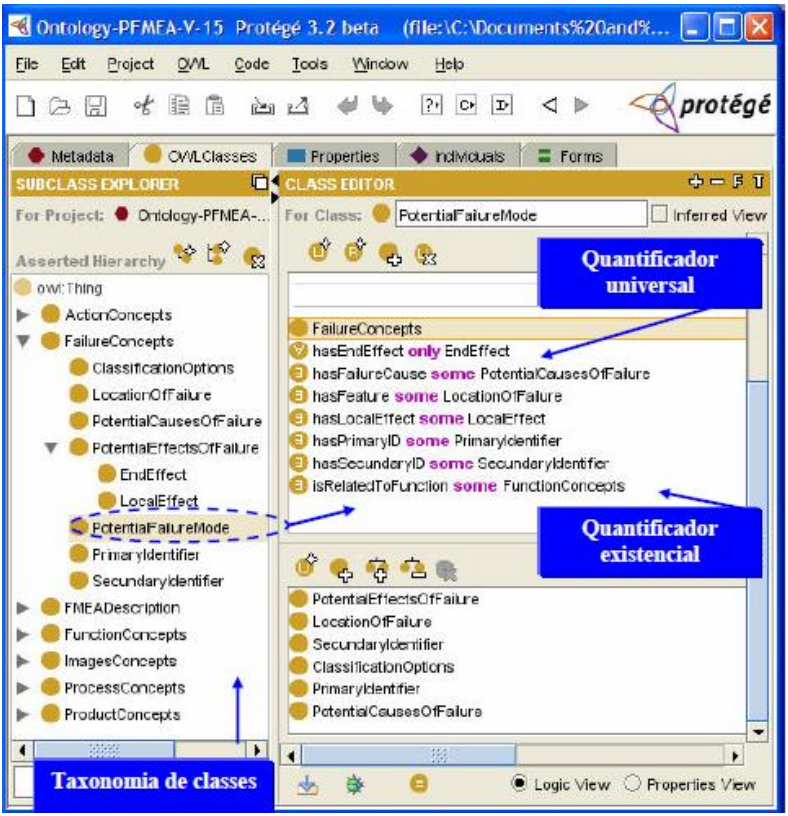

Figura 6-4 - Taxonomia de classes OWL-DL.
Fonte: Mikos, 2009.

De acordo com Horridge et al. (2004), uma das características chave das definições de superclasses-subclasses OWL-DL são as relações de inserção de conceitos para um dado TBox (*concept subsumption*), as quais podem ser computadas automaticamente por sistemas de raciocínio, onde tal análise consiste basicamente em verificar se um conceito particular está sob dependência de outro mais geral mediante o estabelecimento de uma hierarquia de conceitos determinada pela comparação de suas definições. Há também a classificação do TBox (*classify*), que corresponde à inserção automática de um conceito em uma hierarquia, conectando-o, apropriadamente, entre o conceito mais específico do qual ele é dependente (*parent concept*) e ao mais geral dele dependente (*children concept*)

Adicionalmente, a Figura 6.4 mostra a área destinada à definição das classes (*class editor*), a qual mostra a aplicação das restrições em propriedades disponíveis na linguagem OWL-DL, usadas para descrever de maneira precisa a subclasse "*PotentialFailureMode*".

É importante observar que a aplicação das restrições em propriedades através do quantificador universal (∀)[24], mostrado na Equação (3), é análoga à da lógica de predicados, e consiste em descrever uma classe anônima que restringe um grupo de indivíduos da subclasse "*PotentialFailureMode*" que satisfaça esta restrição.

∀hasEndEffect only EndEffect (3)

Assim, todos os indivíduos definidos na classe "PotentialFailureMode" que usam a propriedade "*hasEndEffect*" devem ter como valores associados a esta propriedade indivíduos da subclasse "*EndEffect*".

Entretanto, em termos lógicos, isto não significa necessariamente que um modo de falha em potencial tenha um ou mais efeitos finais mas, se for o caso, estes efeitos devem ser da classe "*EndEffect*", pois um efeito final no método PFMEA representa o impacto de um modo de falha no nível mais alto do processo. Portanto, tal efeito deve ser avaliado a partir da análise dos efeitos locais de todos os níveis intermediários, podendo ser resultante inclusive de múltiplos modos de falha.

Por sua vez, a aplicação do quantificador existencial (∃)[25], mostrado na equação (4), é também análoga à da lógica de predicados, e descreve uma classe anônima que restringe um grupo de indivíduos da subclasse "*PotentialFailureMode*" para os quais existe pelo menos um indivíduo da classe "*OperationFunction*" relacionados pela propriedade "*isRelatedToFunction*".

∃ isRelatedToFunction some OperationFunction (4)

Em termos lógicos, isto significa que todos os indivíduos definidos na classe "*PotentialFailureMode*" estão relacionados a pelo menos um indivíduo da subclasse "*OperationFunction*" (referente à função da operação) pela propriedade "*isRelatedToFunction*", pois no método

[24] Pode ser lido como "somente" ou ainda como "*allValuesFrom*" na terminologia OWL-DL.
[25] Pode ser lido como "ao menos um" ou ainda como "*someValueFrom*" na terminologia OWL-DL.

PFMEA um modo de falha pode ser deduzido a partir de parâmetros funcionais típicos da operação.

Adicionalmente, a propriedade "*isRelatedToFunction*" foi modelada como uma propriedade inversa da propriedade "*hasFailureMode*". Deste modo o sistema de raciocínio pode inferir automaticamente que um indivíduo da subclasse "*OperationFunction*" está relacionado ao indivíduo da subclasse "*PotentialFailureMode*" usando a propriedade "*hasFailureMode*", como mostra a Figura 6.5.

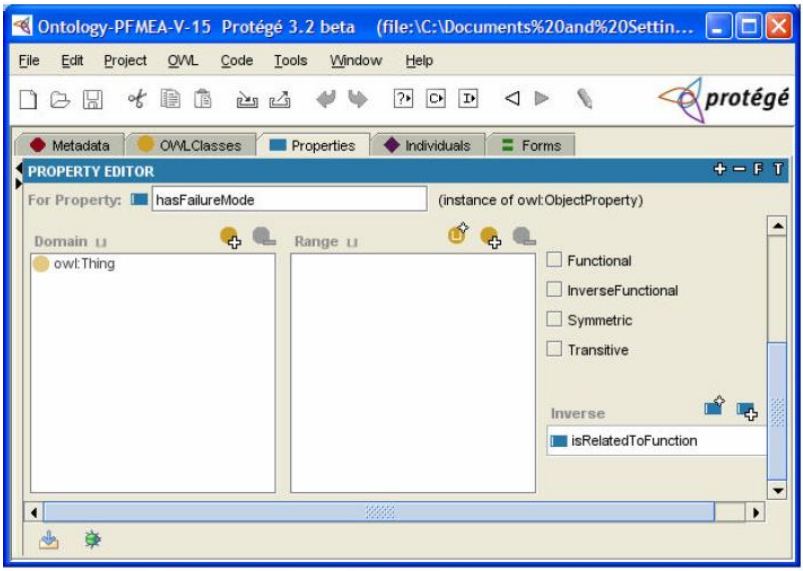

Figura 6-5 - Modelagem de uma propriedade inversa.
Fonte: Mikos, 2009.

Por outro lado, observa-se que a propriedade "*isRelatedToItem*", mostrada na Figura 6.6, não foi modelada usando-se os quantificadores para restrições da subclasse "*PotentialFailureMode"*, mas alternativamente pela especificação do domínio (*domain*) e seus valores (*range*). Em termos lógicos, isto significa que a propriedade "*isRelatedToItem*" tem como domínio as subclasses "*PotentialFailureMode*", "*PotentialCausesOfFailure*", "*RecommendedAction*" e "*CurrentCondition*". Ou seja, esta propriedade estabelece a vinculação dos indivíduos destas classes necessariamente com os indivíduos da subclasse "*SystemLevel*".

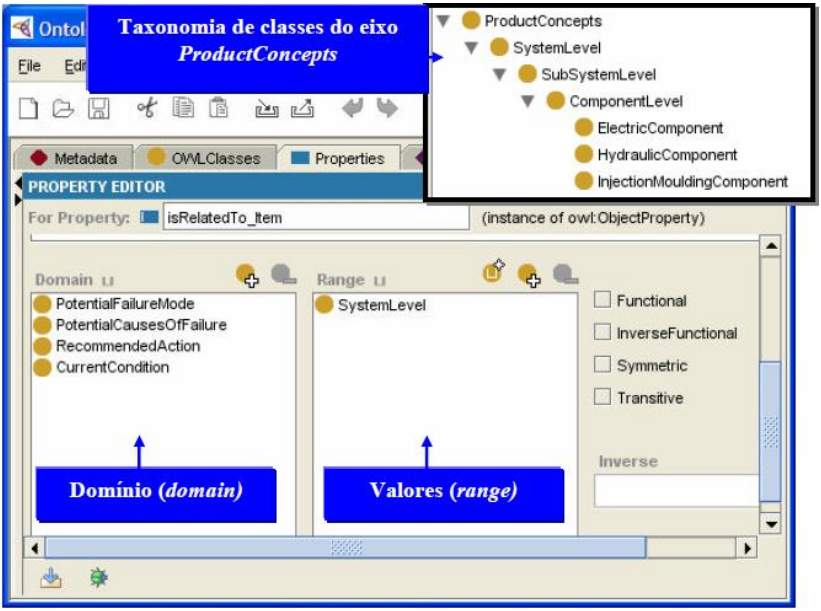

Figura 6-6 - Modelagem usando *domain* e *range*.
Fonte: Mikos, 2009.

6.1.2 Implementação de asserção ABox

Em segundo lugar, foi implementado o componente de asserção ABox (*assertional component*), ou definição de indivíduos OWL-DL. Como mencionado anteriormente, este componente representa o conhecimento extensivo, que especifica os indivíduos de um domínio em particular, representando, portanto, a instanciação da estrutura de conceitos modelada pela definição de classes OWL-DL, como mostra a Figura 6.7.

A interface do editor PROTÉGÉ OWL-DL apresentada nas figuras 6.7 mostra a aba correspondente ao editor de indivíduos (individual editor), cuja configuração decorre diretamente das descrições das classes modeladas pelas definições de classes OWL-DL.

Estas figuras mostram a definição de um indivíduo (ou instância) da subclasse "*PotentialFailureMode*", neste caso o indivíduo "*Dirty_container*", mediante a sua associação aos indivíduos de outras

classes de acordo com a modelagem realizada pela definição das classes OWL-DL discutida anteriormente e mostrada nas Figuras 6.4, 6.5 e 6.6.

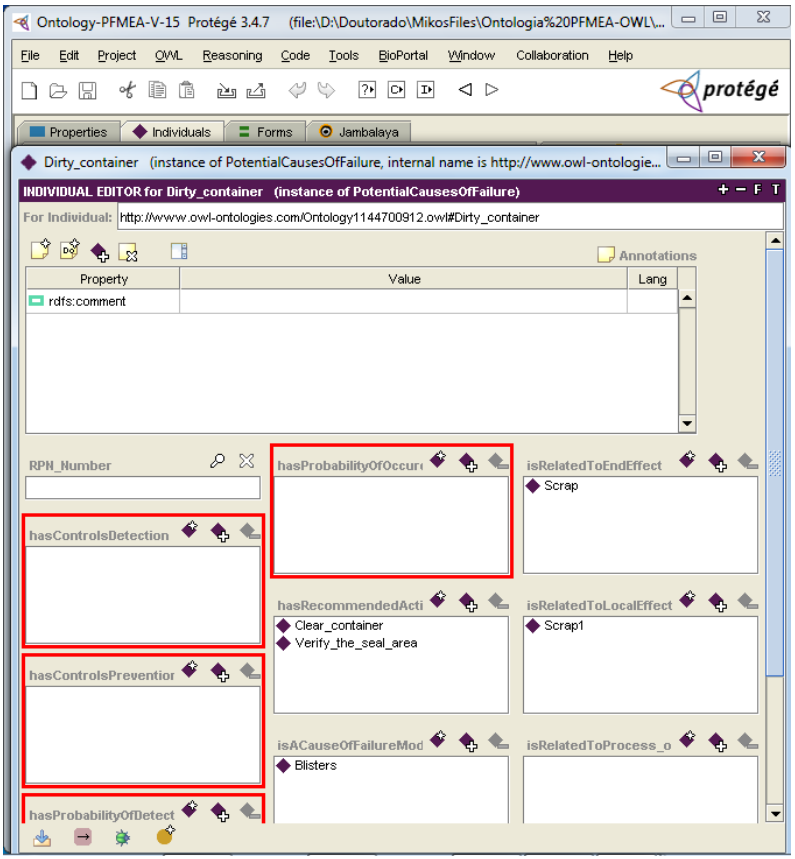

Figura 6-7 - Especificação de indivíduos da subclasse OWL-DL.
Fonte: Criado pelo Autor.

6.2 Conclusões do capítulo

Este capítulo buscou apresentar a pesquisa desenvolvida nesta tese no tocante à forma de implementação e de operacionalização das bases de conhecimento dos agentes previsto no modelo proposto, de acordo com os objetivos propostos neste trabalho.

Neste sentido, este capítulo foi dividido em duas subseções principais. A primeira concentrou-se na apresentação e discussão da implementação das bases de conhecimento dos agentes de recursos RBC, e a segunda subseção concentrou-se nas bases dos agentes de recursos PFMEA.

Na primeira subseção a implementação das bases de conhecimento dos agentes de recursos RBC foi abordada a partir da função de transformação das descrições no nível do conhecimento, cujo objetivo é permitir a transformação das experiências relatadas na forma de linguagem natural em descrições no nível simbólico, as quais podem ser manipuladas pelos métodos computacionais implementados como comportamentos dos agentes.

Por outro lado, na segunda subseção a implementação das bases de conhecimento dos agentes de recursos PFMEA envolveu dois aspectos principais. O primeiro aspecto compreendeu o processo de codificação da ontologia PFMEA mediante a linguagem OWL-DL, e o segundo a função de transformação das descrições no nível do conhecimento factual, isto é, decorrentes da aplicação do método de Análise do Modo de Falha e Efeitos (FMEA) em descrições no nível simbólico envolvendo os conceitos, relações binárias e instâncias da ontologia PFMEA, a qual visa permitir o compartilhamento e reuso deste conhecimento entre pessoas, agentes de interface e agentes de recursos PFMEA.

