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

MODELO BASEADO EM AGENTES EM APOIO À SOLUÇÃO DE PROBLEMAS DE NÃO-CONFORMIDADES EM AMBIENTES DE MANUFATURA COM RECURSOS DISTRIBUÍDOS

Tese submetida à

UNIVERSIDADE FEDERAL DE SANTA CATARINA

para a obtenção do grau de

DOUTOR EM ENGENHARIA MECÂNICA

WALTER LUÍS MIKOS

Florianópolis, SC

2008

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

MODELO BASEADO EM AGENTES EM APOIO À SOLUÇÃO DE PROBLEMAS DE NÃO-CONFORMIDADES EM AMBIENTES DE MANUFATURA COM RECURSOS DISTRIBUÍDOS

WALTER LUÍS MIKOS

Esta tese foi julgada adequada para a obtenção do título de

DOUTOR EM ENGENHARIA

ESPECIALIDADE ENGENHARIA MECÂNICA

sendo aprovada em sua forma final.

_____

Prof. João Carlos Espíndola Ferreira, Ph. D. (Presidente/Orientador)

_____

Fernando Cabral, Ph. D (Coordenador do Curso)

BANCA EXAMINADORA

_____

Prof. Antônio Batocchio, Dr. Eng. (UNICAMP/ Relator)

_____

Prof. Marcelo Teixeira dos Santos, Dr. Eng. – (BRASILMATICS/Joinville)

_____

Prof. Carlos Henrique Ahrens, Dr. Eng.

_____

Prof. Fernando Antônio Forcellini, Dr. Eng.

Dedico este trabalho a Deus, Nosso Senhor, pelo dom da vida. A minha esposa Nádia por seu amor e apoio constante. A minha filha Natalia pela sua paciência e carinho. A meus pais Valderico (*in memorian*) e Odiléa pela formação educacional e exemplo. A minha irmã pelo incentivo e carinho.

AGRADECIMENTOS

Ao Professor João Carlos Espíndola Ferreira, meu orientador, pela inestimável colaboração ao longo do trabalho, pela paciência e pela amizade.

Aos Professores Kazuo Hatakeyama e Abelardo Alves de Queiroz, coordenadores do Programa de Qualificação Institucional PQI / CAPES, a quem admiro pela incansável dedicação à educação.

Aos Professores do Departamento Acadêmico de Mecânica da Universidade Tecnológica Federal do Paraná, em especial, aos professores do Grupo de Metrologia e Qualidade pelo apoio recebido.

Aos bolsistas de iniciação científica do laboratório GRIMA / UFSC: Paulo Eduardo de Albuquerque Botura, Fabbio Gonzalez Correia Gomes, Roman Moura Lorenzo, Leandro da Silva Freitas, João Gabriel Ganacim Granado Rodrigues e Carlos Humberto Barreto de Sousa pelo empenho e suporte técnico.

Aos colegas de laboratório Izabel Cristina Zattar, Julio César Ticona Benavente, Paulo Antonio Reaes, Mário Henrique Mello, Fernado Lemonje Westrupp e Hugo Gaspar Santos meus amigos e companheiros durantes as longas horas de trabalho.

Aos colegas Paulo Rogério da Silveira, Luiz Amilton Pepplow, Alexandre Moeckel, Carlos Henrique Mariano, Emerson Rigoni e Jorge Luiz Erthal meus amigos e companheiros de viagem.

Ao colega Gilberto Wolff pelo empenho na elaboração do Projeto PQI.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACC	- *Agent Communication Channel*
ACL	- *Agent Communication Language*
ADK	- *Agent Development Kit*
AIAG	- *Automotive Industry Action Group*
AMS	- *Agent Management System*
AUML	- *Agent Unified Modeling Language*
CBR	- *Case-Based Reasoning*
CoMMA	- *A multi-agent system for **C**orporate **M**emory **MA**nagement*
CO-ODE	- *Collaborative Open Ontology Development Environment*
DF	- *Directory Facilitator*
DIG	- *Description Logics Implementation Group*
DL	- *Description Logic*
EPS	- *The Effective Problem Solving Guideline*
FaCT ++	- *Fast Classification of Terminologies*
FIPA	- *Foundation for Intelligent Physical Agents*
FMEA	- Análise de Modos de Falhas e seus Efeitos
FMECA	- Análise de Modos de Falha, seus Efeitos e Criticalidade
FRODO	- *A FRamework for **O**pen/**D**istributed constraint **O**ptimization*
GAIA	- Grupo de Aplicações de Inteligência Artificial da Universidade Complutense de Madri
IDEF	- *Integrated Definition Method for Function Modelling*
IEEE	- *Institute of Electrical and Electronics Engineer*
IUCBRF	- *Indiana University Case-Based Reasoning Framework*
JADE	- *Java Agent DEvelopment Framework*
JAS	- *Java Agent Services API*
JDBC	- *Java Database Connectivity*
JDBC	- *Java Database Connectivity*)
JVM	- *Java Virtual Machine*
LGPL	- *Lesser General Public License*

Mindswap	- *Maryland Information and Network Dynamics Lab Semantic Web Agents Project*
MTP	- *Message Transport Protocol*
MTS	- *Message Transport System*
nRQL	- *new Racer Query Language*
OWL-DL	- *Web Ontology Language – Description Logic*
PFMEA	- Análise de Modos de Falhas e seus Efeitos em Processos de Manufatura
PSML	- *Problem Solving Methods Library*
RacerPro	- *Renamed ABox and Concept Expression Reasoner*
RBC	- Raciocínio Baseado em Casos
RMA	- *Remote Management Agent*
RQLD	- *Query Language for RDF*
UML	- *Unified Modeling Language*

RESUMO

Nos últimos anos, a necessidade de atuar em um âmbito de negócios globais, bem como atender a requisitos crescentes em relação à qualidade, diversidade e custo, têm levado as empresas de manufatura a buscar novas estruturas organizacionais como alternativas aos sistemas tradicionais centralizados. Este cenário globalizado vem induzindo novas formas de competição, que deixam de ser somente entre empresas individuais, e passam a ser também entre redes de empresas interconectas e que operam em ambientes de manufatura com recursos distribuídos. Neste cenário, novos desafios também são impostos aos modelos tradicionais de gestão e melhoria da qualidade, os quais devem ser capazes de cobrir não somente processos internos de uma única empresa, mas estender-se também aos processos externos envolvendo as empresas interconectadas. Nestes novos ambientes, em especial, a solução de problemas de não-conformidades caracteriza-se por atividades intensivas em conhecimento e baseadas, fortemente, em experiências, as quais, em casos complexos, podem extrapolar o conhecimento e a experiência dos membros de uma única empresa integrada. Tendo em vista este contexto, esta tese investiga o uso da abordagem de organizações multiagentes destinadas ao compartilhamento e a recuperação de conhecimentos decorrentes da solução de problemas prévios de não-conformidade e da aplicação do método preventivo de análise de modos de falha e efeitos em processos de manufatura (PFMEA). Neste sentido, propõe-se um modelo de organização multiagente em apoio à solução de problemas de não-conformidades em processos de fabricação, como uma alternativa capaz de superar não somente as barreiras relacionadas à própria natureza do conhecimento, mas também quanto à distribuição das fontes deste conhecimento. A noção de distribuição adotada no modelo considera tanto o aspecto da distribuição geográfica das fontes quanto à fragmentação relacionada aos diferentes processos existentes ao longo de uma cadeia de produtiva. Dentro desta ótica, serão considerados no modelo agentes computacionais cujo comportamento envolve o uso de métodos de raciocínio baseado em casos e métodos de recuperação baseada 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 pelos agentes foram instanciadas com conhecimentos no domínio do processo de moldagem por injeção de termoplásticos obtidos a partir da literatura e de pesquisas de campo.

Palavras-chave: Sistemas multiagentes; Raciocínio Baseado em Casos; Ontologias; Solução de problemas de não-conformidades; Moldagem por Injeção de Termoplásticos.

ABSTRACT

In the last years, the need to act in a global business environment, as well as to meet growing requirements with regard to quality, diversity and cost, has lead the manufacturing companies to earch for new organizational structures as alternatives to centralized traditional systems. This global scenario is inducing new competition ways, which do not take place only among the individual companies, but also among networks of interconnected companies that operate in manufacturing environments with distributed resources. In this context, new challenges are also imposed to the traditional management and quality improvement models, which should be capable to encompass not only internal processes of a single company, but also to extend to external processes involving the interconnected companies. In these new environments, especially the solution of non-conformity problems, is characterized by intensive knowledge activities, and are strongly based on experiences that, in complex cases, can go beyond the knowledge and experience of the members of a single integrated company. Considering this scenario, this thesis investigates the use of the multi-agent approach to provide sharing and recovery of knowledge that result from the solution of previous non-conformity problems, and the application of the preventive method of potential failure mode and effects analysis in manufacture processes (PFMEA). Computer agents are considered whose behavior involves the use of case-based reasoning methods, and ontology-based recovery. It is also proposed a model of multi-agent organization in support to the solution of non-conformity problems in production processes, as an alternative to overcome the barriers due to both the knowledge nature and the perspective of knowledge distribution, which includes not only the geographical aspect, but also the extent of processes and organizations along a supply chain. Finally, a computer prototype was developed to allow the verification and validation of the proposed model, and the knowledge bases manipulated by the agents were instantiated with knowledge from the thermoplastic injection molding domain, obtained from the literature and field research.

Key-words: Multi-Agent System, Case-Based Reasoning; Ontology, Nonconformity problem solving, Thermoplastic Injection Molding.

CAPÍTULO 1
INTRODUÇÃO

1.1 TEMA DE PESQUISA

Na atualidade, o ambiente industrial e econômico caracteriza-se pelo surgimento de novas estruturas organizacionais, induzidas por fatores complexos, tais como: a redução do ciclo de vida dos produtos, alta variabilidade na demanda, necessidade de alta flexibilidade e reatividade para operar em um âmbito de negócios globais (ZAIDAT et al., 2005).

Dentro desta lógica, a literatura tem reportado diferentes estruturas organizacionais, entre as quais: Empresa Virtual (*Virtual Enterprise*), Empresa Estendida (*Extended Enterprise*), Organização Virtual (*Virtual Organization*) e Rede de Suprimento Virtual (*Virtual Supply Network*), entre outras (CAMARINHA-MATOS e AFSARMANESH, 1999; WIENDAHL e LUTZ, 2002; ZAIDAT et al., 2005; CECIL et al., 2006; BINDER e CLEGG, 2007). Isto revela o crescente interesse tanto em termos de pesquisas acadêmicas quanto em relação às novas práticas de negócios.

Adicionalmente, a literatura revela que as empresas de manufatura têm procurado estas novas estruturas organizacionais, em contraposição aos sistemas tradicionais centralizados, como uma alternativa para manter a competitividade, aproveitando-se da ampla gama de tecnologias de informação e comunicação disponíveis atualmente (CECIL et al., 2006).

Notadamente, estas novas formas de organização envolvem a noção de redes formadas por clientes, fornecedores, diversas plantas industriais, centros de logística e de desenvolvimento, todos operando como ambientes de manufatura com recursos distribuídos (CECIL et al., 2006).

Nestes ambientes, todavia, o problema chave consiste em integrar efetivamente os recursos, sejam eles os ativos relacionados às competências essenciais altamente especializadas, como por exemplo novas tecnologias proprietárias ou conhecimentos que otimizam processos e recursos específicos da manufatura; ou os ativos menos específicos, como as tecnologias de processo ou informações de produção compartilhadas, que contribuem para a manufatura (CECIL et al., 2006; BINDER e CLEGG, 2007).

Observa-se, ainda, que estas novas estruturas, em geral, são compostas por empresas colaboradoras que se encontram distribuídas geograficamente, cada qual concentrada nas suas competências essenciais, unindo esforços e compartilhando recursos com as demais, para produzir um produto completo. Assim, cada empresa pode ser caracterizada como um nó que

adiciona algum valor à cadeia de produção (SOARES et al., 2000; CAMARINHA-MATOS e PANTOJA-LIMA, 2001; GARITA et al., 2001).

No entanto, estas mudanças no modo de organização e, por extensão, na forma de atuar no mercado, vêm produzindo, naturalmente, uma mudança também na forma de competição, deixando de ser, exclusivamente, entre empresas individuais e integradas e passando para uma competição entre redes (CARRIE, 2000).

Neste novo patamar de competição, a qualidade do produto continua sendo um requisito de desempenho fundamental. Em consequência disso, os modelos tradicionais de sistemas de gestão e melhoria da qualidade, bem como os sistemas de informações para qualidade, desenvolvidos no campo de gestão de operações, precisam ser revistos e expandidos para estas novas estruturas. Isto porque os sistemas de gestão e melhoria da qualidade agora devem ser capazes de cobrir não somente processos internos de uma única empresa integrada, mas estender-se além das fronteiras tradicionais envolvendo as empresas interconectadas e os clientes (TANG e LU, 2002; CHIN et al., 2006; BINDER e CLEGG, 2007).

Nesta ótica, Dhafr et al. (2006) argumentam que a solução efetiva de problemas de não-conformidades, certamente, é um pré-requisito fundamental para a melhoria contínua da qualidade dos produtos e dos processos, bem como para a redução de custos. Pois, mesmo as mais consistentes aplicações de medidas orientadas para a garantia qualidade não podem excluir completamente a possibilidade de que não-conformidades possam ocorrer durante os processos de produção (FÖRSTER et al., 1996; PFEIFER, 1997; KLAMMA, 2000; LARI, 2003; DHAFR et al., 2006).

Neste contexto, também, pode-se citar a referência normativa do setor automotivo para a Solução Efetiva de Problemas[1] (AIAG, 2006), a qual traduz o esforço combinado dos diversos especialistas na área de solução de problemas das empresas pertencentes ao grupo (AIAG), bem como de sua comunidade de fornecedores. Neste sentido, esse guia busca estabelecer um consenso sobre as metodologias referentes à solução de problemas de não-conformidade e os principais conceitos usados atualmente no setor.

Todavia, em termos conceituais, é importante destacar, que a maior contribuição desse referencial está na definição da abrangência do processo de solução efetiva de problemas de não-conformidades. Em especial, nesta referência normativa, um processo de solução efetiva estende-se além da finalidade imediata de resolver um problema e envolve uma ampla gama de atividades de gestão do conhecimento. E, neste sentido, essas atividades visam incorporar

[1] Tradução do título em inglês *The Effective Problem Solving Guideline,* publicado pelo Grupo de Ação da Indústria Automotiva (*Automotive Industry Action Group*) (AIAG, 2006).

as experiências e as lições aprendidas com a solução de um problema de não-conformidade, em particular, em todos os produtos e processos similares.

Neste cenário, é importante observar que o conceito de conhecimento, empregado nos parágrafos anteriores, refere-se à noção de "recurso" ou ativo que pode existir fora da mente humana. Portanto, pode ser representado e manipulado como qualquer outro objeto por meio de ferramentas de gestão do conhecimento[2], na linha de pensamento da teoria de estoque (*Stock-theory*) (van ENGERS, 2001).

A noção de "recurso" contrasta com a noção de "fluxo", na qual o conhecimento é considerado como um fenômeno psicológico e social que não pode ser considerado um ativo transferível, pois os atores envolvidos no processo adicionam a eles seus valores subjetivos (van ENGERS, 2001). Nesta linha teórica, o conhecimento não é transferido, mas recriado quando a mente humana entra em contato com as informações e, subsequentemente, as adapta ao seu modelo mental (LUEG, 2002).

1.2 PROBLEMA DE PESQUISA

A solução de problemas de não-conformidades em processos de manufatura permanece ainda um desafio do ponto de vista acadêmico, pois, fundamentalmente, as soluções de tais problemas envolvem atividades intensivas 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 trivial. A norma internacional NBR ISO 9004:2000 (Sistemas de gestão da qualidade – Diretrizes para melhorias de desempenho) destaca duas perspectivas ou cenários diferentes para esta questão. A primeira refere-se às não-conformidades que já aconteceram concretamente e, portanto, necessitam de ações corretivas para evitar a sua recorrência. E a segunda refere-se às não-conformidades potenciais que, desta forma, necessitam de ações preventivas que evitem a sua ocorrência.

---

[2] Do termo em inglês *Knowledege Management.* Na literatura, não existe uma definição universalmente aceita para gestão do conhecimento, mas atualmente se aceita que a gestão do conhecimento envolve um complexo conjunto de questões relacionadas a pessoas, organizações e tecnologias (MIKA et al., 2004), bem como envolve problemas de identificação, aquisição, armazenagem, acesso, difusão, reutilização e manutenção do conhecimento.

Para a primeira perspectiva, a norma estabelece que especial atenção deva ser dispensada: à análise critica da não-conformidade, determinação das causas, avaliação da necessidade das ações para assegurar que aquelas não-conformidades não ocorrerão novamente, determinação e implementação de ações necessárias, registro dos resultados de ações executadas e análise crítica das ações corretivas executadas.

Para a segunda perspectiva, a norma estabelece que especial atenção deva ser dispensada: 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.

Em relação à primeira perspectiva, um amplo estudo realizado em empresas alemãs com sistemas de manufatura tradicionais dos setores metal-mecânico e químico demonstrou que durante a produção, em média, 60% das não-conformidades ocorridas são recorrentes. Isto é, elas 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).

No entanto, 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, freqüentemente, não eram armazenados de forma apropriada. Isto dificulta em grande medida 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[3] (STAMATIS, 2003), isto porque 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.

Esse 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

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

compartilhamento e reuso no contexto de sistemas inteligentes de recuperação de conhecimento (DITMANN et al., 2004; TEOH e CASE, 2004a e 2004b). 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 (DITMANN et al., 2004; TEOH e CASE, 2004a e 2004b).

Assim, de acordo com o exposto acima e lembrando das palavras de Larroyo (1975) que chamam a atenção para o fato de que toda investigação científica começa pelo problema, não pela teoria, nem pela observação, nem por premissas, 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 desta perspectiva, as próprias características essenciais dos ambientes de manufatura caracterizados pelas novas estruturas organizacionais, entre as quais, por exemplo: 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 para proteger dados pessoais e outros ativos de conhecimento dos inúmeros evolvidos; sugerem o uso de novas abordagens baseadas na disponibilidade de tecnologia de informação e comunicação (HENDERSON-SELLERS e GIORGINI, 2005).

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 e THOMPSON, 2006).

Na literatura, os sistemas multiagente destacam-se ainda como tecnologias adequadas ao desenvolvimento de ambientes para a gestão do conhecimento (SHEN et al., 2001; GANDON, 2002; TACLA e BARTHÈS, 2003; 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 em primeiro lugar, a técnica de raciocínio baseado em casos RBC[4]. 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.

Conceitualmente, o RBC pode ser entendido como uma técnica para a solução de problemas, na qual a idéia fundamental subjacente é que em determinados domínios, em particular, os problemas a serem resolvidos tendem a ser recorrentes e repetirem-se com pequenas variações em relação à sua versão inicial. E, portanto, soluções prévias podem ser reaplicadas também com pequenas modificações (KOLODNER, 1993, AAMODT e PLAZA, 1994; WATSON, 2003; von WANGENHEIM e von WANGENHEIM, 2003).

Do ponto de vista científico, o raciocínio baseado em casos fundamenta-se nos modelos da ciência cognitiva, em particular, na teoria da memória dinâmica e pacotes de organização de memória (PMO's) proposta por Schank (1982).

Neste contexto, entre os estudos da área 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 (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. E, ainda, 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, entre as quais pode-se citar: sistemas de gestão do conhecimento, integração inteligente de informações,

[4] Do termo em inglês *Case-Based Reasoning*.

acesso baseado em semântica para a Internet e sistemas multiagentes (ABDULLAH et al., 2006).

Em relação a este tema, em particular, Dittmann et al. (2004) e Lee (2001) 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. E neste sentido, estas fontes de conhecimento dizem respeito não somente aos diferentes processos de manufatura e organizações envolvidas na cadeia produtiva, como mostra a figura 1.1, mas também às diferentes perspectivas de investigação e de análise dos problemas de não-conformidades.

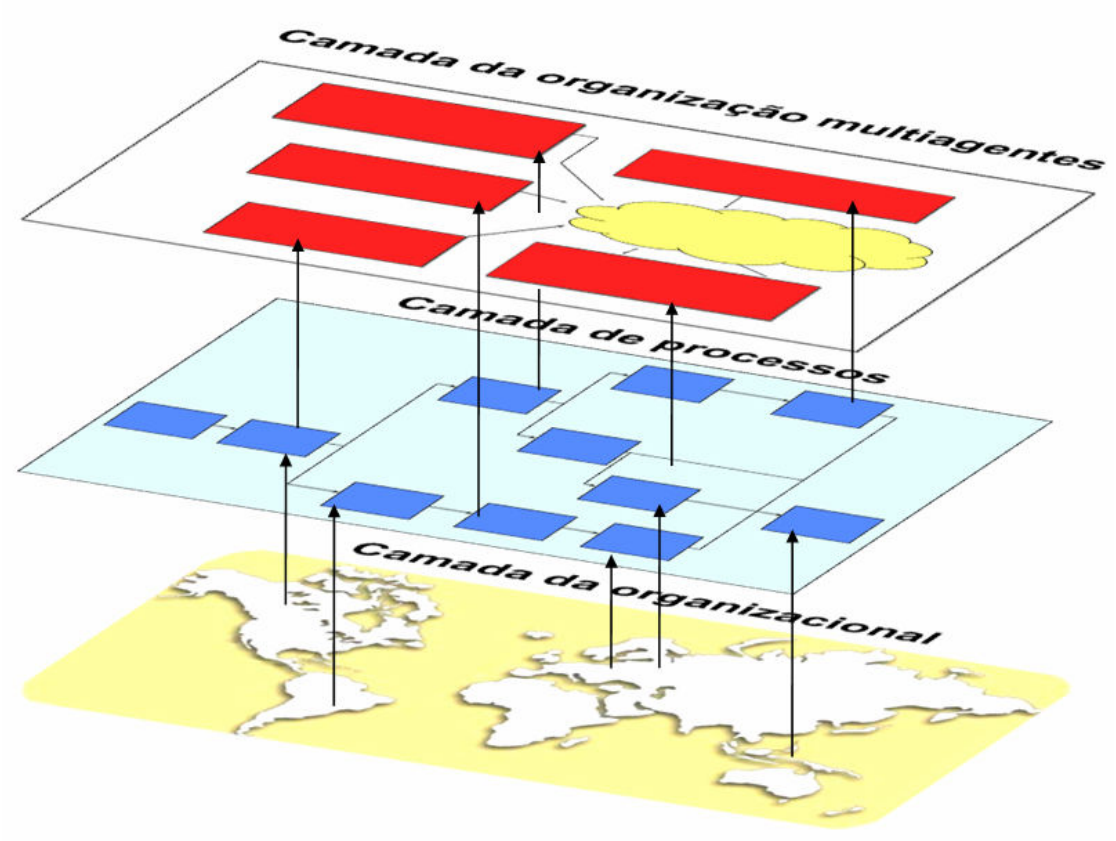


Figura 1.1 - Representação das camadas envolvidas no modelo.

E em decorrência da questão de pesquisa principal e da própria hipótese de trabalho, surgem outras questões mais específicas:

- Como conceitualizar, formalizar e codificar os conhecimentos identificados, na forma de bases de conhecimento, de modo a torná-los acessíveis tanto aos agentes informações computacionais quanto a especialistas humanos envolvidos no processo?
- Como modelar o comportamento dos agentes de informação considerando os métodos de raciocínio baseado em casos e os métodos de recuperação baseados em ontologias?

1.3 JUSTIFICATIVA

A validade e a relevância da pesquisa fundamentam-se particularmente nas agendas e recomendações de pesquisas estratégicas apresentadas nos relatórios: *Visionary Manufacturing Challenges for 2020* (NATIONAL RESEARCH COUNCIL, 1998) e *MANUFUTURE vision for 2020* (MANUFURUTE, 2004), bem como no trabalho de Gupta e Sharma (2003) *Intelligent Enterprises of 21st Century.*

Estas agendas, entre outras recomendações, sugerem, em particular, o desenvolvimento sistemático de sistemas para a captura e o compartilhamento de conhecimento de manufatura como suporte ao conceito de manufatura baseada em conhecimento (*Knowledge-Based Manufacturing*), em contraste ao conceito de manufatura baseada unicamente em recursos (*Resource-based Manufacturing*) (MANUFURUTE, 2004).

Esta recomendação, em especial, propõe investigar o desenvolvimento de repositórios de conhecimento de manufatura que possam ser acessados e compartilhados por plataformas de conhecimento e redes de competências interorganizacionais.

Neste sentido, estas agendas sugerem monitorar e explorar pesquisas e desenvolvimentos de tecnologias com significativos investimentos fora do setor de manufatura e empreender pesquisas para adaptá-las às aplicações de manufatura. Como no caso das tecnologias de informação que podem ser adaptadas e incorporadas em modelos de colaboração para melhoria de métodos de tomadas de decisão.

Dento desta lógica, os agentes de informação, tipicamente, podem acessar fontes de conhecimentos múltiplas, heterogêneas e geograficamente distribuídas por meio de redes de computadores (*Internet* ou *Intranet*) (KLUSCH, 2001; PECHOUCEK e THOMPSON, 2006). Esta funcionalidade inclui: recuperação, análise, manipulação e integração de informações a

partir de fontes de informação autônomas, distintas e distribuídas geograficamente (KLUSCH, 2001).

Pois, de acordo com Ferber et al. (2003), os sistemas multiagentes podem ser considerados como sociedades ou organizações multiagentes, isto é, como conjuntos de agentes computacionais que interagem mutuamente para coordenar o seu comportamento e que, ainda, com freqüência, devem cooperar para atingir algum objetivo comum.

Deste modo, portanto, uma organização multiagente pode ser considerada com uma abordagem cientificamente válida para promover o compartilhamento e reuso de conhecimento relacionado à solução de problemas de não-conformidades como uma alternativa à fragmentação e distribuição que acontece ao longo da cadeia produtiva.

1.4 OBJETIVOS DO TRABALHO

1.4.1 Objetivo geral

Propor um modelo baseado em organizações multiagentes capaz de apoiar a solução de problemas de não-conformidade em processos de manufatura com recursos distribuídos, visando o compartilhamento e a recuperação de conhecimentos decorrentes tanto da solução de não-conformidades prévias quanto da aplicação do método preventivo de Análise de Modos de Falha e seus Efeitos.

1.4.2 Objetivos específicos

Dentre os objetivos específicos, destacam-se os seguintes:

- Desenvolver um modelo conceitual da organização multiagente explorando as inter-relações entre as diferentes camadas de distribuição das fontes de conhecimento como mostradas na figura 1.1;
- Definir e desenvolver a conceituação e a formalização das bases de conhecimento dos agentes propostos na organização;
- Desenvolver uma especificação de projeto do modelo da organização multiagente compatível com os padrões 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.

- Identificar e selecionar os recursos de *software* necessários à implementação do modelo proposto de acordo com as especificações;
- Desenvolver e construir um protótipo de laboratório do modelo proposto de acordo com as especificações;
- Instanciar as bases de conhecimento com conhecimentos a respeito do processo em questão, e avaliar a extensão na qual o modelo proposto é válido.

1.5 DELIMITAÇÃO DO TRABALHO

É importante destacar que embora a etapa de modelagem conceitual da organização multiagentes prevista na tese possa ser conduzida em um alto nível de abstração e, portanto, independente de um domínio de aplicação em particular, as etapas de especificação, implementação e verificação e validação do modelo proposto, 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 verificação e validação.

Assim, nesta tese, foi escolhido o processo de moldagem por injeção de termoplásticos, 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.

Nesta linha raciocínio, o processo de moldagem por injeção é uma das mais versáteis tecnologias de processamento de materiais termoplásticos usados na indústria. E, segundo Chen e Turng (2005), em termos tecnológicos, é capaz de produzir, em massa, peças técnicas complexas já em sua forma final e dentro de estreitas especificações de tolerâncias dimensionais e geométricas.

Ainda, de acordo com a ABIPLAST (2006), no Brasil, o processo de moldagem por injeção representa 18 % do mercado de plásticos, quando considerada uma segmentação por processo de transformação, em um mercado com faturamento do setor de artefatos transformados estimado em 18.661 (Milhões de US$).

Neste cenário, em particular, destaca-se que a indústria de transformação de plástico representa o cerne do setor de plásticos, o qual pode ser considerado como um dos maiores

setores industriais tanto no Brasil quanto nos Estados Unidos e Europa (SPI, 2007; APM, 2007).

Em termos da indústria de transformação, o processo de moldagem por injeção faz parte de uma grande cadeia produtiva (cadeia de transformação de plástico) envolvendo: empresas produtoras de equipamentos, de moldes, de resinas; empresas transformadoras e uma ampla gama de clientes como empresas: automobilísticas e fabricantes de eletro-eletrônicos, entre outros (FLEURY e FLEURY, 2000).

Em termos de produção, esta cadeia envolve a manufatura de produtos plásticos acabados e semi-acabados ou, ainda, operações adicionais de acabamento, tais como: pintura, acabamentos superficiais ou montagens de componentes, direcionadas para outros setores industriais, tais como: embalagens, construção, automotivo, eletro-eletrônico e saúde (SPI, 2007; APM, 2007).

Por outro lado, do ponto de vista tecnológico, a produção de componentes por meio do processo de moldagem por injeção envolve um conjunto complexo de fatores relacionados ao: projeto do produto ou componente, o projeto do molde, a seleção da resina termoplástica, seleção da máquina e a definição e o ajuste de parâmetros do processo (REES, 2002; GOODSHIP, 2004; STEIL, 2004; CHEN e TURNG, 2005, MANRICH, 2005).

Não obstante todos os recursos científicos e tecnológicos empregados, a probabilidade de ocorrência de não-conformidades no processo de moldagem não pode ser completamente eliminada. E, em geral, tais não-conformidades decorrem da complexa inter-relação entre o produto e o molde, entre a resina e processo, os 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 (GOODSHIP, 2004).

1.6 METODOLOGIA DA PESQUISA

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 e quanto aos procedimentos técnicos.

Quanto à natureza, esta tese se enquadra como uma pesquisa aplicada, segundo a linha de Silva e Menezes (2000), 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 Silva e Menezes (2000), 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 Silva e Menezes (2000), pois a pesquisa busca explorar o uso da abstração de agentes computacionais em apoio à solução de problemas de não-conformidade a partir de diferentes fontes de conhecimento.

Quanto aos procedimentos técnicos, conforme Silva e Menezes (2000), 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.

Por fim, a figura 1.2 apresenta o roteiro da pesquisa.

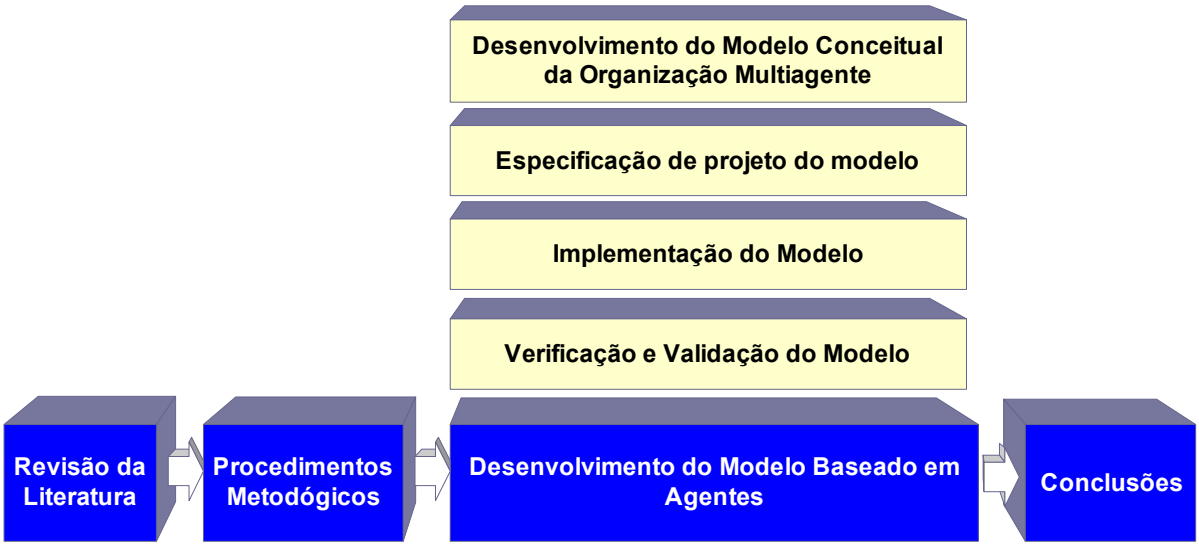


Figura 1.2 – Roteiro de pesquisa.

1.7 ESTRUTURA DA TESE

Para atingir os objetivos propostos esta tese é estruturada em 9 capítulos. O Capítulo 1, introdutório desta pesquisa, descreve a origem do trabalho e o problema de pesquisa e sua delimitação, e em seguida apresenta os objetivos gerais e específicos propostos, a caracterização e o roteiro da pesquisa e, por fim, a estrutura da tese.

O Capítulo 2 apresenta a revisão da literatura que fundamenta a pesquisa.

O Capítulo 3, por sua vez, é devotado a descrever os procedimentos metodológicos que fundamentam cientificamente a pesquisa empreendida pelo autor para o desenvolvimento do modelo baseado em agentes proposto. Nestes procedimentos são previstos o desenvolvimento

do modelo conceitual da organização multiagente, a especificação deste modelo em termos do projeto de uma organização multiagente abstrata e a implementação deste modelo, de modo a permitir a verificação e validação da organização multiagente proposta.

O Capítulo 4 descreve a pesquisa realizada pelo autor no tocante ao desenvolvimento do modelo conceitual da organização multiagente proposta e a sua representação em termos de uma especificação de projeto. Adicionalmente, este capítulo propõe e discute o desenvolvimento das bases de conhecimento dos agentes propostos.

O Capítulo 5 apresenta os detalhes do processo de escolha dos recursos de *software* destinados ao desenvolvimento do protótipo de laboratório que implementa o modelo proposto.

O Capítulo 6 apresenta e discute os detalhes do desenvolvimento formal do protótipo computacional do modelo proposto. E, o capítulo 7, por sua vez, concentra-se na forma de implementação e operacionalizações das bases de conhecimento dos agentes devotados ao raciocínio e recuperação de conhecimento.

O Capítulo 8 destina-se a apresentar os detalhes e discutir os resultados da verificação, validação conceitual e operacional do modelo proposto, considerando as especificações de projeto tal como se acham estabelecidas no Capítulo 4 e em consonância aos procedimentos metodológicos apresentados no Capítulo 3.

Finalmente, o Capítulo 9 apresenta as discussões, conclusões, contribuições do trabalho, além das recomendações para trabalhos futuros.

CAPÍTULO 2
REVISÃO DA LITERATURA

Este capítulo apresenta na primeira parte uma síntese da construção histórica da noção de qualidade ao longo dos últimos anos, buscando demonstrar que a evolução desta noção ocorre em resposta às mudanças na própria estrutura organizacional das empresas de manufatura impostas por pressões sociais e econômicas.

Este enfoque mostra ainda que as abordagens para os sistemas de gestão da qualidade e, em especial, para a análise e solução de problemas de não-conformidades representam um papel essencial para a competitividade das empresas de manufatura. E, portanto, pesquisas devem ser empreendidas visando estendê-las às novas tendências organizacionais.

E na segunda parte deste capítulo, apresentam-se os principais conceitos subjacentes à pesquisa e que visam o entendimento da abordagem proposta neste trabalho de tese. E, neste sentido, concentra-se nos aspectos fundamentais da tecnologia de agentes, representação de conhecimento e raciocínio.

2.1 EVOLUÇÃO E DESAFIOS CONCEITUAIS À GESTÃO DA QUALIDADE

Os conceitos e as práticas 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. E este reconhecimento pode ser observado claramente nas contribuições e trabalhos da área, tais como em Deming (1982), Juran (1988), Feingenbaum (1991), Crosby (1984), Ishikawa (1985), Taguchi (1980) e Pfeifer (1996).

Assim, da ótica da construção histórica, a noção de qualidade evoluiu de forma extraordinária no último século, partindo da idéia de que esta poderia ser alcançada essencialmente por meio de técnicas de inspeção orientada ao produto (*product-oriented measurement*) passando pela noção de controle (*control devices*), 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., 2002; FOSTER e JONKER, 2003; 2007).

Dentro desta linha de pensamento, a literatura revela mais recentemente as abordagens voltadas à garantia da qualidade, as quais expandiram o escopo do controle da qualidade convencional revelando a importância de outras funções da organização para a qualidade, em

adição às 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 (FOLEY, 2000; 2001; SLACK et al., 2002; FOSTER e JONKER, 2003; 2007).

Já nos anos oitenta, por sua vez, foram propostos modelos de referência para a elaboração de sistemas de gestão de qualidade em consonância às 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).

Culminando com a conhecida abordagem de gestão da qualidade total[1], que traduzia 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 (FOLEY, 2000; 2001; SLACK et al., 2002; FOSTER e JONKER, 2003; 2007; FNQ[2], 2006).

Todavia, cumpre ressaltar que mesmo após vinte cinco anos de aplicações industriais e pesquisas desenvolvidas no ocidente, a gestão da qualidade não possui uma teoria universalmente aceita ou mesmo uma descrição (FOLEY, 2000; 2001).

Neste cenário, Foster e Jonker (2007) apresentam uma análise profunda das mudanças que ocorreram na gestão da qualidade visando demonstrar que uma nova mudança está em curso, e ao mesmo tempo buscam estabelecer um embasamento teórico e científico para caracterizar esta mudança.

Foster e Jonker (2007) argumentam que a evolução dos conceitos e práticas ocorre em resposta às mudanças na própria concepção das organizações, e sugerem que a gestão da qualidade inicia agora a sua terceira geração e apresentam um quadro teórico para comparar esta nova geração às anteriores, tendo como base as seguintes características: perspectiva da qualidade, foco, tipo de ação, critério de sucesso, orientação, premissas básicas, relação entre *stakeholders*[3], características do engajamento, natureza conceitual e cultura, como mostra o tabela 2.1.

---

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

[2] Conceitos Fundamentais da Excelência em Gestão da Fundação Nacional da Qualidade

[3] Adota-se nesta tese a definição de *Stakeholder* como pessoa, grupo de pessoas e outras entidades, os quais possuem interesses nas ações e operações das organizações, podendo afetar ou serem afetados por elas.

Tabela 2.1 – Características das três gerações da gestão da qualidade.

1ª Geração da Gestão da Qualidade	2ª Geração GQ	3ª Geração GQ
Perspecitva da qualidade		
Processo	Holística	Relacional
Foco		
Inspeção	Avaliação	Entendimento
Tipo de ação		
Reativa	Proativa	Engajamento
Critério de sucesso		
Confiabilidade	Eficiência e Efetividade	Contabilidade e Transparência
Orientação		
Produção	Processos	Relacionamentos
Premissas básicas		
Controle	Gerenciamento	Interconectividade
Mudanças		
Melhoria	Mudança	Transformação e transação
Relação entre stakeholders		
Não existentes	Periférica	Incorporadas
Características do engajamento		
Não existente	Envolvimento	Complementariedade
Natureza conceitual		
Ferramentas e técnicas	Técnicas, métodos e princípios	Teoria organizacional
Cultura		
Irrelevante	Unidade de igualdade	Unidade de diversidades

(FONTE: Adaptado de FOSTER e JONKER, 2007).

O trabalho de Foster e Jonker (2007) sugere que a terceira geração da gestão da qualidade é fundamentalmente diferente das anteriores, devido a um maior entendimento sobre a importância do papel dos *stakeholders* no comportamento e sobrevivência organizacional. E, neste sentido, os autores baseiam-se na premissa de que a realidade organizacional é construída socialmente apoiando-se na concepção de "nexo de contrato" (que considera as organizações como um conjunto de contratos). Esta concepção presume a necessidade de reciprocidade de ação e o envolvimento de uma grande variedade de *stakeholders* que compreende todos os participantes de um arranjo interorganizacional.

Para Foster e Jonker (2007) a ênfase deve concentrar-se na atitude e complexidade cognitiva do engajamento com os diversos *stakeholders* destes arranjos. Todavia, concretizar esta fundamental reconceitualização, em termos de novos conceitos, métodos e técnicas, pode de fato ser vista como um desafio organizacional para as próximas décadas.

2.1.1 Desafios conceituais à gestão da qualidade nas novas estruturas organizacionais

Como apresentado no capítulo introdutório, novas estruturas organizacionais têm sido reportadas na literatura. Estas estruturas, em essência, podem ser atribuídas a complexos fatores sociais e econômicos caracterizadas pela globalização dos mercados, redução do ciclo de vida dos produtos, alta variabilidade na demanda, necessidade de alta flexibilidade e reatividade e rápido desenvolvimento das tecnologias de informação e comunicação (ZAIDAT e al., 2005).

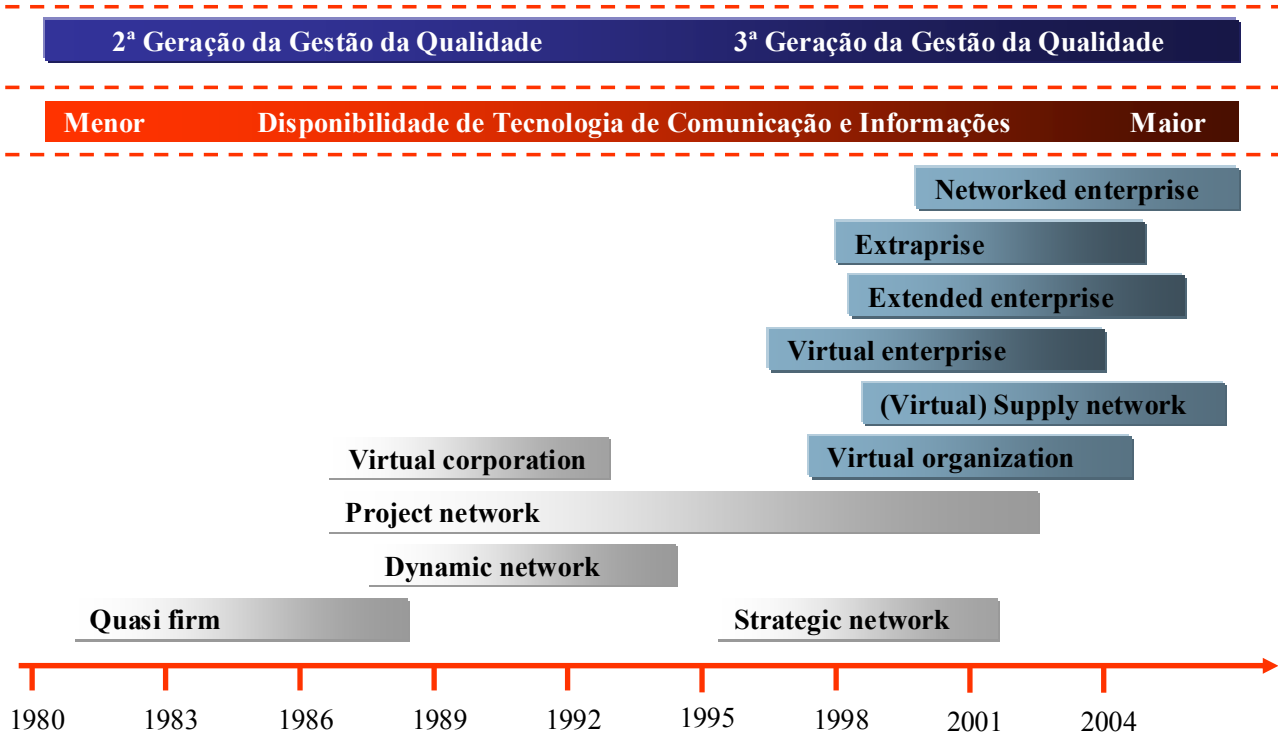


Figura 2.1 – Principais termos usados na literatura para descrever as formas de organização.

A figura 2.1 apresenta um quadro posicionando em uma escala de tempo os principais termos usados na literatura para descrever as novas formas de organização. Estes termos foram apresentados, entre outros, nos trabalhos de Camarinha-Matos e Afsarmanesh (1999), Wiendahl e Lutz (2002), Zaidat et al., (2005), Cecil et al. (2006) e Binder e Clegg (2007). Adicionalmente, foi incluída a estas novas formas de organização, usando-se a mesma linha de tempo, a noção das gerações de gestão da qualidade proposta por Foster e Jonker (2007).

Dentro desta perspectiva e considerando o tema desta tese, uma revisão da literatura revela abordagens com o uso de ferramentas de gestão da informação em apoio à gestão da qualidade. Estas abordagens têm o propósito de capturar as informações da qualidade a partir de fontes internas e externas à organização visando um ambiente de comunicação que possibilite o compartilhamento de informações relativas à qualidade entre organizações,

clientes e fornecedores por meio da *intranet/internet* (TSUNG, 2000; TANG e LU, 2002; CHIN et al., 2006).

Entretanto, embora grandes avanços tenham sido relatados na literatura em relação a sistemas em apoio à decisão e novas tecnologias de informação e comunicação, a proposição de sistemas efetivos em apoio à decisão para questões estratégicas da qualidade, tais como para ações preventivas e corretivas no âmbito de sistemas de gestão da qualidade, mesmo na manufatura tradicional, ainda representa um grande desafio como constatado por Lari (2003), e em particular devido a questões subjacentes, tais como:

- A tomada de decisões em relação a ações corretivas e preventivas no âmbito da melhoria da qualidade são processos estruturados que podem demandar várias horas de trabalho; contudo, situações urgentes, em geral, requerem respostas rápidas;
- Representa um esforço em grupo que envolve trabalho cooperativo para a solução de problemas, resolução de conflitos, negociação e construção de consenso;
- Envolve múltiplos interessados, os quais têm diferentes premissas que precisam ser tornadas explícitas;
- É um processo evolucionário que requer o conhecimento acumulado a partir de casos anteriores já estudados;
- E podem requer uma grande variedade de ferramentas analíticas, variando desde simples diagramas de causa e efeito até sofisticados sistemas de inteligência artificial.

Assim, as decisões referentes às ações preventivas e corretivas, bem como seu respectivo gerenciamento requerem um sistema de informações capaz de tratar um ciclo composto pela identificação, análise, resolução e controle dos resultados dos problemas de qualidade (LARI, 2002).

Portanto, a atividade de gestão deste conhecimento deve ser considerada estratégica e de alta prioridade para todas as organizações de manufatura. Entretanto, em muitas organizações, mesmo com sistemas de manufatura tradicionais, a distribuição física e a forma de registro (papel ou bases de dados) dificultam enormemente a tarefa de reutilização efetiva deste capital valioso (ADAMANTIOS e ROCK, 2002).

2.1.2 Gestão do conhecimento relacionado à qualidade no contexto das novas estruturas

Em primeiro lugar, é importante observar que o conceito de conhecimento vem sendo discutido por filósofos desde os antigos gregos, por pesquisadores da área de teoria dos

sistemas e tecnologia da informação e ainda não está totalmente desmistificado (BRACHMAN e LEVESQUE, 2004).

Assim, o conceito empregado nesta tese refere-se à noção de conhecimento com "recurso" ou ativo que pode existir fora da mente humana. E, portanto, pode ser representado e manipulado como qualquer outro objeto por ferramentas de gestão do conhecimento, na linha de pensamento da teoria de estoque (*Stock-theory*) (van ENGERS, 2001).

Da mesma forma, em relação à gestão do conhecimento, não há um consenso na literatura sobre uma definição de gestão do conhecimento, embora exista um forte e indubitável interesse sobre o tema. Todavia, é importante observar que seus objetivos, em geral, são estruturados em três aspectos (ERMINE, 2000): capitalizar o conhecimento individual; compartilhar (alternar da noção de inteligência individual para a inteligência coletiva) e criar ativos tangíveis (antecipar e inovar buscando a sobrevivência da organização). A figura 2.2 apresenta um quadro com os principais termos usado na literatura para descrever o processo de gestão do conhecimento (CHEN e CHEN, 2006).

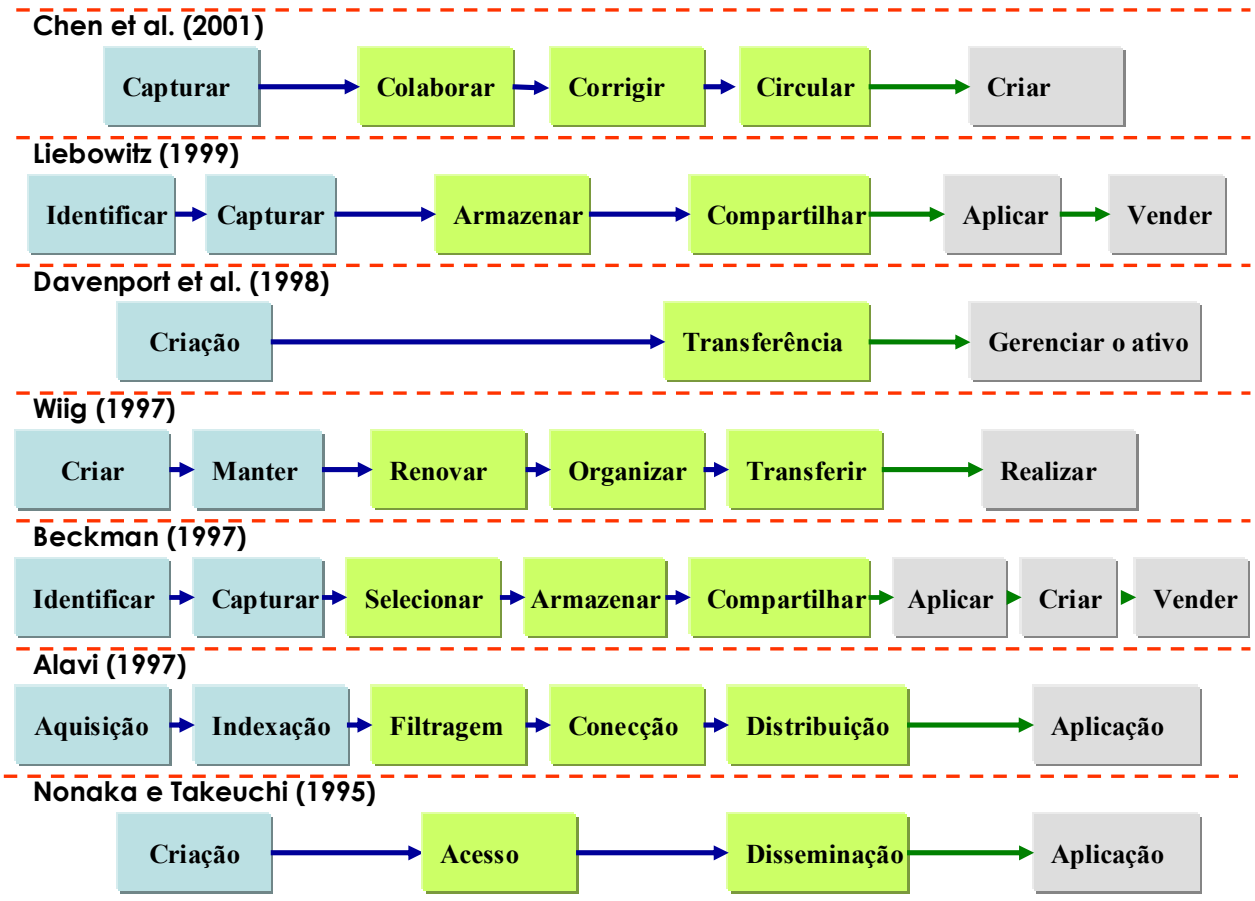


Figura 2.2 – Principais termos usados na literatura para descrever o processo de Gestão do Conhecimento (FONTE: CHEN e CHEN, 2006).

Neste cenário, a questão fundamental da gestão do conhecimento reside em como melhorar a identificação, aquisição, armazenagem, acesso, difusão, reutilização e manutenção do conhecimento interno e externo em uma organização (ERMINE, 2000). Nesta linha, uma possível abordagem para a gestão do conhecimento em uma organização é constituir uma memória organizacional ou corporativa (ERMINE, 2000). Assim, um sistema de gestão de conhecimento e, em particular, uma memória organizacional ou corporativa deve ser capaz de abastecer as pessoas envolvidas com elementos relevantes de conhecimento no tempo apropriado, com o nível de detalhes necessário e em um formato adequado (ERMINE, 2000).

A literatura revela também a noção de capital intelectual associado à qualidade e a confiabilidade dos produtos e serviços de uma organização. Este capital compreende aspectos como lições aprendidas (*lessons learned*), ações corretivas tomadas, melhores práticas, dados relativos à percepção do cliente e retro-alimentação em relação aos produtos, além de dados precisos sobre a confiabilidade de componentes e subsistemas (ADAMANTIOS e ROCK, 2002).

Em relação à memória organizacional, Gandon (2002) sintetiza três aspectos que são apresentados na literatura para definir a memória organizacional (ou corporativa): o conteúdo da memória ("o quê"), o qual diz respeito à soma de todos os recursos e ativos de conhecimento e informações no âmbito de uma organização, isto é, a natureza do conhecimento; a forma da memória ("onde"), que descreve o modo de representação formal do conhecimento visando seu processamento apropriado por sistemas computacionais, isto é, o suporte técnico para armazenagem; o modo de operação da memória ("como"), isto é, como a memória torna os ativos de conhecimento disponíveis para os processos de uma organização que demandam uso intensivo de conhecimento.

Adicionalmente, Gandon (2002) apresenta uma definição de memória organizacional ou corporativa como uma representação e indexação explícita e persistente de conhecimentos e informações ou de suas fontes, cujo propósito é facilitar o acesso, compartilhamento e a reutilização pelos membros de uma organização para execução de suas tarefas individuais ou coletivas.

Do ponto de vista técnico, na literatura são relatadas diversas formas possíveis para se constituir uma memória corporativa e, em particular pode-se citar: a memória baseada em casos como uma das principais, na qual a coleção de experiências passadas (de sucesso ou de insucesso) de uma organização pode ser representada, de forma explícita e apropriada, bem como exploradas efetivamente por meio da técnica de inteligência artificial denominada raciocínio baseado em casos (WATSON, 2003).

A literatura revela ainda que a tecnologia de agentes[4] pode ser considerada como uma possível alternativa para o gerenciamento de memórias corporativa, como demonstram os projetos: CoMMA - *A multi-agent system for **C**orporate **M**emory **MA**nagement* e FRODO - *A FRamework for **O**pen/**D**istributed constraint **O**ptimization* apresentados por Bergenti et al., (2002) e Abecker et al. (2003) respectivamente.

2.1.3 Desafios conceituais à análise e solução de problemas de não-conformidades

Em primeiro lugar, é importante ressaltar as diferentes terminologias e conceitos encontrados na literatura no tocante ao termo "não-conformidade".

De acordo com a norma ISO 9000 (2005), que estabelece os fundamentos e o vocabulário próprio para os sistemas de gestão da qualidade passíveis de certificação de internacional, uma não-conformidade é definida como "o não atendimento a um requisito". A norma define, por sua vez, um requisito como "uma necessidade ou expectativa que é expressa, geralmente, de forma implícita ou obrigatória" e, de modo 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 (ISO 9000, 2005). Neste sentido, é importante notar a diferença entre os conceitos "defeitos" e "não-conformidade", onde o defeito tem uma conotação legal, particularmente em relação à responsabilidade civil pelo fato do produto, a qual se dá diante de situações que põem em risco a saúde e segurança do consumidor. E, ainda, que o uso pretendido pelo cliente pode ser afetado pela natureza da informação disponibilizada pelo fornecedor, tais como: instruções de operação ou manutenção.

Por outro lado, a norma NBR 5462 (1994), referente à confiabilidade e mantenabilidade, define os termos: defeito, falha, pane e erro. No contexto desta norma, o termo defeito é definido como "qualquer desvio de uma característica de um item (qualquer parte, componente, dispositivo, subsistema, unidade funcional, equipamento ou sistema que possa ser considerado individualmente) em relação a seus requisitos".

Ela estabelece ainda que "os requisitos podem, ou não, ser expressos na forma de uma especificação", e também que "um defeito pode, ou não, alterar a capacidade de um item em desempenhar uma função requerida". Isto é, o termo defeito não subentende a perda total da

[4] A tecnologia de agentes é apresentada em detalhes na subseção 2.3.

função principal do item, o qual pode executar a sua função, embora possa não apresentar as características ideais de funcionamento (NBR 5462, 1994).

Por sua vez, uma falha é definida como o "término da capacidade de um item desempenhar a função requerida". E neste sentido a falha pode ser entendida como um evento que determina o estado de pane de um item. E, por extensão, pane é "o estado de um item, caracterizado pela incapacidade de desempenhar uma função requerida, excluindo-se a incapacidade durante os períodos de manutenção preventiva ou outras ações planejadas, ou pela falta de recursos externos".

No tocante ao conceito de falha, a norma SAE J-1739 (2002), relativa ao método de Análise de Modos de Falhas e seus Efeitos (FMEA), define também o termo modo de falha como a maneira pela qual um componente, subsistema, ou sistema (item) poderia, potencialmente, falhar em atender seus requisitos de projeto. E esta definição ressalta, ainda, a relação entre causas, modos de falha e efeitos, em função do nível de análise, pois um modo de falha pode simultaneamente ser considerado uma causa de outro modo de falha em um subsistema imediatamente superior dentro da hierarquia do sistema e o efeito de outro modo de falha em um nível mais baixo, portanto, o modo de falha é uma conexão em uma cadeia de causa e efeito associada a uma determinada falha (SAE, 2002).

O termo "erro" é definido como a diferença entre um valor (uma condição observada ou o resultado de uma medição) e seu respectivo valor verdadeiro convencionado ou teórico, pode ser decorrente de um item em pane ou, ainda, representar um erro humano (SAE, 2002).

Por sua vez, uma ação corretiva é definida pela norma ISO 9000 (2005) 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 critica 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 (ISO, 2005).

Uma ação preventiva é definida como uma ação para eliminar a causa de uma não-conformidade potencial, cujo objetivo é prevenir a ocorrência desta não-conformidade, enquanto a ação corretiva diz respeito a prevenir a repetição de uma não-conformidade (ISO, 2005).

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

- Descrição da não-conformidade estabelecida pelos atributos que envolvem as circunstâncias, nas quais a não-conformidade ocorreu, sua importância e urgência;
- Descrição dos parâmetros que determinam a sua atribuição a um dado componente, subproduto ou produto;
- Descrição dos elementos do processo de análise da não-conformidade, isto é, os sintomas coletados, as causas possíveis e as reais, as ações possíveis e reais tomadas no caso;

Portanto, o tratamento efetivo das não-conformidades por meio de ações corretivas e, em especial, ações preventivas, bem como seu respectivo gerenciamento, é parte estratégica e essencial do processo de melhoria contínua da qualidade, como apontado nos seguintes itens:

- Sistemas de gestão da qualidade estabelecidos de acordo com a norma internacional ISO 9001:2000 – Sistemas de gestão da qualidade: Requisitos;
- Sistemas de gestão que incorporam aspectos da norma ISO 9004:2000 – Sistemas de gestão da qualidade: Diretrizes para melhoria de desempenho;
- Sistemas de gestão fundamentados em modelos mais sofisticados como, por exemplo, Conceitos Fundamentais da Excelência em Gestão (FNQ, 2006), que reflete as melhores práticas em termos gestão da qualidade encontradas nas organizações de classe mundial.

Nesta linha, estudos realizados em empresas alemãs com sistemas de manufatura tradicionais dos setores metal-mecânico e químico financiados pelo governo no âmbito do Projeto FOQUS coordenado pela Universidade de Aachen, demonstraram 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).

Na época, o propósito do projeto foi 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 informação integrado ao processo de trabalho de uma organização (PFEIFER, 1997).

Por sua vez, Dhafr et al. (2006) discutem 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 deste modelo é, em essência, assistir aos envolvidos com um conjunto de estados através do 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 2.3.

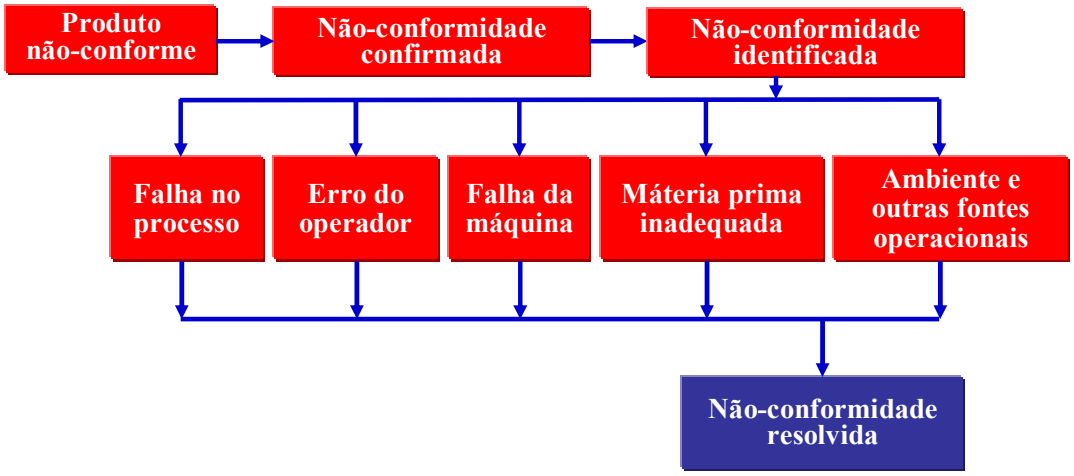


Figura 2.3 – Ciclo de uma não-conformidade (FONTE: DHAFR et al., 2006).

O modelo deve incluir na visão de Dhafr et al. (2006) considerações sobre a ocorrência da não-conformidade a partir de: falhas na máquina, erros dos operadores e qualquer outra fonte operacional. E, ainda, se medidas baseadas no *status* da não-conformidade forem coletadas, estas podem ser usadas para a aprendizagem, e assim melhorar o desempenho da linha de produção.

2.1.3.1 Conceitos do processo de análise e solução de problemas de não-conformidades

De acordo com a argumentação apresentada no capítulo introdutório, a solução de problemas de não-conformidades em processos de manufatura permanece ainda um desafio tanto do ponto de vista acadêmico quanto industrial, pois, não obstante todos os recursos científicos e tecnológicos empregados, a probabilidade de ocorrência de não-conformidades nos processos de manufatura não pode ser completamente eliminada, e a solução destes problemas, em geral, caracteriza-se por envolver atividades intensivas em conhecimento, e baseadas essencialmente em experiências.

Neste contexto, na literatura são apresentados diferentes métodos de análise e solução de problemas, e em especial podem ser citados os trabalhos de Kume (1993) Sipper e Bulfin (1997), Kepner-Tregoe (1986) e Haviland (2004), bem como os Métodos 8D (CHRYSLER CORPORATION, 1997) e Método Toyota (LINKER, 2004) amplamente usadas no setor automotivo.

Campagnaro (2007) apresenta uma ampla revisão destes métodos e identifica as principais etapas dos modelos teóricos para análise e solução de problemas de não-conformidade:

- Identificar a não-conformidade. Esta etapa visa definir claramente o problema relacionado a uma dada não-conformidade e sua extensão, bem como verificar se esta é recorrente e quais ferramentas da qualidade que podem ser empregadas no auxílio da descrição da não conformidade.
- Ação interina (ação de contenção). Nesta etapa o objetivo é avaliar a extensão do problema e determinar as ações para conter a não conformidade se necessário.
- Identificar a causa raiz. Esta etapa visa descobrir as causas principais mediante ferramentas apropriadas, investigar a ocorrência da causa raiz potencial, definir a causa raiz final e testá-la para confirmação.
- Ações corretivas. Agir para eliminar a causa mediante a implementação de ações destinadas a lidar com os efeitos (resultados) e atualizar documentação de modo a permitir o acompanhamento sistemático das ações implementadas.
- Verificação das ações corretivas. Nesta etapa, em geral, ferramentas da qualidade devem ser usadas para auxiliar a obtenção e monitoração de dados que confirmem a eficácia das ações implementadas.
- Ações preventivas. Esta etapa concentra-se em prevenir recorrência do fato causador da não-conformidade, apoiando-se na melhoria contínua de processos, buscando estender os efeitos destas ações a produtos e processos similares.

Neste cenário, destaca-se em particular a referência normativa do setor automotivo para Solução Efetiva de Problemas[5] (AIAG, 2006), desenvolvido por um grupo de alto nível composto por diversos especialistas na área de análise e solução de problemas das empresas pertencentes ao grupo (AIAG), bem como de sua comunidade de fornecedores. Neste sentido, este referencial visa estabelecer um consenso sobre as metodologias referentes à análise e

[5] Tradução do título em inglês *The Effective Problem Solving Guideline* publicado pelo Grupo de Ação da Indústria Automotiva (*Automotive Industry Action Group*) (AIAG, 2006).

solução de problemas de não-conformidade e os principais conceitos usados atualmente no setor.

Neste referencial, a AIAG (2006) recomenda um processo para a solução efetiva de problemas de não-conformidades que compreende as seguintes etapas: a definição do problema, determinação da(s) causa(as) raiz(es) do problema, desenvolver ações corretivas para todas as causas identificadas e institucionalizar estas ações no âmbito da organização, como mostra a figura 2.4.

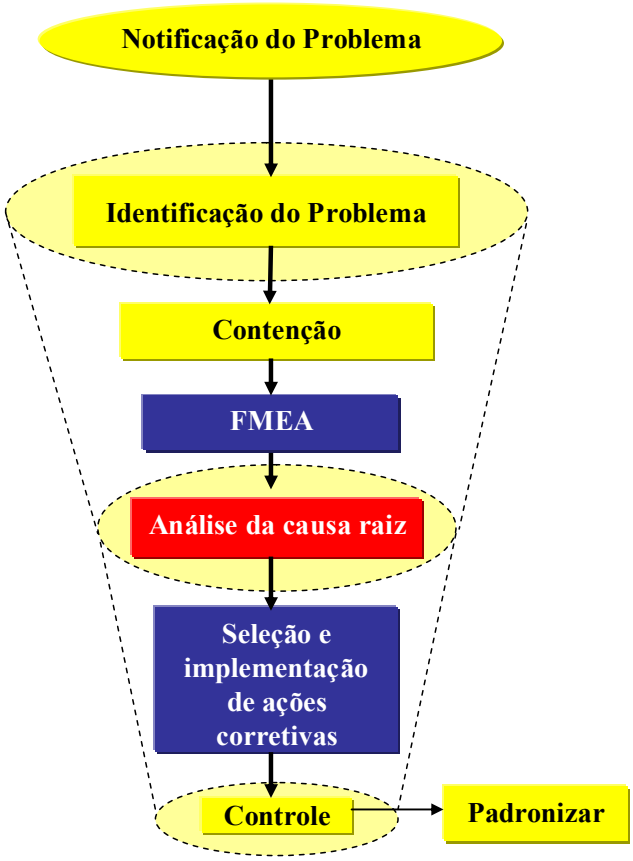


Figura 2.4 – Processo de solução efetiva de problemas (FONTE: AIAG, 2006).

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 às etapas a recomendação inovadora de consultas baseadas em sistemas de lições aprendidas, como mostram as figuras 2.5 e 2.6.

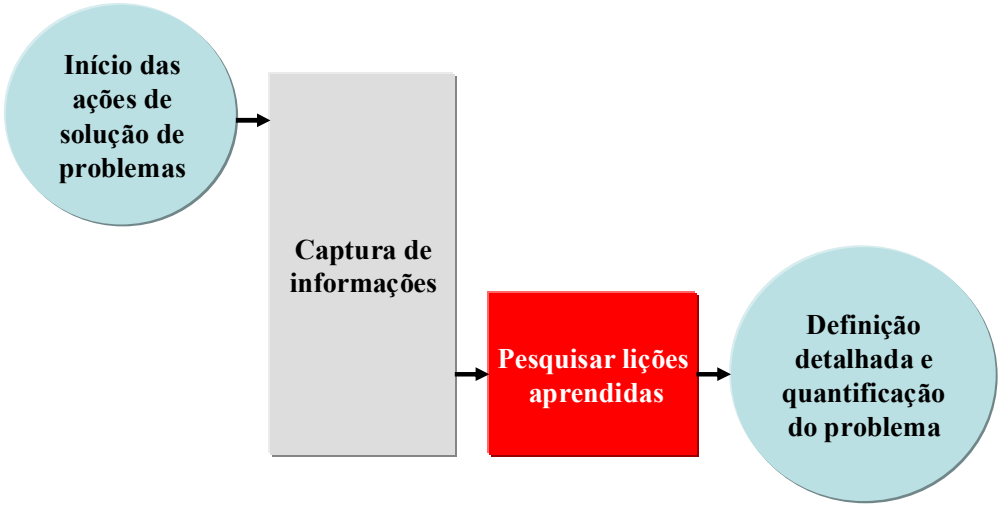


Figura 2.5 – Etapa de identificação do problema (FONTE: AIAG, 2006).

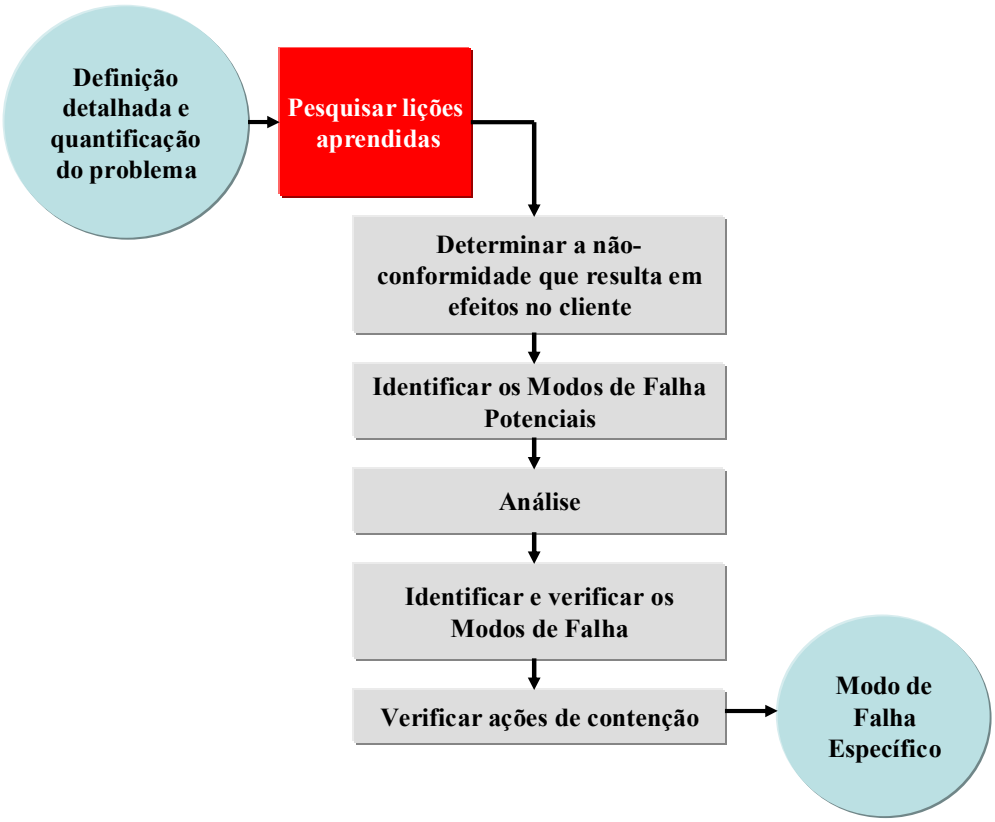


Figura 2.6 – Etapa de análise de modos de falha (FONTE: AIAG, 2006).

Assim, o processo de análise e solução de problema de acordo com esta normativa estende-se além da finalidade imediata de resolver um problema, mas envolve uma ampla gama de atividades de gestão do conhecimento. Neste sentido, a maior contribuição desta referência está justamente em ressaltar a importância de incorporar as experiências com a solução de um problema de não-conformidade, e em particular, em todos os produtos e

processos similares, em especial, na forma de uma abordagem consistente de lições aprendidas.

2.1.3.2 Desafios conceituais à representação do conhecimento na forma de lições aprendidas

Do ponto de vista da gestão do conhecimento, a definição de lições aprendidas (do inglês, *lessons learned*) adotada, de modo geral, pela comunidade envolvida foi proposta por Secchi et al. (1999) como um artefato de conhecimento, que corresponde ao entendimento obtido por meio da experiência, a qual pode ser positiva como o sucesso de um teste ou missão, ou negativa como um acidente ou uma não-conformidade (WEBER et al., 2001; 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 (WEBER et al., 2001; 2002).

Assim, um sistema de lições aprendidas representa uma iniciativa estruturada para gestão do conhecimento sobre um repositório de conhecimento constituído por lições aprendidas (WEBER et al., 2001; 2002).

A literatura relata o uso de sistemas de lições aprendidas em organizações governamentais, em particular, nos Estados Unidos da América: no departamento de defesa em aplicações militares; no departamento de energia onde a prevenção de acidentes é a maior preocupação, bem como nas agências espaciais americana, européia e japonesa, devido aos altos custos envolvidos com as falhas de uma missão (WEBER et al., 2002).

Uma lição aprendida é composta, do ponto de vista formal, por um conjunto de características classificados em dois grupos: os elementos indexadores e os elementos reutilizáveis (WEBER et al., 2002).

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) ao qual a lição é aplicável; as pré-condições que distinguem estado particular que determina quando esta lição é aplicável (WEBER et al., 2002).

Os elementos reutilizáveis englobam, em essência: uma sugestão de lição aprendida em função dos elementos indexadores e descrevem o que foi aprendido através da experiência que deveria ser repetido ou evitado; um argumento lógico (do inglês, *rationale*) que fornece

ao potencial usuário uma justificativa que particulariza como esta lição foi aprendida, e pode envolver os seguintes elementos: 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*) (WEBER et al., 2002).

Não obstante a ampla utilização de sistemas de lições aprendidas, um dos desafios no campo de pesquisa reside na dificuldade de transmitir lições entre o repositório de lições aprendidas e os usuários potenciais (WEBER et al., 2002).

Do ponto de vista da representação formal das lições aprendidas e considerando a limitação reconhecida na literatura, uma das possíveis alternativas é considerá-la como a representação de um caso, em termos da técnica de raciocínio baseado em casos (WEBER et al., 2002).

De forma simplificada, a representação de um caso consiste em dois componentes primários: o problema que descreve o estado do mundo quando o caso ocorreu, e a solução que prescreve como resolver o problema (WATSON, 2003).

Da ótica do raciocínio ou inferência, a descrição do problema é usada para indexar e governar a tarefa de recuperação em função das medidas de similaridade consideradas, enquanto a solução prévia é reutilizada para resolver o novo problema (WATSON, 2003). Desta forma, é possível 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, permitindo o uso da técnica de raciocínio baseado em casos para representar e manipular os elementos das lições aprendidas.

2.1.3.3 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 conseqüência, 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 falhas 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 de 1960, no contexto dos projetos aeroespaciais americanos, em particular, no projeto Apollo da NASA (*National Aeronautics and Space Administration*).

Posteriormente, este método 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 a Análise de Modos de Falhas e seus Efeitos (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-se: a uma montagem subseqüente ou sobre o sistema como um todo ou sobre um processo específico. E, 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) recomenda 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 compensadores. 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[6] denota que a análise do modo de falha produz também uma análise da criticalidade (*criticality*). Esta extensão lógica do método é definida na literatura como FMECA[7] (Análise de Modos de Falha, seus Efeitos e Criticalidade). 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 a classificação dos modos de falha considerando que uma medida de criticalidade quantifica a severidade e probabilidade de ocorrência deste modo de falha (IEC, 2006).

---

[6] *Failure Mode and Effects Analysis*

[7] *Failure Mode Effects and Criticality Analysis*

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 de este 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, 2001, 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 na área de manutenção e na área de manufatura, e sua aplicação é recomendada, e em alguns casos exigida em contratos, como um elemento essencial para os sistemas de gestão da qualidade, entre as quais:

- 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 ISO 9001:2000 para a produção automotiva e serviços relevantes e componentes de organizações fornecedoras (especificação técnica);

A figura 2.6 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, 2001).

É importante ressaltar que a aplicação do método FMEA é precedida de 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 bloco 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 2.7 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).

A função do processo ou operação dever ser expressa de forma concisa mediante o uso de um verbo ou a combinação de um verbo e seu respectivo complemento (STAMATIS, 2003). Nesta linha, um processo de moldagem por injeção de termoplásticos pode ser decomposto de acordo com o seu ciclo, e funções podem ser estabelecidas para cada etapa deste ciclo, por exemplo, para o ciclo do molde: fechar molde, preencher a cavidade, resfriar o molde, abrir o molde e ejetar o componente.

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, e pode ser a causa associada a um modo de falha em uma operação subseqüente 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. E, 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 subseqüente, assistência técnica ou cliente final)?

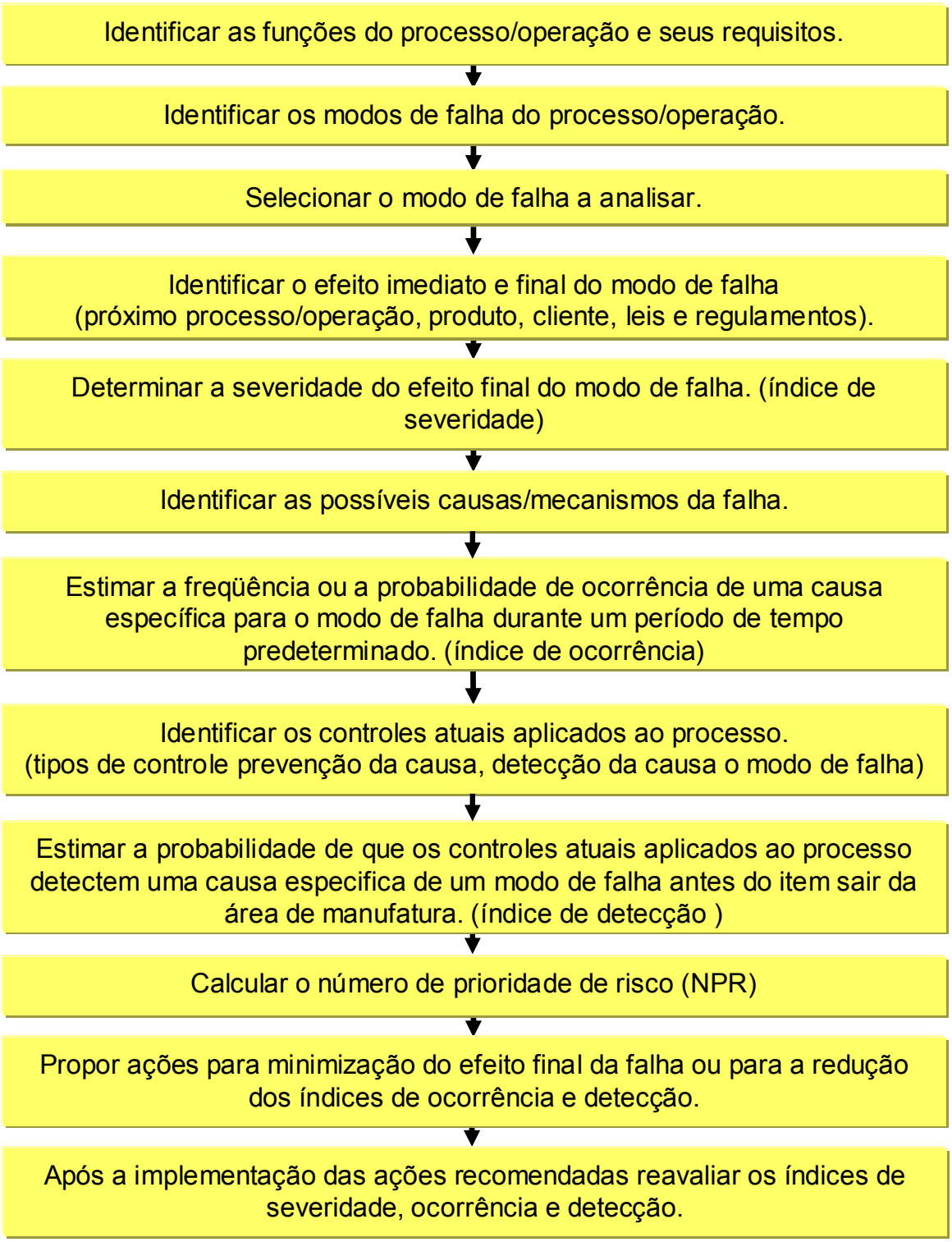


Figura 2.7 – Fluxograma das etapas do FMEA de processo (FONTE: AIAG, 2001).

A norma revela que uma comparação com processos similares e uma revisão de reclamações de clientes (operação ou processo subseqüente, assistência técnica ou cliente final) relacionadas a componentes similares é recomendada como ponto de partida. Por exemplo, no processo de moldagem citado anteriormente, um modo de falha típico para a função ejetar o componente poderia ser marcas de extração.

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

A severidade é um índice de classificação associado ao efeito mais sério da falha. O índice de severidade se aplica somente ao efeito e, em geral, só pode ser modificado por meio de alterações de projeto do componente ou redesenho do processo. Para a avaliação da severidade usualmente utiliza-se tabelas de classificação que refletem questões organizacionais combinadas com exigências de clientes e outras partes interessadas como mostra na tabela 2.2.

A causa ou mecanismo potencial de falha responsabiliza-se por definir como a falha pode ocorrer, e deve ser descrita em termos de algo que possa ser corrigido ou controlado e, em especial, deve representar a causa raiz e não os sintomas da falha.

Por sua vez, a ocorrência é a probabilidade de que uma causa específica irá ocorrer. A ocorrência é representada por um índice correspondente à estimativa da distribuição de freqüências que pode ocorrer para uma dada causa sobre uma determinada quantidade de peças produzidas com os controles atuais empregados no processo. A determinação do índice de ocorrência (ver tabela 2.3) pode ser feita baseada em dados de capacidades dos processos (Cp_k) combinados em tabelas de classificação de ocorrências (STAMATIS, 2003).

Tabela 2.2 – Índice de severidade (FONTE: SAE, 2002).

Severidade	**Critério**	**Índice de severidade**
Perigoso sem aviso prévio.	Índice de severidade muito alto quando o efeito da falha afeta a segurança na operação do veículo e/ou envolve uma nao-conformidade Em relação a legislação. A falha pode pôr em Perigo o operador da máquina ou montador	10

Tabela 2.3 – Índice de ocorrência (FONTE: STAMATIS, 2003).

Ocorrência do modo de falha	Critério	Índice de ocorrência
Moderada.	Processo sob controle estatístico. Falhas isoladas ocorrem algumas vezes o índice $Cp_k \geq 1,0$ a freqüência é de 1 em 4000 peças ou ± 3,5 σ	3

A aplicação do método PFMEA, em geral, analisa os efeitos de falha considerando isoladamente cada modo de falha como mostra a figura 2.8. Todavia, Pickard et al. (2005) apresentam um procedimento que permite a consideração de modos de falha múltiplos mantendo as características do método FMEA.

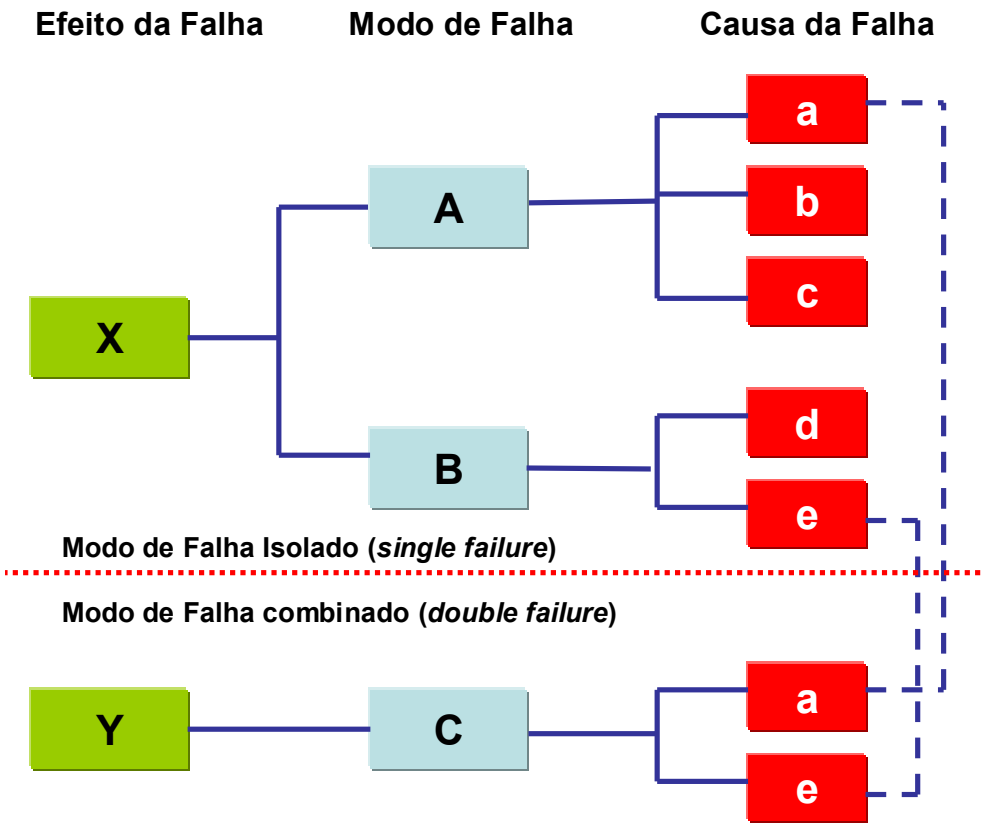


Figura 2.8 – Árvore com modos de falhas isolados ou combinados (FONTE: PICKARD et al., 2005).