7 PROCESSO DE VERIFICAÇÃO E VALIDAÇÃO DO MODELO PROPOSTO

No âmbito da perspectiva metodológica, o processo de verificação e validação representa um aspecto fundamental do desenvolvimento de um modelo baseado em agentes. Portanto, este capítulo é devotado à discussão dos aspectos relacionados à verificação, validação conceitual e operacional do modelo proposto considerando as especificações de projeto tal como se acham estabelecidas no Capítulo 4 e os procedimentos metodológicos apresentados no Capítulo 3.

Nesta linha, a preocupação com a verificação do modelo encontra-se associada à necessidade de se avaliar a viabilidade e a credibilidade geral do modelo que, de acordo com Yilmaz (2006), devem aplicar-se nas especificações do projeto e seu respectivo desenvolvimento computacional. Deste modo, a verificação tem por finalidade assegurar que o modelo tenha sido corretamente implementado a partir das especificações de projeto, ressaltando os processos sociais dos agentes e a consistência da organização multiagentes.

Com esta finalidade são apresentadas as funcionalidades desenvolvidas no protótipo do modelo, buscando-se simultaneamente realizar experimentos destinados à verificação do modelo proposto tendo como base os conjuntos de casos de teste da literatura operacionalizados nas respectivas bases de conhecimento dos agentes, conforme apresentado no Capítulo 6.

Em seguida, busca-se discutir a validade conceitual do modelo considerando-se os conjuntos de casos de teste, mas visando, tal como proposto por Yilmaz (2006), avaliar a extensão na qual o conhecimento social representado no modelo conceitual reflete os construtos elaborados pelos especialistas no domínio. Além disso, busca-se avaliar mediante experimentos a fidedignidade das respostas dos serviços de raciocínio e dos métodos de recuperação de conhecimento encapsulados no comportamento dos agentes de recursos em relação aos casos de testes fornecidos.

Por fim, procura-se discutir a validade operacional do modelo seguindo a linha de Yilmaz (2006), avaliando-se a relevância do comportamento colaborativo do modelo com respeito ao seu propósito geral, a partir de experimentos realizados com casos obtidos pela pesquisa de campo adicionalmente aos casos obtidos na literatura.

7.1 VERIFICAÇÃO DAS FUNCIONALIDADES DO PROTÓTIPO DO MODELO

O protótipo do modelo dispõe de uma janela gráfica do usuário, que foi desenvolvida computacionalmente na forma de um *Java Applet*. Esta interface é apresentada no momento em que os agentes de interface são inicializados a pedido de um usuário em particular. Ela tem como objetivo estabelecer a perspectiva ou cenário de apoio à solução de problemas de não-conformidades em função da necessidade do usuário, como mostra a Figura 7.1.

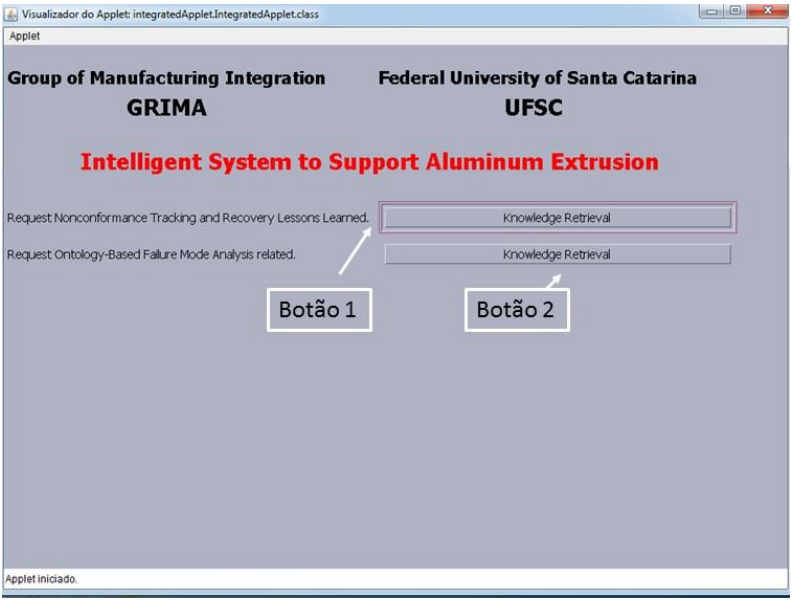

Figura 7-1 - Primeira janela de interface gráfica do protótipo do modelo.
Fonte: Criado pelo Autor.

Esta janela da interface gráfica compreende dois botões que dizem respeito à perspectiva de apoio que o usuário deseja: a primeira concentra-se em experiências de soluções de problemas de não-conformidades que já ocorreram no passado, enquanto a segunda representa problemas com não-conformidades potenciais que não ocorreram, mas que foram identificadas pelo método de Análises de Modos de Falhas e Efeitos.

Deste modo, quando um usuário aciona um dos botões correspondentes às perspectivas de apoio à solução de problemas de não-

conformidades (Botão 1 ou 2), um pedido é enviado diretamente à instância do agente de interface apropriado, o qual por sua vez apresenta ao usuário uma segunda janela da interface gráfica destinada à configuração da consulta.

É importante observar que os agentes de interface são executados a pedido de um usuário em particular, e que estes, após cumprirem as suas tarefas agindo em nome no usuário no âmbito da plataforma, podem ter sua execução encerrada ou mesmo retirados da plataforma.

Tais agentes diferem-se dos agentes de recursos, os quais devem estar sempre ativos aguardando a submissão de uma consulta por meio do sistema de troca de mensagens FIPA-ACL com um ou mais agentes de interface.

Como já revelou o estudo do arcabouço JADE, o suporte à execução dos agentes de interface e de recursos se dá mediante as funcionalidades pré-programadas na Plataforma de Agentes Distribuída (BELLIFEMINE et al., 2012). Estas funcionalidades permitem que os agentes de interface possam ser executados em qualquer servidor separado geograficamente dos servidores usados pelo demais agentes da plataforma, lembrando que todos os servidores (*hosts*) devem permitir a Invocação de Métodos Remotos (RMI - *Remote Method Invocation*) através de uma rede.

Esta funcionalidade pré-programada foi adotada no desenvolvimento geral do protótipo do modelo proposto por meio do uso de abstração de *containers* ou unidades de distribuição da plataforma representando os diversos *hosts* distribuídos. Portanto, este aspecto do protótipo não necessita de verificação, pois já foi amplamente verificado pela comunidade científica da área e desenvolvedores de aplicações voltadas para esta plataforma.

A execução dos agentes propostos neste trabalho de tese é mostrada na Figura 7.2 pela interface gráfica do Agente de Gerenciamento Remoto (*RMA - Remote Management Agent*) disponível na plataforma JADE. Este agente é responsável pelo controle dos estados do ciclo de vida de todos agentes em execução inclusive os distribuídos, no qual pode-se observar os diversos *containers* ativos e os respectivos agentes em execução.

No *container* principal (*Main-Container*) encontra-se em execução, além do próprio Agente RMA, o Sistema Gerenciador de Agentes (ou *Agent Management System* - AMS) que supervisiona o acesso e o uso da plataforma. A Figura 7.2 mostra, adicionalmente, o Agente Facilitador de Diretórios (DF Agent) que, no protótipo em questão, é responsável pelas tarefas do Agente *Matchmaker*.

Os *containers* de 1 a 3 (Figura 7.2) mostram a execução de três instâncias do agente da classe *PFMEA*, denominados *PFMEADLAgent0*, *PFMEADLAgent1* e *PFMEADLAgent2*; cada um com capacidade de acessar a sua própria base de conhecimento. No entanto, apesar do protótipo em questão considerar somente três instâncias desta classe, deve-se ressaltar que outras podem ser incluídas a qualquer momento ou mesmo retiradas de acordo com as necessidades organizacionais.

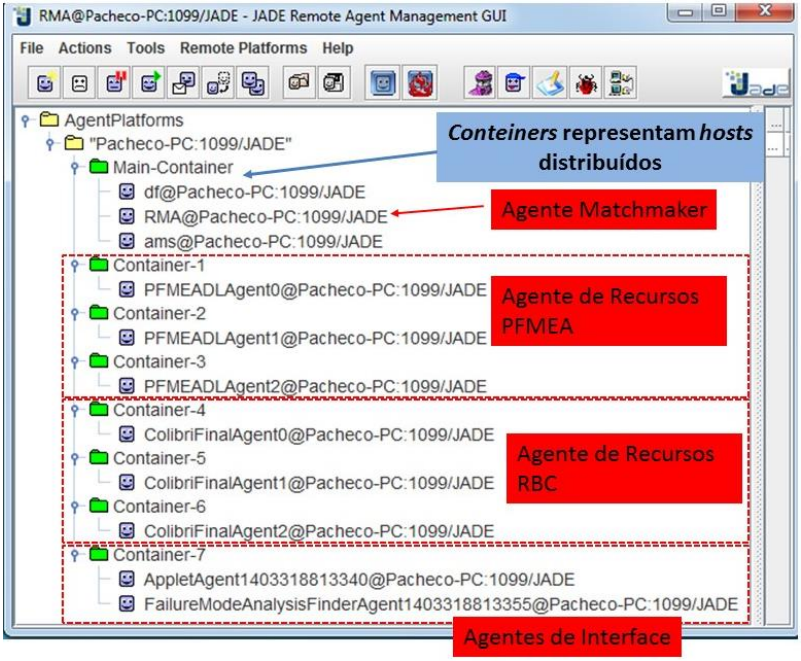

Figura 7-2 - Interface gráfica do Agente RMA registrando os diversos containers e agentes.
Fonte: Criado pelo Autor.

Da mesma forma, os *containers* de 4 a 6 (Figura 7.2) mostram a execução de três instâncias do agente recurso RBC denominados *ColibriFinalAgent0*, *ColibriFinalAgent1* e *ColibriFinalAgent2*, cada um com capacidade de acessar a sua própria base de conhecimento.

Por fim, a Figura 7.3 mostra os agentes de interface para RBC e os agentes de interface para PFMEA, que se encontram em execução no *container* 7 com o nome *AppletAgent1* e *FailureModeAnalysisFinderAgent1*, respectivamente.

Cumpre observar que a interface gráfica do Agente RMA dispõe ainda de outras funcionalidades descritas detalhadamente em (BELLIFEMINE et al., 2012).

Outro ponto a ser explorado diz respeito ao Agente *Matchmaker*, que na plataforma é representado pelo Agente DF, cujo papel é modelado pelo diagrama de sequência AUML mostrado na Figura 7.3. O objetivo das interações é permitir aos agentes de interface o acesso ao conjunto de meta-informações relacionadas aos agentes de recursos ativos na plataforma, disponíveis no serviço de "páginas amarelas" do agente DF (*Matchmaker*), pois deste modo os agentes de interface podem estabelecer interações diretas com os agentes ativos usando estas meta-informações para controle.

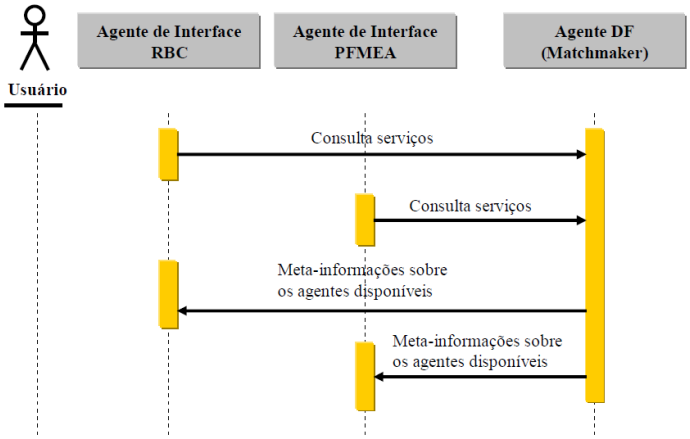

Figura 7-3 - Diagrama de sequência AUML agente de interface / agente DF.
Fonte: Criado pelo Autor.

7.1.1 Verificação das funcionalidades do protótipo do modelo relacionada aos agentes RBC

A perspectiva de apoio à solução de não-conformidades a partir da recuperação de conhecimento oriundos de experiências de soluções de problemas de não-conformidades que já ocorreram no passado é determinada pela ação do Botão 1 (Figura 7.1), que apresenta ao usuário a janela gráfica do agente de interface RBC, como mostra a Figura 7.9.

Na Figura 7.8 a janela gráfica do agente de interface RBC já está configurada para o processo de extrusão de alumínio sob consulta na *Java Applet*.

A Figura 7.4 apresenta o diagrama de sequência AUML que modela a interação entre o agente de interface RBC e os agentes de recursos RBC ativos na plataforma e já identificados por meio da interação com o agente DF.

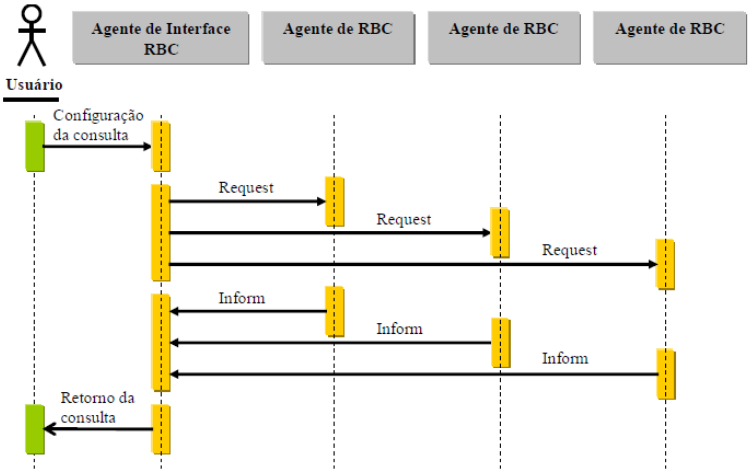

Figura 7-4 - Diagrama de sequência AUML.
Fonte: Criado pelo Autor.

Como mostra o diagrama de sequência, a consulta configurada pelo usuário é submetida aos agentes de recursos pelo agente de interface RBC, o qual, primeiramente, converte a consulta em dois objetos *HashMap QueryValues* e *WeightValues*. Em seguida ele os envia aos agentes de recursos RBC como conteúdo de uma mensagem FIPA-ACL usando o ato comunicativo *Request*, de acordo com o comportamento apresentado e discutido no Capítulo 5.

Então, os agentes de recursos respondem a consulta com mensagens FIPA-ACL usando o ato comunicativo *Inform*, em cujo conteúdo encontra-se o vetor *solutionHashMap*. Este vetor inclui a medida de similaridade global entre os atributos do caso sob consulta e os atributos dos casos mais similares recuperados, bem como os atributos que descrevem a solução sugerida e resultados esperados.