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.

O número de risco de prioridade é o produto dos índices de severidade, ocorrência e detecção, e podem variar entre 1 e 1000. Deste modo, este número pode ser usado para priorizar as ações necessárias para minimizar cada um dos índices de acordo com estratégias apropriadas. Observando-se que o índice de severidade somente pode ser reduzido por alterações de projeto e redesenho do processo, a ocorrência pode ser reduzida com modificações nas especificações e requisitos do processo e o índice de detecção (ver tabela 2.4) pode ser reduzido aumentando-se a capacidade dos métodos de controle empregados (STAMATIS, 2003).

Cumpre observar que uma profunda revisão do método FMEA está além do escopo desta tese, cujo objetivo é apresentar os conceitos essenciais do método visando permitir um entendimento da abordagem proposta nesta tese. Todavia, uma revisão mais profunda do método pode ser encontrada nas referências apresentadas ao longo do texto.

Tabela 2.4 – Índice de detecção (FONTE: STAMATIS, 2003).

Detecção do modo de falha	**Critério**	**Índice de detecção**
Muito alta.	Probabilidade muito baixa de um produto não-conforme ser forncedido (1/10.000) o modo de falha e funcionalmente óbvio e facilmente detectável. A probabilidade de Detecção é no mínimo 99,99 %	1

2.1.3.4 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 a Análise de Modos de Falha e seus Efeitos (FMEA), quando conduzida 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; ATAMER, 2004;).

Tradicionalmente, os elementos de uma análise FMEA são adquiridos ou eliciados 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; que, posteriormente, são registrados em planilhas ou bancos de dados, normalmente na forma de linguagem natural.

Assim, este valioso conhecimento obtido também a 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. E, 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 LEE (2001) propõem o uso de uma ontologia como uma alternativa inovadora para modelar e tratar o conhecimento decorrente da Análise de Modos de Falha e Efeitos (FMEA) de forma a superar a barreira identificada. O conceito de ontologia usado neste contexto tem suas raízes na área de pesquisa da inteligência artificial e, neste campo, 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).

2.2 DESAFIOS À QUALIDADE EM PROCESSOS DE MOLDAGEM POR INJEÇÃO

O processo de moldagem por injeção é uma das mais versáteis tecnologias de processamento de materiais termoplásticos usados na indústria. Este processo é capaz de produzir peças técnicas complexas já em sua forma final e dentro de estreitas especificações de tolerâncias dimensionais e geométricas a custos altamente competitivos (CHEN e TURNG, 2005).

Cumpre ressaltar, que a apresentação de uma ampla revisão da literatura sobre os conceitos fundamentais do processo de moldagem por injeção está além do escopo desta tese, pois o tema é ampla e profundamente discutido na literatura da área. Portanto, esta subseção concentra-se nos conceitos e aspectos relacionados diretamente à qualidade da peça moldada visando unicamente permitir o entendimento da abordagem proposta nesta tese.

Neste contexto, a moldagem por injeção não é um processo contínuo como pode parecer à primeira vista, mas, na verdade, caracteriza-se por um processo que ocorre de acordo com um ciclo complexo e que compreende uma série de eventos pré-definidos. Neste sentido, Manrich (2005) destaca que este ciclo de injeção, em essência, é composto por dois ciclos distintos: um ciclo da perspectiva da rosca recíproca (ou parafuso) e outro ciclo da perspectiva do molde, como mostra a figura 2.9.

A fase de preenchimento do molde (entre os eventos 1-2 da figura 2.9) ocorre em três estágios (MANRINCH, 2005): preenchimento do molde propriamente dito, fase de pressurização, e fase de recalque ou fase de compensação.

Na fase de preenchimento a massa fundida (polímero em estado plástico, moldável e homogêneo tanto em temperatura quanto em distribuição dos elementos presentes) é pressionada, devido ao movimento de avanço do pistão (rosca recíproca), para dentro da cavidade do molde de modo a preenchê-la totalmente, embora ainda sem pressurização. Nesta

fase, é enviada aproximadamente 75 % da massa total, pois a massa ainda está expandida termicamente.

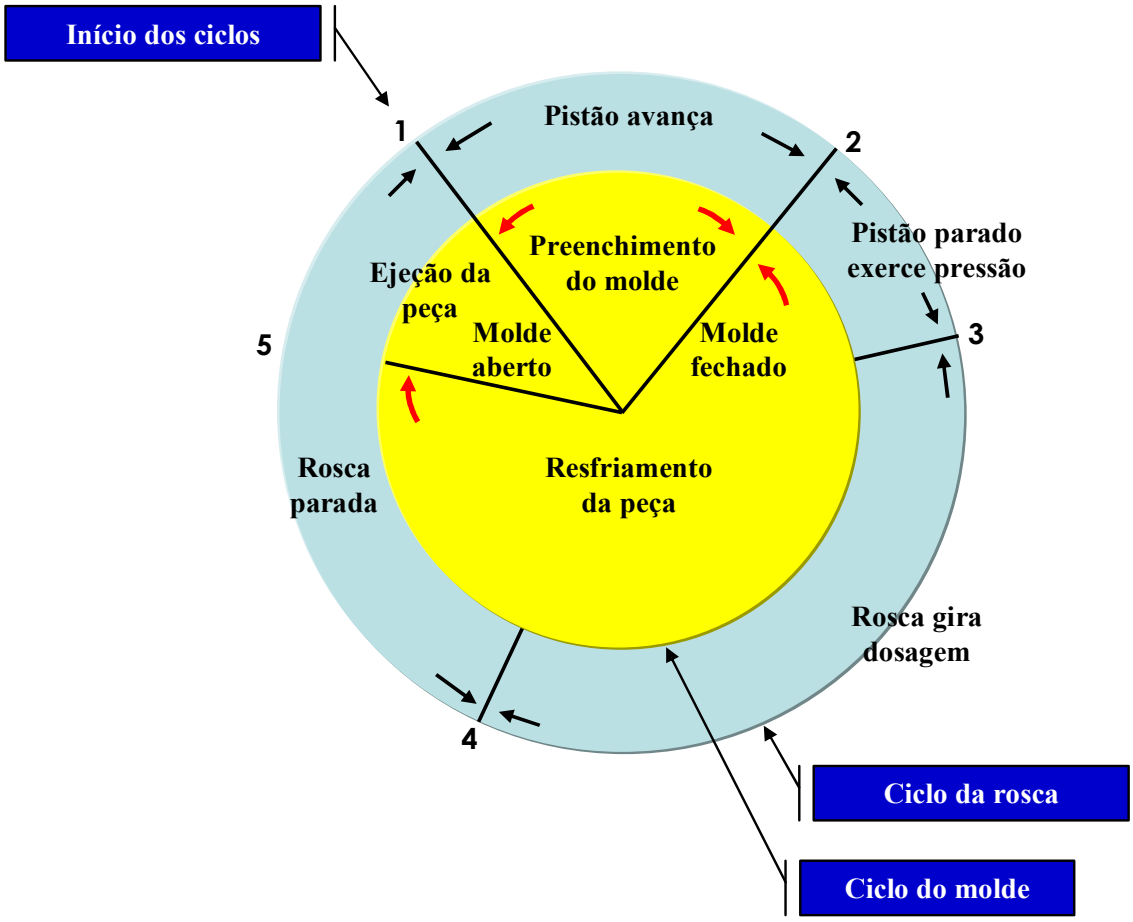


Figura 2.9 – Ciclo de injeção (FONTE: MANRICH, 2005).

É importante observar que a rosca recíproca (parafuso) exerce duas funções distintas em seu ciclo. Na primeira, ela funciona como uma rosca (parafuso) propriamente dita respondendo pela plastificação e homogeneização do polímero mediante as seguintes etapas: transporte do polímero a partir do funil até o molde, elevação da pressão da massa polimérica ao longo do comprimento da rosca, aquecimento do material tanto em função do aquecimento fornecido pelas mantas elétricas acopladas ao canhão (barril) quanto devido ao atrito e cisalhamento.

Na segunda, ela exerce a função de um pistão que pode ser acionado hidráulica ou eletricamente com velocidades e pressões ou força respectivamente controladas (MANRICH, 2005).

Por sua vez, na fase de pressurização, a massa polimérica é pressionada, mediante pressão ou força controlada, e uma quantidade entre 25 % e 10 % a mais de massa é enviada para a cavidade a fim de compensar o encolhimento volumétrico e, normalmente, a pressão atingida nesta fase é a máxima de injeção (MANRICH, 2005).

Já a fase de recalque é responsável pelo envio do restante da massa (entre 0 e 15 %), bem como manter sob pressão a massa dentro da cavidade do molde. Esta fase é realizada mediante um perfil de pressões decrescentes sempre abaixo da pressão máxima de pressurização e a transição e marcada pelo ponto de comutação. Todavia, a literatura não recomenda injetar 100 % da massa possível, pois entre 0,2 % e 2,0 % de contração se faz necessário para a correta extração da peça do molde (MANRICH, 2005).

O controle do colchão[8], isto é, o controle do volume de massa polimérica fundida, restante defronte à rosca após o preenchimento do molde, e que corresponde a 5 % da massa a ser dosada em cada ciclo, é vital para assegurar características de injeção constantes. Isto porque, o colchão é responsável por regular a transmissão da pressão sobre o material na cavidade (MANRICH, 2005).

A literatura da área revela que a otimização do ciclo de injeção e, por extensão, a qualidade da peça injetada, depende da interação de variáveis complexas envolvidas no processo (MANRICH, 2005, CHEN e TURNG, 2005, GOODSHIP, 2004, BAYER, 2007a), e entre estas variáveis tem-se: a taxa de resfriamento das paredes do molde, temperatura da massa fundida, tempo de preenchimento do molde, variações de tensão sobre o material dentro da cavidade, tempo e pressão de recalque.

Assim, da ótica de sistemas de controle de processos, Chen e Turng (2005) apresentam uma ampla análise e revisão das variáveis envolvidas no processo de moldagem por injeção com o objetivo de desenvolver uma estratégia de controle e, neste estudo, sugerem que estas podem ser classificadas em três níveis distintos: variáveis de máquina (nível 1), variáveis de processos (nível 2) e variáveis de qualidade (nível 3).

As variáveis do primeiro nível referem-se à operação da máquina e que podem ser controladas de forma independente por meio de controladores lógicos programáveis (CLP) ou controladores Proporcional-Integral-Derivativo (PID) e sensores apropriados disponíveis na própria máquina injetora, denominados pelos autores como variáveis da máquina, como mostra a tabela 2.5.

As do segundo nível são as variáveis dependentes ou de processo, as quais refletem não somente a dinâmica da máquina de moldagem relacionada às variáveis do primeiro nível, mas

[8] Do termo em inglês *Cushion.*

também dos seguintes elementos: do material de moldagem, da máquina e dos elementos construtivos do molde entre outros e denominado pelos autores como variáveis do processo, como mostra a tabela 2.6.

Tabela 2.5 – Variáveis de máquina – Nível 1 (FONTE: CHEN e TURNG, 2005).

Nível 1 – Variáveis de máquina (controladas independentemente)
Temperaturas: Temperatura do canhão (diversas zonas de aquecimento), Temperatura do bico, Temperatura do fluido refrigerante (entrada). **Pressões:** Pressão de pressurização e pressão de recalque, Contrapressão (controle da plastificação na rosca), Máxima pressão de injeção. **Seqüência e movimento:** Momento de fechamento/pressurização/recalque/ contrapressão/ejeção, Velocidade de injeção, Rotação da rosca recíproca (parafuso), Volume de dosagem e colchão.

As variáveis do terceiro nível representam o conjunto de critérios de qualidade, e seu controle deve ser o foco do sistema de controle. E, de acordo com a aplicação e requisitos funcionais da peça injetada, diferentes características podem ser selecionadas, como mostra a tabela 2.7. Todavia, todas as variáveis de qualidade são de resposta das variáveis do nível 1 e 2.

Nesta perspectiva, Chen e Turng (2005) destacam que a dinâmica da qualidade governada pelas relações entre os parâmetros e variáveis de máquina e processo não são bem compreendidas ou mesmo equacionadas analiticamente até o presente momento. Isto, em parte, é responsável pelo fato de os métodos e formas de controle da qualidade aplicados às características da peça injetada não acompanharem os avanços tecnológicos obtidos em relação aos sistemas de automação e controle dos parâmetros e variáveis da máquina e do processo.

Chen e Turng (2005) dividem os parâmetros do primeiro nível em três categorias: temperaturas, pressões e seqüência e movimentos. As temperaturas envolvem as temperaturas no canhão ou barril[9] em diferentes zonas de aquecimento por resistências elétricas, temperatura do bico e temperatura do fluido refrigerante que controla a temperatura das paredes do molde e por conseqüência a taxa de resfriamento do molde.

Tabela 2.6 – Variáveis de processo - Nível 2 (FONTE: CHEN e TURNG, 2005).

Nível 2 – Variáveis de processo (dependentes)
Temperatura da massa (no bico, canais de alimentação e cavidade), Pressão da massa (no bico e na cavidade), Avanço da frente de fluxo de massa, Máxima tensão de cisalhamento, Taxa de dissipação de calor e refrigeração.

Tabela 2.7 – Variáveis de qualidade – Nível 3 (FONTE: CHEN e TURNG, 2005).

Nível 3 – Características (resposta final)
Massa e espessura da peça injetada, Contração e empenamento, Rechupes, Aparência e resistência da linha de emenda, Outras não-conformidades estéticas: manchas de queimado, entre outros.

As pressões correspondem à pressão de pressurização e de recalque e estão relacionadas com o controle de pressão do atuador hidráulico ou elétrico responsáveis pela força e velocidade axial do pistão. Por sua vez, a contrapressão ou pressão para controle de plastificação é fundamental para a qualidade da plastificação da massa polimérica depositada à frente da rosca da injetora. O monitoramento da contrapressão envolve o controle da

[9] Do termo em inglês *Barrel*.

resistência imposta à rosca recíproca (parafuso) para ser retorno em direção ao funil devido: à variação de pressão gerada pela geometria da rosca durante a fase de plastificação do polímero e pelo material depositado pela rosca à sua frente.

As variáveis relacionadas à seqüência e movimentos compreendem o controle da velocidade de injeção ou gradiente de velocidades, a rotação da rosca recíproca, dosagem inicial e o colchão (via deslocamento da rosca) e o ponto de comutação.

A velocidade de injeção é controlada para permitir o controle do avanço da frente de fluxo de material de acordo com o caminho percorrido pela frente de fluxo (geometria dos canais de alimentação, ponto de injeção e geometria da peça injetada).

Já o controle do ponto de comutação e, por conseqüência, o controle do nível de tensões internas devido ao encolhimento volumétrico da peça injetada pode ser feito monitorando o tempo, o curso ou a pressão de pressurização.

Em síntese, a qualidade da peça moldada decorre dos fatores mostrados na figura 2.10.

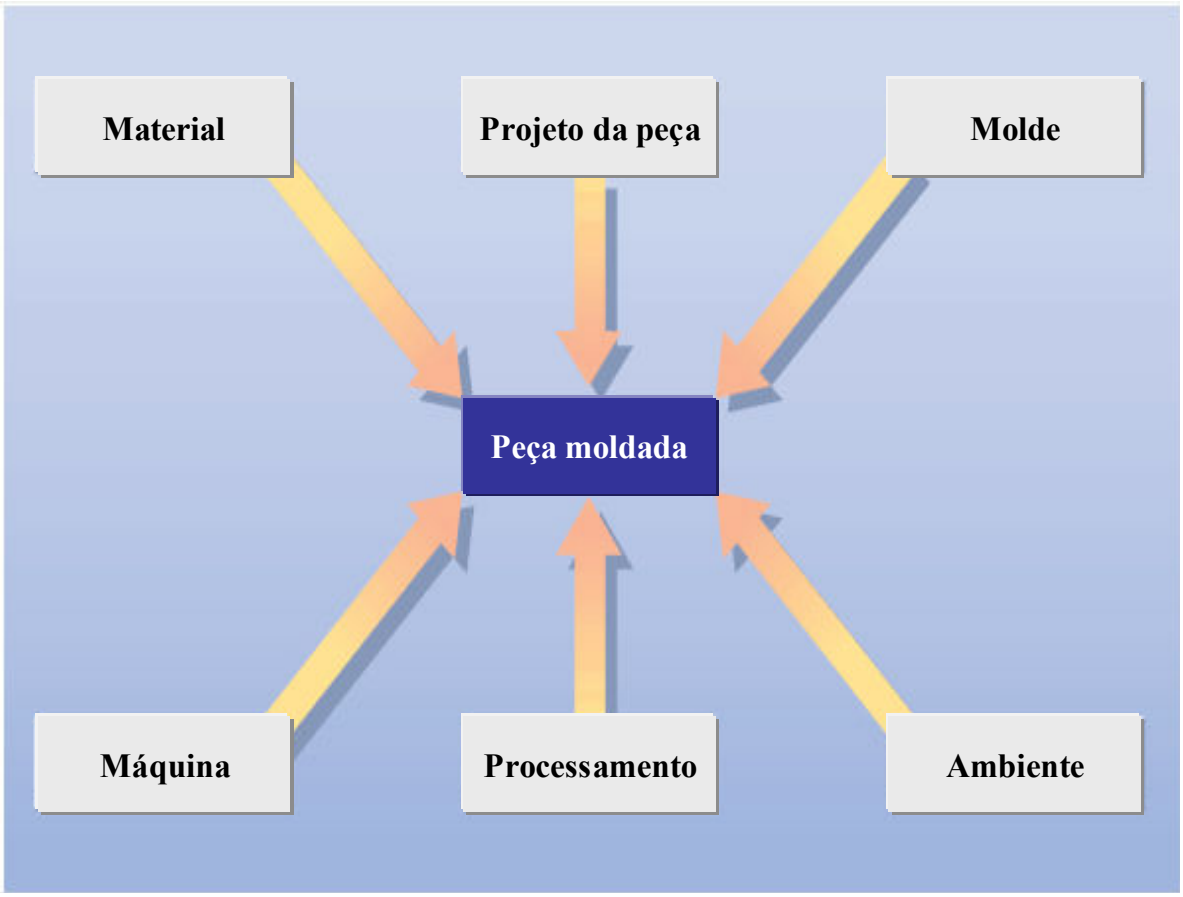


Figura 2.10 – Fatores de influência sobre a qualidade (FONTE: BAYER, 2007a).

Da perspectiva da qualidade da peça moldada, a literatura destaca ainda que, em geral, as propriedades mais importantes de um componente moldado envolvem: a aparência, as propriedades mecânicas e físicas, e as características dimensionais e geométricas (BAYER, 2007a). Estas propriedades são influenciadas diretas ou indiretamente pelos seguintes fatores: material empregado, projeto da peça moldada, projeto do molde, máquina, operação de processamento e condições ambientais, como mostra a figura 2.10. Os elementos que compõem estes fatores são os seguintes:

- O material empregado (seleção, características reológicas do polímero e características de processamento);
- O projeto da peça moldada (projeto adequado às características do material; dimensões, complexidade das features e espessuras de parede da peça);
- O molde (elementos construtivos, disposição das cavidades; configuração: pontos de injeção, canais de alimentação e sistemas de ventilação);
- A máquina (rigidez, capacidade, velocidades, ajustes);
- A operação de processamento (controles das variáveis de máquina e processo);
- O ambientes (localização, unidades periféricas como secadores e habilidades e qualificação dos operadores).

A literatura observa que, não obstante todos os recursos científicos e tecnológicos empregados atualmente, a probabilidade de ocorrência de não-conformidades no processo de moldagem não pode ser completamente eliminada. E, em geral, decorrem da complexa inter-relação dos fatores apresentados na figura 2.10, o que implica em uma difícil modelagem analítica e, frequentemente, tornam a tarefa de descobrir a fonte da não-conformidade e sua respectiva solução demorada e dispendiosa (GOODSHIP, 2004).

Neste sentido, Goodship (2004) apresenta uma classificação das não-conformidades em 17 tipos principais de acordo com a tabela 2.8. E ainda fornece um fluxograma que permite ao usuário de forma sistemática investigar as causas de um tipo particular de não-conformidade, bem como sugere alternativas para eliminar as causas mediante ajuste das variáveis da máquina, ou se alterações no projeto do molde ou da peça precisam ser considerados.

Tabela 2.8 – Classificação das não-conformidades de acordo com Goodship (2004).

- Marcas de rechupe (*Sink marks*);
- Manchas (*Streaks*):
    - Manchas de queimado (*Burnt streaks*);
    - Manchas de umidade (*Moisture streaks*);
    - Manchas na cor (*Colour streaks*);
    - Manchas de ar (*Air streaks, air hooks*);
    - Manchas de fibra de vidro (*Glass fiber streaks*).
- Diferença de brilho (*Gloss differences*);
- Linhas de emenda (*Weld line*);
- Jateamento (*Jetting*);
- Efeito Diesel (*Diesel effect, burns*);
- Casca de laranja (*Record grooves effect*);
- Peças incompletas (*Incompletely filled parts*);
- Rebarbas (*Oversprayed parts, flashes*);
- Deformação durante a desmoldagem;
- Delaminação (*Flaking of the surface layer*);
- Gota fria (*Cold Slugs*) / Linhas de material frio (*Cold flow lines*);
- Bolhas de ar arredondadas ou alongadas na superfície (*Blister formation*).
- Pontos pretos (*Dark spots*).
- Pontos de baixo brilho (fosco) próximo ao canal de injeção (*Dull spots*).

Neste cenário, por outro lado, Sancho (2005) apresenta um estudo desenvolvido no Centro de Tecnológico da Associação Catalã de Empresas de Moldes e Matrizes (ASCAMM[10]), no qual percentualmente as ocorrências de não-conformidades são relacionadas aos fatores que impactam diretamente a qualidade da peça injetada, a saber: projeto da peça, do molde, material e processamento e este estudo é sintetizado nas tabelas 2.9 e 2.10. Entretanto, cumpre observar que uma profunda revisão sobre a ocorrência, as causas e as soluções destas não-conformidades está além do escopo desta tese e pode ser encontrada nas referências apresentadas ao longo do texto, bem como em BASF (2007), BAYER (2007a, 2007b), DOW (2007), DUPONT (2007), GE (2007), IMMET (2007), NOVEON (2007), TEIJIN (2007) e UMG (2007).

[10] http://www.ascamm.es/imatge-es

Tabela 2.9 – Estudo da relação entre não-conformidades e fatores que impactam a qualidade (Parte I) (FONTE: SANCHO, 2005).

Não-confomidades (parte I)	% de não-conformidades associadas				Classificação	Importância	Pontuação
	Projeto		Material	Processo			
	Peça	Molde					
Efeito diesel (*Concentrated blackening, Entrapped air*).	**50 %**	**40 %**		**10 %**	**8**	**Alta**	**19**
Esbranquiçamento por tensão (*Stress whitening*).		**50 %**		**50 %**		**Alta**	**16**
Bolhas de ar arredondadas ou alongadas na superfície (*Blister formation*).				**100%**		**Média**	**14**
Diferenças de brilho (*Gloss differences*).	**55 %**	**20 %**		**25 %**		**Alta**	**17**
Contaminação em geral.		**20 %**		**80 %**		**Alta**	**18**
Diferença de cor.			**20 %**	**80 %**		**Alta**	**17**
Dimensões incorretas.	**25 %**	**25 %**	**25 %**	**25 %**		**Alta**	**19**
Estrutura não-homogenea.			**45 %**	**55 %**		**Alta**	**19**
Delaminação (*Flaking of the surface layer*).			**5 %**	**95 %**		**Baixa**	**3**
Baixa resistência mecânica.		**50 %**		**50 %**		**Alta**	**18**
Gota fria (*Cold Slugs*).		**75 %**		**25 %**		**Média**	**8**
Jateamento (*Jetting*).	**30 %**	**65 %**		**5 %**	**10**	**Alta**	**18**
Linhas de material frio (*Cold flow lines*).	**45 %**	**50 %**		**5 %**	**4**	**Média**	**14**
Irregularidades na linha de emenda (*Knuckle line*).	**40 %**	**40 %**	**10 %**	**10 %**		**Média**	**13**
Linhas de emenda (*Weld lines*).	**40 %**	**38 %**	**10 %**	**12 %**	**5**	**Alta**	**18**
Manchas próximas ao ponto de injeção (*Blush marks around gates*).		**75 %**		**25 %**		**Média**	**13**
Marcas de extração.		**70 %**		**30 %**		**Média**	**14**
Manchas de cor (*Colour streaks, Orientation caused by flow*).	**20 %**	**25 %**	**35 %**	**20 %**		**Alta**	**18**
Casca de laranja (*Record grooves effect*).		**18 %**	**4 %**	**78 %**		**Baixa**	**9**
Peças incompletas.	**15 %**	**20 %**	**5 %**	**60 %**	**3**	**Alta**	**20**
Pontos pretos (*Dark spots*).		**10 %**		**90 %**		**Alta**	**18**
Manchas de umidade (*Streaks*).		**5 %**		**95 %**	**7**	**Alta**	**17**

Tabela 2.10 – Estudo da relação entre não-conformidades e fatores que impactam a qualidade (Parte II) (FONTE: SANCHO, 2005).

Não-confomidades (Parte II)	% de não-conformidades associadas				Classificação	Importância	Pontuação
	Projeto		Material	Processo			
	Peça	Molde					
Rebarbas (*Flashs*).		**48 %**	**1 %**	**51 %**	**1**	**Alta**	**16**
Marcas de rechupe localizadas (*Sink marks*).	**50 %**	**30 %**		**20 %**	**2**	**Alta**	**17**
Trinca por tensão (*Stress cracking*).	**25 %**	**25 %**	**10 %**	**40 %**		**Alta**	**19**
Bolhas na forma de gota (Tear drops).	**40 %**	**30 %**	**10 %**	**20 %**		**Baixa**	**8**
Tensões e deformações.	**40 %**	**30 %**	**10 %**	**20 %**	**6**	**Alta**	**19**
Vazios (*Voids*).	**35 %**	**20 %**	**10 %**	**35 %**	**9**	**Alta**	**18**

Adicionalmente, Sancho (2005) apresenta ainda um estudo que explora a ocorrência de não-conformidades tanto na fase de pré-industrialização (*tryout*) quanto durante a produção, bem como apresenta uma classificação dos 20 mais freqüentes não-conformidade, como mostram as tabelas 2.11 e 2.12.

Tabela 2.11 – Ocorrência de não-conformidades (Parte I) (FONTE: SANCHO, 2005).

Tipos de não-conformidade (parte I)	% de ocorrência de não-conformidade: fases		classificação
	Pré-industrialização	Produção	
Rebarbas (*Flashs*)	65 %	35 %	1
Marcas de rechupe localizadas (*Sink marks*)	65 %	35 %	2
Peças incompletas	75 %	25 %	3
Linhas de material frio (*Cold flow lines*).	65 %	35 %	4
Linhas de emenda (*Weld lines*).	80 %	20 %	5
Tensões e deformações.	65 %	35 %	6
Manchas de umidade (*Streaks*)	70 %	30 %	7
Efeito diesel (*Concentrated blackening, Entrapped air*).	70 %	30 %	8

Tabela 2.12 – Ocorrência de não-conformidades (Parte II) (FONTE: SANCHO, 2005).

Tipos de não-conformidade (parte II)	% de ocorrência de não-conformidade: fases		classificação
	Pré-industrialização	Produção	
Vazios (*Voids*).	80 %	20 %	9
Jateamento (*Jetting*).	80 %	20 %	10
Diferenças de brilho (*Gloss differences*).	60 %	40 %	11
Manchas de cor (*Colour streaks, Orientation caused by flow*).	75 %	25 %	12
Casca de laranja (*Record grooves effect*).	80 %	20 %	13
Gota fria (*Cold Slugs*).	60 %	40 %	14
Manchas próximas ao ponto de injeção (*Blush marks around gates*).	60 %	40 %	15
Contaminação em geral.	20 %	80 %	16
Diferença de cor.	20 %	80 %	17
Marcas de extração.	75 %	25 %	18
Dimensões incorretas.	60 %	40 %	19
Estrutura não-homogenea.	20 %	80 %	20

Por sua vez, Chen e Turng (2005) revelam que o controle de qualidade do processo de moldagem por injeção é frequentemente conduzido de forma indireta, isto é, por meio de controles de processos dinâmicos *online* ou controle estatístico de processo *off-line*. E ressaltam que os dois principais desafios à implementação de uma forma de controle de qualidade em tempo real são a complexa dinâmica da qualidade em processos de moldagem e a falta informações de *feedback* de qualidade *online*.

Em relação ao primeiro desafio, eles argumentam que as informações quantitativas sobre as relações entre as variáveis de máquina, processo e qualidade são insuficientes. Mesmo para o controle no nível do processo, onde a dinâmica do processo pode ser descrita por um conjunto de equações governantes, mas os graus de liberdade são insuficientes para manipular o estado do polímero fundido através do tempo e do espaço.

E em relação ao segundo, Chen e Turng (2005) revelam que com a tecnologia existente, medidas diretas e *online* da qualidade da peça dentro do ciclo de tempo permanecem como tarefas desafiadoras para as aplicações comerciais. Em especial, devido ao fato de que alguns dos critérios de qualidade de natureza qualitativa (por exemplo: aspectos estéticos) são difíceis de medir quantitativamente por métodos e instrumentações convencionais. Além da

grande diversidade de requisitos de qualidade entre as múltiplas aplicações do processo de moldagem.

A literatura reporta ainda diversas outras abordagens tecnológicas para os processos de injeção que se enquadram no mesmo contexto dos desafios relatados pela literatura. Entre as quais se destacam, de acordo com Manrich (2005), além da moldagem por injeção convencional, os seguintes processos de injeção: a gás, com água, por compressão, por transferência, de espumas estruturais, com superfícies micro estruturadas, de peças com paredes finas, com decoração direta no molde, com núcleos fundidos; co-injeção ou múltipla injeção, microinjeção e nanoinjeção.

2.3 TECNOLOGIA DE AGENTES EM APLICAÇÕES NA ÁREA DE MANUFATURA

Na medida em que o foco das pesquisas nas áreas da ciência da computação e da engenharia de software deslocou-se, naturalmente, dos sistemas computacionais caracterizados pela concepção centralizada, com códigos não-reutilizáveis e fechados para o mundo exterior (*standalone systems*), para a investigação de aplicações de softwares que buscam atender demandas comerciais e industriais contemporâneas, em termos de sistemas robustos e com capacidade de operar em ambientes abertos, dinâmicos e distribuídos, foi inevitável o confronto com enormes desafios tanto de natureza conceitual e metodológica quanto tecnológicas (LUCK et al., 2004, ZAMBONELLI e PARUNAK, 2003).

Neste sentido, as próprias características destes novos ambientes, tais como: a necessidade de interação entre sistemas heterogêneos; expansão das fronteiras organizacionais e geográficas; operação eficiente e capacidade de adaptação às rápidas mudanças de requisitos e aumento da quantidade de informações disponíveis e, ainda, operar com segurança suficiente para proteger dados pessoais e outros ativos de conhecimento dos inúmeros evolvidos sugerem, claramente, que revisões e melhorias nas abordagens metodológicas para desenvolvimento de sistemas seriam necessárias (HENDERSON-SELLERS e GIORGINI, 2005).

Neste novo quadro, 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 visto por muitos pesquisadores como um aspecto fundamental a ser considerado nos novos modelos. E, dentro desta perspectiva, a abordagem orientada a agentes vem se tornando, ao longo da última década, uma importante abordagem para responder de forma concreta a estes desafios, bem como contornar a complexidade dos sistemas (LUCK et al., 2004).

E, ainda, devido à natureza horizontal do modelo de agentes, é altamente provável que o sucesso na adoção deste novo modelo terá um impacto profundo e de longo prazo sobre a competitividade e viabilidade da indústria da tecnologia da informação (TI). E também sobre a forma pela qual futuros sistemas computacionais serão concebidos e implementados (LUCK et al., 2004).

Na literatura atual, o conceito de agente de software no campo da inteligência artificial distribuída, em geral, é enfocado por três perspectivas diferentes. Da primeira perspectiva, os agentes são apresentados e discutidos como uma metáfora social para a análise e o projeto de sistemas, considerando aspectos da ciência cognitiva, e envolve aspectos como: metodologia de desenvolvimento orientada a agentes, arquiteturas conceituais de agentes, e a infra-estrutura dos sistemas multiagentes e sua conexão com outras disciplinas (LUCK et al., 2004).

A segunda perspectiva discute os agentes do ponto de vista tecnológico ou como solução tecnológica dentro de um ramo da informática e engenharia, concentrado-se em desenvolver técnicas para resolver problemas de forma cooperativa e distribuída, e por sua vez envolve: o planejamento de sistemas multiagentes, linguagens de comunicação de agentes, mecanismos de coordenação e estratégias de negociação entre agentes, arquiteturas e algoritmos de controle, mecanismos de aprendizagem e principais desafios tecnológicos. E a terceira engloba a apresentação dos casos de aplicações e desenvolvimentos nas áreas industriais e comerciais (LUCK et al., 2004).

2.3.1 Agentes como metáforas para projeto

O uso de agentes como uma abstração ou, ainda, uma metáfora que sugere uma analogia entre humanos e agentes que operam em um micro mundo, com a finalidade de projetar e construir sistemas serviu como ímpeto inicial para desenvolvimentos neste campo (HOPKINS e FISHWICK, 2001).

Desta ótica, os agentes sugerem também um modo apropriado para considerar sistemas complexos com múltiplos componentes, sendo estes distintos e independentes. Além disto, eles permitem agregar diferentes funcionalidades, antes dissociadas, tais como: planejamento, aprendizagem e coordenação em um todo conceitualmente coerente (LUCK et al., 2004).

Não obstante estes esforços, o termo agente, ainda, não encontra na literatura uma definição formal amplamente aceita pela comunidade científica. Porém, é possível encontrar uma definição funcional pela revisão das várias características e habilidades usadas na literatura para descrever agentes de software, como as compiladas por Hopkins e Fishwick (2001) entre as quais:

- Representação autônoma de outras entidades;
- Habilidade para comunicar-se, interagir e colaborar com outras entidades;
- Habilidade para perceber, mover através de, reagir, alterar um complexo e dinâmico ambiente;
- Habilidade para raciocinar, aprender e adaptar;
- Habilidade de ter iniciativa e perseguir objetivos.

Adicionalmente, apesar de não representar um consenso geral, a definição proposta por Wooldridge e Jennings (1995) é citada por vários autores relacionados à aplicação da tecnologia de agentes em sistemas de manufatura, entre eles Bussmann et al. (2004), Paolucci e Sacile (2005), Shen et al. (2005, 2001), na qual um agente pode ser considerado como um processo de software com as seguintes propriedades:

- **Autonomia**: agentes operam sem a intervenção direta de humanos ou outros e tem controle sobre suas ações e estados internos [...]. Neste sentido, esta propriedade compreende a autonomia de decisão, isto é, a capacidade de analisar uma dada situação, gerar as alternativas de solução e selecionar o curso de ação que melhor atende os seus objetivos de projeto com base em um processo de decisão racional e planejador de ações visando atingir seus objetivos e metas; bem como a autonomia de execução.
- **Habilidade social:** agentes interagem com outros agentes (inclusive humanos) via algum tipo de linguagem de comunicação de agentes (*agent comunication language* - ACL) [...]. Esta habilidade, associada aos mecanismos de coordenação e negociação, configuram um ambiente de atuação multiagentes;
- **Reatividade:** agentes percebem seu ambiente [...], e respondem de uma forma temporal às mudanças que ocorrem no seu ambiente;
- **Pró-atividade:** agentes não agem somente em resposta ao seu ambiente, mas são capazes de exibir um comportamento dirigido a um objetivo por iniciativa própria.
- **Adaptabilidade:** significa que os agentes podem ser capazes de modificar o seu comportamento ao longo do tempo em resposta a mudanças nas condições do ambiente [...];
- **Veracidade:** isto é, a premissa que um agente não irá, propositalmente, comunicar uma falsa informação [...];

- **Racionalidade:** é a premissa que um agente irá agir de forma a atingir os seus objetivos de atingir se estes forem viáveis.
- **Mobilidade:** considerada uma propriedade opcional e significa que os agentes podem mudar a sua localização física para melhorar sua capacidade de resolver problemas [...];

Estas propriedades e descrições funcionais de alto nível sugerem que os agentes participam de um ciclo de tomada de decisão (do termo em inglês, *perceive-decide-act*), o que implica em considerar, por conseqüência, um conjunto de características de nível mais baixo, tais como: agentes desempenhando papéis em um sistema, a metáfora de um agente como possuindo um estado mental, incluindo habilidades, responsabilidades, atitudes e capacidades (HENDERSON-SELLERS e GIORGINI, 2005).

E, ao considerar-se suas interações via percepções e ações com outros agentes e o ambiente, introduz-se as noções de percepções, ações e linguagem de comunicação entre agentes, onde a habilidade de negociação envolve considerar redes de restrições, estratégias de ação e questões de competição versus negociação (HENDERSON-SELLERS e GIORGINI, 2005).

Neste cenário, é importante ressaltar a diferença conceitual entre agentes e objetos ativos das linguagens computacionais orientadas a objetos, tendo em vista que muitas das infra-estruturas computacionais para implementação de agentes são escritas em linguagem orientada a objetos, em particular a linguagem Java.

No contexto das linguagens orientadas a objetos, um modelo padrão de objeto é definido como uma entidade computacional que encapsula algum estado e é capaz de realizar ações (ou métodos) sobre estes estados e comunicar-se mediante troca de mensagens (WOOLDRIDGE, 2002). Embora esta definição apresente uma grande similaridade com as propriedades funcionais dos agentes, Wooldridge (2002) e Guessoum e Briot (1999) revelam que os agentes apresentam as seguintes características adicionais:

- Um agente pode apresentar vários comportamentos (*Behaviors*), como por exemplo: interação assíncrona e concorrente com o seu ambiente; manipular as requisições de outros agentes e gerar respostas adequadas a estas requisições; raciocinar para determinar a ação mais apropriada para um dado contexto e, ainda, estes comportamentos podem ser decompostos em vários outros comportamentos

elementares. Deste modo, o modelo de agente pode apresentar grande flexibilidade em termos de comportamentos (reativos, pró-ativos e sociais).

- O modelo de agente pode incorporar estruturas de inteligência artificial em seus comportamentos para integrar formalismos de representação e mecanismos de raciocínio sobre ações, crenças, intenções e conhecimentos sobre o ambiente e sobre outros agentes para atuação coordenada.
- Um agente é uma entidade autônoma, que opera sem a direta supervisão de humanos ou outros agentes. Neste sentido, o modelo de agente inclui algum tipo de controle sobre suas ações e estados internos, assim um sistema de agente é, essencialmente, *multi-threaded* onde cada agente possui pelo menos uma *thread* ("linha de execução") de controle.

Do ponto de vista computacional estas características são modeladas pela arquitetura interna do agente, que fundamentalmente consiste na máquina subjacente aos componentes autônomos que suportam o efetivo comportamento no mundo real (GUESSOUM e BRIOT, 1999).

Por extensão, descrever sistemas multiagentes (SMA) representa também uma tarefa complexa; contudo, na maioria das definições dadas na literatura um sistema multiagentes é descrito como um sistema composto por agentes cooperativos ou competitivos que interagem de forma a atingir objetivos individuais ou comuns (HENDERSON-SELLERS e GIORGINI, 2005; LUCK et al., 2004; WOOLDRIDGE, 2002).

2.3.1.1 Arquiteturas de agentes

A arquitetura interna de um agente representa o seu *modus operandi* e compreende os mecanismos de decisão que lhe permitam selecionar um curso de ação, em função: de uma agenda de objetivos, de um perfil de atuação determinado e para uma dada condição de ambiente (GARCIA e SICHMAN, 2005).

Assim, do ponto de vista da arquitetura interna, um agente pode ser representado em um *continuum* de complexidade, tendo em uma extremidade um comportamento puramente reativo (comportamental) operando em um modo simples de estímulo – resposta, no qual uma gama de entradas é recebida mediante componentes de percepção e processadas de forma lógica, produzindo, assim, uma saída, em geral, na forma de início de uma ação (BRENNER et al, 1998).

É importante perceber que este tipo de agente decide o que fazer sem referência à sua história, isto é, sua decisão é tomada com base inteiramente no presente e, sem nenhuma referência ao passado, e tais agentes são denominados agentes puramente reativos, pois estes tão somente respondem diretamente ao ambiente (WOOLDRIDGE, 2002).

E, no outro extremo deste continuum, a representação interna de um agente possui alguma estrutura de dados interna, a qual é, tipicamente, usada para registrar informações sobre o estado do ambiente e sua história ao longo do tempo. Além de um processo de tomada de decisões baseado, em parte, nestas informações, tais agentes são classificados na literatura como agentes cognitivos ou também deliberativos (WOOLDRIDGE, 2002).

Portanto, a arquitetura de um agente é, em essência, um mapa da natureza interna do agente, isto é: sua estrutura de dados, operações que podem ser realizadas sobre estas estruturas de dados, e o controle de fluxo entre estas estruturas de dados (WOOLDRIDGE, 2002).

Por sua vez, a ação "inteligente" apresentada pelos agentes cognitivos ou deliberativos tem como base um modelo simbólico explícito do ambiente e uma estrutura de processo de tomada de decisão lógica, onde o agente deliberativo, como parte de seu processo de tomada de decisão, usa os conhecimentos contidos em seu modelo do ambiente para modificar o seu estado interno (BRENNER et al., 1998).

Na literatura, um dos modelos fundamentais que descrevem o estado interno de um agente cognitivo (racional) foi formulado por Rao e Georgeff (1995), expandindo o trabalho de Bratmann (1988), denominado de princípios BDI, como definidos a seguir:

- **Crenças** (*Beliefs*), contém a visão fundamental de um agente no que diz respeito ao seu ambiente. E um agente irá usá-lo, em particular, para expressar a sua expectativa de possíveis estados futuros.
- **Desejos** (*Desires*) são decorrentes das crenças e representam o julgamento do agente para futuras situações.
- **Metas** (*Goals*) representam um subconjunto dos desejos dos agentes, realistas e não conflitantes, e estas metas configuram o escopo de ações potenciais representando as alternativas de processamento disponíveis em um momento específico.
- **Intenções** (*Intentions*) representam um subconjunto das metas, e se um agente decide perseguir uma determinada meta esta passa a ser a sua intenção.
- **Planos** (*Plans*) combinam as intenções do agente em uma unidade consistente.

Assim, a estrutura do processo de tomada de decisões pode ser entendida de modo amplo como um *loop*, no qual o agente continuamente (WOOLDRIDGE, 2002):

- Observa o mundo e atualiza suas crenças;
- Decide, deliberadamente, qual intenção atingir (deliberar significa: primeiro, determinar as opções disponíveis e posteriormente filtrá-las).
- Adota um processo de decisão para determinar qual plano será usado para atingir a sua intenção.
- Executa este plano.

O modelo BDI exerceu um efeito significativo na implementação de arquitetura de agentes cognitivos atuais (BORDINI e MOREIRA, 2004), além de servir como base para com outros modelos como *Distributed Multi-Agent Reasoning System* (dMARS) (D'INVERNO, 2004).

2.3.1.2 Infra-estrutura para agentes

Na literatura, as questões relacionadas à infra-estrutura para sistemas multiagentes dizem respeito, por um lado, ao suporte operacional e à infra-estrutura física necessárias à operação do sistema, tais como: suporte à comunicação visando a troca de informação a partir de mecanismos de transporte de mensagens e protocolos robustos, bem como suporte à segurança objetivando garantir que somente agentes propriamente autenticados possam executar as ações requeridas no sistema (LUCK et al., 2004).

Além disto, a literatura revela contribuições significativas em relação à produção de especificações padrão de software para agentes heterogêneos, interativos e sistemas baseados em agentes. Em particular, as especificações FIPA (*Foundation for Intelligent Physical Agents*), organização normativa da Sociedade de Computação da IEEE (*IEEE Computer Society*), responsável por promover a tecnologia de agentes e a interoperabilidade de seus padrões com outras tecnologias.

As especificações da FIPA consistem, basicamente, da definição de um conjunto de elementos chave de uma plataforma de agentes (infra-estrutura onde os agentes são construídos). Estas especificações devem ser seguidas rigorosamente por todos os desenvolvedores de plataformas de agentes comerciais ou de pesquisa, assegurando-se assim a plena interoperabilidade dos sistemas multiagentes implementados nestas plataformas (FIPA, 2002a).

Adicionalmente à ciência da computação, a literatura revela também a interface da tecnologia de agentes com outras disciplinas tanto do ponto de vista teórico quanto aplicado, as mais importantes conexões são com as seguintes áreas: filosofia, lógica, economia, ciências sociais e biologia (LUCK et al., 2004).

2.3.2 Agentes como fonte de tecnologia

O domínio de pesquisa da inteligência artificial distribuída (IAD) remonta à década de 1970, e compreende as pesquisas que visam resolver uma classe de problemas cuja distribuição física ou funcional é inerente (GARCIA e SICHMAN, 2005).

E, do ponto de vista histórico, também o domínio da inteligência artificial distribuída pode ser divido em duas linhas de pesquisa distintas: a área de resolução distribuída de problemas (RDP) e os sistemas multiagentes (SMA) (GARCIA e SICHMAN, 2005).

Em síntese, na abordagem de resolução distribuída de problemas, não existem agentes previamente desenvolvidos. Deste modo, sua concepção, organização e interações são *ad hoc* e dependem do problema a ser solucionado, e por conseqüência, a reutilização de sua estrutura organizacional e interações não são viáveis em outras resoluções (GARCIA e SICHMAN, 2005).

Por sua vez, os problemas abordados no contexto dos sistemas multiagentes são atividades de um conjunto de agentes autônomos, onde a estrutura organizacional e interações são genéricas e, assim, podem ser instanciadas dinamicamente e, de forma apropriada, para um caso particular, quando um determinado problema no domínio do sistema é apresentado a esta sociedade de agentes para ser resolvido (GARCIA e SICHMAN, 2005).

Assim, a concepção do sistema, os agentes, protocolos de interação e modelos organizacionais são independentes de um problema e, portanto, reutilizáveis (GARCIA e SICHMAN, 2005).

2.3.2.1 Comunicação entre agentes

Devido à natureza dos sistemas multiagentes, um dos fatores fundamentais para o sucesso do sistema reside no poder da comunicação entre os agentes. Neste sentido, os agentes devem ser capazes de comunicar-se com o usuário, com outros agentes, com recursos do sistema e, ainda, comunicar-se mutuamente, se o projeto do sistema requer a cooperação, colaboração ou negociação entre agentes (LUCK et al., 2004).

Na atualidade, existem duas linguagens de comunicação usadas, amplamente, nos sistemas multiagentes: a linguagem KQML (*Knowledge Query and Manipulation Language*) e a linguagem FIPA ACL (*Agent Comunication Language* seguindo o padrão FIPA).

A linguagem KQML foi desenvolvida no início da década de 1990, como parte integrante de um grande projeto financiado pela ARPA (*Advanced Reasearch Projects Agency*), atualmente, DARPA (*Defense Advanced Reasearch Projects Agency*), denominado KSE (*Knowledge Sharing Effort*) (FININ et al., 1997). Este projeto tinha por objetivo definir, desenvolver e testar infra-estruturas e tecnologias de apoio que permitissem a construção de bases de conhecimentos e sistemas baseados em conhecimentos em larga escala, compartilhados e reutilizáveis (FININ et al., 1997).

De uma perspectiva histórica, é importante ressaltar que este projeto foi organizado em torno de quatro grupos de trabalho denominados: *InterLíngua*, KRSS (*Knowledge Representation System Specification*), SRKR (*Shared, Reusable Knowledge Bases*) e *External Interfaces*, cujos resultados são amplamente utilizados na atualidade (FININ et al., 1997).

O grupo de trabalho *InterLíngua* desenvolveu uma linguagem comum para expressar o conteúdo de bases de conhecimento conhecida, atualmente, como KIF (*Knowledge Interchage Formalism*). O grupo KRSS concentrou-se na definição de especificações terminológicas comuns dentro de famílias de linguagens de representação, culminando com família KL-ONE.

Por sua vez, o grupo SRKB desenvolveu repositórios para ontologias compartilhadas, outras ferramentas e metodologias para compartilhar bases conhecimentos (FININ et al., 1997). E o grupo *External Interfaces* desenvolveu a linguagem KQML, que é uma linguagem e um protocolo de comunicação para troca de informações e conhecimentos entre agentes de software inteligentes (FININ et al., 1997).

Neste contexto, o padrão FIPA para a comunicação entre agentes é denominado FIPA – ACL (*Agent Communications Language*) e consiste em uma definição de sintaxe e de semântica baseada na teoria de ações de fala do campo de pesquisa filosofia da linguagem (FIPA, 2002b).

As mensagens FIPA – ACL contêm um conjunto de um ou mais parâmetros de mensagens, os quais são essenciais para a efetiva comunicação entre agentes e irão variar de acordo com as circunstâncias envolvidas (FIPA, 2002b).

Estes parâmetros incluem no mínimo uma performativa (um tipo de ação comunicativa, tais como: *inform, request, agree*, entre outras), e, adicionalmente: um remetente (*sender*, a identificação do agente que desempenha o papel de remetente); um destinatário (*receiver*, a identificação do agente que recebe a mensagem) e o conteúdo (*content*, referem-se a

elementos de uma ontologia do agente remetente e podem incluir um conceito, um predicado ou uma ação, expressa na linguagem de conteúdo) entre outros. A lista completa de parâmetros encontra-se nas especificações FIPA (FIPA, 2002b).

Do ponto de vista semântico, é importante observar que a comunicação envolve uma ontologia, que neste caso consiste em uma representação das categorias que existem em um determinado domínio (conceitos, predicados e ações). Isto é, um vocabulário que descreve os termos e relações entre estes em um dado tema por meio de linguagem de representação formal, que de acordo com o padrão podem ser: a *Semantic Language* (SL), *Constraint Choice Language* (CCI), *Knowledge Interchange Formalism* (KIF) e *Resource Description Framework* (RDF) (PAOLUCCI e SACILE, 2005).

Deste modo, a ontologia permite especificar todos os tipos de termos que um agente pode tratar e que tipo de manipulação e raciocínio ele é capaz de realizar sobre eles assegurando uma comunicação precisa entre os agentes.

Adicionalmente, a especificação FIPA estabelece um serviço de transporte de mensagens, responsável por encapsular a codificação ACL e transmiti-la por meio de protocolos de comunicação padronizados (PAOLUCCI e SACILE, 2005).

2.3.2.2 Interações entre agentes

No contexto de sistemas multiagentes, a interação entre os agentes representa também uma propriedade fundamental do sistema e, neste sentido, a interação, em geral, significa qualquer tipo troca de informações ou conhecimentos (troca de mensagens) que de algum modo influência as ações dos outros agentes (WOOLDRIDGE, 2002).

Desta perspectiva, estas interações podem de diversas formas, mas na literatura relativa a sistemas de manufatura, em geral, são agrupadas nas atividades de coordenação e negociação (BUSSMANN et al., 2004).

Embora estas questões tenham sido amplamente estudadas e discutidas ao longo da última década, existe um grande debate sobre estes temas sendo, ainda, considerado um tema de difícil equacionamento (LUCK et al., 2004).

A coordenação de um conjunto de agentes envolve o problema do gerenciamento das interdependências entre as atividades destes agentes, e torna-se particularmente difícil em sistemas multiagentes sem um controle central (BUSSMANN et al., 2004).

Da mesma forma, a negociação entre agentes envolve a tomada de decisão em conjunto por dois ou mais agentes e, na atualidade, é um dos objetivos da pesquisa sobre sistemas

multiagentes, estabelecendo mecanismos mais sofisticados de resolução de conflitos (WOOLDRIDGE, 2002).

2.3.3 Aplicações da tecnologia de agentes na área industrial

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

- Os agentes assistentes pessoais que são devotados à captura de informações ou a execução de ações em nome de usuário humano principal na internet.
- Sistemas multiagentes em apoio à decisão.
- Sistemas de simulação multiagentes, onde o sistema multiagentes é usado como um modelo para simular algum domínio do mundo real. Tipicamente, os modelos multiagentes são usados para muitos domínios com diferentes componentes, interagindo de diversas e complexas formas e onde as propriedades não são, prontamente, inferidas a partir das propriedades dos componentes, tais como: sistemas de controle de tráfego.

Em relação a aplicações comerciais e industriais, a literatura revela que as principais áreas, nas quais aplicações baseadas em agentes têm sido evidenciadas são: manufatura, controle de processos, sistemas de telecomunicações, controle de tráfego aéreo, gerenciamento de transporte e tráfego, captura e filtragem de informações, comércio eletrônico, gerenciamento de processos de negócios, de capital humano, de habilidades, de forças-tarefa móveis e sistemas de defesa, entretenimento e tratamentos médicos(LUCK et al., 2004).

Na área de manufatura, em especial, um conjunto de novos requisitos a serem considerados durante o projeto de sistemas de manufatura da próxima geração têm motivado a pesquisa e o desenvolvimento de aplicação com a tecnologia de agentes, tais como:

- **Integração empresarial**, com a finalidade de apoiar a competitividade 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 (tais como: compras, projeto, produção, planejamento e seqüenciamento da produção, controle, transportes, recursos, pessoal, materiais, qualidade entre outros) extensível aos seus parceiros comerciais via redes de comunicação;

- **Organizações distribuídas**, para uma efetiva integração empresarial através de organizações distribuídas, serão necessários sistemas baseados em conhecimento capazes de conectar a gestão da demanda, diretamente, recursos, planejamento da capacidade e seqüenciamento da produção;
- **Ambientes heterogêneos**, os novos sistemas de manufatura deverão acomodar *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;
- **Arquitetura dinâmica e aberta** permitirá 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 e entre outros. Esta cooperação deveria ocorrer de forma eficiente e rápida;
- **Integrar profissionais humanos com *software* e *hardware***, pessoas e computadores necessitam ser integrados para trabalhar coletivamente em vários estágios do desenvolvimento de produto e mesmo ao longo de todo o ciclo de vida do produto, como rápido acesso aos conhecimentos e informações necessárias. Fontes heterogêneas de informações devem ser integradas para apoiar as necessidades e para melhorar a capacidade de tomada de decisão do sistema. Ambientes de comunicação bidirecional são necessários para permitir uma efetiva e rápida comunicação entre humanos e computadores para facilitar sua comunicação;
- **Agilidade**, considerável atenção deve ser dispensada para a redução de tempo de ciclo de um produto de forma a permitir uma resposta rápida ao consumidor. A manufatura ágil é caracterizada pela habilidade em adaptar-se a ambientes com contínuas e imprevisíveis mudanças, por meio da rápida re-configuração das plantas industriais e,ou pela interação com outros sistemas heterogêneos de outros parceiros comerciais.

- **Escalabilidade** significa que recursos podem ser incorporados na 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.

Neste contexto, a literatura tem reportado uma ampla gama de aplicações da tecnologia de agentes ao longo dos últimos anos. Shen e Norie (1999) apresentam um ampla revisão referente à década de 1990, agrupando as principais aplicações da tecnologia de agentes em sistemas de manufatura inteligente distribuída em quatro grupos:

- **Aplicações com encapsulamento de sistemas de software existentes**, cuja finalidade é resolver os problemas relativos à integração de softwares legados, bem como a integração das atividades de manufatura tais como: projeto, planejamento, seqüenciamento, simulação, execução e a distribuição de produtos com os software e atividades de seus fornecedores, clientes e parceiros comerciais em um ambiente inteligente e distribuído conectado via redes de comunicação.
- **Aplicações como representação de recursos de manufatura,** onde os agentes de software representam os recursos de manufatura, tais como: células, máquinas, ferramentas, fixações entre outros, bem como: produtos, componentes e operações com o objetivo de facilitar os aspectos de planejamento de recursos, seqüenciamento e controle de execução.
- **Aplicações como modelo de serviços especiais em sistemas de manufatura,** visando facilitar a comunicação e coordenação entre agentes.
- **Aplicações que incorporam todo o planejamento ou seqüenciamento,** cujo propósito é construir um sistema completo de planejamento e seqüenciamento de manufatura.

Na atualidade, os sistemas de manufatura representam, indubitavelmente, um dos mais promissores campos para a aplicação da tecnologia de agentes, em grande parte devido à capacidade inerente desta tecnologia em dar suporte à integração de sistemas de informação de manufatura (PAOLUCCI e SACILE, 2005).

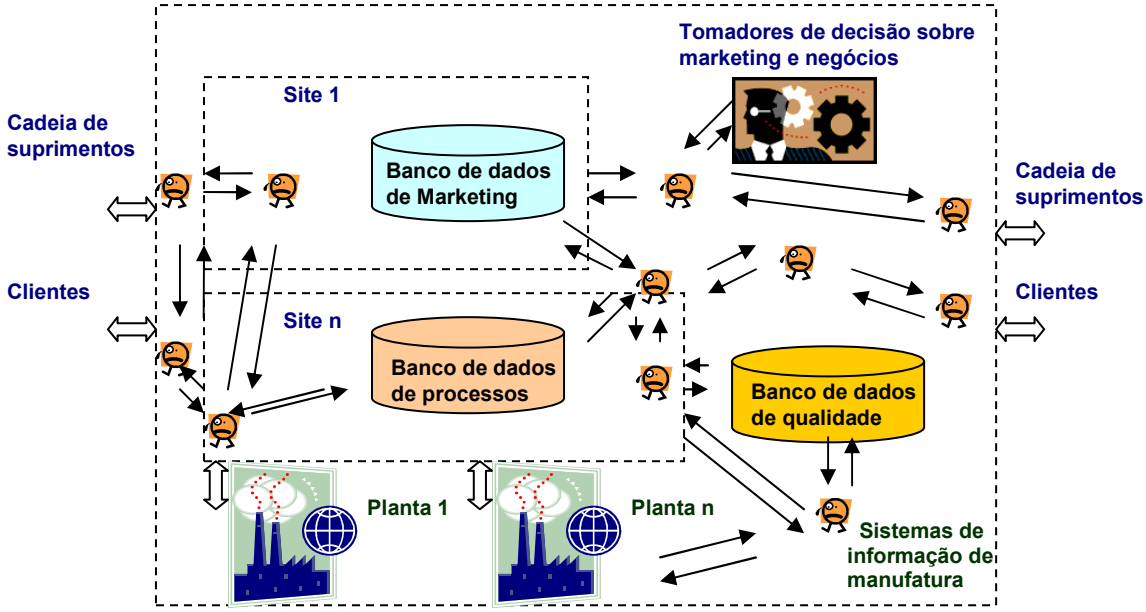


Figura 2.11 – Integração do sistema de informações de manufatura (FONTE: PAOLLUCI e SACILE, 2005).

Neste sentido, a integração pode se dar em relação a uma dimensão vertical, isto é, possibilitando a integração entre diferentes plantas e processos de negócios. E, 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 2.11 (PAOLUCCI e SACILE, 2005).

Dentro deste escopo, 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 seqüenciamento. Além de calcular as datas de fornecimento, 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 (PAOLUCCI e SACILE, 2005).

No tocante ao desempenho requerido pelo processo de recuperação de informação de manufatura, dadas as extensões geográficas e aos variados desdobramentos das modernas organizações de produção, percebe-se, claramente, que um sistema de informações centralizado não representa mais uma alternativa de solução apropriada (PAOLUCCI e SACILE, 2005).

Neste cenário, os principais conceitos relacionados aos sistemas de informação de manufatura envolvem o uso do potencial da *Internet* e das *Intranets,* como uma infra-estrutura de comunicação poderosa para a distribuição e compartilhamento de informações entre e dentro das modernas organizações de produção. E, 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).

Não obstante, a tecnologia de agentes também encontra um campo de aplicação promissor na área de engenharia simultânea, em particular no desenvolvimento integrado de produto e planejamento de processos, cuja finalidade é aumentar a capacidade dos sistemas atuais de: projeto auxiliado por computador (CAD), de manufatura auxiliada por computador (CAM) mediante a integração destes sistemas aos sistemas de planejamento de processos auxiliado por computador (CAPP), bancos de dados de fatores de projeto e bases de conhecimentos existentes, tais como: forma, material, tolerância, acabamento superficial, entre outros, e os conhecimentos que permitem mapear estes fatores de projeto para os possíveis processos de manufatura e seus respectivos recursos (FENG, 2005).

Neste sentido, a literatura a revela alguns problemas fundamentais, em particular, para a integração do projeto preliminar e o planejamento do processo, onde a tecnologia de agentes pode contribuir, significativamente, para uma solução factível, entre os quais Feng (2005) relata:

- Não interoperabilidade entre os softwares legados de CAD/CAM e CAPP envolvidos no projeto preliminar e planejamento de processos.
- A natureza do ambientes computacionais envolvidos que são, usualmente, distribuídos.
- Falta de ambientes colaborativos baseados na internet envolvendo o projeto preliminar e planejamento, o que impede atividades colaborativas em ambientes de computação distribuída.

2.3.4 Aplicações da tecnologia de agentes na gestão do conhecimento

No contexto da gestão do conhecimento, é consenso na literatura que organizações de sucesso são aquelas que criam novos conhecimentos e o disseminam, amplamente, através da organização e, rapidamente, o incorporam em novas tecnologias, produtos e processos. Assim, este processo alimenta a inovação e o desenvolvimento culminando em vantagem competitiva para a organização (METAXIOTIS et al., 2005).

Justifica-se, assim, o número crescente de organizações que vêm se conscientizando que as suas próprias redes internas de computadores (*intranets*) são repositórios de valiosas informações e conhecimentos corporativos, embora sem, ainda, entender como aplicar estas informações de modo efetivo e útil (LUCK at al., 2004).

Desta perspectiva, a gestão deste conhecimento diz respeito a um conjunto de processos distintos, porém interdependentes que compreendem: a criação, a armazenagem, a recuperação, a transferência e a aplicação destes conhecimentos. E, por conseqüência, a cada instante a organização e seus membros podem estar envolvidos em uma múltipla cadeia de processos de gestão de conhecimento, e como tal, a gestão do conhecimento não é monolítica, mas um fenômeno organizacional contínuo e dinâmico (ALAVI e LEIDNER, 2001).

Adicionalmente, o trabalho de Cook e Cook (2000) sugerem que 80 % das informações referentes aos negócios são de natureza não-quantitativa ou estruturada de uma forma que não podem ser capturadas em uma base de dados relacional.

Além disto, a complexidade, os recursos requeridos, as ferramentas subjacentes e a abordagem variam de acordo com o tipo, escopo e as características de cada processo da gestão do conhecimento (COOK e COOK, 2000). Portanto, transformar informações em recursos úteis torna-se um grande desafio tecnológico.

Neste cenário, os agentes de informação, tipicamente, podem acessar múltiplas, heterogêneas e geograficamente distribuídas fontes de conhecimentos através da *internet* ou em *intranet* (KLUSCH, 2001). Esta funcionalidade inclui: recuperação, análise, manipulação e integração de informações a partir de fontes de informação autônomas, distintas e distribuídas geograficamente (KLUSCH, 2001).

Neste sentido, o agente deveria ser capaz de apresentar diferentes perspectivas da informação ao usuário por meio de um processo de fusão de dados e conhecimentos heterogêneos (KLUSCH, 2001). Assim, a tecnologia de agentes pode ser vista como uma tecnologia viável para a gestão do conhecimento.

2.4 RACIOCÍNIO BASEADO EM CASOS

Na última década, a técnica de Raciocínio Baseado em Casos (RBC)[11] evoluiu de uma área de pesquisa isolada e específica da Inteligência Artificial (IA) para um campo de interesse amplo no desenvolvimento de Sistemas de Gestão do Conhecimento[12] (WATSON, 2003).

Esta evolução pode ser evidenciada pelo crescimento do número de artigos publicados na literatura científica, pelos relatórios sobre aplicações em ambientes industriais e comerciais, e também pela participação crescente de pesquisadores e empresas em conferências específicas

[11] Do inglês, *Case-Based Reasoning (CBR*).

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

dedicadas a área, tais como: a Conferência Internacional sobre Raciocínio Baseado em Casos[13], com a primeira em 1995 e, ainda a Conferência Européia sobre Raciocínio Baseados em Casos[14], cuja primeira edição ocorreu em 1993, as quais se alternam a cada ano, respectivamente.

Neste cenário, entre os estudos pioneiros 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 prévia, mais similar, a partir de uma base de casos. E, se necessário, esta solução recuperada ou a estratégia de solução do problema pode ser adaptada usando-se o conhecimento geral do domínio. E, ainda, mediante 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.

Por sua vez, Aamodt e Plaza (1994) observam também que o termo "solução de problemas no âmbito do raciocínio baseado em casos" é usado em um sentido amplo, coerente com a prática comum dentro da área de sistemas baseados em conhecimento. Onde, em geral, resolver um problema não significa, necessariamente, descobrir uma solução concreta para um problema requerido, mas também apoiar a decisão do usuário, justificar ou criticar uma solução proposta pelo usuário, interpretar a situação problema, gerar um conjunto de soluções possíveis ou gerar expectativas sobre dados observáveis.

Neste sentido, a idéia básica subjacente a um sistema de raciocínio baseado em casos é que, para determinados domínios, em particular, os problemas a serem resolvidos tendem a ser recorrentes e repetir-se com pequenas variações em relação à sua versão primária. Deste modo, soluções prévias podem ser reaplicadas também com pequenas modificações.

Portanto, em essência, o raciocínio baseado em casos envolve as tarefas de: identificar a atual situação-problema, encontrar um caso passado similar ao novo caso, usar este caso prévio para sugerir uma solução para o corrente problema, avaliar a solução proposta e atualizar a base de conhecimento do sistema pelo aprendizado obtido a partir desta experiência.

Na atualidade, 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

[13] Do inglês, *Internacional Conference on Case-Based Reasoning (ICCBR).*

[14] Do inglês, *European Conference on Case-Based Reasoning (ECCBR*).

aplicabilidade da técnica de raciocínio baseado em casos (WATSON, 2003; von WANGENHEIM e von WANGENHEIM, 2003; SHIU e PAL, 2004; BANDINI et al., 2004):

- O domínio de aplicação não dispõe de modelos gerais de conhecimento estabelecidos;
- Existem exceções e novos casos disponíveis;
- Os casos, em geral, são recorrentes;
- Casos prévios relevantes podem ser obtidos;

Adicionalmente, a literatura observa, ainda, que os principais aspectos relacionados à aplicabilidade da técnica de raciocínio baseado em casos referem-se a (SHIU e PAL, 2004):

- 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 obter um conjunto de regras de especialistas humanos. Neste sentido, a técnica consiste em: coletar um conjunto de experiências relevantes / casos existentes, representá-los e armazená-los adequadamente.
- 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. As informações sobre o que causou a falha no passado podem ser usadas para predizer as falhas em potencial no futuro.
- Permitir a flexibilidade na modelagem do conhecimento, pois os sistemas baseados em modelos causais, devido à sua rigidez na formulação e modelagem do problema, algumas vezes, 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 freqüência, 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 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) não sendo, assim, 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 corrente contexto 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.

Portanto, a idéia básica subjacente a um sistema de raciocínio baseado em casos é que, para determinados domínios, em particular, os problemas a serem resolvidos, tendem a ser recorrentes e repetir-se com pequenas variações em relação à sua versão primária. Deste modo, soluções prévias podem ser reaplicadas também com pequenas modificações.

Desta perspectiva, os sistemas baseados em conhecimento denominados como raciocínio baseado em casos, representam, na verdade, uma classe genérica de sistemas que é composta por tipos específicos de sistemas em função dos diferentes métodos de organização, recuperação, utilização e indexação dos conhecimentos retidos nos casos prévios (AAMODT e PLAZA, 1994; WATSON, 2003).

Nestes diversos métodos, os casos podem ser mantidos como experiências concretas, ou um conjunto de casos similares pode formar um caso generalizado. Os casos podem ser armazenados como unidade de conhecimentos única ou separados em partes distintas, formando subunidades distribuídas em uma estrutura de conhecimento. Estes casos podem ser indexados por um vocabulário prefixado e aberto e dentro de uma estrutura de índices plana ou hierárquica (AAMODT e PLAZA, 1994; WATSON, 2003).

Ademais, a solução obtida a partir de um caso prévio pode ser diretamente aplicada ao problema em questão 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, raso ou compilado ou ser baseado, unicamente, em uma aparente similaridade. Além disso, os métodos podem ser ainda totalmente completos e automáticos ou podem ser fortemente interativos com os usuários para apoio e condução de escolhas (AAMODT e PLAZA, 1994; WATSON, 2003).

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. Este ciclo tem por objetivo enfatizar a idéia de ciclo em etapas seqüenciais, permitindo uma visão global e externa do processo, no qual são identificados os subprocessos, suas interdependências e

produtos. E, neste sentido, Aamdot e Plaza (1994) sugerem um modelo composto dos quatro processos seguintes (4RE's)[15], como mostra a figura 2.12:

- Recuperação do caso ou casos mais similar(es);
- 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.

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 denominada de base 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.

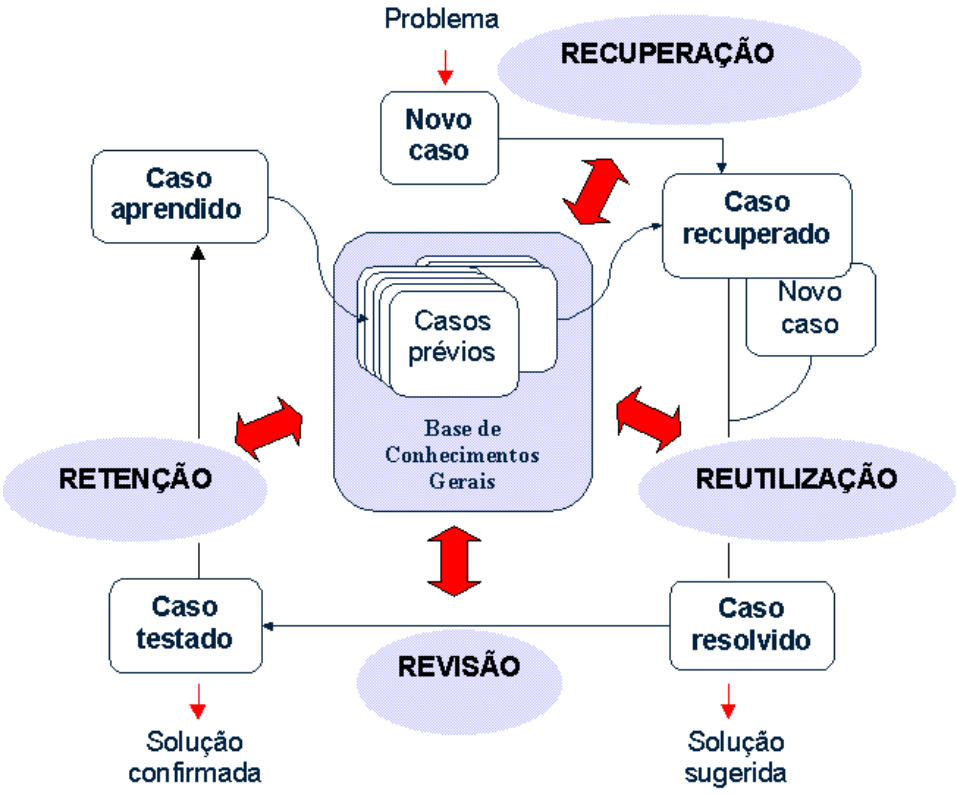


Figura 2.12 – Ciclo RBC (Fonte: AAMODT e PLAZA, 1994).

[15] Em referência ao mnemônico da primeira sílaba dos sub-processos.

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

Por fim, o processo de retenção deve armazenar a informação útil para futuras reutilizações e a base de casos prévios deve ser atualizada com o novo caso aprendido.

Neste modelo, a base de conhecimento geral (modelos causais) indicada na figura, normalmente contribui para o desenvolvimento do ciclo RBC. Esta contribuição pode ser representada em um continuum, tendo em um extremo uma contribuição insignificante ou mesmo nenhuma e, em outro extremo, uma contribuição essencial em função do método de RBC em questão.

Neste contexto, este conhecimento significa um conhecimento genérico e dependente do domínio, em contraste ao conhecimento específico incorporado nos casos. Por exemplo, em um diagnóstico médico realizado com a recuperação e reutilização de casos de pacientes prévios, o modelo de anatomia humana em conjunto com as relações causais entre os estados patológicos dos casos ou regras podem constituir-se em conhecimento geral a ser usado pelo sistema RBC (AAMODT e PLAZA, 1994; WATSON, 2003).

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, 2002 e 2003).

2.4.1 Representação de conhecimento para Raciocínio Baseado em Casos

Em primeiro lugar, é fundamental estabelecer-se um entendimento preciso do termo caso e, nesta linha, adota-se a definição proposta por Kolodner (1993), onde um caso é "um fragmento do conhecimento representando a experiência que ensina uma lição fundamental para atingir o objetivo do raciocínio". Um caso, normalmente, compreende:

- A situação-problema: que descreve o estado do mundo real quando o caso ocorreu;
- A solução: que postula uma solução derivada para aquele problema, podendo ser uma ação, um plano ou uma informação útil ao usuário, e/ou;
- O resultado: que descreve o estado do mundo real após implementar-se a solução proposta, ou, ainda, quão bem a solução resolveu o problema.

Contudo, parece não haver um consenso no que diz respeito a como o problema deve descrito, isto é, seu conteúdo. Os estudos de Kolodner (1993) sugerem duas medidas pragmáticas que podem ser levadas em conta para decidir o conteúdo a ser descrito: a

funcionalidade e a facilidade de aquisição da informação; e, por sua vez, von Wangenheim e von Wangenheim (2003) sugerem que partes significativas da descrição do problema podem incluir:

- Objetivos, os quais devem ser atingidos pela solução do problema (por exemplo, sugerir um diagnóstico de falha de um componente).
- Restrições, às quais os objetivos estão sujeitos (por exemplo, evitar ensaios destrutivos).
- Atributos, da situação-problema e o relacionamento entre suas partes (por exemplo, taxa de falhas do componente, temperatura de trabalho, característica do ambiente e etc.). De modo geral, todas as informações consideradas essenciais para se atingir os objetivos.

Nesta ótica, o conteúdo das soluções deverá variar com o tipo de aplicação do sistema RBC, e pode incluir (von WANGENHEIM e von WANGENHEIM, 2003):

- Uma solução em si;
- Um conjunto de etapas de raciocínio necessárias para resolver um problema;
- Explicações para as decisões tomadas durante a solução do problema;
- As soluções alternativas viáveis, mas não sugeridas e suas respectivas explicações;
- As soluções inviáveis que foram excluídas do processo e suas respectivas explicações;
- Expectativas a respeito do ambiente após a implementação da solução sugerida e possíveis problemas.

Similarmente, o conteúdo do resultado pode incluir informações detalhadas do mundo real com respeito à implementação da solução sugerida entre outras (von WANGENHEIM e von WANGENHEIM, 2003):

- O resultado em si;
- Se o resultado atingiu ou violou as expectativas e as devidas explicações;
- Sobre o êxito ou não da solução sugerida e as devidas explicações;
- As possíveis causas da falhas e alternativas para correção

Por outro lado, a representação do caso significa codificar o conhecimento contido nos casos mediante uma ampla gama de formalismos representacionais ou linguagens desenvolvidas no âmbito da IA, permitindo armazená-lo em uma forma simples, concisa, completa e clara. Isto é, não contendo suposições ou elementos ambíguos de forma (RUSSEL e NORVIG, 2004).

A seguir são apresentados dois dos principais formalismos relatados na literatura: a representação por lógica descritiva e por vetores atributo-valor.

2.4.1.1 Representação de casos por meio de vetores atributo-valor

O formalismo mais simples para a representação do conhecimento no contexto dos sistemas de raciocínio baseado em casos é por meio dos vetores atributo-valor, e esta forma de representação pode resolver grande parte dos problemas de aplicação de RBC (WATSON, 2003; von WANGENHEIM e von WANGENHEIM, 2003).

Neste formalismo de representação, a linguagem de descrição de exemplos procura descrever um objeto (instâncias) do mundo real mediante um vetor de valores de atributos, onde um atributo é uma característica ou uma informação que visa descrever o objeto, como mostra a tabela 2.13.

Tabela 2.13 – Exemplo de representação com vetores atributo-valor.

Objetivo do sistema: **Especificação de motor elétrico de indução**	
Caso 001	
Atributo	**Valor**
Faixa de altitude que o motor deverá operar:	1500 [m]
Temperatura máxima de trabalho: Temperatura mínima de trabalho:	60 [ºC] -40 [ºC]
Ambiente de operação:	agressivo

A escolha dos pares atributo e valor deve levar em conta sua relevância em relação ao objeto (caso) representado e tem como vantagens: simplicidade na representação, simplificação das medidas de similaridade, facilidade de armazenar em bases de dados com fácil recuperação. Entretanto, apesar desta representação ser a mais utilizada nos sistemas RBC, outras representações mais complexas contendo modelos de relações entre atributos ou conjunto de atributos podem ser necessárias para representar estruturas de informação mais complexas (von WANGENHEIM e von WANGENHEIM, 2003).

2.4.2 Tarefa de Recuperação de casos

Uma visão orientada à tarefa permite descrever, detalhadamente, o mecanismo de raciocínio a partir da perspectiva interna ao processo. Nesta linha, Aamodt e Plaza (1994) revelam que uma descrição do sistema pode ser feita a partir de três perspectivas: tarefas, métodos e modelos de domínio de conhecimento, como mostram a figura 2.13.

As tarefas podem ser entendidas como um conjunto de objetivos do sistema de raciocínio, as quais são representadas nas figuras pelos nomes dos nós (letra em negrito), todas as tarefas envolvidas no ciclo RBC estão representadas e algumas são decompostas em subtarefas necessárias para completar o ciclo (ligadas por linhas cheias).

As tarefas são executadas pela aplicação de um ou mais métodos, os quais especificam os algoritmos que identificam e controlam a execução das tarefas e acessam e utilizam os conhecimentos e informações necessárias ao processo (em letras normais, ligados por linhas pontilhadas).

É importante ressaltar que os métodos apresentados na figura 2.13 representam, na verdade, uma superclasse de métodos. E, portanto, um ou mais métodos podem ser necessários para completar a tarefa ou, ainda, algumas tarefas de nível mais baixo na hierarquia podem ser completadas, diretamente, por um método de execução de tarefa (AAMODT e PLAZA, 1994).

A partir desta percepção, é importante ressaltar que não existe um método universal de raciocínio baseado em casos apropriado para qualquer domínio de aplicação, assim o desafio é determinar quais métodos melhor se adaptam para a solução e o aprendizado em determinados domínios e ambientes de uma dada aplicação em particular.

Desta perspectiva, a tarefa de recuperação de casos tem por finalidade encontrar a melhor correspondência (*match*) entre uma descrição parcial de uma situação-problema (denominada de novo caso) e os casos prévios armazenados em uma base de conhecimento.

Com este objetivo, a tarefa é dividida em quatro subtarefas como mostra a Figura 2.13: identificar as entidades de informação (*features*), pesquisar índices, encontrar uma correspondência inicial de casos potencialmente candidatos, e selecionar a melhor correspondência.

Neste cenário, algumas questões essenciais devem ser esclarecidas, tais como: (i) o que significa, precisamente, um caso útil para a solução de um problema? (ii) como se pode medir a utilidade de um caso?

A primeira questão diz respeito à hipótese básica subjacente aos sistemas de raciocínio baseado em casos, a qual sugere que problemas similares possuem soluções similares. Deste

modo, o critério a posteriori da utilidade da solução fica condicionado, necessariamente, ao critério a priori de similaridade das descrições de um problema (von WANGENHEIM e von WANGENHEIM, 2003).

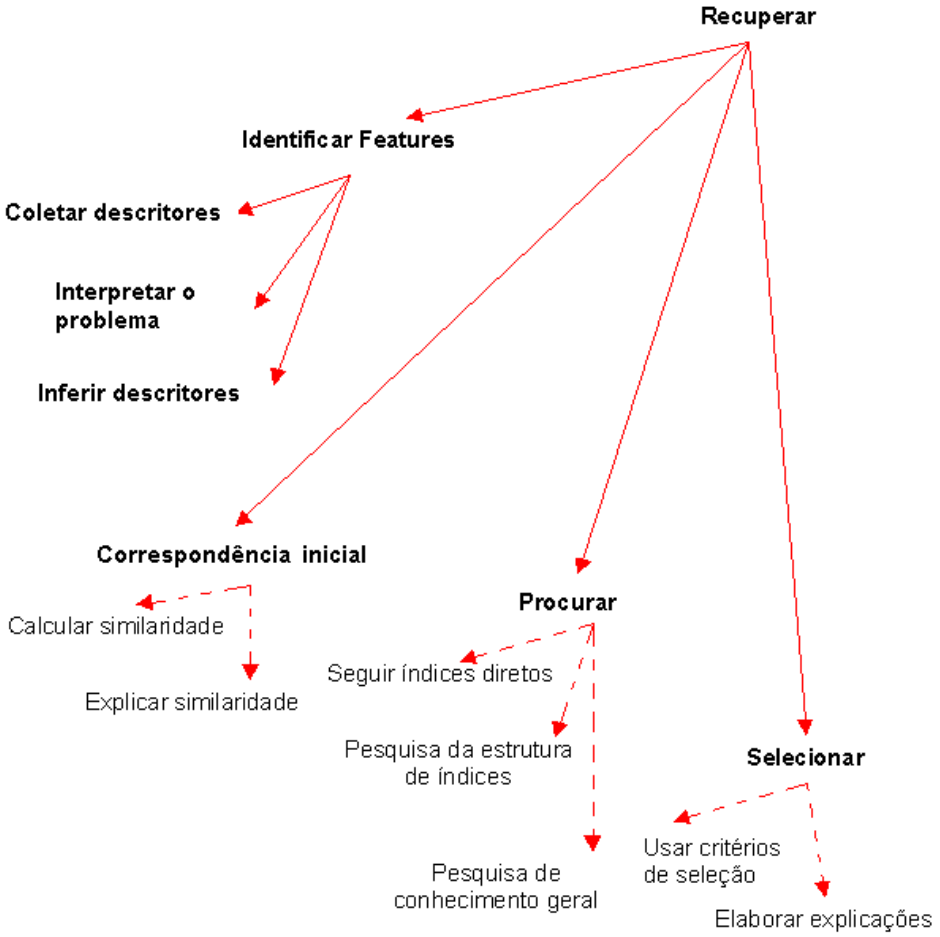


Figura 2.13 – Estrutura hierárquica da tarefa de recuperação de casos (Fonte: AAMODT e PLAZA, 1994).

Assim, o conceito abstrato de utilidade de um caso torna-se um aspecto essencial da tarefa de recuperação e, segundo von Wangenheim e von Wangenheim (2003), um caso pode ser considerado útil quando:

- Permite a solução do novo problema de alguma forma;
- Evita a repetição de um erro anterior;
- Permite uma solução eficiente do problema, isto é, mais rápida do que aquela obtida por uma solução sistemática (passo a passo);
- Permite oferecer a melhor solução para o problema de acordo com um dado critério de ótimo[16];

[16] O conceito de *ótimo* envolve a idéia de uma solução não necessariamente a melhor possível, mas uma solução que depende de vários critérios, em que a melhora de um deles implica, necessariamente, a piora de outro.

- Permita oferecer ao decisor uma solução cuja explicação lógica possa ser compreendida e aceita.

A segunda questão envolve o conceito de similaridade em um domínio de aplicação específico, e este conceito pode ser determinado levando-se em conta os seguintes aspectos (von WANGENHEIM e von WANGENHEIM, 2003).

- A definição de cenários de uso dos casos e as respectivas metas de recuperação;
- A determinação das entidades de informação (EI) importantes para a determinação da similaridade;
- A definição de métricas de similaridades.

2.4.2.1 Conceitos de similaridade em raciocínio baseado em casos

De forma intuitiva, os julgamentos sobre similaridade podem ser entendidos como um conceito que descreve a impressão de semelhança entre dois objetos, experimentada por observadores humanos (von WANGENHEIM e von WANGENHEIM, 2003). No entanto, modelos de similaridade buscam estabelecer os axiomas subjacentes ao processo de julgamento de similaridade, realizado por humanos com o objetivo de formalizar estes julgamentos. Neste sentido, algumas premissas devem ser assumidas, tais como:

- Reflexidade: um objeto é similar a si mesmo.
- Simetria: se um objeto A é similar a B, então B também é similar a A.
- Transitividade: se um objeto A é similar a B e B é similar a C, então A também e similar a C.
- Monotonicidade: a similaridade de dois objetos cresce monotonicamente com o aumento de correspondências e a redução das diferenças.

Todavia, não existe uma teoria universal da determinação da similaridade, tendo em vista os muitos aspectos que devem ser considerados, bem como pela incerteza associada ao processo e, em geral, a sua determinação é feita em função do cenário de aplicação (von WANGENHEIM e von WANGENHEIM, 2003).

No sistema de raciocínio baseado em casos os conceitos de similaridade são formalizados por meio de três formas(von WANGENHEIM e von WANGENHEIM, 2003):

- Como predicado, isto é, uma relação entre objetos ou fatos.

- Como relação de preferência, considera a noção de uma similaridade maior ou menor entre objetos ou fatos.
- Como medida, que sugere a quantificação por meio de uma medida numérica de distância ou similaridade.

E, ainda, devido à ampla variedade de tipos e escalas dos valores dos atributos, a medida de similaridade deve ser selecionada cuidadosamente e as características utilizadas para definir similaridade devem ser especificadas e combinadas em uma medida a ser calculada para todos os pares de casos (von WANGENHEIM e von WANGENHEIM, 2003).

Uma medida de similaridade sobre um universo U é uma função do tipo:

$$sim(x, y) : (U \times U \rightarrow [0,1] \tag{1}$$

para

$\forall x \in U$	$sim\ (x,x) = 1$	reflexidade
$\forall x, y \in U$	$sim\ (x,y) = sim\ (y,x)$	simetria

Pela equação (1) pode-se perceber que os casos de uma base podem ser organizados segundo um valor concreto de similaridade com intervalo [0,1] no domínio dos números reais. Esta medida considera a correspondência de atributos entre o novo caso e o caso da base, em geral, procura-se a máxima similaridade, isto é:

$$\exists y \in CB\left(\forall z \in CB\right) \qquad \boldsymbol{sim\ (x,y) \geq sim\ (x,z)} \tag{2}$$

Todavia, esta medida pode representar também similaridades média, mínima ou uma interpretação geométrica – medida de distância (TECINNO, 1999).

Em relação à interpretação geométrica, pode-se considerar que após determinação dos atributos relevantes para descrever o novo caso, vetores de valores desses atributos podem ser utilizados para descrever todo o conjunto de casos. Considerando os atributos como dimensões de um espaço multidimensional, a descrição de cada caso corresponde a um ponto no espaço; então, a similaridade entre os casos pode ser medida usando uma função que calcula a distância entre os pontos no espaço de descrição ao dos casos (WATSON, 2003).

As medidas de distância são, na verdade, medidas de dissimilaridade, pois quanto maior o valor da medida de distância, menor a similaridade. As medidas de distâncias descritas na literatura da área de recuperação de informações e gerência de banco de dados são: *nearest-*

neighbour[17], *nearest-neighbour* ponderado Euclidiana, Euclidiana Ponderada, *Manhattan*, *Minkowsky*, *Mahalanobis*, *Camberra*, *Chebychev*, *Chi-quadrado* e *Hamming*. E, dentre elas, as mais utilizadas no contexto dos sistemas RBC são, segundo von Wangenheim e von Wangenheim (2003), medida de distância *nearest-neighbour*, *nearest-neighbour* ponderado, Euclidiana, Euclidiana Ponderada e *Hamming*.

Adicionalmente, um outro tipo de medida de similaridade é a medida de associação, as quais são utilizadas para comparar casos cujas características são medidas apenas para atributos com valores discretos, como por exemplo, para valores de atributos que poderiam ser "sim" ou "não". Assim, uma medida de associação poderia verificar o grau de concordância entre cada par de casos (MARTINS, 2003). É importante ressaltar que as diversas medidas devem satisfazer as premissas assumidas pelos modelos formais.

Assim, a partir do conceito formal de similaridade e medidas de similaridade apresentadas acima, os modelos para a determinação de similaridade entre descrição de um novo caso e os casos já indexados e armazenados na memória de casos podem ser estabelecidos (vonWANGENHEIM e von WANGENHEIM, 2003), entre os quais os modelos:

- Para a determinação da similaridade global (medidas de similaridade entre todos os índices do caso da base);
- Para determinação de similaridade local (medidas de similaridade entre os atributos específicos);
- Para determinação de similaridade entre objetos (medidas de similaridade específicas aplicadas quando a representação dos casos adota o paradigma de orientação a objetos, que aproveitam o conhecimento contido na hierarquia de classes).

2.4.2.2 Conceito de similaridade global

A medida de similaridade global envolve o cálculo da similaridade da descrição do novo caso (entidades de informação) e um caso da base levando em consideração todos os índices deste caso, realizados usando a função dada na equação (1).

Segundo Watson (1997, 2003) a técnica de *nearest-neighbour* é a mais simples das medidas de similaridade, e freqüentemente é usada em sistemas de RBC. Em essência, trata-se de uma medida de distância espacial entre os pontos, correspondentes aos valores de

[17] Cuja tradução para o português é vizinho-mais-próximo.

atributos do novo caso e do caso da base em avaliação, em um espaço multidimensional. Como exemplo, um caso simples indexado por: rendimento mensal líquido [R$] e pagamento mensal do empréstimo [R$] podem ser usados para considerar o risco de um empréstimo em um contexto bancário, como mostram as Figuras 2.14, 2.15, 2.16 e 2.17.

Na figura 2.14, o cálculo da medida de similaridade é realizado considerando a distância do Caso T (novo caso) até o A e B, respectivamente:

$$d_A = X_A + Y_A \tag{3}$$

$$d_B = X_B + Y_B \tag{4}$$

Nesta técnica, o caso que apresentar a menor distância (d) em relação à descrição do novo caso será considerado o vizinho-mais-próximo. Não obstante a simplicidade da técnica, a mesma pode ser aplicada em situações com múltiplos índices, implicando em um espaço multidimensional, e ainda suportar atributos com valores simbólicos, *booleanos* e textuais.

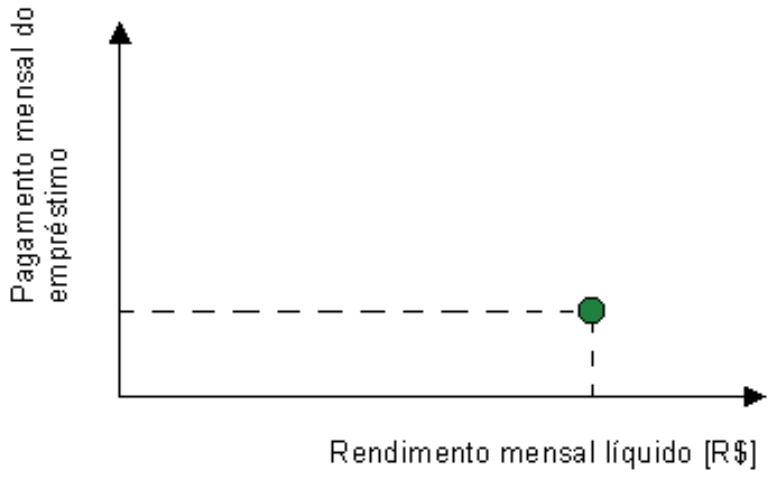


Figura 2.14 – Distância dos vizinhos mais próximos (Fonte: WATSON, 1997).

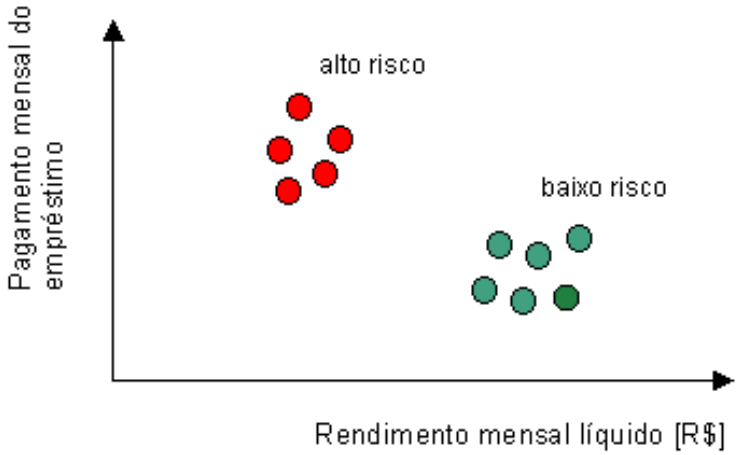


Figura 2.15 - Agrupamento de casos em função dos índices (Fonte: WATSON, 1997).

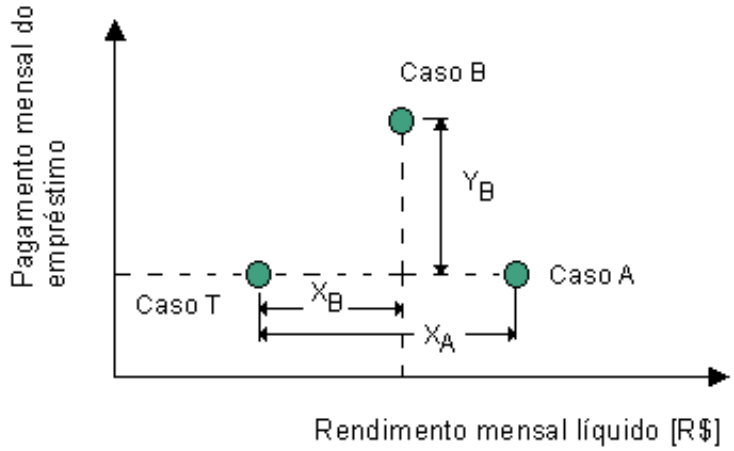


Figura 2.16 - Distâncias entre os casos (Fonte: WATSON, 1997)

Adicionalmente, esta técnica pode considerar também a importância relativa de cada índice, mediante a ponderação destes índices, passando a denominar-se *nearest-neigbour* ponderado, com as seguintes modificações:

$$d_A = (X_A \times W_x) + (Y_A \times W_y) \quad (5)$$

$$d_B = (X_B \times W_x) + (Y_B \times W_y) \quad (6)$$

É importante notar que a definição dos pesos (Wx ; Wy) deve levar em consideração as metas de recuperação, em geral baseadas em engenharia do conhecimento.

Em resumo, algoritmos *nearest-neighbour* e *nearest-neighbour* ponderado são usados amplamente nos sistemas de RBC, as medidas de distância são normalizadas para uma faixa de [0,1], onde 0 representa a dissimilaridade máxima e 1 a correspondência absoluta.

Por sua vez, a distância euclidiana representa a distância espacial real entre os pontos, correspondentes aos valores de atributos do novo caso e do caso da base em avaliação em um espaço multidimensional, dada pela função:

$$d(T,C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \quad (7)$$

onde T representa o novo caso e C o caso da base, dados pelo par atributo-valor:

$$Q = (q_1, q_2, ..., q_n) \quad e \quad C = (c_1, c_2, ..., c_n) \quad (8)$$

Adicionalmente, é possível introduzir pesos para considerar também a importância relativa entre os atributos, para uma medida euclidiana ponderada, como mostra a equação 7.

$$d(T,C) = \sqrt{\sum_{i=1}^{n} w_i \times (q_i - c_i)^2} \quad (9)$$

A literatura revela, ainda, outras possibilidades para a representação compacta, tais como: medidas de similaridade como tabelas de contingências, medidas de similaridade invariantes e modelos de contraste aplicados em casos específicos de indexação (TECINNO, 1999).

2.4.2.3 Conceito de similaridade local

A medida de similaridade local representa a similaridade entre atributos relevantes específicos da descrição do novo caso e do caso da base sob análise. No item anterior, a função de cálculo da similaridade global considera a similaridade local como sendo igual a 1, isto é, similaridade absoluta entre os atributos específicos; logo, somente os casos com atributos com valores iguais eram possíveis de serem distinguidos de outros durante o cálculo.

Entretanto, em algumas metas de recuperação, um caso, mesmo com o valor de um atributo específico diferente da descrição do novo caso, poderia ainda ser considerado como

similar. Como exemplo, a tabela 2.14 apresenta os valores para alguns atributos considerados em tarefa de recuperação, onde se pode observar que o atributo específico "modelo" conta com valor de similaridade local, o qual pode ser integrado, então, ao cálculo da medida de similaridade global, tornado-a muito mais expressiva (TECINNO, 1999).

Neste contexto, diversas funções podem ser estabelecidas para o cálculo das medidas de similaridade local, levando em consideração os diversos tipos possíveis de valores para os atributos específicos, entre eles (TECINNO, 1999):

- Tipo numérico: funções escada, funções lineares, funções assintóticas polinomiais;
- Tipos não-numéricos: escalas ordinais para símbolos ordenados (por exemplo, temperatura baixa, média e alta), matrizes de similaridade para símbolos não-ordenados (por exemplo: comando *Siemens*, *Fagor*, *Fanuc*) e símbolos taxionômicos (por exemplo, valores similaridade para nós internos de árvores ou tamanho do caminho entre objetos da árvore);
- Tipo conjuntos: elementos máximos, intervalos;
- Tipo string: correspondência exata, correção ortográfica, contagem de palavras e taxa do maior *substring*.

Tabela 2.14 – Exemplos de valores de similaridade local

Atributos específicos	Descrição: Novo caso	Caso 1 (da base)	Valor: similaridade local
Descrição da falha	Máquina não fecha placa autocentrante	Máquina não liga	0
Modelo	Cosmos 10 G	Cosmos 10 U	0,6
Luz indicadora do estado do comando	Apagada	Apagada	1,0

2.4.2.4 Conceito de similaridade entre objetos de classe

As medidas de similaridade para representações orientada a objetos são definidas pela similaridade entre dois objetos, onde um objeto representa o caso da base ou parte deste e o outro a descrição do novo caso. A literatura apresenta duas abordagens para o cálculo da medida de similaridade entre objetos.

Na primeira abordagem a similaridade é calculada de forma recursiva de cima para baixo na hierarquia de classes, usando as medidas de similaridade local para os atributos específicos dos objetos (objeto do novo caso e objeto do caso da base). Em seguida os valores da

similaridade são agregados por meio de uma soma ponderada, por exemplo, resultando assim em uma medida similaridade entre os objetos sob análise (TECINNO, 1999).

A segunda adota o conhecimento contido na hierarquia de classe (estrutura do objeto), e tem como base a idéia que objetos mais próximos na hierarquia de classe são mais similares que os objetos mais distantes (por exemplo, a instância CD-R é mais similar à instância CD-RW do que a instância *floppy Disk*, como mostra a figura 2.17). Do ponto de vista prático, em implementações de sistemas de RBC, isto significa comparar as instâncias em uma mesma classe - intraclasse e de classes diferentes – interclasses (TECINNO, 1999).

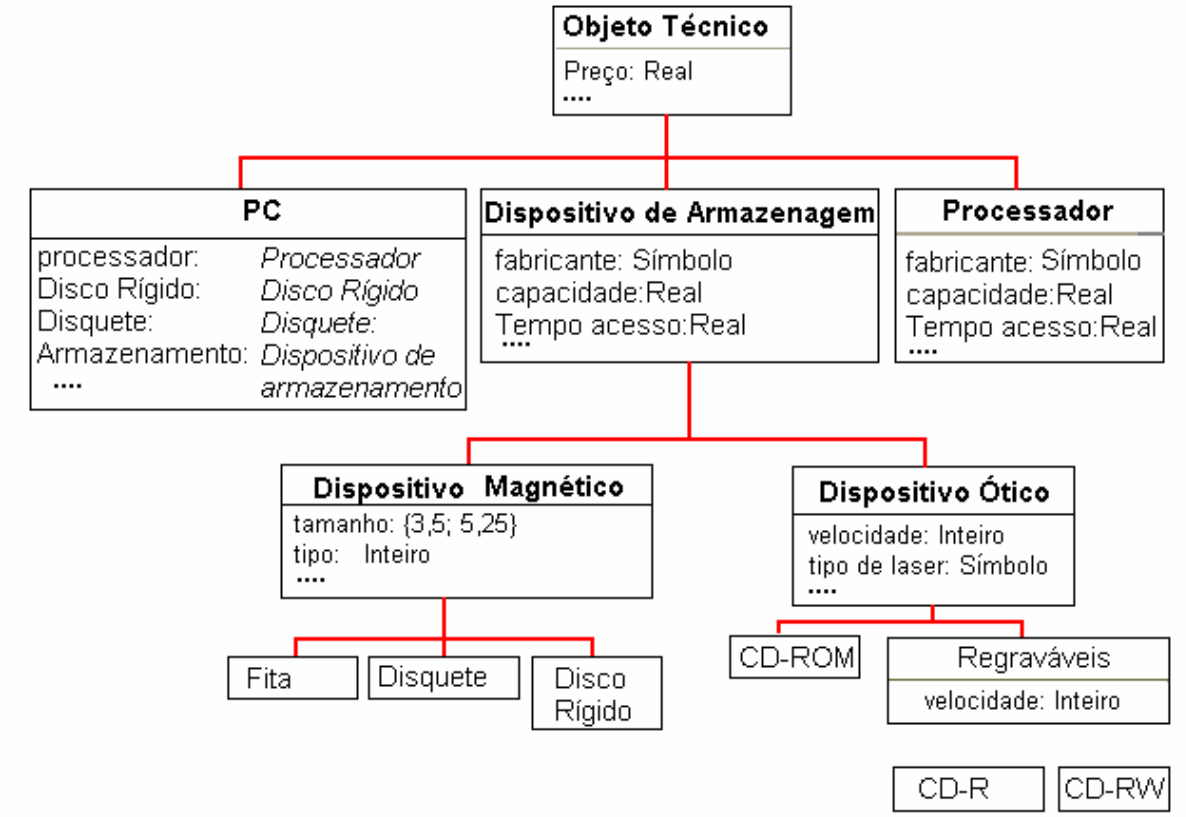


Figura 2.17 - Medida de similaridade em hierarquia de classes. (FONTE: TECINNO, 1999)

2.4.3 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 concreto contido no caso (descrição da solução) para o novo caso. Neste sentido, esta tarefa pode ser subdividida em duas subtarefas: copiar (a forma mais trivial de reutilizar o caso) e adaptar a solução considerando as diferenças entre o novo caso e o recuperado, bem como identificando qual parte da solução pode ser transferida para o novo caso.

Nesta linha, Wilke e Bergmann (1998) observam que existe uma relação entre a complexidade da tarefa de solução do problema e a complexidade do processo de adaptação no ciclo de RBC, e sugerem que a própria adaptação é um tipo especial de solução de problema que envolve: a descrição do novo problema, problemas similares e suas respectivas soluções, os quais em conjunto com um algoritmo especial de solução possam permitir uma solução mais eficiente do que uma solução sistemática (passo a passo).

Segundo Leake (1996), os sistemas de RBC que têm por objetivo: classificar, realizar diagnósticos e apoiar à decisão, podendo-se contornar o problema de adaptação pelo uso de uma base de casos mais extensa ou, ainda oferecer ao decisor a possibilidade de pesquisar outros casos relacionados de acordo com a situação ou, ainda, realizar as alterações necessárias interativamente.

Por outro lado, um sistema de RBC cujo objetivo é realizar tarefas de síntese, tais como projeto e planejamento de soluções, requer processos de adaptação automáticos de solução, pois, em geral, o espaço de soluções é muito amplo para um dado contexto.

2.4.4 Tarefa de Revisão de casos

A solução gerada (a partir do processo de adaptação) 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 gerada. Neste contexto, se a solução gerada não é adequada, surge uma oportunidade para a correção da falha e, conseqüentemente, a possibilidade da aprendizagem sustentada do sistema de RBC.

A primeira etapa do processo de revisão envolve a subtarefa de avaliação, que em geral é externa ao 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. Por exemplo, em aplicações de diagnóstico médico, a confirmação de uma solução pode demorar dias, meses ou mesmo anos; logo, este caso pode permanecer na base de casos, mas deve ser identificado como um caso não validado. Outro exemplo é o sistema CHEF desenvolvido por *Hammond* (apud KOLODNER, 1993), que dispõe de um simulador baseado em modelos, para procedimentos de cozinha, capaz de prover uma informação quanto à adequação da solução (AAMODT e 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.

Um excelente exemplo deste procedimento foi apresentado no sistema CHEF, onde um conhecimento causal é usado para gerar uma explicação de porque certos objetivos do plano de solução não foram atingidos. Neste sentido, o sistema identifica a situação geral que irá causar a falha, usando uma técnica de aprendizagem baseada em explicações. 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.

2.4.5 Tarefa de Retenção de novos casos

A tarefa de retenção de novos casos tem por objetivo incorporar, de forma contínua e sustentada, 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 gerada, 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, visando futuras recuperações e como integrar este novo caso na estrutura da memória de casos.

2.4.6 Aplicações de RBC e trabalhos correlatos

Em síntese, o raciocínio baseado em casos tem sido utilizado para a criação de numerosas aplicações em diferentes áreas e domínios de aplicação, incluindo a análise financeira, assessoramento de riscos, manutenção, controle de processo, controle de qualidade, diagnóstico médico entre outros (WATSON, 2003).

Em relação ao domínio do processo de moldagem por injeção de termoplásticos Mok at al. (2000) apresentam um sistema inteligente híbrido para a determinação de parâmetros iniciais de processo, e Malek et al. (1998) apresentam um sistema baseado em raciocínio baseado em casos para suporte ao operador para o controle de processos de injeção.

2.5 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.

Kitamura e Mizoguchi (2004), 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 estes conceitos conjuntamente com sofisticadas teorias e tecnologias. Assim, a engenharia ontológica pesquisa formas e meios para facilitar o compartilhamento e reúso de conhecimentos de informações relativas à engenharia de manufatura.

2.5.1 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 são apresentadas na literatura nem sempre sob a mesma interpretação.

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.

Neches et al. (1991) revelam que este termo tem uma conotação especial no contexto da organização da informação, e estabelecem assim que uma ontologia "define os termos básicos e relações compreendendo o vocabulário de um tópico de uma área, bem como as regras para combinar termos e relações para definir extensões para o vocabulário".

Por sua vez, Gruber (1993) define uma ontologia como "uma especificação explícita de uma conceitualização", e esta talvez seja uma das definições mais conhecidas para ontologias na área de representação de conhecimento.

Borst (1997) adiciona a idéia de uma "conceitualização compartilhada" à definição de Gruber, assim ontologias são definidas agora como "uma especificação formal de uma conceitualização compartilhada".

Não obstante as definições de Gruber e Borst serem amplamente reconhecidas na área de pesquisa, seu entendimento não é trivial. Neste sentido, Studer et al. (1998) apresentam uma extensão, na qual uma ontologia "é uma especificação formal e explícita de uma conceitualização compartilhada".

É importante observar que *formal* refere-se ao fato de que a ontologia pode ser expressa em uma linguagem formal e manipulável computacionalmente (*machine-readable*) e *explícita* significa que o tipo de conceito e as restrições sobre o seu uso são explicitamente definidos. Por sua vez, *compartilhada* reflete a noção de que uma ontologia captura o conhecimento consensual, isto é, que não é privativo de alguns indivíduos, mas aceito dentro da comunidade (STUDER et al., 1998).

E, nesta definição, a *conceitualização* refere-se a um modelo abstrato de algum fenômeno do mundo real identificado pelos conceitos relevantes deste fenômeno (STUDER et al., 1998).

Posteriormente, Guarino (1998) apresenta uma interpretação do termo ontologia referindo-se "a um artefato de engenharia, constituído por um vocabulário específico usado para descrever certa realidade, mais um conjunto de premissas explícitas e aceitas que dizem respeito ao sentido pretendido para as palavras do vocabulário". A partir desta interpretação, ele propõe a definição de uma ontologia como "uma teoria lógica que responde pelo sentido pretendido de um de um vocabulário formal, isto é, seu compromisso ontológico com uma conceitualização particular do mundo". Para Guarino (1998), o sentido pretendido (significado intencional) é a base para o compartilhamento entre diferentes usuários.

Nesta perspectiva, Uschold e Jasper (1999) enfatizam a importância do vocabulário de termos para uma ontologia, e argumentam que uma ontologia pode assumir diferentes formas, mas sempre irá incluir necessariamente um vocabulário de termos e alguma especificação de seu significado. E isto inclui definições e a indicação de como os conceitos coletivamente estão interrelacionados para impor uma estrutura sobre o domínio e restringindo as interpretações dos termos em questão.

A linha de pensamento de Guarino (1998) considera que a sua definição refina a de Gruber (1993), tornando clara a diferença entre uma ontologia e uma conceitualização, pois a definição de Gruber (1993), para Guarino (1998), não fornece uma consideração intencional (*intensional*) à noção de conceitualização.

Guarino (1998) argumenta que a distinção entre extensão (*extension*) e intenção (*intension*) tem tradicionalmente relação com as formas de expressões, pois podem existir diferentes expressões que podem aplicadas à mesma coisa no mundo real. Por exemplo, considerando as expressões, "A capital da França" e "A Cidade Luz" ambas as expressões se aplicam a Paris, embora a primeira nos traga à mente uma perspectiva política de Paris como a localização oficial do governo Francês, enquanto a segunda relaciona-se a uma perspectiva romântica de uma cidade bonita à noite. Consequentemente, estas expressões têm duas distintas intenções, mas a mesma extensão.

A noção de intenção e extensão de um conceito pode ser explicada também a partir da noção de característica (DAHLBERG, 1978), onde a intenção de um conceito é definida como a soma de todas as suas características (características mais específicas do conceito) e a extensão como a soma de todos os conceitos individuais para os quais a intenção e verdadeira.

Dentro desta lógica, o conhecimento intencional (*intensional knowledge*) refere-se claramente às características dos conceitos, enquanto o conhecimento extensivo (*extensional knowledge*) especifica os indivíduos ou instâncias do conhecimento intencional.

Züñiga (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 esta 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", e "por conseqüência, 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) é devotada a fornecer uma visão geral das diferentes séries de partes da norma, nas quais são definidas a linguagem de especificação de processo (*Process Specification Language – PSL)*. A linguagem *PSL* visa a identificação, definição formal e a estrutura dos conceitos semânticos intrínsecos à captura e troca de informações de processo relacionados 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 existir concordância sobre os seus significados.

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

2.5.2 Lógica de descrições como linguagem formal para ontologias

A pesquisa na área de representação do conhecimento e raciocínio, usualmente, concentra-se no desenvolvimento de métodos para estabelecer descrições de alto nível para o mundo real, as quais possam ser efetivamente usadas para construir aplicações inteligentes (NARDI e BRACHMAN, 2003).

E, neste contexto, o termo "inteligente" refere-se à habilidade do sistema de raciocínio encontrar conseqüências implícitas a partir do conhecimento representado explicitamente e, neste caso, tais sistemas são, portanto, caracterizados com sistemas baseados em conhecimento (NARDI e BRACHMAN, 2003).

Na literatura, as abordagens para a representação do conhecimento desenvolvidas, ainda, nos anos de 1970 são, tipicamente, classificadas em dois grupos: os formalismos baseados na lógica, desenvolvidos considerando a noção de que o cálculo de predicados poderia ser usado de maneira não ambígua para capturar fatos sobre o mundo real; e outras representações não baseadas na lógica, desenvolvidas sobre noções de natureza cognitivas, como por exemplo representações baseadas em regras e estruturas de redes derivadas a partir da experiência recuperada da memória humana e da forma de execução humana de tarefas como a solução de problemas (NARDI e BRACHMAN, 2003).

Nas abordagens não lógicas, baseadas frequentemente no uso de interfaces gráficas, o conhecimento é representado por meio de estrutura de dados *ad hoc*, e o raciocínio é realizado, de modo similar, por procedimentos *ad hoc* que manipulam estas estruturas. Entre estas representações especializadas encontram-se as redes semânticas (*semantic networks*) e os *frames*, enquanto nas abordagens baseadas em lógica, a linguagem de representação é,

usualmente, uma variante do cálculo de predicado de primeira-ordem (NARDI e BRACHMAN, 2003).

Nas estruturas de representação de conhecimento baseadas em redes, de forma geral, os nós (*nodes*) são, tipicamente, usados para caracterizar conceitos, isto é, conjunto de objetos individuais; as ligações (*links*) são usadas para caracterizar as relações entre os nós. Em alguns casos, relações mais complexas podem ser representadas também na forma de nós, contudo, deve existir uma distinção cuidadosa entre este e os nós que representam conceitos.

Adicionalmente, estes conceitos podem incluir propriedades, denominadas de atributos, as quais são anexadas aos nós. A figura 2.18, como exemplo, representa o conhecimento relativo ao domínio pessoas: pais (genitores), filhos e etc. Esta estrutura também se refere a uma terminologia ou vocabulário que, na verdade, tem a intenção de representar a generalidade ou a especificidade dos conceitos envolvidos, por exemplo, a ligação entre os conceitos mãe e pais (genitores) é, em alguns casos, denominada relação *é_um* (NARDI e BRACHMAN, 2003).

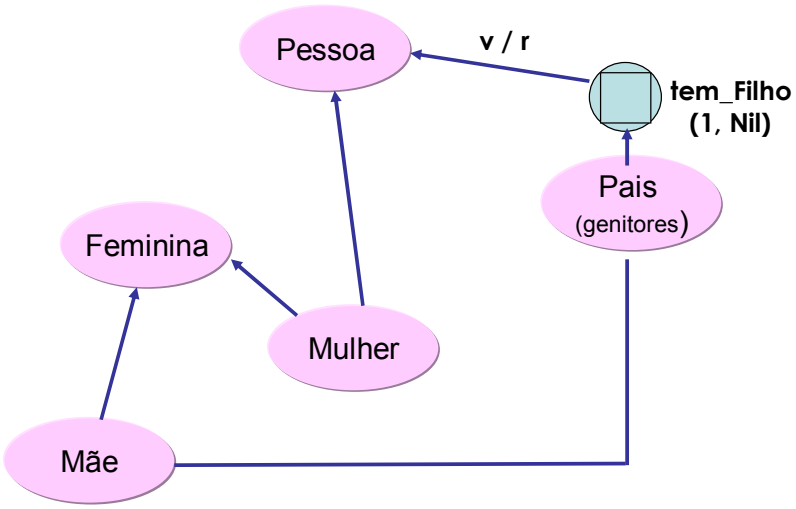


Figura 2.18 – Estrutura de representação baseada em rede (NARDI e BRACHMAN, 2003).

Neste sentido, as relações do tipo *é_um* definem uma hierarquia sobre os conceitos e fornecem a base para a "herança" das propriedades. Assim, quando um conceito é mais específico do que algum outro, ele "herda" as propriedades daquele mais geral, por exemplo, se o conceito "pessoa" tem o atributo "idade" então o conceito "mulher" também o tem. Isto representa um conjunto típico denominado de "herança monotônica" da rede (NARDI e BRACHMAN, 2003).

Adicionalmente, a lógica de descrições tem a capacidade de representar outros tipos de relação entre os conceitos, além do tipo *é_um*, da figura 2.18. Por exemplo, o conceito "pais" (genitor) tem uma propriedade que é, usualmente, conhecida como papel (*role*), expressa pela ligação a partir do conceito até o nó que representa o papel (denominado por *tem_filho*). Além de um "valor de restrição" (v/r), o qual expressa a limitação para a gama de objetos que podem desempenhar este papel e um "número de restrição" (1, *NIL*), representando o limite inferior e superior para a quantidade de filhos (*NIL* representa um número infinito) (NARDI e BRACHMAN, 2003).

É importante ressaltar que podem ser inferidas importantes relações implícitas entre os conceitos representados, como por exemplo: observa-se que o conceito "mulher" é definido, de forma explícita, a partir do conceito "pessoa" e do conceito "feminina", assim toda mulher é uma "pessoa" e do sexo "feminino", mas a relação de que toda "mãe" é uma "mulher", na verdade, representa uma relação implícita. Isto porque, neste caso, pode-se observar que ambos os conceitos "mãe" e "mulher" são conectados, explicitamente, aos conceitos "feminina" e "pessoa", respectivamente. Todavia, o caminho entre o conceito "mãe" para o conceito "pessoa" inclui o nó "pais" (genitor), o qual é mais específico do que o conceito "pessoa". Assim, permite-se concluir, por conseqüência lógica, que o conceito "mãe" é mais específico do que o conceito "pessoa" (NARDI e BRACHMAN, 2003).

Em essência, esta habilidade potencial de inferência ou raciocínio sobre o conhecimento explícito em situações muito mais complexas é o elemento chave para a aplicabilidade e sucesso dos sistemas baseados em conhecimento.

Inicialmente, os sistemas baseados em representações não lógicas foram considerados mais interessantes e efetivos do ponto de vista prático devido, em parte, à sua origem centrada na cognição humana. Contudo, os sistemas construídos com bases nestas representações, em geral, apresentavam resultados finais restritos em função das lacunas que apresentavam em termos de uma caracterização precisa na semântica do conhecimento (NARDI e BRACHMAN, 2003).

Neste sentido, já no final dos anos de 1970, surgiram importantes questões de pesquisa referentes a: como determinar uma semântica apropriada para representar estruturas de conhecimento? Em particular, para as redes semânticas e *frames*, de modo a obter ganhos tanto em termos de facilitar a representação do conhecimento quanto em termos de eficiência de raciocínio (NARDI e BRACHMAN, 2003).

Dentro desta perspectiva, em meados dos anos de 1980, as lógicas de descrições, que correspondem a uma denominação mais recente de uma família de formalismos de

representação de conhecimento, surgiram como uma alternativa para representar o conhecimento de um dado domínio.

Inicialmente, pela definição de conceitos relevantes do domínio (terminologia) e a partir destes conceitos para especificar propriedades de objetos e indivíduos existentes dentro deste domínio (descrição do mundo real), mas ao contrário de suas predecessoras, dispunha formalmente de uma semântica baseada na lógica de predicados (BAADER e NUTT, 2003).

A lógica de descrições compreende também dois níveis: o nível terminológico (*intensional*) e o nível factual (*extensional*). O nível terminológico envolve os conceitos atômicos (*atomic concepts*) e papéis atômicos (*atomic rules*) que expressam a relação binária entre estes conceitos para um dado domínio. E o nível factual (*extensional*) envolve as asserções (proposições que se apresentam como verdadeiras) e os indivíduos deste domínio.

Neste contexto, as asserções constituem o componente denominado "ABox" enquanto as primitivas ontológicas sobre as quais as asserções de "ABox" são construídas são denominadas de componente terminológico "TBox" (BAADER e NUTT, 2003).

Nesta ótica, o componente TBox compreende um conjunto de conceitos formado por axiomas ou outros conceitos definidos a partir destes usando-se um conjunto de papéis. Assim, um conceito representa uma entidade genérica de um dado domínio de aplicação valido para um dado um conjunto de indivíduos. Por sua vez, um indivíduo é uma entidade particular, uma instância de um conceito (BAADER e NUTT, 2003).

Adicionalmente, um conjunto de construtores (operadores lógicos) é usado para construir novos conceitos e papéis a partir dos axiomas, bem como conectar os papéis e suas restrições aos conceitos estabelecidos para o domínio (BAADER e NUTT, 2003).

Desta perspectiva, as tarefas fundamentais de inferência ou raciocínio no contexto da lógica de descrições são:

- Análise computacional da relação de subsunção entre conceitos, que consiste basicamente em verificar se um conceito particular está sob dependência de outro mais geral, de modo a estabelecer-se uma taxionomia mediante a comparação de suas definições;
- Classificação, isto é, a inserção automática de um conceito em uma hierarquia, conectando-o, apropriadamente, entre o mais específico e ao mais geral. De modo a verificar se uma instância (indivíduo) refere-se a um conceito dentro da hierarquia, e algumas linguagens incluem a verificação da consistência de uma definição de conceito por meio de critérios de pertinência (RUSSELL e NORVIG, 2004).

Na literatura são apresentadas e discutidas diferentes linguagens para a família das lógicas de descrições. Estas linguagens diferem entre si em função do conjunto de construtores lógicos que cada linguagem pode suportar. Neste sentido, a partir da linguagem de atributos ($\mathcal{AL}$ - *Atributive Language*) complementada ($\mathcal{C}$ - *Complement*) por outros construtores (como por exemplo, $\mathcal{U}$ disjunção, $\mathcal{N}$ negação, $\mathcal{E}$ quantificação existencial, $\mathcal{H}$ hierárquico, $\mathcal{I}$ papel inverso, $\mathcal{Q}$ qualificador de número de restrições), derivam-se as linguagens ($\mathcal{ALCN}$) e ($\mathcal{ALCNH}_{\mathcal{R}}$) entre outras (BAADER e NUTT, 2003).

O Sowa (1995) revela que "a lógica é a pura forma e ontologia é o conteúdo" e "sem a ontologia, a lógica não diz nada e, sem a lógica, a ontologia somente pode ser analisada, representada e discutida com base em generalidades vagas". Ressaltando ainda que a primeira etapa para o projeto de uma base de conhecimento pode ser a seleção de uma apropriada coleção de categorias ontológicas.

Cumpre observar que uma descrição das diferentes linguagens da família das lógicas de descrições está fora do alcance desta subseção e pode ser encontrada nas referências citadas no texto. Adicionalmente, ressalta-se que nos Capítulo 4 e 7 são apresentados outros aspectos da fundamentação teórica junto ao desenvolvimento da ontologia PFMEA e implementação da respectiva base de conhecimento.

2.6 SÍNTESE DO CAPÍTULO

Este capítulo apresentou a revisão da literatura relevante para esta pesquisa. Neste sentido, este capítulo foi dividido em cinco subseções principais: evolução e desafios conceituais à gestão da qualidade, desafios à qualidade em processos de moldagem por injeção de termoplásticos, 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.

A evolução e os desafios conceituais à gestão da qualidade, bem como à qualidade em processos de moldagem por injeção foram discutidas no contexto do compartilhamento de conhecimento, buscando-se ressaltar a importância e a estrutura deste compartilhamento para a análise é solução de problemas de não-conformidades em processos de manufatura envolvendo estruturas organizacionais emergentes.

Por sua vez, a tecnologia de agentes, técnicas de raciocínio baseado em casos e as ontologias foram discutidas considerando as suas potencialidades para o suporte efetivo do

compartilhamento de conhecimento decorrente dos processos de análise e solução dos problemas de não-conformidades.

CAPÍTULO 3
PROCEDIMENTOS METODOLÓGICOS PARA O DESENVOLVIMENTO DO MODELO

O propósito deste capítulo é apresentar a metodologia adotada para o desenvolvimento do modelo baseado em agentes proposto neste trabalho de tese. Esta metodologia compreende o desenvolvimento de um modelo conceitual, a especificação deste modelo em termos do projeto de uma organização multiagente abstrata e a implementação deste modelo. Adicionalmente, apresentam-se as questões relacionadas à verificação e validação do modelo.

Neste sentido, este capítulo é estruturado de forma a apresentar as abordagens metodológicas e científicas que fundamentam as etapas de desenvolvimento e processos de verificação e validação do modelo em consonância aos objetivos deste trabalho.

3.1 METODOLOGIA PARA DESENVOLVIMENTO DO MODELO

De acordo com Ferber et al. (2003), desde o seu aparecimento, no início dos anos 80, os sistemas multiagentes têm sido considerados como sociedades de agentes, isto é, como conjuntos de agentes que interagem mutuamente para coordenar o seu comportamento e que ainda, com freqüência, devem cooperar para atingir algum objetivo comum. Nesta concepção, fica claro que o campo de pesquisa em sistemas multiagentes está relacionado tanto aos agentes quanto a própria sociedade de agentes.

Entretanto, Ferber et al. (2003) revelam que na literatura da área percebe-se uma importante ênfase no lado do agente, pois os sistemas multiagentes têm sido particularmente estudados em um nível micro, isto é, no nível dos estados de um agente e na relação entre estes estados e seu comportamento geral. Nesta visão, as comunicações entre os agentes são tratadas como atos comunicativos (*speech acts*), cujo sentido pode ser descrito em termos de um estado mental do agente, onde o desenvolvimento de linguagens como a FIPA-ACL decorrem diretamente desta perspectiva.

Na literatura observa-se ainda o uso do termo “sistemas multiagentes” associado aos agentes quando se trata deste tipo clássico de sistema projetado em termos do estado mental dos agentes (GASSER, 2001 e FERBER et al., 2003).

Por outro lado, mais recentemente, especial interesse tem se verificado no tocante ao uso de conceitos organizacionais dentro de sistemas multiagentes, tais como papéis (*roles*) (ou funções), grupos (comunidades), tarefas (atividades) e protocolos de interação (WOOLDRIDGE et al., 1999 e 2000; GASSER, 2001; FERBER et al., 2003; ZAMBONELLI

et al., 2003). Neste contexto, os sistemas multiagentes projetados a partir destes conceitos recebem a qualificação geral de sistemas multiagentes centrados na organização.

Neste cenário, Ferber et al. (2003) argumentam, em especial, que é possível desenvolver um sistema multiagente unicamente a partir de conceitos organizacionais, entretanto isto não significa abandonar completamente a idéia de estados mentais dos agentes, mas enfatizar a possibilidade de construir organizações multiagentes como arcabouços onde agentes com diferentes habilidades reativas ou cognitivas podem interagir.

É importante ressaltar que na literatura existem diferentes definições para o termo "organização multiagente". Para Gasser (2001) uma organização multiagente fornece um arcabouço para a atividade e interação através da definição de papéis, expectativas comportamentais e relações de autoridade ou controle. Todavia, esta definição é bastante geral e não fornece uma noção clara de como projetar uma organização multiagente.

Por sua vez, Wooldridge et al. (2000) apresentam uma organização multiagente como uma coleção de papéis (*roles*), os quais se mantêm em certos relacionamentos de uns para com os outros e que participam de padrões sistemáticos e institucionalizados de interação com outros papéis.

Nesta ótica, Ferber et al. (2003) ressaltam ainda que as principais características de uma organização multiagente podem ser sintetizadas como: uma organização é constituída de agentes individuais que manifestam um comportamento; uma organização pode ser subdividida em partes que se sobrepõem; os comportamentos dos agentes são funcionalmente relacionados à atividade organizacional geral (conceito de papel); os agentes estão engajados em relacionamentos dinâmicos (também chamados de padrões de atividades, os quais podem ser tipificados usando a taxonomia de papéis, tarefas e protocolos definindo assim uma forma de supra-individualidade) e os tipos de comportamentos são relacionados por meio dos relacionamentos entre estes papéis, tarefas e protocolos.

Portanto, um elemento chave de uma organização é o conceito de papel (*role*). Um papel é uma descrição de um comportamento abstrato de um agente. Um papel descreve as restrições (obrigações, requisitos e habilidades) que um agente deverá atender para obter um papel, bem como as responsabilidades associadas ao papel. Um papel é também um espaço reservado para a descrição de um padrão de interação, ao qual um agente que desempenha aquele papel estará atrelado (FERBER et al., 2003).

Neste contexto, a coerência de uma organização multiagente deve ser assegurada considerando metodologias de desenvolvimento apropriadas para esta finalidade. Nesta perspectiva, Luck et al. (2004) revelam que os trabalhos sobre metodologias de

desenvolvimento de organizações multiagentes apresentados na literatura exploram, em geral, a sinergia a partir da interação das comunidades científicas da engenharia de software e da engenharia de conhecimento, e as principais áreas de interesse incluem: requisitos de engenharia para sistemas de agentes; modelos de análise conceitual e projeto de organizações multiagentes; ontologias específicas para agentes, modelos de agentes e modelos de organização; ferramentas para suporte do processo de desenvolvimento de sistemas multiagentes, tais como plataformas de agentes.

Nestes trabalhos, o propósito de uma metodologia é prescrever todos os elementos necessários para o desenvolvimento de uma organização multiagente e envolve dois importantes componentes: a descrição dos elementos do processo e o produto do trabalho e sua documentação (HENDERSON-SELLERS e GIORGINI, 2005).

Adicionalmente, no tocante à definição do termo "metodologia" neste contexto, Bussmann et al. (2004), revelam que uma metodologia consiste dos seguintes elementos: uma definição (opcional) do espaço de problemas para o qual a metodologia é aplicável; um conjunto de modelos que representam diferentes aspectos de um domínio de problemas ou a sua solução em diferentes estágios; um conjunto de métodos que transformam instâncias de um modelo em outro modelo com menor nível de abstração e um conjunto de diretrizes que definem uma ordem para a aplicação sistemática dos passos da metodologia.

Dentro desta linha, vários trabalhos podem ser encontrados na literatura sobre a modelagem e o projeto de organizações multiagentes, entre as quais: a Metodologia *MAS-CommonKADS* proposta por Iglesias et al. (1997), a Metodologia GAIA[1] formulada por Wooldridge et al. (1999), MASE (***M**ulti-**A**gent **S**ystem **E**ngineering*) apresentada por DeLoach et al. (2001), MESSAGE (***M**ethodology for **E**ngineering **S**ystems of **S**oftware of **AGE**nts*) estabelecida no âmbito do projeto EURESCOM project P907 (CAIRE e LEAL, 2001), PASSI (***P**rocess for **A**gent **S**ocieties **S**pecification and **I**mplementation*) formulada por Cossentino e Potts (2001), Metodologia Tropos proposta por Bresciani et al. (2004) e a Metodologia *Prometheus* apresentada por Padgham e Winikoff (2002).

Todavia, uma ampla revisão bibliográfica sobre metodologias para desenvolvimento de organizações multiagentes, bem como detalhados estudos comparativos estão além do escopo deste trabalho e podem ser encontrados nos trabalhos de Henderson-Sellers e Giorgini (2005), Bussmann et al. (2004) e Dam e Winikoff (2003) além dos citados no parágrafo anterior.

[1] Em alusão à mitologia grega na qual GAIA era a figura da mãe Terra e que, posteriormente, foi usada por James Lovelock no sentido de que todo organismo vivo na Terra pode ser percebido como um componente de uma entidade única que regula o ambiente da Terra (CAPRA, 1996).

No contexto deste trabalho de tese será adotada a Metodologia GAIA, formulada por Wooldridge et al. (1999, 2000) e, posteriormente, complementada por Zambonelli et al. (2003) e Cernuzzi e Zambonelli (2005). A metodologia GAIA é baseada na idéia central de conceber uma complexa organização multiagente a partir da metáfora de uma organização computacional, bem como pela abstração de papéis e responsabilidades e que, em essência, define o que um agente e uma organização devem realizar considerando um conjunto de requisitos funcionais.

É importante observar que a escolha da Metodologia GAIA levou em consideração a consistência científica, sua atualidade, significância, rigor e também os diversos trabalhos correlatos. Em especial, o trabalho de Moraïtis et al. (2003), que apresenta diretrizes para a implementação dos modelos de projeto gerados pela Metodologia GAIA por meio de arcabouços desenvolvidos em conformidade às especificações FIPA 2000 - *Foundation for Intelligent Physical Agents*, as quais são amplamente utilizadas tanto em aplicações de natureza acadêmica quanto industrial para a implementação da tecnologia de agentes.

É importante destacar também que os elementos centrais da Metodologia GAIA caracterizados pelos processos de análise conceitual e de projeto da organização multiagente estão em consonância com a idéia de desenvolvimento de um modelo conceitual e da especificação do modelo previstos neste trabalho. Além disso, em especial, são coerentes com o arcabouço de verificação e validação proposto por Yilmaz (2006), destinado a verificar e validar os processos sociais no âmbito dos modelos de organização computacional baseado em agentes.

Por fim, cumpre ressaltar que a Metodologia GAIA trata da concepção da sociedade de agentes tanto no nível do próprio agente quanto no nível da organização multiagente, sem no entanto assumir qualquer arquitetura em particular. A seguir são apresentadas em síntese as etapas de análise conceitual e de projeto adotadas para o desenvolvimento do modelo proposto neste trabalho de tese.

3.1.1 Metodologia GAIA - Análise conceitual

A primeira etapa da metodologia consiste da análise conceitual, destinada a modelar a organização em termos de um conjunto de papéis que serão desempenhados no contexto da organização multiagente (modelo de papéis) e identificar a forma de interação destes papéis (modelo de interação) representados na figura 3.1. Esta etapa buscar estabelecer um profundo entendimento da funcionalidade da organização e de sua estrutura, abstraindo-se dos detalhes de implementação dos agentes (ZAMBONELLI et al., 2003 e 2005).

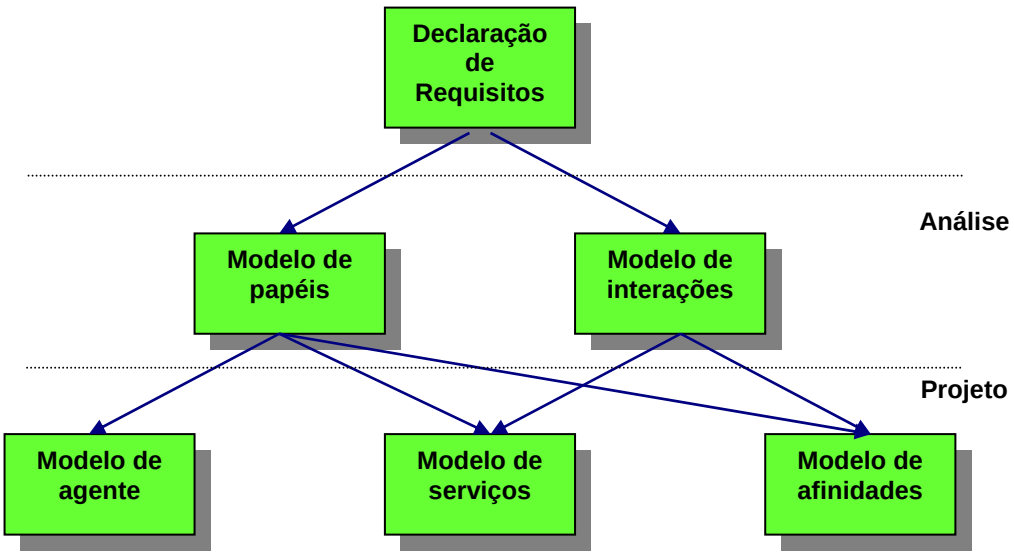


Figura 3.1 – Os modelos da metodologia GAIA (Fonte: WOOLDRIDGE et al., 2000).

Ao final da etapa de análise são produzidos os modelos de papéis que descrevem de forma clara e estruturada os detalhes de cada papel, bem como modelos de interações que estabelecem o fluxo de informações e envio de mensagens envolvidas.

3.1.2 Metodologia GAIA - Projeto do sistema

Após a etapa de análise, os conceitos abstratos e modelos de análise (modelos de papéis e de interações) devem ser traduzidos para um novo conjunto de modelos de projeto com níveis de abstração suficientemente baixos de modo a servir como especificações de projeto para a implementação dos agentes computacionais, a saber: modelos de agentes, modelos de serviços e modelos de afinidades. Estes modelos visam estabelecer como a sociedade de agentes coopera para atingir as metas no nível da organização multiagente e o que é requerido de cada agente individual.

O propósito do modelo de agentes é formalizar as classes (ou famílias) de agentes, as quais devem ser implementadas computacionalmente por meio dos processos de mapeamento, associação e atribuição de papéis a uma determinada classe de agente. Estes processos devem considerar a lógica funcional e, em geral, existirá uma correspondência direta de um para um entre os papéis e classes de agentes, embora possa existir mais de um papel relacionado a uma classe de agente.

O objetivo do modelo de serviços é identificar os serviços associados a cada papel (classe de agente) e especificar as principais propriedades do serviço, tais como entrada, saída, pré-condições e pós-condições do serviço. Um serviço pode ser entendido como uma função do

agente e que corresponde à atividade e protocolo de interação, previamente identificados na análise conceitual.

O modelo de afinidades é, provavelmente, o modelo mais simples e define os elos de comunicação que existem entre as classes de agente, isto é, quais agentes podem se comunicar diretamente com quais outros agentes. O modelo, porém, não define quais mensagens são enviadas ou quando estas são enviadas. Este modelo somente indica que um elo de comunicação existe, além de identificar gargalos de comunicação durante a execução e possíveis inconsistências na troca de mensagens.

Em síntese, a aplicação da metodologia GAIA permite conceber o modelo baseado em agentes em termos da metáfora de uma organização computacional e das abstrações de papéis organizacionais e responsabilidades (conjunto de metas) por meio dos modelos ilustrados, os quais, em essência, definem o que os agentes e a sociedade de agentes devem realizar em função de requisitos funcionais estabelecidos.

3.1.3 Considerações sobre a implementação dos modelos de projeto

A Metodologia GAIA concentra-se nas etapas de análise conceitual e projeto e, portanto, não cobre a etapa da implementação computacional. Entretanto, o trabalho de Moraïtis et al. (2003) apresenta um conjunto de premissas e procedimentos válidos para guiar a implementação das especificações dos modelos de projeto, as quais são resumidas a seguir:

- As expressões de execução serão traduzidas diretamente para os comportamentos dos agentes na plataforma de agentes. As atividades e protocolos serão traduzidos como métodos (os quais são partes dos comportamentos dos agentes). Neste sentido, podem ser usados os diversos comportamentos, que em geral são pré-programados nas plataformas.
- Definir todas as mensagens FIPA-ACL (*Agent Comunication Language*) usando os modelos de protocolo e interação;
- Observar a implementação de condições de segurança de cada papel no projeto do comportamento do agente, buscando validar as pré-condições do modelo de serviço;
- Projetar as necessidades de estrutura de dados e módulos de software que irão ser usados pelos agentes considerando os papéis e modelos de agentes;
- Definir os comportamentos dos agentes da organização iniciando com aqueles de menor nível, usando as várias classes de comportamentos disponíveis nas plataformas, a partir do modelo de papéis. Os comportamentos que são ativados com base no

recebimento de mensagens específicas devem ser adicionados aos comportamentos dos destinatários das mensagens.

- Ter em mente que os papéis traduzidos para comportamentos são trechos reutilizáveis de código.

Em termos metodológicos, a escolha dos recursos de *software*[2] representa um aspecto fundamental da implementação do modelo proposto e, neste sentido, o Capítulo 5 (Recursos de software para implementação do modelo proposto), será devotado à apresentação das questões relacionadas ao processo de escolha destes recursos.

Por sua vez, os Capítulos 6 (Implementação do modelo baseado em agentes proposto) e o Capítulo 7 (Implementação das bases de conhecimento dos agentes de recursos) serão destinados à apresentação é discussão da etapa de implementação do modelo em questão.

3.2 VERIFICAÇÃO E VALIDAÇÃO DE ORGANIZAÇÕES MULTIAGENTE

As pesquisas desenvolvidas atualmente no contexto da avaliação de modelos baseados em agentes são marcadamente influenciadas pela percepção do conceito de validação de modelos no âmbito da pesquisa operacional clássica como revelam Yilmaz (2006) e Windrum et al. (2007). Deste modo, visando um melhor entendimento e a contextualização da abordagem para a validação adotada neste trabalho, torna-se importante resgatar os aspectos fundamentais relacionados ao conceito de validação.

3.2.1 Considerações sobre o conceito de validação na Pesquisa Operacional

Na literatura relevante, os trabalhos relacionados à validação, em geral, podem ser classificados considerando-se duas perspectivas distintas: na primeira o conceito de validação é tratado a partir da visão da filosofia da ciência, enquanto na outra perspectiva considera-se uma orientação operacional e mais concreta.

Dentro da perspectiva filosófica, a construção de modelos é considerada, em essência, uma atividade de produção de conhecimento e, portanto, constitui-se em um tema a ser tratado em bases epistemológicas, como apontam Landry e Oral (1993).

Nesta linha, Déry et al. (1993) apresentam os modelos e a validação a partir de uma leitura epistemológica e contemporânea, considerando-se a visão particular das principais correntes

[2] Recursos de *software* compreendem os pacotes de software, bem como as licenças.

de pensamento na filosofia da ciência no tocante ao conhecimento científico. Em especial, eles discutem o impacto dos modelos e da validação sobre o campo da pesquisa operacional clássica.

Déry et al. (1993) assumem a premissa de que a resposta para a questão fundamental “O que é um modelo válido?” não pode ser dissociada da visão de ciência e de conhecimento científico de cada corrente e revelam, em primeiro lugar, que não existe um método científico universal ou mesmo um conjunto universal de critérios de validação que possa garantir a cientificidade de modelos produzidos pela pesquisa operacional.

Em segundo lugar, estabelecem que a construção e a validação de modelos não envolvem somente atividades cognitivas, mas também atividades sociais e, portanto, os modelos não podem ser julgados somente com respeito a um conjunto de critérios de natureza cognitiva.

Em terceiro lugar, afirmam que é fundamental reconhecer a influência da organização social das atividades de pesquisa na escolha de critérios de avaliação que determinam a validade de um modelo.

Nesta mesma linha, Mackenzie et al. (2002) observam que o conceito de validação pode ser interpretado de diferentes maneiras dependendo do contexto e dos tipos de problemas em questão. Eles sustentam também que os avanços nos paradigmas de modelagem, metodologias e tecnologias são em grande medida influenciados pela própria mudança na natureza dos problemas a serem tratados, tendo um considerável impacto no modo pelo qual o conceito de validação é percebido pela comunidade científica.

Por outro lado, para uma perspectiva operacional e mais concreta, Miser (1993) define o termo “validação” como “o processo pelo qual cientistas asseguram a si mesmos e aos outros que uma teoria ou modelo é uma descrição de um fenômeno determinado, sendo adequado ao uso para o qual será aplicado”.

Entretanto, cumpre ressaltar que o termo “verificação” é usado também nesse sentido, embora o próprio Miser (1993) apresente uma definição mais restrita como sendo “o processo pelo qual cientistas asseguram a si mesmos e aos outros que a teoria ou o modelo construído é, na verdade, aquele que eles pretendiam construir”. Nesse caso, a verificação pode ser considerada, então, como uma parte do processo de validação.

Nesse contexto, Landry et al. (1983) já haviam contribuído de maneira significativa para o entendimento desta questão, argumentando que a validação não é uma fase separada e independente do processo de construção do modelo, mas é interligada e contínua ao longo de todo o ciclo de desenvolvimento, propondo atrelar as atividades de validação ao processo de construção do modelo estabelecendo o conceito de “processo de modelagem e validação”.

Nessa linha, Smith (1993) reconhece que os modelos são usados como apoio à compreensão de problemas complexos. E o modelo pode possuir simplificações que buscam organizar e reduzir a complexidade de modo a propiciar às pessoas um melhor discernimento sobre os elementos do problema. Não obstante a sua utilidade, frequentemente tais modelos impõem significativas dificuldades para a sua validação via abordagens tradicionais tendo em vista a sua complexidade técnica e conflitos sobre a sua relevância e rigor.

Todavia, uma detalhada discussão sobre os aspectos epistemológicos, bem como sobre orientações mais operacionais e concretas em relação ao conceito de validação está além do escopo deste trabalho, porém orientações podem ser encontradas nos trabalhos de Miser (1993), Roy (1993) e Pala et al. (1999), além dos trabalhos de Déry et al. (1993), Mackenzie et al. (2002), Oral e Kettani (1993), Gass (1993) e Smith (1993).

3.2.2 Considerações sobre a verificação e validação em organizações multiagentes

Neste cenário, Yilmaz (2006) sugere uma perspectiva orientada ao processo para a validação e verificação (V&V) de modelos de organização computacional multiagente. Dentro desta visão, os modelos baseados em agentes são caracterizados por padrões de interação sociais geralmente dinâmicos e complexos entre agentes autônomos que representam entidades do mundo real. Neste sentido, os agentes de *software* são considerados como entidades que funcionam de forma independente em um ambiente particular, frequentemente coexistindo com outros agentes e processos (SHOHAN, 1994).

Yilmaz (2006) propõe um arcabouço (ou *framework*) para a V&V, cujo escopo inclui: o modelo conceitual, a especificação de projeto do modelo e a programação computacional do modelo. As propriedades primárias de interesse para a V&V da organização de agentes são: a estrutura social, processos sociais e o comportamento colaborativo em nível macro-social que ocorre como resultado da interação local entre agentes.

A figura 3.2 ilustra os componentes do arcabouço de V&V proposto por Yilmaz (2006), onde o modelo conceitual, a especificação de projeto do modelo e a programação computacional do modelo são descritas pelos nós representados no grafo. As setas em linhas cheias relacionam os modelos (i.e. modelo conceitual, especificação do modelo e a implementação do modelo, respectivamente) e as setas em linhas pontilhadas indicam as fases de V&V.

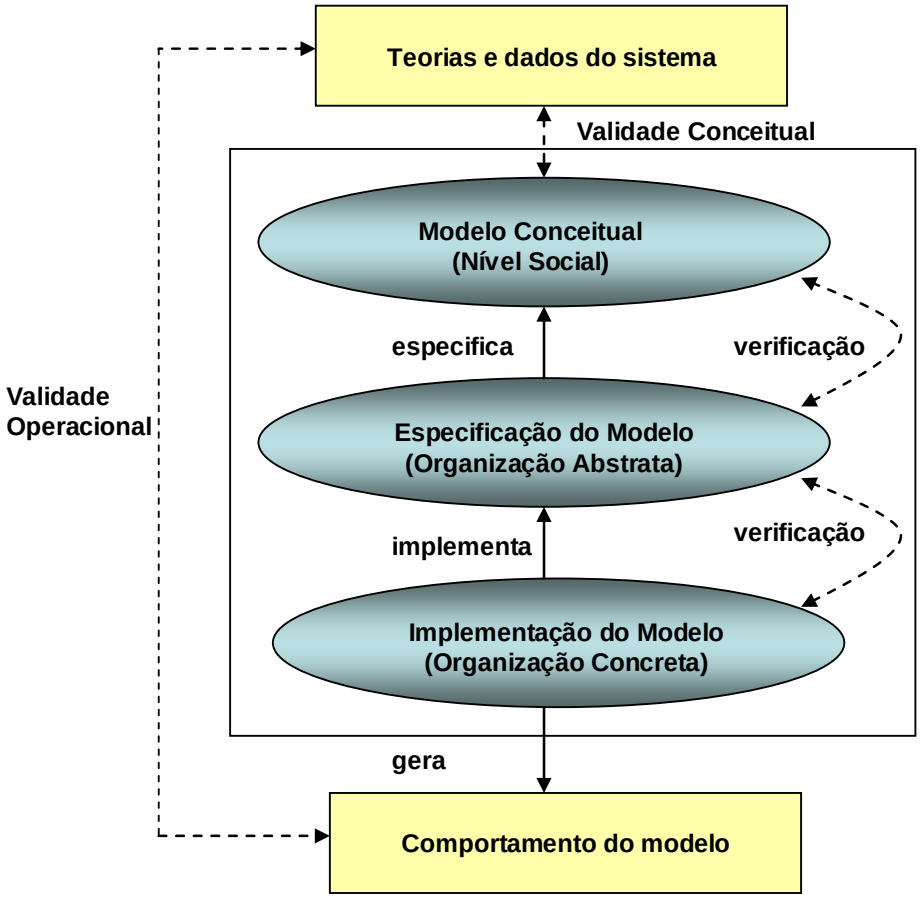


Figura 3.2 – Verificação e Validação de Organizações Multiagentes (Fonte: YILMAZ, 2006).

Neste sentido, Yilmaz (2006) parte da definição apresentada por Gasser et al. (1993), na qual um modelo conceitual de uma organização multiagente (do inglês *Multi Agent Organization*) é constituído pela representação das redes de interações sociais dos agentes e pelas bases de conhecimento específicas do domínio de cada agente.

Por sua vez, esta organização multiagente representada em uma forma abstrata apresenta os processos sociais e restrições em termos de uma especificação de projeto da organização. E, por fim, esta organização multiagente torna-se concreta a partir da implementação ou programação computacional do modelo que implementa de modo formal as especificações do projeto do modelo sendo a programação computacional responsável pelo comportamento do modelo.

Com relação à validade conceitual, Yilmaz (2006) argumenta que uma variedade de métodos pode ser aplicada para avaliar a fidedignidade de um modelo conceitual como uma entidade completa em relação ao fenômeno social sob estudo. E propõe o uso de casos de teste previamente preparados por especialistas no domínio, os quais permitem avaliar a

extensão na qual o conhecimento social representado no modelo conceitual reflete os construtos[3] elaborados pelos especialistas humanos.

Dentro desta visão é importante ressaltar que o modelo conceitual é um modelo qualitativo que busca capturar a essência da organização multiagente e, portanto, especial atenção deve ser dada aos procedimentos metodológicos usados na modelagem conceitual, bem como no tocante à modelagem das bases de conhecimentos.

Por outro lado, a verificação é um componente essencial do processo de avaliação da credibilidade geral, uma vez que ela é aplicada durante as especificações do projeto do modelo e sua respectiva implementação, de modo a assegurar que o modelo seja construído corretamente. Neste sentido, a abordagem de verificação deve enfatizar os processos sociais dos agentes e sua consistência no contexto da organização multiagente (YILMAZ, 2006).

Por sua vez, para Yilmaz (2006) a validade operacional diz respeito ao comportamento coletivo da sociedade de agentes no âmbito do modelo organizacional. E sua avaliação não busca concluir se comportamento coletivo dos agentes é válido ou inválido, mas busca determinar a acuracidade e a relevância do comportamento colaborativo resultante da interação entre agentes do modelo que foi implementada computacionalmente considerando o propósito do modelo.

3.2.3 Abordagem para verificação e validação adotada para o modelo proposto

A abordagem para a verificação e validação (V&V) adotada neste trabalho de tese segue a linha geral proposta por Yilmaz (2006).

Em primeiro lugar, busca-se demonstrar a viabilidade do modelo baseado em agentes proposto no trabalho de tese por meio da implementação computacional deste modelo a partir da especificação de projeto referente ao modelo conceitual. O autor considera que somente após esta implementação é possível realizar a verificação e estabelecer a sua validade conceitual e operacional, levando-se em conta os métodos e tecnologias necessárias para a implementação da solução.

No âmbito deste trabalho de tese será desenvolvido um protótipo de laboratório que segundo Waterman (1986) e Lee e O'Keefe (1994), tem por objetivo demonstrar a adequação da proposta, bem como a disponibilidade de metodologias de desenvolvimento. Através deste protótipo busca-se essencialmente demonstrar que as tecnologias envolvidas no

[3] Do latin *constructus* é o conjunto de habilidades e/ou conhecimentos que podem ser plausivelmente argumentados e/ou teoricamente justificados como esperados.

desenvolvimento são aplicáveis e viáveis ao domínio pretendido, considerando um conjunto de casos de teste que representam uma amostragem representativa do domínio em questão.

No protótipo de laboratório que foi desenvolvido os diversos agentes propostos foram implementados usando-se diferentes ferramentas de apoio, sendo que as mais importantes são: a ferramenta de desenvolvimento de agentes (ou a Plataforma de Desenvolvimento de Agentes); a ferramenta usada para formalizar e implementar as bases de conhecimento; o sistema responsável pelos serviços de inferência e recuperação de conhecimento necessários ao processamento destas bases de conhecimento.

Em especial, destaca-se no modelo implementado os elementos chave da organização, que são os próprios agentes, a ontologia, a estrutura conceitual da base de casos, protocolos de negociação e coordenação dos agentes, definição da sintaxe das mensagens a serem trocadas de modo que os agentes possam decodificar e analisar as intenções de comunicação (*communicative intentions*).

A verificação do modelo proposto consiste em avaliar a viabilidade e a credibilidade geral do modelo e que, de acordo com Yilmaz (2006), deve concentrar-se nas especificações do projeto do modelo e seu respectivo desenvolvimento computacional. Deste modo, a verificação tem por escopo assegurar que o modelo tenha sido corretamente implementado do ponto de vista computacional a partir das especificações de projeto, enfatizando os processos sociais dos agentes e a consistência da organização multiagente.

Em seguida, busca-se estabelecer a validade conceitual como proposto por Yilmaz (2006), isto é, usando-se casos de teste (ou *test cases*) previamente preparados por especialistas no domínio, os quais permitem avaliar a extensão na qual o conhecimento social representado no modelo conceitual reflete os construtos elaborados pelos especialistas humanos.

Nesta linha, as bases de conhecimento dos agentes instanciados no protótipo de laboratório em questão, devem ser preenchidas com um corpo de conhecimentos válidos encontrados na literatura da área e preparados por especialistas. De modo a avaliar a fidedignidade das respostas dos serviços de inferência e dos métodos de recuperação de conhecimento dos agentes de acordo com a especificação de projeto e referentes ao modelo conceitual em relação aos casos de testes fornecidos.

Por fim, busca-se estabelecer a validade operacional, como proposta por Yilmaz (2006), estabelecendo-se a fidedignidade e a relevância do comportamento colaborativo do modelo decorrente da implementação computacional com respeito ao propósito do modelo. Neste sentido, as bases de casos dos agentes instanciados no protótipo de laboratório devem ser preenchidas com casos obtidos em um trabalho de pesquisa de campo de uma aplicação

industrial no domínio, buscando-se identificar as potencialidades e as oportunidades de melhoria do modelo proposto em relação ao suporte à solução de problemas de não-conformidades no domínio.

CAPÍTULO 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 proposta 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. (1999, 2000) e complementada por Zambonelli et al. (2003) e Cernuzzi e Zambonelli (2005). Adicionalmente, este capítulo descreve o desenvolvimento das bases de conhecimento dos agentes propostos.

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 encoraja 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 e assim por diante. Por conseqüência, 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 desempenhar 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 como 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 seqüência 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 de 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 denominados 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):

1) 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.

2) 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.

3) Usando 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[1] (KBSI, 2007) da figura 4.2.

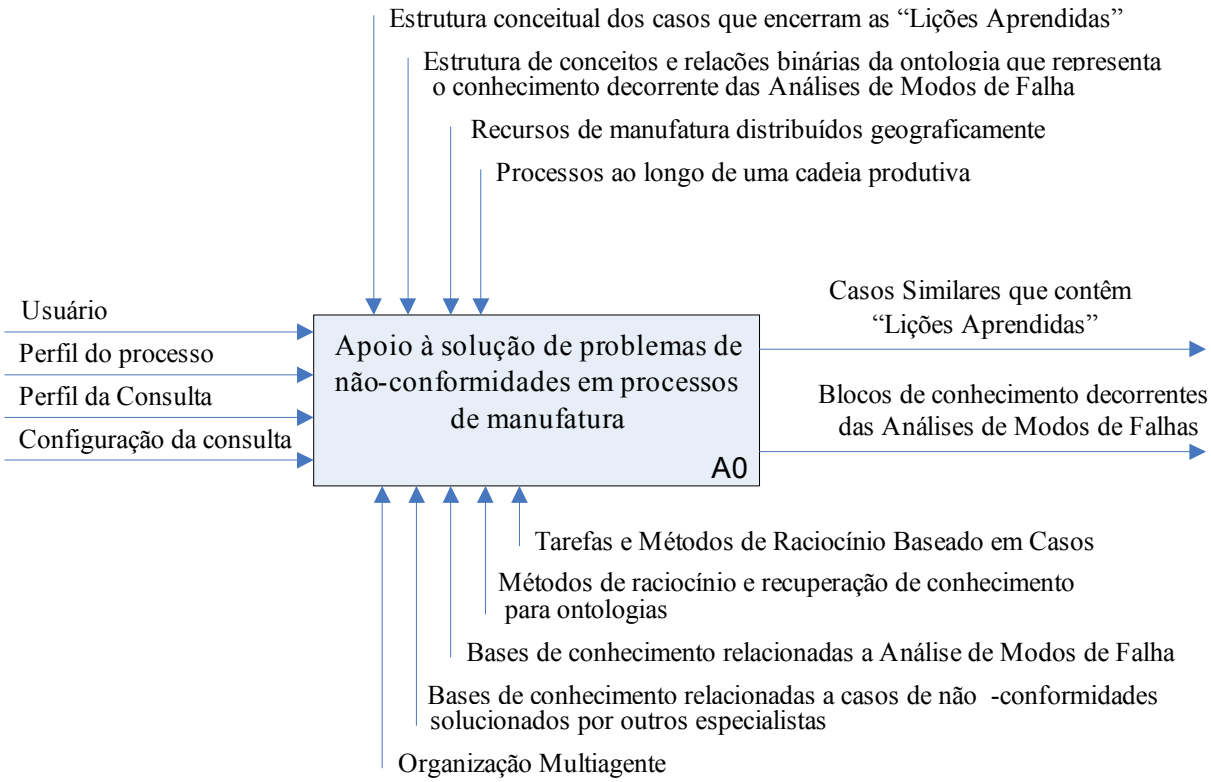


Figura 4.2 – Diagrama IDEF0 – Nível A0 para a função de apoio proposta

De acordo com a terminologia adotada no conjunto de padrões IDEF0 (*Integrated Definition Method for Function Modelling*) (figura 4.2) a caixa central descreve a função principal representada aqui como o apoio à solução de problemas de não-conformidades em processos de manufatura em seu nível mais alto (nível A-0). As interações desta com outras funções ou atividades são representadas por meio das entradas, controles, saídas e mecanismos (ICOM - *Input Control Output Mechanism*), os quais não incluem apenas dados ou informações, mas tudo quanto possa ser usado para descrever estas interações.

Neste sentido, as entradas são representadas pelo perfil do processo de manufatura em questão, pelo perfil da consulta e pela própria configuração da consulta *per se*. Em especial,

[1] 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).

considera-se que o apoio ao usuário pode envolver duas perspectivas distintas. A primeira diz respeito a um cenário de recuperação do conhecimento referente às lições aprendidas pelos especialistas a partir da solução de casos de não-conformidades similares ocorridas no passado, mas que podem ocorrer novamente no presente. E a segunda é relacionada à recuperação de conhecimentos produzidos a partir da aplicação do método preventivo (PFMEA) no tocante à investigação de causas e efeitos associados a todos os modos de falha que não ocorreu, mas que, de acordo com os especialistas, pode potencialmente ocorrer.

Por outro lado, a função de apoio é levada a efeito por meio de um conjunto de mecanismos, o qual compreende os recursos e meios necessários para a realização da função e que aparecem, no diagrama, conectados por setas à parte inferior da caixa. Neste sentido, os mecanismos envolvem a própria estrutura da organização multiagente, as bases de conhecimento e os métodos de raciocínio que podem ser aplicados a estas bases.

Os controles estabelecem as condições ou as restrições que são impostas à função e são representadas no diagrama pelas setas ligadas à parte superior da caixa. No presente modelo, as restrições envolvem a própria estrutura conceitual do caso de não-conformidades e à natureza dos conceitos e relações binárias usadas para representar o conhecimento no domínio em questão.

As saídas esperadas da função de apoio, representadas pelas setas à direita da caixa são caracterizadas pelos casos similares englobam as lições aprendidas e os blocos de conhecimento decorrentes das análises de modos de falha, os quais devem ser apresentados de forma adequada ao usuário, de modo a aumentar a compreensão sobre o problema em si e, por conseqüência, refinar o seu próprio processo de solução.

Por fim, a organização multiagente deve ser concebida para ser flexível o suficiente para permitir que agentes existentes possam deixá-la e que novos agentes possam ser adicionados, sem que, no entanto, o funcionamento e a autonomia da organização sejam afetados considerando a natureza da dinâmica empresarial. Além disso, busca-se o compartilhamento e o reuso do conhecimento, bem como o aprendizado sustentável dos agentes dotados de recursos de conhecimento.

Nas próximas subseções apresenta-se a transformação da definição dos requisitos no modelo conceitual e nas especificações de projeto respectivamente, de acordo com as etapas da metodologia GAIA.

4.1.2 Modelo de papéis

De acordo com a definição dos requisitos, o modelo proposto da organização multiagente, conceitualmente, pode ser modelado considerando cinco papéis distintos: o usuário, o assistente do usuário, o raciocinador e recuperador de casos similares de solução de problemas de não-conformidades e o raciocinador e recuperador de análises de modo de falha e o identificador de raciocinadores.

Neste contexto, o usuário pode ser um membro de uma equipe de melhoria da qualidade ou da engenharia de fabricação entre outros, que busca apoio à solução de um problema de não-conformidade para um dado processo de manufatura dentro de uma cadeia de produção com recursos de manufatura distribuídos.

Este usuário busca não somente conhecer as soluções já adotadas por outros especialistas em casos similares, mas também as lições aprendidas com o caso. Ou ainda conhecer os resultados decorrentes da aplicação do método preventivo (PFMEA). Neste sentido, o modelo do papel do usuário é apresentado nas figuras 4.3 e 4.4, e de acordo com a metodologia GAIA, este modelo envolve: o esquema do papel, a descrição, protocolos, atividades e as propriedades de execução e de segurança.

ESQUEMA DO PAPEL: **USUÁRIO**

DESCRIÇÃO: o usuário inicia as ações na organização multiagente pela formulação de consultas baseadas em atributos que descrevem a não-conformidade em um dado processo de manufatura.

PROTOCOLOS: *DefinirProcessoDeManufatura;*
DefinirPerspectivaDaConsulta;
ConfigurarConsultaDeAcordoComEstruturaDoCaso.

ATIVIDADES:

PERMISSÕES:

gerar *PerfilDoProcessoDeManufatura;*
PerspectivaDaConsulta;
ConfiguraçãoDaConsultaDeAcordoComEstruturaDoCaso;
ConfiguraçãoDaConsultadeAcordoComConceitosOntológicos.

ler **informado** *SoluçoesSugeridasPelosCasosRecuperados;*
informado *ResultadosReportadosPelosCasosRecuperados;*
informado *MedidasDeSimilaridadeCalculadas;*
informado *ConceitosInstâciasRecuperadasDaBaseOntológica;*

Figura 4.3 – Modelo de papéis – Usuário (Parte I).

PROPRIEDADES: Execução
USUÁRIO = CriarPerfilDaConsulta . (ConfigurarConsulta*).*
CriarPerfilDaConsulta =*DefinirProcessoDeManufatura . DefinirPerspectivaDaConsulta*
ConfigurarConsulta = *ConfiguraçãoDaConsultaDeAcordoComEstruturaDoCaso*
ConfigurarConsulta = *ConfiguraçãoDaConsultadeAcordoComConceitosOntológicos*

PROPRIEDADES: Segurança : Ativo

Figura 4.4 – Modelo do papéis – Usuário (Parte II).

Por sua vez, o modelo do papel do assistente do usuário é apresentado nas figuras 4.5 e 4.6, a qual ilustra como o assistente age em nome do usuário no contexto da organização multiagente durante a comunicação com outros papéis. Em especial, o modelo mostra a comunicação mediante a troca de mensagens com o raciocinador/recuperador de casos similares de solução de problemas de não-conformidades e de análises de modo de falha e o identificador de raciocinadores.

Neste contexto, o assistente do usuário deve guiar o usuário durante a configuração da consulta, particularmente quanto aos atributos que descrevem a não-conformidade e seu respectivo contexto. Adicionalmente, o assistente do usuário deve transcrever as consultas formuladas pelo usuário para uma sintaxe apropriada para, então, direcioná-las aos raciocinadores capazes de respondê-la mediante a respectiva troca de mensagens. Além disso, o assistente transcreve o conteúdo da mensagem recebida, de modo a apresentá-la ao usuário de forma inteligível.

ESQUEMA DO PAPEL: ASSISTENTE DO USUÁRIO

DESCRIÇÃO: Age em nome do usuário no contexto da organização e é responsável pela interface com o usuário.

PROTOCOLOS: *GuiarAConfiguraçãoDaConsulta;*
ObterAConsultaConfigurada;
EncapsularAConsultaConvertidaNoConteúdoDaMensagem;
LocalizarRaciocinadoresAtivosJuntoAoIdentificador;
EnviarMensagemAosRaciocinadoresAtivos;
AguardarRespostaDaConsulta;
ApresentarRespostaConvertidaAoUsuário.

ATIVIDADES: *ConverterAConsultaParaSintaxePrópria;*
ConverterARespostaRecebida

Figura 4.5 – Modelo de papéis – Assistente do Usuário (Parte I).

PERMISSÕES:
ler **informado** *PerfilDoProcessoDeManufatura;*
informado *PerspectivaDaConsulta;*
informado *ConfiguraçãoDaConsultaDeAcordoComEstruturaDoCaso;*
informado *ConfiguraçãoDaConsultadeAcordoComConceitosOntológicos.*

gera *ConsultaConvertida;*
RespostaConvertida.

PROPRIEDADES: Execução
ASSISTENTE = CriarConsulta . (SubmeterAConsulta . ReceberRespostaDaConsulta)
ASSISTENTE = CriarResposta

CriarConsulta = *GuiarAConfiguraçãoDaConsulta . ObterAConsultaConfigurada*
SubmeterAConsulta = *ConverterAConsultaParaASintaxePrópria .*
EncapsularAConsultaConvertidaNoConteúdoDaMensagem .
LocalizarRaciocinadoresAtivosJuntoAoIdentificador .
EnviarMensagemAosRaciocinadoresAtivos
ReceberRespostaDaConsulta = *AguardarRespostaDaConsulta;*
CriarResposta = *ConverterARespostaRecebida. ApresentarRespostaConvertidaAoUsuário*

PROPRIEDADES: Segurança : Ativo

Figura 4.6 – Modelo de papéis – Assistente do Usuário (Parte II).

Por sua vez, o raciocinador/recuperador deve aguardar continuamente por mensagens apropriadas enviadas pelo assistente do usuário, e ao recebê-la ele deve executar a decodificação de seu conteúdo e iniciar os métodos de raciocínio e recuperação destinados ao acesso à base de conhecimento. Em seguida, ele deve codificar os resultados obtidos como conteúdo de uma mensagem e, então, enviá-la ao assistente do usuário envolvido na troca de mensagens, como ilustra o modelo do papel da figura 4.7.

ESQUEMA DO PAPEL: RACIOCINADOR/RECUPERADOR

DESCRIÇÃO: Responsável pela execução dos métodos de inferência e recuperação sobre as bases de conhecimentos.

Figura 4.7 – Modelo do papel – Raciocinador/recuperador (Parte I).

PROTOCOLOS: *AguardarContinuamenteMensagensDoAssistente;*
ObterAConsultaEncapsuladaNoConteúdoDaMensagem;
EncapsularARespostaConvertidaNoConteúdoDaMensagem;
EnviarAMensagemAoAssistente..

ATIVIDADES: *RealizarRegistroJuntoAoIdentificador*
ConverterAMensagemRecebidaParaASintaxePrópria;
IniciarOsMétodosDoRaciocínio;
AcessarBaseDeConhecimento;
ConverterAsRespostaDoRaciocínioParaASintaxeInicial;

PERMISSÕES:
ler **informado** *AConsultaEncapsuladaNoConteúdoDaMensagem;*

gera *MensagemRecebidaParaASintaxePrópria;*
RespostaDoRaciocínioParaASintaxeInicial;
altera *BaseDeConhecimento;*

PROPRIEDADES: Execução
RACIOCINADOR=RecebeAConsulta . (ExecutaMétodosDeRaciocínio .
EnviaAResposta);

RecebeAConsulta = *AguardarContinuamenteMensagensDoAssistente .*
ObterAConsultaEncapsuladaNoConteúdoDaMensagem
ConverterAMensagemRecebidaParaASintaxePrópria
ExecutaMétodosDeRaciocínio = *ConverterAMensagemRecebidaParaASintaxePrópria .*
IniciarOsMétodosDoRaciocínio
EnviaAResposta = *ConverterAsRespostaDoRaciocínioParaASintaxeInicial .*

GuiarAConfiguraçãoDaConsulta . ObterAConsultaConfigurada .
EncapsularARespostaConvertidaNoConteúdoDaMensagem .
EnviarAMensagemAoAssistente

PROPRIEDADES: Segurança
ComunicaçãoAtiva = True
AssistentesAtivos = True
IdentificadoresAtivos = True

Figura 4.8 – Modelo do papel – Raciocinador/recuperador (Parte II).

O identificador de raciocinadores deve ser capaz de manter um registro da localização dos raciocinadores ativos na organização e de suas capacidades, de modo a permitir ao assistente do usuário comunicar-se, como mostra o modelo da figura 4.9.

ESQUEMA DO PAPEL: IDENTIFICADOR DE RACIOCINADORES

DESCRIÇÃO: Responsável por identificar todos os raciocinadores ativos capazes de executar o método de raciocínio desejado em função da perspectiva da consulta escolhida pelo usuário.