Por fim, o agente de interface converte o vetor *solutionHashMap* e os apresenta de forma compreensível ao usuário por meio da segunda janela gráfica, como mostra a Figura 7.5.

O desenvolvimento desta funcionalidade pode ser observado pela interface do agente *Sniffer*[26], uma das ferramentas disponíveis na plataforma JADE, como mostra a Figura 7.5.

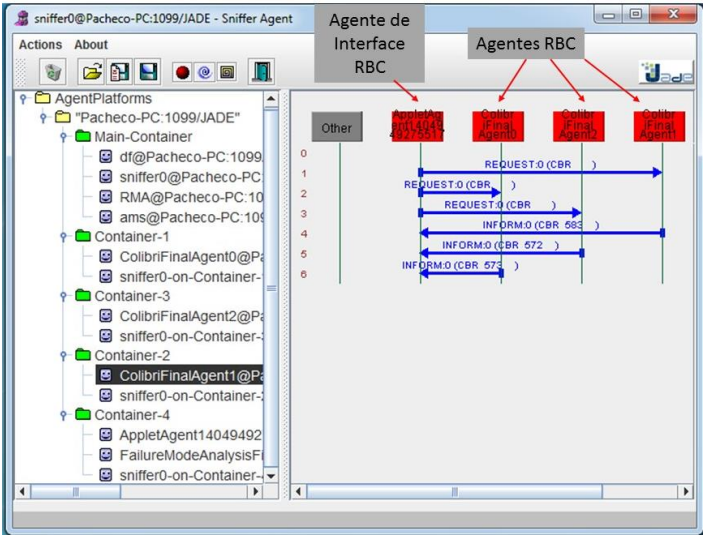

Figura 7-5 - Interface gráfica do Agente *Sniffer*.
Fonte: Criado pelo Autor.

A interface da Figura 7.5 é dedicada à depuração durante o processo de desenvolvimento do protótipo da organização multiagente que, basicamente, permite verificar computacionalmente como ocorre a interação e a troca de mensagens entre os agentes propostos. Assim, nesta figura observa-se as interações e as trocas de mensagens entre o agente de interface RBC denominado *AppletAgent* e os agentes de recursos RBC denominados *ColibrifFinalAgent0*, *ColibrifFinalAgent1*, *ColibrifFinalAgent2*, os quais se encontram ativos na plataforma.

Com relação aos agentes de recursos RBC, deve-se mencionar que a questão essencial a ser verificada diz respeito ao cálculo da medida de similaridade global determinada a partir do valor dos descritores e dos

[26] O agente *sniffer* é responsável pelo rastreamento de todas as trocas de mensagens entre os agentes ativos na plataforma.

pesos associados a cada um destes descritores, os quais são recebidos do agente de interface na forma dos objetos tipo *HashMap QueryValues* e *WeightValues*. Este cálculo determina a ordem na qual os casos, recuperados pelos métodos de raciocínio baseado em casos encapsulados no comportamento dos agentes, devem ser apresentados aos usuários. Esta ordem é apresentada no menu tipo *ComboBox* 1, como mostra a Figura 7.6.

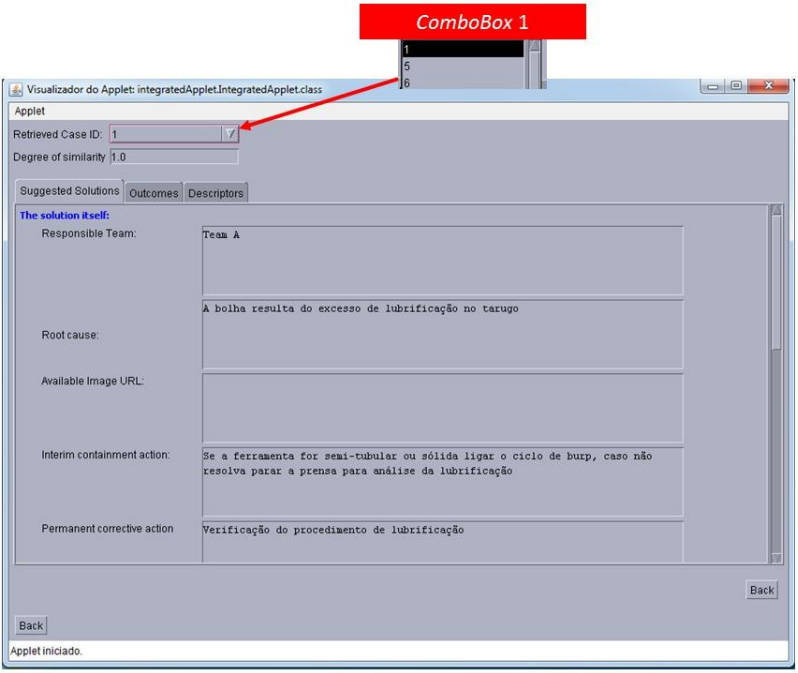

Figura 7-6 - Janela gráfica para apresentação dos casos recuperados.
Fonte: Criado pelo Autor.

Nesta perspectiva, é importante observar a medida de similaridade global que determina o grau de similaridade entre o novo caso (caso sob consulta) e um caso armazenado na base de conhecimento do agente. Esta medida considera todos os descritores modelados, e é calculada pelo algoritmo *nearest neighbour* normalizado, de acordo com a Equação (2) apresentada no capítulo 4.

Nesta equação da similaridade global o peso (w_i) expressa a importância de um único descritor do caso no cálculo geral da similaridade. É importante destacar que neste protótipo o valor do peso

(*wi*) pode ser ajustado dinamicamente a cada nova consulta em função das preferências do usuário.

A Figura 7.7 mostra que o caso 1 é o caso mais similar disponível nas bases de conhecimento dos agentes de recurso RBC ativos, considerando os descritores enviados pelo agente de interface, e este caso obteve uma medida de similaridade de 1,0, isto é, 100% de coincidência entre os seus descritores e os descritores da consulta configurada pelo usuário.

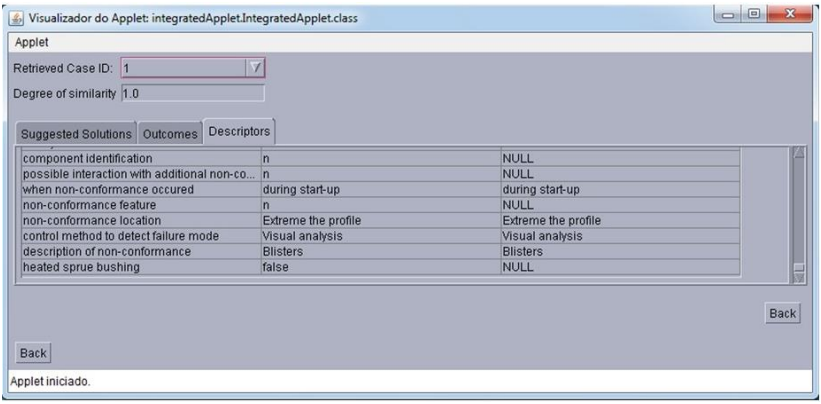

Figura 7-7 - Verificação da medida de similaridade para descritores idênticos.
Fonte: Criado pelo Autor.

Por fim, após a troca de mensagens o agente de interface RBC apresenta ao usuário um segundo conjunto de janelas gráficas destinadas à visualização dos casos mais similares encontrados pelos agentes de recursos nas diferentes bases de conhecimento, como decorrência dos serviços de recuperação realizados.

A Figura 7.6 mostra a primeira aba correspondente às soluções sugeridas pelo especialista para o caso mais similar obtido pelos métodos de raciocínio baseado em casos encapsulados no comportamento do agente de recurso RBC.

Adicionalmente, com o objetivo de verificar se o algoritmo responsável pelo cálculo da similaridade global foi corretamente implementado no comportamento dos agentes de recursos RBC, uma consulta especial foi configurada. Esta consulta envolveu um conjunto de descritores para o novo caso de não-conformidade exatamente igual ao conjunto de descritores indexadores do caso 1 armazenado na base de

conhecimento de um dos agentes de recursos RBC. Adicionalmente, os pesos correspondentes aos descritores foram ajustados com o valor 1,0.

Submetendo esta consulta pelo agente de interface, espera-se que o agente de recurso em questão, após realizar os cálculos da medida de similaridade local e global, deva recuperar o caso 1 com uma medida de similaridade global com valor igual a 1,0, que mostra a total coincidência entre os descritores, e isso foi o exatamente o que aconteceu, como ilustrado na Figura 7.7.

Em síntese, os experimentos realizados durante o processo de verificação demonstraram que o modelo foi implementado de modo correto do ponto de vista computacional, considerando as especificações de projeto. Especial atenção foi dada ao comportamento dos agentes em relação às medidas de similaridade, bem como em relação às interações e trocas de mensagens entre os agentes propostos.

É importante observar que o modelo proposto comporta outras instâncias dos agentes de interface e de recursos que podem ser destinados a outras etapas do ciclo de vida de um produto. Contudo, no presente protótipo os agentes de interface e de recursos de conhecimento RBC concentram-se no processo de extrusão direta de alumínio.

7.2 Exemplo de execução do protótipo do sistema para RBC

A Figura 7.8 mostra a GUI do agente solicitante, que busca recuperar os casos que são semelhantes à descrição da não-conformidade considerada. Mediante a interface da Figura 7.8 o usuário insere a descrição, classificação e condições da não-conformidade. Com o intuito de auxiliar o usuário a introduzir adequadamente os dados da consulta, as opções disponíveis são apresentados na GUI como menus suspensos. Além disso, o usuário pode livremente ajustar o peso que expressa a importância do descritor, de modo a personalizar o cálculo da medida de similaridade, e um peso padrão para cada descritor é sugerido pelo agente solicitante.

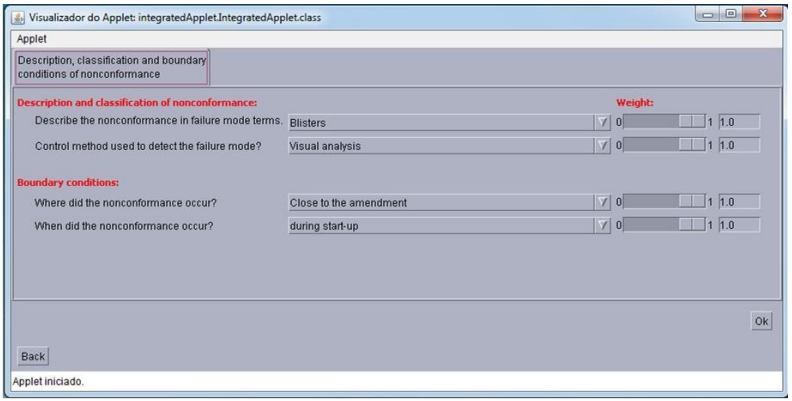

Figura 7-8 - Interface de consulta sobre a não-conformidade "bolha"
Fonte: Criado pelo Autor.

Depois que o usuário completa a entrada dos dados referentes à não-conformidade, o agente solicitante passa à identificação de todos os agentes CBR registrados no agente CBR *Matchmaker* que contêm os casos que são considerados semelhantes à não-conformidade inserida pelo usuário. A comunicação entre os agentes é realizada mediante troca de mensagens. A Figura 7.9 mostra uma sequência de mensagens trocadas entre os agentes instanciados em diferentes containers JADE: *main-container*, *container-1*, *container-2* e *container-3*, que representam diferentes *hosts*. Deve-se mencionar que cada um desses *containers* pode estar localizado em diferentes protocolos de internet (IPs), configurando uma característica de sistema distribuído.

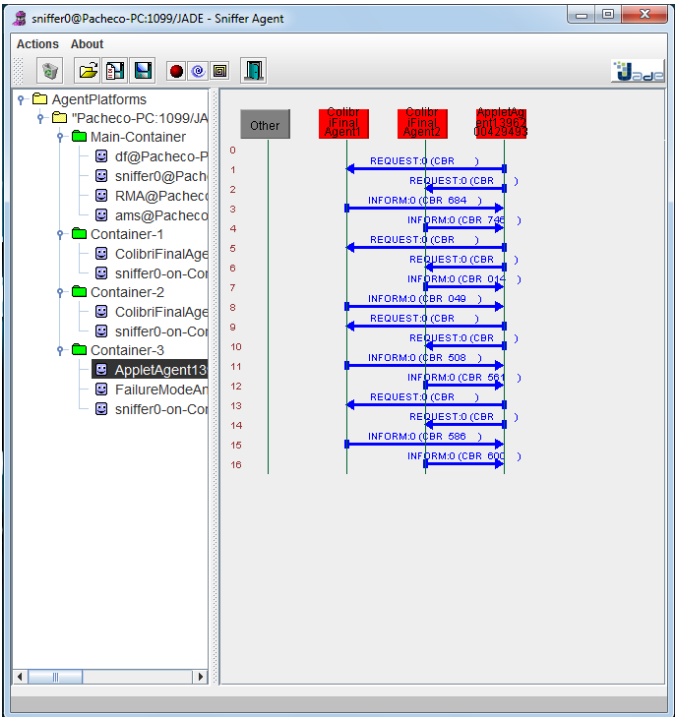

Figura 7-9 - Troca de mensagens entre os agentes - Sniffer Agent em JADE
Fonte: Criado pelo Autor.

A troca de mensagens começa com o pedido, que indica que o agente solicitante (*Applet Agent*) espera que todos os agentes CBR (*ColibriFinalAgents*) executem uma ação de recuperação dos casos na respectiva base de conhecimento que são mais semelhantes à consulta feita pelo usuário. O agente solicitante encapsula o conteúdo das mensagens de um *HashMap Object Java*, mediante o qual os agentes emissor e receptor codificam/decodificam as intenções de comunicação.

Em seguida, cada um dos agentes CBR (*ColibriFinalAgents*) comunica os resultados da ação de recuperação, os quais são apresentados nos painéis das soluções sugeridas e resultados. O motor de inferência do raciocínio baseado em casos (ou intenções do agente) é implementada na plataforma JADE mediante o uso de comportamentos, que são *threads* lógicas de execução que podem ser configuradas de várias maneiras para estabelecer padrões de execução. Os agentes podem ser gerados, iniciados e suspensos em qualquer tempo (Bellifemine et al. 2012).

O agente solicitante, ao receber os casos mais semelhantes enviados pelos agentes CBR, compara-os e organiza-os em uma lista ordenada de acordo com os seus valores de similaridade. Desta forma, na fase de reuso do conhecimento do ciclo CBR, esta lista poderá ser utilizada para dar suporte aos engenheiros e trabalhadores envolvidos no processo de resolução do problema da não-conformidade, de maneira a encontrar rapidamente uma solução adequada para o caso.