PROTOCOLOS: *AguardarSolicitaçãoDeIdentificãoDeRaciocinadores;*
ComunicarLocalizaçãoDosRaciocinadoresAoAssistente;

ATIVIDADES: *ManterRegistroDosRaciocinadoresAtivos;*
ConsultarRegistros;

PERMISSÕES:
ler **informado** *SolicitaçãoDeLocalizaçãoDeRaciocinadores;*

gera *LocalizaçãoECapacidadesDosRaciocinadoresAtivos;*
Altera *RegistroDeRaciocinadores;*

PROPRIEDADES: Execução
IDENTIFICADOR=RecebeSolicitação . InformaLocalizaçãoAoAssistente

RecebeSolicitação = *AguardarSolicitaçãoDeIdentificãoDeRaciocinadores*

AConsulta = *AguardarContinuamenteMensagensDoAssistente* .
ObterAConsultaEncapsuladaNoConteúdoDaMensagem
ConverterAMensagemRecebidaParaASintaxePrópria
ExecutaMétodosDeRaciocínio = *ConverterAMensagemRecebidaParaASintaxePrópria* .
IniciarOsMétodosDoRaciocínio
EnviaAResposta = *ConverterARespostaDoRaciocínioParaASintaxeInicial* .

GuiarAConfiguraçãoDaConsulta . *ObterAConsultaConfigurada* .
EncapsularARespostaConvertidaNoConteúdoDaMensagem .
EnviarAMensagemAoAssistente

PROPRIEDADES: Segurança
ComunicaçãoAtiva = True
AssistentesAtivos = True
RaciocinadoresAtivos = True

Figura 4.9 – Modelo do papel – Identificador de raciocinadores.

Neste cenário, é importante destacar que os raciocinadores/recuperadores devem ser dotados de bases de conhecimento apropriadas à natureza do conhecimento do domínio envolvido e ao tipo de serviço de raciocínio necessário, as quais são apresentadas na subseção 4.3.

4.1.3 Modelo de interações

Os modelos de interação apresentados na figura 4.10 e 4.11 expressam o propósito da interação entre os papéis do assistente do usuário e o identificador de raciocinadores.

PROPÓSITO: *LocalizarRaciocinadoresAtivosJuntoAoIdentificador*		
INICIADOR: ASSISTENTE	**RESPONDEDOR:** IDENTIFICADOR	**Entrada:** *PerfilDoProcessoDeManufatura PerspectivaDaConsulta;*
PROCESSAMENTO: Localizar os raciocinadores ativos capazes de responder a consulta desejada.		**Saída:** *SolicitaçãoDeLocalizaçãoE_ CapacidadesDos_ RaciocinadoresAtivos*

Figura 4.10 – Modelo de interações Usuário/Assistente visando localizar raciocinadores.

PROPÓSITO: *ComunicarLocalizaçãoDosRaciocinadoresAoAssistente*		
INICIADOR: IDENTITICADOR	**RESPONDEDOR:** ASSISTENTE	**Entrada:** *SolicitaçãoDeLocalizaçãoE_Ca pacidadesDosRaciocinadores_A tivos*
PROCESSAMENTO: Comunicar ao assistente a localização e capacidade dos raciocinadores ativos.		**Saída:** *LocalizaçãoDeRaciocinadores_ Ativos*

Figura 4.11 – Modelo de interações Assistente/Usuário visando comunicar a localização de raciocinadores.

As interações apresentadas nas figuras 4.10 e 4.11 envolvem seis atributos: propósito, iniciador, respondedor, entradas, saídas e processamento. O *propósito* destas interações é identificar todos os raciocinadores/recuperadores que se encontram ativos na organização a partir dos registros mantidos pelo identificador de raciocinadores, de modo que o assistente do usuário conheça a localização e os serviços disponíveis dos raciocinadores/recuperadores ativos na organização a partir do perfil do processo e do perfil da consulta requerida pelo usuário.

Assim, o assistente do usuário poderá comunicar-se com estes raciocinadores mediante uma troca de mensagens diretas como mostra a interação entre o assistente e os raciocinadores da figura 4.12.

PROPÓSITO: *EnviarMensagemAosRaciocinadoresAtivos*		
INICIADOR: ASSISTENTE	**RESPONDEDOR:** RACIOCINADORES	**Entrada:** *ConsultaConvertida*
PROCESSAMENTO: Enviar uma mensagem aos raciocinadores localizados pelo identificador.		**Saída:** *MensagemComConsulta_ EncapsuldaEnviada*

Figura 4.12 – Modelo de interações Assistente/Raciocinadores visando enviar mensagens.

É importante observar que, de forma a manter a clareza do texto da tese os demais modelos de interação são apresentados no apêndice 1.

Na próxima subseção apresenta-se o desenvolvimento das especificações de projeto a partir da transformação do modelo conceitual, de acordo com as etapas da metodologia GAIA.

4.2 DESENVOLVIMENTO DAS ESPECIFICAÇÕES DE PROJETO

O objetivo clássico de um processo de projeto é transformar os modelos abstratos resultantes do processo de análise conceitual em modelos com um nível de abstração suficientemente mais baixo, de modo a permitir a sua implementação e sua posterior verificação e validação. Neste sentido, na metodologia GAIA o processo de projeto envolve a geração de três modelos como mostra a figura 4.13.

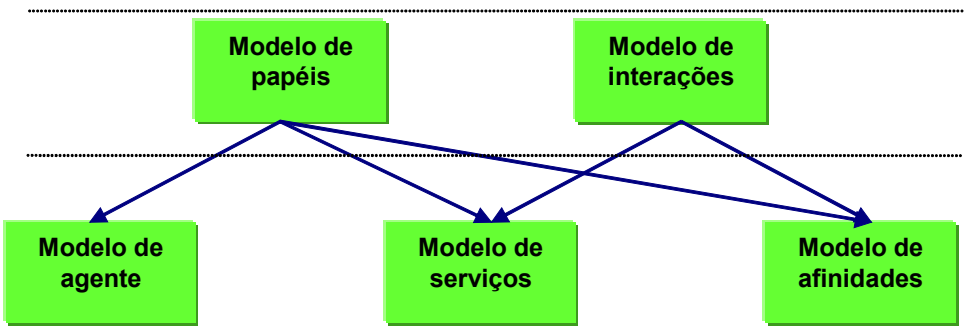


Figura 4.13 – Relação entre modelos da metodologia (Fonte: WOOLDRIDGE et al., 2000).

O modelo de agentes identifica as classes (ou tipos ou famílias) que irão fazer parte da organização multiagente sendo desenvolvida, bem como as instâncias que devem ser geradas a partir destas classes e implementadas computacionalmente.

Uma classe de agentes é a melhor concepção possível para um conjunto de papéis (como identificado no modelo de papéis) definida por meio do processo de mapeamento que considera a lógica funcional para a associação e a atribuição de papéis a uma determinada classe de agente. De fato, uma correspondência de um para um entre o modelo de papéis e a classe de agentes pode ocorrer, mas não necessariamente. Neste sentido, buscou-se selecionar um pacote de papéis correlatos mais apropriados e relacioná-los a uma classe particular de agentes.

Por sua vez, o modelo de serviços tem por objetivo identificar os serviços associados com cada uma das classes de agentes identificadas. O termo serviço na metodologia GAIA significa a função do agente e são decorrentes da lista de protocolos, atividades e responsabilidades associadas a cada papel e, em especial, a partir das propriedades de execução definidas previamente no modelo de papéis.

É importante observar que o modelo de serviços não recomenda uma forma de implementação, métodos ou recursos de software, mas deixa o desenvolvedor livre para realizar a implementação dos serviços a partir de arcabouços e métodos mais apropriados.

O último modelo da etapa de projeto, isto é, o modelo de afinidades é, provavelmente, o modelo mais simples e define os elos (conexões) de comunicação existentes entre as classes de agentes, em particular, os caminhos das mensagens.

Este modelo é representado por um grafo, onde os nós correspondem às classes de agentes e os arcos correspondem aos caminhos. O modelo estabelece quais agentes podem se comunicar diretamente com quais outros agentes, porém não define quais mensagens são enviadas ou quando estas são enviadas. Assim, ele somente indica que um elo de comunicação existe e tem por finalidade identificar gargalos de comunicação durante a execução e possíveis inconsistências na troca de mensagens.

Por fim, o processo de projeto pode ser sintetizado nos seguintes estágios:

1) Criar um modelo de agentes: agregar e atribuir papéis às classes de agentes e refinar a hierarquia de classes de agentes, bem como definir as instâncias das classes.

2) Desenvolver o modelo de serviços mediante o exame da lista de atividades, protocolos e responsabilidades definidas para cada papel.

3) Desenvolver o modelo de afinidades a partir do modelo de interações e do modelo de agentes.

4.2.1 Modelo de agentes

A figura 4.14 apresenta o modelo de agentes propostos 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).

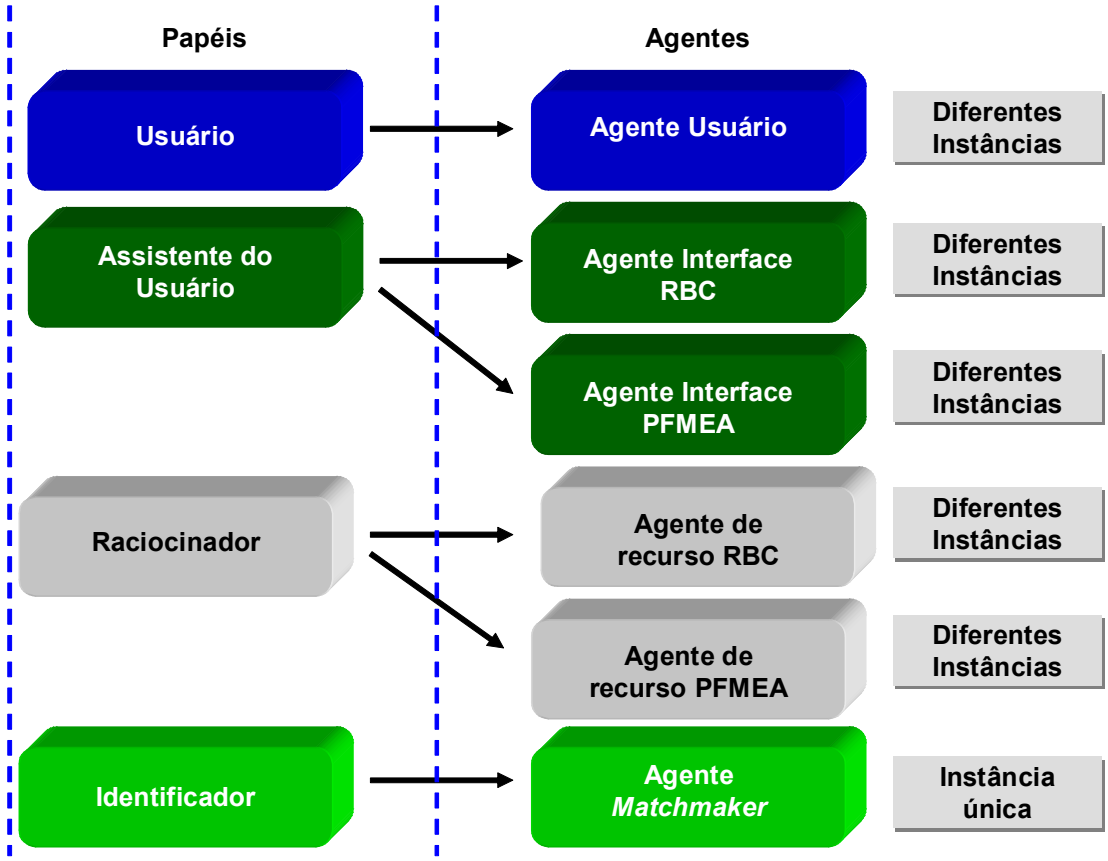


Figura 4.14 – Modelo de agentes proposto.

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 4.14, 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 quanto forem os usuários requisitando apoio.

Por outro lado, o papel do identificador corresponde a somente um agente, denominado Agente *Matchmaker*[2], o qual centraliza todos os registros de serviços e localizações dos demais agentes e deve ser mantido sempre ativo na organização.

4.2.2 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).

Dentro deste contexto, destaca-se que no serviço em questão um caso é definido como:

$$Ci =< Di, Si, Oi > \tag{1}$$

Onde D_i é o descritor (atributo) do caso que representa o novo caso sob consulta ou a situação problema, S_i a solução sugerida e O_i os resultados esperados.

A medida de similaridade global, prevista para a determinação do grau de similaridade entre o novo caso sob consulta e um caso da base de conhecimento considerando todos os descritores modelados, é calculada pelo algoritmo *nearest neighbour* normalizado como mostra a equação (2) (KOLODNER, 1993).

[2] 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.

$$sim(C_{novo}, C_{base}) = \frac{\sum_{i=1}^{n} sim_i(v_i^{novo}, v_i^{base}) \times w_i}{\sum_{i=1}^{n} w_i} \quad (2)$$

Onde w_i é o peso que expressa a importância do descritor do caso i, v_i^{novo}, v_i^{base} são os valores do descritor i no novo caso e no caso da base de conhecimento respectivamente, e $sim_i(v_i^{novo}, v_i^{base})$ é a similaridade entre este dois valores.

Para um dado descritor x_i a similaridade entre os dados atômicos do tipo numérico é calculada usando-se a função linear (equação (3)) que decresce com a distância entre os valores ponderada pelo tamanho do intervalo l_i assumido pelo domínio dos valores dos descritores (KOLODNER, 1993).

$$sim_i(v_i^{novo}, v_i^{base}) = \begin{cases} 1 \quad se \quad v_i^{novo} = v_i^{base} \\ 1 - (\left| v_i^{novo} - v_i^{base} \right| / l_i) \quad se\ v_i^{novo} \neq v_i^{base} \end{cases} \quad (3)$$

Para um dado descritor x_i a similaridade entre os dados atômicos do tipo *string*, símbolos e *booleanos* é calculada pela correspondência exata do novo caso e do caso da base.

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.

4.2.3 Modelo de afinidades

O modelo de afinidades é representado na figura 4.15 e 4.16 e ilustra a comunicação prevista entre os agentes propostos pelo modelo.

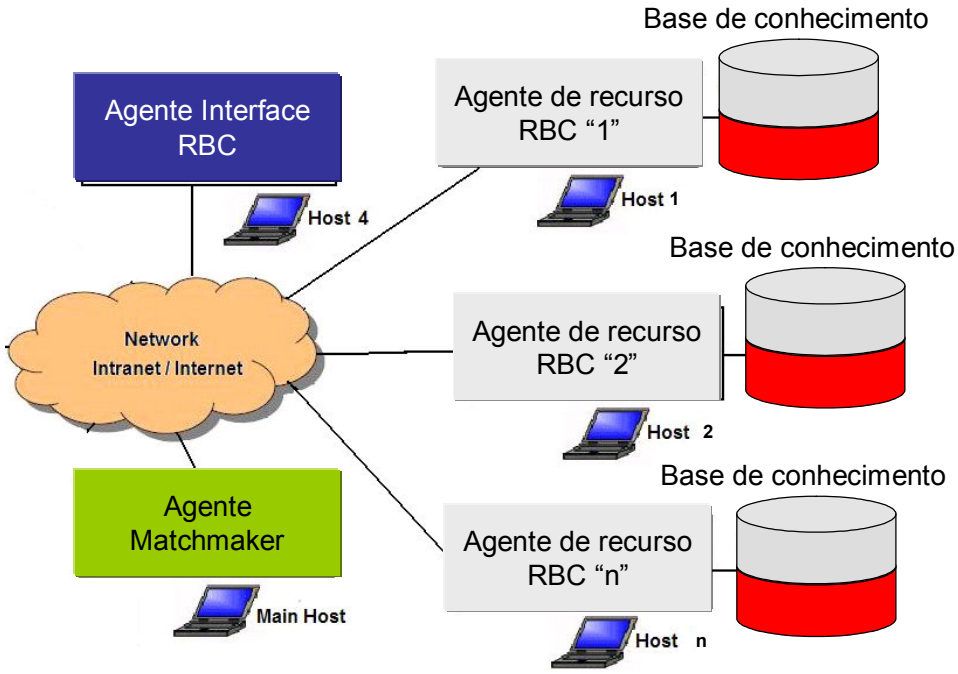


Figura 4.15 – O modelo de afinidades proposto (Parte I).

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

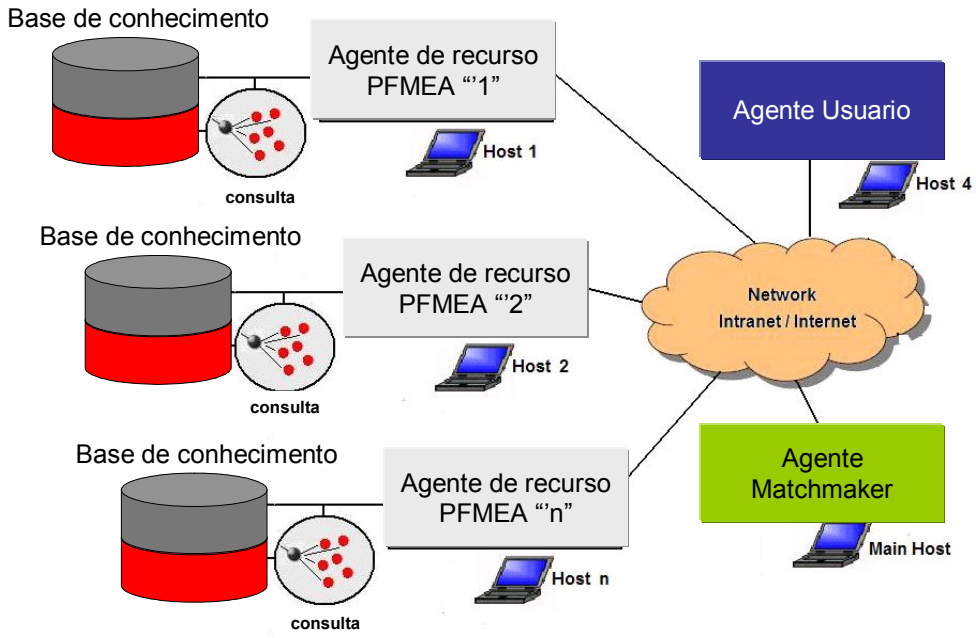


Figura 4.16 – O modelo de afinidades proposto (Parte II).

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.

Na próxima subseção apresenta-se a conceituação e a formalização das bases de conhecimento dos agentes de recursos RBC e PFMEA.

4.3 CONCEITUAÇÃO E FORMALIZAÇÃO DAS BASES DE CONHECIMENTO

De acordo com o modelo conceitual e as especificações de projeto da organização multiagente apresentadas nas subseções anteriores deste capítulo, cada uma das classes (ou famílias) de agentes de recursos de conhecimento devem ser dotadas de bases de conhecimento específicas considerando a finalidade e a natureza de cada classe.

Neste sentido, de um ponto de vista metodológico e científico a conceituação e a formalização destas bases fundamentam-se, inicialmente, no trabalho seminal de Buchanan et al. (1983) no campo de pesquisa de aquisição de conhecimento para sistemas inteligentes. Esse trabalho revela um amplo processo de aquisição que compreende as fases de identificação das fontes, conceituação, formalização, implementação e testes.

Dentro desta perspectiva, Garcia et al. (2005) argumenta que a fase de identificação é semelhante à fase de análise de requisitos em Engenharia de Software, e consiste em aprofundar a análise dos elementos do domínio junto às fontes disponíveis na literatura visando identificar as classes de problemas de não-conformidades para os quais a base de conhecimento deve ser projetada.

Por sua vez, a fase de conceituação envolve o processo de modelagem dos objetos e estruturas do domínio em termos de conceitos e relações permitindo a representação do conhecimento sobre o domínio, de modo que este possa ser compartilhado no âmbito de sistemas computacionais de recuperação e inferência (GARCIA et al., 2005).

E a fase de formalização concentra-se no processo de modelagem computacional do problema e envolve a escolha do formalismo que melhor se adapte a para representar o problema/solução, bem como os métodos de busca no espaço de soluções (GARCIA et al., 2005).

A fase de implementação envolve a codificação destas bases de conhecimento mediante linguagens e ferramentas computacionais apropriadas, enquanto que a fase de testes

compreende os procedimentos de verificação e validação. Neste trabalho de tese, estas duas fases são apresentadas e discutidas nos Capítulos 7 e 8, respectivamente.

Nas próximas subseções são apresentadas e discutidas as etapas de conceituação e formalização das bases de conhecimento tanto para as classes de agentes de recursos de conhecimento tanto para a técnica de raciocínio baseado em casos (agente RBC) quanto para as classes de agentes de análise de modos de falha (agente PFMEA).

4.3.1 Conceituação e formalização da base de conhecimento dos agentes RBC

As aplicações da técnica de raciocínio baseado em casos (RBC) são influenciadas significativamente pelo conteúdo e pela estrutura da representação dos casos armazenados na base de conhecimento ou memória de casos (KOLODNER, 1993; AAMODT e PLAZA, 1994; WATSON, 2003; von WANGENHEIM e von WANGENHEIM, 2003). De acordo com Kolodner (1993) e von Wangenheim e von Wangenheim (2003) um caso pode ser entendido como um elemento de conhecimento contextualizado que representa uma experiência ou um episódio concreto que ensina uma lição fundamental visando atingir os objetivos de um raciocínio. Aqueles autores revelam ainda que um caso contém uma lição aprendida representada pelo conteúdo do caso e pelo contexto em que a lição pode ser reusada.

Neste sentido, um caso pode ser considerado como um registro de experiências ou lições aprendidas que contém conhecimento explícito sobre o domínio, o qual não é somente um resumo de uma solução para um problema prévio, mas um bloco de conhecimento que pode ser reutilizado para a solução de novos problemas similares. Segundo Kolodner (1993) e von Wangenheim e von Wangenheim (2003) um caso, em geral, inclui:

- A descrição de uma situação que ilustra o estado do mundo quando o caso ocorreu.
- A solução sugerida que postula a solução derivada para o novo problema. Uma solução pode ser também uma ação, um plano ou uma informação útil ao usuário da aplicação de RBC.
- O resultado que descreve o estado do mundo após a implementação da solução sugerida, ou ainda quão bem a solução resolveu o novo problema.

A descrição do problema, no âmbito deste trabalho de tese, detalha em essência não-conformidades e o contexto das mesmas nos processos de manufatura. É importante lembrar, conforme apresentado no Capítulo 2, que uma não-conformidade é definida pela norma

internacional ISO 9000:2005, devotada aos fundamentos e ao vocabulário próprio para os sistemas de gestão da qualidade, como "o não atendimento a um requisito".

Por sua vez, a mesma norma define um requisito como "uma necessidade ou expectativa que é expressa, geralmente, de forma implícita ou obrigatória" e, de modo 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. Além disso, esta norma ressalta a diferença entre os termos defeito e não-conformidade, para a qual um defeito tem uma conotação legal, particularmente em relação à responsabilidade civil pela falha do produto.

Em um sentido mais amplo, Pfeifer (1997), Klamma (2000), Lari (2003) e Dhafr et al. (2006) argumentam que o tratamento efetivo das não-conformidades certamente é um pré-requisito inalienável para a melhoria contínua da qualidade dos produtos e dos processos, bem como para a redução de custos.

Estas afirmações são corroboradas no contexto da norma internacional NBR ISO 9004:2000 para sistemas de gestão da qualidade e diretrizes para melhoria de desempenho, bem como no contexto do referencial para a solução de problemas intitulado "*Effective Problem Solving Guideline*[3]*"* (AIAG, 2006).

Adicionalmente, Pfeifer (1997), Klamma (2000) e Dharf et al. (2006) revelam que para o desenvolvimento de um sistema de melhoria é necessária uma forma de aquisição, classificação e análise de não-conformidades, de modo a permitir a comunicação e interpretação dos casos relativos às não-conformidades. Klamma (2000) sugere o uso de uma estrutura conceitual para a não-conformidade caracterizada por meio das seguintes descrições:

- Da não-conformidade, estabelecida pelos atributos que envolvem as circunstâncias nas quais a não-conformidade ocorreu, sua importância e urgência;
- Dos parâmetros que determinam a sua atribuição a um dado componente, subsistema ou sistema;
- Dos elementos do processo de análise da não-conformidade, que são os sintomas coletados, as possíveis e as reais causas, as possíveis e reais ações tomadas no caso;

[3] "*The Effective Problem Solving Guideline*" publicado pelo Grupo de Ação da Indústria Automotiva (*Automotive Industry Action Group*) (AIAG, 2006).

4.3.1.1 Estrutura conceitual proposta para um caso de não-conformidade

Neste cenário, a estrutura conceitual de um caso de não-conformidade representa o cerne da base de conhecimento que será acessada e manipulada pelos agentes de recursos de conhecimento RBC previstos. Esta estrutura envolve três elementos principais: os descritores da não-conformidade *per se*, os descritores da solução sugerida, e os descritores dos resultados obtidos após a implementação da solução, como mostra a figura 4.17.

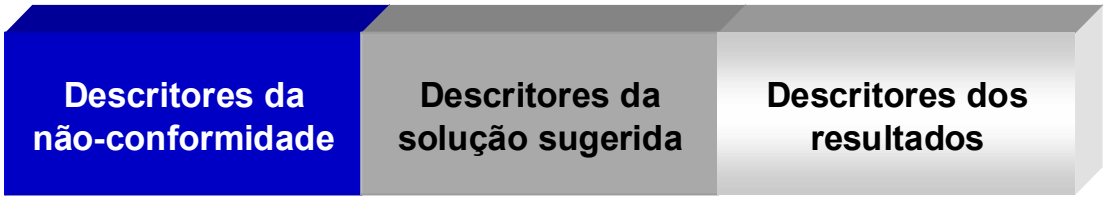


Figura 4.17 – Estrutura conceitual proposta para um caso de não-conformidade.

O primeiro elemento da estrutura proposta visa codificar a situação ou o contexto no qual a não-conformidade ocorreu, sendo, portanto, o ponto de partida para as tarefas e métodos de raciocínio baseado em casos, como mostra a figura 4.18.

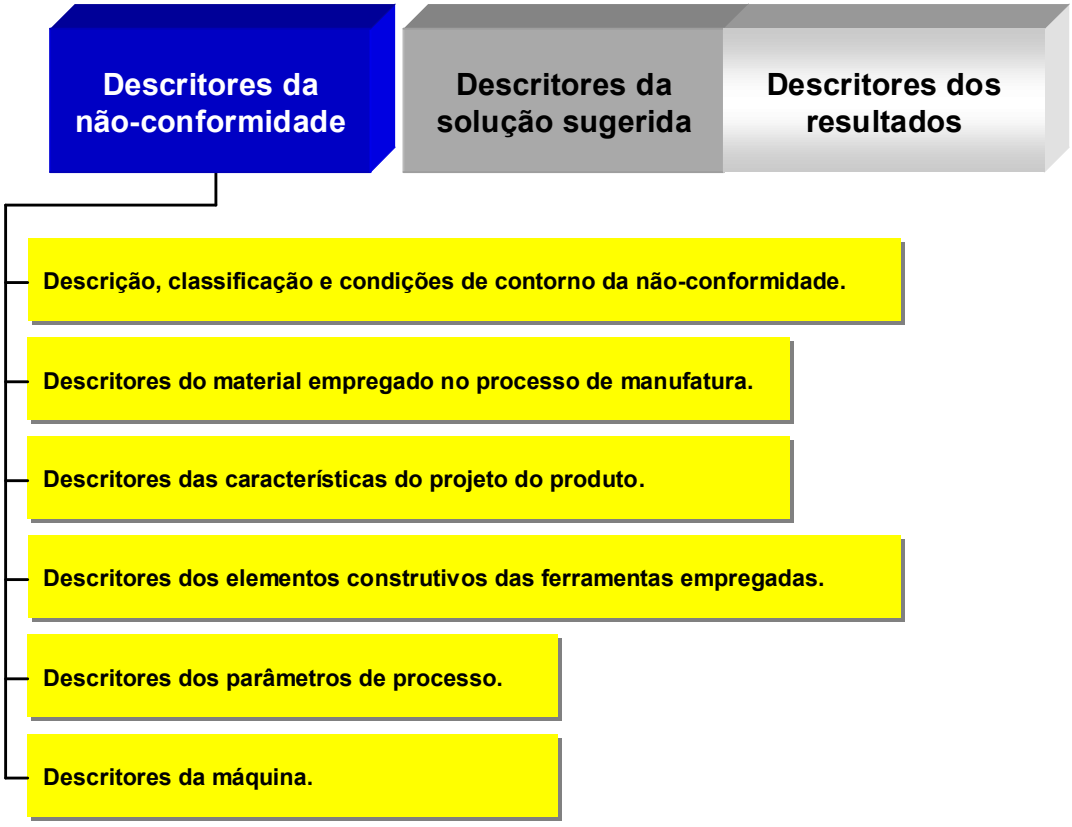


Figura 4.18 – Estrutura conceitual proposta para descrição de uma não-conformidade.

Este elemento (figura 4.18) é composto por seis grupos principais de descritores um grupo relativo à descrição, classificação e condições de contorno da não-conformidade, descritores do material empregado no processo manufatura, descritores das características do projeto do produto, descritores dos elementos construtivos das ferramentas empregadas, dos parâmetros de processo, e descritores da máquina empregada.

Por outro lado, o segundo elemento da estrutura conceitual refere-se aos descritores da solução sugerida, e segue a linha proposta por Kolodner (1993), von Wangenheim e von Wangenheim (2003) e Watson (2003). Este elemento inclui os seguintes descritores: uma solução sugerida *per se*, as etapas de raciocínio usadas pelos especialistas durante a análise da não-conformidade e suas justificativas; as ações corretivas alternativas viáveis identificadas pelos especialistas, mas não adotadas e suas respectivas justificativas; as ações corretivas alternativas não viáveis que também foram identificadas e suas justificativas e, ainda as expectativas que podem surgir após a implementação da solução sugerida, como mostra a figura 4.19.

Neste segundo elemento é importante destacar-se a importância do descritor relativo às etapas de raciocínio usadas por outros especialistas durante a análise de problemas de não-conformidades similares. Pois, em geral, estas etapas podem apresentar os fundamentos que sustentam a definição da solução sugerida e podem ser repetidas durante a análise e solução de uma nova não-conformidades que tenha ocorrido em circunstâncias semelhantes.

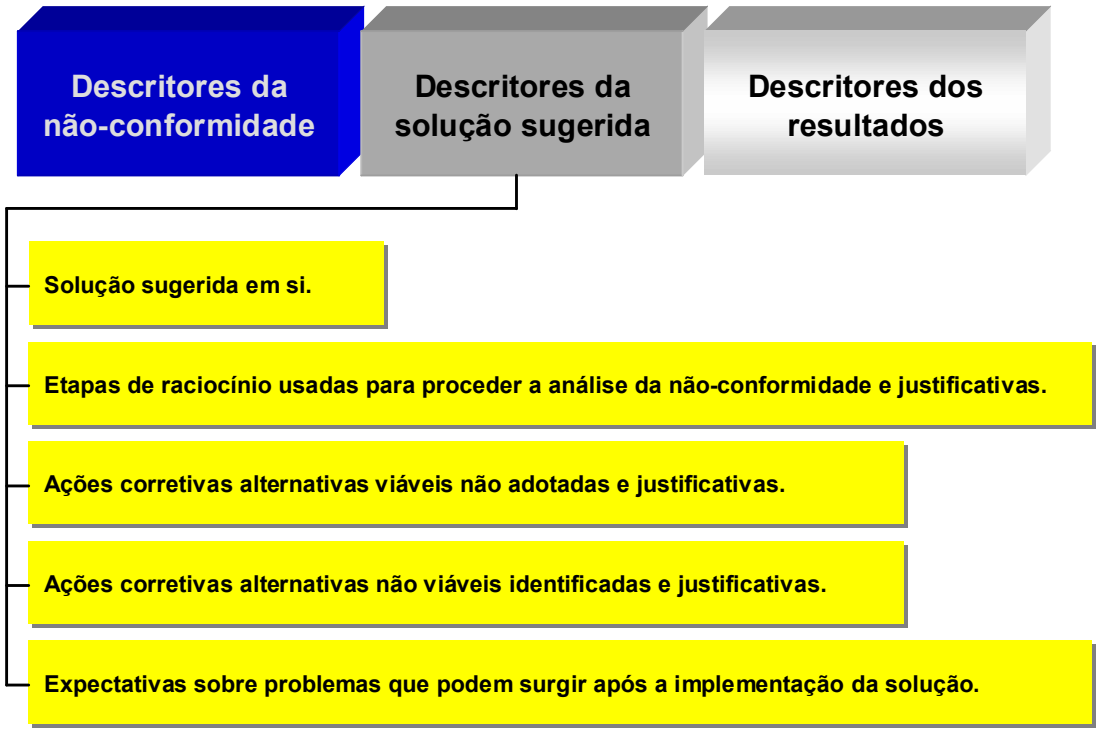


Figura 4.19 – Estrutura conceitual proposta para a descrição da solução sugerida.

Por fim, o terceiro elemento da estrutura conceitual busca descrever os resultados esperados após a implementação da solução sugerida, demonstrando quão bem a solução sugerida resolveu o problema da não-conformidade, como mostra a figura 4.20. Este elemento segue também a linha proposta por Kolodner (1993), von Wangenheim e von Wangenheim (2003) e Watson (2003).

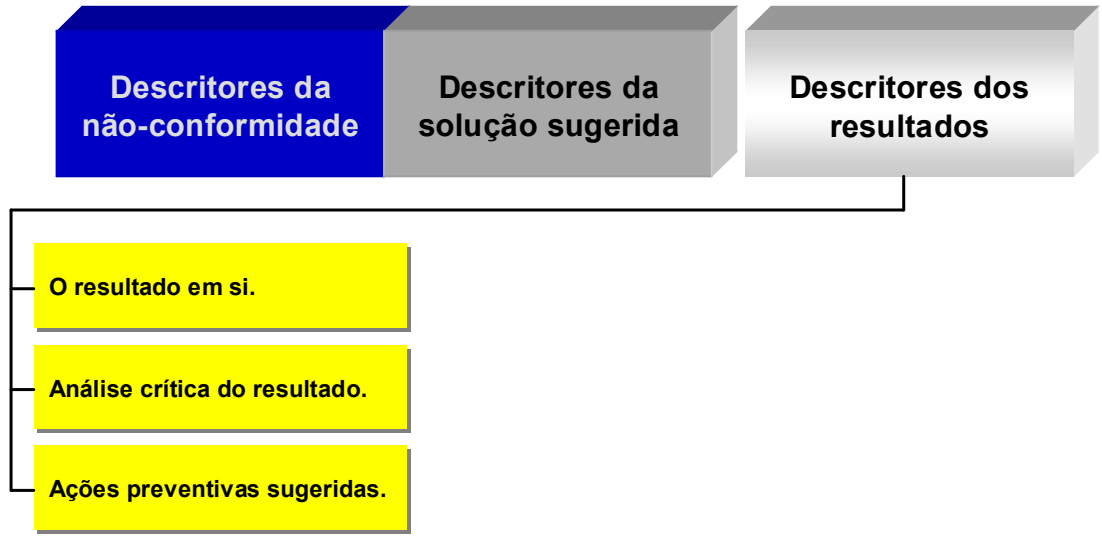


Figura 4.20 – Estrutura conceitual proposta para a descrição dos resultados.

Como mostra a figura 4.20, o terceiro elemento da estrutura conceitual inclui: (a) o resultado *per se*; (b) a análise crítica do resultado que destaca, se este foi condizente com as expectativas e se atendeu aos objetivos, bem como suas respectivas explicações e justificativas; e (c) as ações preventivas sugeridas, que estabelecem diretrizes para evitar a recorrência da não-conformidade ou mesmo elementos para elaboração de um plano de controle.

É importante observar que a estrutura conceitual prevista para um caso de não-conformidade foi desenvolvida com o objetivo de ser independente do domínio de aplicação. Isto é, independente do processo de manufatura modelado dentro do ciclo de vida do produto, especialmente em relação ao segundo e terceiro elemento do modelo.

Todavia, para tornar esta estrutura aderente a um processo de manufatura em particular, é necessário um processo de especialização do primeiro elemento da estrutura (figura 4.17) levando em consideração as características do processo em questão, como será demonstrado na próxima subseção.

4.3.1.2 Estrutura de um caso de não-conformidade para o processo de moldagem por injeção

No contexto de um processo de moldagem por injeção de termoplásticos, adotado neste trabalho para especialização da estrutura conceitual do caso de não-conformidade, o primeiro elemento da estrutura é composto por seis grupos de descritores especializados, como mostra a figura 4.21.

O primeiro grupo diz respeito à descrição, classificação e condições de contorno da não-conformidade no processo de moldagem em questão, enquanto os demais se referem ao material empregado no processo de moldagem, às características do projeto da peça moldada, aos elementos construtivos do molde de injeção, aos parâmetros de processamento e às características da máquina empregada.

O grupo de descritores relativo à descrição, classificação e condições de contorno da não-conformidade é desdobrado, por sua vez, em quatro subgrupos (figura 4.22), a saber: descrição da não-conformidade, condições de contorno, não-conformidade adicional que pode estar associada à não-conformidade principal e a indicação de um componente em particular.

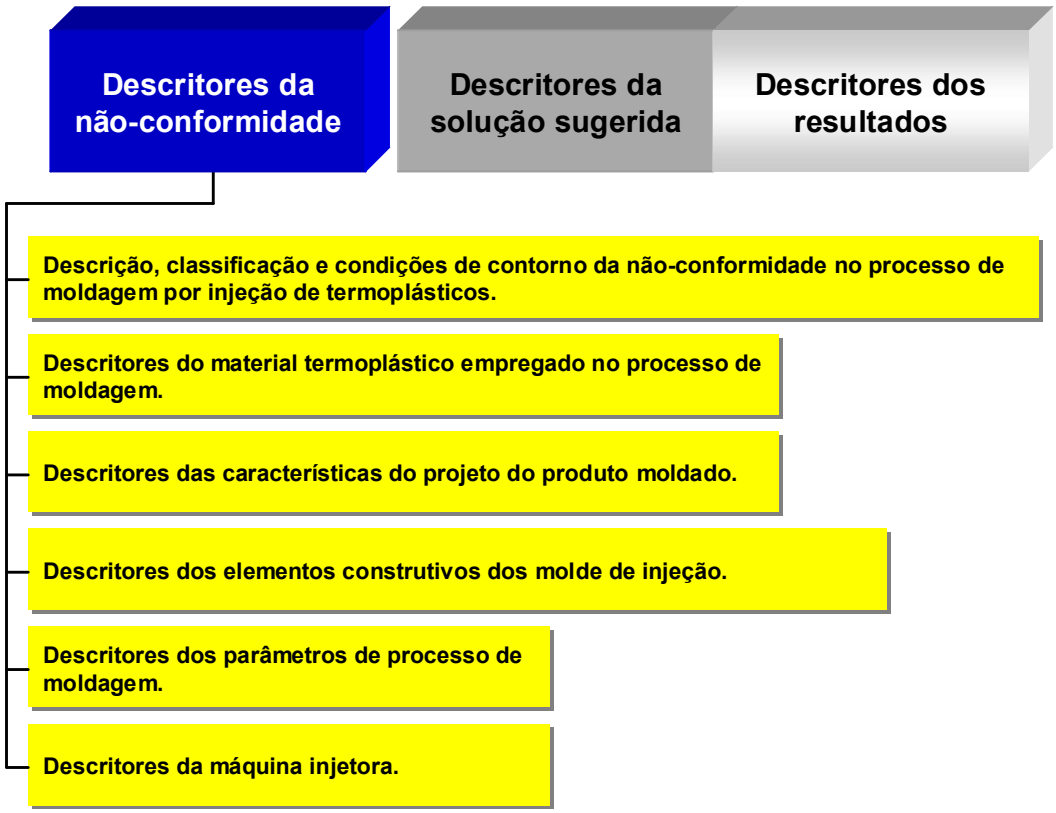


Figura 4.21 – Estrutura conceitual de um caso de não-conformidade no processo de moldagem por injeção.

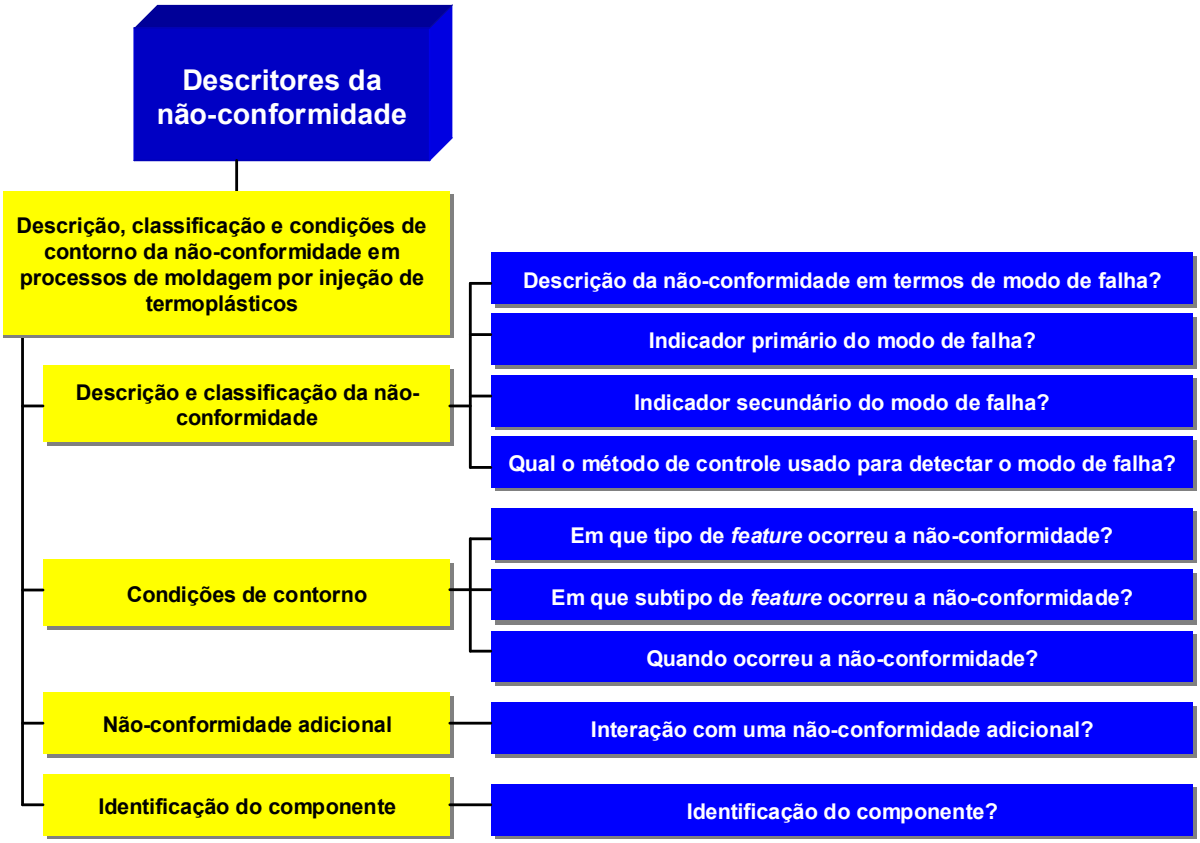


Figura 4.22 – Descritores relativos à descrição, classificação e condições de contorno no processo de moldagem por injeção.

No primeiro subgrupo (figura 4.22) o primeiro descritor busca representar a não-conformidade em termos de um modo de falha, isto é, em relação à maneira na qual o processo de moldagem por injeção falha em atingir os requisitos ou as funções previstas, como por exemplo: variações na massa e espessura da parede da peça, contrações excessivas ou empenamentos, marcas de rechupe, aparência e resistência das linhas de solda inadequadas ou aspecto estético comprometido por manchas ou diferenças de brilho, entre outros.

Do ponto de vista científico este descritor fundamenta-se na ampla literatura da área tanto do ponto de vista acadêmico quanto de aplicações industriais, especialmente no tocante às características do modo de falha, bem como em relação à terminologia adotada. Entre tais publicações se pode citar os trabalhos de Bryce (2003), Goodship (2004), Sancho (2005) e Chen e Turng (2005) relacionados à área de solução de problemas de moldagem (*Troubleshooting Injection Moulding*), além da literatura produzida e disponibilizada pelos fabricantes de resinas termoplásticas como GE *Plastics* (2007), BASF (2007), UMG ABS (2007), DOW (2007), DUPONT (2007), NOVEON (2007) e TEIJIN (2007).

O segundo e terceiro descritores (figura 4.22) correspondem aos identificadores primário e secundário (*primary and secondary identifiers*) que podem ser associados a um dado modo falha. Estes indicadores foram propostos originalmente por Arunajadai et al. (2002) e complementados por Tumer et al. (2003) para o contexto do método de Análise do Modo de Falha e Efeitos em Projetos (FMEA), e aplicados neste trabalho de tese com o objetivo de aumentar a expressividade semântica da descrição do modo de falha de processo.

Neste sentido, o indicador primário define aspectos relacionados à manifestação da falha ou ao seu agente indutor, envolvendo as principais características do modo de falha, ou ainda as características do ambiente no qual o modo de falha ocorreu ou ao tipo de solicitação. Por outro lado, o identificador secundário define os aspectos relacionados com o tipo de material envolvido, características da falha ou presença de outros fatores ou meios específicos.

Neste cenário, por exemplo, o modo de falha "Manchas de queimado" (do termo em inglês *Burn marks*) pode ser associado ao identificador primário "Característica Estética", o qual define a sua principal característica, e o identificador secundário "Cor" que representa a característica secundária do modo de falha. Deve-se lembrar que as manchas de queimado são caracterizadas quando a resina fundida é danificada termicamente devido às altas temperaturas ou então devido ao tempo elevado de permanência no canhão da máquina. Em decorrência disto, subprodutos gasosos são liberados e passam a ser vistos na superfície da peça moldada como manchas marrons ou prateadas (GE, 2007).

Portanto, esta representação permitirá a um agente recurso de conhecimento RBC recuperar casos relacionados a falhas de natureza estética envolvendo a cor de uma peça injetada, aumentado assim o leque de soluções prévias similares que podem ser usadas para apoiar o tratamento de uma nova não conformidade.

Por fim, o quarto descritor do subgrupo busca representar qual o meio de controle usado na identificação do modo de falha, sendo que no exemplo anterior pode-se identificar este tipo de modo através do exame visual da peça.

As condições de contorno que formam o segundo subgrupo (figura 4.22) compreendem três descritores principais, os dois primeiros relacionados com a taxonomia de *features* de moldabilidade à qual o modo de falha pode ser atribuído, e o terceiro relacionado à localização temporal do modo falha ao longo do ciclo do processo.

Dentro desta perspectiva, uma *feature* de acordo com Shah e Mäntilä (1995) pode ser entendida como uma forma geométrica definida por um conjunto de parâmetros que têm um significado especial para engenheiros de projeto e manufatura, e representam entidades relacionadas com a geometria e a topologia de uma peça.

Por sua vez, uma *feature* de moldabilidade pode ser definida como um conjunto de características destinadas a auxiliar a aplicação do projeto orientado para a moldabilidade, e a mesma pode ser subdividida em dois tipos: *feature* de moldabilidade prismática ou rotacional (CANCIGLIERI JUNIOR, 1999; CANCIGLIERI JUNIOR e YOUNG, 2003),

A figura 4.23 ilustra a taxonomia de *features* de moldabilidade prismáticas proposta por Canciglieri Junior (1999), a qual compreende as seguintes *features* prismáticas: de linha de partição de molde, modificadoras, primárias e de transição.

As *features* prismáticas primárias são as *features* básicas ou as primeiras a serem geradas, e nesta taxonomia elas são representadas pelas *features* de parede (paralelas ou perpendiculares à linha de partição do molde). Isto porque, segundo Canciglieri Junior (1999), uma peça moldada por injeção, em geral, é composta principalmente por paredes ou volumes positivos a partir das quais outras *features* denominadas modificadoras podem ser aplicadas, tais como: reforços, nervuras, ressaltos circulares e furos entre outras.

As *features* prismáticas de linha de partição do molde são também mostradas na figura 4.23 considerando-se diferentes alternativas, as quais podem variar em função das *features* primárias envolvidas, bem como em função da posição do degrau formado pela variação das espessuras das *features* em relação à parte externa ou interna da peça.

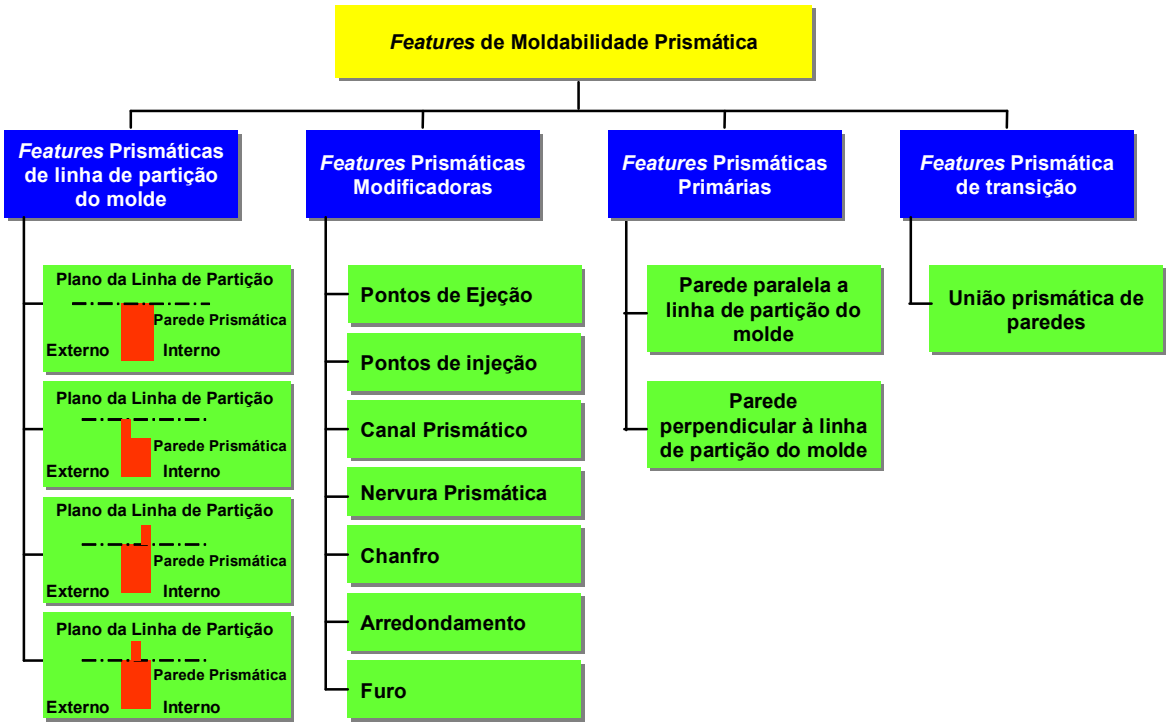


Figura 4.23 – Taxonomia de *features* de moldabilidade prismáticas (Fonte: CANCIGLIERI JUNIOR, 1999).

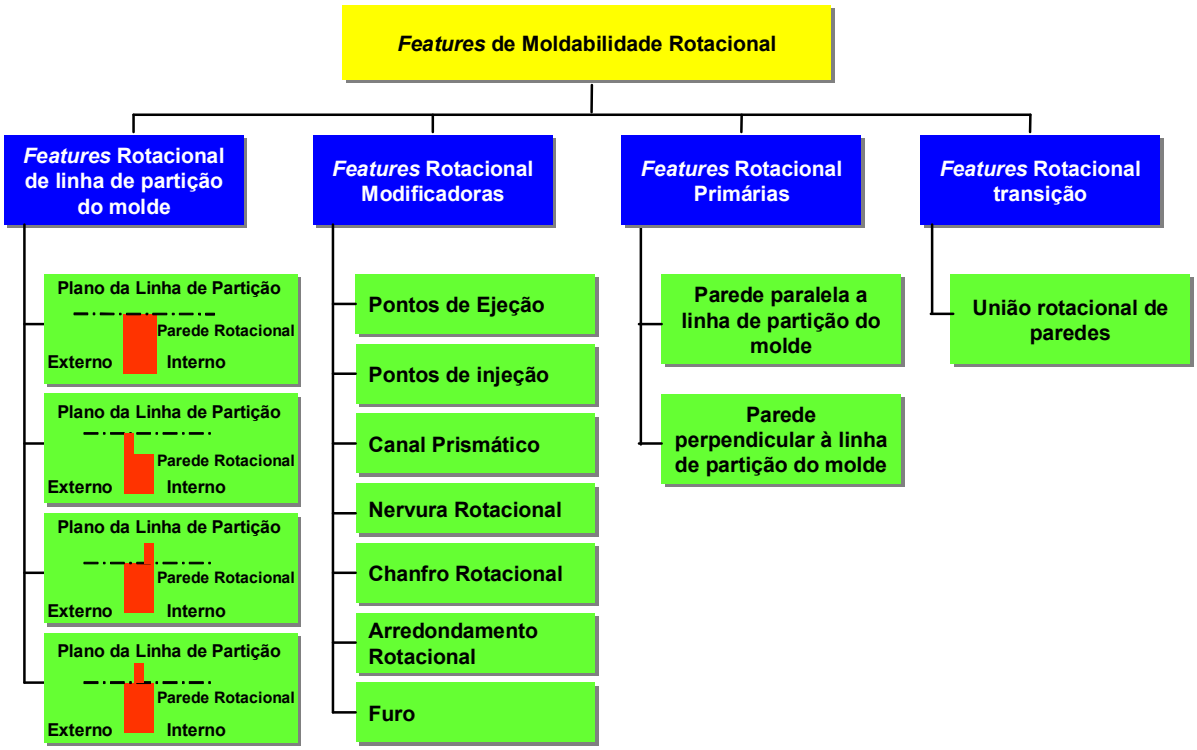


Figura 4.24 – Taxonomia de *features* de moldabilidade rotacionais (Fonte: CANCIGLIERI, 1999).

A taxonomia das *features* de moldabilidade rotacionais é apresentada na figura 4.24, de uma forma equivalente às *features* prismáticas.

Por fim, os dois últimos descritores deste subgrupo (figura 4.22) relacionam: (a) a possível interação entre a não-conformidade descrita em termos de um modo de falha do primeiro descritor com um outro modo de falha e (b) a atribuição do modo de falha a um componente específico se for o caso.

Em síntese, este subgrupo de descritores tem por objetivo aumentar ainda mais a expressividade da descrição de uma não-conformidade referente a um caso em relação à classificação e as condições de contorno de uma não-conformidade no processo de moldagem por injeção de termoplásticos. Isto porque, desta forma, é possível associar a representação de um certo modo de falha ou seus indicadores primários e secundários a uma *feature* de moldabilidade específica, bem como localizá-lo dentro do ciclo do processo.

Esta expressividade na representação permitirá a um agente recurso de conhecimento RBC recuperar casos de não-conformidades onde um dado modo de falha está associado a determinadas *features* em particular, como por exemplo: rebarbas em furos, linha de partição de molde ou nervuras, entre outras *features*.

O grupo de descritores do material de moldagem (figura 4.21) concentra-se em representar o polímero empregado no processo de moldagem por injeção, e a literatura da área revela a

influência decisiva das características reológicas do polímero na qualidade do produto final moldado (SHENOY e SAINI, 1996; MANRICH, 2005; BARIANI et al., 2007). Neste sentido, o material termoplástico é representado na estrutura conceitual do caso de não-conformidades basicamente pela abreviatura estabelecida pela norma da Sociedade Americana para Testes e Materiais ASTM D1600-07 (*Standard Terminology for Abbreviated Terms Relating to Plastics*), como mostra a figura 4.25.

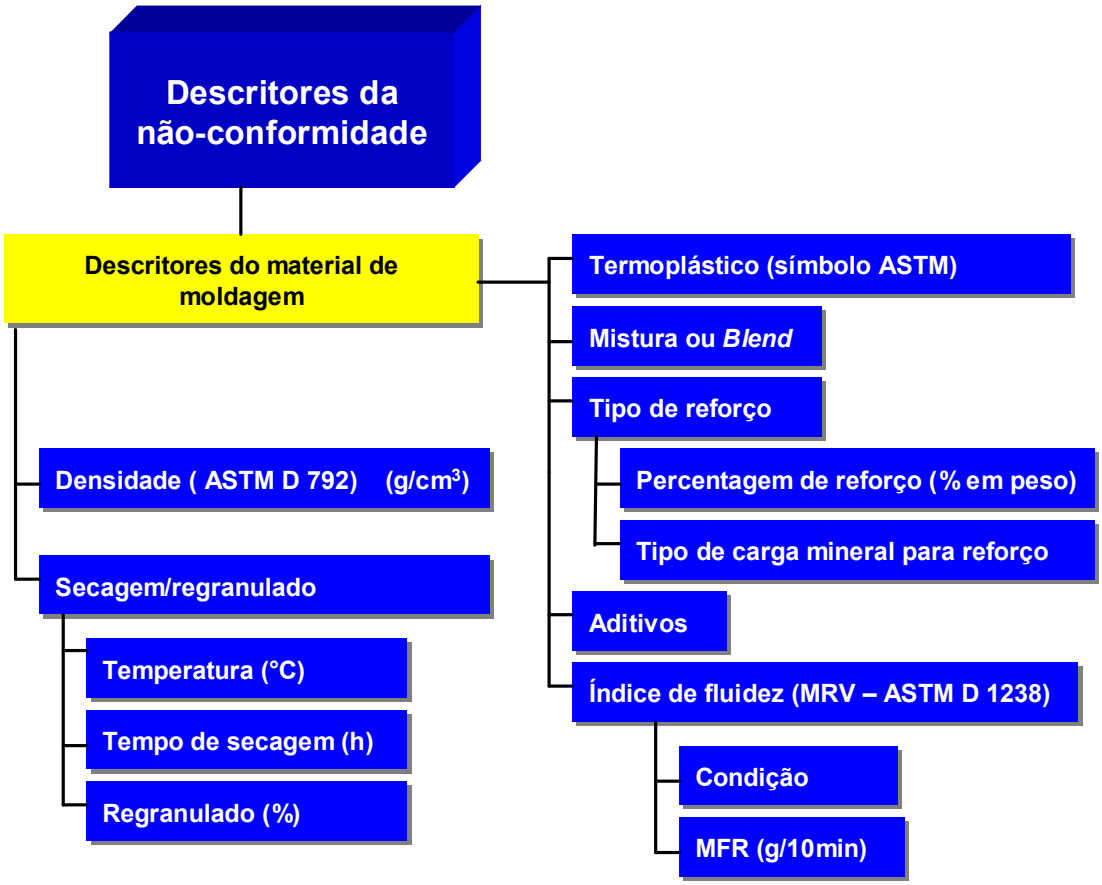


Figura 4.25 – Grupo de descritores do material de moldagem.

Todavia, a representação do material pode ser refinada com o uso dos descritores para duas outras propriedades fundamentais como o índice de fluidez e a densidade do polímero, as quais podem ser encontradas nas especificações técnicas dos materiais comerciais, aqui representadas pelos métodos ASTM D 792 e ASTM D 1238, respectivamente.

O objetivo de incluir o índice de fluidez e a densidade do polímero à estrutura do caso de não-conformidade consiste em refinar a representação do material usando-se apenas a abreviatura ASTM, abreviatura esta que representa uma classe de materiais como, por exemplo, PP (Polipropileno), que pode ser encontrado comercialmente com diferentes formulações de acordo com a aplicação pretendida.

Deste modo, é possível particularizar a representação do polímero ajustando-a em relação às especificações técnicas de um dado fabricante ou, ainda, a um "contratipo", isto é, "polímeros com grupos funcionais ou estrutura química análoga que apresentam características de processamento e propriedades finais semelhantes" (PINTO, 2002).

Adicionalmente, outros descritores tais como agentes de reforço e aditivos podem ser usados para complementar a descrição do material empregado, pois estes componentes afetam significativamente o processamento de um dado material termoplástico (MANRICH, 2005).

Além disso, inclui-se a descrição das características de secagem e, em especial, quando são adicionadas quantidades de material regranulado ao material virgem. Este material regranulado decorre do reaproveitamento de materiais já processados, tais como canais de alimentação moídos. Isto é muito importante, pois, a literatura revela que as propriedades mecânicas do polímero são afetadas por operações de reprocessamento (SU et al., 2007).

Em resumo, os descritores do material de moldagem permitem a atribuição de um modo de falha a um tipo específico de polímero, o qual pode ser representado tanto pela abreviatura ASTM isoladamente quanto combinando outras características reológicas mais refinadas visando uma maior efetividade dos métodos RBC.

O grupo de descritores do projeto da peça moldada, mostrados na figura 4.26, desempenha um papel essencial na estrutura de um caso de não-conformidade. Mok et al. (2000) discutem a importância destes descritores para a determinação dos parâmetros iniciais de moldagem por meio de sistemas inteligentes. Na linha de Mok et al. (2000), a estrutura proposta compreende a área projetada, o volume, a espessura de parede e a complexidade do projeto da peça.

Em relação à complexidade de uma peça moldada, Mok et al. (2000) fundamentam-se no trabalho de Dixon e Poli (1999), o qual foi proposto, originalmente, para a determinação de custos de fabricação de moldes. Dixon e Poli (1999) estabelecem o conceito de complexidade básica e de complexidade subsidiária de uma peça a ser moldada por injeção.

A complexidade básica (C_b) é determinada a partir de duas tabelas, uma para peças classificadas como planas (*flat shape; 2-D or rectilinear*) e a outra para peças prismáticas (*box shape; 3-D or curvilinear*), levando em consideração as dimensões do envelope[4] da peça.

[4] Volume convexo mínimo circunscrito à peça.

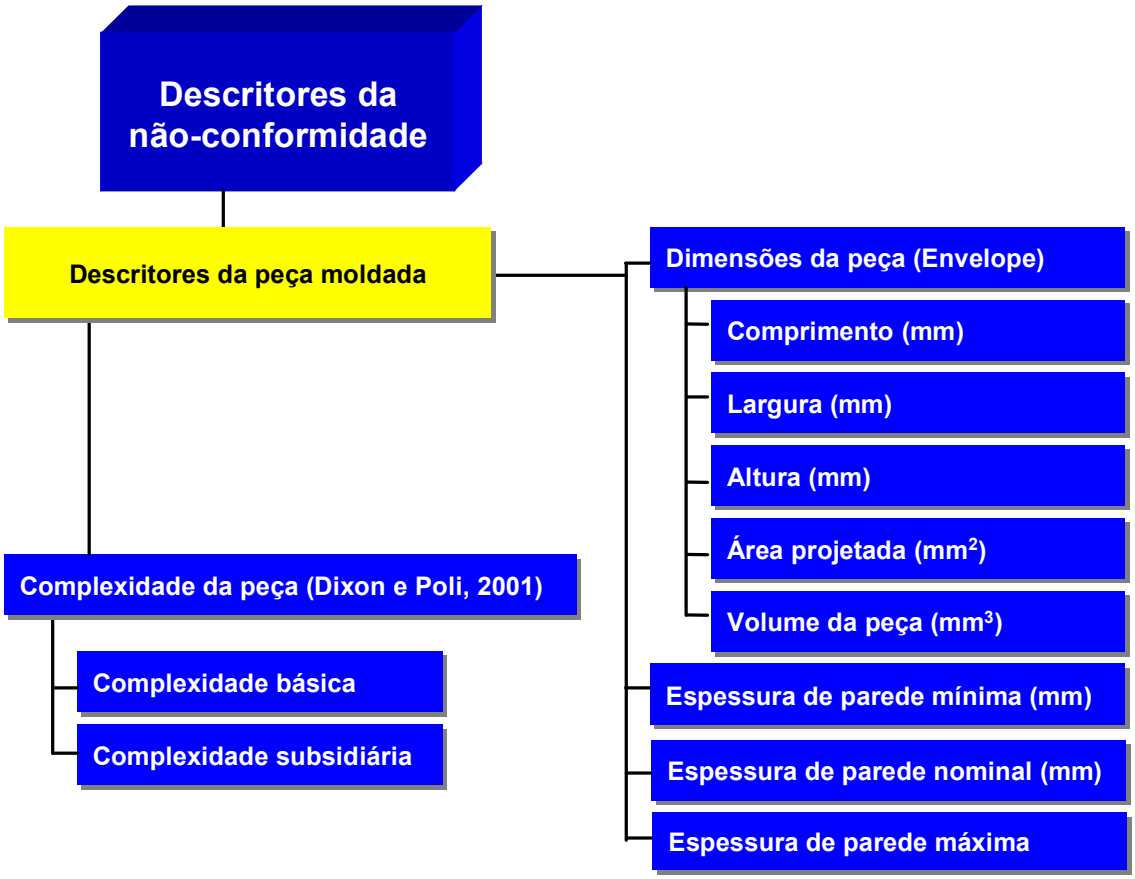


Figura 4.26 – Grupo de descritores do projeto da peça moldada.

Nessa tabelas a complexidade básica (C_b) é determinada considerando: (a) o número de rebaixos ou ressaltos localizados no lado externo da peça (*External Undercuts*), e (b) o número de rebaixos ou ressaltos localizados no lado interno da peça (*Internal Undercuts*). As características dos rebaixos e ressaltos são mostradas na figura 4.27.

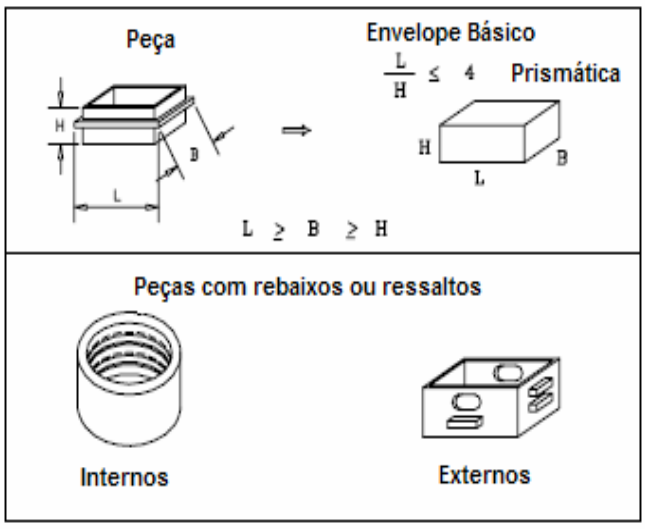


Figura 4.27 – Características dos rebaixos e ressaltos (Fonte: DIXON e POLI, 1999).

A complexidade subsidiária (C_s) é determinada por meio uma tabela que considera: (a) os detalhes da cavidade a partir de uma escala (Baixa, Moderada, Alta e Muito Alta); (b) e a complexidade dos rebaixos e ressaltos externos (*External Undercuts*).

Os detalhes da cavidade levam em consideração a complexidade das *features* da peça moldada que contribuem para os detalhes desta cavidade e que podem ser classificadas como: furos circulares, retangulares e irregulares ou depressões (*holes or depressions*), ressaltos sólidos ou vazados (*Bosses*), nervuras ou paredes não periféricas (*non-peripheral ribs/walls*) e superfícies de fechamento (*side shut-offs*).

É importante observar que as tabelas desenvolvidas por Dixon e Poli (1999), bem como um *software* para a determinação da complexidade básica (C_b) e da complexidade subsidiária (C_s) podem ser encontrados no site do Grupo de Projeto Mecânico[5] (*Mechanical Design Group*) pertencente à Universidade de Massachusetts Amherst.

Em síntese, a inclusão dos descritores do projeto da peça moldada na estrutura do caso de não-conformidade tem por objetivo relacionar um dado modo de falha ou seus indicadores primários e secundários à geometria da peça. Permitem assim que os métodos de recuperação dos agentes RBC possam extrair casos similares mais refinados levando em conta as dimensões e a complexidade da peça moldada.

O molde de injeção é a ferramenta responsável pela forma final dada ao polímero no processo de moldagem e, portanto, seus complexos elementos construtivos têm um papel fundamental na qualidade da peça moldada. Neste trabalho, a escolha dos descritores que compõem a estrutura do caso de não-conformidade fundamenta-se nas publicações de Rees (2002), Sors et al. (2002), Steil (2004), Harada (2004) e Manrich (2005), e são apresentados na figura 4.28.

O primeiro descritor representa a forma construtiva básica do molde, classificada na literatura, em geral, como molde com canais (ou câmaras) quentes ou molde convencional de duas e três placas. Este grupo envolve também o número e a distribuição das cavidades, a distribuição e configuração dos canais de alimentação e entradas de injeção, bem como a configuração do sistema de ventilação empregada no molde. Este grupo de descritores pode ser usado de forma flexível permitindo maior ou menor rigor na representação dos elementos construtivos do molde.

[5] http://www.ecs.umass.edu/mie/labs/mda/dlib/peter/dfm11/dfm11.htm

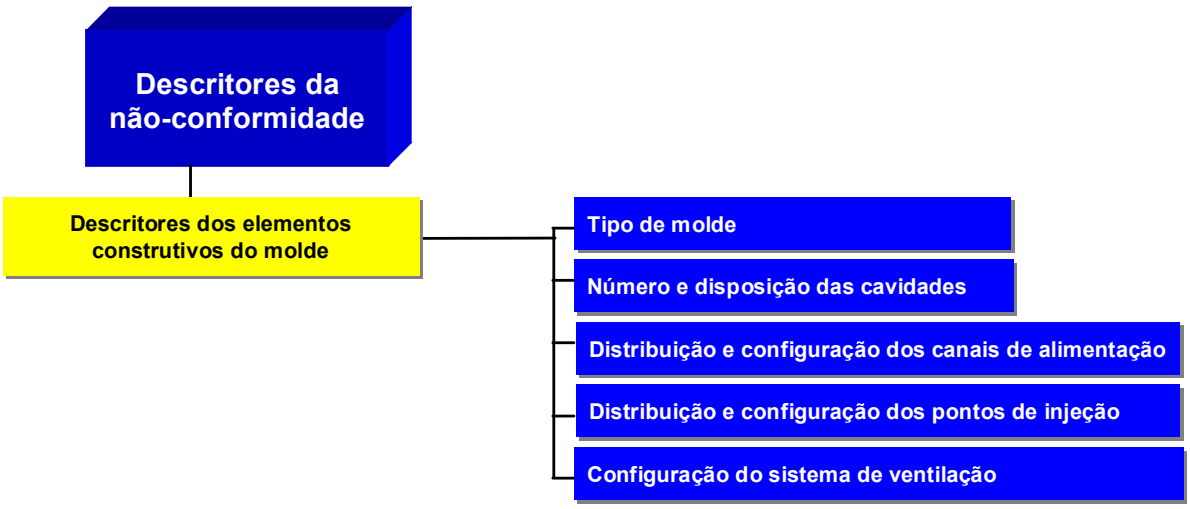


Figura 4.28 – Grupo de descritores dos elementos construtivos do molde.

Os grupos de descritores inter-relacionados restantes têm um papel fundamental e amplamente reconhecido na literatura, tendo em vista a sua influência direta sobre a qualidade final da peça moldada e sobre a produtividade, a saber, os parâmetros do processo de injeção e as características da máquina injetora, como mostram as figura 4.29 e 4.30.

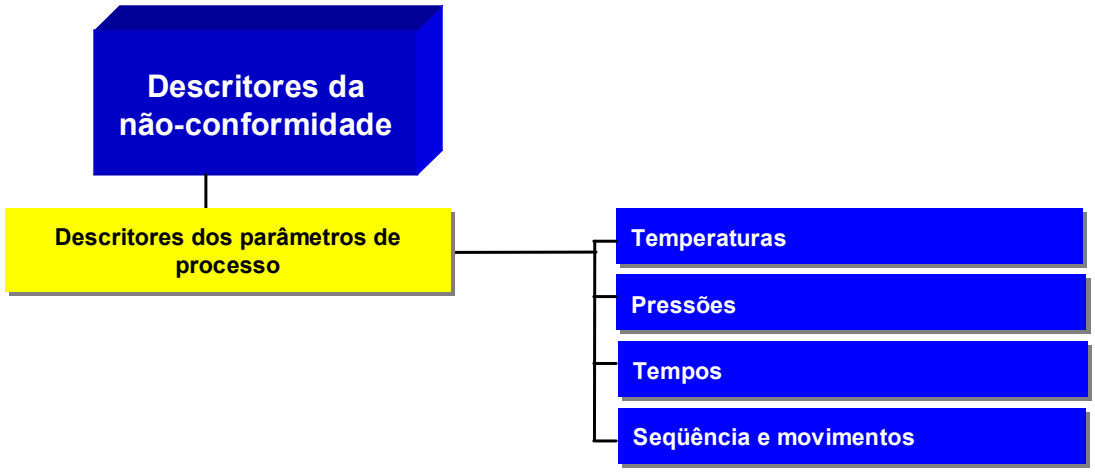


Figura 4.29 – Grupo de descritores dos parâmetros de processamento.

Os parâmetros do processo de injeção, de acordo com Chen e Turng (2005), podem ser divididos em dois níveis distintos. O primeiro nível envolve os parâmetros que podem ser controlados independentemente por meio de controladores lógicos programáveis (CLP) ou controladores Proporcional-Integral-Derivativo (PID), e sensores apropriados disponíveis na própria máquina injetora, denominados pelos autores como variáveis da máquina.

O segundo nível compreende os parâmetros dependentes, os quais dependem não somente dos parâmetros de processos do primeiro nível, mas também do material de moldagem, da máquina e dos elementos construtivos do molde, denominados por Chen e Turng (2005) como variáveis do processo.

Chen e Turng (2005) dividem os parâmetros do primeiro nível em três categorias: temperaturas, pressões e seqüência e movimentos. As temperaturas envolvem as temperaturas no canhão ou barril (do termo em inglês *Barrel*) em diferentes zonas de aquecimento, temperatura do bico e temperatura do fluido refrigerante do molde. As pressões correspondem à pressão de injeção, de recalque e contrapressão.

Por sua vez, os parâmetros de seqüência e movimentos compreendem a velocidade de injeção ou gradiente de velocidades, a rotação da rosca recíproca, dosagem inicial, colchão (via deslocamento da rosca) e o ponto de comutação. Outros autores como Manrich (2005) incluem neste nível o tempo de recalque e o tempo de resfriamento do molde.

Os parâmetros do segundo nível, segundo Chen e Turng (2005), envolvem a temperatura do fundido (no bico de injeção, nos canais de alimentação e na cavidade do molde), a pressão do fundido na cavidade do molde, deslocamento da frente do fundido e a taxa de dissipação de calor.

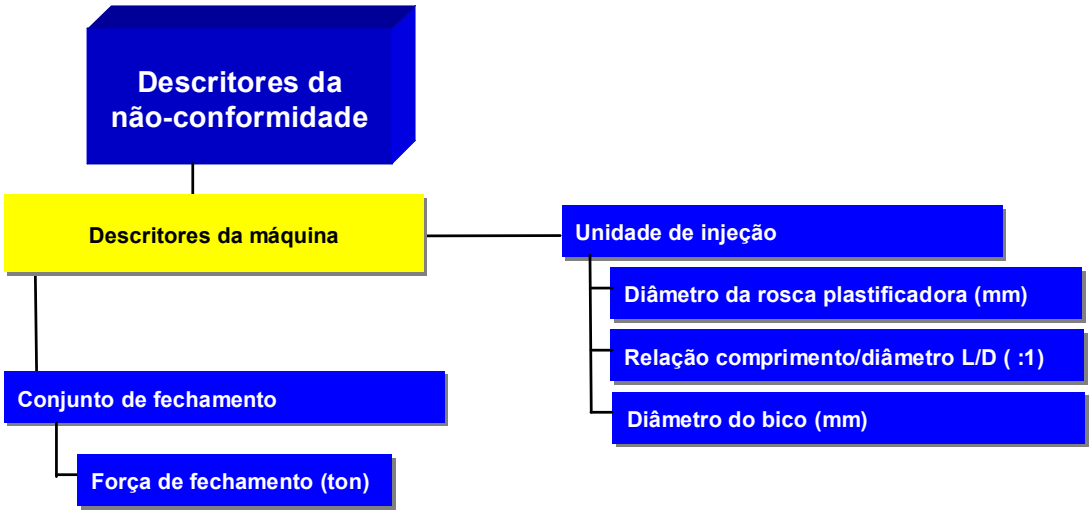


Figura 4.30 – Grupo de descritores da máquina.

Os descritores da máquina envolvem a unidade de injeção e o conjunto de fechamento. A unidade de injeção descreve as principais características da rosca plastificadora e do bico de injeção, enquanto o conjunto de fechamento é caracterizado pela força de fechamento.

Por fim, em resumo, a estrutura conceitual dos descritores dos casos de não-conformidade visa permitir a indexação destes casos no contexto dos métodos de raciocínio baseado em casos encapsulados no comportamento dos agentes RBC previstos no trabalho de tese.

4.3.1.3 Formalização da estrutura conceitual proposta do caso de não-conformidades

A representação da estrutura conceitual dos casos de não-conformidades proposta na subseção anterior será formalizada por meio de vetores atributo-valor, representação esta que é adequada para uma grande parte das aplicações de raciocínio baseado em casos (WATSON, 2003; von WANGENHEIM e von WANGENHEIM, 2003).

Em uma representação de vetores atributo-valor um caso é representado como um conjunto de pares atributo-valor, como ilustra a figura 4.31.

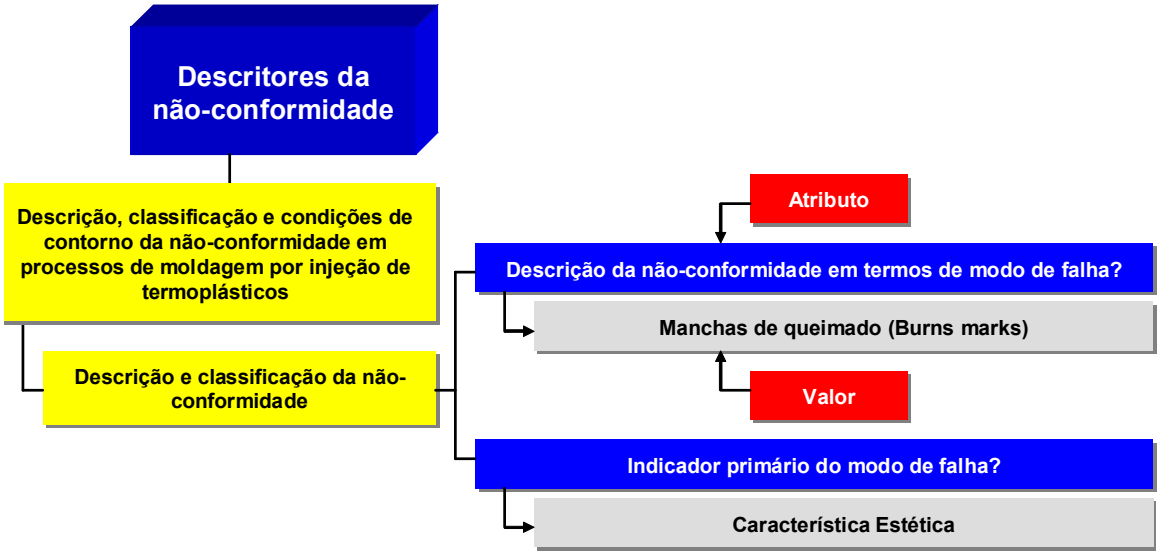


Figura 4.31 – Representação da estrutura do caso por meio de vetores de atributo-valor.

Segundo von Wangenheim e von Wangenheim (2003), cada atributo é, em geral, associado a um domínio que corresponde ao tipo de dado ou à faixa de valores possíveis do atributo. Na estrutura proposta são previstos os seguintes tipos de dados: números inteiros e reais. Os números inteiros são destinados a representar, por exemplo, o valor do atributo < número de cavidades >, enquanto os números reais são usados para representar, por exemplo, o valor do atributo < pressão de recalque >.

A estrutura proposta prevê ainda dados do tipo “símbolo não-ordenado” para representar, por exemplo, o atributo < Descrição da não-conformidade em termos de modo falha >. Este

tipo corresponde a um conjunto finito de valores sem, no entanto impor uma ordem de precedência entre os valores do atributo.

Dados do tipo caractere (*string*) de tamanho arbitrário são usados para representação de atributos em forma textual como, por exemplo, para representar uma solução em si, bem como algum dado na forma de hipertextos (*HTML*) para a representação da localização das imagens associadas a uma dada solução. Além de dados tipo *booleano*.

A escolha da representação por vetores atributo-valor levou em conta a recomendação da aplicação deste formalismo para tarefas de diagnóstico e para grandes bases de casos, bem como pela possibilidade da implementação de medidas de similaridade e métodos de recuperação mais eficientes (von WANGENHEIM e von WANGENHEIM, 2003).

4.3.2 Conceituação e formalização da base de conhecimento dos agentes PFMEA

De acordo com o modelo conceitual e as especificações de projeto da organização multiagente, a base de conhecimento dos agentes PFMEA, cujo conhecimento decorre da aplicação do método de Análise do Modo de Falhas e Efeitos, será representada na forma de uma ontologia cujos fundamentos foram apresentados no Capítulo 2. Neste sentido, na literatura da área diversas abordagens têm sido reportadas para o desenvolvimento de ontologias, entre as quais destacam-se aquelas apresentadas nos trabalhos de Uschold e King (1995), Grüniger e Fox (1995), Fernández-López et al. (1997, 1999), Staab et al. (2001) e Noy e Guinness (2002).

Para o desenvolvimento da ontologia proposta neste trabalho de tese foi adotada a metodologia apresentada por Fernández-López et al. (1997, 1999) denominada de METHONTOLOGY. Esta metodologia foi desenvolvida pelo Laboratório de Inteligência Artificial da Universidade Politécnica de Madri (UPM) e fundamenta-se nos padrões IEEE (*Institute of Electrical and Electronics Engineers*) para desenvolvimento de software e é recomendada pela FIPA[6] (*Foundation for Intelligent Physical Agents*) (2000). Atualmente, ela é considerada uma das metodologias mais maduras em relação aos processos de desenvolvimento de ontologias (CORCHO et al., 2003).

Nesta metodologia a atividade de conceituação organiza e converte uma visão percebida informalmente a partir de um domínio em uma especificação semi-formal mediante um conjunto de representações intermediárias baseadas em notações gráficas ou tabulares que podem ser compreendidas pelos especialistas no domínio. Nesta ótica, o resultado da atividade de conceituação é um modelo conceitual da ontologia.

[6] http://www.fipa.org/specs/fipa00086

Por sua vez, a atividade de formalização transforma este modelo conceitual em um modelo formal que pode ser implementado computacionalmente usando uma linguagem apropriada para ontologias.

4.3.2.1 Modelo conceitual da ontologia

Os principais componente, envolvidos na modelagem conceitual de uma ontologia são conceitos (*concepts*), relações (*relations*) e instâncias (*instances*). Os conceitos em uma ontologia, segundo Fernández-López et al. (1997, 1999), são organizados em taxonomias através das quais os mecanismos de herança podem ser aplicados.

Nesta linha de raciocínio, pode-se representar a taxonomia das entidades no domínio do método de Análise de Modos de Falha e Efeitos, como por exemplo: o "Efeito Potencial de uma Falha" pode ser considerado como uma conseqüência ou impacto de um "Modo de Falha Potencial", em termos da função de uma operação, status de um subsistema ou sistema (IEC, 2005; Stamatis, 2003).

Assim, o conceito "Efeito local" pode referir-se aos efeitos do "Modo de Falha Potencial" no elemento subseqüente, isto é, operação, subsistema ou sistema sob análise, enquanto o conceito "Efeito final" representa o impacto deste "Modo de Falha" no maior nível possível do sistema sob análise. Portanto, os conceitos "Efeito local" e "Efeito final" podem ser representados como subclasses da classe mais geral definida como "Efeito Potencial de uma Falha".

Por outro lado, as relações representam o tipo de associação entre os conceitos de um domínio. Deste modo, relações binárias do tipo "*is Subclass-Of*" podem ser usadas para construir taxonomias como a apresentada no exemplo anterior. Ou ainda, outras relações podem ser estabelecidas, como por exemplo a relação binária "*isRelatedToFunction*", que em termos lógicos relaciona os conceitos "*PotentialFailureMode*" e "*OperationFunction*". Isto é, essa relação liga os conceitos "Modo de Falha Potencial" (ou "*PotentialFailureMode*") e "Função da Operação" (ou "*OperationFunction*"). Além disso, cada relação binária pode ter uma relação inversa associada, a qual associa os conceitos em uma direção inversa, como por exemplo a relação "*hasFailureMode*", que representa a relação inversa de "*isRelatedToFunction*".

Por sua vez, instâncias são usadas para representar elementos ou indivíduos em uma ontologia. Um exemplo de instância do conceito "*OperationFunction*" poderia ser "compactar o material termoplástico na cavidade do molde de injeção", a qual representa uma das funções do ciclo do processo de moldagem de injeção de termoplásticos segundo Manrich (2005).

Neste cenário, a ontologia PFMEA-DL proposta neste trabalho de tese representa o conhecimento no domínio do método de Análise do Modo de Falhas e Efeitos em Processos de Manufatura e Montagem (PFMEA ou *Process-FMEA*). É importante destacar que esta representação está conceitualmente em consonância aos conceitos, relações e termos estabelecidos nas normas SAE J1739 (2002), IEC 60812 (2005), AIAG *Reference* (2001), bem como em relação à literatura relevante da área (STAMATIS, 2003) e, portanto, representa conceitos e relações de amplo consenso e reconhecimento na área.

Assim, o conhecimento no domínio foi representado na forma de uma ontologia a partir de sete eixos de conceitos principais: Produtos (*Product Concepts*), Processo (*Process Concepts*), Funções (*Function Concepts*), Falhas (*Failure Concepts*), Ações (*Actions Concepts*), Descrição do PFMEA (*PFMEA Descriptions*) e Imagens (*Images Concepts*).

O eixo de conceitos do produto (*Product Concepts*) representa o domínio do modelo de produto, particularmente, a estrutura lógica representada pela taxonomia de classes e subclasses que correspondem em termos qualitativos aos seguintes níveis: Sistema (*System*), Subsistemas (*Subsystem*) e Componentes (*Component*), interligados pela relação binária tipo "*is Subclass-Of*".

Por sua vez, o eixo de conceitos do processo (*Process Concepts*) representa a taxonomia da estrutura lógica e temporal dos níveis de Processos (*Process*), Subprocessos (*SubProcess*) e Operações (*Operation*). E o eixo de conceitos de funções (*Function Concepts*) compreende a taxonomia do modelo de funções associados a cada Processo (*Process Function*), Subprocesso (*SubProcess Function*) e Operação (*Operation Function*), respectivamente.

Por outro lado, no eixo de conceitos de falhas (*Failure Concepts*) são representados os principais conceitos e relações do método PFMEA, os quais incluem: Modo de Falha Potencial (*Potential Failure Mode*), Causa Potencial da Falha (*Potential Cause of Failure*), Efeito Potencial da Falha (*Potential Effect of Failure*) entre outros.

O eixo de conceitos de descrição do PFMEA (*PFMEA Descriptions*) representa os conceitos relativos ao perfil de análise e ao perfil do processo de acordo com as indicações das normas SAE J1739 (2002), IEC 60812 (2005), AIAG *Reference* (2001). O perfil de análise representa o conhecimento sobre o grupo de especialistas envolvidos (*Core team*) na análise e o perfil do processo representa os detalhes da realização do próprio PFMEA, tais como datas e referências normativas aplicadas.

Por fim, a ontologia compreende um eixo de conceitos de imagem (*Image Concepts*) que têm por objetivo representar os conceitos e relações binárias no tocante: ao conteúdo da

imagem (*Image Content*), características da imagem (*Image Features*) e fonte da imagem (*Image Source*).

Este eixo busca permitir a indexação e a recuperação de imagens baseada em semântica, isto é, uma imagem de um determinado "Efeito Potencial de Falha" para um dado "Modo de Falha Potencial". Isto porque, em geral, o uso de imagens representa um componente essencial de um processo de análise de falhas.

A figura 4.32 ilustra a conceitualização da ontologia PFMEA a partir da perspectiva do eixo de conceitos de falhas (*Failure Concepts*). A representação gráfica adotada ilustra os conceitos e relações binárias do eixo de conceitos de falhas por meio das caixas de texto na cor cinza e setas de ligação respectivamente. A figura mostra, em especial, a inter-relação destes conceitos com os conceitos modelados em outros eixos, tais como: Funções (*Operation Function*), Ações Recomendadas (*Recommended Action*) e Imagens (*Images*) representadas pelas caixas de texto na cor azul, bem como a inter-relação com os conceitos representados nas taxonomias de processo, produtos, funções, ações, imagens e descrição PFMEA representados pelas caixas vermelhas e verdes respectivamente.

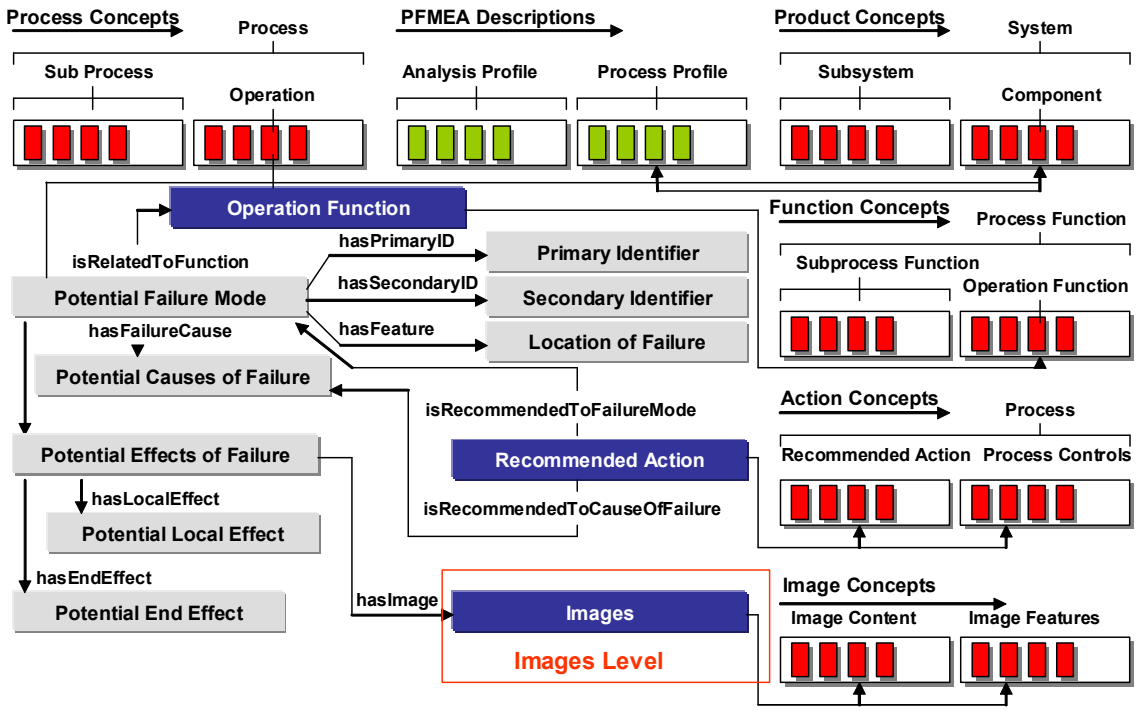


Figura 4.32 – Representação gráfica dos conceitos e relações binárias do eixo de falhas.

Adicionalmente, considerando a finalidade deste trabalho, foram associados também ao conceito "Modo de Falha Potencial" os conceitos de identificador primário e secundário (*primary and secondary identifiers*), bem como a localização da falha com relação a um

modelo de *features* na linha proposta por Shah e Mäntilä (1995). Estes conceitos visam reduzir as possíveis ambigüidades entre as instâncias do conceito "Modo de Falha Potencial", de modo a aumentar a expressividade da representação semântica do conhecimento relacionado ao conceito, bem como a capacidade dos serviços de recuperação de conhecimento.

Deve-se mencionar que o conceito de identificador primário ("*Primary identifier"*) no contexto do método de Análise do Modo de Falha e Efeitos em Projetos, foi proposto originalmente por Arunajadai et al. (2002) e estendido por Tumer et al. (2003), tal conceito define aspectos relacionados à manifestação da falha ou ao seu agente indutor, envolvendo as principais características do modo de falha ou, ainda, as características do ambiente na qual o modo de falha ocorreu ou ao tipo de solicitação.

Por sua vez, o conceito de identificador secundário ("*Secondary identifier"*) define os aspectos relacionados com o tipo de material envolvido, características da falha ou presença de outros fatores ou meios específicos.

Neste sentido, Tumer et al. (2003) apresentam entre outros o seguinte exemplo: uma instância do conceito "Modo de Falha Potencial" pode ser representada como "corrosão galvânica". A esta instância se pode associar o identificador primário "corrosão" que define a principal característica deste modo de falha, e o identificador secundário "corrosão de metais com potenciais eletroquímicos diferentes imersos em um mesmo eletrólito" definindo o tipo de material e a presença de meios específicos relacionados ao modo de falha. Dentro desta perspectiva, estes conceitos foram estendidos neste trabalho também para o método PFMEA.

O eixo de conceitos de ação (*Action Concepts*) (ver figura 4.33) diz respeito aos conceitos e relações no tocante à análise de riscos do método e inclui: as formas de controles atuais aplicadas ao processo visando a detecção ("*Current Process Control Detection*") e prevenção ("*Current Process Control Prevention"*), as quais são associadas a cada um dos "Modos de Falha Potencial".

Neste eixo são representados também os conceitos e relações binárias relativas aos critérios de avaliação do risco (*Rating Criteria*), dentre as quais têm-se: a escala de severidade (*Severity*) ou gravidade de um efeito, em particular, de dado modo de falha; a escala de probabilidade de ocorrência (*Probability of Occurence*) de uma potencial causa de falha de um determinado modo de falha; a escala de probabilidade de detecção (*Probability of Occurence*) que indica a capacidade dos controles atuais de processo de identificar a ocorrência de um dado modo de falha ou causa associada a este modo. Este eixo também contém o número de prioridade de risco (*Risk Priority Number*), como mostra a figura 4.33.

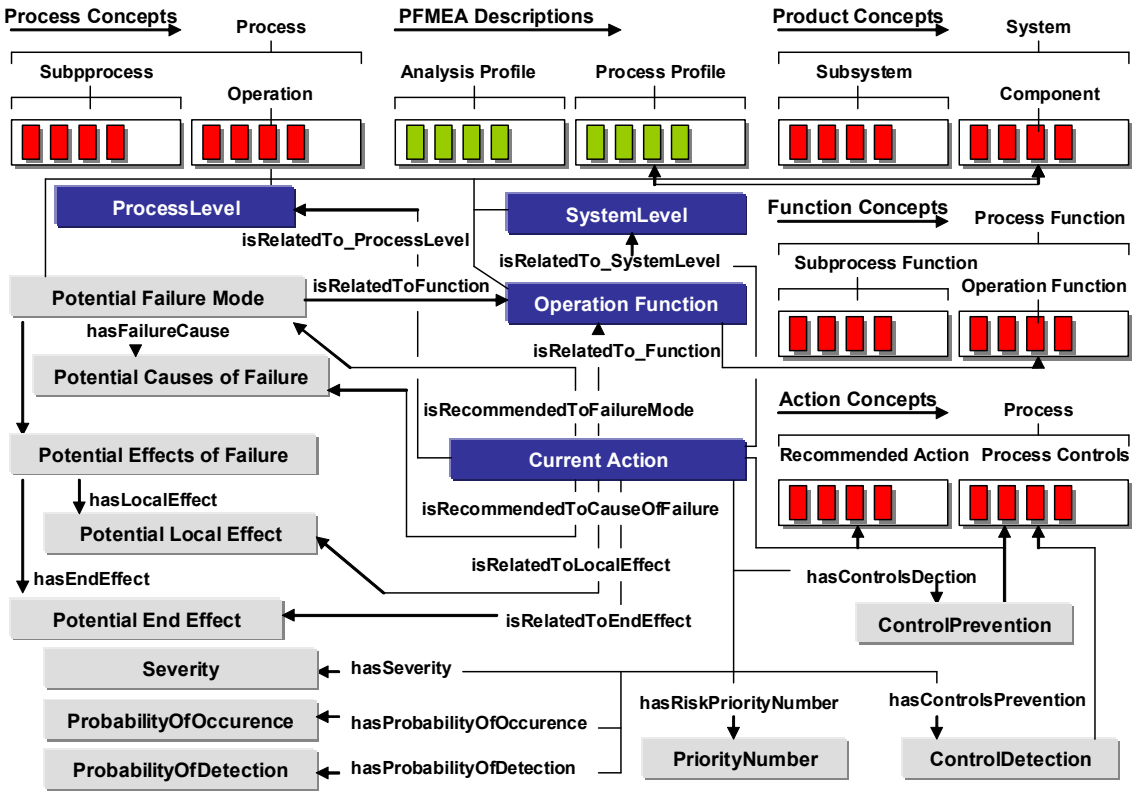


Figura 4.33 – Representação dos conceitos e relações binárias do eixo de ações (Parte I).

Em especial, neste eixo (ver figura 4.34) são representados ainda os conceitos e relações binárias no que diz respeito às ações recomendadas (*Recommended Actions*) quando o número de prioridade de risco ultrapassa os limites máximos admissíveis para um dado processo, bem como os responsáveis pelas ações (*Actions Responsability*). Adicionalmente, neste eixo são representados os resultados das ações previstas durante a análise, indicando-se quais ações foram efetivamente tomadas e seus resultados, além no recalculo do número de prioridade de risco considerando as ações implementadas.

Em síntese, a etapa de conceituação da ontologia PFMEA-DL buscou construir taxonomias considerando os termos normalizados, com amplo consenso e reconhecimento dentro da área de engenharia da qualidade. Adicionalmente, envolveu a construção dos diagramas que representam as relações binárias pertinentes ao domínio do método de Análise do Modo de Falhas e Efeitos.

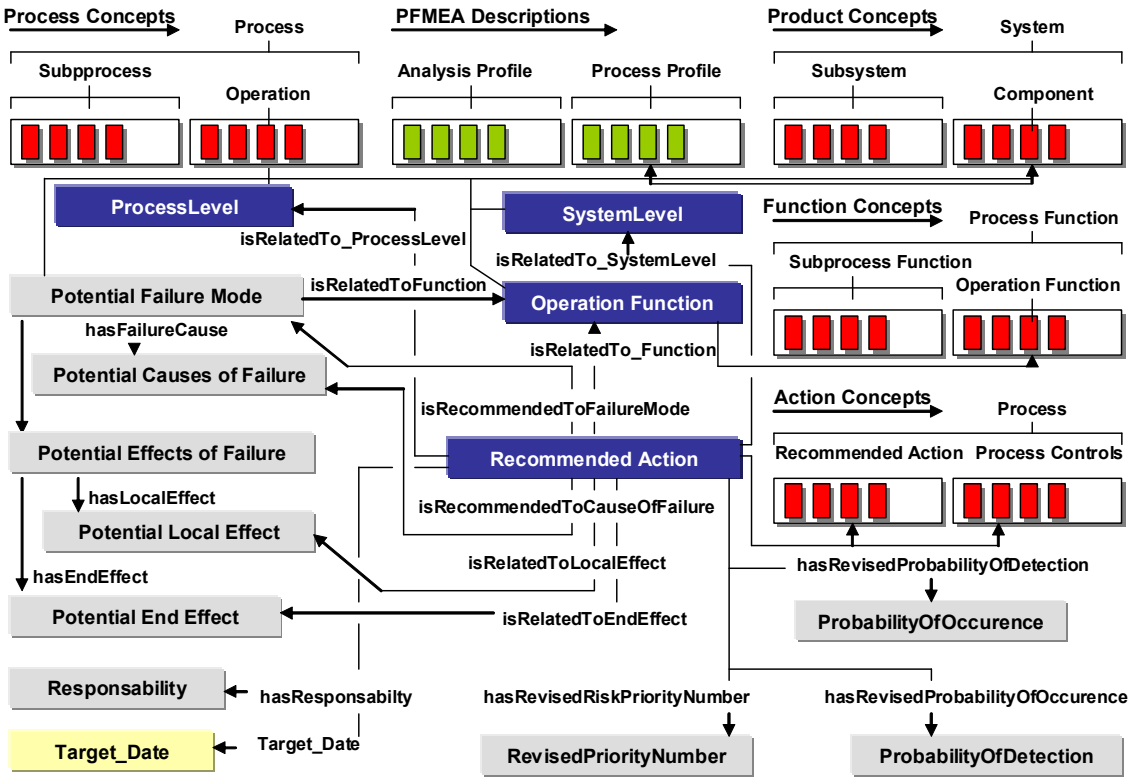


Figura 4.34 – Representação dos conceitos e relações binárias do eixo de ações (Parte II).

4.3.2.2 Lógica de descrições como linguagem formal da ontologia

A Lógica de Descrições (*Description Logics – DLs*), descrita no Capítulo 2, corresponde à denominação mais recente de uma família de linguagens formais de representação de conhecimento baseadas na lógica de primeira ordem (*first-order logic*). Ela surgiu como uma alternativa à representação do conhecimento com base em estruturas de dados *ad hoc* tais como: redes semânticas (*semantic networks*) e os *frames* (BAADER e NUTT, 2003).

Dentro desta perspectiva, a essência da representação do conhecimento por meio de linguagens baseadas em lógica de descrições consiste em estabelecer as descrições elementares do domínio de aplicação a partir de conceitos e papéis atômicos.

Os conceitos atômicos (*atomic concepts*) representam as classes ou conjunto de indivíduos dotados de características afins. Os papéis atômicos (*atomic roles*) representam as propriedades destes conceitos na forma de relações binárias entre indivíduos, e a partir destes, outras descrições complexas são construídas como axiomas por meio de um conjunto de operadores lógicos denominados construtores de conceitos (*concepts construtors*) (BAADER e NUTT, 2003).

Neste sentido, diferentes linguagens formais da família das lógicas de descrições são apresentadas e discutidas na literatura (HAARSLEV e MÖLLER, 2001; BAADER e NUTT,

2003; HORROCKS, 2005b), as quais diferem entre si em função do conjunto de construtores lógicos que cada linguagem pode suportar.

Neste contexto, uma base de conhecimento, referente a um domínio de aplicação, formalizada por meio de lógica de descrições compreende, dois componentes fundamentais: o primeiro, denominado como componente terminológico TBox (*terminological component*) da aplicação, representa o conhecimento sobre as características dos conceitos da ontologia PFMEA-DL (*intensional knowledge*). Este componente é independente do processo de manufatura modelado pela aplicação e compreende, por sua vez, um conjunto de axiomas terminológicos (*terminological axioms*) que são usados para definir os conceitos mais complexos a partir de outros conceitos e papéis primitivos como revelado em Baader e Nutt (2003).

O segundo, denominado de componente de asserção ABox (*assertional component*), representa o conhecimento extensivo que especifica os indivíduos de um domínio de aplicação ou processo de manufatura em particular e suas relações dentro do mesmo nível de abstração. Representa, portanto, a instanciação da estrutura de conceitos modelada pelo componente terminológico TBox (BAADER e NUTT, 2003).

Não obstante o potencial das linguagens formais baseadas em lógica de descrições no âmbito da representação de conhecimento, a sua real aplicabilidade se dá por meio dos sistemas computacionais que as implementam, bem como na capacidade destes sistemas processarem o conhecimento representado de forma explícita com o objetivo de inferir conhecimento implícito por meio de serviços de inferência específicos (BAADER e SATTLER, 2001; HAARSLEV e MÖLLER, 2001, TSARKOV e HORROCKS, 2005).

Nesta linha, atualmente, diversos sistemas computacionais têm sido desenvolvidos dotados destes serviços de inferência baseados no algoritmo *tableaux* (BAADER e SATTLER, 2001) para lógicas de descrições. Em particular, pode-se citar entre outros o sistema RacerPro Server (*Renamed ABox and Concept Expression Reasoner Professional*) que é um sistema de representação de conhecimento que implementa o algoritmo de *tableaux* para linguagem de lógica de descrições $ALCQHI_{R+}$[7], também conhecida com *SHIQ* (HAARSLEV e MÖLLER, 2001, RACER SYSTEMS, 2005). Este sistema oferece diferentes serviços de inferência, tais como: verificação da consistência de conceitos e a verificação da consistência do componente ABox com respeito a um dado TBox no tocante aos possíveis erros de modelagem ou contradições na definição dos conceitos e instâncias.

[7] (*AL* - *Atributive Language*) complementada (*C* - *Complement*) por outros construtores (como por exemplo, *U* disjunção, *N* negação, *E* quantificação existencial, *H* hierárquico, *I* papel inverso, *Q* qualificador de número de restrições), $_{R+}$ mais regras de transitividade.

Portanto, considerando o acima exposto, no presente trabalho de tese a ontologia é formalizada usando-se a linguagem de lógica de descrições, formalização esta que se dá diretamente pelo mapeamento dos conceitos e relações binárias definidas no modelo conceitual da ontologia para o os conceitos e papéis da lógica de descrições.

A figura 4.35 apresenta a estratégia de mapeamento dos conceitos e relações binárias da ontologia em relação aos conceitos, papéis e axiomas do componente TBox, enquanto as instâncias são representadas no componente ABox.

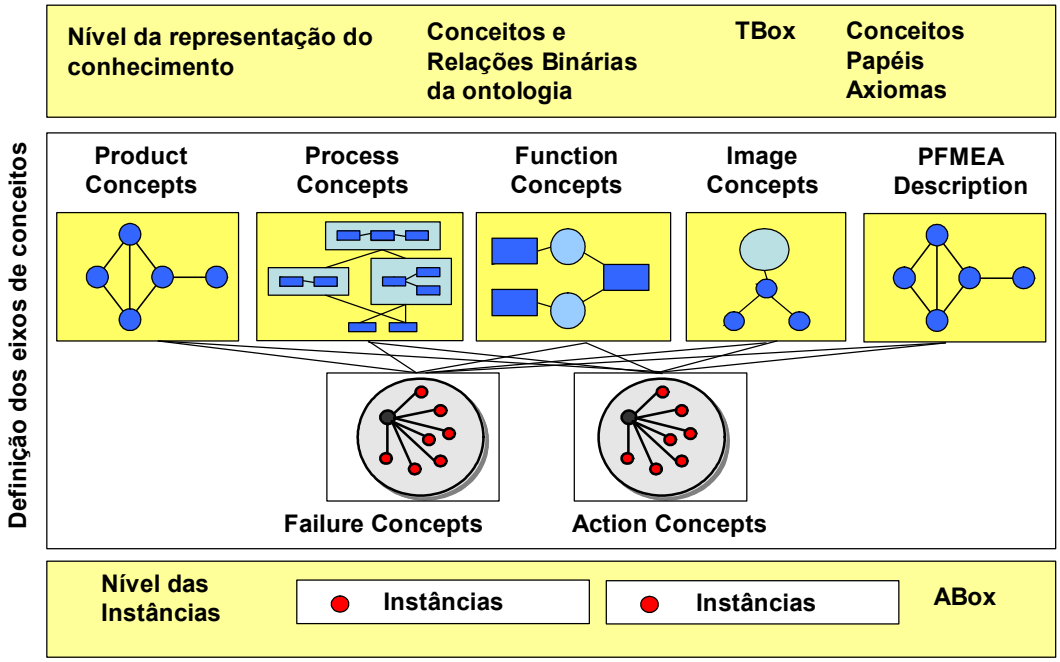


Figura 4.35 – Representação gráfica dos eixos de conceitos da Ontologia PFMEA-DL.

Adicionalmente, é importante ressaltar que, recentemente, foi desenvolvida a linguagem padrão para codificação de ontologias denominada OWL DL (*Web Ontology Language – Description Logic*), proposta pelo *World Wide Web Consortium* W3C (PATEL-SCHNEIDER et al., 2004). Esta linguagem combina um grande poder de expressividade com a possibilidade dos serviços de inferência característicos das lógicas de descrições que pode ser usada para a codificação da ontologia proposta (HORROCKS, 2005a).

CAPÍTULO 5
RECURSOS DE *SOFTWARE* PARA A IMPLEMENTAÇÃO DO MODELO PROPOSTO

O propósito deste capítulo é apresentar as questões relacionadas ao processo de escolha dos recursos de *software*[1] 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[2]) 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.

---

[1] Recursos de software compreendem os pacotes de software, bem como as licenças.
[2] Do inglês, *Case-Based Reasoning (CBR*).

5.1 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[3].

Uma plataforma de agentes é projetada como um conjunto de componentes de software do tipo *middleware*[4], 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 5.1, 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.

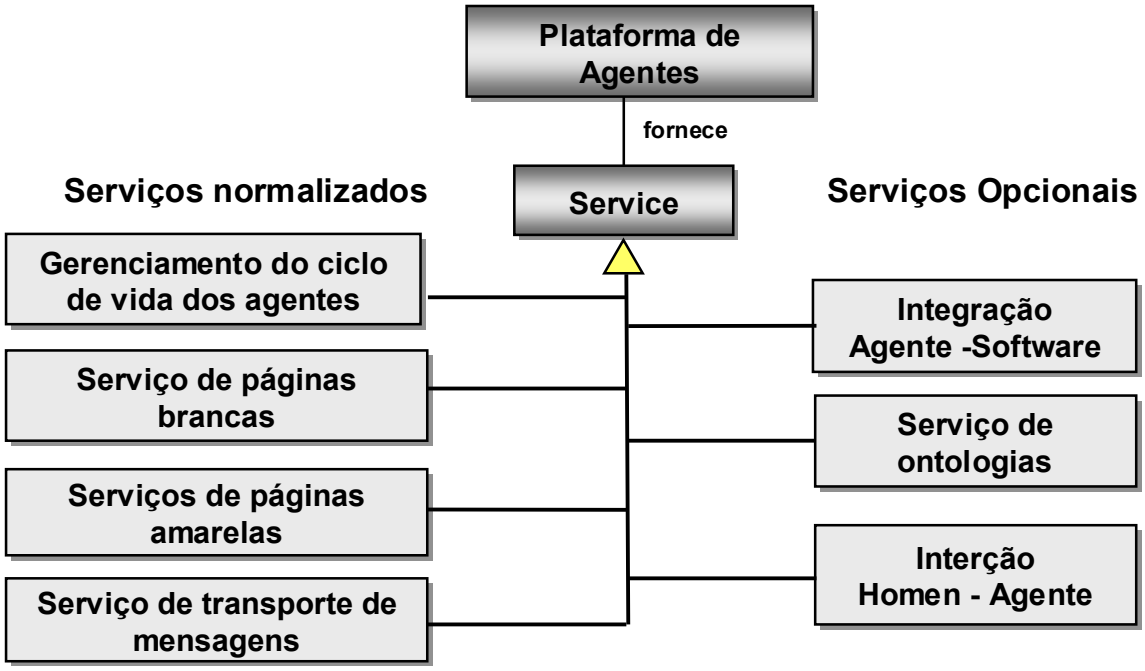


Figura 5.1 – Modelo de serviços fornecidos por uma plataforma de agentes FIPA 2000 (Fonte: FIPA, 2002a)

É importante ressaltar, entretanto, que de acordo com as especificações FIPA 2000 o conceito de serviço é definido em termos de um conjunto de mecanismos funcionais

[3] Do termo em inglês *Frameworks*.
[4] 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).

empregados 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 podem 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). Nesta mesma linha, Fayad et al. (1999) descrevem um arcabouço como uma aplicação de software semicompleta dotada de componentes de software estáticos e dinâmicos que podem ser especializados para produzir uma aplicação específica para um dado domínio e usuário e, devido à sua natureza, pode ser reutilizado como base em muitas outras aplicações.

Os próximos itens apresentam o processo de escolha do recurso de *software* para a implementação do modelo e a avaliação do recurso escolhido descrevendo-se as seguintes características: a plataforma de agentes, o modelo de programação, o modelo de tarefa dos agentes, o modelo de comunicação entre agentes e as características das mensagens e trocas de mensagens.

5.1.1 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), como definido no escopo do trabalho, e ainda considerar as definições de requisitos estabelecidas no Capítulo 4.

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 à tecnologia multiagentes, 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., 2006a), 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 são mantidos, se continuam a ser desenvolvidos pelo grupo de pesquisa original, e se este 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., 2006a), JATLite (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., 2006a), 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., 2006a) 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., 2006a) 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*[5]). 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.

5.1.2 Características do arcabouço JADE essenciais à implementação do modelo

O recurso de *software* adotado para a implementação da tecnologia de multiagentes, denominado JADE[6] (***J**ava **A**gent **DE**velopment Framework*), é um arcabouço de *software* desenvolvido totalmente em linguagem Java (SUN DEVELOPER NETWORK, 2007), 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, 2007), 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, 2003; HORSTMANN e CORNELL, 2001).

De acordo com Bellifemine et al. (2006a) o principal objetivo do arcabouço JADE é facilitar e simplificar o desenvolvimento de sistemas multiagentes assegurando um padrão de

---

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

[6] 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., 2006a).

interoperabilidade com outros sistemas multiagentes. Neste sentido, o recurso JADE contém funções pré-implementadas que compreendem os aspectos de um sistema multiagentes que são independentes de um tipo particular de aplicação, e não envolvem diretamente as particularidades internas dos agentes.

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 apresentada na figura 5.2.

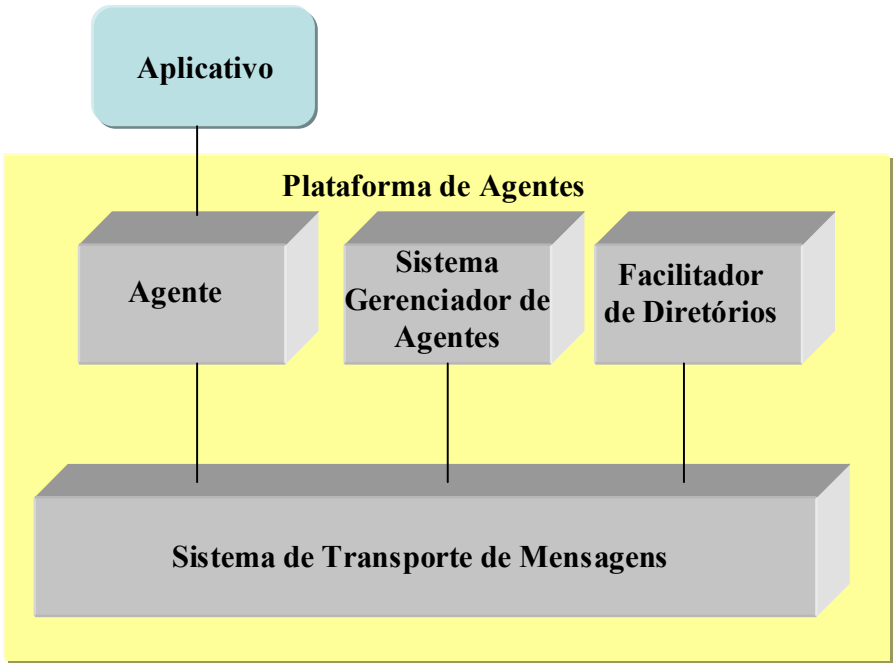


Figura 5.2 – Arquitetura de referência da plataforma de agentes FIPA 2000

(Fonte: BELLIFEMINE et al., 2006a).

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 e 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*[7] 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 provendo todos os serviços de suporte necessário para a execução destes agentes.

Por conseqüência, uma plataforma na terminologia JADE é composta por um *container* principal (ou *main container*) que deve estar sempre ativo, e de todos os outros *container's* (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 5.3 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 Registro RMI (*Remote Method Invocation Registry*).

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

[7] um *thread* (linha de execução) é um fluxo único de controle sequencial dentro de um programa escrito em linguagem JAVA (SUN, 2007)

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*.

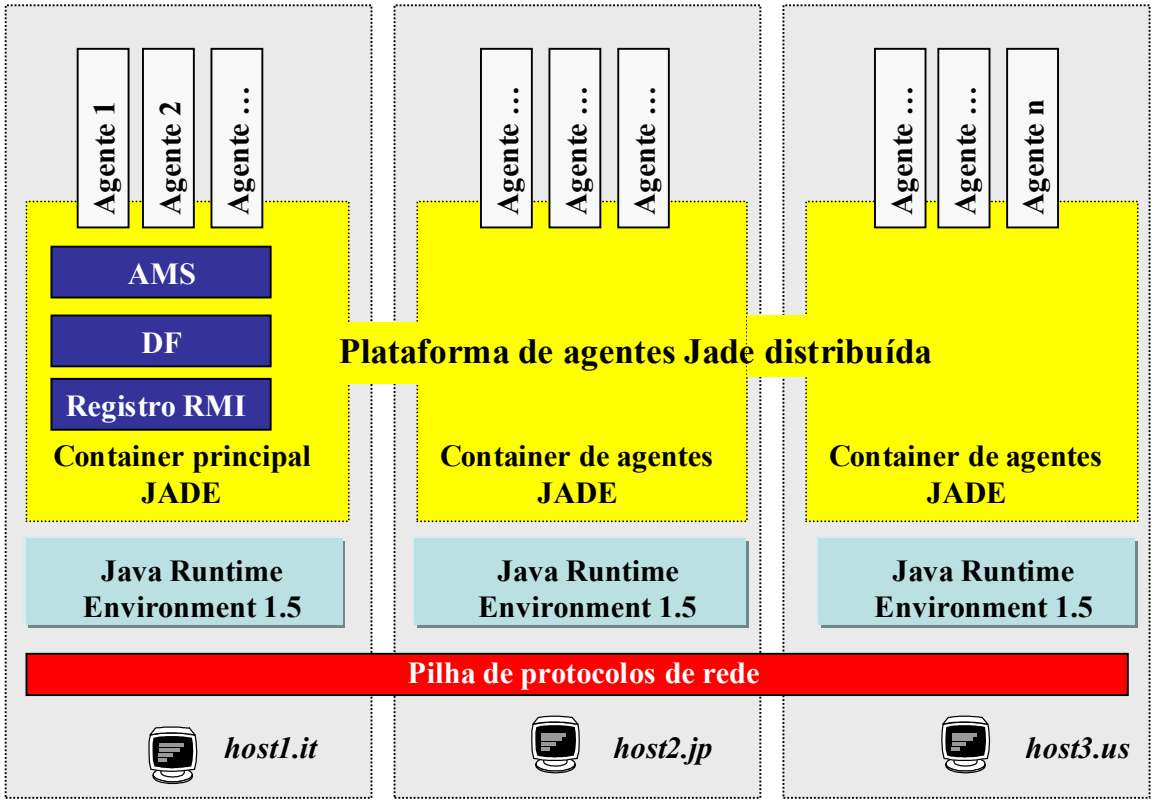


Figura 5.3 – Plataforma de agentes JADE distribuída entre vários *container's*

(Fonte: BELLIFEMINE et al., 2006a).

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.

5.1.3 Modelo de programação de agentes no arcabouço JADE

No contexto do arcabouço JADE, um agente pode ser entendido como um processo computacional que representa entidades autônomas e independentes, as quais são dotadas de uma identidade globalmente única, bem como da capacidade de comunicação com outros agentes de modo a cooperar visando atingir seus objetivos de projeto (BELLIFEMINE et al., 2006a).

Todavia, do ponto de vista de programação concorrente, um agente residente na plataforma JADE é um objeto ativo com um *thread* de controle interno, pois o arcabouço JADE adota o modelo de um *thread* por agente ao invés de um *thread* por tarefa, de modo a manter uma quantidade reduzida de *threads* necessários para executar a plataforma de agentes (BELLIFEMINE et al., 2006a).

Um agente é uma instância da classe *Agent* do arcabouço JADE, que consiste em uma classe básica que pode ser estendida ou especializada pelo programador para desenvolver suas próprias aplicações de agentes. Deste modo, esta instância herdará todas as características da classe *Agent,* responsáveis por realizar as interações necessárias com a plataforma de agente, entre as quais o registro, a configuração e gerenciamento remoto dos agentes, reduzindo de modo significativo o esforço de programação. Assim, o desenvolvedor fica livre para concentrar-se somente no modelo de tarefas que o agente deve desempenhar de acordo com seu projeto. Neste sentido, a classe *Agent* dispõe de um conjunto de métodos destinados a implementar um modelo de tarefas ou comportamentos de um agente, e uma visão detalhada destes métodos pode ser encontrada em Bellifemine et al. (2006a). Entretanto, cumpre destacar os principais métodos relacionados à implementação do modelo proposto:

- ***protected void setup()***: é um método protegido, destinado à inserção da codificação de inicialização de um agente. Este método pode ser sobrescrito pelo programador ou desenvolvedor de modo a fornecer os comportamentos planejados para o agente em questão, além de tarefas comuns de inicialização do agente como: o registro do agente no facilitador de diretórios (DF);
- ***public void addBehaviour*** (adicionar comportamento): responsável por adicionar um novo comportamento ao agente, o qual será executado de forma simultânea com outros comportamentos do agente. Este método, usualmente, é chamado dentro do método *setup()* para iniciar algum comportamento em particular, embora possa ser usado também para gerar comportamentos dinamicamente, os quais serão adicionados à fila de comportamentos ativos do agente. Por sua vez, o método *removeBehaviour()* realiza o processo de remoção de determinado comportamento do agente;

- ***public final void send*** (enviar mensagem): responsável por enviar uma mensagem na linguagem ACL (*Agent Communication Language*) (FIPA, 2002b) para outro agente. Este agente destino é especificado no campo *receiver,* da mensagem na qual um ou mais agentes podem ser definidos como destinatários;
- ***public final ACLMessage receive***(): recebe uma mensagem na linguagem ACL (*Agent Communication Language*) da fila de mensagens do agente. Este método não pode ser bloqueado, e retorna a primeira mensagem da fila caso ela possua mensagens, ou o valor *null* caso a fila esteja vazia.

Por outro lado, de acordo com a especificação FIPA 2000 (FIPA, 2002a), um agente, no tocante ao seu ciclo de vida, pode estar em um dos seis estados seguintes: Ativo, Desconhecido, Suspenso, Iniciado, Em trânsito ou Em espera, os quais são gerenciados pela plataforma JADE. Entretanto, a classe *Agent* dispõe de um conjunto de métodos destinados à mudança de estados em função das necessidades da aplicação. É importante observar que os agentes só têm permissão para executar suas tarefas quando estiverem no estado ativo.

5.1.4 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"[8], é uma 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 conhecimento específicas de maneira a potencializar a arquitetura interna destes agentes (FIGUEIRA e RAMALHO, 2000).

A figura 5.4 apresenta a arquitetura interna genérica de um agente no arcabouço proposto. Nesta figura é possível observar as várias tarefas do agente representadas por meio de uma coleção de comportamentos, cuja seqüência de execução é ordenada por meio de um escalonador interno da classe *Agent* (BELLIFEMINE et al., 2006a).

[8] do termo em inglês *Behaviour*

Este escalonador gerencia, de forma automática e transparente ao desenvolvedor, a seqüência de execução dos comportamentos mediante um algoritmo denominado circular *não-preemptivo* (ou 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., 2006a).

A figura 5.4 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 Language*), ela pode decidir quando e quais mensagens recebidas serão lidas.

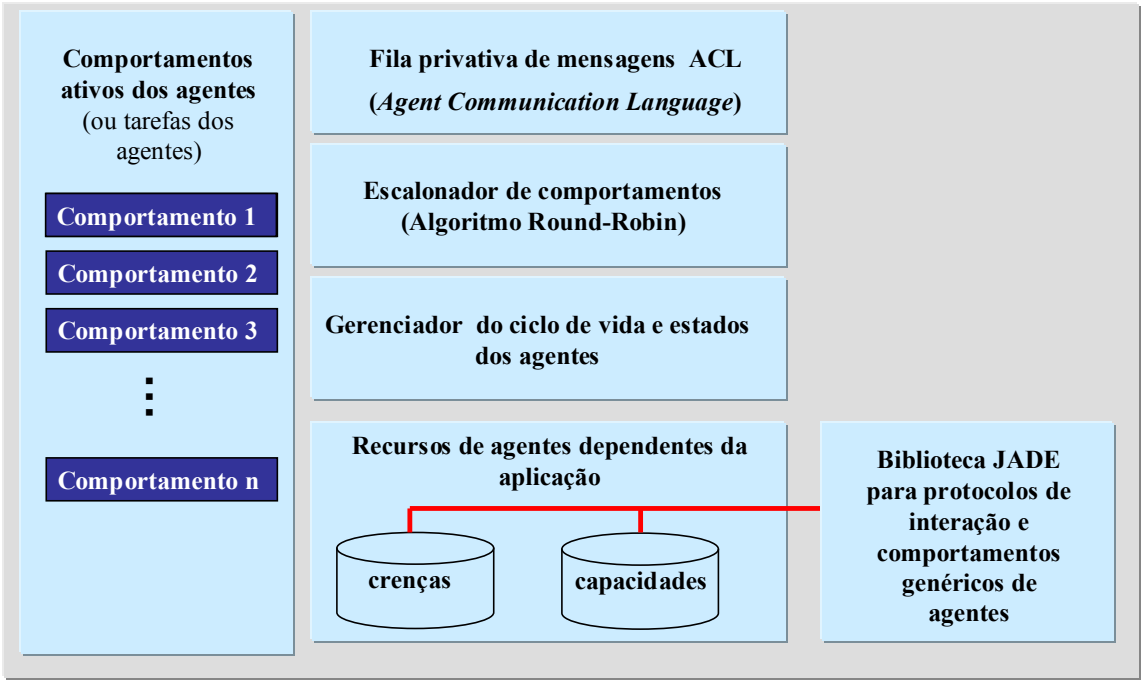


Figura 5.4 – Arquitetura interna genérica de um agente no arcabouço JADE

(Fonte: BELLIFEMINE et al., 2006a).

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 através deste 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 como mostra a figura 5.5, as quais podem ser especializadas de acordo com uma necessidade específica do agente em questão (BELLIFEMINE et al., 2006a).

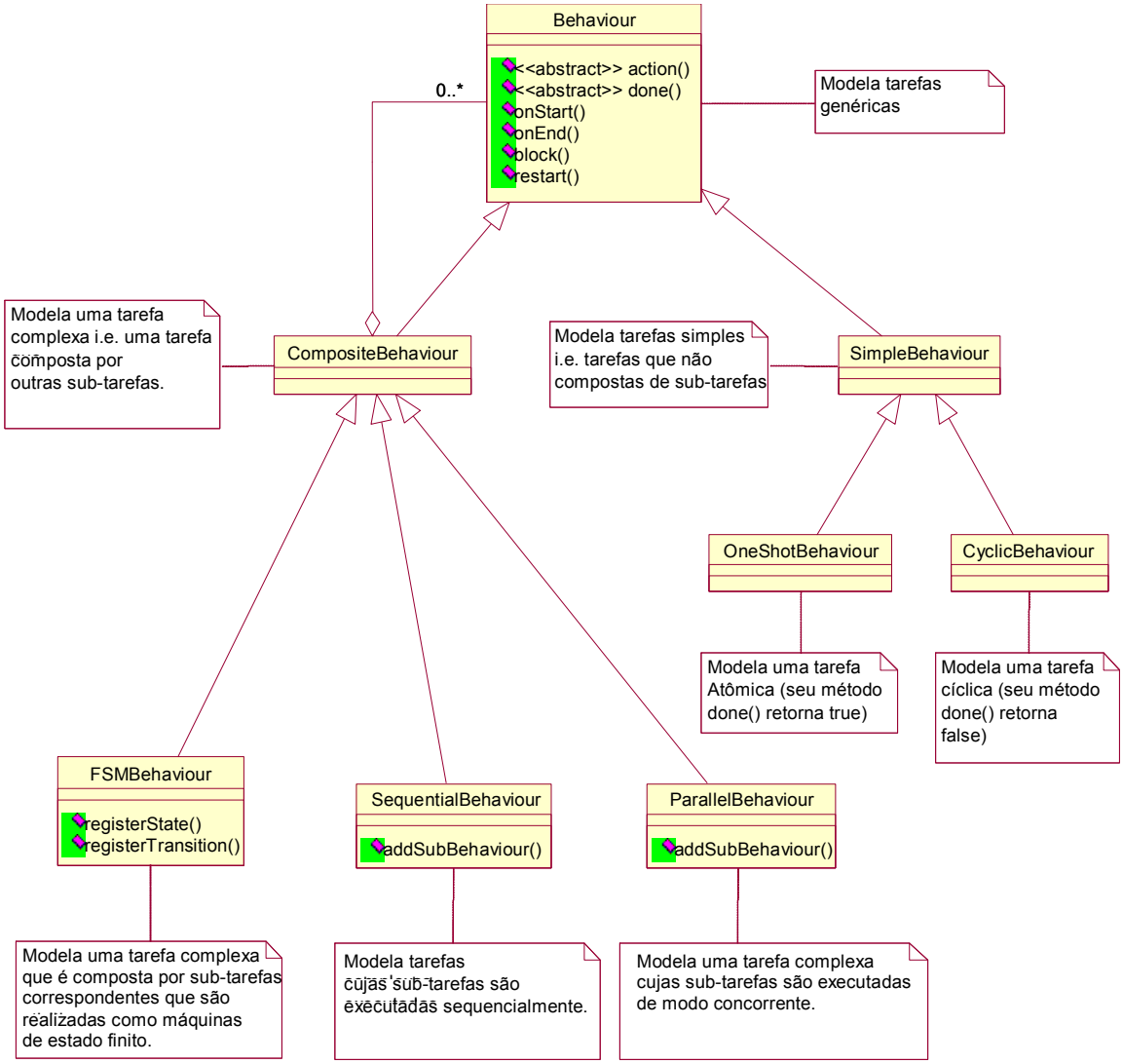


Figura 5.5 – Modelo UML da hierarquia da classe *Behaviour*
(Fonte: BELLIFEMINE et al., 2006a).

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.

Uma visão detalhada da classe *Behaviour* voltada para o desenvolvedor pode ser encontrada em Bellifemine et al. (2006a). Entretanto, cumpre destacar que a classe abstrata *SimpleBehaviour* pode ser utilizada para modelar comportamentos atômicos, isto é, aqueles comportamentos projetados para serem únicos, e que não podem ser interrompidos durante sua a execução. Neste sentido, esta classe é composta pelas seguintes subclasses:

- ***jade.core.behaviours.CyclicBehaviour***: define um comportamento atômico que deve ser executado sempre. Trata-se de uma classe abstrata que pode ser herdada a para criação de comportamentos cuja execução será contínua. A característica deste tipo de comportamento faz com que ele se repita indefinidamente, como se estivesse em um *loop* infinito, porque o método *done()* herdado da superclasse *Behaviour* sempre retorna o valor falso para comportamentos cíclicos;
- ***jade.core.behaviours.OneShotBehaviour***: modela uma tarefa atômica. Essa classe abstrata pode ser herdada para a criação de comportamentos para tarefas que precisam ser executadas apenas uma única vez e seu método *done()* retorna *true*. Ela tem como classe filha a classe *SenderBehaviour*;
- ***jade.core.behaviours.SenderBehaviour:*** é um comportamento do tipo *OneShotBehaviour* para envio de mensagens ACL. Essa classe encapsula o método *send*() como uma operação atômica. Seu funcionamento restringe-se a realizar o comportamento de enviar determinada mensagem ACL e logo em seguida ser finalizado. A mensagem ACL a ser enviada é passada como parâmetro no construtor da classe.

Tendo em vista o exposto acima, 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 e todas 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, ele 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., 2006a).

5.1.5 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 5.6 apresenta o modelo de comunicação instanciado no arcabouço JADE, que executa, de forma transparente ao desenvolvedor, as trocas de mensagens entre os agentes. Assim, no modelo de programação do arcabouço, a especificação das trocas de mensagens entre agentes ocorre pela instanciação de um objeto da classe *ACLMessage* responsável por codificar todas as mensagens de acordo com as especificações FIPA ACL (2002b).

De acordo com estas especificações, a estrutura da mensagem (FIPA ACL *message*) envolve um conjunto de um ou mais parâmetros de mensagem. Os parâmetros que devem ser especificados para uma comunicação efetiva entre os agentes pode variar de acordo com a

aplicação. Entretanto, alguns parâmetros são obrigatórios, tais como: o ato comunicativo (ou *performative*), o remetente, o destinatário e o formato de representação da mensagem.

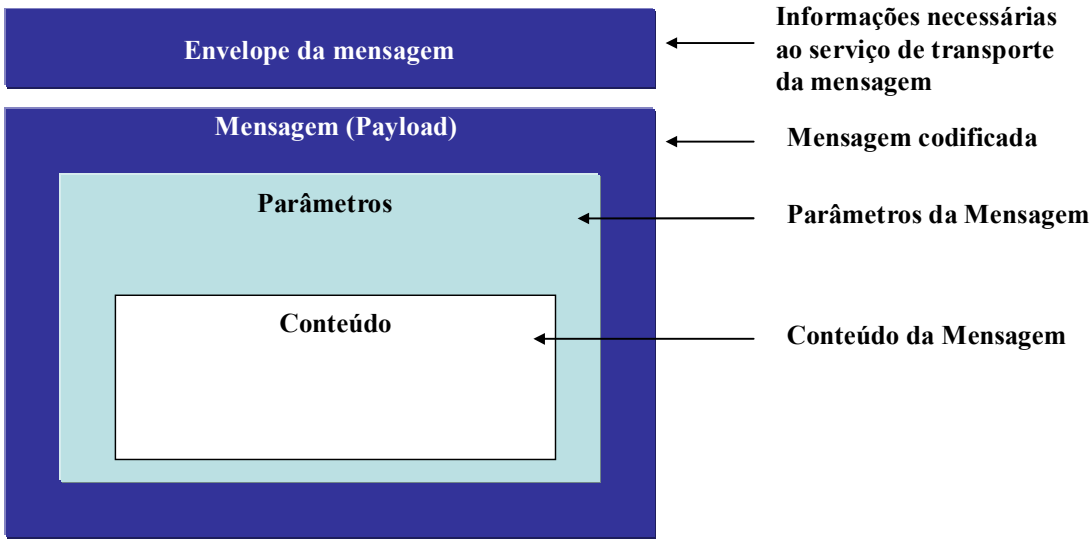


Figura 5.6 – Modelo de comunicação de acordo com as especificações FIPA-ACL (Fonte: FIPA, 2002b)

Cumpre observar que os parâmetros nativos da especificação FIPA-ACL são divididos em cinco categorias: tipo do ato comunicativo, participantes da comunicação, conteúdo de mensagem, descrição do conteúdo, e controle de conversação. A necessidade e definição destes parâmetros dependem da aplicação, e a relação completa de parâmetros pode ser obtida na especificação FIPA (2002b).

Os agentes interpretam a mensagem por meio dos atos comunicativos, e a lista completa de atos comunicativos previstos nas especificações FIPA-ACL e suas respectivas explicações podem ser encontradas em FIPA (2002b), entre os quais destacam se: (a) a performativa *Request*, que estabelece a comunicação entre um agente emissor e um receptor, onde o agente emissor solicita ao receptor que este execute algum serviço (usualmente um ato comunicativo), e o conteúdo da mensagem expressa qual serviço foi requisitado; (b) a performativa *Inform*, que é utilizada pelo emissor para informar o receptor que uma dada sentença é verdadeira, e o conteúdo da mensagem é a própria sentença ou proposição.

Bellifemine et al. (2006a) apresentam uma visão detalhada dos métodos disponíveis na classe *ACLMessage,*e aqui são destacados os métodos principais:

- ***public void addReceiver****(AID idAgente)*: método responsável pela adição do AID (*directory of agent identifiers*) de um agente como receptor ou destinatário da mensagem FIPA ACL, especificando o remetente e o destinatário da mensagem;

- ***public ACLMessage createReply()***: método responsável pela criação de uma nova mensagem FIPA ACL em resposta a uma determinada mensagem;
- ***public java.lang.String getContent()***: retorna uma *string* contendo o conteúdo da mensagem.
- ***public void setContent****(java.lang.String)*: define o conteúdo da mensagem a ser enviada.

É importante destacar que as especificações FIPA ACL não definem nenhum tipo de padrão para o conteúdo das mensagens trocadas pelos agentes, no entanto o seu uso no arcabouço JADE permite que a transmissão de mensagens seja efetuada com conteúdos envolvendo um conjunto de caracteres (*strings*) e, em especial, objetos Java do tipo *java.io.Serializable* (SUN DEVELOPER NETWORK, 2007).

5.1.6 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*)[9] através de uma rede. Esta funcionalidade permite a implementação do protótipo de laboratório do modelo formal usando-se a abstração de *container's* como *hosts* distribuídos geograficamente.
- A abstração do modelo de tarefas dos agentes no arcabouço JADE denominada como comportamento (do inglês *Behaviour*), que apresenta, entre outras vantagens, a possibilidade de integração com outros recursos de *softwares*, como por exemplo,

[9] O mecanismo RMI ou Chamada de Métodos Remotos permite o acesso a objetos em máquinas diferentes (HORSTMANN e CORNELL, 2001).

sistemas de raciocínio capazes de acessar e processar bases 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.*

5.2 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*[10], Interfaces de Programas Aplicativos RBC[11], 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 por meio da respectiva linguagem de programação, 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 semicompleta 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,

[10] Programa que controla a interação do usuário da ferramenta com o núcleo (*kernel*) do sistema CBR.
[11] Do termo em inglês *CBR API's - Application Programming Interfaces.*

bem como descrever as características principais da ferramenta selecionada para o presente trabalho.

5.2.1 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 no capítulo 4, 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[12]; (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[13].

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

---

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

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

- 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 que reduz significativamente os esforços de instanciação e o tempo de aprendizagem necessário para gerar novas aplicações.

5.2.2 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 idéia 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.

Dentro desta 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-GARCIA et al., 2005). Dentre os conceitos relacionados ao domínio de RBC tem-se: Casos, Base de Casos, Descrição do Casos, Solução do Caso, Função de similaridade, 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-GARCIA et al., 2005). 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 idéia 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.

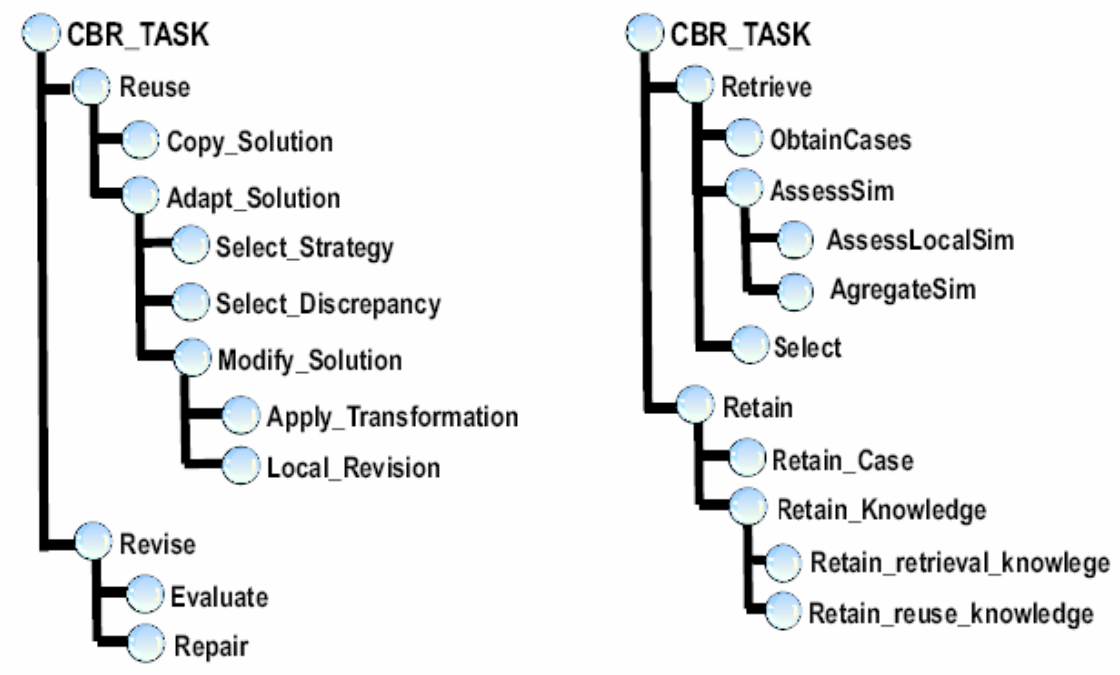


Figura 5.7 – Estrutura de tarefas na ontologia CBROnto do arcabouço jCOLIBRI

(Fonte: BELLO-TOMÁS et al., 2004)

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 5.7.

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 5.8 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.

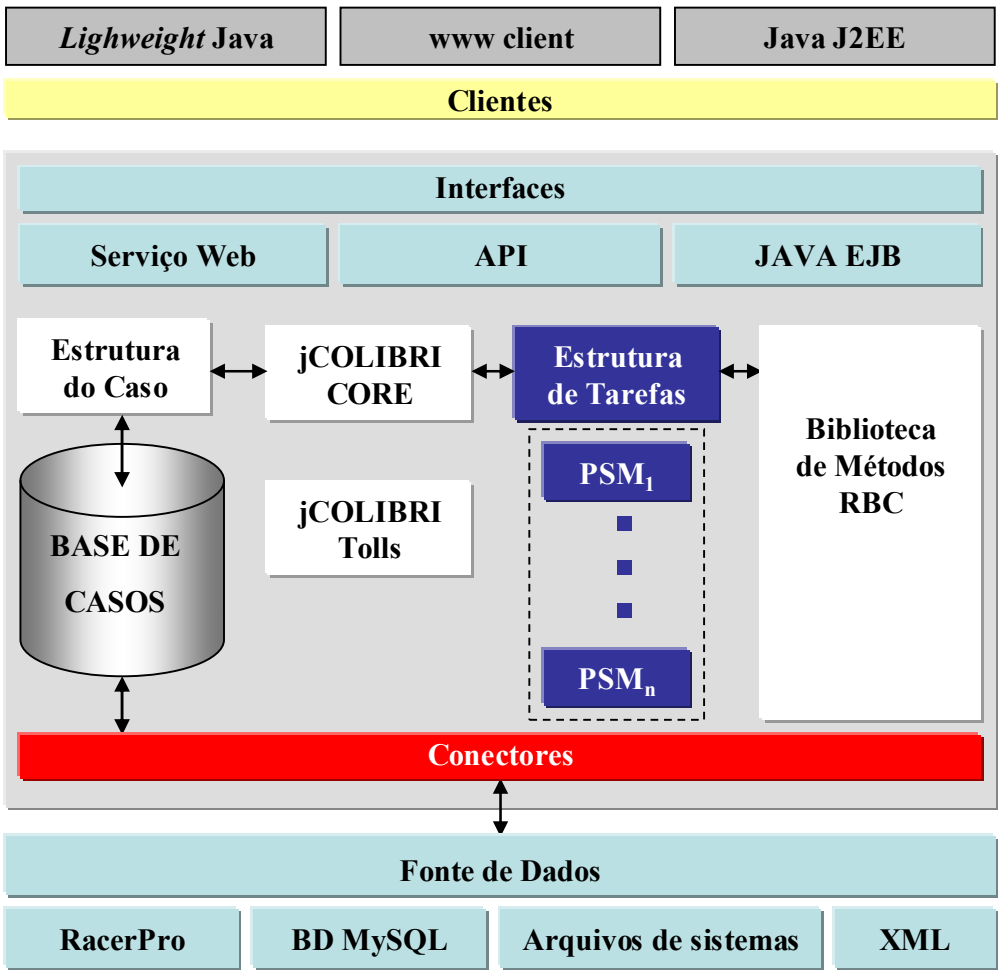


Figura 5.8 – Arquitetura geral do arcabouço jCOLIBRI

(Fonte: BELLO-TOMÁS et al., 2004).

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 5.9. 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, 2006); 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, 2006).

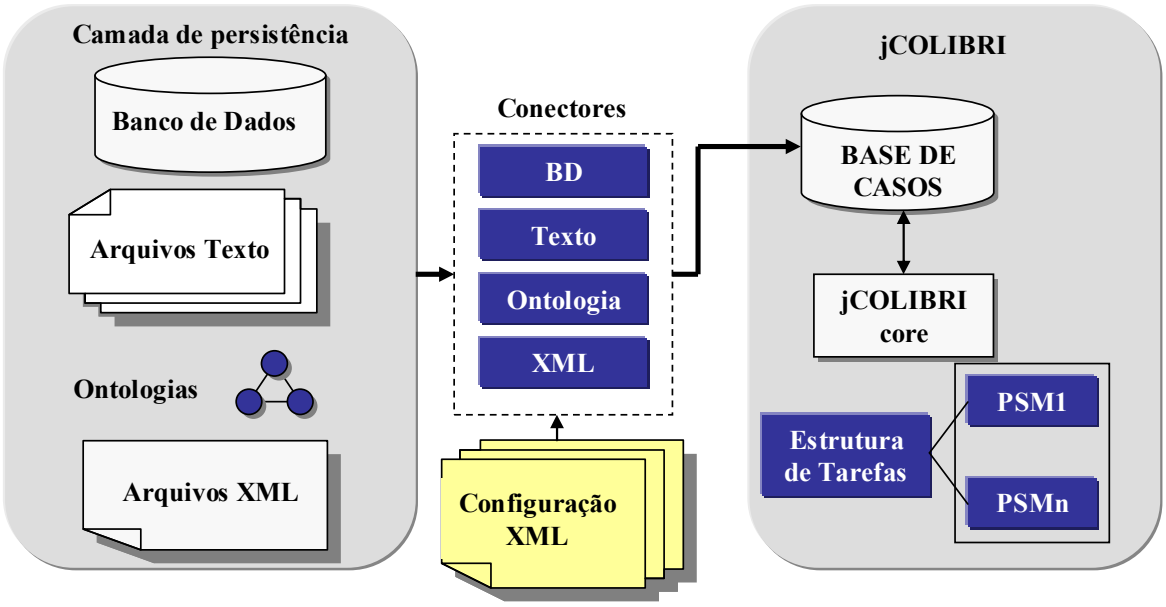


Figura 5.9 – Arquitetura de conectores disponíveis no arcabouço jCOLIBRI

(Fonte: BELLO-TOMÁS et al., 2004).

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 diagrama de classes escritas em linguagem JAVA, diagramas de seqüência 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), bem como em tutoriais e em ampla documentação técnica disponibilizada pelo Grupo de Aplicações de Inteligência Artificial da Universidade Complutense de Madri (GAIA, 2007).

5.2.3 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, é possível gerar um código completo em linguagem JAVA para disponibilizar a aplicação. Adicionalmente, é possível 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 implementaçao do modelo proposto neste trabalho de tese.

5.3 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.

5.3.1 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 corretude dos algoritmos usados nestes serviços de raciocínio e interfaces para acesso ao sistema.

Por sua vez, Gardiner et al. (2006) empreendem uma avaliação profunda dos sistemas de raciocínio em questão, a partir de um arcabouço de comparação automática de raciocinadores envolvendo as seguintes etapas: identificar todas as classes (*atomic concepts*) da ontologia, verificar a consistência da ontologia pela capacidade de satisfazer as definições de classes OWL (*concepts*) e classificar a taxonomia da ontologia.

Neste sentido, Gardiner et al. (2006) buscam testar, em especial: (a) a corretude 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*[14] (*Query Language for RDF*). Este sistema pode ser acessado através da Interface DIG[15] (*Description Logics Implementation Group*).

[14] 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/

[15] 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).

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

E, 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*[16]) para a linguagem de lógica de descrições $ALCQHI_{R+}$, 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 em função da 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

[16] (BAADER e SATTLER, 2001)

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.

5.3.2 Características do sistema RacerPro essenciais à implementação do modelo

O sistema RacerPro, 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 em relação a 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.

A gama de serviços de raciocínio e de recuperação de conhecimento disponíveis o sistema RacerPro pode também ser usado 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 proposto neste trabalho de tese.

5.4 EDITOR GRÁFICO PARA A IMPLEMENTAÇÃO DA ONTOLOGIA

Em primeiro lugar, o editor responsável pela implementação computacional da base de conhecimento ontológica, formalizada no capítulo 4, 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.

5.4.1 Processo de escolha do editor para implementaçao 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), OntoEdit (ONTOPRISE, 2003) e OilEd (BECHHOFER et al., 2001).

O estudo comparativo de Su e Ilebrekke (2002) empreendeu uma análise profunda dos editores a partir de seis perspectivas: qualidade física, qualidade empírica, qualidade sintática, qualidade semântica, qualidade semântica percebida e qualidade 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 *per se*, 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, 2007), cujo objetivo é integrar ambos os editores em uma plataforma única denominada Protégé-OWL, a qual une três paradigmas: *frames*[17], lógica de descrições e *RDF*[18], a partir das funcionalidades *plug-and-play* existentes no Protégé-2000 visando dar acesso a uma futura gama de outras ferramentas em código aberto.

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

No contexto deste trabalho de tese foi escolhido o editor Protégé-OWL (versão 3.2), 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

[17] 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* (onde os *slots* consistem de um conjunto de atributos de valores particulares denominados de facetas).

[18] 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/

permite a integração direta com o sistema de raciocínio RacerPro para a verificação da consistência da ontologia e classificação de taxonomias.

5.5 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 exaustiva destes recursos estão além do escopo deste trabalho, tais informações podem ser encontradas nas referências bibliográficas indicadas ao longo do texto.

Neste cenário, selecionou-se, em primeiro lugar, o arcabouço JADE (***J**ava **A**gent **DE**velopment 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, 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.

CAPÍTULO 6
IMPLEMENTAÇÃO DO MODELO BASEADO EM AGENTES PROPOSTO

Este capítulo é devotado às questões relacionadas ao desenvolvimento formal de um protótipo do modelo baseado em agentes a partir das especificações de projeto do modelo estabelecidas no Capítulo 4. Nesta perspectiva, apresentam-se 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*[1]. 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) em primeiro lugar 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) em seguida 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) por fim 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

[1] 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.

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.

6.1 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 (*directory of agent identifiers* - *AID*) de todos os agentes residentes na plataforma, e em seguida o agente é colocado no estado ativo.

E, 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.

6.2 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.

6.2.1 Desenvolvimento computacional do comportamento do agente

O modelo de programação adotado para o desenvolvimento dos agentes de interface RBC visa desenvolver computacionalmente um conjunto de tarefas iniciais diretamente no método denominado `setup` do agente de interface, mediante a extensão da superclasse `jade.core.Agent` do arcabouço JADE. Em seguida adicionar-se, um comportamento chamado de "principal", que responde pela interação entre o usuário e os agentes de recursos de conhecimento.

Neste sentido, a tarefa inicial implementada pelo método `setup` do agente de interface é destinada a realizar uma pesquisa junto ao Agente *Matchmaker* (ou neste protótipo o Agente Facilitador de Diretórios – DF, que é pré-programado no arcabouço). Esta pesquisa busca identificar todos os agentes de conhecimento que dispõem do serviço de raciocínio baseado em casos e que se encontram ativos na plataforma. A pesquisa busca obter os identificadores destes agentes para futuras comunicações ponto a ponto. Esta tarefa foi implementada a partir do método `search` da classe `jade.domain.DFService` disponível no arcabouço JADE.

6.2.2 Desenvolvimento computacional do comportamento principal do agente de interface

O agente de interface RBC foi desenvolvido a partir de um comportamento principal denominado *RequestPerformer*, que foi implementado pela extensão da classe `jade.core.behaviours.Behaviour` (BELLIFEMINE et al., 2006a).

Em termos metodológicos, a figura 6.1 apresenta o diagrama de seqüência AUML[2] que modela a interação entre o usuário, o agente de interface PFMEA e um agente de recurso PFMEA ativo na plataforma, indicando a abrangência do comportamento implementado pela extensão da classe original.

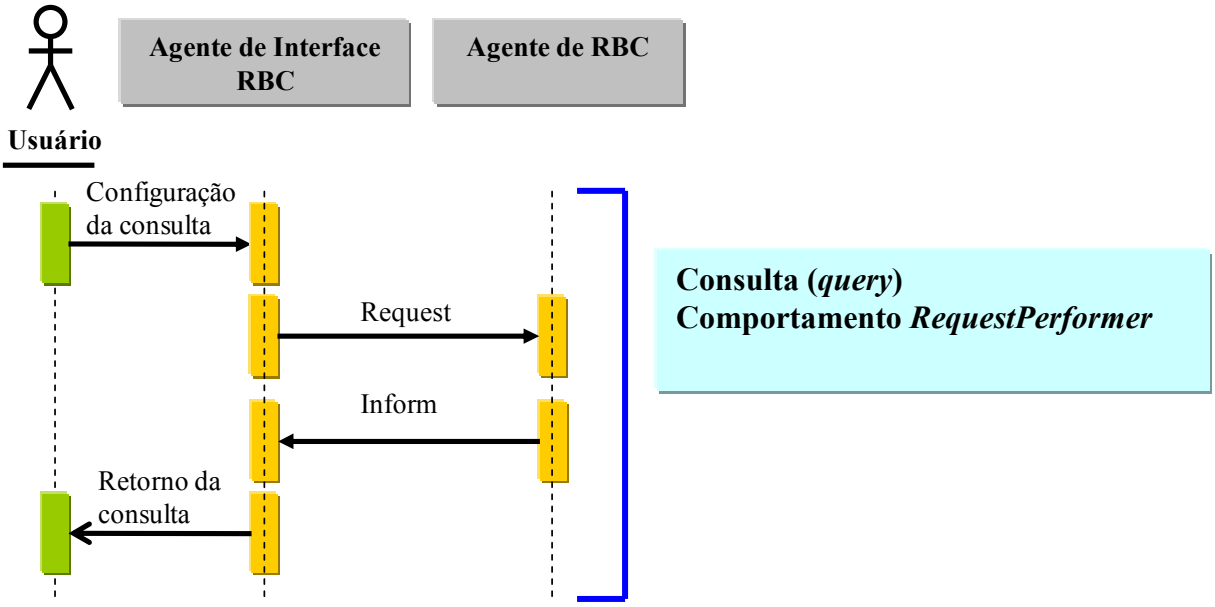


Figura 6.1 – Diagrama de seqüência AUML e comportamentos do agente de interface.

Este comportamento está em concordância com as especificações de projeto do modelo, pois os agentes foram projetados para permitir consultas (*queries*), onde 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 6.2. 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 respectivos pesos, os quais podem ser atribuídos de forma personalizada pelo usuário em função de suas percepções e necessidades de recuperação.

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

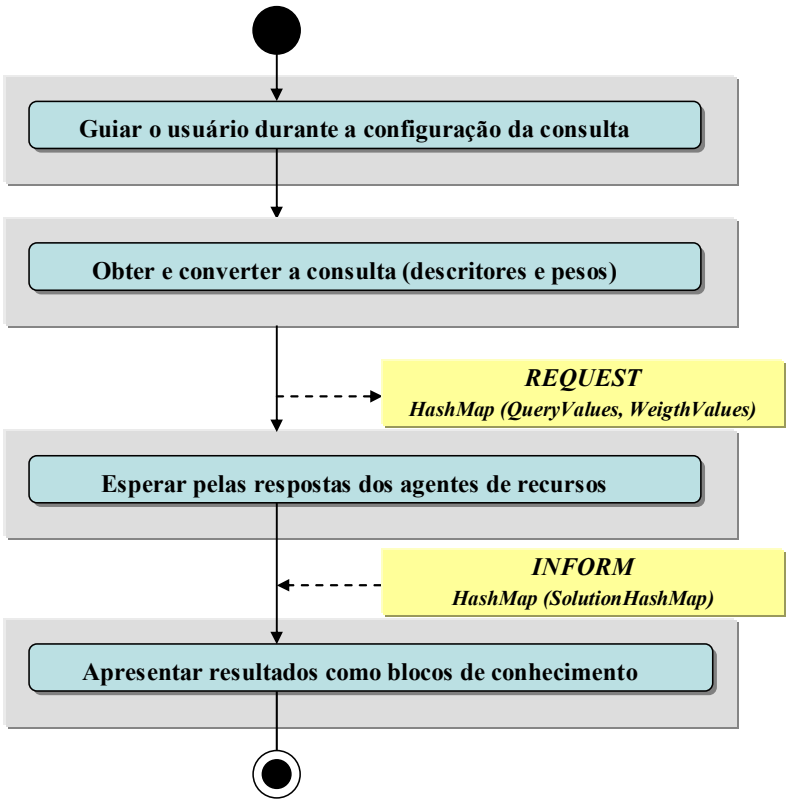


Figura 6.2 – Modelo de tarefas do comportamento principal dos agentes de interface RBC.

Neste protótipo, estes descritores estão relacionados, em particular, a processos de moldagem por injeção de termoplásticos, definidos na estrutura conceitual do caso de não-conformidades estabelecida no capítulo 4. Entretanto, ressalta-se que outras instâncias desta família de agentes (para outros tipos de processos) podem ser incluídas no modelo seguindo a mesma linha de comportamento apresentada nesta subseção.

Em relação à estrutura conceitual do caso de não-conformidade, destaca-se que o mesmo envolve 114 descritores organizados em nove categorias, as quais são; descrição, classificação e condições de contorno da não-conformidade; descritores do material de moldagem; descritores da peça moldada; descritores do projeto do molde; descritores das temperaturas de moldagem; descritores de pressões de moldagem; descritores de tempos de moldagem; descritores de seqüência e movimentos, e descritores da máquina.

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*[3] na forma de objetos Java tipo *HashMap* mediante a instanciação da classe `java.util.HashMap`, cujos detalhes são apresentados na subseção 6.2.4.

Deste modo, o agente de interface pode mobilizar 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 6.3 sintetiza o ato comunicativo.

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 6.3 – Síntese da comunicação entre o agente de interface e os agentes de recursos.

Então, o agente de interface entra em modo de espera de acordo com o previsto em seu comportamento (figura 6.2), aguardando por mensagens dotadas do ato comunicativo ***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; o bloco de conhecimento correspondente aos 10 descritores relativos à solução sugerida, bem como os 3 descritores referentes aos resultados; os valores dos 114 descritores armazenados com o este caso e usados para o cálculo 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 inteligível ao usuário por meio de blocos de conhecimento disponibilizados na interface gráfica.

[3] 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.

6.2.3 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 voltado para a apresentação dos respectivos resultados da consulta, como mostra a figura 6.4.

O primeiro conjunto de janelas gráficas (figura 6.4) é organizado em nove abas, as quais correspondem às nove categorias de descritores de uma não-conformidade estabelecidas no capítulo 4. Cada uma destas abas foi projetada para guiar o usuário durante a configuração de uma consulta, de modo a descrever o estado do processo no momento em que ocorreu a não-conformidade considerando a estrutura conceitual que descreve um caso de não-conformidade.

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 6.4. 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 6.4.

A configuração a partir de caixas tipo *ComboBox* visa assegurar 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 respaldados na literatura da área.

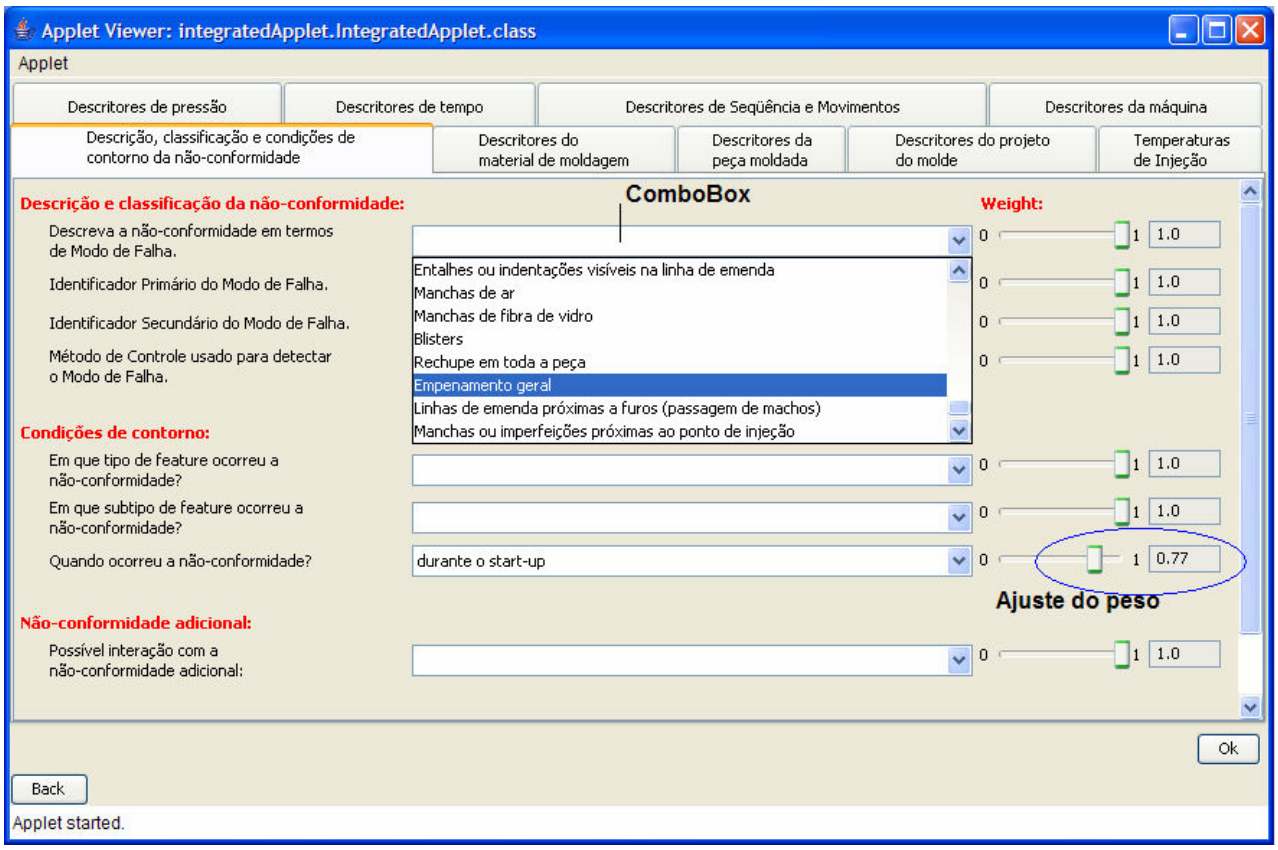


Figura 6.4 – Janela para configuração dos descritores de entrada da interface gráfica do agente.

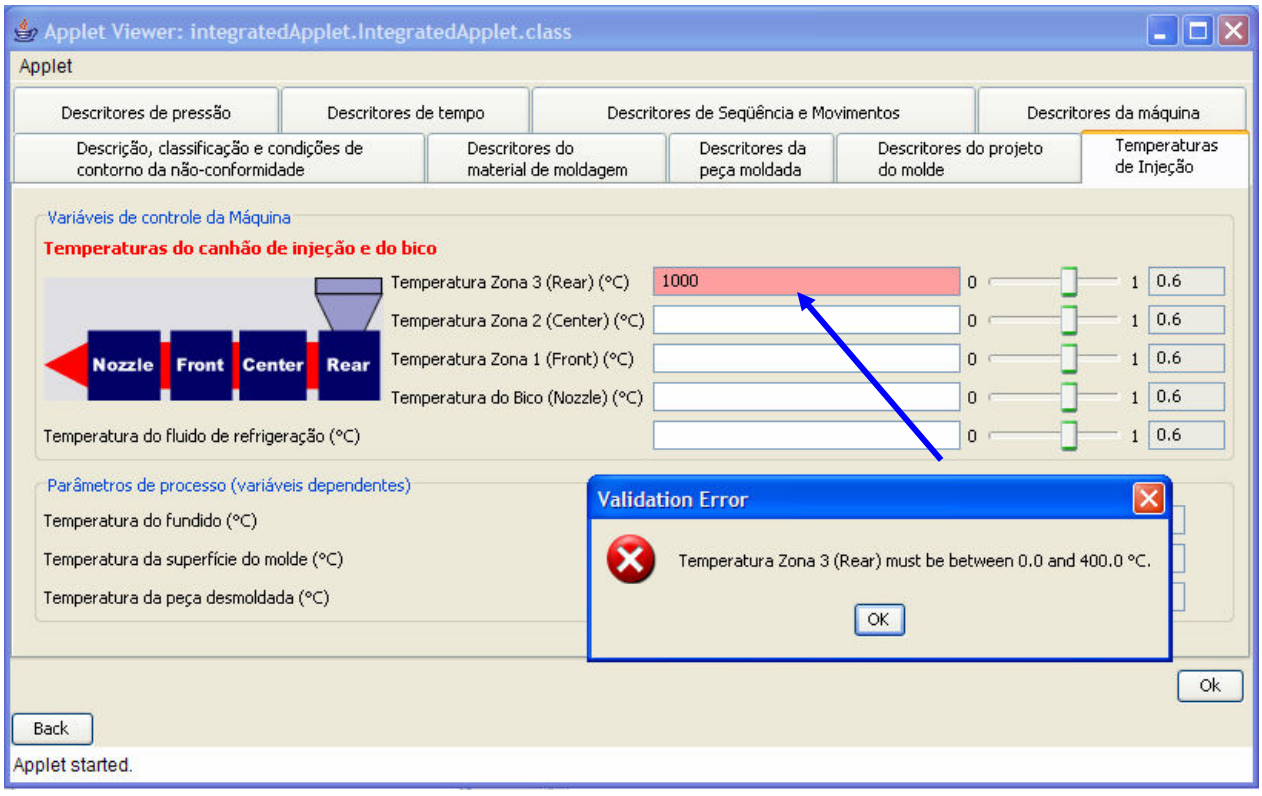


Figura 6.5 – Validação de dados numéricos na interface gráfica do agente.

Um outro aspecto importante é que a estratégia de configuração executa a validação de todos os dados numéricos considerando um intervalo pré-definido, o qual é usado para o cálculo da medida de similaridade local entre descritores numéricos (como discutido no capítulo 4), e isto é mostrado na figura 6.5.

Por sua vez, o segundo conjunto de janelas gráficas, mostrado na figura 6.6, 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.

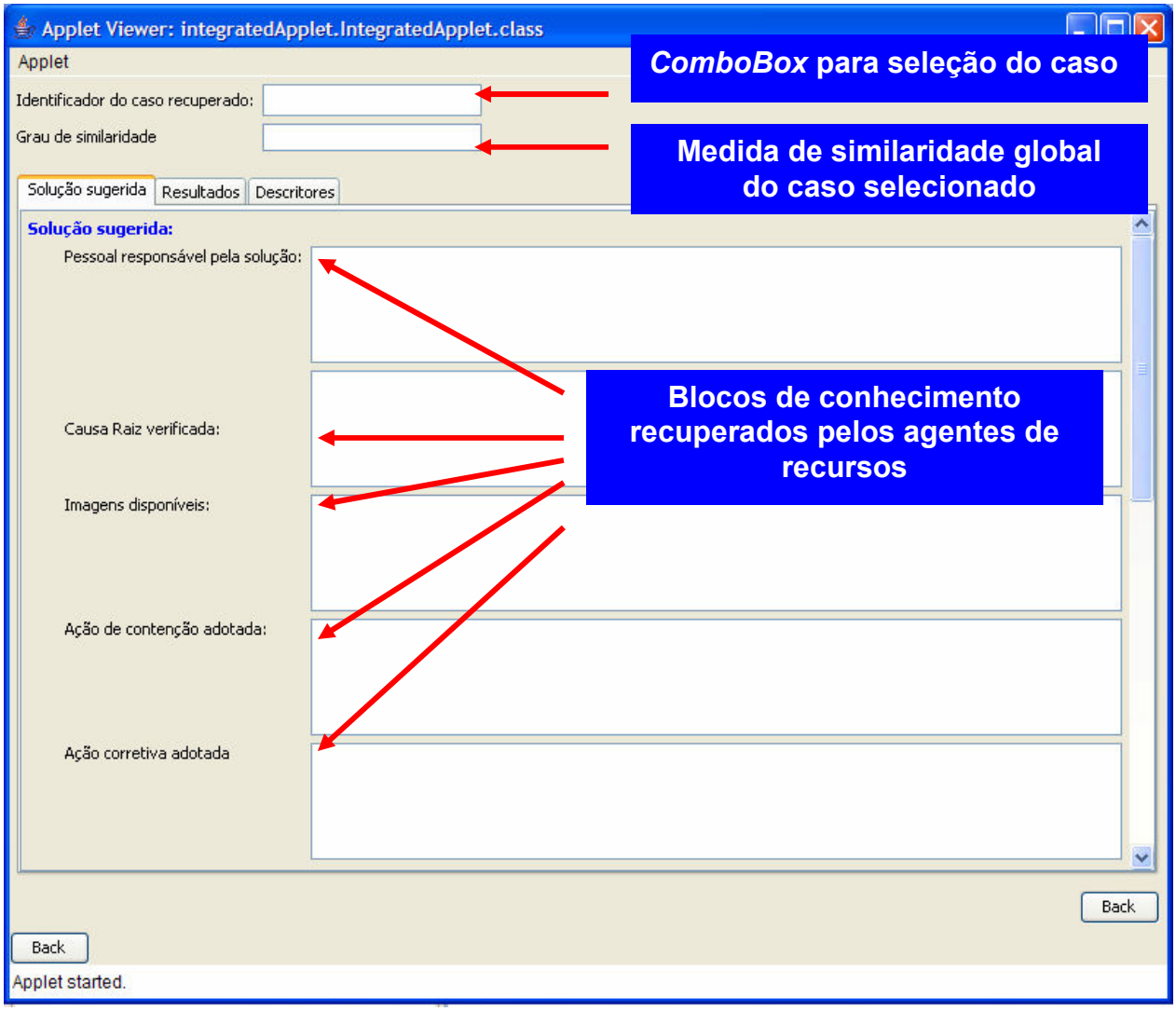


Figura 6.6 – Janela de apresentação dos casos mais similares recuperados.