A Figura 7.10 ilustra uma solução sugerida para uma dada não-conformidade, e seus descritores de solução com similaridade de 98,88%, enquanto a Figura 7.11 mostra a ação aplicada pelo Team A durante a análise da não-conformidade, que foi ajustar a pressão da área de selagem.

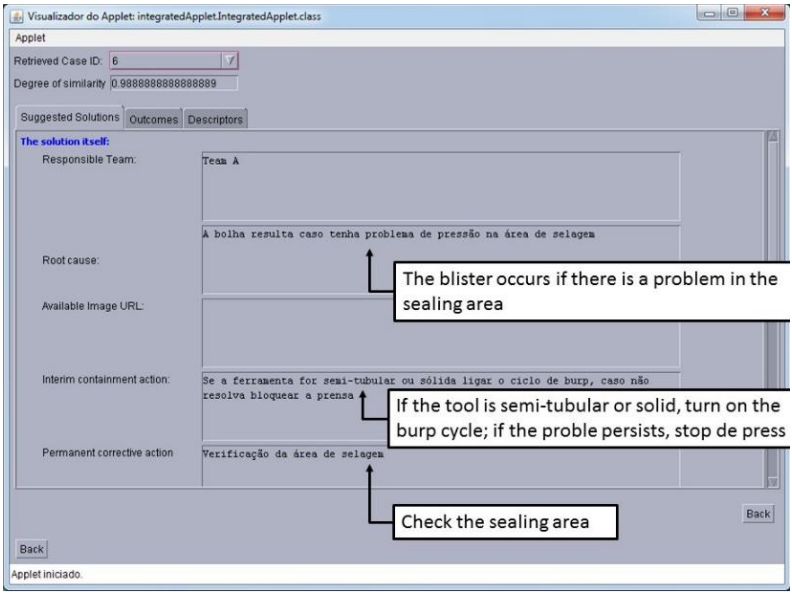

Figura 7-10 - Algumas descrições de uma solução sugerida para "bolhas" - Parte 1
Fonte: Criado pelo Autor.

Ainda na aba de soluções sugeridas (Figuras 7.10 e 7.11), ao mover a tela para baixo, visualiza-se os resultados da aplicação da solução, os quais são mostrados na Figura 7.12. Esses resultados incluem a criação de um período de verificação do ajuste de pressão da área de selagem do recipiente.

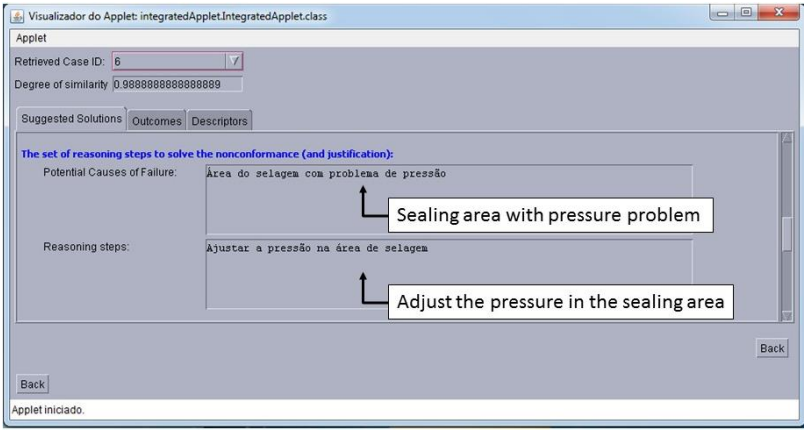

Figura 7-11 - Algumas descrições de uma solução sugerida para "bolhas" - Parte 2
Fonte: Criado pelo Autor.

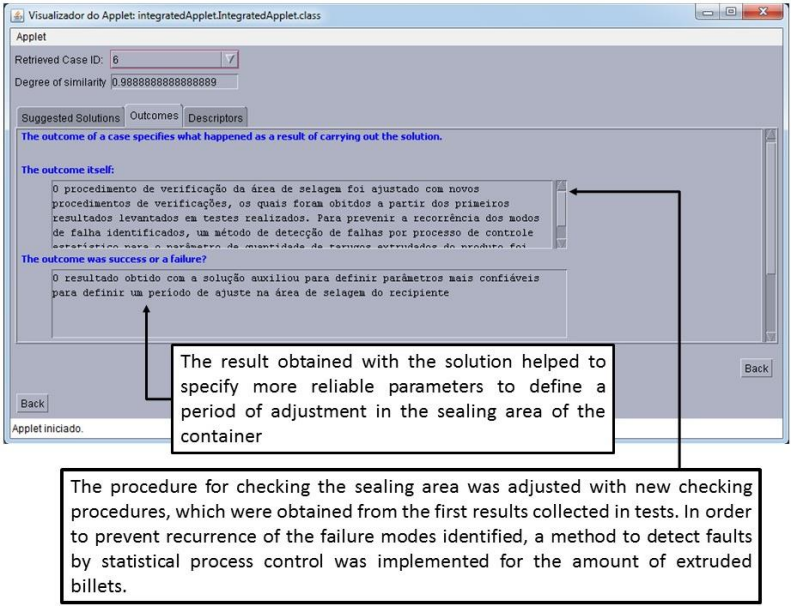

Figura 7-12 - Descritores da solução "bolhas"
Fonte: Criado pelo Autor.

A Figura 7.13 ilustra uma segunda solução, com grau de similaridade de 97,8% resultante da consulta à base de casos referente à não-conformidade “bolha”. A Figura 7.14 mostra a ação efetuada pelo Team A durante a análise da não-conformidade, que foi diminuir a quantidade de lubrificante. Este resultado é mostrado na Figura 7.15, que apresenta uma descrição do resultado obtido.

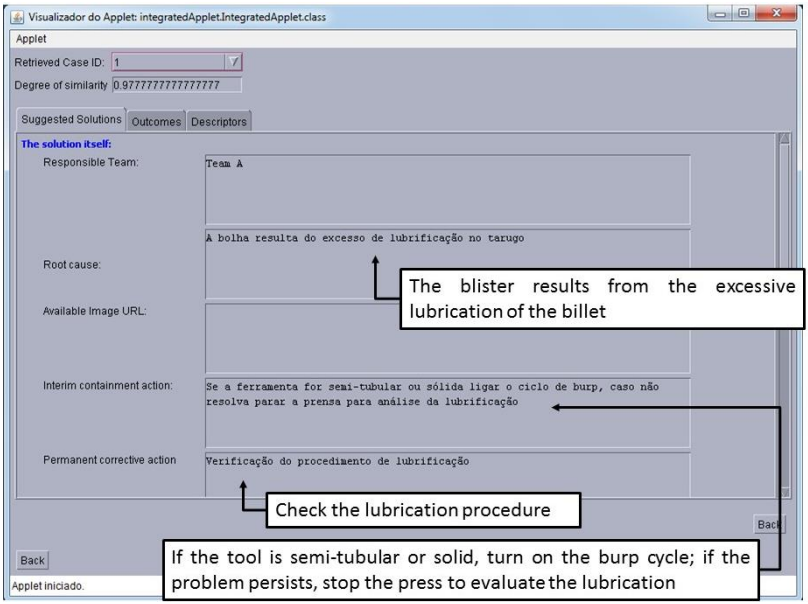

Figura 7-13 - Algumas descrições de outra solução sugerida para "bolhas" - Parte 1
Fonte: Criado pelo Autor.

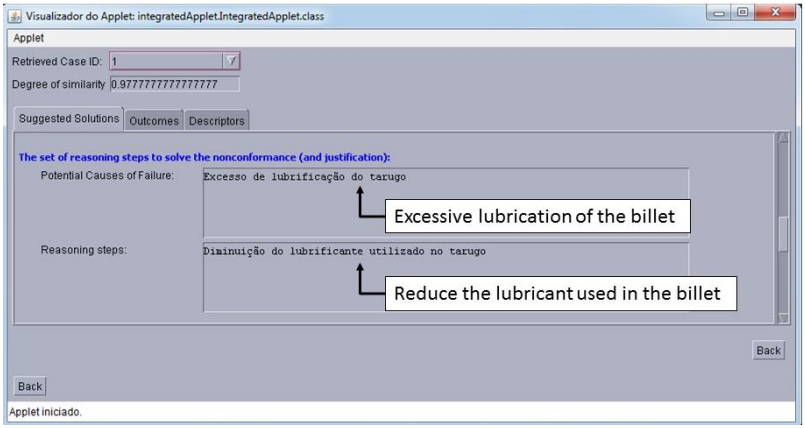

Figura 7-14 - Algumas descrições de outra solução sugerida para "bolhas" - Parte 2
Fonte: Criado pelo Autor.

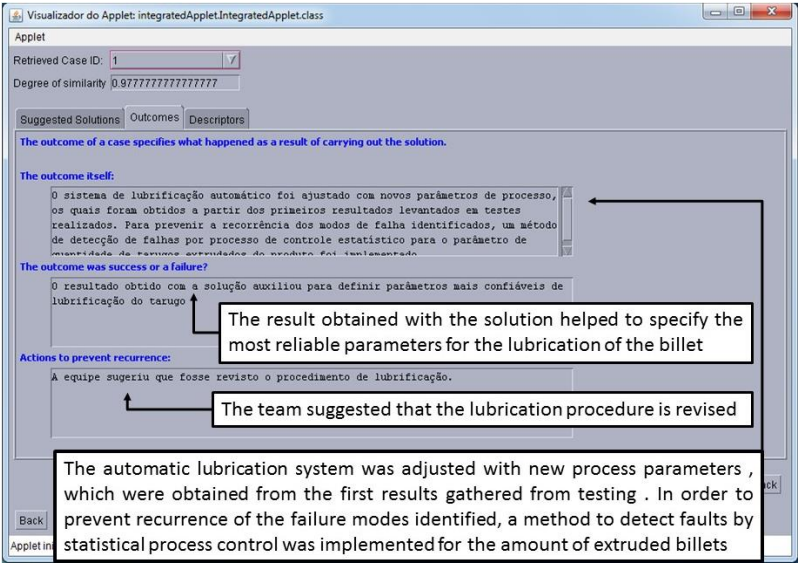

Figura 7-15 - Descritores da solução "bolhas"
Fonte: Criado pelo Autor.

A Figura 7.16 mostra a entrada de dados para a não-conformidade "arrancamento" na interface do agente solicitante. A Figura 7.17 mostra uma solução obtida da base de casos, cuja ação efetuada pelo Team C

durante a análise da não-conformidade “arrancamento” foi não bloquear a prensa e baixar a temperatura de extrusão.

Ainda na aba de soluções sugeridas, mostrada na Figura 7.18, mostra-se a causa potencial de falha, e as etapas para solucionar a não-conformidade. Este resultado é mostrado na Figura 7.19, que ilustra que esta solução inclui uma possível ação preventiva para evitar a recorrência deste modo de falha nos perfis a serem extrudados, que nesse caso consiste na alteração da receita referente a esse perfil extrudado, definindo assim novos parâmetros referentes à temperatura do forno.

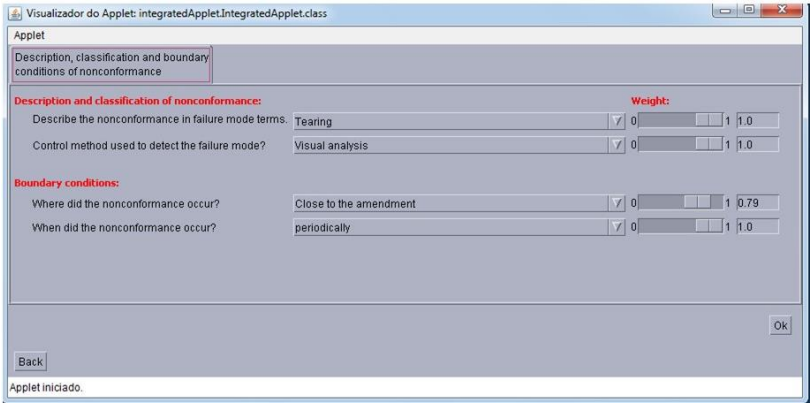

Figura 7-16 - Interface de consulta sobre a não-conformidade "arrancamento".
Fonte: Criado pelo autor

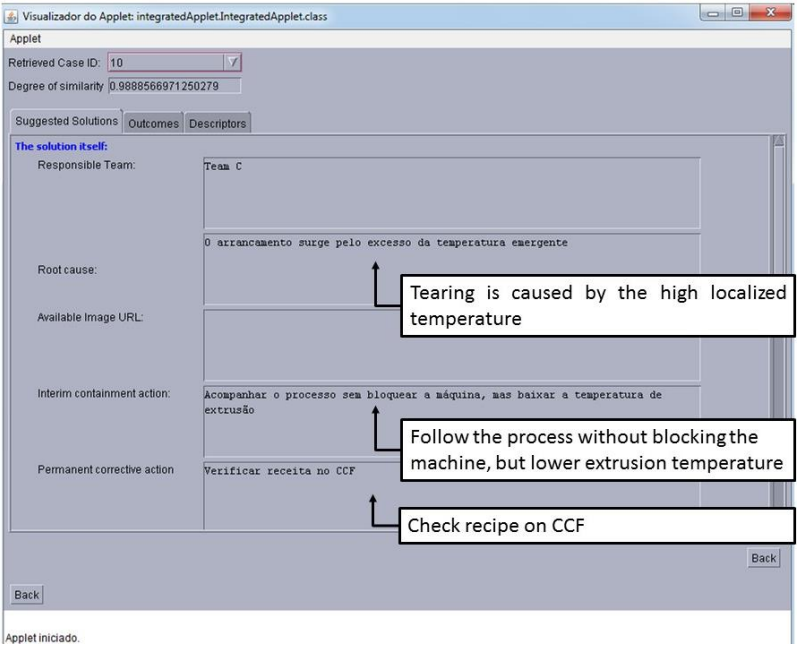

Figura 7-17 - Algumas descrições de outra solução sugerida para "arrancamento" - Parte 1.
Fonte: Criado pelo Autor