A figura 6.6 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. Deve-se observar que estas abas correspondem à estrutura conceitual de um caso de não-conformidade, a qual foi apresentada no Capítulo 4.

Para visualizar os casos mais similares, o usuário deve selecionar um caso em particular por meio da caixa tipo *ComboBox* que apresenta a lista de identificadores dos casos recuperados. E 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 estratégia de configuração permite ao usuário a criação de uma consulta totalmente personalizada a partir de descritores de entrada validados de forma transparente. Além disso, a estratégia implementada permite a configuração de consultas mesmo que muitos dos descritores de entradas não sejam preenchidos (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 diferenciados visando ajustar o cálculo da medida de similaridade global.

Cumpre observar que, visando manter a clareza do texto desta subseção, a apresentação dos conjuntos de janelas gráficas destinadas à configuração da consulta e à apresentação dos casos mais similares são apresentadas no capítulo 8.

6.2.4 Mensagens FIPA ACL usadas para interação social dos agentes

O conteúdo das mensagens FIPA ACL trocadas pelos agentes em questão representa um aspecto fundamental nas ações a serem realizadas pelo agente de recursos de acordo com o modelo de comportamento adotado.

Em primeiro lugar, é importante observar que o conteúdo das mensagens FIPA ACL, que envolvem os atos comunicativos ***Request*** e ***Inform***, trocadas entre o agente de interface e os agentes da classe de Raciocínio Baseado em Casos, é expresso por meio de tabelas ou mapas *hash* na forma de objetos da classe `java.util.HashMap`, cujos detalhes podem ser encontrados em *Sun Developer Network* (2007).

A escolha da alternativa de configuração do conteúdo da mensagem por meio de mapas *hash* se deve, em especial, às necessidades de comunicação, as quais implicam na troca de mensagens envolvendo grande volume de dados. Gera-se um grande volume de dados devido ao fato que o ato comunicativo tipo ***Request*** deve conter informações sobre os todos os descritores de uma não-conformidade e seus respectivos pesos.

Deve-se destacar que estes dados serão usados pelos agentes de recursos RBC no cálculo das medidas de similaridade e que, por sua vez, o ato comunicativo tipo ***Inform*** deve conter tanto os descritores dos casos mais similares recuperados e seus respectivos graus de similaridade quanto as soluções sugeridas e resultados. Para esta finalidade, foi usada a classe `jade.lang. acl.ACLMessage`, pré-implementada no arcabouço JADE, que dispõe dos métodos `setContentObject()` e `getContentObject()`, sendo que o primeiro é destina-se ao suporte computacional do processo de "serialização"[4] de objetos Java e sua posterior transmissão como conteúdo das mensagens, enquanto o segundo incumbe-se do processo inverso (ou "desserialização") do conteúdo da mensagem de um conjunto de *Bytes* para um objeto Java.

6.3 AGENTES DE RECURSOS DE CONHECIMENTO RBC

De acordo com as especificações de projeto do modelo proposto, os agentes de Raciocínio Baseado em Casos são, em essência, agentes de recursos de conhecimento dotados de um modelo de tarefas que engloba as tarefas e métodos de raciocínio baseado em casos. Estes métodos são destinados à recuperação de conhecimento a partir de uma estrutura de caso codificada em uma base de conhecimento específica.

Dentro desta perspectiva, esta subseção apresenta o desenvolvimento computacional do modelo de tarefas dos agentes de recursos RBC, bem como o formato e conteúdo das mensagens FIPA ACL projetadas para as interações sociais e comunicação entre os agentes.

6.3.1 Desenvolvimento computacional do comportamento do agente RBC

O conjunto de atividades iniciais implementadas no método `setup` destina-se a realizar a atividade de registro de cada um dos agentes RBC ativos plataforma de agentes (ou na organização multiagente) na base de conhecimento do agente *Matchmaker* (ou agente DF) que responde pelo serviço de "páginas amarelas" da plataforma. Assim, são registradas a descrição dos agentes e o serviço oferecido como um subdomínio particular destas "páginas amarelas" usando-se os métodos próprios da classe `jade.domain.DFService`, cujos detalhes podem ser encontrados em Bellifemine et al. (2006a).

[4] O processo de serialização de objetos Java (do inglês *Serializable)* visa converter a representação de um objeto em memória para uma seqüência de bytes que pode ser associada a um meio de persistência física.

6.3.2 Comportamento principal dos agentes de recursos

O comportamento principal do agente é implementado computacionalmente pela extensão da classe `jade.core.behaviours.CyclicBehaviour`, que modela um comportamento que deve ser mantido em execução contínua, como se estivesse em *loop* (BELLIFEMINE et al., 2006a). Este modelo de comportamento, em particular, está em conformidade às especificações de projeto dos agentes de recursos, os quais foram projetados para esperar, continuamente, por mensagens do tipo FIPA ACL com o ato comunicativo ***Request***.

Este ato comunicativo, por definição, estabelece a comunicação entre o emissor, no caso o agente de interface, e o receptor ou o agente de recursos em questão, onde o agente emissor solicita ao receptor que este execute um serviço de raciocínio baseado em casos específico. Neste cenário, o comportamento principal dos agentes da classe RBC modela o conjunto de atividades relacionado à estrutura de tarefas e métodos RBC de acordo com a ontologia *CBROnto* do arcabouço jCOLIBRI (BELLO-TOMÁS et al., 2004). Este conjunto de atividades é implementado computacionalmente a partir da especialização, deste arcabouço, bem como pela extensão da classe gerada pela especialização além das classes do próprio arcabouço jCOLIBRI.

A figura 6.7 apresenta a arquitetura do arcabouço jCOLIBRI e sua respectiva estrutura de tarefas e métodos de RBC já instanciada, bem como o conector selecionado e a fonte de dados estabelecida para a persistência física dos dados, no caso o sistema de gerenciamento de banco de dados relacionais de código aberto MySQL® (MYSQL, 2006).

Nesta figura, a estrutura do caso compreende os 114 descritores que serão usados como indexadores do caso e que representam um caso de solução de não-conformidades relacionadas aos processos de moldagem por injeção de termoplásticos conforme definido no capítulo 4.

A cada um dos descritores representados na estrutura do caso associam-se parâmetros como: o nome do descritor, tipo de dado (p.ex. *string, integer, double, etc.*) e, em especial, a definição de um tipo de medida de similaridade local estabelecida também de acordo com as especificações.

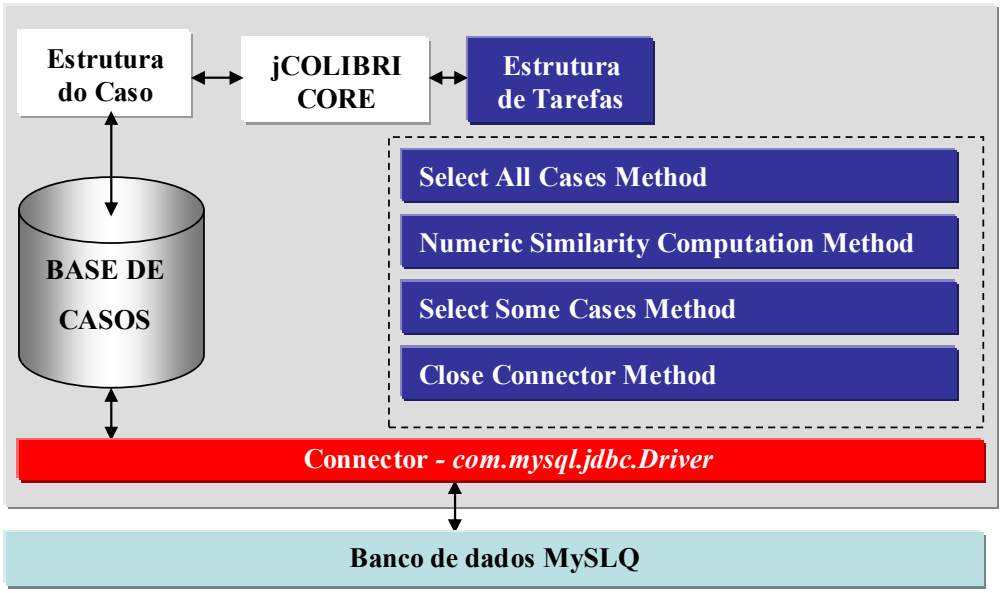


Figura 6.7 – Arquitetura jCOLIBRI com a estrutura de tarefas e métodos RBC instanciada

Na figura 6.7 destaca-se também a configuração do conector, que é responsável pela conexão e mapeamento da estrutura do caso com relação às tabelas do sistema de gerenciamento de banco de dados relacionais MySQL®, que envolve dois aspectos distintos. O primeiro aspecto diz respeito às configurações para acesso ao sistema de gerenciamento, tais como: definição do *driver*[5] para conexão, do sub-protocolo de comunicação, do *host* e da porta de acesso.

E o segundo aspecto refere-se ao mapeamento entre os parâmetros dos descritores definidos na estrutura do caso e as colunas das tabelas do sistema de gerenciamento de banco de dados criadas especificamente para esta finalidade.

Por sua vez, a figura 6.8 apresenta o comportamento principal dos agentes de raciocínio baseado em casos destacando as atividades executadas no âmbito deste comportamento, tendo como referência as tarefas e métodos de RBC instanciadas a partir do arcabouço jCOLIBRI.

Neste cenário, o comportamento principal do agente inicia com a recepção da mensagem com o ato comunicativo ***Request*** enviada pelo agente de interface e tendo como conteúdo os objetos Java tipo *HashMap* denominados *QueryValues* e *WeightValues*. Estes objetos referem-se aos descritores da não-conformidade sob consulta e aos pesos atribuídos a cada um destes descritores, de forma personalizada, pelo usuário responsável pela consulta.

[5] O *driver* é o componente de software responsável por converter as chamadas JDBC (Java Database Connectivity) diretamente para o protocolo do banco de dados MySQL, lembrando que JDBC é um conjunto de classes e interfaces escritas em Java usadas para enviar instruções no formato SQL para um sistema de gerenciamento de banco de dados.

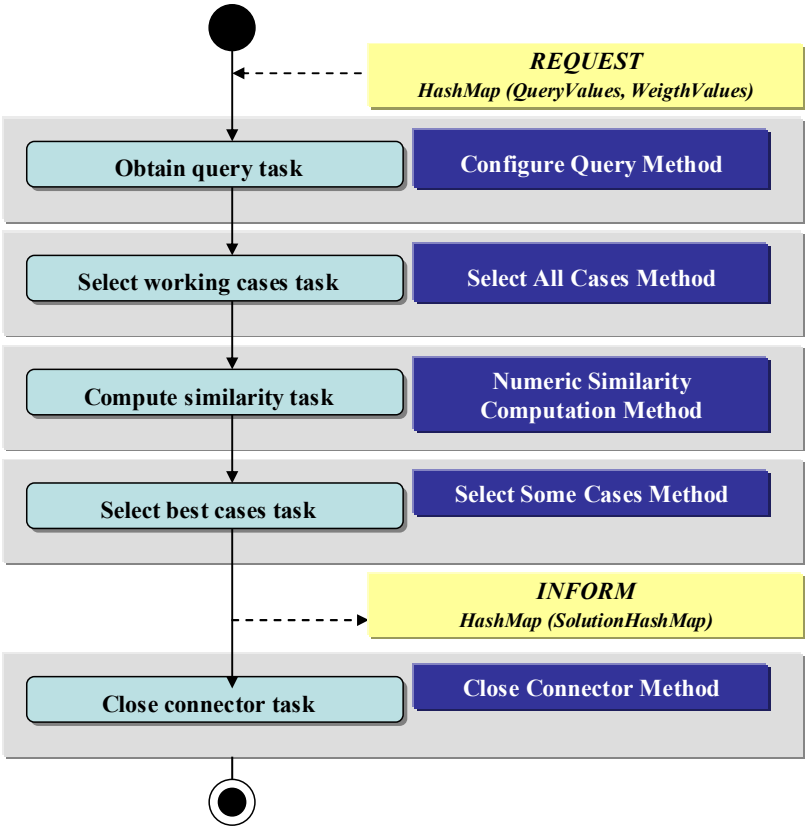


Figura 6.8 – Comportamento dos agentes de RBC.

A subtarefa de raciocínio baseado em casos *Obtain query task* é resolvida pela execução do método de RBC *Configure Query* que, por sua vez, é implementado do ponto de vista computacional pela classe `jcolibri.method.ConfigureQueryMethods`. Esta classe é responsável por carregar a estrutura do caso e receber dinamicamente as consultas (*queries*). Neste sentido, ressalta-se que esta classe foi estendida para tratar as consultas recebidas através do objeto *HashMap QueryValues*.

A subtarefa de raciocínio baseado em casos *Select working cases task* refere-se diretamente à tarefa de recuperação, em termos do ciclo RBC, e é resolvida pela execução do método de RBC *Select All Cases*, que do ponto de vista computacional é implementado pela classe `jcolibri. method.SelectAllCasesMethods.` Esta subtarefa, em síntese, é responsável por selecionar e carregar os casos prévios de solução de não-conformidades a serem manipulados de acordo com a estrutura do caso a partir do sistema de gerenciamento de banco de dados MySQL® usando o conector definido.

Por sua vez, a subtarefa de raciocínio baseado em casos *Compute similarity task* é resolvida pela execução do método RBC *Numeric Similarity Computation*, que ponto de vista computacional é implementado pela classe `jcolibri.method. NumericSimComputation.`

Esta atividade é responsável por executar os cálculos das medidas de similaridade, inicialmente, relativos à similaridade local, e em seguida à similaridade global, em função da qual são selecionados os casos mais similares. É importante observar também que esta classe, bem como outras classes relacionadas, foram estendidas para permitir a atualização dos pesos para o cálculo da similaridade global através do objeto Java *HashMap WeightValues*.

A subtarefa de raciocínio baseado em casos *Select best cases task* é resolvida pela execução do método RBC que é implementado ponto de vista computacional pela classe `jcolibri.method.SelectSomeCasesMethod`, cuja função é selecionar os três casos mais similares em função de suas medidas de similaridade.

Destaca-se também que esta classe foi estendida de modo a criar três objetos Java tipo *HashMap* para cada um dos casos recuperados, que serão encapsulados em um vetor denominado *solutionHashMap* e enviados como conteúdo da mensagem FIPA ACL com o ato comunicativo ***Inform***, no qual está incluída a medida de similaridade entre o novo caso e o caso recuperado, além dos descritores de não-conformidade armazenados com o caso recuperado, as soluções sugeridas para o tratamento da nova não-conformidade e os resultados esperados.

6.4 AGENTES DE INTERFACE PARA PFMEA

De acordo com as especificações de projeto do modelo, o agente de interface para recuperação de conhecimento decorrentes da aplicação do método PFMEA (ou agentes de interface PFMEA) tem por objetivo 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.

6.4.1 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 obter 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.

6.4.2 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., 2006a).

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 6.9 apresenta o diagrama de seqüência 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.

Nesta figura 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 6.10.

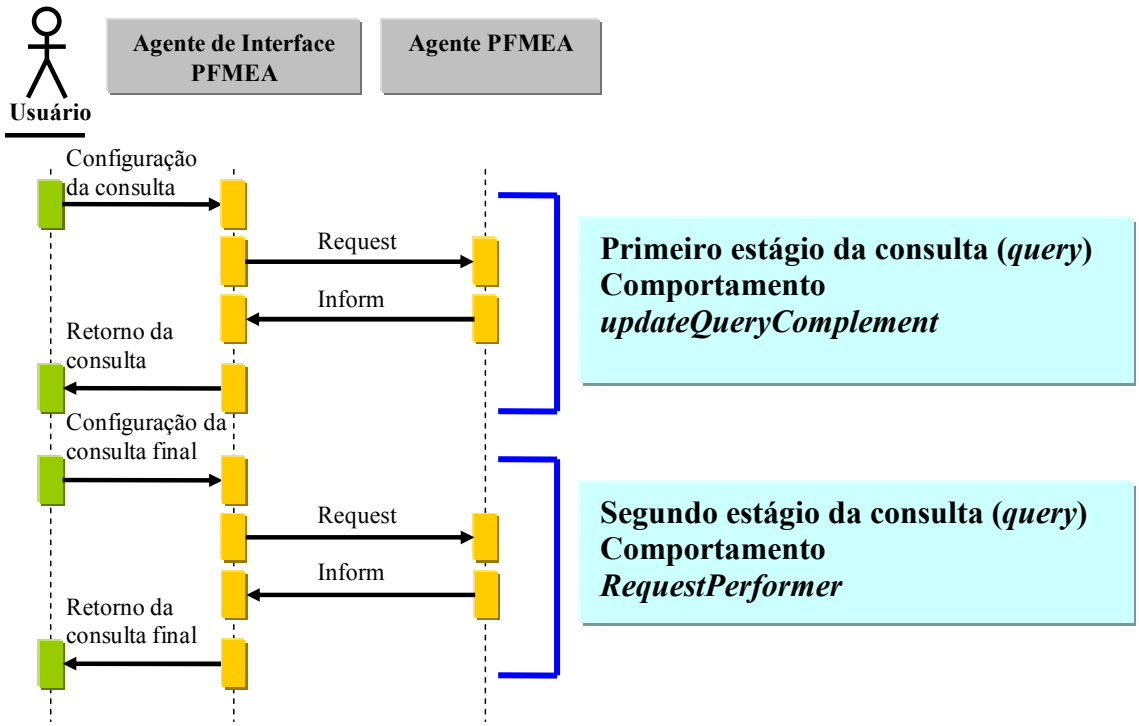


Figura 6.9 – Diagrama de seqüência AUML e comportamentos do agente de interface

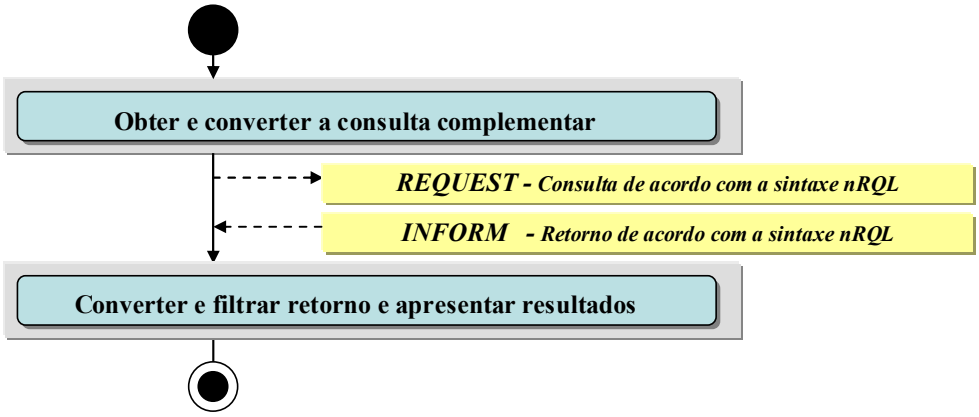


Figura 6.10 – Comportamento *updateQueryComplement* do agente de interface.

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 implementadas 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 inteligível ao usuário para que este possa realizar a consulta.

A janela gráfica do usuário apresentada na figura 6.11 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. Obviamente, a mesma estratégia pode ser aplicada para frases em língua portuguesa.

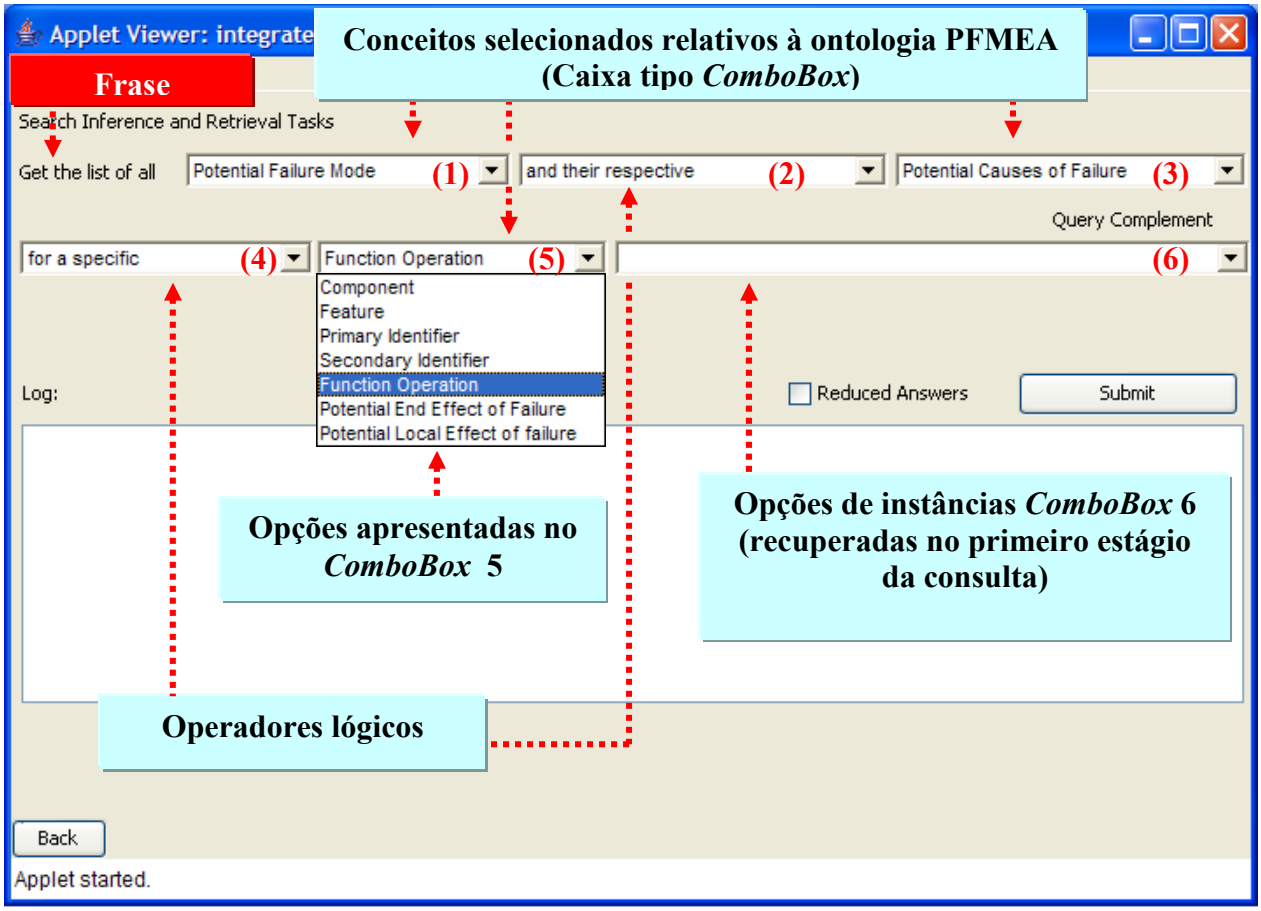


Figura 6.11 – Janela gráfica para configuração da consulta complementar.

É importante destacar que a delimitação da frase (***Get the list of all ...***) se dá pelo processo de escolha seqüencial dos conceitos da ontologia PFMEA, os quais são apresentados ao usuário como opções nas diversas caixas tipo *ComboBox* ao longo da estrutura da frase. Os detalhes relativos ao processo de configuração são apresentados no Capítulo 8 durante a etapa de verificação das funcionalidades do protótipo do modelo.

Neste trabalho de tese, os métodos responsáveis por converter a consulta para um formato que permita seu compartilhamento entre os agentes, consistem em converter a consulta, configurada pelo usuário, em um conjunto de caracteres (*strings*) concatenados na forma de uma "linha de comando" de acordo com sintaxe da linguagem nRQL[6] através da qual os agentes serão capazes de decodificar e analisar sintaticamente o conteúdo das mensagens.

A figura 6.12 mostra o resultado da conversão de uma consulta complementar para a sintaxe nRQL, no exemplo observa-se no cabeçalho da consulta o nome atribuído a uma variável, no caso (FUNCTION_OPERATION), de acordo com a escolha do usuário no *ComboBox* 5, bem como o conceito da ontologia PFMEA no corpo da consulta, no caso (*OperationFunction,* relacionado às funções de operações de manufatura) que contêm as instâncias a recuperar.

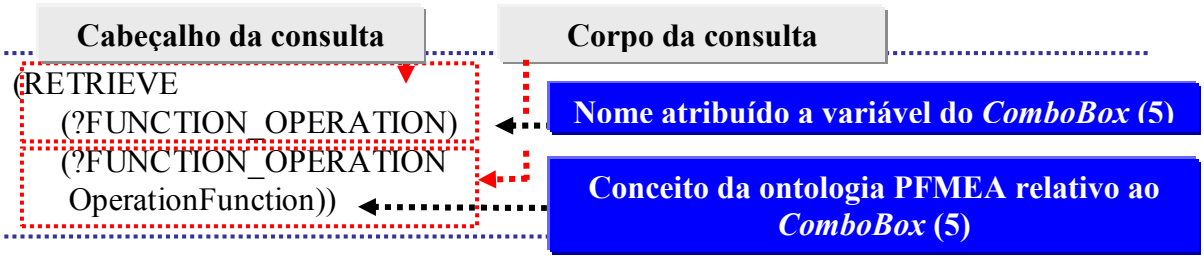


Figura 6.12 – Resultado da conversão da consulta complementar para a sintaxe nRQL.

Em seguida, a atividade utiliza a classe `jade.lang.acl.ACLMessage` pré-implementada no arcabouço JADE, e o métodos `setContent()`, o qual oferece suporte computacional ao processo transmissão do conteúdo das mensagens FIPA ACL na forma de um conjunto de caracteres (*strings*), como discutido no Capítulo 5.

Deste modo, o agente de interface mobiliza os agentes de recursos de conhecimento identificados na pesquisa realizada inicialmente junto ao agente *Matchmaker* (ou agente DF) para a execução do serviço de recuperação de conhecimento, tendo como referência o conteúdo da mensagem FIPA ACL preparada acima.

A última atividade do comportamento *updateQueryComplement* ocorre quando o agente de interface recebe as mensagens FIPA ACL com o ato comunicativo ***Inform***, em cujo conteúdo encontram-se as instâncias recuperadas pelos agentes PFMEA representadas na forma de um conjunto de caracteres de acordo com a sintaxe nRQL. Este conjunto de caracteres deve ser, então, convertido e tratado em relação a duplicidades ou respostas nulas e

[6] *new Racer Query Language* (RACER SYSTEMS, 2005)

apresentado no caixa tipo *ComboBox* 6 (figura 6.13) que corresponde ao complemento da consulta.

Deste modo, o usuário pode concluir o segundo estágio da consulta ou a consulta final considerando as instâncias das funções de operações de manufatura modeladas nas bases de conhecimento dos agentes de recursos ativos na organização, como mostra a figura 6.13.

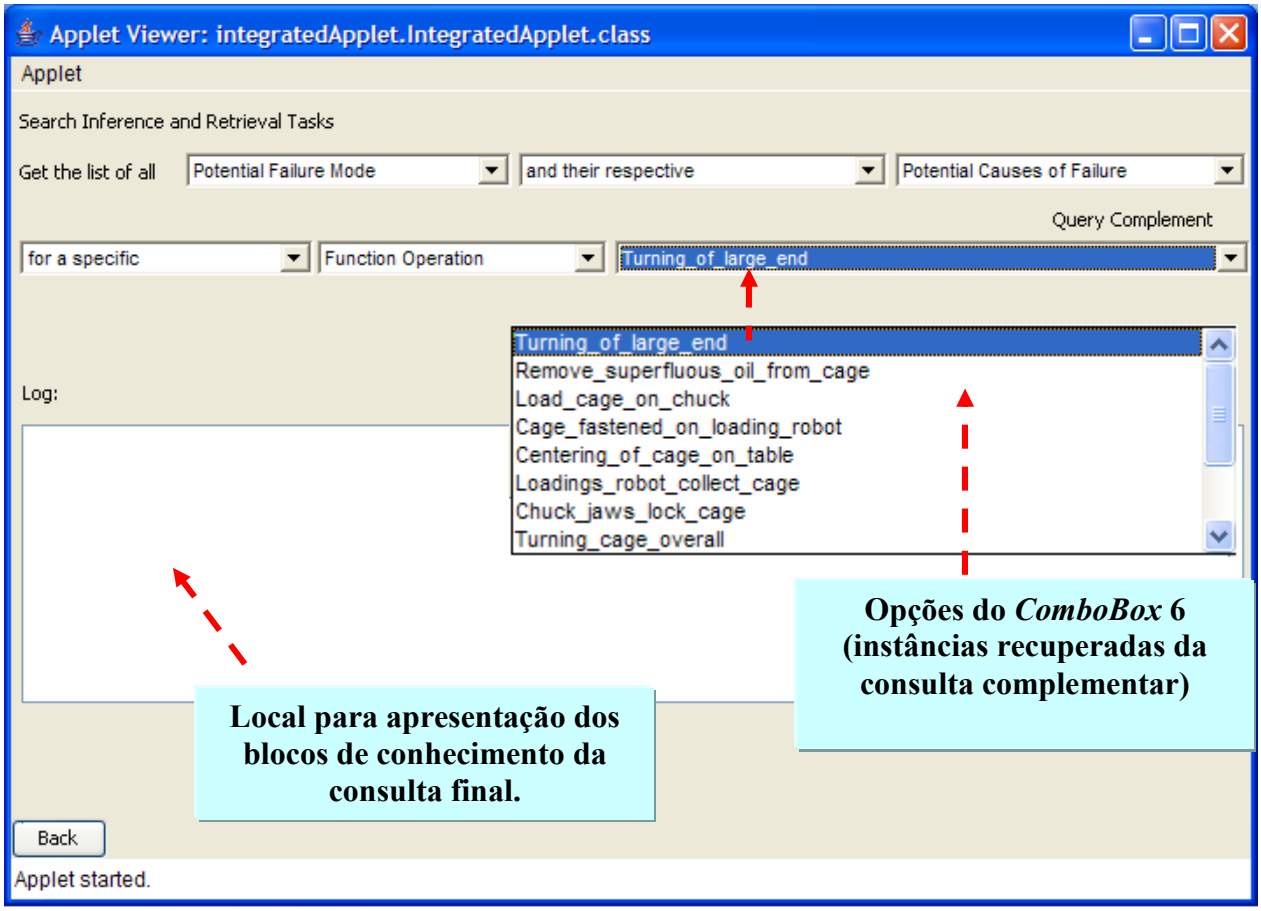


Figura 6.13 – Janela gráfica para configuração da consulta complementar.

Por sua vez, o comportamento *RequestPerformer* implementa os métodos responsáveis pela obtenção das instâncias requisitadas pelo usuário no segundo estágio da consulta ou consulta final, como mostra a figura 6.14. Deve-se ressaltar que esta obtenção deve acontecer também 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.

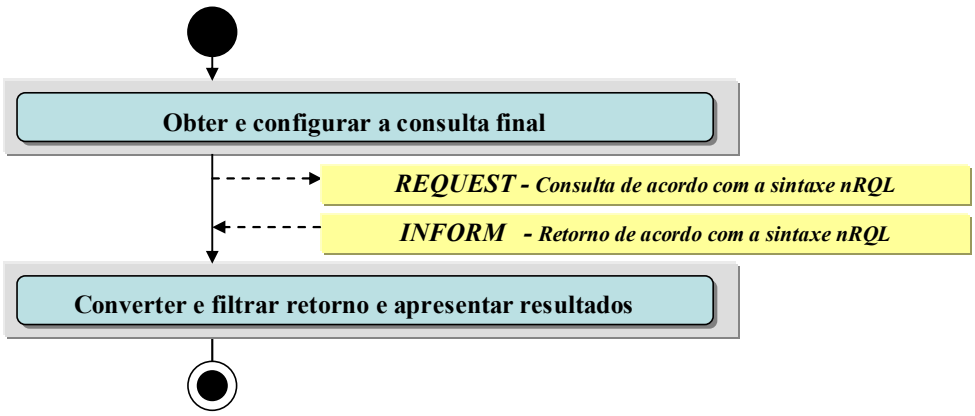


Figura 6.14 – Comportamento *RequestPerformer* do agente de interface.

Por outro lado, a figura 6.15 mostra o resultado da conversão da consulta final para a sintaxe nRQL, no exemplo observa-se, da mesma forma que na consulta anterior, os nomes atribuídos às variáveis, os conceitos da ontologia PFMEA, bem como o construtor lógico "AND", pois, neste exemplo, o objetivo da consulta é recuperar todos os modos de falha e as suas respectivas causas considerando à função de uma operação específica de manufatura selecionada no *ComboBox* 6.

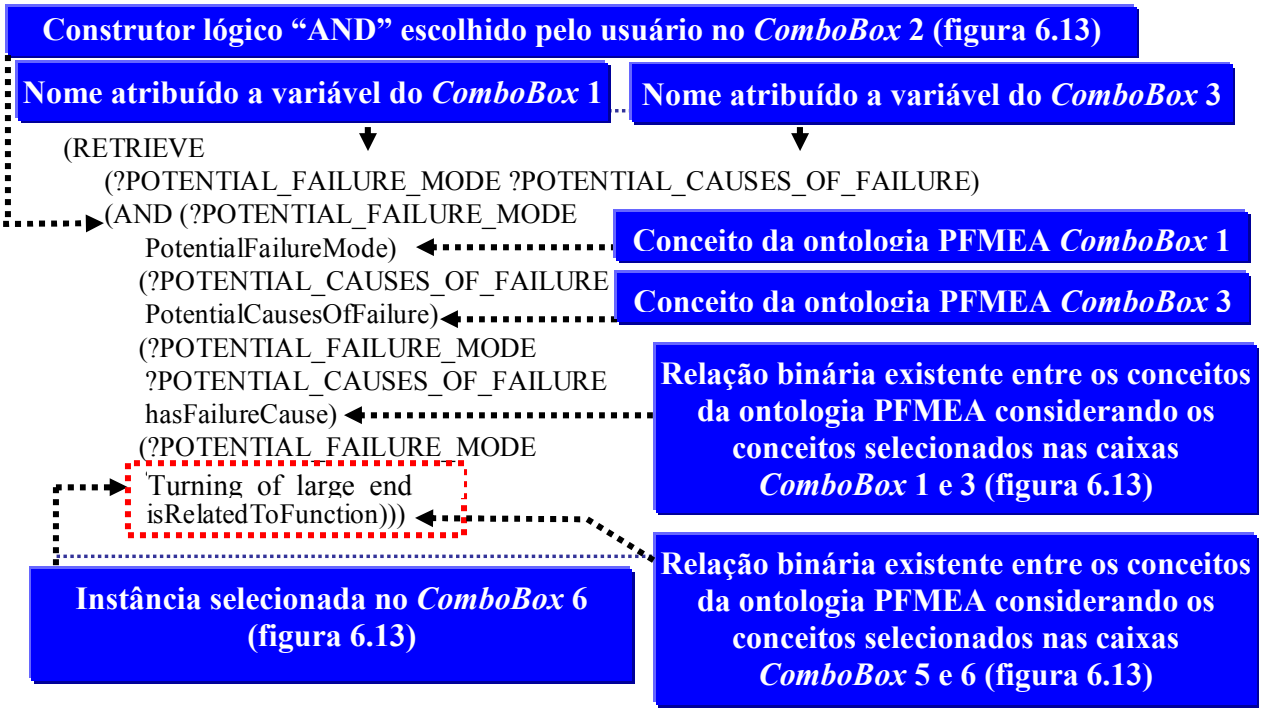


Figura 6.15 – Resultado da conversão da consulta final (figura 6.13) para a sintaxe nRQL.

Em seguida, a atividade utiliza também a classe `jade.lang.acl.ACLMessage`, pré-implementada no arcabouço JADE, e o métodos `setContent()` para o processo transmissão do conteúdo das mensagens FIPA ACL na forma de um conjunto de caracteres (*strings*).

Assim, o agente de interface mobiliza os agentes de recursos PFMEA para a execução do serviço de recuperação de conhecimento, tendo como referência o conteúdo da mensagem FIPA ACL preparada de acordo com a figura 6.15.

Por fim, na última atividade do comportamento, o agente de interface, após receber as mensagens FIPA ACL com o ato comunicativo ***Inform*** com as instâncias recuperadas pelos agentes PFMEA na sintaxe nRQL, deve converter o conteúdo da mensagem e apresentá-la na forma de blocos de conhecimento na área correspondente mostrada na figura 6.13. Mais detalhes sobre os blocos de conhecimento são apresentados no Capítulo 8 durante a verificação das funcionalidades do protótipo do modelo.

6.5 AGENTES DE RECURSOS DE CONHECIMENTO PFMEA

De acordo com as especificações de projeto do modelo proposto no capítulo 4, os agentes de recursos para Análise de Modos de Falha e Efeitos (Agentes de recursos PFMEA) são agentes capazes de acessar bases de conhecimento que decorrem da aplicação do método de Análises de Modos de Falha e Efeitos no domínio de um processo de manufatura específico. Dessa forma, o comportamento dos agentes de recursos foi projetado para acessar e processar uma base de conhecimento específica na forma de uma ontologia codificada de acordo com a linguagem OWL-DL (*OWL DL document*).

Portanto, para levar a efeito este objetivo, é necessário recorrer a um sistema de raciocínio e recuperação de conhecimento de modo a responder as consultas específicas encapsuladas no conteúdo de mensagens FIPA ACL enviadas por pelo agente de interface descrito na subseção anterior. Dentro desta perspectiva, na subseção seguinte são apresentados os detalhes de implementação desta classe de agentes. Cumpre observar que os detalhes de desenvolvimento e codificação da base de conhecimento usando-se o recurso de software Protégé-OWL (CO-ODE, 2007), de acordo com o modelo conceitual da ontologia, são apresentados no Capítulo 7.

6.5.1 Desenvolvimento computacional do comportamento do agente de recursos PFMEA

No comportamento dos agentes de recurso PFMEA um conjunto de tarefas iniciais é implementado no método `setup`, tarefas estas que são responsáveis por efetuar o registro do agente PFMEA ativo plataforma na base de conhecimento do agente *Matchmaker* (ou agente DF), que responde pelo serviço de “páginas amarelas” da plataforma, na qual são registradas a descrição dos agentes e o serviço oferecido como um subdomínio particular destas “páginas

amarelas". Estas tarefas são implementadas por meio dos métodos próprios da classe *jade.domain.**DFService***, cujos detalhes podem ser encontrados em Bellifemine et al. (2006a).

6.5.2 Comportamento principal dos agentes de recursos PFMEA

O comportamento principal do agente de recurso PFMEA, por sua vez, é implementado computacionalmente pela extensão da classe *jade.core.behaviours.**CyclicBehaviour***, destinada a modelar um comportamento que deve ser mantido em execução contínua como se estivesse em *loop* (BELLIFEMINE et al., 2006a).

Este comportamento busca atender às especificações de projeto do modelo dos agentes de recursos PFMEA, os quais foram projetados para esperar, continuamente, por mensagens do tipo FIPA ACL com o ato comunicativo ***Request*** enviadas pelo agente de interface para PFMEA. Este ato comunicativo é responsável por estabelecer a comunicação entre o emissor, no caso, o agente de interface para PFMEA e o receptor ou o agente de recursos em questão, onde o agente emissor solicita ao receptor que este execute um serviço de raciocínio e recuperação a partir de uma base de conhecimento ontológica.

O diagrama da figura 6.16 expressa o comportamento principal dos agentes de recurso PFMEA em termos de seu modelo de tarefas. A primeira tarefa especifica a função a ser executada inicialmente pelos agentes de recursos ativos após o recebimento da mensagem FIPA ACL com o ato comunicativo ***Request*** em cujo conteúdo encontra-se uma consulta já na sintaxe nRQL. Esta tarefa implementa os métodos responsáveis por adicionar a indicação do componente a ABox correspondente à base ontológica particular associada ao agente de recurso, tendo em vista este componente ABox ser distinto para cada uma das bases de conhecimento, em contraste ao componente TBox que é comum.

A implementação da segunda tarefa prevista no comportamento, mostrada na figura 6.16, foi realizada pela extensão da Interface de Programação de Aplicativos baseada em JAVA (*Application Programming Interface*) denominada JRacer (RACER SYSTEMS, 2007) desenvolvida para permitir o comunicação de aplicações escritas em linguagem JAVA com o RacerPro Server "*kernel*" via TCP/IP *server object*.

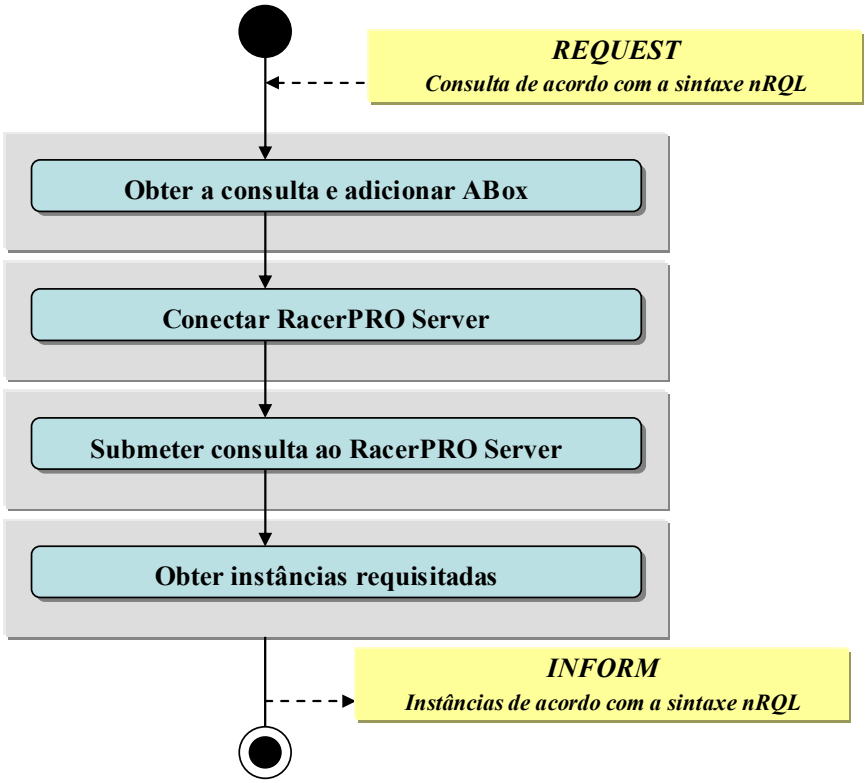


Figura 6.16 – Comportamento dos agentes de recursos PFMEA.

A terceira e quarta tarefas implementam os métodos responsáveis por submeter a consulta preparada na primeira atividade ao RacerPro Server via conexão TCP/IP padrão aberta, de acordo com os comandos nativos do sistema, e receber a resposta, respectivamente.

Em seguida, as repostas fornecidas pelo sistema de raciocínio são incluídas como conteúdo da mensagem, de acordo com a sintaxe nRQL, por meio da classe pré-implementada no arcabouço JADE `jade.lang.acl.ACLMessage` e pelo método correspondente `setContent()` de modo a viabilizar a transmissão do conteúdo das mensagens FIPA ACL com o ato comunicativo ***Inform*** para o agente de interface que requisitou o serviço de raciocínio.

6.6 AGENTE TIPO MATCHMAKER

O agente tipo *Matchmaker* fundamenta-se conceitualmente no padrão de interação *matchmaking* apresentado em Jha et al. (1998), e tem como comportamento principal a função de identificar todos os agentes de interface e agentes de recursos de conhecimento ativos na sociedade de agentes.

Os passos desta interação compreendem: o registro e o anúncio das capacidades e serviços disponibilizados pelos agentes ativos na plataforma com o agente *Matchmaker*, o qual é

responsável por armazená-las na forma de meta-informação. Deste modo, os agentes de interface podem comunicar-se com o agente *Matchmaker* buscando identificar outros agentes que oferecem um dado serviço, para em seguida iniciar uma interação direta com estes usando-se as meta-informações para controlar estas interações. Deve-se destacar que o agente *Matchmaker* foi implementado usando as funcionalidades do agente facilitador de diretórios (ou agente DF) pré-programado no arcabouço JADE.

6.7 SÍNTESE DO CAPÍTULO

Este capítulo apresentou os detalhes da implementação formal do comportamento de cada uma das classes de agentes propostos a partir das especificações de projeto do modelo em relação a cada uma destas classes, que em termos metodológicos tem o objetivo de indicar a forma adotada para a implementação visando permitir novas implementações.

Neste contexto, destaca-se a noção de comportamento, o qual pode ser entendido como um conjunto de ações e reações de um agente face às interações e realimentações decorrentes do meio no qual os agentes estão inseridos. Assim, do ponto de vista computacional, um comportamento (*behaviour*) é basicamente um processo ativado por evento (*Event Handler*) e consiste em métodos que descrevem como o agente age em relação a este evento (BELLIFEMINE et al., 2006a).

Formalmente, um evento pode ser considerado como uma mudança relevante no estado do agente, que em termos práticos significa a recepção de uma mensagem ou a interrupção de uma contagem de tempo.

Deloach (2001) argumenta ainda que o comportamento de um agente pode ser expresso por um conjunto de "n" tarefas concorrentes, as quais não somente definem o processamento interno que um agente deve realizar, mas também como as interações com outros agentes relacionam se com os processos internos. Neste sentido, cada tarefa especifica uma única *thread* de controle que define o comportamento de um agente e integra as interações inter-agentes, bem como intra-agentes.

Este capítulo revelou também a estratégia implementada no agente de interface RBC para a configuração das consultas envolvendo a seleção de descritores e a atribuição de pesos em função das percepções e necessidades de recuperação de um usuário em particular. Da mesma forma, foi descrita a estratégia implementada no agente de interface PFMEA, que baseia-se em conceitos, relações e instâncias modeladas na ontologia correspondente e é realizada em dois estágios. Foram também apresentadas as alternativas escolhidas para os formatos dos

conteúdos das mensagens, que são fundamentais para a realização dos métodos de raciocínio sobre as bases de conhecimento.

No próximo capítulo são apresentados os detalhes da implementação das bases de conhecimento dos agentes de recursos de conhecimento a partir das especificações do modelo, bem como a forma de persistência física destas bases.

CAPÍTULO 7
IMPLEMENTAÇÃO DAS BASES DE CONHECIMENTO DOS AGENTES DE RECURSOS DE CONHECIMENTO

Este capítulo tem por objetivo apresentar e discutir os aspectos fundamentais relacionados à forma de implementação das bases de conhecimento dos agentes de recursos de conhecimento. Estas bases seguem as especificações de projeto do modelo estabelecidas no Capítulo 4. Este capítulo busca, em especial, mostrar as estratégias usadas para tornar as bases de conhecimento operacionais, de modo que os agentes possam cumprir seus objetivos de projeto.

Cientificamente, 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 subseqüentemente 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 provado uma tarefa extremamente desafiadora, e ao longo das duas últimas décadas diversas abordagens têm sido propostas para a aquisição interativa de conhecimento.

Neste trabalho de tese, o foco da aquisição do conhecimento concentra-se na captura 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 (2002) 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 ressaltar que 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 produtora de peças pelo processo de moldagem por injeção de termoplásticos para aplicações no setor eletro-eletrônico situada no sul do país.

7.1 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 (*troubleshooting cases*), os quais englobam 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, como discutido no Capítulo 4.

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 7.1.

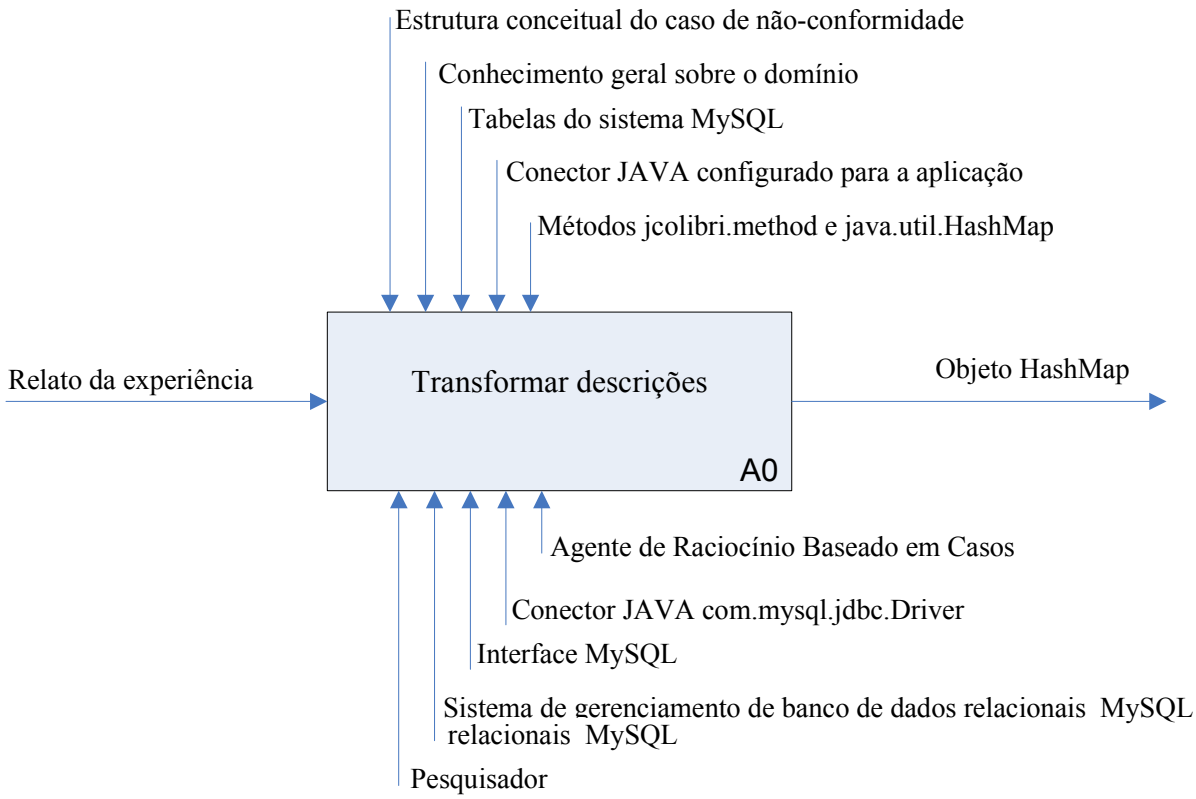


Figura 7.1 – Diagrama IDEF0 : A0 – Função de transformação das descrições.

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 7.2.

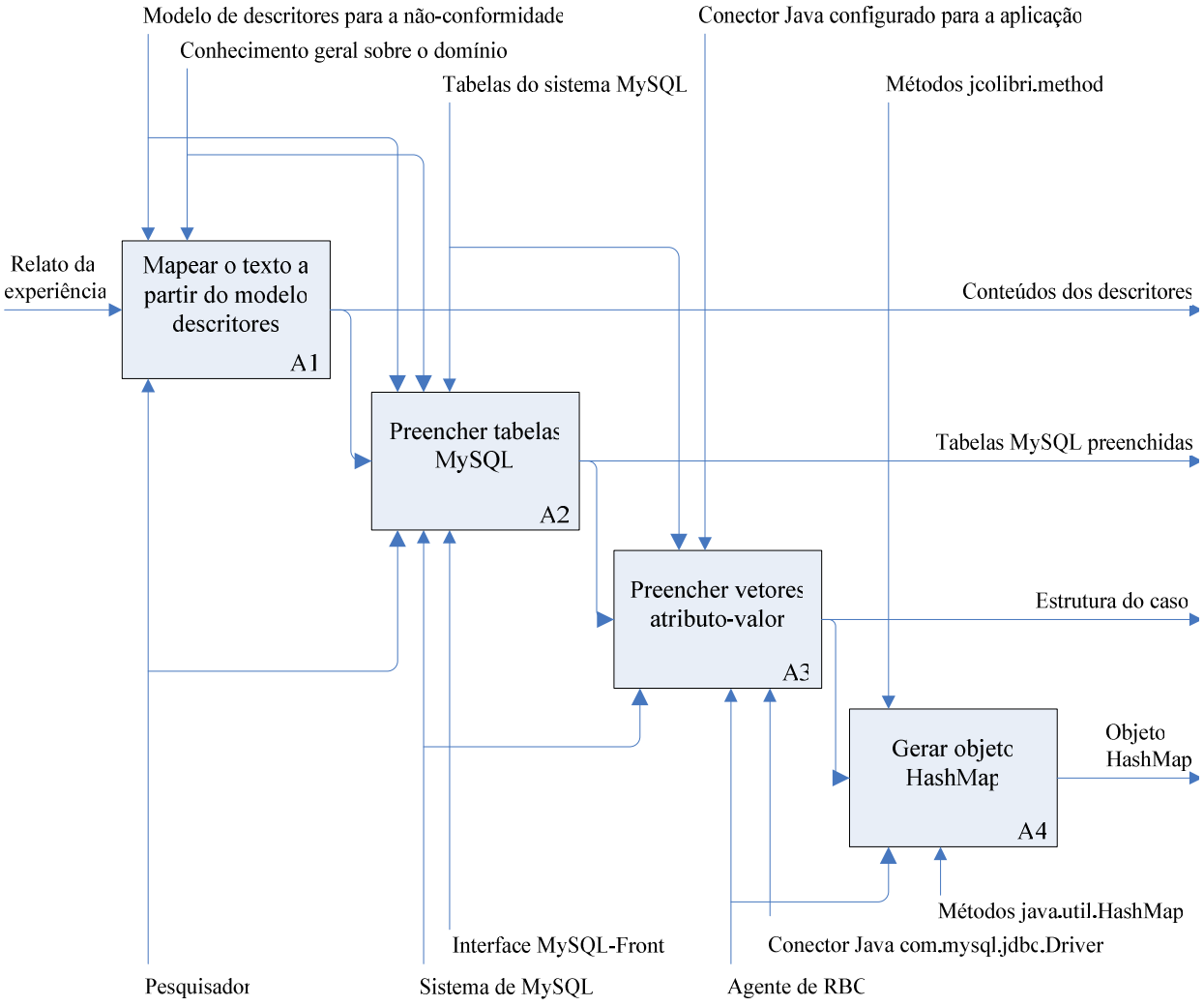


Figura 7.2 – Diagrama IDEF0 com o desdobramento da função transformação.

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 7.3 mostra a atividade A1 (Mapear o texto a partir do modelo de descritores) modelada no diagrama IDEF0 (figura 7.2).

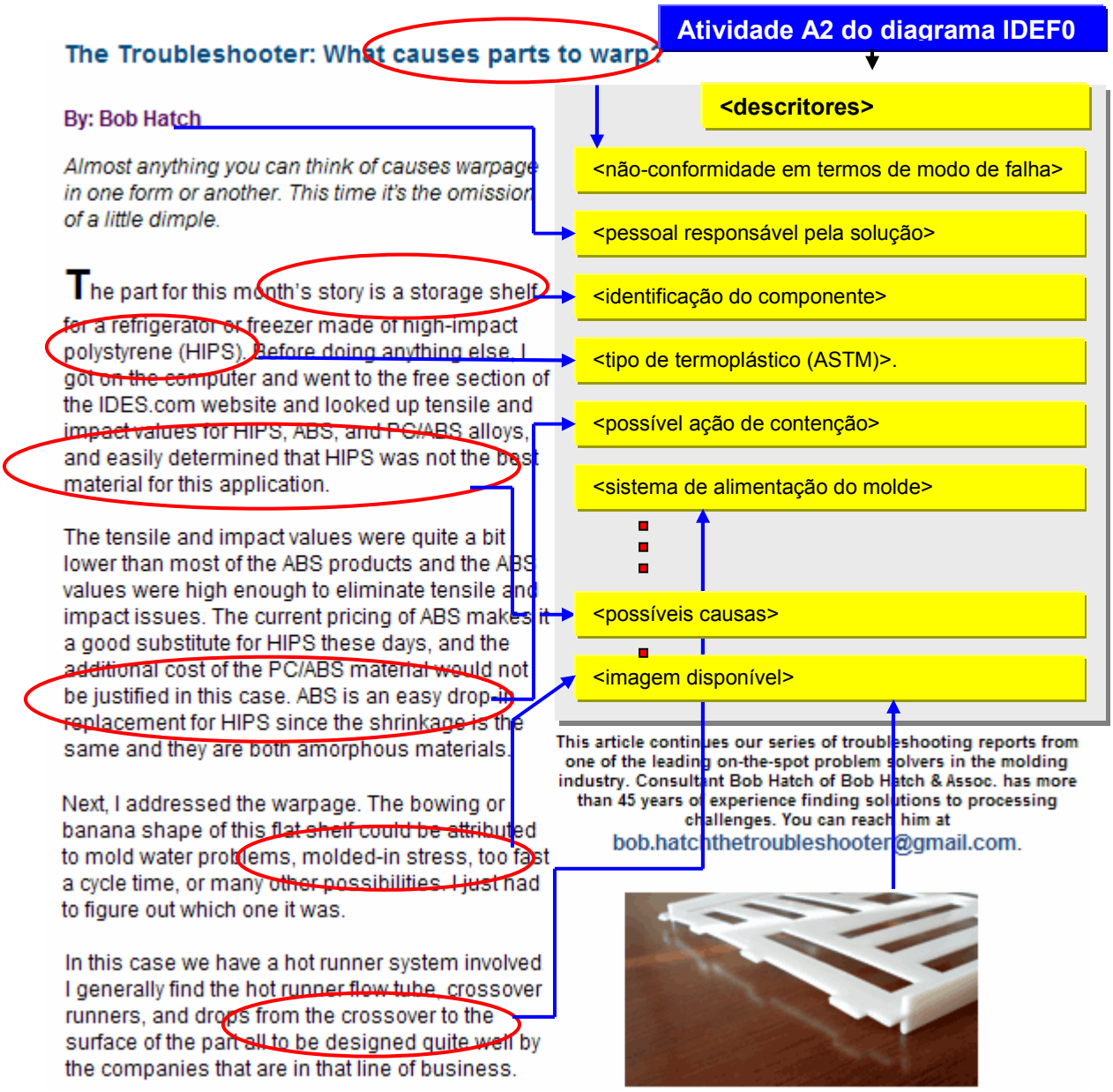


Figura 7.3 – Mapeamento unívoco para a função de transformação A1.

Este mapeamento realiza-se de forma não automática e consiste fundamentalmente em relacionar o conhecimento factual dos elementos textuais disponíveis no relato considerando a estrutura conceitual do caso de não-conformidade e tendo por base o conhecimento geral sobre o domínio.

Dentro desta perspectiva, foram usados inicialmente os casos de teste correspondentes aos relatos de soluções de problemas não-conformidade em processos de moldagem por injeção apresentados pelo especialista Bob Hatch publicados em língua inglesa no periódico *Injection Molding Magazine* (IMMNET, 2007). E, posteriormente, foram usados os relatos obtidos durante a pesquisa de campo, embora outras fontes tenham sido consultadas durante as etapas iniciais do processo de aquisição de conhecimento.

A figura 7.4 demonstra a execução da atividade A2 (Preencher tabelas MySQL®), que tem como entrada os conteúdos dos descritores resultantes da atividade A1, que são transferidos diretamente para as colunas correspondentes na tabela do sistema de gerenciamento de banco de dados relacionais MySQL® configuradas especialmente para tal. Deve-se mencionar que cada linha representa um relato em particular, e que esta função é realizada de forma não automática mediante a interface MySQL-Front (MySQL, 2006).

Neste trabalho, adota-se o sistema de gerenciamento de banco de dados relacionais MySQL® (MYSQL, 2006) como meio de persistência física (ou estática) somente do conteúdo dos descritores de cada caso. Entretanto, deve-se observar que isto *per se* não caracteriza a base de conhecimento dos agentes, pois a verdadeira base de conhecimento ou base de casos é estabelecida dinamicamente pelos agentes de recursos RBC ativos de acordo com a atividade.

A figura 7.4 ilustra a atividade A3 (Preencher vetores atributo-valor), que é executada automaticamente quando o agente está ativo na plataforma de agentes por meio do conector (*com.mysql.jdbc.Driver*) configurado previamente quando da especialização do arcabouço jCOLIBRI. Assim, este conector é responsável pelo mapeamento dinâmico dos parâmetros dos descritores pré-definidos, e formalizado de acordo com a estrutura conceitual do caso e as respectivas colunas das tabelas do sistema de gerenciamento de banco de dados relacionais MySQL®.

Por outro lado, a atividade A4 (Gerar o objeto *HashMap*) é executada também automaticamente pelos agentes de recursos RBC quando ativos na sociedade por meio dos métodos implementados no comportamento destes agentes conforme apresentado no Capítulo 6. Como mencionado anteriormente, estes objetos *HashMap* são destinados às interações sociais entre estes agentes e o agente de interface RBC.

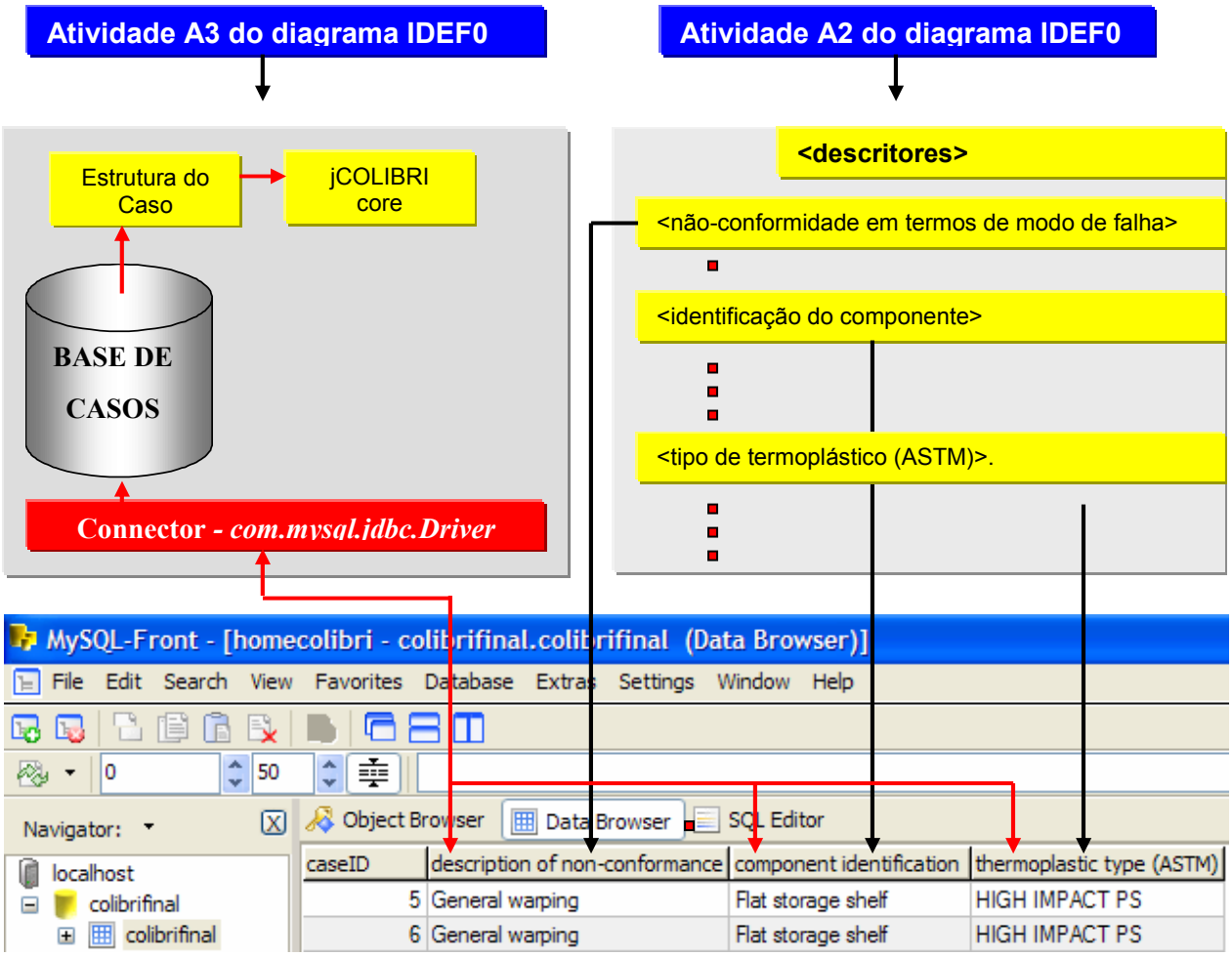


Figura 7.4 – Mapeamento unívoco para a função de transformação A2.

Por fim, destaca-se que a mesma estratégia de aquisição de conhecimento foi utilizada também para o preenchimento das bases com conhecimentos obtidos por meio do trabalho de campo. Isto pode ser feita porque, em essência, a forma de relato relacionada às experiências de solução de problemas de não-conformidade na empresa pesquisada seguiu mesma linha de raciocínio encontrada na literatura.

7.2 IMPLEMENTAÇÃO DAS BASES DE CONHECIMENTO DOS AGENTES PFMEA

A etapa de implementação das bases de conhecimento dos agentes de recursos PFMEA é apresentada nesta subseção considerando dois aspectos principais. O primeiro aspecto diz respeito ao processo de codificação da ontologia PFMEA mediante a linguagem OWL-DL (*Web Ontology Language – Descrition Logic*) desenvolvida pelo W3*Consortium* (PATEL-SCHNEIDER *et al.*, 2004) a partir da ontologia PFMEA formalizada no Capítulo 4.

O segundo aspecto concentra-se na função de transformação da descrição no nível do conhecimento, isto é, o conhecimento factual decorrente da aplicação do método de Análise

do Modo de Falha e Efeitos em uma descrição no nível simbólico que envolve os conceitos, relações binárias e instâncias da ontologia PFMEA. O nível simbólico é importante porque assegura o compartilhamento e reuso entre pessoas, agentes de interface e agentes de recursos PFMEA.

7.2.1 Codificação das bases de conhecimento ontológicas

Na etapa de codificação da base de conhecimento, que corresponde em essência à codificação da ontologia PFMEA, usou-se o editor gráfico de ontologias PROTÉGÉ OWL-DL desenvolvido no âmbito do projeto CO-ODE (*Collaborative Open Ontology Development Environment*) (CO-ODE, 2007), cuja seleção foi descrita no Capítulo 5.

7.2.1.1 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 7.5.

A interface do editor de ontologias PROTÉGÉ OWL-DL, mostrada na figura 7.5, 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 tal os sete eixos formalizados na ontologia PFMEA a saber: produto (*ProductConcepts*), processo (*ProcessConcepts*), funções (*FunctionConcepts*), falhas (*FailureConcepts*), ações (*ActionsConcepts*), descrição (*FMEADescription*) e imagens (*ImagesConcepts*).

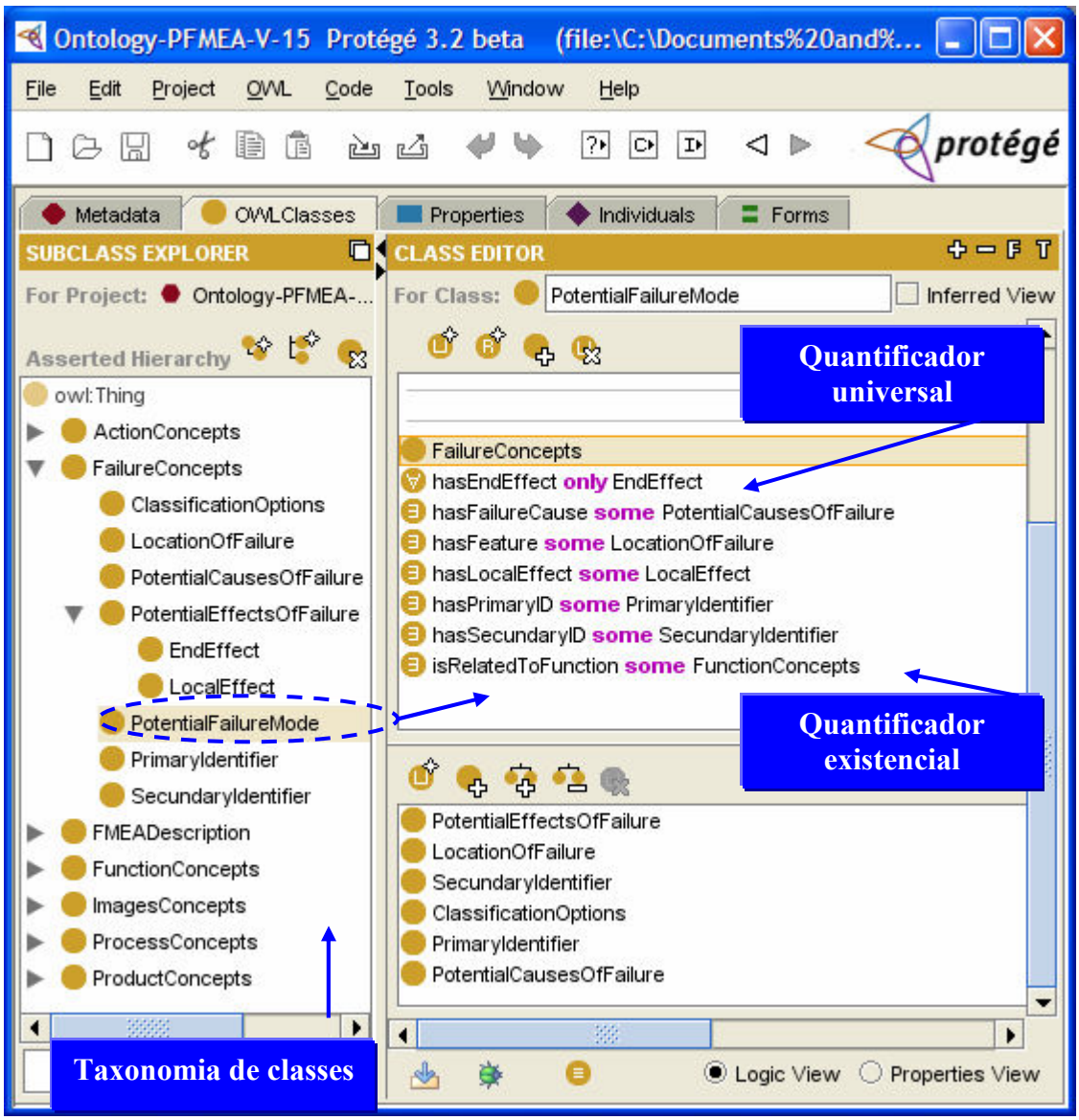


Figura 7.5 – Taxonomia de classes OWL-DL.

De acordo com Horridge et al. (2004), uma das características chave das definições de superclasses-subclasses OWL-DL são as relações de subsunção 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 em particular está sob dependência de outro mais geral mediante o estabelecimento de uma hierarquia de conceitos determinada pela comparação de suas definições. Além, da classificação do TBox (*classify*) isto é, a 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 7.5 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 $(\forall)$[1], mostrado na equação 1, 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.

$$\forall hasEndEffect \quad only \quad EndEffect \tag{1}$$

Assim, todas 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 $(\exists)$[2], mostrado na equação 2, 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*".

$$\exists \ isRelatedToFunction \quad some \quad OperationFunction \tag{2}$$

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

Adicionalmente, a propriedade "*isRelatedToFunction*" foi modelada como uma propriedade inversa da propriedade "*hasFailureMode*", deste modo o sistema de raciocínio

[1] Pode ser lido como "somente" ou ainda como "*allValuesFrom*" na terminologia OWL-DL.

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

pode inferir automaticamente que um indivíduo da subclasse "*OperationFunction*" está relacionado ao indivíduo da subclasse "*PotentialFailureMode*" usando a propriedade "*hasFailureMode*", como mostra a figura 7.6.

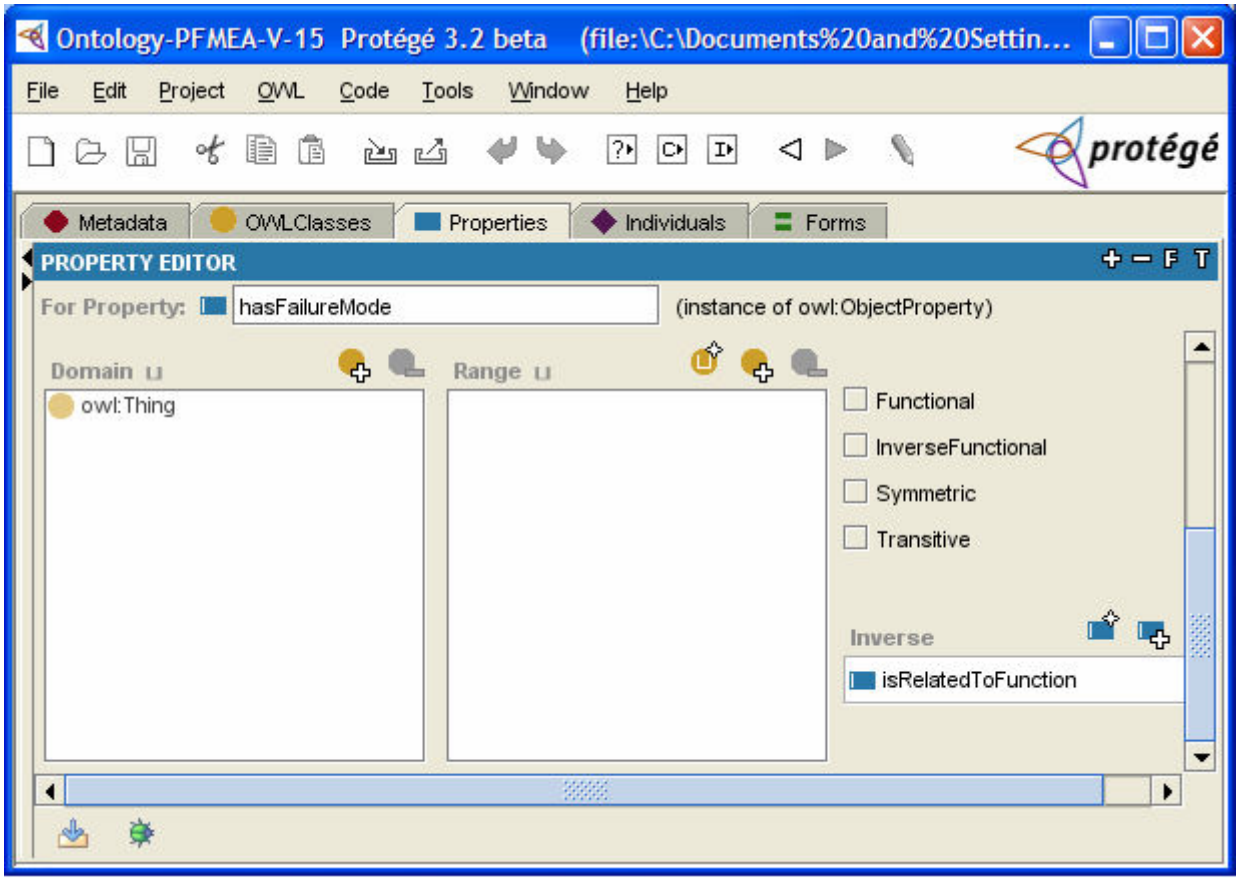


Figura 7.6 – Modelagem de uma propriedade inversa.

Por outro lado, observa-se que a propriedade "*isRelatedTo_Item*", mostrada na figura 7.7, 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 dizer que a propriedade "*isRelatedTo_Item*" 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*".

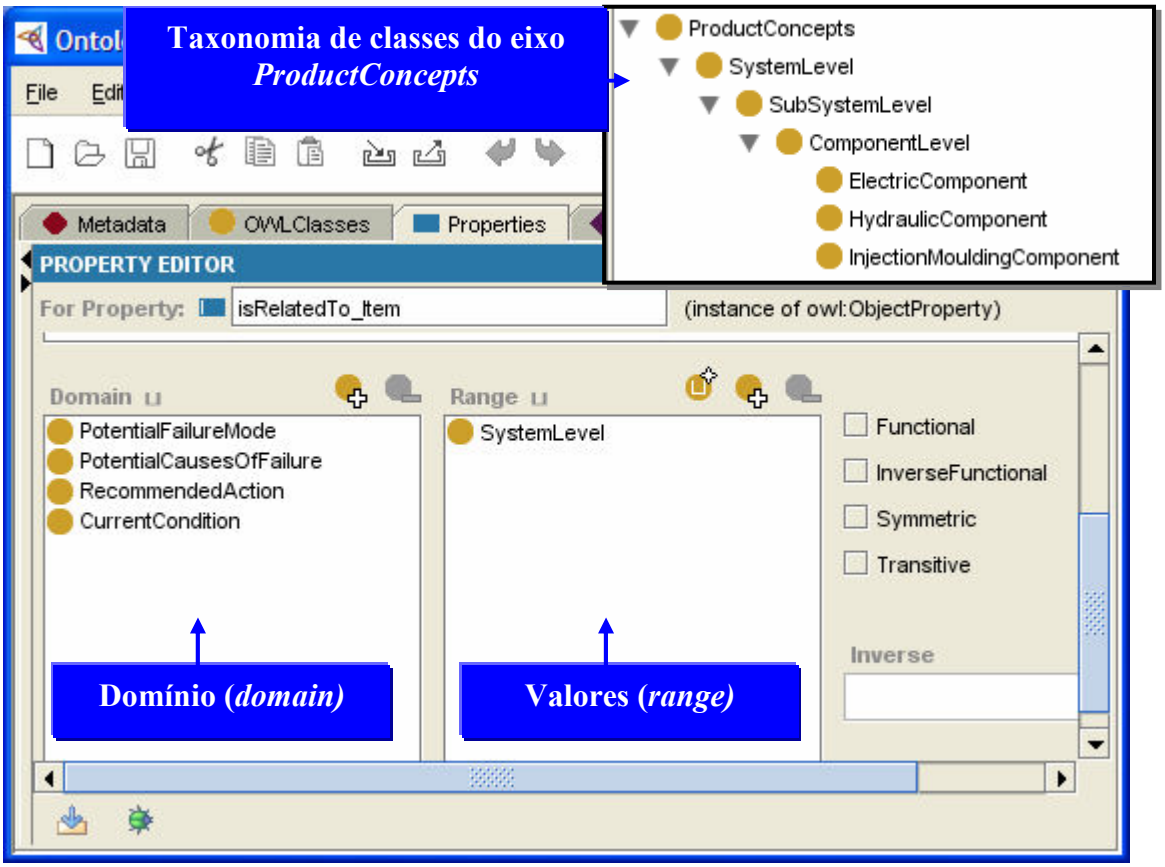


Figura 7.7 – Modelagem usando *domain* e *range*.

7.2.1.2 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 7.8.

A interface do editor PROTÉGÉ OWL-DL apresentada nas figuras 7.8 e 7.9 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 "*Oil_remainder_on_cage*", 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 7.5, 7.6 e 7.7.

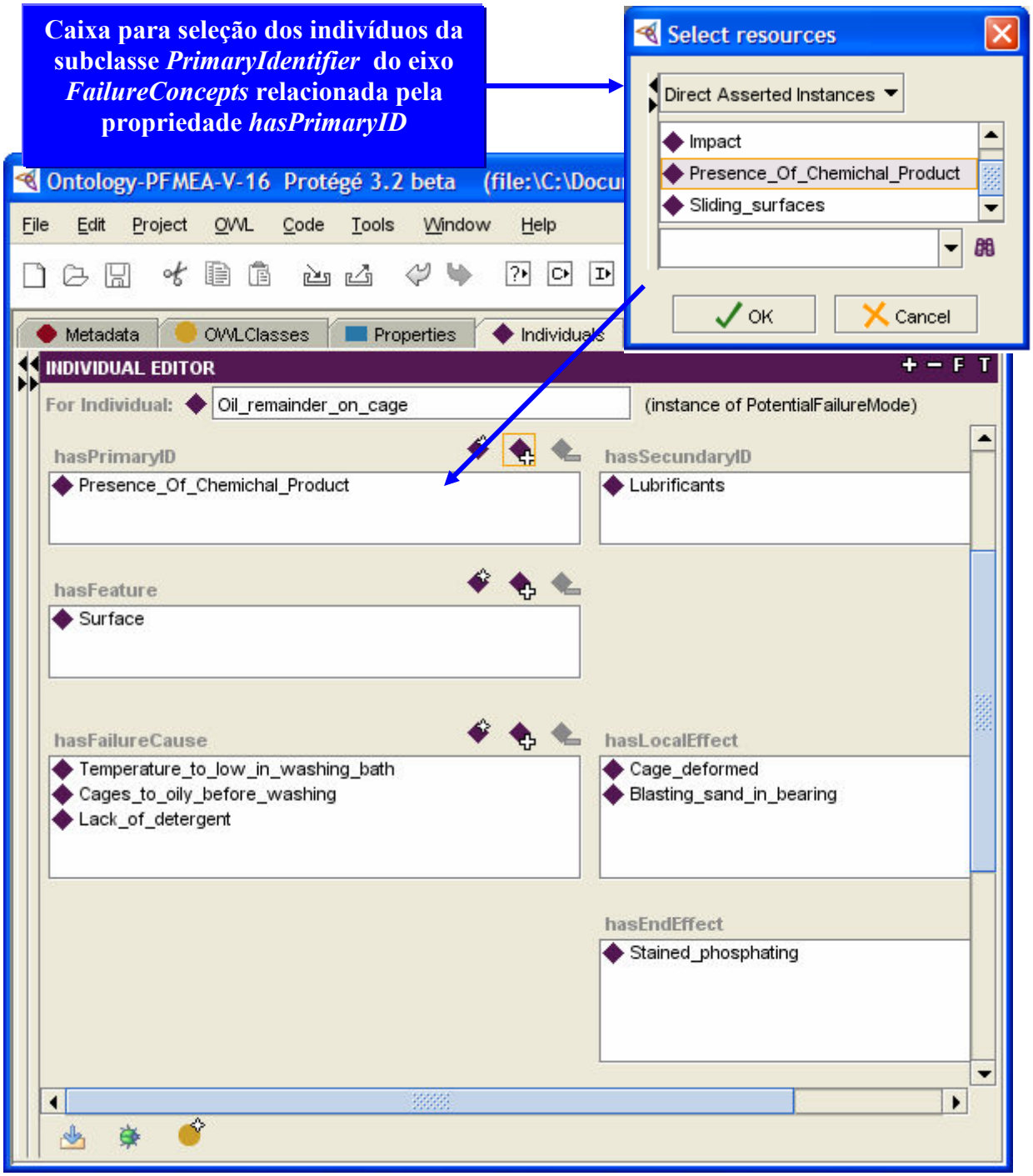


Figura 7.8 – Especificação de indivíduos da subclasse OWL-DL (Parte I).