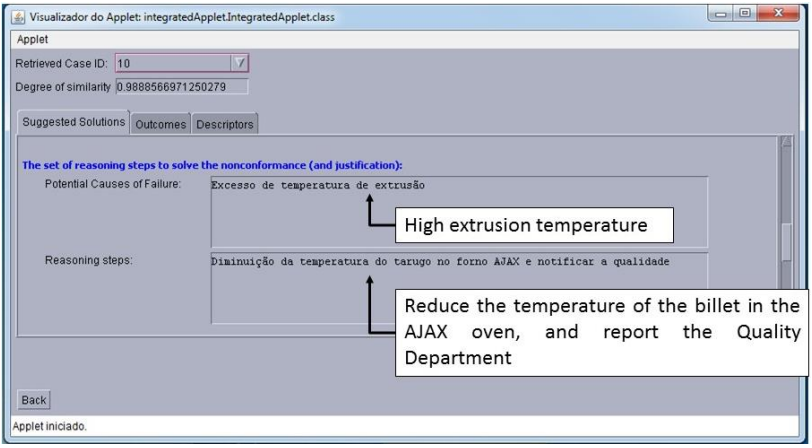

Figura 7-18 - Algumas descrições de uma solução sugerida para "arrancamento" – Parte 2.
Fonte: Criado pelo Autor

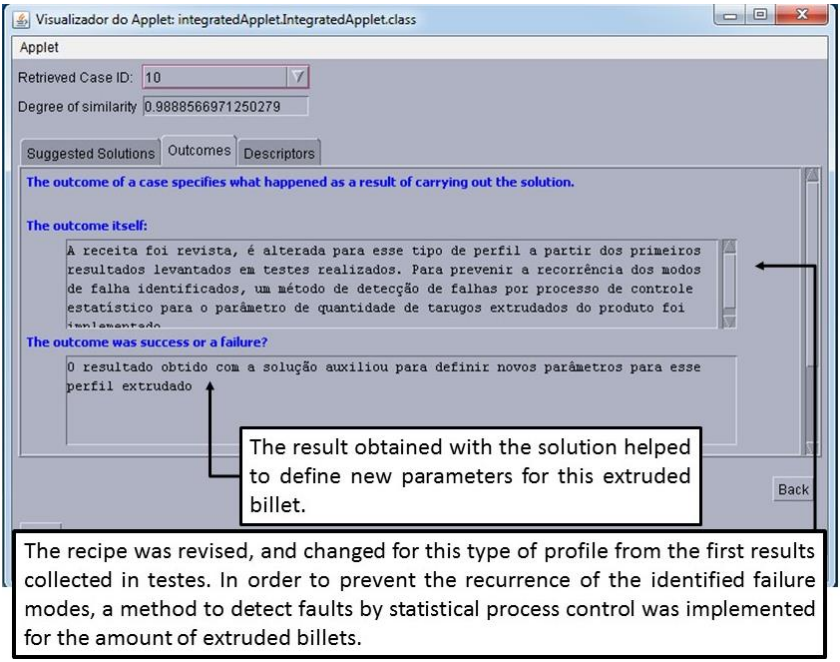

Figura 7-19 - Descritores da solução para "arrancamento".
Fonte: Criado pelo Autor

7.3 Exemplo de execução do protótipo do sistema para PFMEA

A perspectiva de apoio à solução de não-conformidades a partir da recuperação de conhecimento decorrentes da aplicação do método de Análise de Modos de Falha e Efeitos em Processos de Manufatura (PFMEA) é determinada pela ação do Botão 2 (Figura 7.1). Esta ação determina a apresentação ao usuário da janela gráfica do agente de interface PFMEA, como mostra a Figura 7.20.

Esta estratégia baseia-se na configuração dinâmica de uma consulta em linguagem natural, mais precisamente na forma de uma frase na língua inglesa (por exemplo, *Get the list of all ...*), onde o usuário deve delimitar a extensão semântica verbal, determinando o objeto direto e os complementos da frase. Neste sentido, a configuração dinâmica é realizada por meio da escolha em sequência de conceitos da Ontologia *PFMEA DL*, conforme apresentado no Capítulo 4, os quais são disponibilizados ao usuário ao longo da estrutura da frase em diversos menus tipo *ComboBox*, como mostra a Figura 7.20.

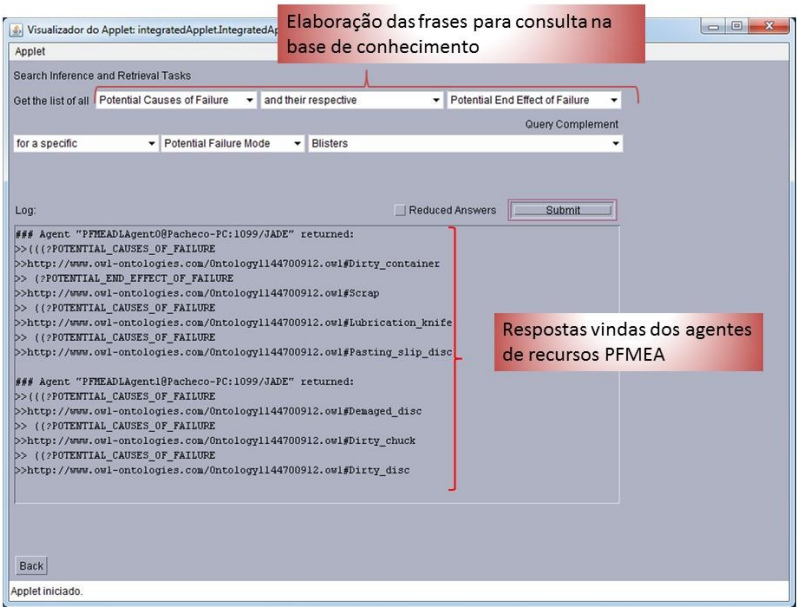

Figura 7-20 - Janela gráfica do agente de interface PFMEA.
Fonte: Criado pelo Autor.

Deve-se lembrar que a escolha de um conceito particular para uma dada posição na estrutura da frase restringe, necessariamente, os conceitos ou instâncias que podem ser apresentadas no menu *ComboBox* seguinte, de modo a assegurar a consistência semântica da consulta de acordo com os conceitos e relações binárias representadas no componente terminológico *TBox* (ou definições de classes OWL DL) da ontologia PFMEA DL. Por exemplo, se no primeiro menu *ComboBox* o usuário selecionar o conceito *Potential Failure Mode*, no menu *ComboBox3* este conceito não pode ser apresentado, do contrário ter-se-ia uma inconsistência semântica.

A Figura 7.20 ilustra a configuração da consulta que visa recuperar todos os potenciais modos de falha e suas respectivas causas potenciais, as quais foram identificadas pelo método de Análise de Modos de Falha e Efeitos para a produção um componente específico. Assim, é necessário recuperar, em primeiro lugar, os componentes modelados como instâncias (ou indivíduos) do conceito *ComponentLevel* representados nas bases de conhecimento dos agentes de recursos PFMEA.

Na Figura 7.21 a implementação desta funcionalidade pode ser comprovada pela interface do agente *Sniffer* (BELLIFEMINE et al., 2012). Nesta figura pode-se observar a interação e a troca de mensagens entre o agente de interface PFMEA denominado como *FailureModeAnalysisFinderAgent* e os agentes de recursos PFMEA denominados como *PFMEADLAgent0*, *PFMEADLAgent1* e *PFMEADLAgent2*, que estão ativos na plataforma.

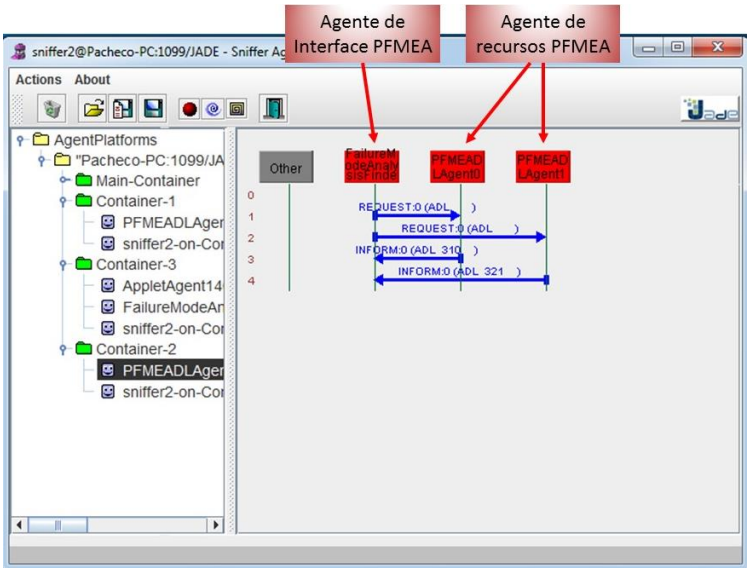

Figura 7-21 - Interface gráfica do Agente *Sniffer*.
Fonte: Criado pelo Autor.

7.4 Conclusões do capítulo

Neste capítulo foi apresentado e discutido o processo de verificação, validação conceitual e operacional do modelo baseado em agentes proposto neste trabalho. Os procedimentos metodológicos adotados neste capítulo, bem como seus fundamentos científicos, podem ser encontrados no Capítulo 3.

Os resultados obtidos demonstram a cientificidade e o potencial de aplicabilidade do modelo proposto, bem como não evidenciaram problemas incontornáveis. Evidentemente, novas implementações computacionais poderão ser efetuadas com os recursos emergentes de

software, atualmente em desenvolvimento no campo de pesquisa de sistemas multiagentes. Contudo, deve-se destacar que esta não é uma questão de fundo teórico, mas de natureza de suporte computacional.

Neste cenário, é importante refletir sobre o pensamento de Feigenbaum (2003) em relação ao primeiro princípio da engenharia de conhecimento, no qual o poder de solução de problemas exibido por sistemas inteligentes é, fundamentalmente, uma consequência de sua base de conhecimento e, apenas de forma secundária, consequência do método de inferência utilizado.

Logo, os sistemas baseados em conhecimento precisam ser ricos em conhecimento, pois os métodos de inferência implementados no modelo, apesar de engenhosos e cientificamente validados, não são perfeitos.

Do ponto de vista de padrões internacionais, o modelo proposto é baseado nas especificações FIPA 2000 - *Foundation for Intelligent Physical Agents* (FIPA, 2002a) o que assegura sua compatibilidade com novas tecnologias da área de multiagentes.

8 DISCUSSÃO E RESULTADOS

O modelo baseado em agentes sugerido nesta tese foi implementado a partir de um modelo conceitual da organização multiagente e de sua respectiva representação em termos de uma especificação de projeto, os quais foram desenvolvidos cientificamente a partir da metodologia GAIA. Nesta perspectiva, o modelo proposto consiste das classes de agentes de interface e de recursos de conhecimento, além do agente *Matchmaker*.

Os agentes da classe de recursos de conhecimento foram dotados individualmente com um modelo de comportamento projetado para acessar e manipular bases de conhecimento específicas por meio de métodos de raciocínio baseado em casos ou métodos de raciocínio e recuperação de conhecimento a partir de ontologias, reciprocamente. Neste sentido, a pesquisa mostrou que cada instância da classe de agentes de recurso de conhecimento pode estar associada a uma fonte de conhecimento em particular, e estas fontes, por sua vez, podem representar um dos diferentes processos de manufatura ao longo da cadeia produtiva, bem como encontrar-se distribuída em diferentes *hosts*.

Adicionalmente, estas instâncias podem, eventualmente, originar-se da organização multiagente ou mesmo novas instâncias podem ser incluídas a qualquer momento, tendo em vista a dinâmica da organização da cadeia produtiva. No entanto, esta flexibilidade não afeta a autonomia da estrutura da organização multiagente.

Por sua vez, os agentes da classe de interface foram dotados de modelos de comportamentos destinados à interação e à comunicação com os usuários e com as instâncias da classe de agentes de recursos de conhecimento. Estes agentes de interface são capazes de comunicar-se diretamente com aqueles agentes que dispõem do serviço de raciocínio e recuperação apropriados às necessidades do usuário, além de possuírem estratégias próprias para conduzir o usuário durante a configuração de consultas personalizadas.

Neste sentido, a pesquisa confirmou que estas instâncias dispõem de um suporte computacional adequado à complexidade das interações necessárias ao compartilhamento e recuperação de conhecimento, as quais são caracterizadas pela distribuição intrínseca das fontes de conhecimento.

O agente *Matchmaker* foi dotado de um modelo de comportamento destinado a registrar os serviços e as localizações dos demais agentes ativos na organização e, portanto, deve ser mantido sempre ativo na organização em um *host* principal.

No modelo sugerido outro aspecto fundamental diz respeito às bases de conhecimento dos agentes de recursos e, portanto, especial atenção foi dispensada tanto à modelagem conceitual destas bases quanto à formalização, implementação e operacionalização destas bases.

A pesquisa demonstrou que, em essência, a estrutura conceitual proposta para descrever o caso de não-conformidade, elemento central da base de conhecimento dos agentes de recursos dotados de métodos de RBC, ajusta-se adequadamente à forma dos relatos das experiências de solução de problemas de não-conformidades que foram identificados.

Nesta ótica, a pesquisa revelou a eficácia da estratégia adotada na etapa de implementação e operacionalização das bases de conhecimento dos agentes de recursos RBC. Esta estratégia caracteriza-se pela função de transformação de uma descrição no nível do conhecimento, isto é, das experiências relatadas na forma de linguagem natural em uma descrição no nível simbólico capaz de ser tratada pelos métodos encapsulados no modelo de comportamentos dos agentes em questão.

Em relação às bases de conhecimento ontológicas, a pesquisa comprovou que os conceitos, relações binárias e instâncias modeladas na ontologia PFMEA, decorrentes da aplicação do método de Análise do Modo de Falha e Efeitos em Processos de Manufatura, possuem a expressividade semântica necessária à representação do conhecimento no domínio. Neste sentido, a comprovação da expressividade da ontologia PFMEA envolveu o uso de casos de teste disponíveis na literatura, bem como em relação à pesquisa de campo.

Em especial, a pesquisa demonstrou ainda a eficácia da função de transformação adotada na operacionalização da base ontológica, função esta que compreende a transformação da descrição no nível do conhecimento, isto é, o conhecimento factual decorrente da aplicação do método PFMEA em uma descrição no nível simbólico considerando os conceitos, relações binárias e instâncias modeladas na ontologia PFMEA.

Verificou-se ainda o componente intencional *TBox* (*terminological component*) formalizado na ontologia PFMEA e passível de reutilização em diferentes processos de manufatura considerando a operacionalização de conhecimentos de referência em diferentes domínios.

Avaliando os resultados do processo de validação conceitual e operacional empreendidos neste trabalho de tese, são enumeradas as seguintes conclusões:

(1) O modelo proposto pode ser analisado como uma abordagem praticável e promissora para o compartilhamento e reuso de

conhecimentos entre especialistas e agentes computacionais no domínio de soluções de problemas de não-conformidades.

(2) O modelo proposto permite ainda um suporte adequado às ações de raciocínio e à recuperação de conhecimentos no domínio.

(3) As bases de conhecimento operacionalizadas pelos agentes de recursos de conhecimento podem representar de forma adequada o conhecimento consequente da solução de não-conformidades prévias mais similares ou da aplicação do método preventivo de análise de modos de falha e efeitos para processos de manufatura (PFMEA).

(4) Os métodos de RBC encapsulados no modelo de comportamento dos agentes de recursos mostraram-se viáveis para a recuperação de casos, mesmo quando estes são representados mediante o formalismo mais simples como o vetor de atributo-valor, graças à estratégia de configuração da consulta adotada pelo agente de interface RBC. Isto porque esta estratégia aceita que o usuário ajuste de forma personalizada e de acordo com suas preferências os pesos dos atributos a serem usados nas medidas de similaridade.

(5) Os métodos de recuperação de conhecimento a partir de bases ontológicas que foram encapsulados no modelo de comportamento dos agentes de recursos PFMEA mostraram-se viáveis graças à estratégia de configuração dinâmica da consulta adotada no modelo de comportamento do agente de interface PFMEA. Esta estratégia, em especial, incorpora uma funcionalidade que permite a verificação da consistência semântica da consulta em função dos conceitos e relações binárias representadas no componente terminológico TBox da ontologia PFMEA DL.

O protótipo foi testado em uma empresa que fabrica perfis de alumínio, o qual teve uma boa aceitação pelos usuários. Obteve-se através desse teste em campo opiniões dos colaboradores da empresa, os quais irão contribuir para a melhoria contínua pretendida para o software.

O protótipo desenvolvido foi capaz de mostrar que o modelo proposto e as tecnologias envolvidas são aplicáveis e válidas no domínio e para o uso pretendido, considerando um conjunto de casos de teste que consiste em uma amostragem representativa do domínio em questão.

Portanto, as conclusões e as recomendações aqui apresentadas devem ser consideradas à luz destas limitações.

9 CONCLUSÕES

A investigação descrita nesta tese buscou explorar a potencialidade da abordagem de modelos baseados em agentes em apoio à solução de problemas de não-conformidades em processos de manufatura, a partir do compartilhamento e recuperação de conhecimentos de fontes e cenários distribuídos.

As classes de agentes constituídas no modelo sugerido representam conceitualmente abstrações que visam superar os obstáculos impostos pela complexidade decorrente tanto dos cenários de apoio quanto da distribuição espacial e organizacional das fontes, além da própria variedade das características envolvidas em uma aplicação desta natureza, tais como: diferenças entre hardwares, sistemas operacionais e tipos de redes.

O modelo sugerido foi implementado na forma de um protótipo, de maneira a permitir a validação do trabalho de pesquisa realizado nesta tese.

Dentro deste contexto, este capítulo apresenta a discussão dos resultados obtidos, as conclusões e as sugestões para trabalhos futuros.

9.1.1 Desenvolvimento do protótipo de laboratório do modelo proposto

O desenvolvimento do protótipo de laboratório destinado à verificação e validação do modelo proposto na tese exigiu um processo de integração não trivial de diferentes recursos de software em uma arquitetura apropriada e coerente.

Em primeiro lugar, para a implementação dos agentes propostos foi usado o arcabouço JADE (*Java Agent DEvelopment Framework*) desenvolvido em conformidade as especificações FIPA 2000 - *Foundation for Intelligent Physical Agents*, e amplamente utilizado em aplicações da tecnologia de agentes tanto de natureza acadêmica quanto industrial.

O arcabouço jCOLIBRI (BELLO-TOMAS et al., 2004) desenvolvido pelo Grupo de Aplicações de Inteligência Artificial da Universidade Complutense de Madri (Espanha) foi utilizado para a implementação das tarefas e métodos de raciocínio baseado em casos no modelo de comportamento dos agentes de recursos RBC.

O sistema RacerPro Server (*Renamed ABox and Concept Expression Reasoner Professional*), que corresponde a uma nova versão do sistema Racer desenvolvido por Haarslev e Moller (2001), foi usado

para o suporte aos serviços padrão de raciocínio automático, bem como para possibilitar a realização de consultas conjuntivas visando a recuperação de conhecimento a partir de bases ontológicas. A conexão e a comunicação com este sistema foram implementadas no modelo de comportamentos dos agentes de recursos PFMEA.

No entanto, as dificuldades encontradas no desenvolvimento do protótipo foram de ordem computacional, pois os modelos de programação de agentes ainda não podem ser considerados maduros quando comparados com outros paradigmas de engenharia de software.

Além das questões metodológicas e conceituais tratadas nesta tese, cumpre ressaltar que a principal barreira a ser considerada em aplicações industriais em larga escala do modelo diz respeito à atitude dos especialistas frente ao compartilhamento do conhecimento, pois o conhecimento gera valor dentro da cadeia produtiva e representa um valioso capital intelectual do especialista.

9.2 CONTRIBUIÇÕES

Como principais contribuições deste trabalho de tese, podem-se citar:

(1) O sistema desenvolvido tem como finalidade resolver rapidamente estes problemas, compartilhar o conhecimento e lições aprendidas sobre as soluções de diferentes não-conformidades, o que resulta em soluções mais eficazes dos problemas, alcançando maior garantia de qualidade, além de poder ser utilizado como meio de treinamento de novos colaboradores que iniciam no processo de manufatura.

(2) A definição e a formalização por meio de vetores atributo-valor da estrutura conceitual que descreve os casos de não-conformidades, os quais envolvem os descritores da não-conformidade por si só, a solução sugerida e os resultados obtidos após a implementação solução. Ressalta-se que a estrutura proposta representa o centro da base de conhecimento que será acessada e manipulada pelos agentes de recursos de conhecimento RBC previstos no domínio do processo de extrusão de alumínio direta.

(3) A definição e a formalização da ontologia PFMEA mediante o uso de lógica de descrições (*Description Logics DL*) e a sua respectiva codificação por meio da linguagem OWL-DL (*Web Ontology Language – Descrition Logic*). Esta contribuição visa,

em particular, superar as barreiras semânticas identificadas no capítulo introdutório, além de representar o centro da base de conhecimento que será acessada e manipulada pelos agentes de recursos de conhecimento PFMEA.

(4) Devido à sua modularidade, a arquitetura proposta permite, em trabalhos futuros, a adição de recursos de conhecimento que têm casos de conhecimentos específicos sobre outros processos de manufatura.

9.3 SUGESTÕES PARA TRABALHOS FUTUROS

As sugestões para trabalhos futuros têm por finalidade indicar aspectos que exigem novas pesquisas e desenvolvimentos.

(1) Analisar a possibilidade de interligar o sistema protótipo desenvolvido com os controladores da própria máquina, de modo a obter os parâmetros de processo diretamente da máquina, evitando o apontamento manual do processo.

(2) Pré-selecionar as não-confomidades mais comuns no *ComboBox* da interface e deixá-las como opções iniciais.

(3) Pesquisar métodos para automatizar o mapeamento de instâncias das classes OWL DL diretamente nos bancos de dados, em especial para aquelas organizações que já dispõem de sistemas de bancos de dados próprios para armazenamento de dados do método de análise de modos de falha e efeitos (FMEA).

(4) Investigar métodos para automatizar o mapeamento do conhecimento nos elementos textuais, disponíveis no relato de não-conformidades, considerando a estrutura conceitual do caso de não-conformidade e o uso de processadores de linguagem natural existentes.

10 REFERÊNCIAS

AAMODT, A.; PLAZA, E. **Case a Based Reasoning: Foundational Issues, Methological Variations, and System Approaches**. AI Communications. IOS Press, v.7, n.1, p. 39-59. 1994.

ABDULLAH, M.; KIMBLE, C.; BENEST, I.; PAIGE, R. **Knowledge-based systems: a reevaluation.** Journal of Knowledge Management, v. 10, n. 3, p. 127-142, 2006.

AGENT ORIENTED SOFTWARE GROUP AOS. **The JACK Development Environment (JDE) version 5.0** - 2006. Disponível em: <http://www.agent-software.com/shared/products/index.html>. Acesso em: 15 jan 2013.

AGENTBUILDER LITE. **An Integrated Toolkit for Constructing Intelligent Software Agents - Release 2004**. Disponível em: <http://www.agentbuilder.com/>. Acesso em: 10 maio 2011.

ARPÍREZ, J. C.; CORCHO, O.; FERNÁNDEZ-LÓPEZ, M.; GÓMEZ-PÉREZ. A. **WebODE: A scalable workbench for ontological engineering**. In: International Conference on Knowledge Capture, First, 2001, Canada. **Proceedings**⋯ Canada: ACM Press, 2001. p. 6-13.

Associação Brasileira do Alumínio – ABAL. **Introdução.** Disponível em: http://www.abal.org.br/industria/introducao.asp. Acesso em: 1 janeiro 2014.

ASSOCIACÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 5462: Confiabilidade e Mantenabilidade**. Rio de Janeiro, 1994. 37p.

AUTOMOTIVE INDUSTRY ACTION GROUP – AIAG. **Quality Manual: CQI-10 Effective Problem Solving Guideline**, Southfield, Michigan, 2006.

AUTOMOTIVE INDUSTRY ACTION GROUP - AIAG. **Reference Manual: Failure Modes and Effects Analysis**, 3rd ed. Southfield, Michigan, 2006. 78 p.

BAADER, F.; NUTT, W. Basic Description Logics. In: BAADER, F. (Editor). **Description Logic Handbook: Theory, Implementation and Applications**. West Nvack, New York, USA: Cambridge University Press, 2003, p. 43-95.

BAADER, F.; SATTLER, U. **An Overview of Tableau Algorithms for Description Logics.** Studia Logica, v. 69, n. 1, p. 5-40, 2001.

BÄUMER, Christoph, BREUGST, Markus; CHOY, Sang; MAGEDANZ, Thomas Grasshopper - **A universal agent platform based on OMG MASIF and FIPA standards.** In: International Workshop on Mobile Agents for Telecommunication Applications, First, 1999. Canada. **Proceedings**⋯ Canada: World Scientific Pub., 1999. p. 1-18.

BECHHOFER, S.; MÖLLER, R.; CROWTHER, P. **The DIG Description Logic Interface.** In: International Workshop on Description Logics (DL2003), 2003, Italy. Proceedings⋯Italy: CEUR-WS, 2003. p. 1-8.

BELLIFEMINE, F., CAIRE, G., TRUCCO, T., and RIMASSA, G., 2012. **Jade Programmer's Guide**. Disponível em: http://jade.tilab.com/doc/programmersguide.pdf. Acesso em: 6 maio 2012.

BELLIFEMINE, Fabio; CAIRE, Giovanni; TRUCCO, Tiziana; RIMASSA, Giovanni; MUNGENAST, Roland. **JADE Administrators guide** – last update: 10 November 2006. JADE 3.4.1 Disponivel em: <http://jade.tilab.com/>. Acesso em: 6 maio 2012b.

BELLO-TOMÁS, J.J.; GONZÁLEZ-CALERO, P.A.; DÍAZ-AGUDO, B. **jCOLIBRI: An Object-Oriented Framework for Building CBR Systems.** In: European Conference on Case- Based Reasoning (ECCBR 2004), 7th, Advances in Case-Based Reasoning, 2004, Madrid, Spain. Proceedings⋯ Spain: Lecture Notes in Artificial Intelligence, v. 3155, Springer, 2004.

BLUVBAND, Z.; GRABOV, P.; NAKAR, N. **Expanded FMEA (EFMEA)**. In: International Symposium on Product Quality & Integrity,

2004, Los Angeles, CA, USA. Anais…Proceedings…USA: Annual Reliability and Maintainability Symposium, IEEE, 2004. p. 31-36.

BOGAERTS, Steven; LEAKE, David. **IUCBRF: A Framework for rapid and modular Case-Based Reasoning System Development**. Indiana, USA: Computer Science Department, Indiana University, 2005. 63 p. (Technical Report 617, Report Version 1.0).

BRANDT, S. C.; JARKE, M.; MIATIDIS, M.; RADDATZ, M.; SCHLÜTER, M. **Management and Reuse of Experience Knowledge in Extrusion Processes,** Springer-Verlag Berlin Heidelberg, P. 675-695, 2008.

BUCHANAN, B., BARSTOW, D.. BECHTEL, R., BENNETT, J., CLANCEY, W., KULIKOWSKI, C.. MITCHELL, T., and WATERMAN, D.A. **Constructing an expert system**. In: Building Expert Systems, F. HAYES-ROTH, D.A. WATERMAN, and D.B. LENAT (Eds.) Addison-Wesley, Reading, MA, 1983.

BUTDEE, S., NOOMTONG, C. e TICHKIEWITCH, S. **A Process Planning System with Feature Based Neural Network Search Strategy for Aluminum Extrusion Die Manufacturing**, *Asian International Journal of Science and Technology in Production and Manufacturing Engineering*, p. 137-157, 2009.

B. DÍAZ-AGUDO, Maria B. **Una aproximación ontológica al desarrollo de sistemas de razonamiento basado en casos.** 2002. 281 f.. Tesis (Doctorado en Informática) - Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid, Madrid, 2002.

CAMARINHA-MATOS, L.M.; AFSARMANESH, H. **The virtual enterprise concept.** In: IFIP TC5 WG5.3 / PRODNET WORKING CONFERENCE ON INFRASTRUCTURES FOR VIRTUAL ENTERPRISES: NETWORKING INDUSTRIAL ENTERPRISES. Proceedings… Kluwer Academic: Boston, 1999.p.3–14.

CARRIE, A. **From integrated entreprises to regional clusters: the changing basis of competition**. Computers inIndustry.v.42, n.2, p.289-298, 2000.

CERNUZZI, Luca; ZAMBONELLI, Franco. **Dealing with Adaptive Multi-agent Organizations in the Gaia.** In: International Workshop on Agent-Oriented Software Engineering VI (AOSE 2005), 6th, 2005, Utrecht, The Netherlands. Proceedings⋯ Germany: Lecture Notes in Computer Science, Springer, v. 3950, p. 109-123. 2006.

CERVO, A.; BERVIAN, P. **Metodologia Científica.** 5. ed. São Paulo: Pearson Prentice Hall, 2002. 242 p.

CO-ODE. **Collaborative Open Ontology Development Environment – About the Project.** Disponível em: http://www.co-ode.org/about/ Acesso em: 15 junho 2011.