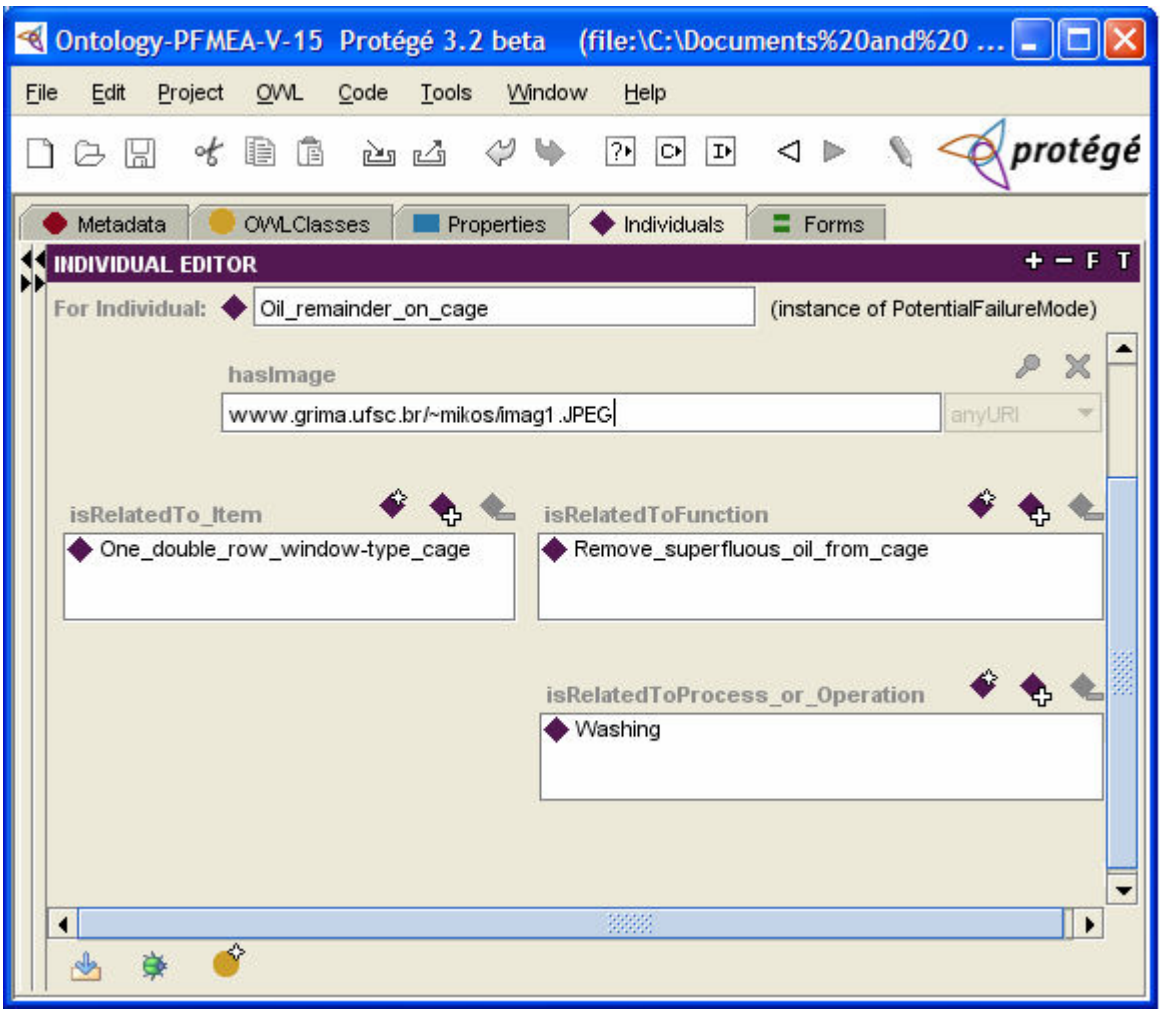


Figura 7.9 – Especificação de indivíduos da subclasse OWL-DL (Parte II).

Esta associação se dá pela escolha de indivíduos diretamente a partir de caixas tipo *ComboBox* que são apresentadas ao usuário em função da propriedade OWL-DL (modelada no componente TBox). E importante ressaltar que para os indivíduos relacionados por propriedades inversas (modeladas como propriedades inversas no TBox) a associação é realizada de forma automática pela interface. Por exemplo, se o indivíduo "*Remove_superfluous_oil_from_cage*" da classe "*OperationFunction*" durante a sua definição tiver sido associado ao indivíduo "*Oil_remainder_on_cage*" mediante a propriedade "*hasFailureMode*", a caixa *ComboBox* correspondente à propriedade "*isRelatedToFunction*" (mostrada na figura 7.9), será preenchida automaticamente, pois as propriedades em questão foram modeladas como inversas.

7.2.2 Função de transformação dos níveis de descrição do conhecimento

A função de transformação da descrição no nível do conhecimento no contexto do método de Análise do Modo de Falha e Efeitos, para uma descrição no nível simbólico, ocorre de acordo com o diagrama IDFE0 mostrado na figura 7.10.

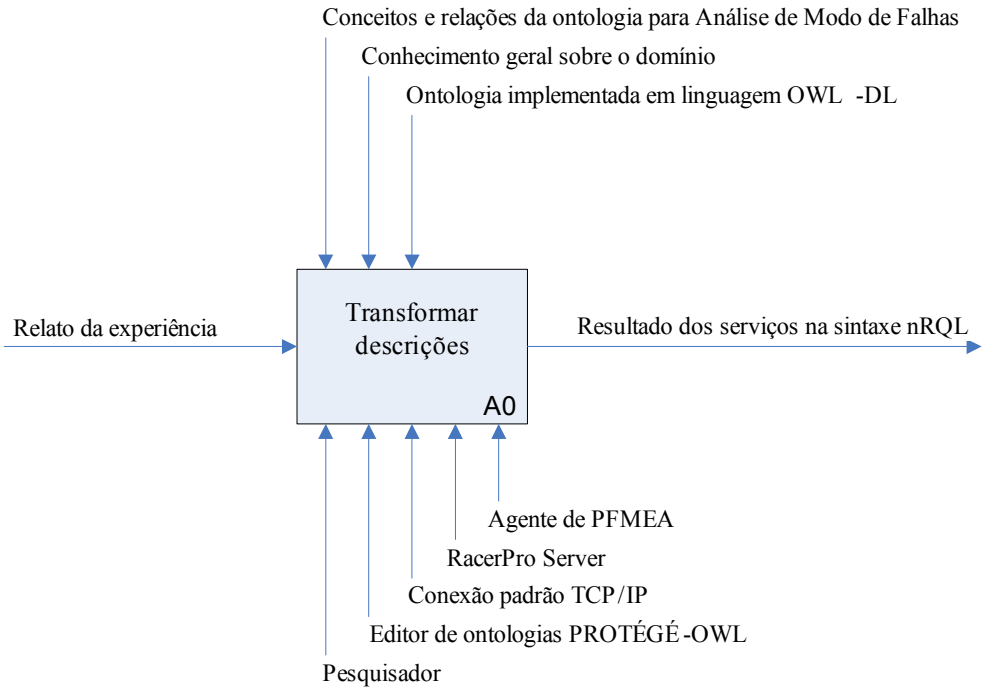


Figura 7.10 – Diagrama IDEF0 – nível A0: Transformação da descrição no nível do conhecimento em descrição simbólica.

A estratégia adotada nesta etapa usa também a técnica de aquisição de conhecimento tipo *top-down* baseada em modelos apresentada por van Heijst *et al*. (1997), na qual as classes OWL-DL, propriedades e indivíduos, são usados para guiar o processo de aquisição do conhecimento relevante, indicando quais elementos necessitam ser obtidos das fontes disponíveis visando permitir o compartilhamento e reuso entre pessoas, agentes de interface e agentes de recursos PFMEA.

Dentro desta perspectiva, inicialmente foram usados como casos de teste um corpo de conhecimento já validado, o qual é resultante da aplicação do método PFMEA por especialistas em manufatura de uma empresa produtora de componentes automotivos no âmbito de um projeto SeisSigma (*SixSigma*) apresentado na literatura por Lennartsson e Vanhatalo (2004), além de relatos obtidos durante a pesquisa de campo, embora outras fontes

tenham sido consultadas durante as etapas iniciais do processo de aquisição de conhecimento, as quais foram listadas no Capítulo 4.

Esta estratégia tem como fundamento também uma cadeia de mapeamentos unívocos, os quais têm início com os elementos factuais contidos nos documentos decorrentes do método de Análise do Modo de Falhas e Efeitos, como mostra o diagrama IDEF0 da figura 7.11.

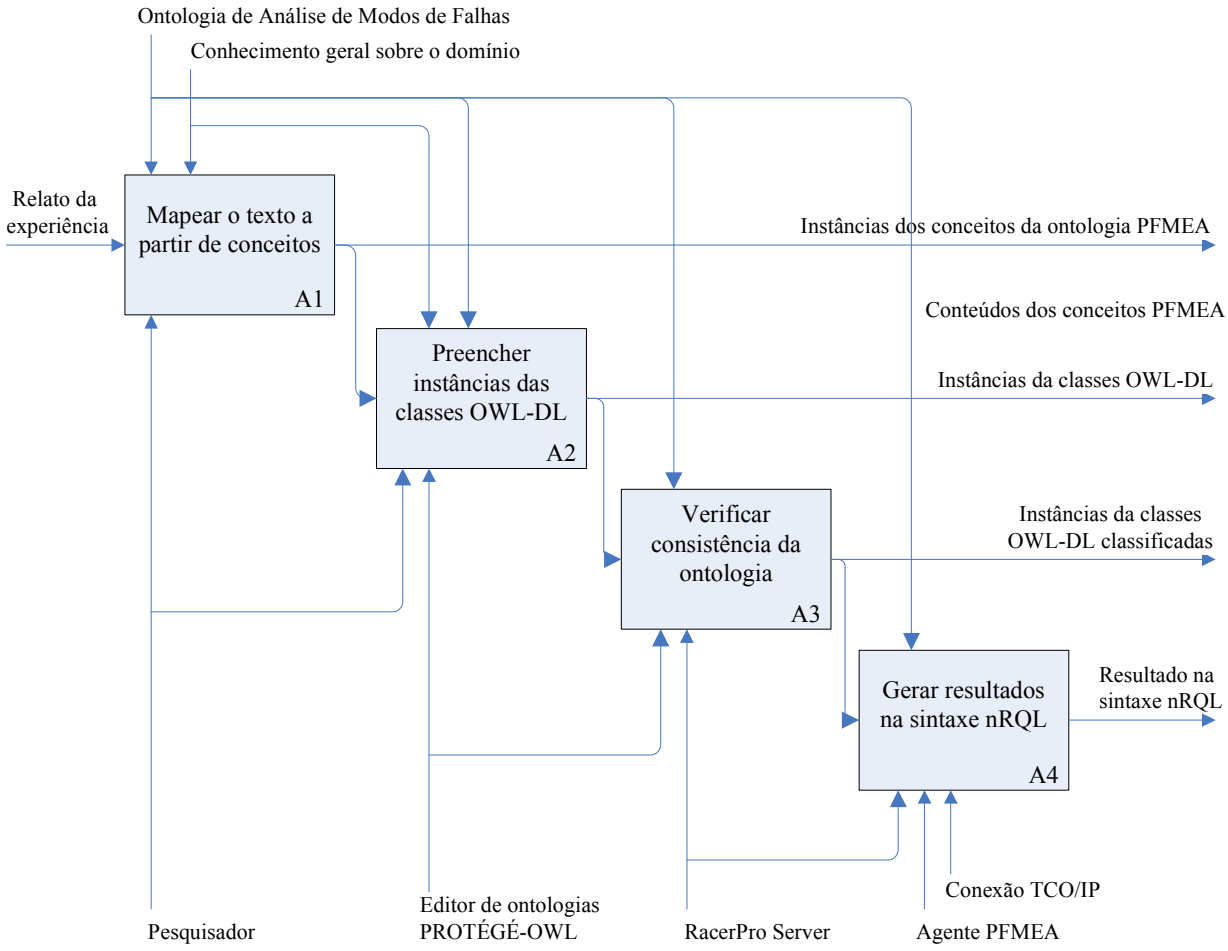


Figura 7.11 – Diagrama IDEF0 referente ao desdobramento da função transformação.

A figura 7.12 mostra a atividade A1 (Mapear o texto a partir de conceitos) da estratégia do diagrama IDEF0 ilustrado na figura 7.11.

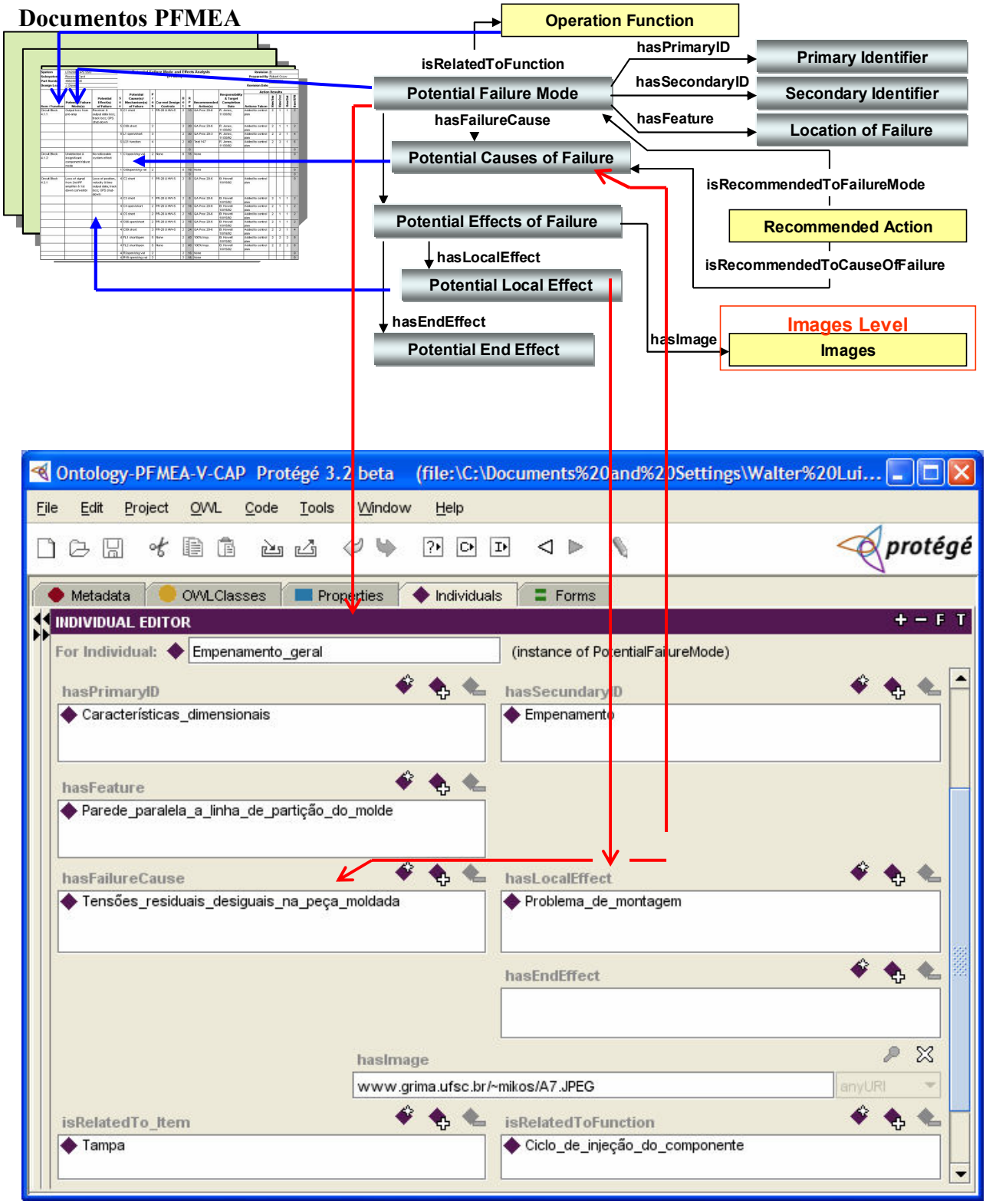


Figura 7.12 – Mapeamento do texto a partir de conceitos, da estratégia do diagrama IDEF0.

Destaca-se que este mapeamento realiza-se de forma não automática, e consiste fundamentalmente em relacionar o conhecimento factual dos elementos textuais disponíveis

no documento FMEA (ver anexo 1) considerando as definições de classes e indivíduos OWL-DL.

Por sua vez, a execução da atividade A2 (Preencher indivíduos ou instâncias das classes OWL-DL) tem como entradas os indivíduos ou instâncias resultantes da atividade A1, os quais são codificados usando-se a interface do editor de ontologias PROTÉGÉ OWL-DL configurada especialmente para tal finalidade. A figura 7.13 apresenta um trecho do código OWL-DL gerado pelo editor PROTÉGÉ OWL descrevendo a classe "PotentialFailureMode" usando as restrições em propriedades mostradas na figura 7.5.

```
</owl:Ontology>
 </owl:Class>
  <owl:Class rdf:about="#PotentialFailureMode">
   <owl:disjointWith rdf:resource="#PotentialEffectsOfFailure"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="hasFailureCause"/>
     </owl:onProperty>
     <owl:someValuesFrom rdf:resource="#PotentialCausesOfFailure"/>
    </owl:Restriction>
  </rdfs:subClassOf>
```

Figura 7.13 – Código OWL-DL relativo à definição da classe "*PotentialFailureMode*".

A atividade A3 (Verificar a consistência da ontologia) é executada automaticamente pelo sistema de raciocínio RacerPro Server conectado diretamente ao editor PROTÉGÉ OWL, usado para detectar inconsistências na definição do componente ABox.

Por fim, a atividade A4 (Gerar resultados na sintaxe nRQL) é executada também automaticamente pelos agentes de interface e recursos PFMEA quando ativos na organização por meio de métodos implementados como comportamentos destes agentes.

7.3 SÍNTESE DO CAPÍTULO

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

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

E 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, 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.

CAPÍTULO 8
PROCESSO DE VERIFICAÇÃO E VALIDAÇÃO DO MODELO PROPOSTO

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 concentrar-se nas especificações do projeto e seu respectivo desenvolvimento computacional. Deste modo, a verificação tem por escopo assegurar que o modelo tenha sido corretamente implementado a partir das especificações de projeto, enfatizando os processos sociais dos agentes e a consistência da organização multiagentes.

Com esta finalidade são apresentadas as funcionalidades desenvolvidas no protótipo de laboratório 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 7.

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 de testes obtidos pela pesquisa de campo adicionalmente aos casos de testes obtidos na literatura. Nesta linha, busca-se ainda identificar as potencialidades e as oportunidades de melhoria do modelo proposto em relação ao apoio à solução de problemas de não-conformidades no domínio, a partir de uma análise crítica.

8.1 VERIFICAÇÃO DAS FUNCIONALIDADES DO PROTÓTIPO DO MODELO

O protótipo do modelo dispõe de uma primeira janela gráfica do usuário, que foi desenvolvida computacionalmente na forma de um *Java Applet*. Esta interface é apresentada, automaticamente, 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 8.1.

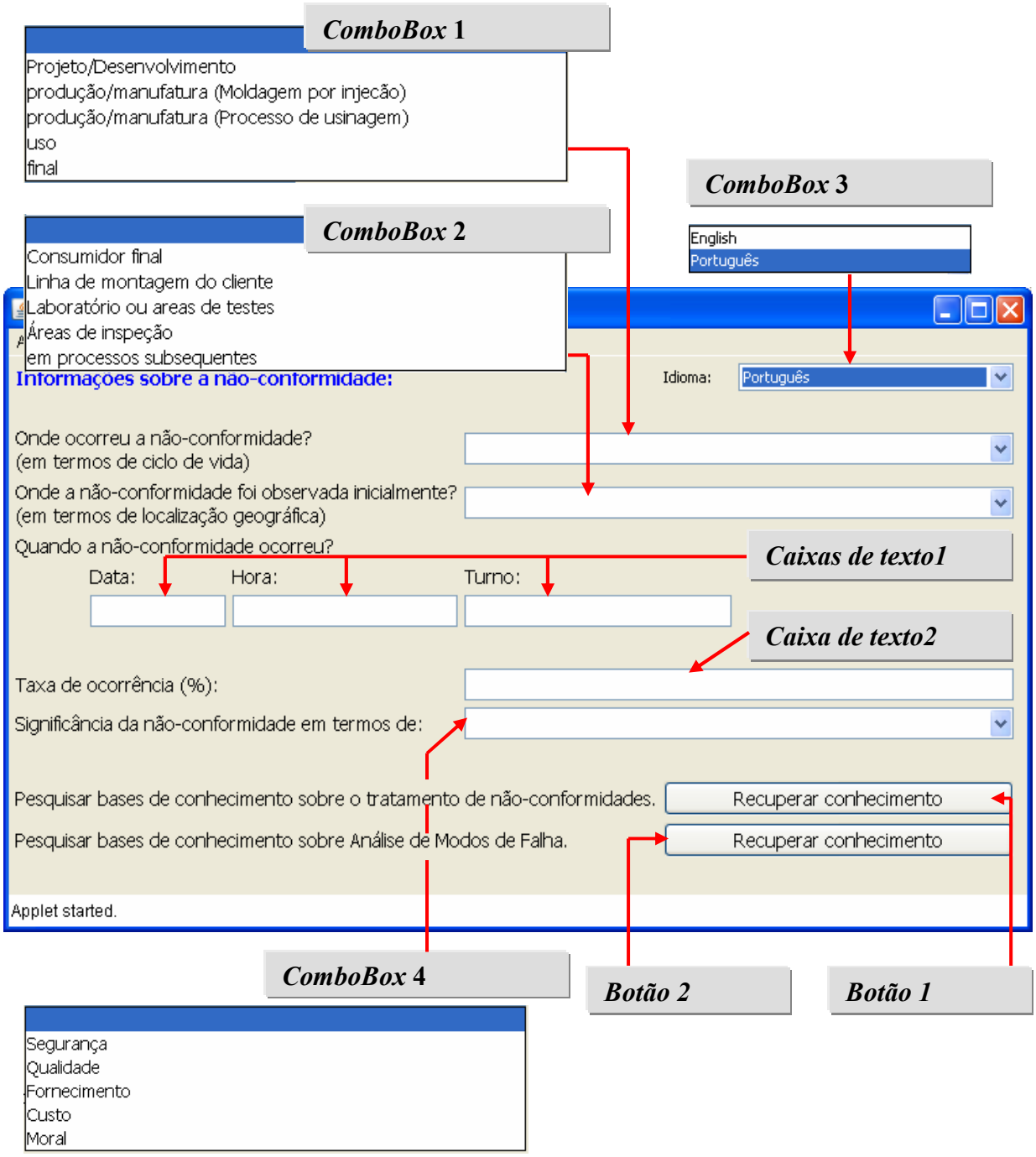


Figura 8.1 – Primeira janela de interface gráfica do protótipo do modelo.

Esta janela da interface gráfica compreende um conjunto de componentes *menus* tipo *ComboBox* que permitem ao usuário entre outras funções selecionar o idioma a ser usado no apoio ao tratamento de não-conformidades por meio do *ComboBox 3*, bem como informar outros detalhes sobre a ocorrência da não-conformidade.

Deve-se destacar a importância da seleção da etapa do ciclo de vida de um produto mediante o *ComboBox 1*, cujo propósito é determinar quais instâncias do agente de interface devem ser apresentadas ao usuário. Isto porque, cada instância deste agente é configurada para um domínio em especial em função da etapa em questão, como no seguinte exemplo: um agente de interface configurado para o processo de moldagem por injeção de termoplásticos, ou para um processo de co-injeção, ou processo de usinagem, etc.

Adicionalmente, outra faceta importante desta interface diz respeito à escolha dos botões 1 e 2 (figura 8.1), que definem qual a perspectiva de apoio 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.

Em termos metodológicos, a figura 8.2 apresenta o diagrama de seqüência AUML que modela esta interação inicial do usuário com os agentes de interface.

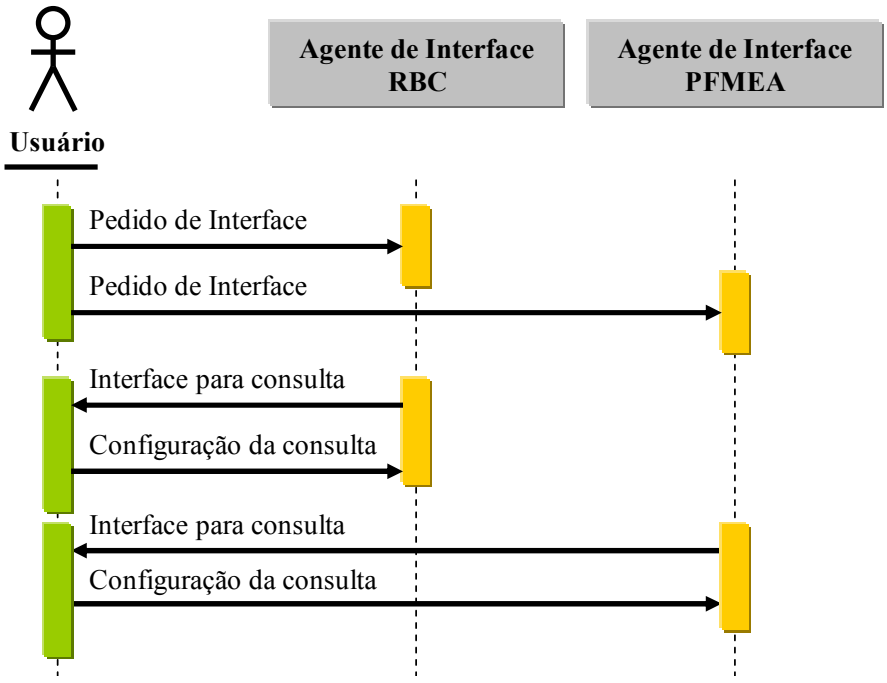


Figura 8.2 – Diagrama de seqüência AUML de interação entre o usuário e os agentes de interface.

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 de interface é enviado diretamente à instância do agente de interface apropriado. Deve-se lembrar que cada agente é configurado especialmente para a etapa do ciclo de vida selecionado no *ComboBox 1*, 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 de recursos, os quais que devem estar sempre ativos aguardando a submissão de uma consulta por meio do sistema de troca de mensagem 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., 2006a). 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 abstração de containeres 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 demonstrada na figura 8.3 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 é possível observar os diversos containeres 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 8.3 mostra, adicionalmente, o Agente Facilitador de Diretórios (*DF Agent*) que no protótipo em questão é responsável pelas tarefas do Agente *Matchmaker*.

Os containeres de 1 a 3 (figura 8.3) mostram a execução de três instâncias do agente de recurso RBC denominados *ColibriFinalAgent0*, *ColibriFinalAgent1* e *ColibriFinalAgent2*; 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 tempo ou mesmo retiradas de acordo com as necessidades organizacionais.

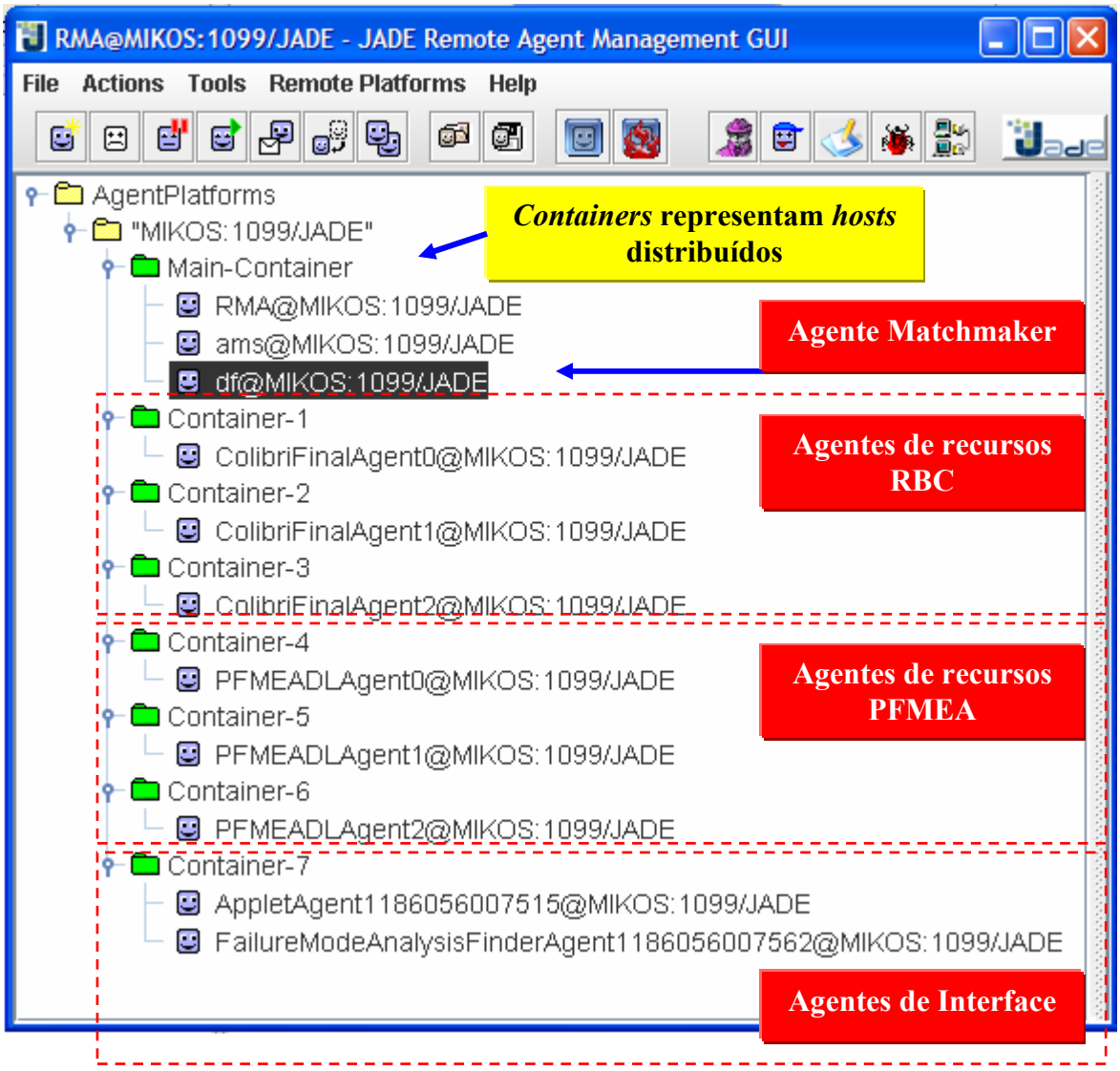


Figura 8.3 – Interface gráfica do Agente RMA registrando os diversos containeres e agentes.

Da mesma forma, os containeres de 4 a 6 (figura 8.3) 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.

Por fim, a figura 8.3 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., 2006a).

Ainda, outra questão a ser explorada neste trabalho diz respeito ao Agente *Matchmaker*, que na plataforma é representado pelo Agente DF, cuja função é modelada pelo diagrama de seqüência AUML mostrado na figura 8.4. 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 como controle.

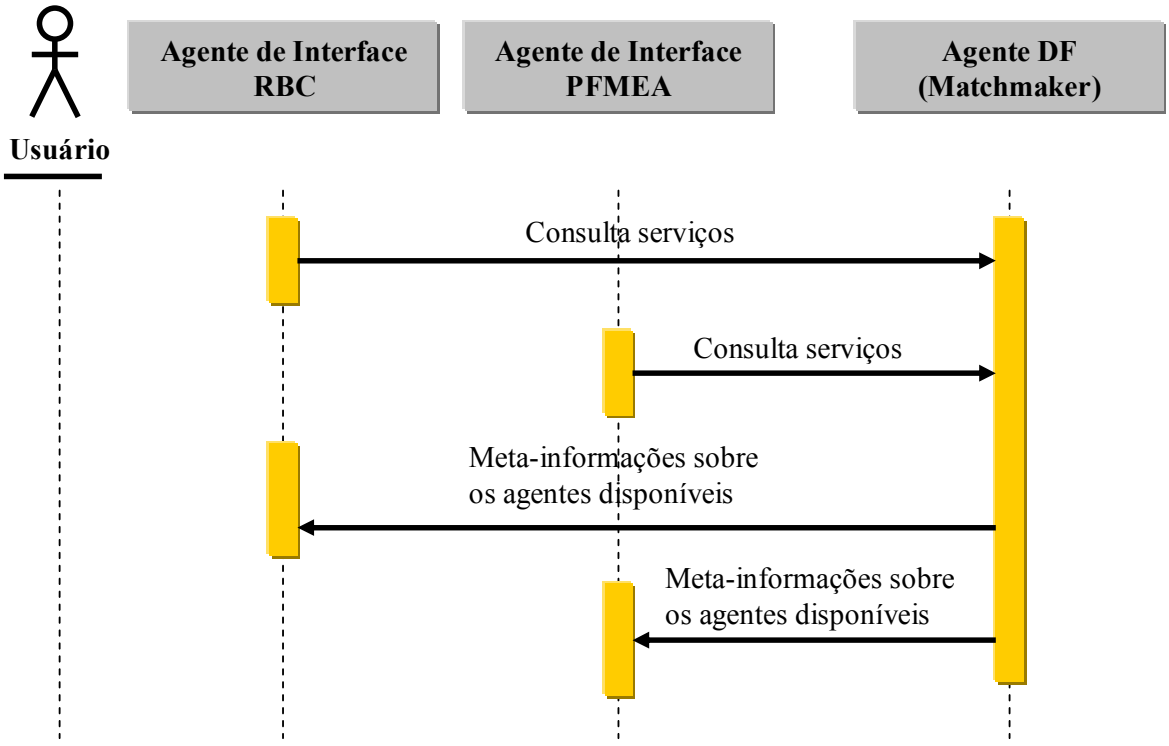


Figura 8.4 – Diagrama de seqüência AUML agente interface / agente DF.

Estas meta-informações incluem, em particular, o índice físico (*directory of agent identifiers* – AID) dos agentes de recursos registrados junto ao Agente DF, que dispõe do serviço requisitado pelos agentes de interface.

Na próxima subseção busca-se apresentar a verificação do modelo, mas por uma questão de coerência metodológica o processo será conduzido a partir da ótica dos agentes de recursos RBC e PFMEA, respectivamente.

8.1.1 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 decorrentes de experiências de soluções de problemas de não-conformidades que já ocorrem no passado é determinada pela ação do *Botão 1* (figura 8.1), que apresenta ao usuário a janela gráfica do agente de interface RBC, como mostra a figura 8.5.

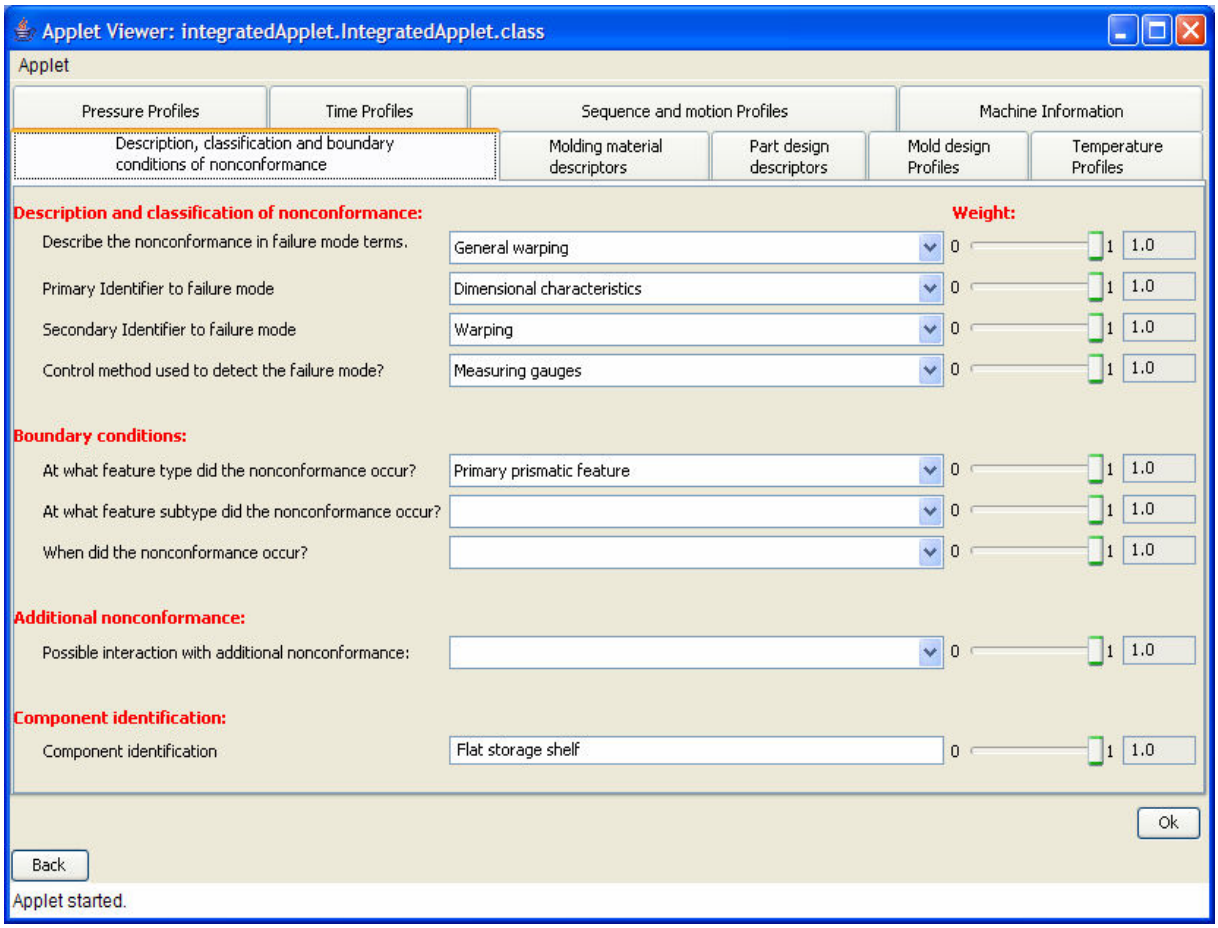


Figura 8.5 – Janela gráfica do agente de interface de RBC.

Nesta figura, a janela gráfica do agente de interface RBC já está configurada para o processo de moldagem por injeção. Particularmente, apresenta-se nesta figura, no idioma inglês, a aba correspondente a entrada de dados dos descritores referentes à descrição,

classificação e condições de contorno da não-conformidade sob consulta na *Java Applet*. Utiliza-se a língua inglesa porque o objetivo aqui é consultar as bases de conhecimento preenchidas com casos de testes obtidos da literatura em língua inglesa. Deve-se observar que, para manter a clareza do texto, as demais abas da interface são apresentadas na subseção 8.3.

A figura 8.6 apresenta o diagrama de seqüência 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.

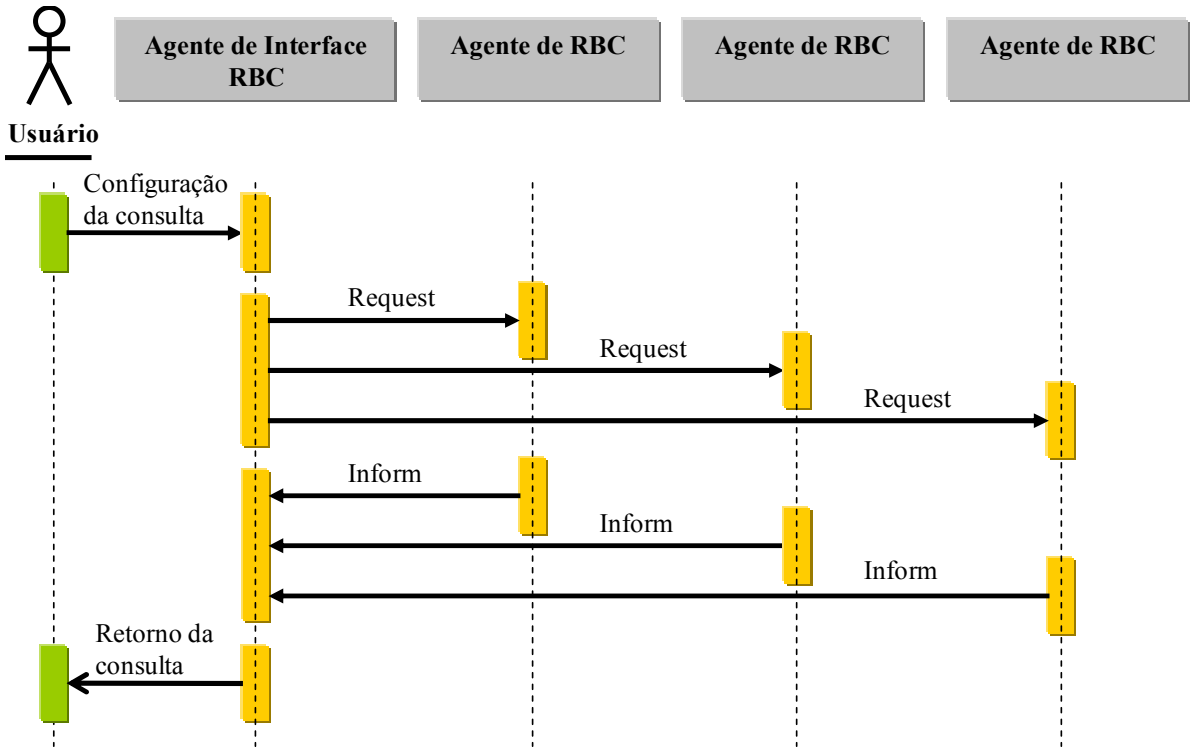


Figura 8.6 – Diagrama de seqüência AUML.

Como mostra o diagrama de seqüência, 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 comportamento apresentado e discutido no Capítulo 6.

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 dos casos mais similares recuperados, e 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 inteligível ao usuário por meio da segunda janela gráfica.

O desenvolvimento desta funcionalidade pode ser observado pela interface do agente *Sniffer*[1], uma das ferramentas disponíveis na plataforma JADE (BELLIFEMINE et al., 2006a) como mostra a figura 8.7.

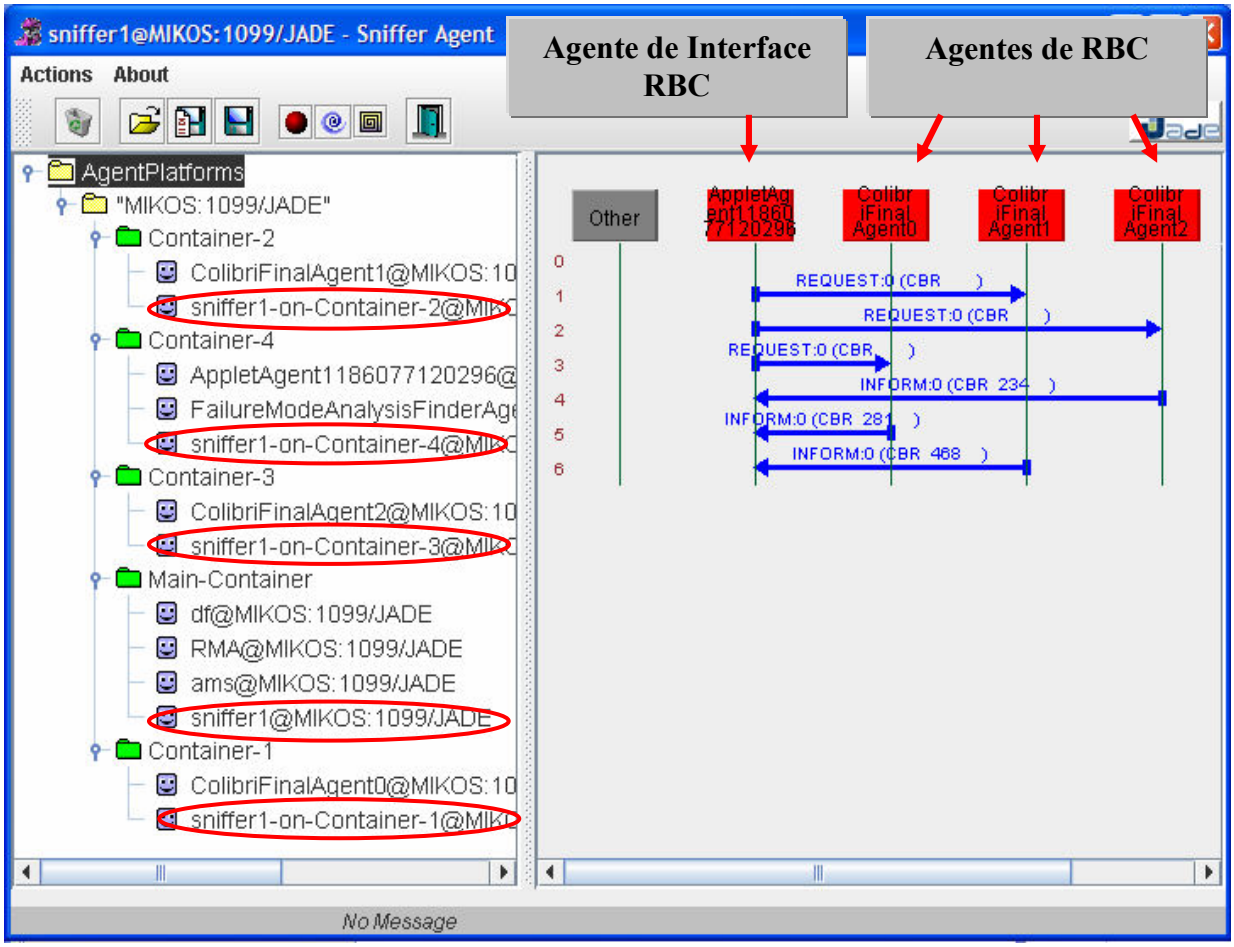


Figura 8.7 – Interface gráfica do Agente Sniffer.

Esta interface é devotada 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 é possível observar as interações e as troca de mensagens entre o agente de interface RBC denominado *AppletAgent* e os agentes de recursos RBC denominados como: *ColibrifFinalAgent0*, *ColibrifFinalAgent1*, *ColibrifFinalAgent2*, que se encontram ativos na plataforma.

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

A interface do agente *Sniffer* permite ainda verificar o conteúdo das mensagens trocadas pelos agentes, como mostram a figuras 8.8 e 8.9 para o ato comunicativo *Request* e *Inform* respectivamente, neste caso representado pelos mapas *hash* na forma de objetos da classe `java.util.HashMap` (SUN DEVELOPER NETWORK, 2007).

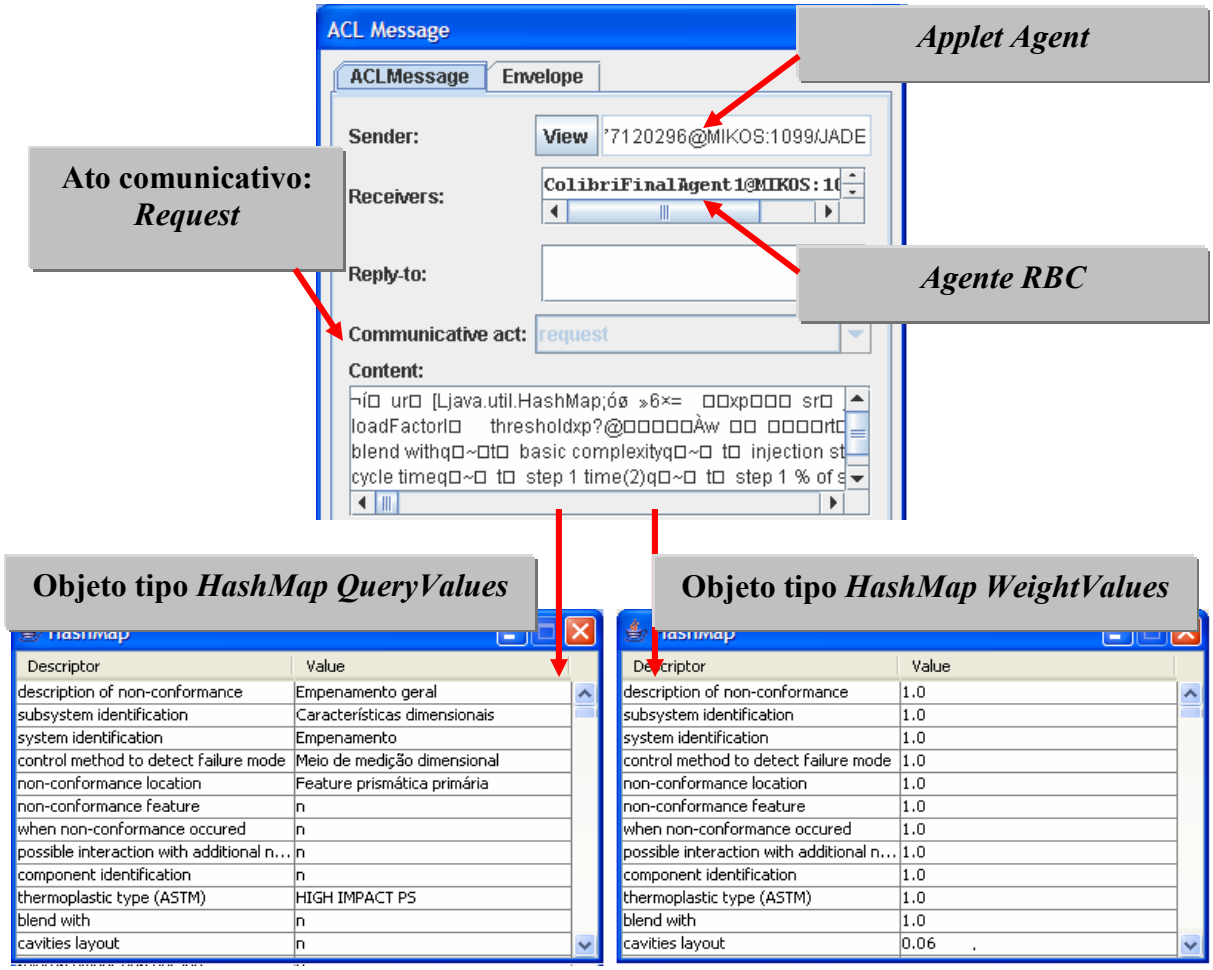


Figura 8.8 – Conteúdo do ato comunicativo *Request*: objetos tipo *HashMap*

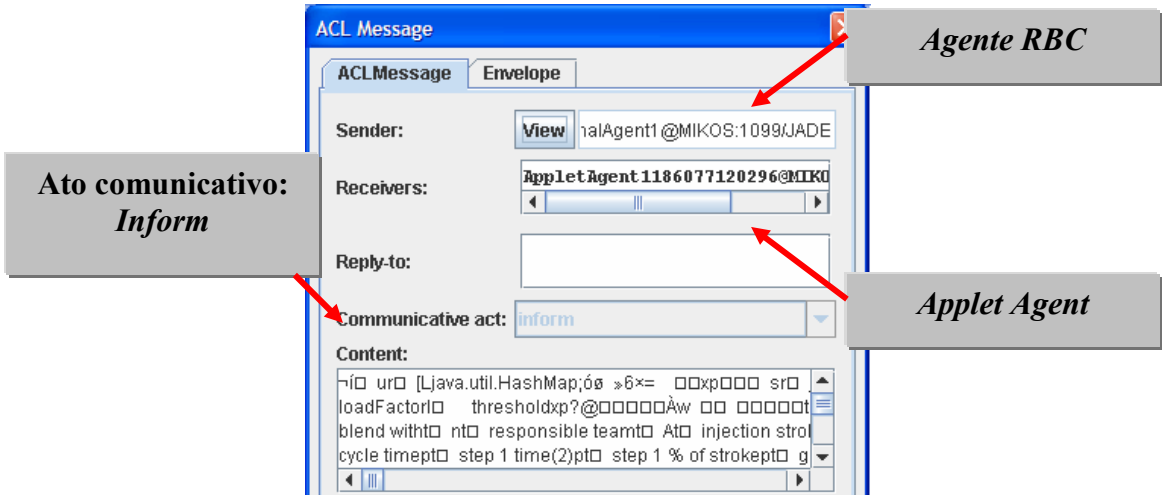


Figura 8.9 – Conteúdo do ato comunicativo *Inform*: objetos tipo *HashMap*

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 8.10 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, de acordo com o modelo de tarefas apresentado e discutido no Capítulo 6.

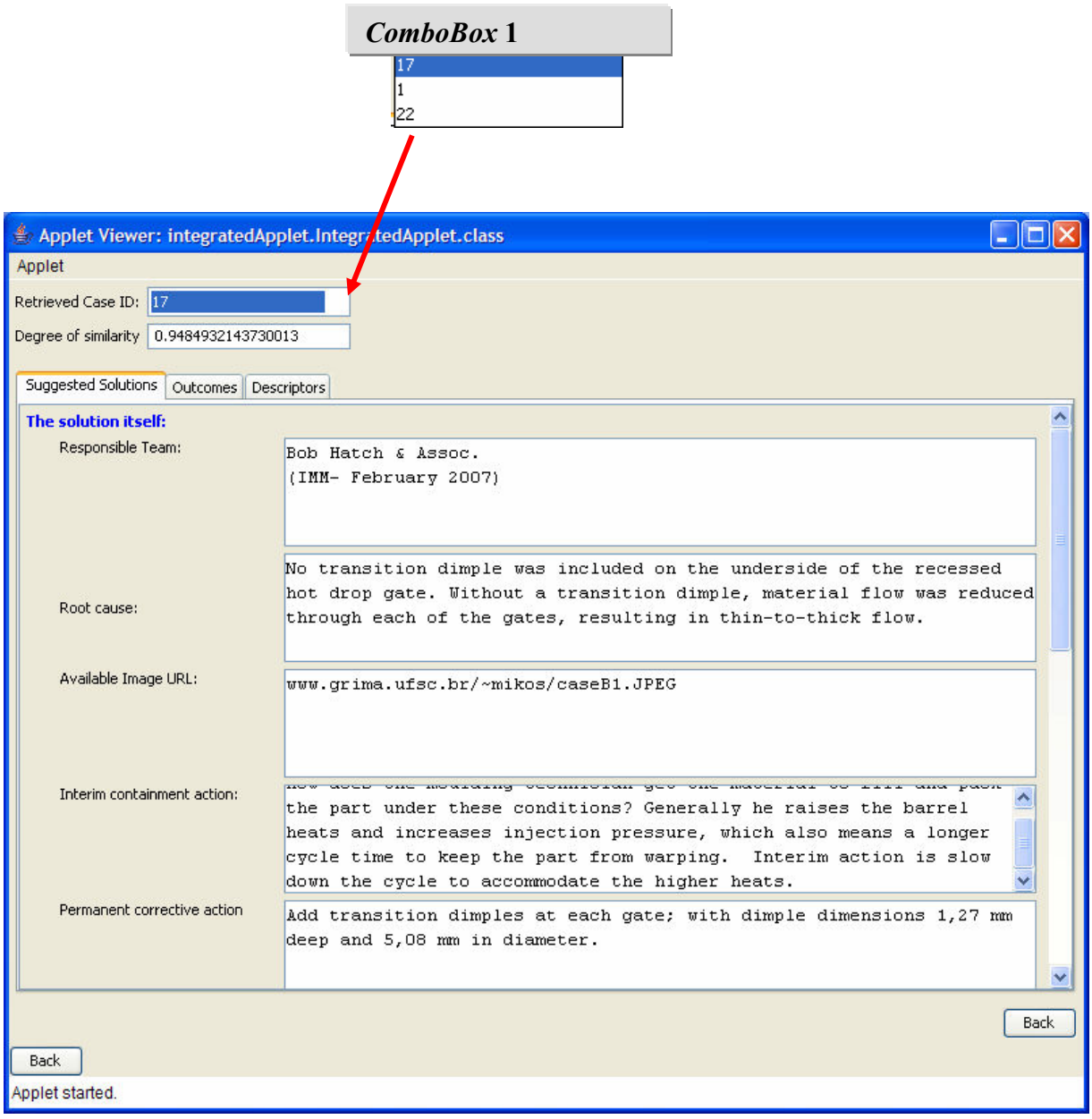


Figura 8.10 – Janela gráfica para apresentação dos casos recuperados.

Cumpre observar que, para manter a clareza do texto desta subseção, as demais abas deste conjunto de interfaces gráficas são apresentadas em detalhes na etapa de validação operacional do modelo discutida na subseção 8.3.

Com relação aos agentes de recurso 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 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 8.10.

Dentro desta 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 cômputo geral da similaridade. É importante destacar que neste protótipo o valor do peso (w_i) pode ser ajustado dinamicamente a cada nova consulta em função das preferências do usuário.

Para um dado descritor x_i a similaridade entre os dados atômicos do tipo *string* é calculada pela correspondência exata dos *strings* do novo caso e do caso da base, lembrando que os valores dos descritores não numéricos são selecionados a partir de menus tipo *ComboBox* para assegurar a consistência destes dados.

A tabela 8.1 apresenta os casos recuperados (considerando a consulta configurada pelo usuário no agente de interface RBC da figura 8.5), bem como os valores das medidas de similaridade calculadas pelos agentes de recursos RBC usando o algoritmo *nearest neighbour* normalizado, o somatório dos pesos (w_i) e o número de descritores usados no cálculo.

Esta tabela mostra que o caso 17 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 0,9484932143730013, isto é 94,84 % de coincidência entre os seus descritores e os descritores da consulta configurada pelo usuário.

Tabela 8.1 – Medida de Similaridade Global (algoritmo *nearest neighbour* normalizado)

Caso (Id)	Valor das medidas de similaridade entre os casos
17	02/08/2007 20:31:40 jcolibri.util.CBRLogger log INFO: Similarity with case 17 :0.9484932143730013 Somatório dos pesos (w_i): 76.29999999999998 (114 descritores)
1	02/08/2007 20:31:40 jcolibri.util.CBRLogger log INFO: Similarity with case 1 :0.9437596663429596 Somatório dos pesos (w_i): 76.29999999999998 (114 descritores)
22	02/08/2007 20:31:40 jcolibri.util.CBRLogger log INFO: Similarity with case 22 :0.9175473465526586 Somatório dos pesos (w_i): 76.29999999999998 (114 descritores)

Por sua vez, a tabela 8.2, por sua vez, avalia a sensibilidade das medidas de similaridade, característica fundamental do método RBC implementado no comportamento dos agentes de recursos, a qual é responsável por ordenar os casos similares recuperados. Esta tabela apresenta nas colunas os valores das medidas de similaridade calculadas usando-se o algoritmo *nearest neighbour* normalizado, os valores das mesmas medidas de similaridade, porém sem a normalização (isto é, sem efetuar a divisão pelo número de descritores), bem como a diferença entre os casos recuperados em termos do número de descritores que diferem entre eles.

Portanto, é possível observar que os casos 17 e 1 divergem entre si por uma diferença encontrada somente em um descritor numérico em cujo cálculo usou-se a função linear (equação 3 do capítulo 4). E, também, que os casos 1 e 22 divergem entre si por uma diferença com valor inteiro que corresponde a dois descritores não numéricos.

Tabela 8.2 – Sensibilidade da Medida de Similaridade Global (algoritmo *nearest neighbour*)

Caso	Similaridade Global (algoritmo normalizado)	Similaridade Global (algoritmo não normalizado)	Diferença em termos de descritores divergentes
17	0,9484932143730013	72,3700322566599	0,36116971469218
1	0,9437596663429596	72,0088625419677	2
22	0.9175473465526586	70,0088625419677	

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 17 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.

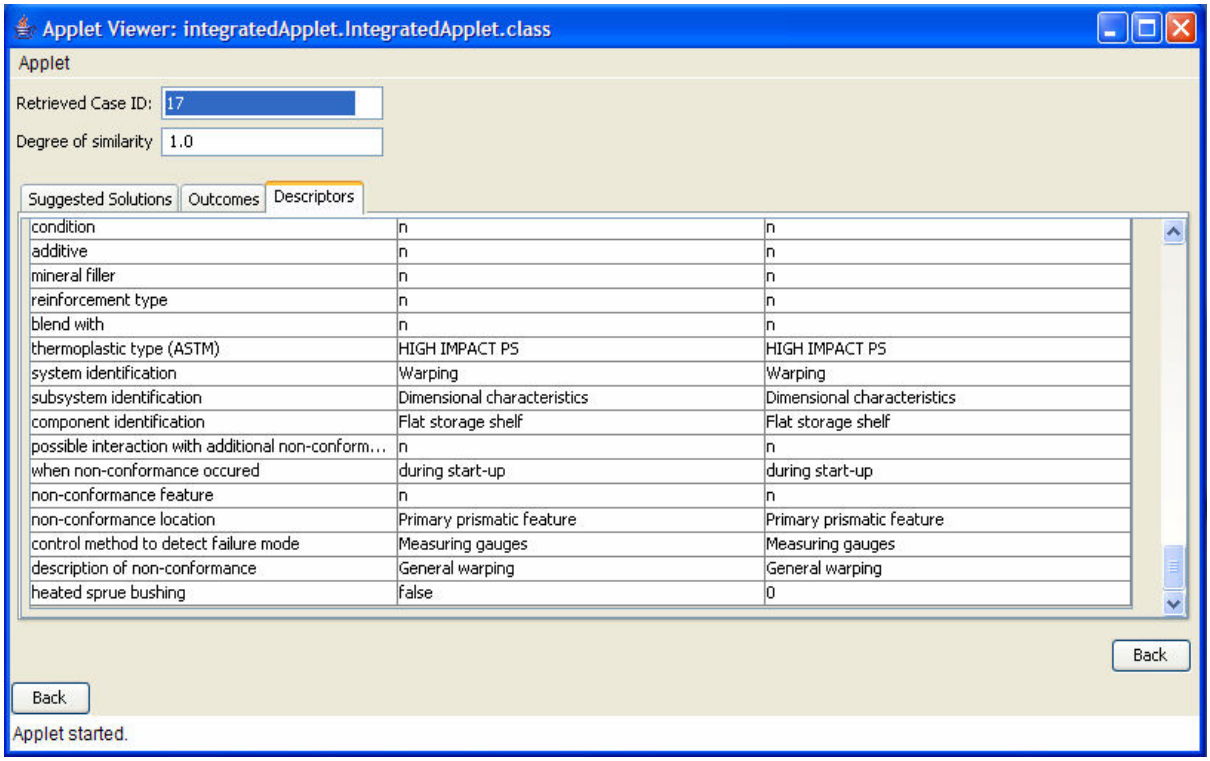


Figura 8.11 – Verificação da medida de similaridade para descritores idênticos.

Ao submeter-se 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 17 com uma medida de similaridade global com valor igual a 1,0, que demonstra a total coincidência entre os descritores, e isso foi o exatamente o que aconteceu, como mostra a figura 8.11.

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. E 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 destinadas 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 moldagem por injeção de termoplásticos.

8.1.2 Verificação das funcionalidades do protótipo do modelo para os agentes 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 8.1). Esta ação determina a apresentação ao usuário da janela gráfica do agente de interface PFMEA, como mostra a figura 8.12.

É importante notar que a figura 8.12 apresenta a janela gráfica do agente de interface PFMEA na versão para o idioma inglês, uma vez que este foi o idioma dos casos de testes escolhidos para o processo de verificação em questão, os quais foram operacionalizados no Capítulo 7. Nesta figura, em particular, apresenta-se a estratégia de recuperação de conhecimento baseado em ontologias, implementada no modelo de tarefas do agente de interface PFMEA conforme apresentada no Capítulo 6.

Esta estratégia baseia-se na configuração dinâmica de uma consulta em linguagem natural, mais precisamente na forma de uma frase na língua inglesa (***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 seqüência de conceitos da Ontologia PFMEA DL, conforme apresentados no Capítulo 4, os quais são disponibilizados ao usuário ao longo da estrutura da frase em diversos menus tipo *ComboBox*, como mostra a figura 8.12.

Deve-se lembrar, que a escolha de um conceito em 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. Como por exemplo, se no primeiro menu *ComboBox* o usuário selecionar o conceito *Potential Failure Mode* no menu *ComboBox3* este conceito não pode ser apresentado, pois senão teríamos uma inconsistência semântica.

A figura 8.12 ilustra 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.

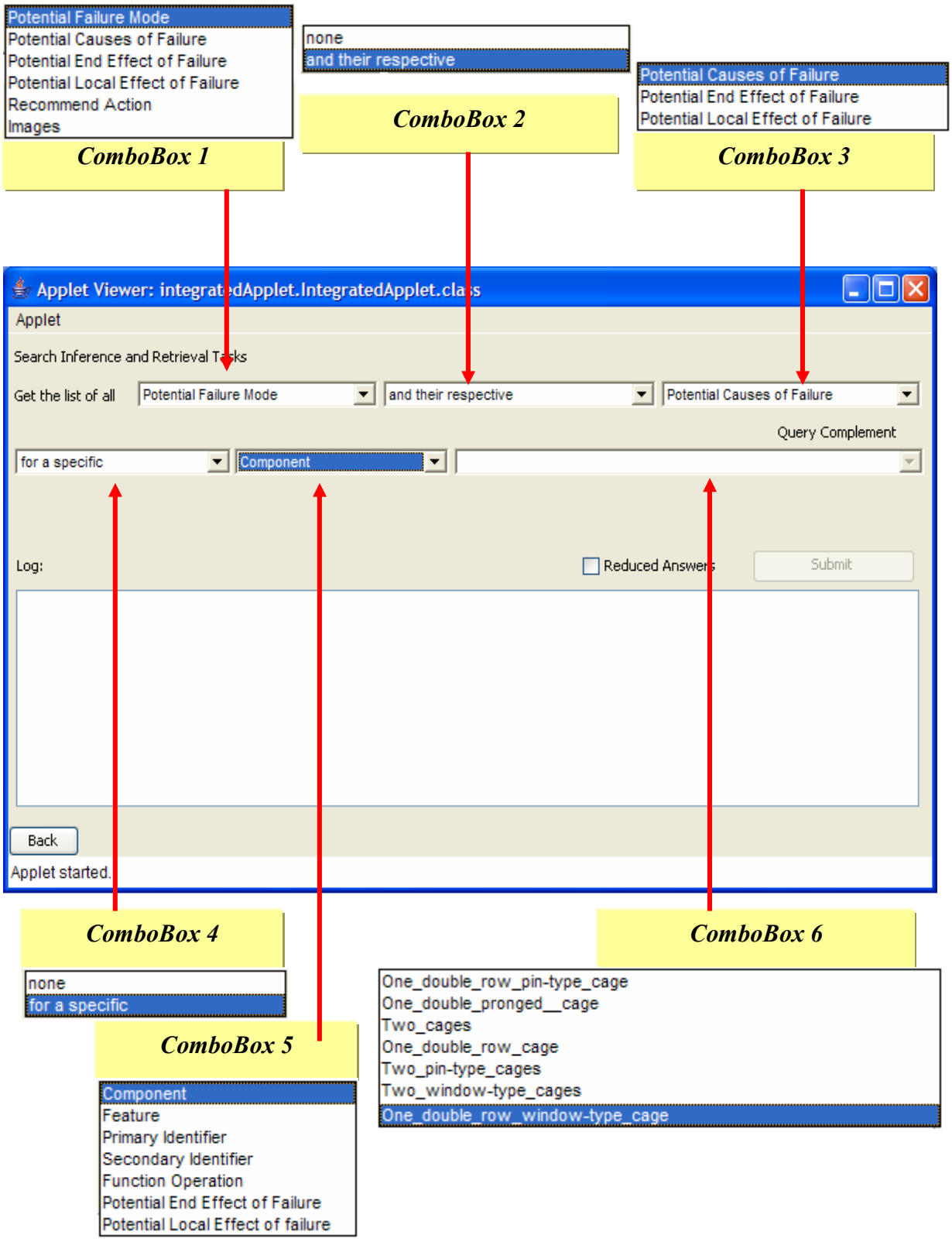


Figura 8.12 – Janela gráfica do agente de interface PFMEA.

A figura 8.13 apresenta o diagrama de seqüência AUML que modela a interação entre o usuário, o agente de interface PFMEA e os agentes de recursos PFMEA ativos na plataforma e identificados por meio da interação com o agente DF.

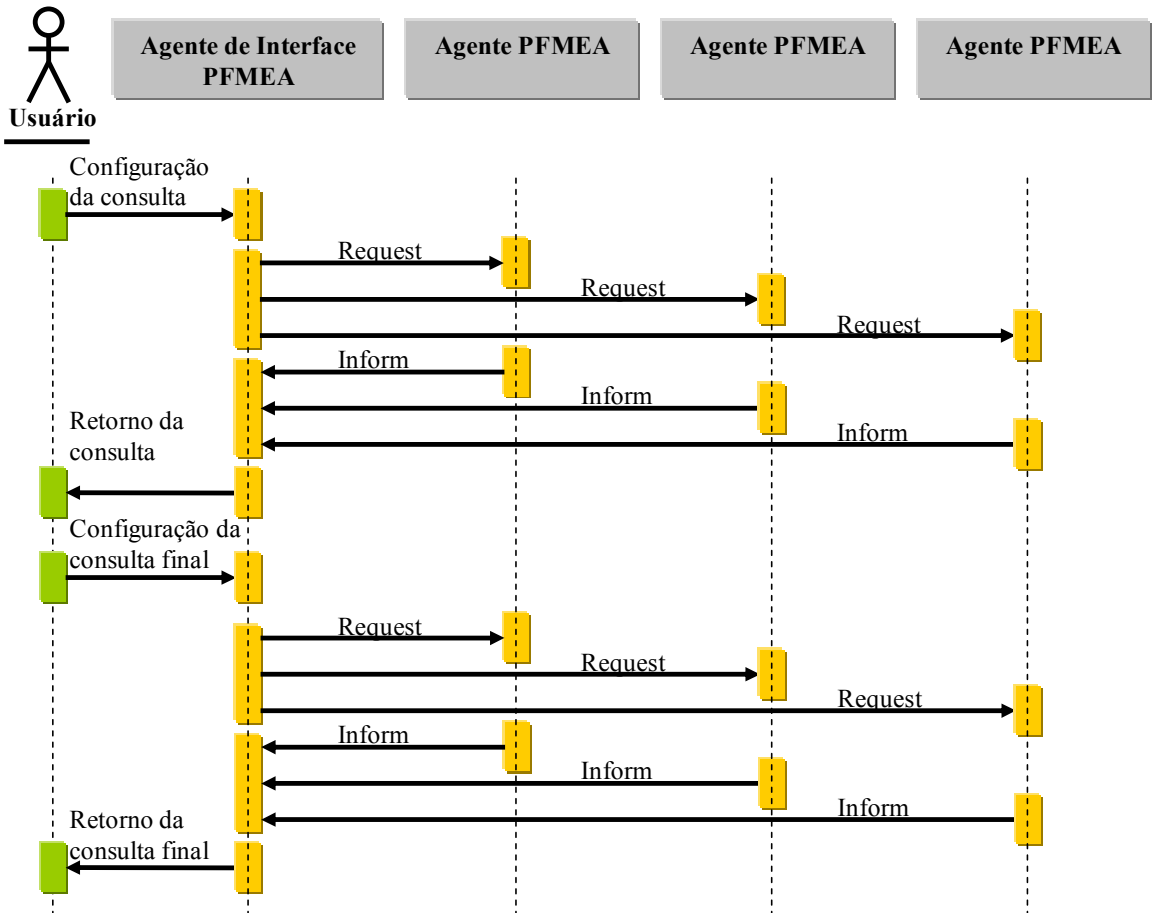


Figura 8.13 – Diagrama de seqüência AUML usuário / agente de interface / agentes de recursos.

A implementação desta funcionalidade pode ser comprovada pela interface do agente *Sniffer* (BELLIFEMINE et al., 2006a) mostrada na figura 8.14. 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.

Como ilustra o diagrama de seqüência (figura 8.13), levando em conta as especificações de projeto do modelo, a configuração desta consulta é realizada em dois estágios: no primeiro o agente de interface deve recuperar todas as instâncias do conceito (ou subclasse OWL DL) *ComponentLevel* submetendo uma consulta complementar (*Query complement*) aos agentes de recursos PFMEA. Esta recuperação deve ocorrer de modo autônomo e transparente ao

usuário, mediante a troca de mensagens FIPA-ACL com o ato comunicativo ***Request***. E o agente de interface deve transcrever a consulta complementar para a sintaxe nRQL (*new Racer Query Language*), através da qual os agentes envolvidos são capazes de decodificar e analisar sintaticamente o conteúdo dos atos comunicativos em questão.

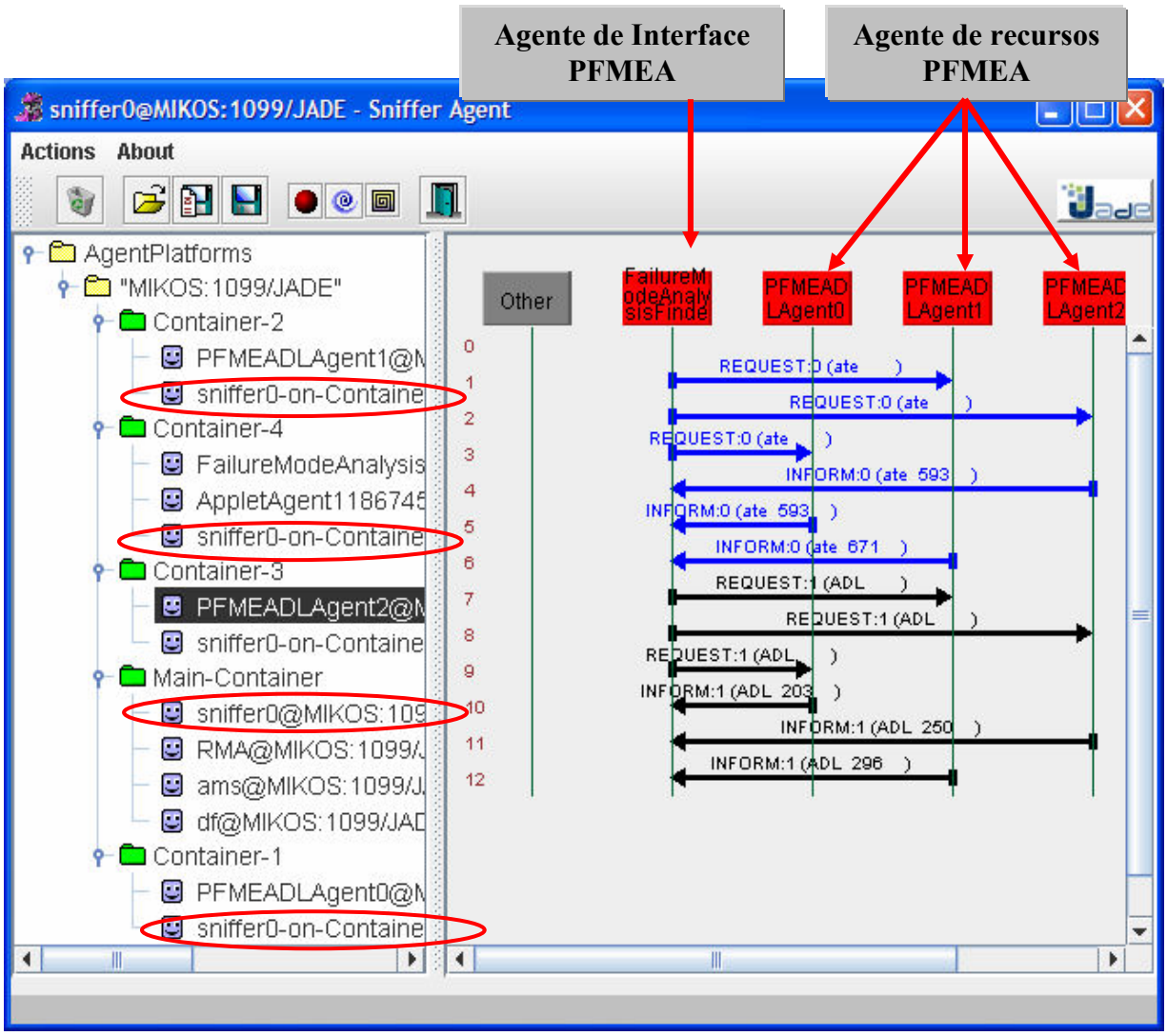


Figura 8.14 – Interface gráfica do Agente *Sniffer*.

Então, os agentes de recursos ativos respondem à consulta complementar com outras mensagens FIPA-ACL, usando agora o ato comunicativo ***Inform***, em cujos conteúdos encontram-se as instâncias disponíveis nas bases de conhecimento de cada agente. Por sua vez, o agente de interface deve converter os conteúdos das mensagens e apresentá-los ao usuário no *ComboBox 6*.

Por fim, no segundo estágio, o usuário pode configurar a consulta em sua forma final e submetê-la pelo agente de interface, o qual deve repetir os passos usados no primeiro estágio. Isto é, a consulta é transcrita para a sintaxe nRQL e é enviada como conteúdo da mensagem

FIPA-ACL com o ato comunicativo ***Request***. E, então, aguardar pelas mensagens FIPA-ACL com o ato comunicativo ***Inform*** para, então, transcrevê-las novamente e apresentá-las ao usuário.

A interface do agente *Sniffer* permite ainda verificar o conteúdo das mensagens trocadas pelos agentes, como mostram as figuras: 8.15, 8.17, 8.19 e 8.21 para os atos comunicativos *Request* e *Inform.*

A figura 8.15 mostra a estrutura da mensagem FIPA-ACL, enquanto a figura 8.16 apresenta o conteúdo da mensagem expressa em um conjunto de caracteres (*strings*) concatenados na forma de uma "linha de comando" de acordo com sintaxe da linguagem nRQL conforme apresentado no capítulo 6. Para maiores detalhes, sobre a linguagem nRQL sugere-se consultar a documentação disponível em RACER SYSTEM (2005).

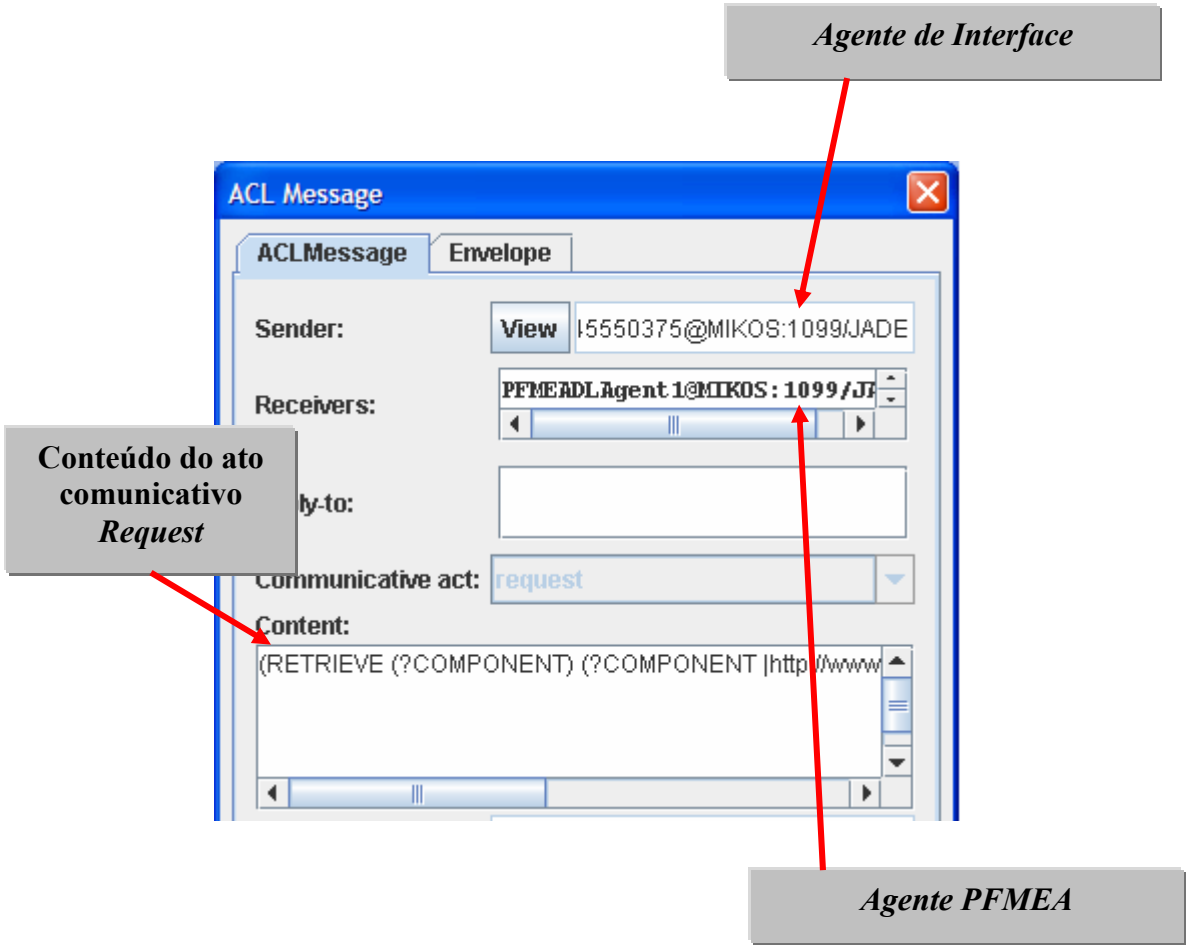


Figura 8.15 – Interface gráfica do Agente *Sniffer* - ACL Message.

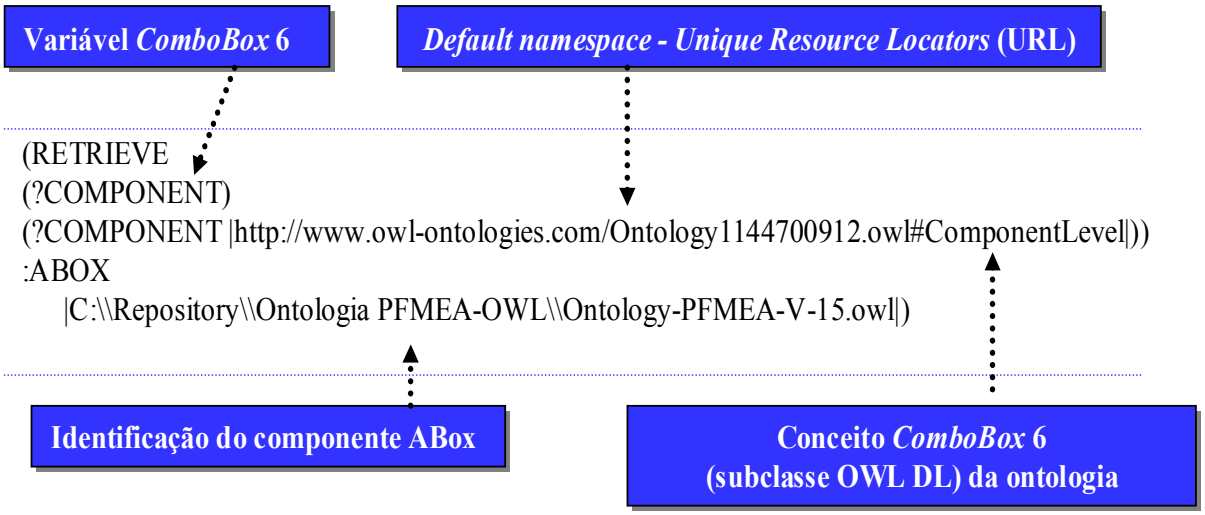


Figura 8.16 – Consulta complementar de acordo com a sintaxe nRQL.

Todavia, é oportuno destacar que a URL usada para a representação dos conceitos (ou subclasse OWL DL) corresponde ao *namespace* padrão gerado pelo editor PROTÉGÉ OWL, usado para evitar coincidências entre a definição de classes, propriedades e indivíduos com outras ontologias. Além, da indicação do o nome e a localização do arquivo (OWL) relativo ao componente ABox correspondente às instâncias (indivíduos) e que é personalizada para cada base de conhecimento.

A figura 8.17 apresenta a estrutura da mensagem FIPA-ACL enviada pelo agente de recurso PFMEA denominado *PFMEADLAgent1*, como indica o campo *Sender* do envelope da mensagem. E a figura 8.18, por sua vez, detalha o conteúdo da mensagem enviada pelo agente em questão, e que corresponde a todas as instâncias representadas na parte ABox da base de conhecimento deste agente em particular.

Deste modo, o agente de interface PFMEA deve converter e tratar este conteúdo antes de apresentá-lo ao usuário no *menu* tipo *ComboBox 6*, isto porque o agente de interface deve aguardar as demais mensagens enviadas pelos outros agentes. E, para isso as mensagens são filtradas para evitar a apresentação de instâncias repetidas, bem como tratar as respostas dos agentes de recursos que se não possuem nenhuma instância que satisfaça a consulta.

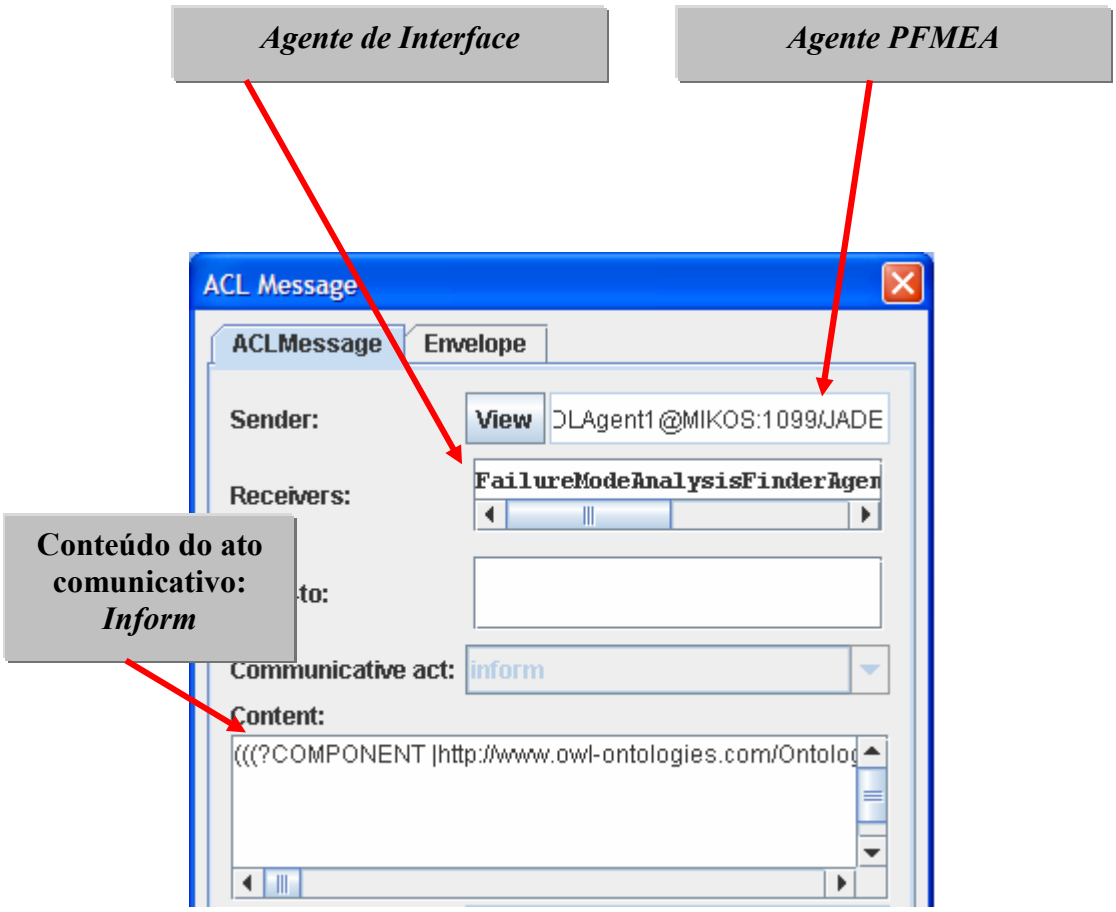


Figura 8.17 – Interface gráfica do Agente *Sniffer* ACL Message.