CORCHO, O.; FERNÁNDEZ-LÓPEZ, M; GÓMEZ-PÉREZ, **A. Methodologies, Tools and Languages for Building Ontologies: Where is the Meeting Point?**. Data and Knowledge Engineering, v. 46, n. 1, p. 41-64, 2003.

CROSBY, P. B. **Quality With out Tears: The Art of Hassle-free Management**. New York: McGraw-Hill, 1984. p. 205.

DALE, Jonathan; KNOTTENBELT, Johnny. **Network Agents Research: April Agent Platform ‑ version 4.4.3 - 2002.** Disponível em: <http://sourceforge.net/projects/networkagent>. Acesso em: 14 jan 2013.

DEITEL, H. M.; DEITEL, P. J. **JAVA como programar**. 8. ed. São Paulo: Pearson, 2010.

DEMING, W. E. **Out of the Crisis**. Boston, MA: MIT Press, 1982. 507 p.

DENNY, Michael. Ontology Building: A survey of editing tools. 2002. Disponível em: <http://www.xml.com/lpt/a/2002/11/06/ontologies.html >. Acesso em: 10 setembro 2011.

DHAFR, Nasreddin; AHMAD, Munir; BURGESS, Brian; CANAGASSABABADY, Siva. **Improvement of quality performance in manufacturing organizations by minimization of production defects.** Robotics and Computer-Integrated Manufacturing, v. 22, n. 5-6, p. 536-542, 2006.

DI STEFANO, Antonella; SANTORO, Corrado. **eXAT: an Experimental Tool for Programming Multi-Agent Systems in Erlang.** In: AI*IA/TABOO Joint Workshop "From Objects to Agents" (WOA 2003): Intelligent Systems and Pervasive Computing, 4th , 2003, Villasimius, CA, Italy. Proceedings··· Italy: WOA, Pitagora Editrice Bologna, 2003, p. 121-127.

DITTMANN, L.; RADEMACHER, T.; ZELEWSKI, S. **Performing FMEA using ontologies.** In: International Workshop On Qualitative Reasoning, 18th, 2004, Evanston, USA. Proceedings... USA: Northwestern University, 2004, p. 209‑216.

DOMINGUE, J.; MOTTA, E.; CORCHO-GARCIA, O. **Knowledge Modeling in Webonto and OCML: A user guide version 2.4**, 2000. Disponível em: http://kmi.open.ac.uk/projects/webonto/user_guide.2.4.pdf. Acesso em: 19 junho 2013.

DUINEVELD, A., STOTER, R., WEIDEN, M. & KENEPA, B. & BENJAMINS, V. Wonder. **Tools: A Comparative study of ontological engineering tools.** International Journal of Human-Computer Studies, v. 52, n. 6, p. 1111-1133, 2000.

EBRAHIMIPOUR, V. REZAIE, K. SHOKRAVI, S. **An ontology approach to support FMEA studies.** Expert Systems with Applications, v. 37, p. 671-677, 2010.

EMORPHIA. FIPA-OS ‑ version 2002. Disponível em: <http://www.emorphia.com/research/about.htm>. Acesso em: 12 jan 2013.

ERIKSSON, H., SHAHAR, Y., TU, S.W., PUERTA, A.R., MUSEN, M. **Task modeling with reusable problem-solving methods.** Artificial Intelligence, v. 79, p. 293‑326, 1995.

FARQUHAR, A.; FIKES, R.; RICE, J. **The ontolingua server: a tool for collaborative ontology construction.** In: Knowledge Acquisition for Knowledge-Based Systems (KAW'96), 10th, Canada, 1996. Proceedings

⋯Canada: KAW98 on web. Disponível em: http://ksi.cpsc.ucalgary.ca/KAW/. Acesso em: 19 junho 2011.

FAYAD, Mohamed; SCHIMIDT, Douglas; RALPH, Johnson. **Implementing application frameworks: object-oriented framework at work**. New York: John Wiley & Sons, 1999. 729 p.

FEIGENBAUM, A. V. **Total Quality Control.** New York: McGraw-Hill, 1991. 586 p.

FERNÁNDEZ-LÓPEZ, M.; GÓMEZ-PÉREZ, A.; PAZOS, A.; PAZOS, J. **Building a Chemical Ontology Using Methontology and the Ontology Design Environment.** IEEE Intelligent Systems & their applications, v. 4, n. 1, p. 37‑46, 1999.

FIGUEIRA FILHO, C.; RAMALHO, G. JEOPS ‑ **The Java Embedded Object Production System.** In: Ibero-American Conference on AI: Advances in Artificial Intelligence, 7th, 2000. Proceedings⋯ London: M. C. Monard and J. S. Sichman (Eds.), Lecture Notes In Computer Science, Springer-Verlag, vol. 1952. p. 53-62, 2000.

FILHO, E. B., SILVA, I. B., BATALHA, G. F. e BUTTON, S. T. **Conformação Plástica dos Metais**, 1ª ed. São Paulo: EPUSP, 2011.

FILIPOV, V. e CHRISTOVA, N. **A Solution for Integrated Manufacturing Operation Management,** *Computational Intelligence for Modelling Control e Automation*, p. 527-532, 2008.

FÖRSTER, H.; WARNECKE, G.; KLONARIS, P.; PFEIFER, T. DerRegelkreisistnochnichtgeschlossen. **Qualität und Zuverlässigkeit**, München: Carl HanserVerlag, v. 41, n. 10, 1996.

FOUNDATION FOR PHYSICAL AGENTS. SC00001L: FIPA **Abstract Architecture Specification.** Geneva, Switzerland, 2002a. 67 p. Disponível em: http://www.fipa.org/specs/fipa00001/SC00001L.pdf. Acesso em: 11setembro 2011.

FOUNDATION FOR PHYSICAL AGENTS. SC00061G: **FIPA ACL Message Structure Specification. Geneva, Switzerland,** 2002b. 8 p.

Disponível em: http://www.fipa.org/specs/fipa00061/SC00061G.pdf. Acesso em: 11setembro 2011.

FOUNDATION FOR PHYSICAL AGENTS. SC00067F: **FIPA Agent Message Transport Service Specification.** Geneva, Switzerland, 2002c. 12 p. Disponível em: <http://www.fipa.org/specs/fipa00067/SC00067F.pdf >. Acesso em: 11 setembro 2011.

FUNDAÇÃO NACIONAL DA QUALIDADE. **Conceitos Fundamentais da Excelência em Gestão**. São Paulo, 2006.

GAIA, 2012. GAIA – Group for Artificial Intelligence Applications, jCOLIBRI CBR Framework. Disponível em: http://gaia.fdi.ucm.es/research/colibri/jcolibri. Acesso em: 8 maio 2012.

GARCIA, A. C. B.; SICHMAN, J. S. Agentes e Sistemas Multiagentes. In: REZENDE, S. O. (Org.) **Sistemas Inteligentes: fundamentos e aplicações**. São Paulo: Editora Manole, 2005. p. 269-306.

GIL, A. C.; **Como elaborar projetos de pesquisa**. São Paulo: Editora Atlas, 2010.

GROSSO, W. E.; ERIKSSON, H.; FERGERSON, R. W.; GENNARI, J. H.; TU, S. W.; MUSEN, M. A. **Knowledge modeling at the millennium.** In: Knowledge Acquisition for Knowledge-Based Systems (KAW'98), 12th, Canada, 1996. Proceedings⋯ Canada: KAW99 on web. Disponível em: http://ksi.cpsc.ucalgary.ca/KAW/. Acesso em: 20 junho 2011.

GRUBER, T. R. **A translation to portable ontology specification.** Knowledge Acquisition, v.5, n. 2, p. 199-220, 1993.

GUNENDRAN, A.; YOUNG, R. I. M. **State of the art review: ontological approaches to manufacturing knowledge and information management.** International Journal of Production Research, TPRS-2006-IJPR-0793, 2006.

HAARSLEV, V.; MÖLLER, R. **RACER System description**. In: Automated Reasoning. First International Joint Conference (IJCAR

2001), 2001, Siena, Italy. Proceedings⋯ Germany: Lecture Notes in Artificial Intelligence, Springer, v. 2083, p. 701-705, 2001.

HASSAN, Alaa; SIADAT, Ali; DANTAN, Jean-Yves; MARTIN, Patrick. **Conceptual process planning – an improvement approach using QFD, FMEA, and ABC methods.** Robotics and Computer-Integrated Manufacturing. v. 26 p. 392-401, 2010.

HE, Z., WANG, H., WANG, M., LI, G. **Simulation of extrusion process of complicated aluminium profile and die trial**. Transactions of Nonferrous Metals Society of China, Volume 22, Issue 7, July 2012, Pages 1732–1737, Elsevier.

HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-oriented Methodologies: An Introduction.** In: HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-oriented Methodologies**. Hershey, PA: Idea Group Publishing, 2005.

HENDRICKS, K. B., SINGHAL, V. R., e STRATMAN, J. K., **The impact of enterprise systems on corporate performance: A study of ERP. SCM, and CRM system implementations**, Journal of Operations Management, Vol. 25, Issue 1, p. 66-82. 2007.

HORRIDGE, Matthew; KNUBLAUCH, Holger; RECTOR, Alan; STEVENS, Robert; WROE, Chris. **A practical guide to building OWL ontologies using the PROTÉGÉ-OWL Plugin and CO-ODE Tools** - Edition 1.0. UK: University of Manchester, 2004, 117 p.

HORSTMANN, C.S.; CORNELL, G. **Core Java 2: Fundamentos**. São Paulo: MAKRON Books Ltda. 2001. 654 p.

IBM. **Aglets Software Development Kit - ASDK** - Release 2.0.2 - 2002. Disponível em: <http://www.trl.ibm.com/aglets/>. Acesso em: 10 maio 2011.

INTERNATIONAL ELECTROTECHNICAL COMMISSION. **IEC 60812**: **Analysis techniques for system reliability: Procedure for failure mode and effect analysis (FMEA).** Geneva, Switzerland, 2006. 7 p.

INTERNATIONAL STANDARDS ORGANIZATION. **ISO 18629-1: Industrial Automation Systems and Integration - Process Specification Language - Part 1 - Overview and basic principles,** 2004. 27 p.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/TS 16949: Quality Management Systems - Particular requirements for the application of ISO 9001:2000 for automotive production and relevant service part organizations.** Geneva, Switzerland, 2002. 34p.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/TS 16949: Quality Management Systems - Particular requirements for the application of ISO 9001:2000 for automotive production and relevant service part organizations.** Geneva, Switzerland, 2002. 34p.

ISHIKAWA K. **What is Total Quality Control? The Japanese Way**. New York: Prentice-Hall, 1985. 240 p.

JACZYNSKI, M.; TROUSSE, B. **An Object-Oriented Framework for the Design and Implementation of Case-Based Reasoners.** In: German Workshop on Case-Based Reasoning, 6th, 1998, Berlin. Proceedings··· Germany: German Workshop on Case-Based Reasoning, 1998. v. 1. p. 1-10.

JAKKILINKI, R., N. SHARDA, M. G. Developing an Ontology for Teaching Multimedia Design and Planning - Preprint v2 – 2004. Disponível em: http://sci.vu.edu.au/~nalin/MUDPYOntologyPreprintV2.pdf. Acesso em: 17 junho 2011.

JAVA AGENT SERVICES. JSR 87: **JAVA SPECIFICATION REQUESTS - Review 2002**. Disponível em: <http://www.jcp.org/en/jsr/detail?id=87>. Acesso em: 11 julho 2012.

JEON, Heecheol; PETRIE, Charles; CUTKOSKY, Mark R.. **JATLite: A Java Agent Infrastructure with Message Routing.** IEEE Internet Computing, vol. 4, n. 2, p. 87-96. 2000.

JURAN, J. M.; GRYNA, F. M. **Quality Control Handbook.** 4th ed., New York: McGraw-Hill, 1988. 1774 p.

KIM, Jihie; GIL, Yolanda. **Incorporating tutoring principles into interactive knowledge acquisition**. International Journal of Human-Computer Studies, v. 65, p. 852‑872, 2007.

KLAMMA, R. **Vernetztes Verbesserungsmanagement mit einem Unternehmensgedächtnis - Repository.** 2000. 259 f. Doktors der Naturwissenschaften. Fakultät für Mathematik, Informatik und Naturwissenschaften der Rheinisch-Westfälischen Technischen Hochschule Aachen. Aachen, 2000.

KNOWLEDGE BASED SYSTEMS INC. ‑ KBSI. **Integrated Definition Methods - IDEF.** Disponível em: < http://www.idef.com/>. Acesso em: 23 outubro 2011.

KOLODNER, Janet L. **Case-Based Reasoning**. San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1993, 668 p.

LARI, A. **A decision support system for solving quality problems using case-based reasoning.** TQM & Business Excelence. v. 14, n. 6, p.733-745, 2003.

LEE, S.; O' KEEFE, R.M. **Developing a Strategy for Expert System Verification and Validation.** IEEE Transactions on Systems, Man and Cybernetics, v. 24, n. 4, p. 643-655, 1994.

LESZCZYNA, Rafał. **Evaluation of agent platforms**. In: ISPRA - Institute for the Protection and Security of the Citizen, 2004, Technical Report - European Commission, Joint Research Centre, 2004.

LUCK, M.; MCBURNEY, P.; PREIST, C. **A Manifesto for Agent Technology: Towards Next Generation Computing.** Journal of Autonomous Agents and Multi-Agent Systems, v. 9, n.3. p. 203-252, 2004.

MARKIC, I. STULA, M. MARAS, J. **Intelligent Multi Agent Systems for decision support in insurance industry**. Information and

Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on. p. 1118-1123, 2014

MARTIN, David L.; CHEYER, Adam J.; MORAN, Douglas B. **The Open Agent Architecture: A Framework for Building Distributed Software Systems.** Applied Artificial Intelligence, vol. 13, n. 1-2, p. 91-128. 1999.

MENDES, W.; GIRARDI, R.; LEITE, A. **Arquitetura baseada em ontologias de um agente RBC ontology-based architecture of a CBR agente.** Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on, p. 1-6. 2013.

METAXIOTIS, K.; ERGAZAKIS, K.; PSARRAS, J. **Exploring the world of knowledge management: agreements and disagreements in the academic/practitioner community.** Journal of Knowledge Management, v. 9, n. 2, p.6-18, 2005.

MIKOS, W. L. **Modelo Baseado em Agentes em Apoio à Solução de Problemas de Problemas de Não-Conformidades em Ambientes de Manufatura com Recursos Distribuídos**. Tese – Universidade Federal de Santa Catarina, Santa Catarina, 2009.

MIKOS, W.L., FERREIRA, J.C.E., BOTURA, P.E.A., FREITAS, L.S., **A distributed system for rapid determination of nonconformance causes and solutions for the thermoplastic injection molding process: A case-based reasoning agents approach**, IEEE Conference on Automation Science and Engineering (CASE), p. 755-760, 2011a.

MIKOS, W.L., FERREIRA, J.C.E., BOTURA, P.E.A., FREITAS, L.S., **A System for Distributed Sharing and Reuse of Design and Manufacturing Knowledge in the PFMEA Domain Using a Description Logics-based Ontology**, Journal of Manufacturing Systems, Vol 30, No. 3, p. 133-143, Elsevier, 2011b.

MYSQL. MySQL 5.5 Reference Manual, 2010. Disponível em: http://dev.mysql.com/doc/refman/5.5/en/. Acesso em: 10 janeiro 2012.

NAGARAJAN, R. e ABDULKAREEM, A. **Expert data capture system for shop floor management.** Computers & Industrial Engineering, Vol 19, p. 122-126. 1990.

NODINE, Marian; FOWLER, Jerry; KSIEZYK, Tomasz; PERRY, Brad; TAYLOR, Malcom; UNRUH, Amy. **Active Information Gathering in InfoSleuth**. International Journal of Cooperative Information Systems, vol. 9, n. 1-2, p. 3-28. 2000.

NWANA, Hyacinth S.; NDUMU, Divine T.; LEE, Lyndon C.; COLLIS, Jaron C. ZEUS: **A Toolkit for Building Distributed Multiagent Systems.** Applied Artificial Intelligence, v. 13, n. 1-2, p. 129-185, 1999.

OLIVEIRA, José A. B. **Coalition Based Approach for Shop Floor Agility - A Multiagent Approach**. 2003. 302 f.. Thesis (PhD degree in Electrical Engineering, speciality of Robotics and Integrated Manufacturing) - Faculdade de Ciências e Tecnologia, Universidade Nova Lisboa, Lisboa, 2003.

ONTOPRISE GmbH. **OntoEdit Tutorial: How to work with OntoEdit- User's guide for OntoEdit Version 2.6**, 2003. Disponível em: http://www.ontoprise.de/documents /tutorial_ontoedit.pdf. Acesso em: 28 junho 2011.

PATEL-SCHNEIDER, P.; HAYES, P. HORROCKS, I. W3C **Recommendation: OWL Web Ontology Language Semantics and Abstract Syntax.** 2004. Disponível em: <http://www.w3.org/TR/owl-semantics/>. Acesso em: 10 junho 2011.

PAOLUCCI, M.; SACILE, R. **Agent-Base Manufacturing and Control Systems**: New Agile Manufacturing Solutions for Archieving Peak Performance. Florida: CRC Press, 2005.

PECHOUCEK, Michal; THOMPSON, Simon. **Agents in Industry: The Best from the AAMAS 2005 Industry Track**. IEEE Intelligent Systems, v. 21, n. 2, p. 86-95, 2006.

PFEIFER, T. **Qualitätsmanagement: Strategien, methoden, techniken**. München: Hanser, 1996. 551 p.

PFEIFER, T. (Hrsg): **Fehlermanagement mit objektorientierten Technologien in der qualitätsorientierten Produktion.** FZKA-PFT 183, Kalsruhe, 1997.

PFEIFER, T.; KLONARIS, P.; LESMEISTER, F. Produktivität erhöhen; verbesserungs-managemet als effektives KVP-Werkzeug. **Werkstattstechnik.** v.88, n.5, p.208-210, 1998.

PFEIFER, T., LESMEISTER, F., WENDT, M. P. – AusFehlernlernen. **Planung + Produktion**. n.5, 2000.

PICKARD, Karsten; MÜLLER, Peter; BERTSCHE, Bernd. **Multiple Failure Mode and Effects Analysis – An Approach to Risk Assessment of Multiple Failures with FMEA.** In: International Conference on Robotics and Automation, 2005, Barcelona, Spain. Proceedings...Spain: IEEE Robotics and Automation Society, 2005, p. 457-462.

RACER SYSTEMS GMBH & Co. KG, **RacerPro Server, User's Guide** Version 1.9, 2005. Disponível em: http://www.racer-systems.com. Acesso em: 12 de junho 2011.

RAINER, Umland; KLUSCH, Matthias; CALISTI, Monique. **Software Agent-Based Applications, Platforms and Development Kits**. Whitestein Series in Software Agent Technologies and Autonomic Computing. Swiss: Birkhäuser Verlag, 2005. 448 p.

RECIO-GARCÍA, Juan A.; SÁNCHEZ, Antonio; DÍAZ-AGUDO, Maria B.; GONZÁLEZCALERO, Pedro A. **jCOLIBRI 1.0 in a nutshell: a software tool for designing CBR systems.** In: UK Workshop on Case Based Reasoning, 10th, 2005, Cambridge. Proccedings···Cambridge: CMS Press, University of Greenwich, 2005. p. 20-28.

RECIO-GARCÍA, J. A. jCOLIBRI: **A Multi-Level Platform for Building and Generating Case-Based Reasoning Systems**. Tese – Facultad de Informática - Universidad Complutense de Madrid, Spain, 2008.

RECIO-GARCÍA, J. A.; DÍAZ-AGUDO, B.; GONZÁLEZ-CALERO, Pedro A. **jcolibri2: A framework for building Case-based reasoning systems.** In Journal: Science of Computer Programming, 2014. v. 79, p. 126-145.

RYMER, J. R. The Muddle in the Middle. **Byte.com**, v. 21, n. 4, p. 67-70, April 1996.

SAHA, P. K. **Aluminum Extrusion Technology**. ASM International. Ohio, 2000.

SCHREIBER, A. T. The KADS approach to knowledge engineering: editorial special issue. **Knowledge Acquisition**, v. 4, n. 1, 1992.

SECCHI, P.; CIASCHI, R.; SPENCE, D. A concept for an ESA lessons learned system. In: P. Secchi, Editor, Proceedings of Alerts and LL: An Effective Way to Prevent Failures and Problems, ESTEC, Noordwijk, Netherlands, 1999, p. 57‑61 Technical Report WPP-167.

SHEN, W., NORRIE, D. H., BARTHÈS. J. P. **Multi-agent systems for concurrent intelligent design and manufacturing.** London, New York: Taylor & Francis, 2001.

SHEN, W.; LANG, S.Y.T.; WANG, L. ***iShopFloor:*** **An Internet-Enabled Agent-Based Intelligent Shop Floor.** IEEE Transaction on Systems, Man and Cybernetics - part C: Applications and Reviews. v.35, n. 3, p. 371-381, 2005.

SHIU, S. C. K.; PAL, S. K. **Case-Based Reasoning: Concepts, Features and Soft Computing.** Applied Intelligence, v. 21, n. 3, p. 233-238, 2004.

SIRIN, Evren; PARSIA, Bijan; GRAU, Bernardo C.; KALYANPUR, Aditya; KATZ, Yarden. **Pellet: A practical OWL-DL reasoner**. Maryland: University of Maryland, Institute for Advanced Computer Studies, 2005. 26 p. (UMIACS Technical Report CS 4766).

SISLAK, David; REHAK, Martin; PECHOUCEK, Michal; ROLLO, Milan; PAVLICEK, Dusan. **A-globe: Agent Development Platform with Inaccessibility and Mobility Support**. In: UNLAND, R.; KLUSCH, M.; CALISTI, M. (Editors) Software Agent-Based Applications, Platforms and Development Kits. Berlin: Birkhauser Verlag Germany, 2005. p. 1-25.

SLACK, N., CHAMBERS, S., JOHNSTON, R. **Administração da Produção.** São Paulo: Editora Atlas, 2009.

SOCIETY OF AUTOMOTIVE ENGINEERS INTERNATIONAL. SAE J1739: **Potential Failure Mode and Effects Analysis**. USA, PA, 2002. p.60.

STAMATIS, D. H. **Failure Mode and effect analysis: FMEA from theory to execution.** 2nd ed. Milwaukee, Wisconsin: ASQ Quality Press, 2003. 455 p.

SU, X.; ILEBREKKE, L. **A Comparative Study of Ontology Languages and Tools.** In: international Conference on Advanced information Systems Engineering, 14th, 2002, Toronto, Canada. Proceedings··· Canada: Lecture Notes In Computer Science, Springer-Verlag, v. 2348, 2002. p. 761-765.

SULTAN, Khalid; BENTAHAR, Jamal; WAN, Wei; AL-SAGGAR, Faisal. **Modeling and verifying probabilistic Multi-Agent Systems using knowledge and social commitments**. Expert Systems with Applications. v. 14, p. 6291-6304, 2014.

SUN DEVELOPER NETWORK. **JAVA SE Overview**. Disponível em: <http://java.sun.com/javase/technologies/index.jsp#overview>. Acesso em: 14 março 2012.

SURE, Y., STAAB, S., STUDER, R. **Methodology for development and employment of ontology based knowledge management applications.** ACM SIGMOD Record, v. 31, n. 4, p. 18-23, 2002.

SYCARA, Katia; PAOLUCCI, Massimo; van VELSEN, Martin; GIAMPAPA, Joseph. A. **The RETSINA MAS Infrastructure**. Pittsburgh, PA: Robotics Institute, Carnegie Mellon University, 2001. 26 p. (CMURI-TR-01-05).

TAGUCHI, G., WU, Y. **Introduction to Off-line Quality Control.** Nagoya: Japan Quality Control Organization, 1980. 111 p.

TEOH, P.C.; CASE, K. **Failure modes and effects analysis through knowledge modeling.** Journal of Materials Processing Technology, v. 153 ‑154, p. 253‑260, 2004a.

TEOH, P.C.; CASE, K. **Modelling and reasoning for failure mode and effects analysis generation.** In: Institution Of Mechanical Engineers, 218, 3, Mar 2004. Proceedings⋯ ProQuest Science Journals, 2004b. p. 289-300.

TRYLLIAN. **ADK: Agent Development Kit - version 3.2.0** - 2000. Disponível em: <http://www.tryllian.com /solutions.html>. Acesso em: 15 junho 2011.

VALIENTE, M. C.; GARCIA-BARRICIONAL, E.; SICILIA, M. A. **Applying an ontology approach to IT service management for business-IT integration.** Knowledge-Based Systems. v. 28 p. 76-87, 2012.

van ELST, L.; DIGNUM, V.; ABECKER, A. **Towards Agent-Mediated Knowledge Management.** In: International Symposium in Agent Mediated Knowledge Management (AMKM), 2004, Stanford, CA, USA. Proceedings... Heidelberg: LNAI 2926, Springer, p. 1-31, 2004.

van HEIJST, G. A.; SCHREIBER, A. T.; WIELINGA, B. J. **Using explicit ontologies in KBS development**. International Journal of Human-Computer Studies, v. 46, n. 2/3, p.183-292, 1997.

von WANGENHEIM, Christiane Gresse; von WANGENHEIM, Aldo. **Raciocínio Baseado em Casos.** São Paulo: Editora Manole Ltda, 2003. 293 p.

W3C: World Wide Web Consortium: Web Ontology Language - OWL. Disponível em: <http://www.w3.org/2004/OWL/>. Acesso em: 10 julho 2011.

WATSON, I. **Applying Knowledge Management: Techniques for building Organizational Memories.** San Francisco, CA: Morgan Kaufmann Publishers Inc., p. 252, 2003.

WEBER, R.O.; AHA, D.W. **Intelligent delivery of military lessons learned.** Decision Support System. v. 34, p. 287-304, 2002.

WEBER, R.O.; AHA, D.W.; BECERRA-FERNANDEZ, I. **Intelligent lessons learned systems.** International Journal of Expert Systems – Research and Applications. v. 20, n.1, p. 17-34, 2001.

WIRTH, Rüdiger; BERTHOLD, Bernd; KRÄMER, Anita; PETER, Gerhard. **Knowledge Based Support of System Analisys for Failure Mode and Effects Analysis**. Engineering Applications of Artificial Intelligence, v. 9, n. 3, 1996, p. 219-229.

WOOLDRIDGE, M.; JENNINGS, N.; KINNY, D. **The Gaia Methodology for Agent-Oriented Analysis and Design.** Journal of Autonomous Agents and Multi-Angent Systems, v. 3, p. 285-312, 2000.

WOOLDRIDGE, M. **An Introduction to Multiagent Systems**. London: JohnWiley&Sons Ltd, 2002.

WONG, B. K, e MONACO, J. A., **Expert system aplications in business: A review and analysis of the literature (1977-1993)**, Information& Management, ELSEVIER, p. 141-152, 1995.

YAN, W.; Zanni-Merk, C.; Cavallucci, D.; Collet, P. **An ontology-based approach for inventive problem solving.** Engineering Applications of Artificial Intelligence. p.175-190, 2014.

YILMAZ, Levent. **Validation and verification of social processes within agent-based computational organization models.** Computational & Mathematical Organization Theory, v. 12 , n. 4, p. 283 - 312, 2006.

ZAMBONELLI, F.; JENNINGS, N.; WOOLDRIDGE, M. **Developing Multiagent Systems: The Gaia Methodology.** ACM Transaction on Software Engineering and Methodology. v. 12, n. 3, p. 317-370, 2003.

ZAMBONELLI, F.; JENNINGS, N.R.; WOOLDRIDGE, M. **Multi-Agent Systems as Computational Organizations: The Gaia Methodology.** In: HENDERSON-SELLERS, B.; GIORGINI, P. Agent-oriented Methodologies. Hershey, PA: Idea Group Publishing, 2005.

ZATTAR, I. C., **Modelo de Simulação Baseado Agentes para o Estudo da Influência de Planos de Processos Alternativos na Programação**

da Produção em Sistemas de Manufatura com Layout Funcional. Tese (Doutorado em Engenharia Mecânica) - Universidade Federal de Santa Catarina – 2008.

ZATTAR, I.C., FERREIRA, J.C.E., RODRIGUES, J.G.G.G. e SOUZA, C.H.B., **A Multi-agent System for the Integration of Process Planning and Scheduling Using Operation-Based Time-Extended Negotiation Protocols**, International Journal of Computer Integrated Manufacturing, Vol. 23, No. 5, maio 2010, pag. 441-452, Taylor & Francis, Inglaterra.

ZÚÑIGA, G. 2001. **Ontology: its transformation from philosophy to information systems.** In: International Conference on Formal ontology in information Systems, 2001, Ogunquit, Maine, USA. Proceedings⋯ New York: Formal Ontology in Information Systems, ACM Press, v. 2001, 2001. p. 187-197.