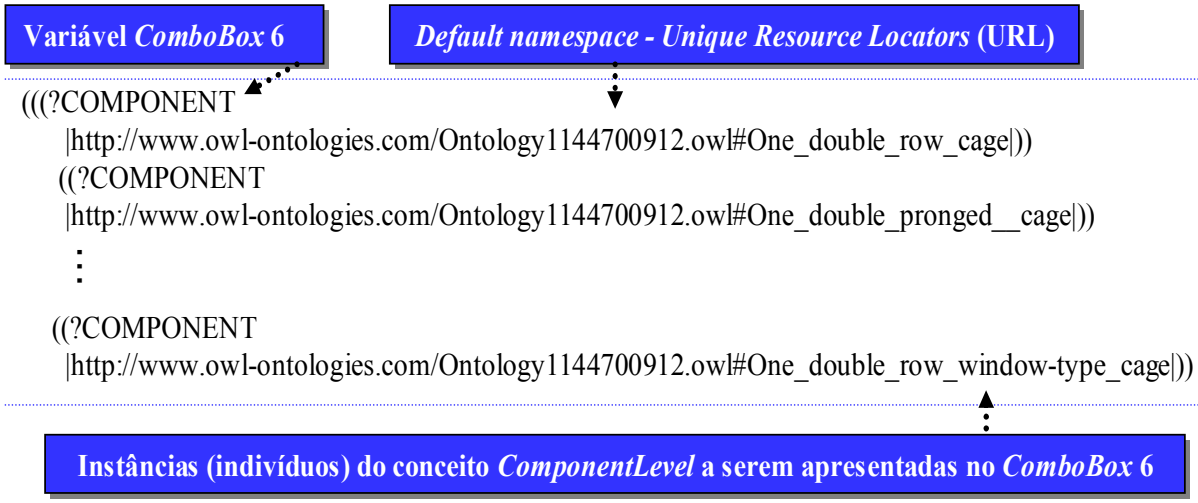


Figura 8.18 – Resposta da consulta complementar de acordo com a sintaxe nRQL.

As figuras 8.19 e 8.20 mostram a estrutura da mensagem FIPA-ACL e a sintaxe nRQL para o segundo estágio da consulta, enviada, em particular, ao *PFMEADLAgent1*.

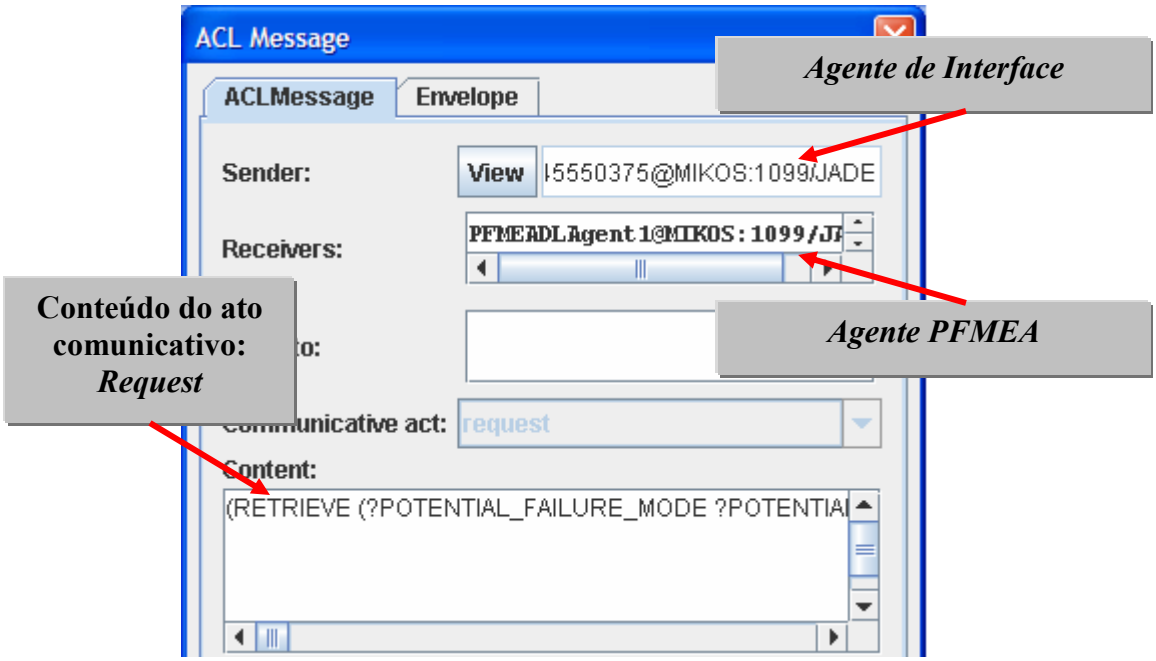


Figura 8.19 – Interface gráfica do Agente *Sniffer* ACL Message.

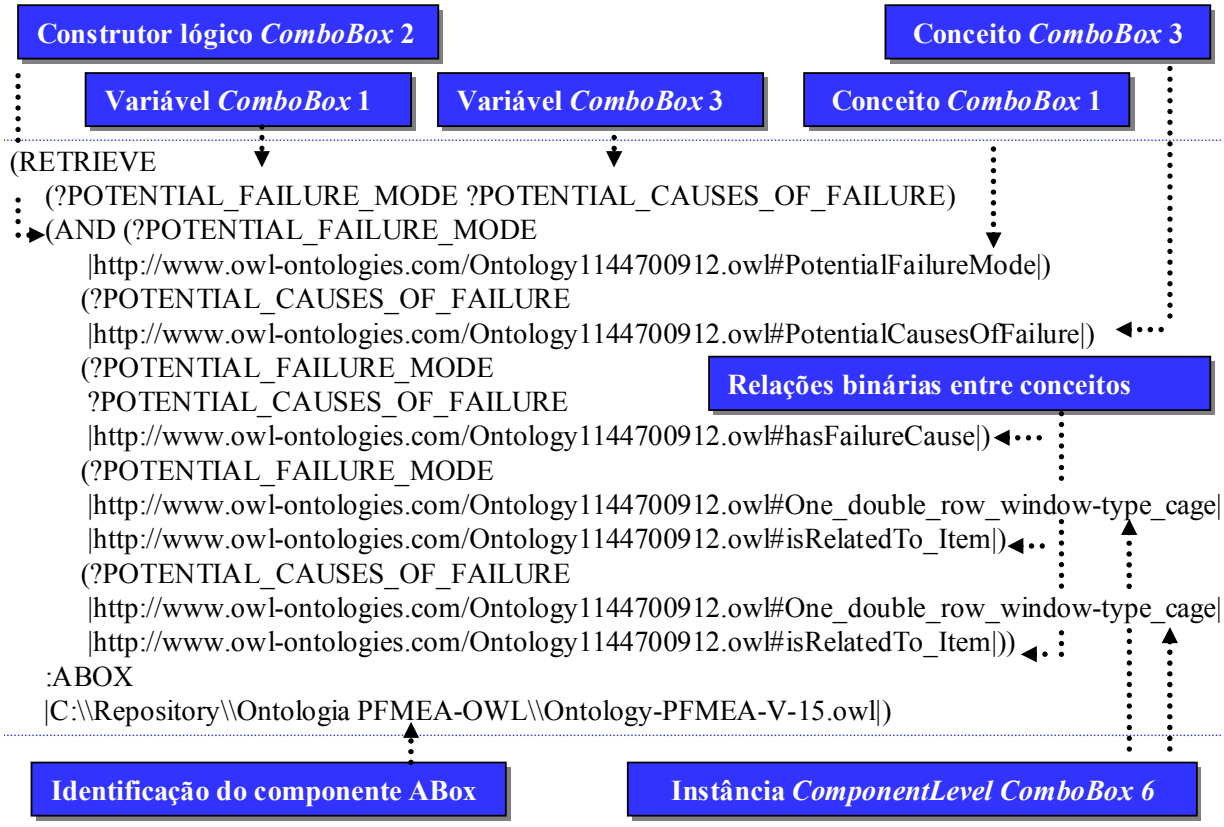


Figura 8.20 – Consulta final de acordo com a sintaxe nRQL.

A sintaxe nRQL permite a construção de consultas complexas para o componente ABox a partir de conceitos atômicos (*concepts query atoms*), relações binárias ou papeis atômicos (*role query atoms*) ou a combinação deste por meio de operadores lógicos típicos da álgebra relacional.

A figura 8.20 descreve a construção de consulta usando os conceitos (*concepts atoms*) (#PotentialFailureMode) e (#PotentialCausesOfFailure) que estão relacionados pela propriedade ou relação binária (*role atoms*) (#hasFailureCause). Esta descrição é feita particularmente para a instância específica (#One_double_row_window-type_cage), que está relacionada aos conceitos em questão pela relação binária (#isRelatedTo_Item). Deve-se destacar que os detalhes sobre os conceitos e relações binárias modeladas na ontologia PFMEA DL podem ser observados nas discussões apresentadas no Capítulo 4.

Neste mesma linha, as figuras 8.21 e 8.22 mostram a estrutura da mensagem enviada pelo agente de recurso *PFMEADLAgent1* ao agente de interface e o respectivo conteúdo da mensagem.

Por fim, o agente de interface deve aguardar as mensagens dos agentes de recursos, as quais devem ser transcritas e organizadas de modo a tornarem-se inteligíveis ao usuário e, em seguida, apresentá-las na interface gráfica do usuário como mostra a figura 8.23.

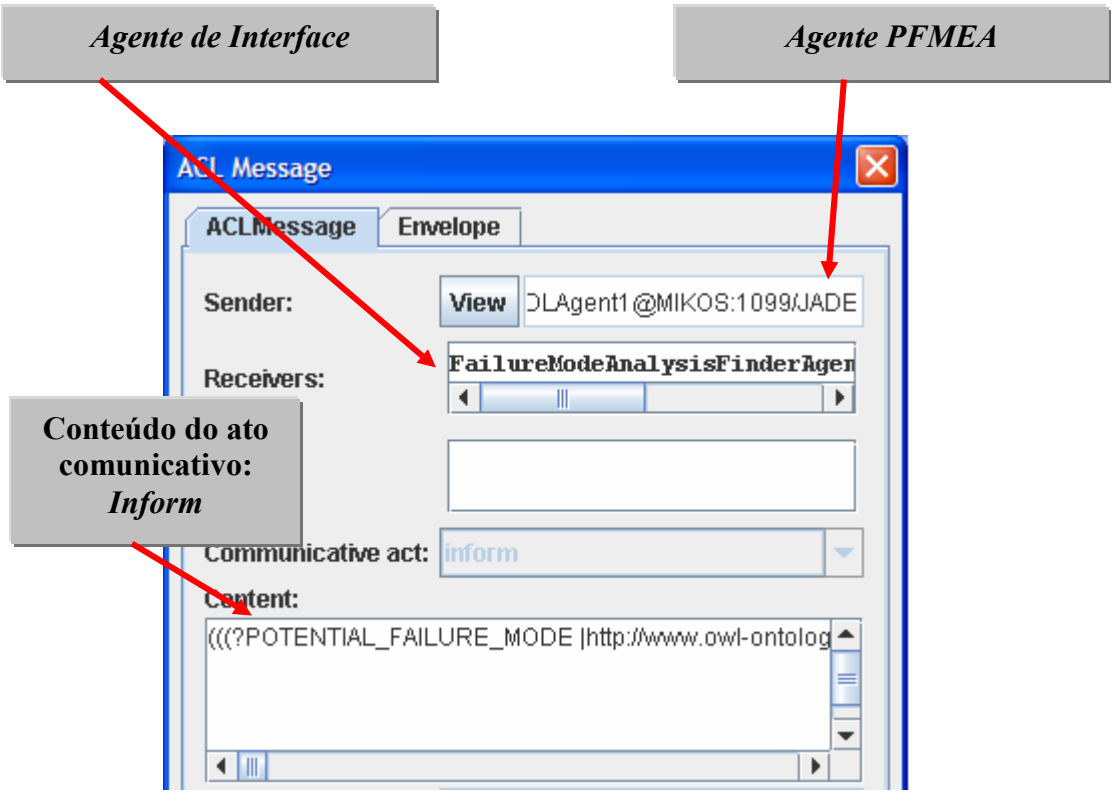


Figura 8.21 – Interface gráfica do Agente *Sniffer* - ACL Message.

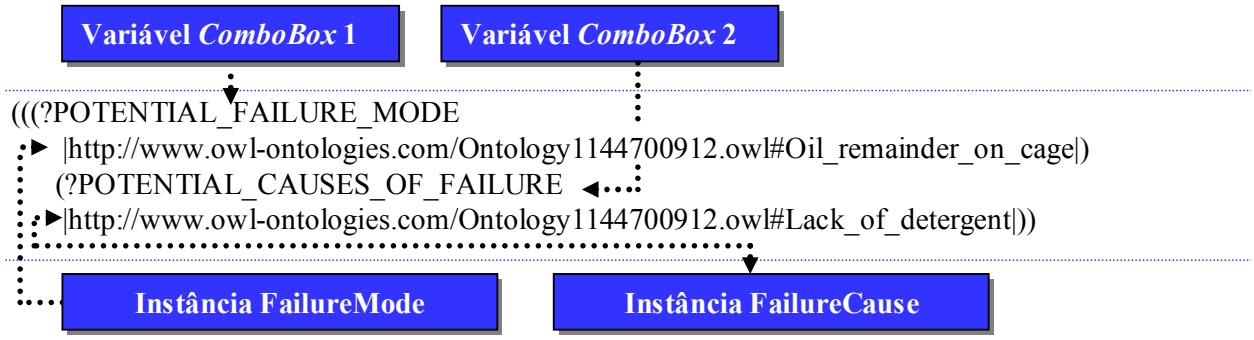


Figura 8.22 – Resposta da Consulta final de acordo com a sintaxe nRQL.

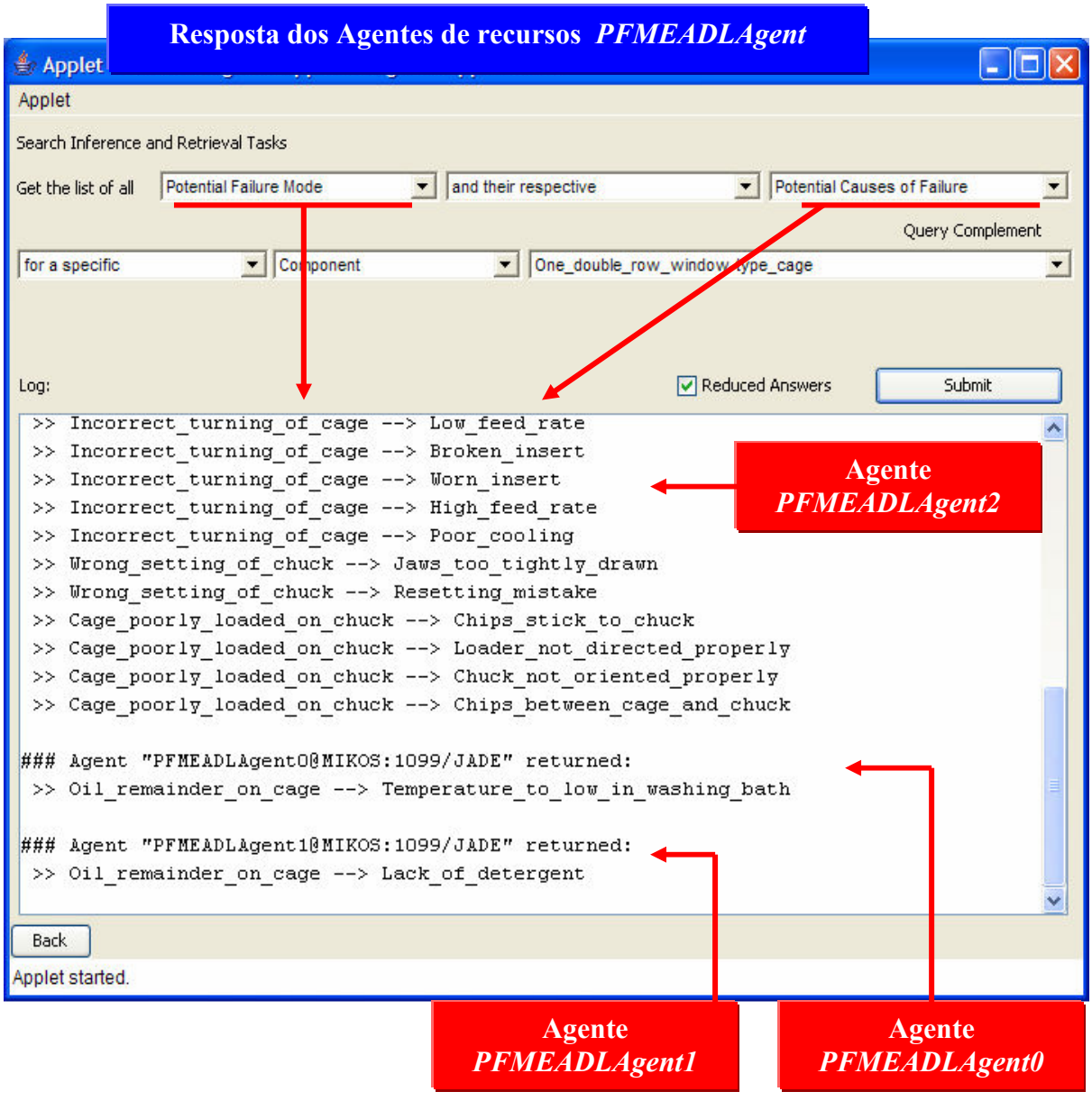


Figura 8.23 – Apresentação da resposta da consulta final na janela do agente de interface.

Em resumo, os experimentos realizados durante o processo de verificação das funcionalidades dos agentes de recursos PFMEA demonstraram que o protótipo do modelo foi implementado de modo satisfatório do ponto de vista computacional considerando as especificações de projeto.

Especial atenção foi dispensada ao comportamento dos agentes em relação à recuperação de conhecimento a partir de bases de conhecimento ontológicas distribuídas, às interações entre usuário, agente de interface e agentes de recursos, bem como às trocas de mensagens e entre os agentes propostos.

O modelo proposto pode comportar outras instâncias dos agentes de recursos PFMEA destinados a outros processos de manufatura, e para isso é necessário somente representar novas instâncias na parte *extensional* do ABox ou indivíduos (nas definições de classes OWL DL) da ontologia de acordo com a etapa do ciclo que se deseja modelar. Desta forma não será necessário alterar o componente TBox ou (definições de classes OWL DL). Isto porque o componente TBox da ontologia proposta foi desenvolvido para ser reutilizado em outras bases de conhecimento com a adição de novo componente ABox, fato este demonstrado aqui pelo uso de instâncias (ou indivíduos) no componente ABox, decorrentes da análise do modo de falhas e efeitos para outros processos de manufatura de uma empresa produtora de componentes automotivos no âmbito de um projeto SeisSigma (*SixSigma*) apresentado na literatura por Lennartsson e Vanhatalo (2004) como apresentado no Capítulo 7.

8.2 VALIDAÇÃO CONCEITUAL DO MODELO

Esta subseção discute a validade conceitual do modelo proposto seguindo a linha apresentada nos procedimentos metodológicos no Capítulo 3, mas por uma questão de coerência o processo de validação conceitual também serão conduzidos a partir do prisma dos agentes de recursos RBC e PFMEA, respectivamente.

A questão fundamental a ser colocada é: em que medida o conhecimento social representado no modelo conceitual reflete os construtos elaborados pelos especialistas nos domínios. Neste sentido, este conhecimento social pode ser entendido a partir de duas dimensões: a primeira em relação às bases de conhecimento específicas dos agentes, e a segunda em relação às redes de interações sociais e de comunicações empregadas na da organização dos agentes propostos.

Deve-se notar que para manter a clareza do texto a avaliação da fidedignidade das respostas dos serviços de raciocínio e dos métodos de recuperação de casos encapsulados nos comportamentos dos agentes de recursos RBC e PFMEA, baseou-se nos experimentos

realizados com os conjuntos de casos de teste usados durante a etapa de verificação das funcionalidades do protótipo do modelo, bem como nos resultados dos serviços e métodos analisados neste processo.

8.2.1 Validação conceitual do modelo sob o prisma dos agentes de recursos RBC

A primeira dimensão do conhecimento social representado no modelo conceitual sendo tratado diz respeito às bases de conhecimento específicas dos agentes. Neste sentido, foi possível comprovar que a estrutura conceitual do caso de não-conformidade proposta, representada pelos descritores, ajusta-se adequadamente aos relatos das experiências de soluções de não-conformidades disponíveis na literatura. E esta comprovação se deu tanto durante as atividades de implementação destas bases quanto durante o processo de verificação das funcionalidades do protótipo.

Deve-se mencionar que a estrutura de descritores representa tem papel fundamental como guia no processo de transformação de uma descrição no nível do conhecimento, relatada em linguagem natural, em uma descrição no nível simbólico capaz de ser manipulada pelos agentes computacionais.

A segunda dimensão do conhecimento social diz respeito às redes de interações e comunicações, cujos detalhes podem ser observados na subseção anterior e demonstram que tanto as interações entre o usuário, agentes de interface e de recursos comprovaram ser viáveis e também confiáveis. Em especial, quando considerando uma perspectiva de distribuição espacial e ao longo da cadeia de processos.

Por sua vez, a escolha do formalismo de representação de casos por meio de um vetor de pares atributo-valor, descrita no Capítulo 7, revelou-se acertada e traduz-se nas seguintes vantagens: simplicidade na representação, simplificação da implementação computacional de medidas de similaridade eficientes, facilidade de armazenar em bases de dados com fácil recuperação. Tal representação demonstrou-se viável graças à estratégia de configuração da consulta adotada no tocante à possibilidade do usuário ajustar de forma personalizada de acordo com suas reais necessidades, os pesos dos atributos.

Por fim, os métodos de raciocínio baseado em casos implementados como comportamento dos agentes de recursos mostraram-se capazes de fornecer respostas corretas em relação aos casos de testes.

8.2.2 Validação conceitual do modelo sob o prisma dos agentes de recursos PFMEA

A avaliação da extensão na qual a base de conhecimento ontológica do agente de recurso PFMEA reflete os construtos elaborados pelos especialistas no domínio do Método de Análise de Falhas e Efeitos para processos de manufatura e montagem (PFMEA) diz respeito, em essência, à avaliação de uma ontologia. Assim, adota-se aqui a abordagem proposta por Gangemi et al. (2006), discutida no Capítulo 4 e denominada pelos autores como avaliação da dimensão funcional de uma ontologia, no que se refere ao seu propósito principal, isto é, especificar uma dada conceitualização ou premissas contextuais sobre o domínio ou área de interesse.

Neste sentido, Gangemi et al. (2006) revelam que tais especificações são sempre aproximações, posto que a relação entre a ontologia e a conceitualização (semântica cognitiva) depende sempre do agente racional que a concebeu e do espaço semântico que a codifica formalmente (semântica formal). Desta maneira, segundo os autores a avaliação funcional deve concentrar-se em analisar como estas dependências foram implementadas considerando a ontologia como uma linguagem que compreende o objeto informação e a conceitualização pretendida.

Neste contexto, Gangemi et al. (2006) sugerem avaliar a expressividade da ontologia de acordo com a sua adequação e ajuste referentes aos conceitos e relações presentes em um corpo de conhecimento de referência.

Em relação à expressividade, foi possível comprovar mediante os experimentos a adequação e o ajuste dos conceitos e relações modeladas na ontologia durante a operacionalização da base de conhecimento ontológica apresentada e discutida no Capítulo 7. Esta adequação foi comprovada em especial na subseção 7.2, durante a transformação da descrição no nível do conhecimento do método PFMEA estabelecido nos casos de teste de referência para a descrição simbólica representada pelas definições de classe e indivíduos na linguagem OWL-DL.

Deve-se destacar também que as bases de conhecimento foram desenvolvidas em consonância aos conceitos e termos estabelecidos pelas normas IEC 60812 (2006) e SAE J1739 (2002) e pela referência AIAG (2001), amplamente reconhecidas na área. Além disso, as bases foram formalizadas por meio da lógica de descrições denominada $ALCQHI_{R+}$, ou *SHIQ* (HAARSLEV e MÖLLER, 2001), a qual foi implementada através da linguagem padrão de ontologias denominada OWL DL (*Web Ontology Language – Description Logic*), desenvolvida pelo *World Consortium* (W3C, 2004).

Deve-se lembrar que o potencial da linguagem baseada em lógica de descrições, no âmbito da representação do conhecimento, se dá por meio dos sistemas de raciocínio que a implementam, neste trabalho de tese representado pelo sistema RacerPro Server (RACER SYSTEMS, 2006). Este sistema tem a capacidade de processar o conhecimento representado de modo explícito com o objetivo de inferir conhecimentos implícitos (HAARSLEV e MÖLLER, 2001).

Portanto, diante do exposto foi possível comprovar experimentalmente que as bases de conhecimento dos agentes de recursos PFMEA são adequadas e promissoras para o compartilhamento de conhecimento no domínio em questão.

Por outro lado, as redes de interações e comunicações implementadas na organização multiagente mostraram-se viáveis, em especial, quando considerando uma perspectiva de distribuição espacial e ao longo da cadeia de processos.

Por fim, foi possível avaliar experimentalmente a fidedignidade das respostas fornecidas pelos agentes de recursos PFMEA ativos na organização multiagentes, esta fidedignidade pode ser comprovada comparando as respostas produzidas pelos métodos de recuperação implementados comportamento dos agentes de recursos PFMEA (apresentados no capítulo 6) com as respostas possíveis de serem extraídas das bases de conhecimento alimentadas com um corpo de conhecimento de referência (casos de testes) para uma dada consulta.

8.3 VALIDAÇÃO OPERACIONAL DO MODELO

De acordo com os procedimentos metodológicos apresentados no Capítulo 3, a validação operacional consiste em avaliar a relevância do comportamento colaborativo da organização multiagente com respeito ao seu propósito, o qual foi estabelecido no objetivo geral do trabalho de tese como apoiar a solução de problemas de não-conformidades em processos de manufatura, a partir do compartilhamento e recuperação de conhecimentos decorrentes da solução de não-conformidades prévias similares e, da aplicação do método preventivo de análise de modos de falha e efeitos considerando a distribuição especial e organizacional das fontes de conhecimento.

Dentro desta lógica, esta subseção discute a validação operacional do modelo proposto a partir de experimentos baseados em estudos de caso de solução de problemas de não-conformidades em componentes moldados por injeção de termoplásticos. Estes experimentos envolvem casos de testes relatados pelo especialista em análise e solução de problemas em processos de moldagem por injeção (*troubleshooter*) da empresa pesquisada, os quais foram coletados durante o trabalho de campo, bem como considerando ainda os casos de testes

obtidos da literatura e operacionalizados nas respectivas bases de conhecimento como descrito no Capítulo 7.

É importante destacar, que a pesquisa de campo foi realizada em uma empresa produtora de componentes técnicos pelo processo de moldagem por injeção de termoplásticos para aplicações eletro-eletrônicas situada no Estado de Santa Catarina, a qual produz componentes moldados pelo processo de injeção em três turnos de trabalho, bem como aloca parte da produção de componentes em empresas transformadoras parceiras localizadas em outras regiões do Estado caracterizando, neste aspecto, a distribuição física de recursos de manufatura.

Neste sentido, as figuras 8.24 e 8.25 mostram um experimento que envolve a configuração da consulta junto ao agente de interface RBC, cujo propósito geral é recuperar as "lições aprendidas" com a análise e a solução de problemas de empenamento de peças moldadas pelo processo de injeção de termoplásticos que estão armazenadas nas bases de conhecimento dos agentes de recursos de conhecimento RBC ativos na organização multiagente.

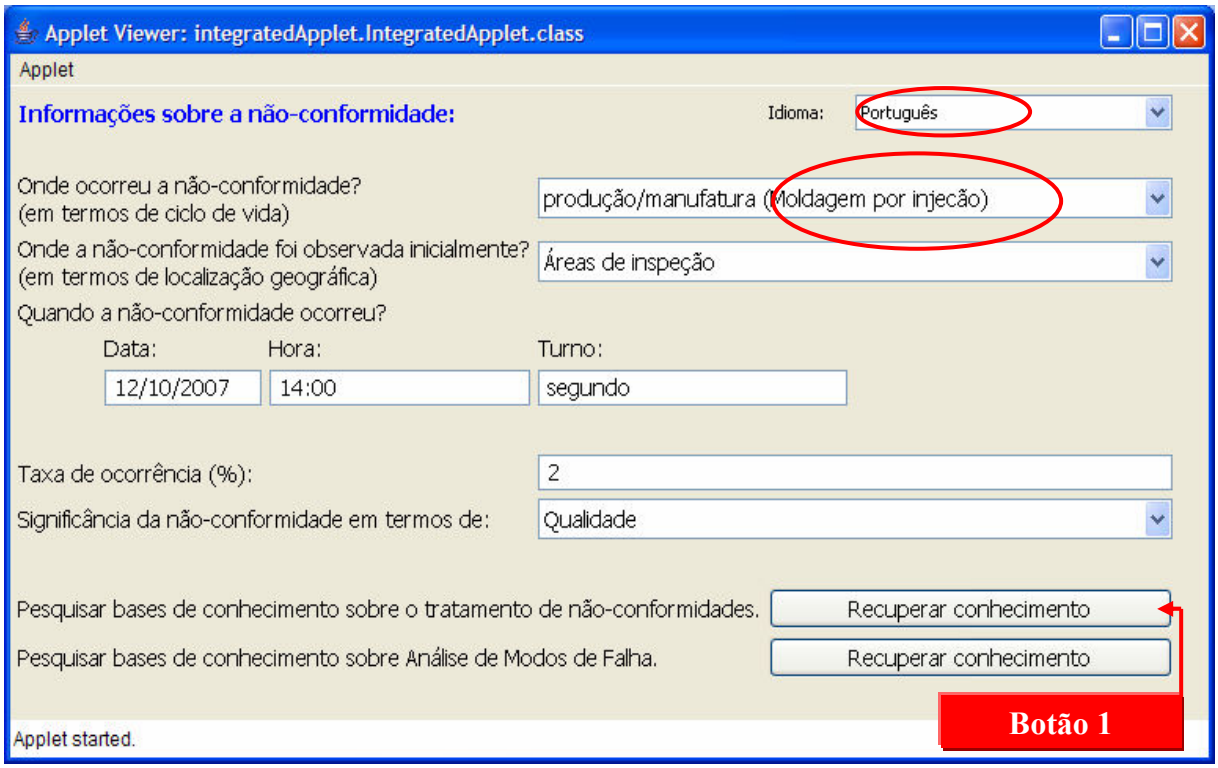


Figura 8.24 – Janela de interface gráfica do protótipo apresentada ao usuário.

A figura 8.24 mostra a primeira interface gráfica apresentada ao usuário destinada a capturar as informações primárias relacionadas à ocorrência da não-conformidade, bem como identificar a perspectiva de apoio à solução de problemas de não-conformidades requerida

pelo usuário, o que permite identificar quais instâncias dos agentes de interface devem ser apresentadas e quais instâncias dos agentes de recursos de conhecimento devem ser contatadas.

Neste experimento a consulta é direcionada às bases de conhecimento relativas às "lições aprendidas" com o tratamento de não-conformidades similares (Botão 1) e, por consequência, a instância do agente de interface apresentada ao usuário foi a do Agente de Interface RBC relativa ao processo de moldagem por injeção preparada para consultas na língua portuguesa.

Por sua vez, a figura 8.25 mostra a janela gráfica deste Agente de Interface RBC, onde a configuração da consulta envolveu os atributos correspondentes à descrição, classificação e condições de contorno da não-conformidade em questão. Nesta aba, a não-conformidade foi descrita em termos do modo de falha "empenamento geral" e sua localização foi atribuída a uma *feature* prismática primária com parede paralela à linha de partição do molde.

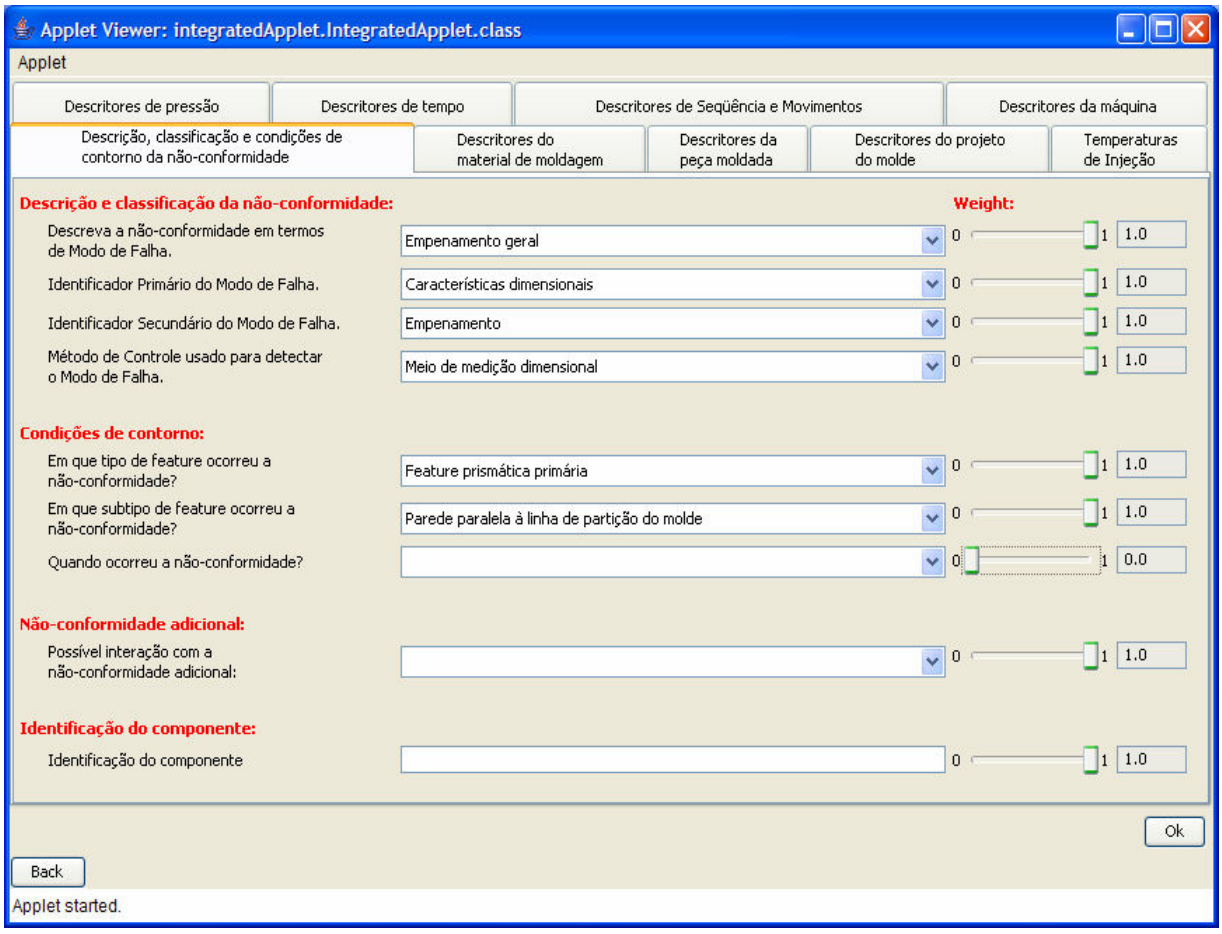


Figura 8.25 – Janela gráfica do Agente de Interface RBC – Aba: descrições.

Adicionalmente, a aba mostra que esta consulta em particular incluiu os indicadores primário e secundário do modo de falha e o método de controle usado para detectá-lo,

enquanto os demais descritores foram deixados em aberto pelo usuário. É importante observar, que os descritores não preenchidos serão considerados no cálculo da medida da similaridade global com os casos armazenados, a menos que o valor do peso associado a estes descritores seja ajustado em zero.

A figura 8.26 ilustra o aspecto visual do modo de falha “empenamento geral”, que de acordo com Bryce (2003) pode ser definido como uma distorção da peça moldada após a sua ejeção do molde no final do processo de moldagem por injeção.

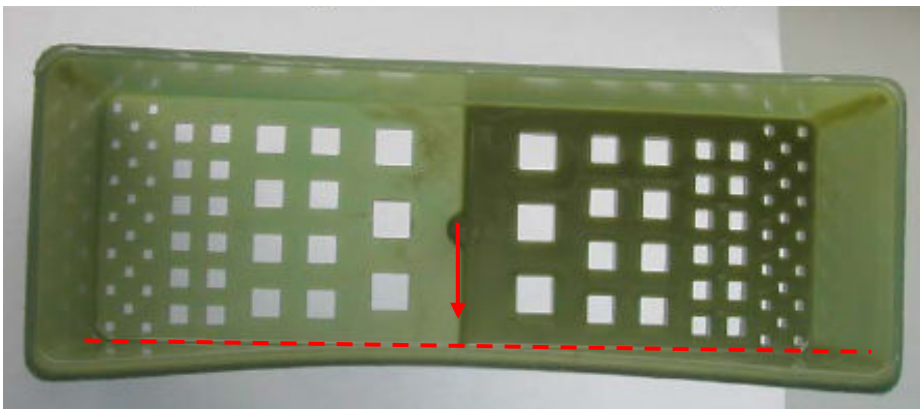


Figura 8.26 – “Empenamento” de um componente (Fonte: SANCHO, 2005).

A figura 8.27, por outro lado, mostra a configuração da consulta em relação a aba correspondente aos descritores do material usado no processo de moldagem que gerou a não-conformidade em questão, neste caso o material indicado é um termoplástico do grupo polimérico poliestireno de alto impacto (*High Impact PS,* de acordo com o símbolo ASTM). Esta aba inclui também outros descritores, tais como indicações do índice de fluidez e densidade, os quais foram projetados para permitir o refinamento do cálculo da medida de similaridade entre materiais termoplásticos, usando para isso a noção de materiais considerados como contratipos.

Neste experimento, a configuração dos descritores do material (figura 8.27) não indicou a presença de aditivos, cargas de reforço ou a mistura com outros polímeros, indicações de secagem ou uso de material regranulado no processo. A figura 8.27 mostra ainda que os pesos associados a estes descritores foram ajustados para o valor zero, isto indica que estes descritores não serão levados em conta durante o cálculo da similaridade entre os casos armazenados nas bases de conhecimento dos agentes de recursos RBC.

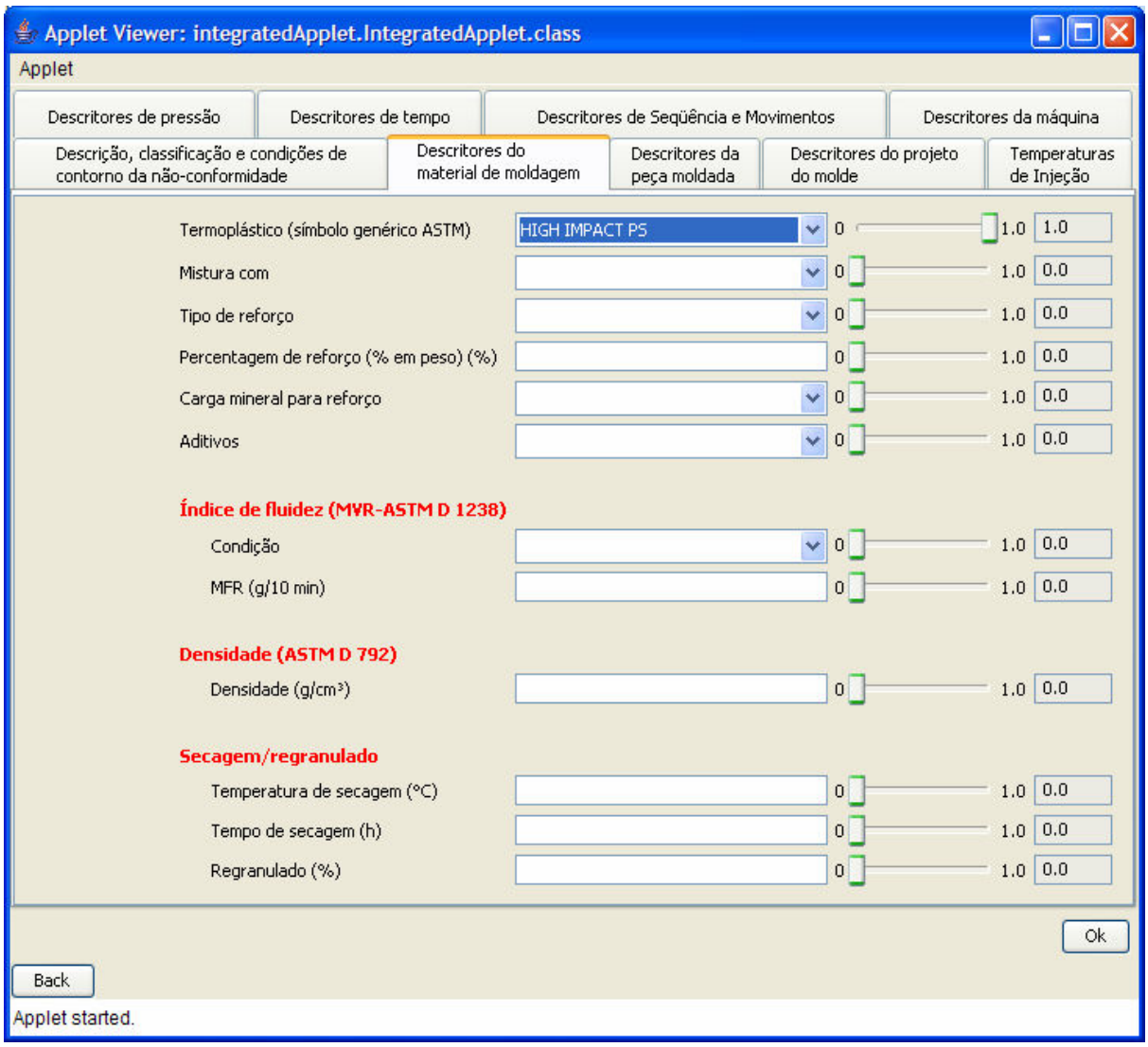


Figura 8.27 – Janela gráfica do Agente de Interface RBC – Aba: descritores do material de moldagem.

Por sua vez, a figura 8.28 mostra a configuração dos descritores correspondentes ao projeto básico da peça moldada. Destacando-se que a obtenção da forma e dos detalhes de projeto da peça moldada depende do modo como o processo de moldagem opera, enquanto o tamanho da peça é limitado pelas pressões e disponibilidades de equipamento (TRANTINA, 2003).

Neste sentido, foram configuradas a área projetada, espessura de parede mínima e a indicação da complexidade da peça moldada. Uma complexidade moderada para a forma da peça moldada foi representada pelos fatores de complexidade básica (que podem variar de 1,0 a 9,96) e de complexidade subsidiária (que podem variar de 1,0 a 2,15) como propostas por Dixon e Poli (1999). Nesta aba os pesos foram ajustados de modo a ressaltar a importância dos fatores de complexidade da peça moldada, contudo os demais descritores também serão

levados em consideração no cálculo da medida de similaridade (peso 0,6) devido a sua influência sobre a ocorrência do modo de falha em questão.

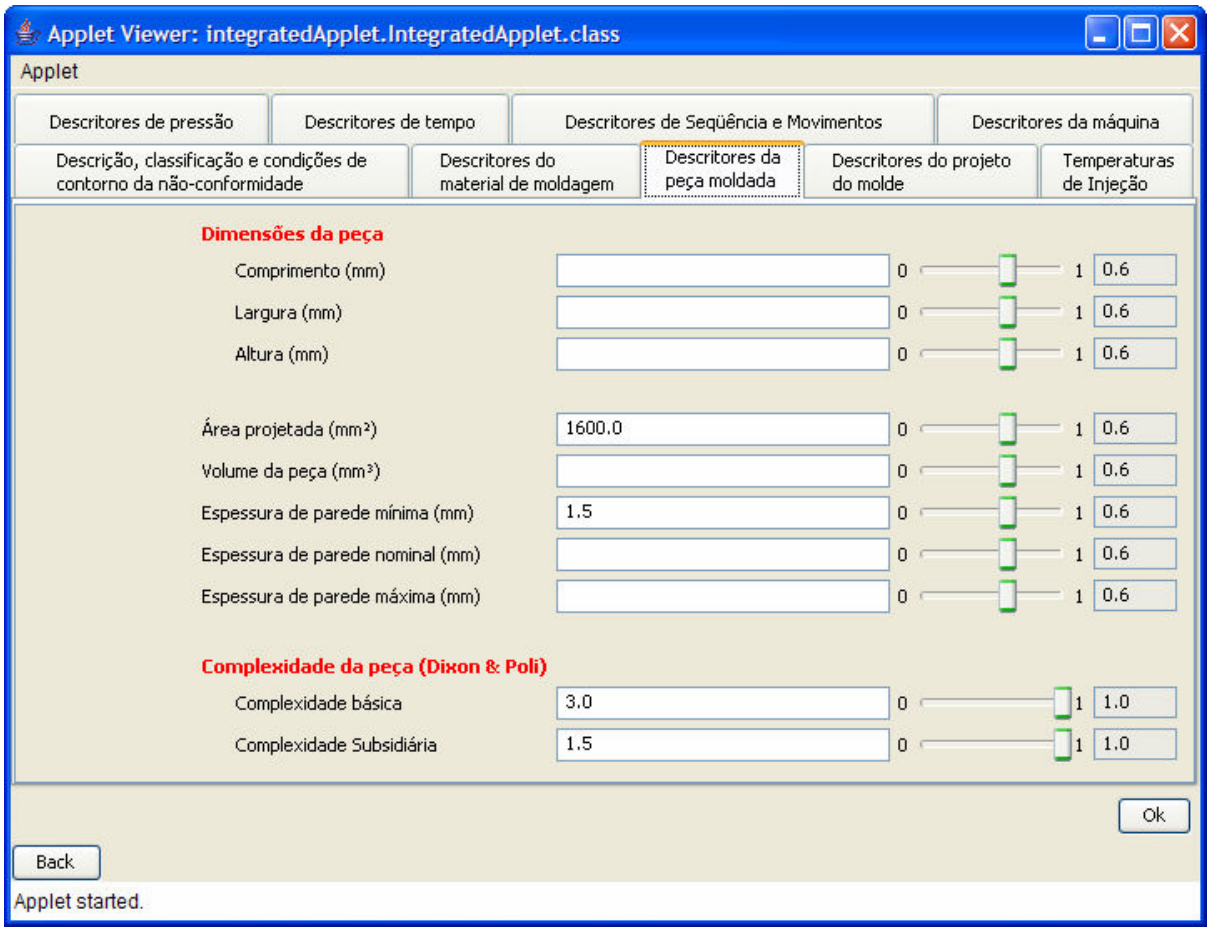


Figura 8.28 – Janela gráfica do Agente de Interface RBC – Aba: descritores da peça.

A figura 8.29 mostra a configuração dos descritores correspondentes ao projeto básico do molde usado no processo de moldagem por injeção e, que neste caso, indicam somente que o molde é dotado de um sistema de câmara quente (ver ilustração na figura 8.30), revelando que os canais alimentação e distribuição podem ser mantidos sempre na temperatura ideal de fluxo de injeção amplamente usados nos processos de moldagem atualmente. Embora, a configuração dos descritores do molde não faça a indicação do arranjo e das dimensões dos canais de alimentação/distribuição e dos pontos de injeção.

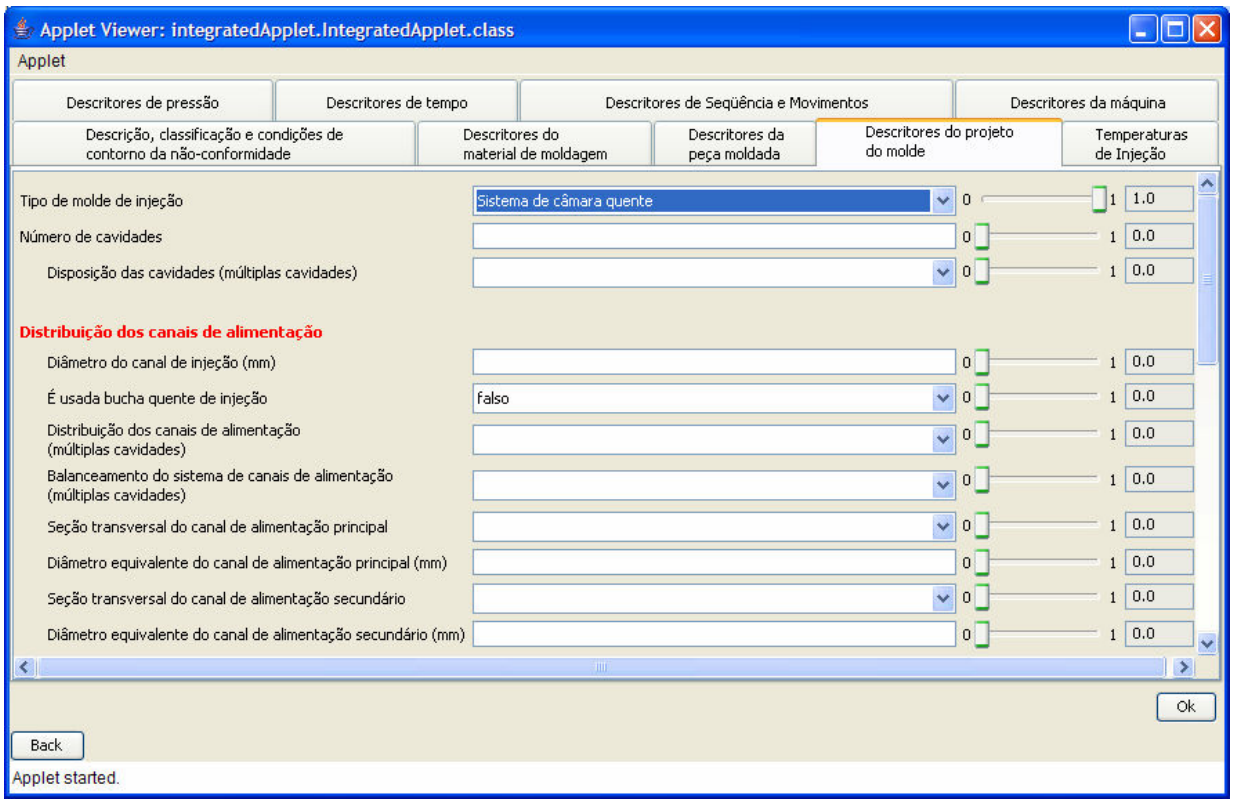


Figura 8.29 – Janela gráfica do Agente de Interface RBC – Aba: descritores do molde de injeção.

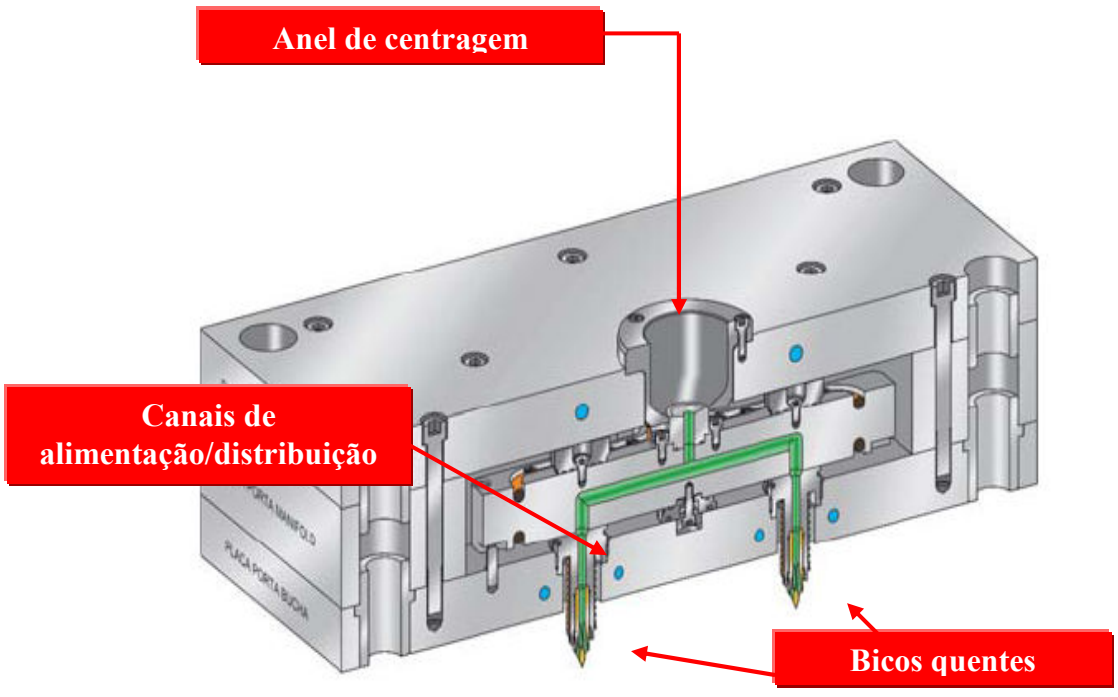


Figura 8.30 – Exemplo de sistema de câmara quente (Fonte: POLIMOLD, 2007).

A figura 8.31 mostra as temperaturas empregadas no processo de moldagem e, no caso, indica somente a temperatura do fundido e a temperatura da superfície do molde.

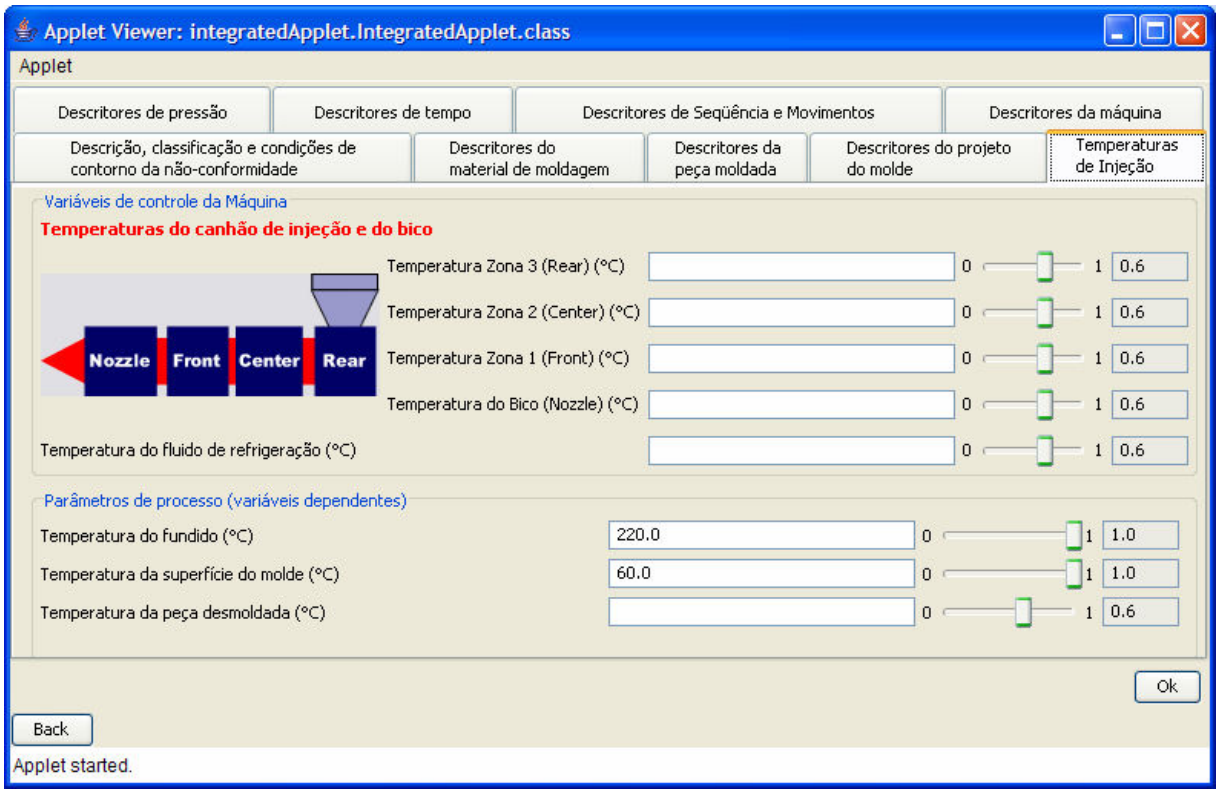


Figura 8.31 – Janela gráfica do Agente de Interface RBC – Aba: temperatura de injeção.

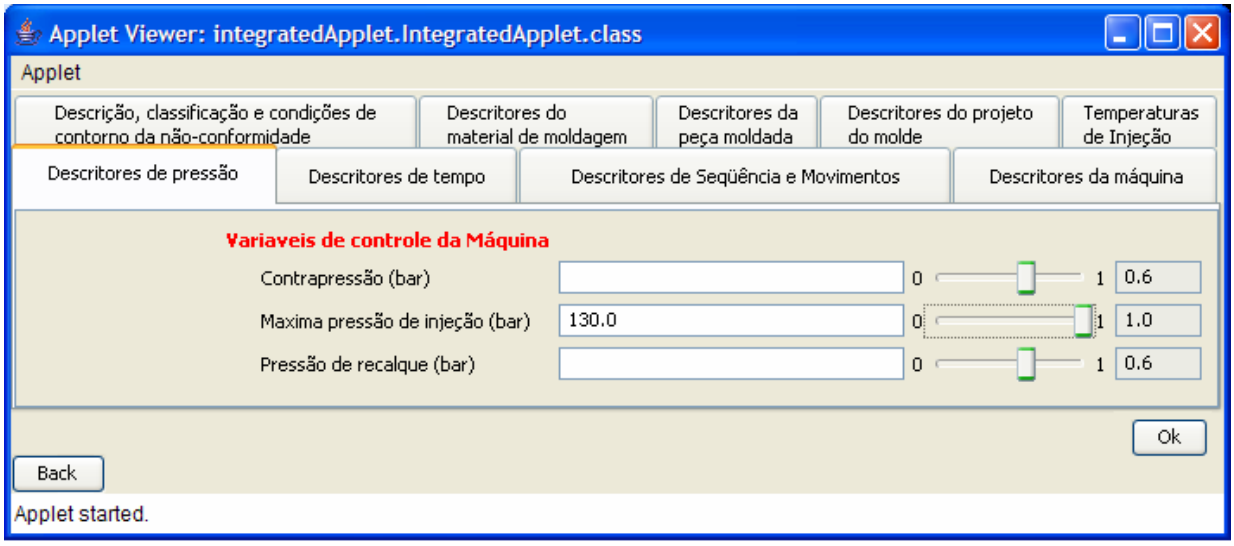


Figura 8.32 – Janela gráfica do Agente de Interface RBC – Aba: descritores de pressão.

Na figura 8.32 é apresentada a configuração da consulta com relação aos descritores de pressão empregados no processo representada aqui somente a pressão máxima de injeção usada.

Por fim, a figura 8.33 mostra a configuração da consulta em relação aos descritores de tempo, em particular, o tempo de refrigeração e o tempo total de ciclo, observando-se que não foram incluídos os descritores de seqüência e movimentos nesta consulta em particular.

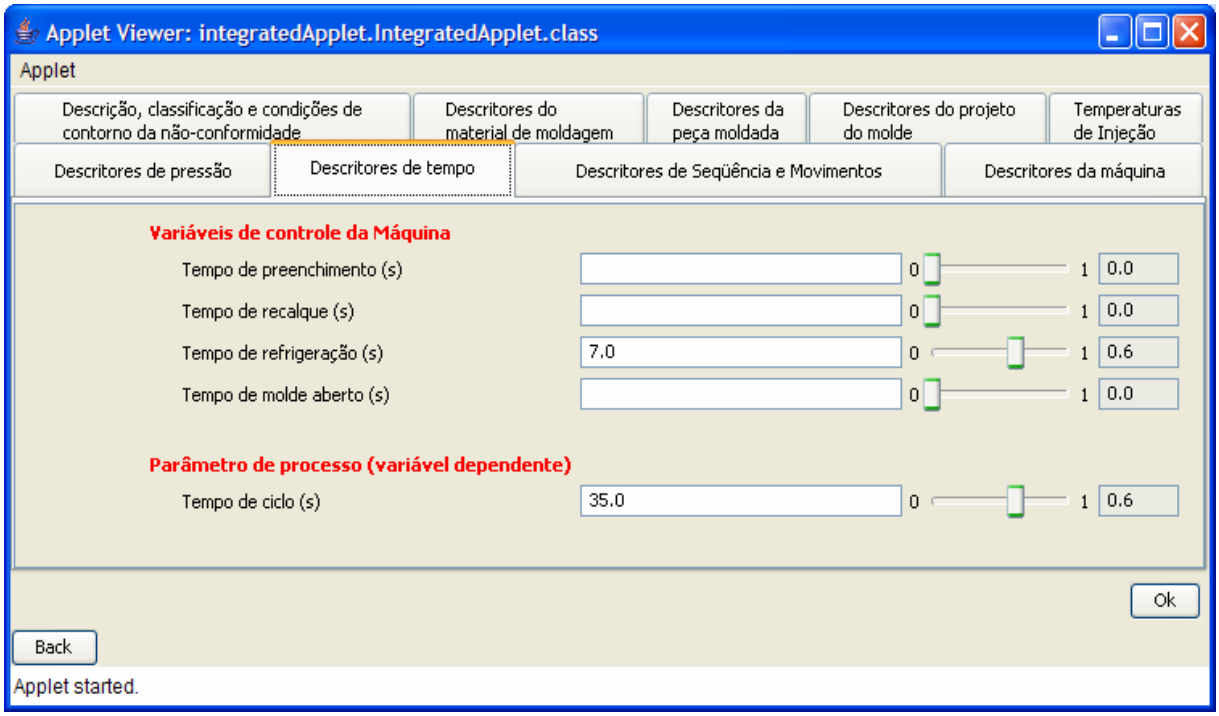


Figura 8.33 – Janela gráfica do Agente de Interface RBC – Aba: descritores de tempo.

Em síntese, as figuras de 8.24 a 8.33 correspondem à configuração de uma consulta por meio do serviço do agente de interface RBC, a qual compreende os descritores essenciais de um caso de não-conformidade a ser tratado, bem como os pesos associados a cada um dos descritores representando a importância de cada descritor de acordo com as percepções do usuário. É importante observar que novos descritores podem ser incluídos, bem como reajustes nos pesos podem ser feitos de acordo com a estratégia de consulta ou outras necessidades do usuário.

A figura 8.34 apresenta os casos mais similares recuperados pelos métodos de raciocínio baseado em casos encapsulados no comportamento dos agentes de recursos de conhecimento RBC ativos na sociedade e os respectivos valores dos graus de similaridade em relação aos descritores iniciais configurados junto ao agente de interface RBC.

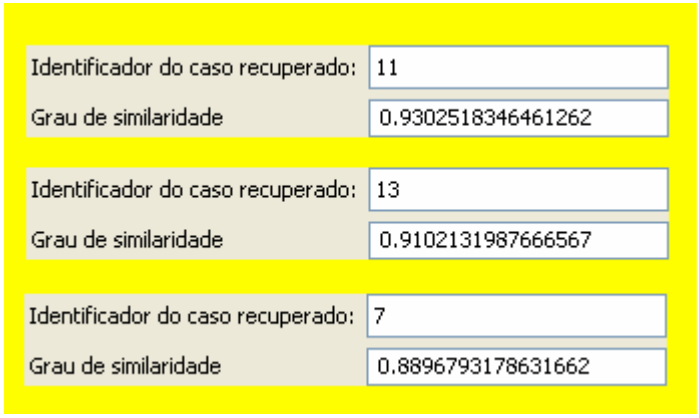


Figura 8.34 – Casos mais similares recuperados pelos agentes de recursos RBC.

Por sua vez, as figuras de 8.35 a 8.37 apresentam os blocos de conhecimento correspondentes à solução sugerida armazenada junto ao caso mais similar (caso 11) recuperado pelos agentes de recursos RBC.

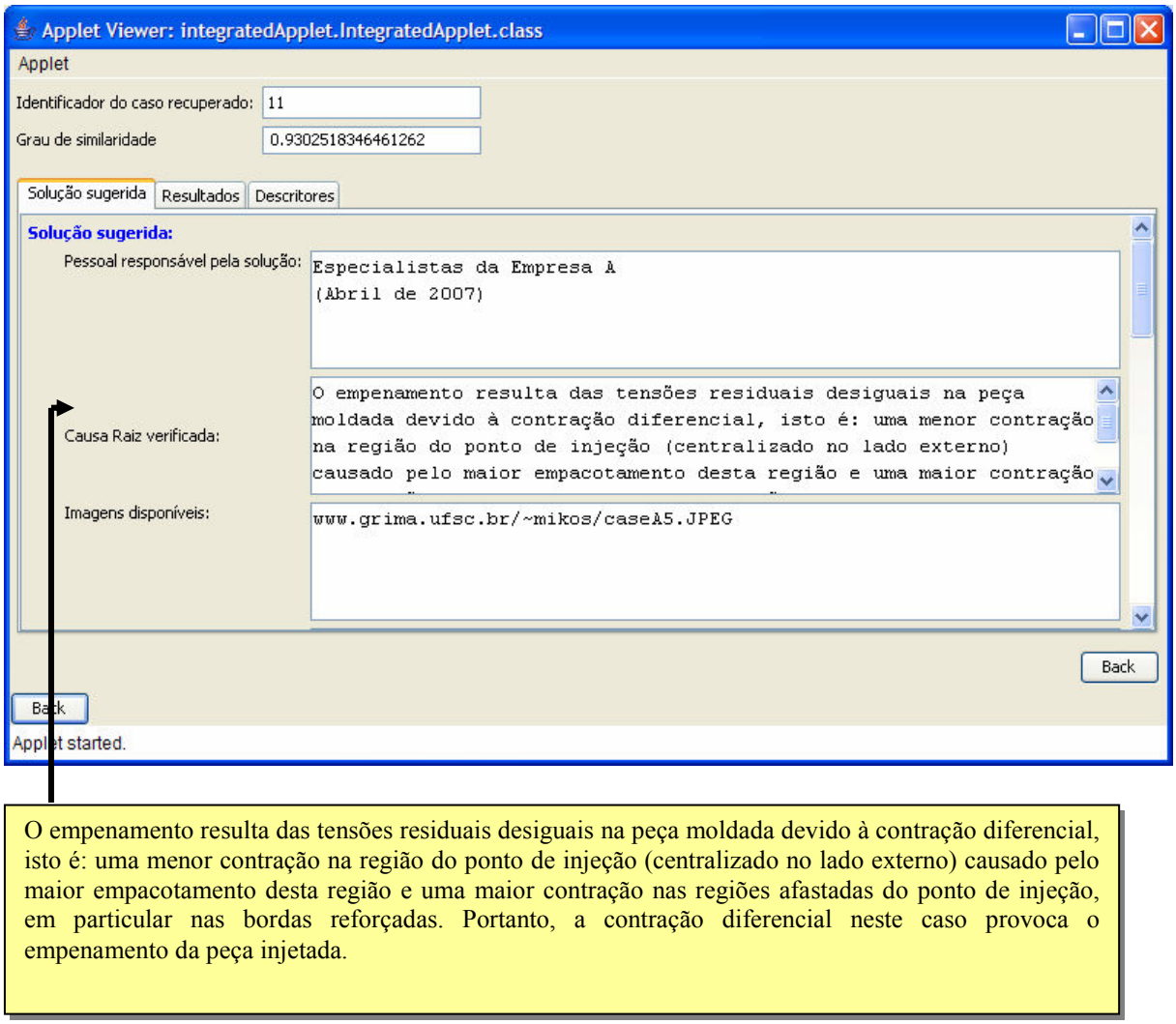
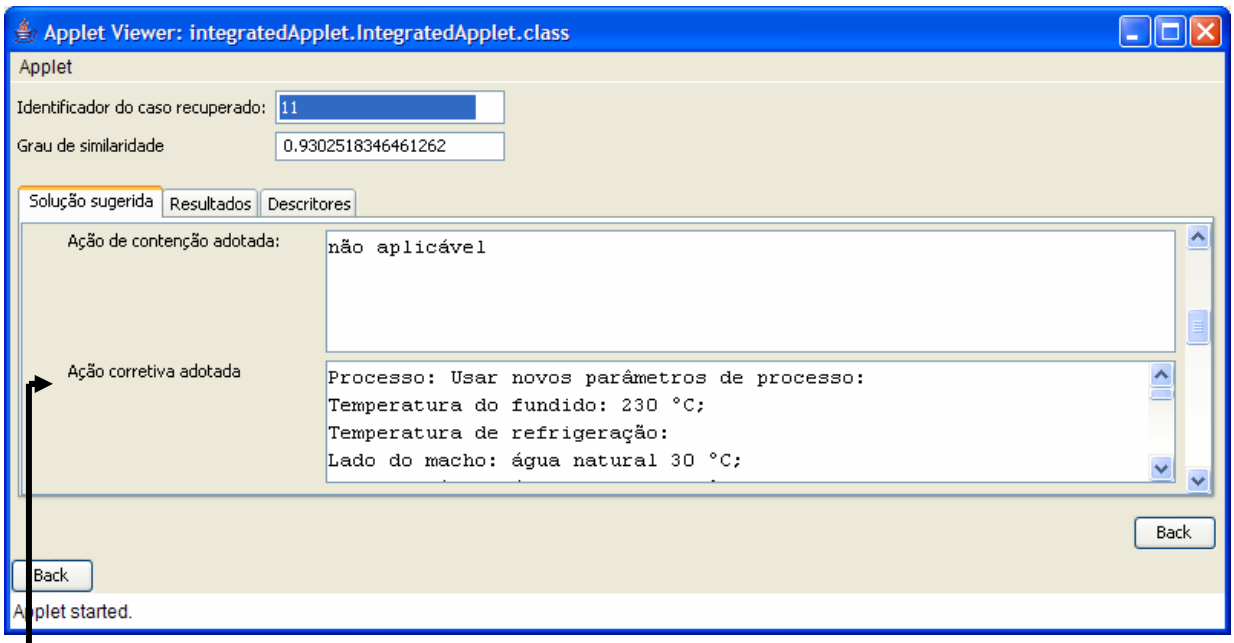


Figura 8.35 – Resultado da recuperação de casos similares pelos agentes de recursos RBC – Solução sugerida – Parte I.



Processo: Usar novos parâmetros de processo:
Temperatura do fundido: 230 °C;

Temperatura de refrigeração:
Lado do macho: água natural 30 °C;
Lado da fêmea: água natural 30 °C.
Curso de dosagem: 48mm; descompressão: 2,0mm.

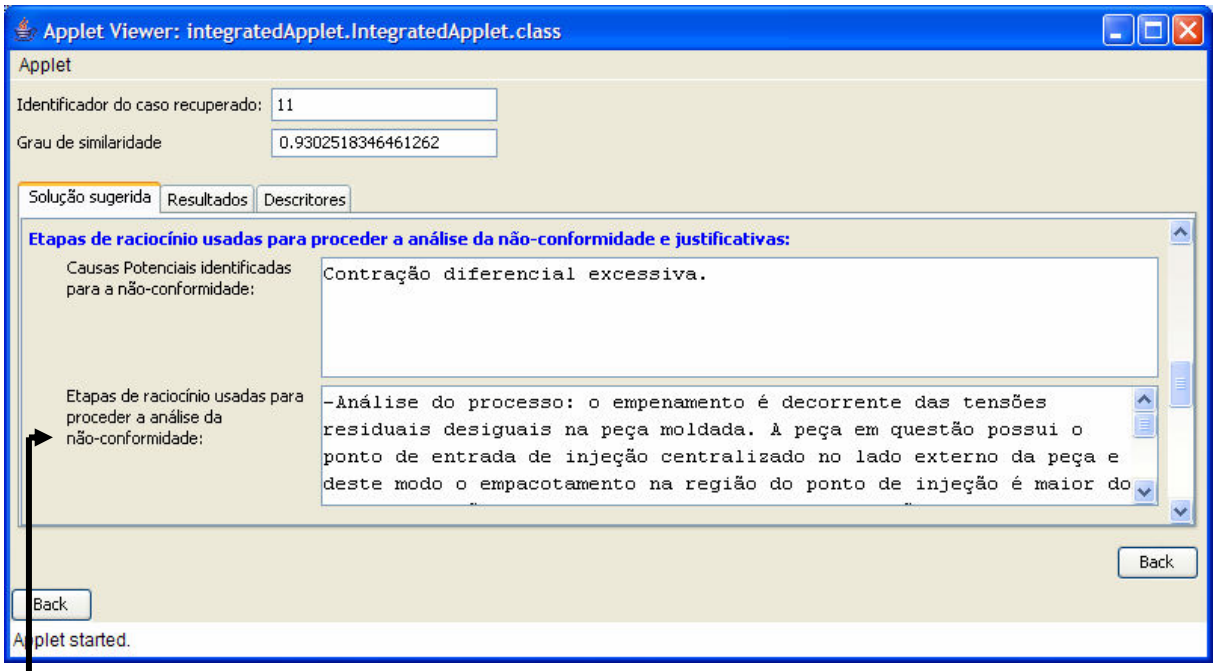
- Usar um perfil de velocidade de injeção (posição /velocidade %velocidade máxima):
pos.: 50mm – v: 25%; pos.: 47mm – v: 25%;
pos.: 42mm – v: 35%; pos.: 36mm – v: 35%;
pos.: 31mm – v: 55%; pos.: 26mm – v: 55%;
pos.: 21mm – v: 40%; pos.: 16mm – v: 30%;
pos.: 10mm – v: 25%; pos.: 5,2mm – v: 25%.
Pressão injeção: 75 bar.
- Usar ponto de comutação em 28 mm.
- Ressaltando o uso de um perfil da pressão de recalque (tempo/pressão) de:
t: 0,2s - p: 22bar; t: 0,4s – p: 22bar; t: 0,6s – p: 22bar;
t: 0,8s - p: 22bar; t: 1,0s – p: 22bar; t: 1,2s – p: 22bar;
t: 1,4s - p: 22bar; t: 1,6s – p: 22bar; t: 1,8s – p: 22bar;
t: 2,0s – p: 22bar;

Molde: foi modificado o projeto do botão “page“ aumentando a parede do botão de 0,8mm para 1,5 para facilitar o preenchimento (convite para injeção) de modo a evitar o preenchimento incompleto da peça.

Figura 8.36 – Resultado da recuperação de casos similares pelos agentes de recursos RBC – Solução sugerida – Parte II.

Na figura 8.35 destaca-se a causa da não-conformidade identificada pelo especialista (*Troubleshooter*) responsável pelo processo de análise e solução do problema relatado no caso recuperado, o qual atribuiu o modo de falha em questão a diferenças nos níveis de

empacotamento ao longo da peça. E na figura 8.36, por sua vez, destaca-se a ação corretiva adotada pelo especialista no caso, a qual consiste de um novo conjunto de parâmetros de injeção para o processo. É importante observar que uma modificação do molde também foi incluída, pois este modo de falha podia ocorrer em conjunto com o preenchimento incompleto da peça na região de um botão de acionamento (na forma de uma lâmina articulada) localizado próximo à borda da peça.



- Análise do processo: o empenamento é decorrente das tensões residuais desiguais na peça moldada. A peça em questão possui o ponto de entrada de injeção centralizado no lado externo da peça e deste modo o empacotamento na região do ponto de injeção é maior do que nas regiões mais distantes do ponto de injeção em especial nas bordas o que implica em menor empacotamento, gerando uma contração ou encolhimento diferencial.

Deste modo, a peça deve ser moldada com um perfil de pressão de recalque buscando a menor compactação possível da peça mantendo a massa da injetada em 85,6 gramas.
Pois as áreas diferentes da peça podem apresentar diferenças no nível de tensão interna no volume específico.

Figura 8.37 – Janela gráfica do Agente de interface RBC – Solução sugerida –Parte III.

A figura 8.37 revela a linha de raciocínio empregada pelo especialista na abordagem do problema, que neste caso concentrou-se na fase de recalque do processo de moldagem, ou seja, buscando o menor empacotamento possível para a peça, mas dentro de um limite específico para a massa, cujo objetivo é evitar a ocorrência de rechupes.

Por fim, a figura 8.38 mostra os resultados relatados pelo especialista após a implementação da solução sugerida, em especial, a sugestão de uso de um gráfico de controle para a massa da peça visando melhorar a detecção das causas relacionadas ao modo de falha.

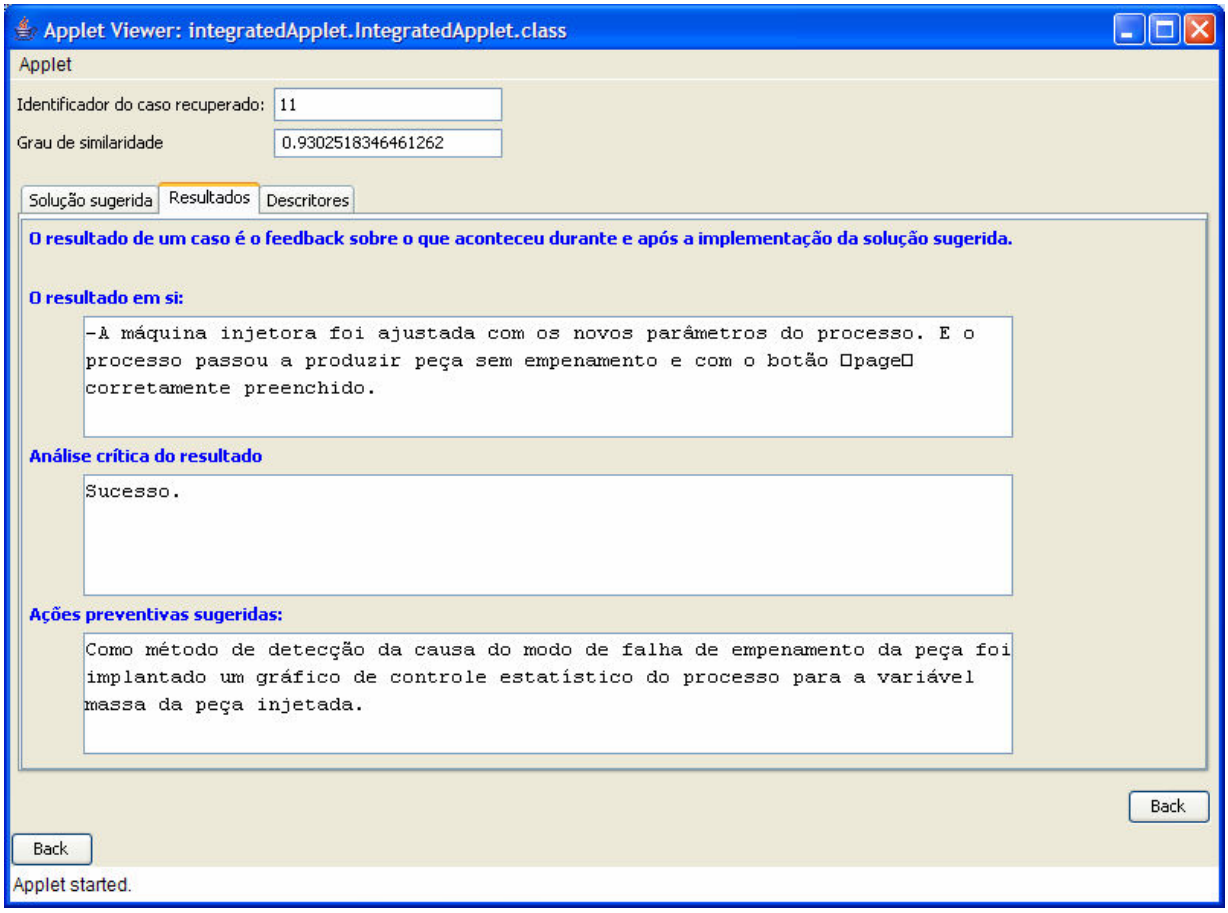
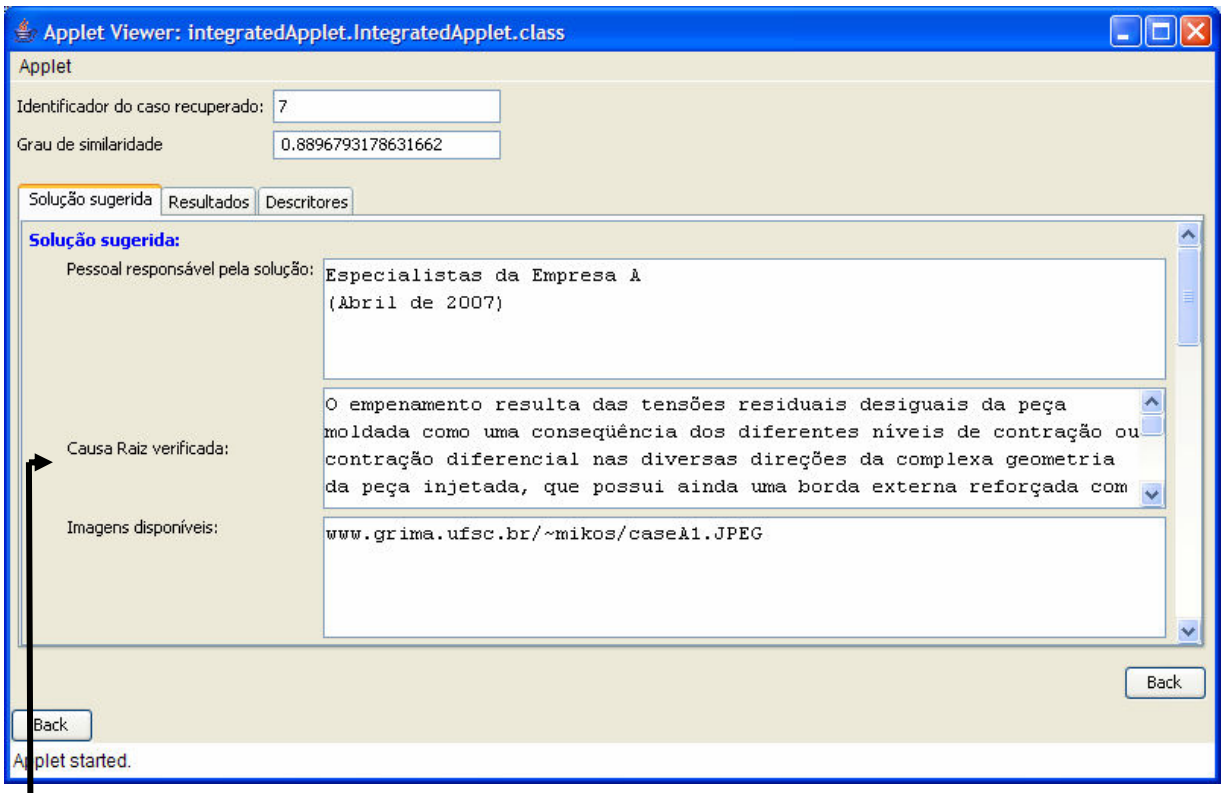


Figura 8.38 – Janela gráfica do Agente de interface RBC – Resultados.

Para manter a clareza do texto, as janelas gráficas correspondentes à solução sugerida e aos resultados referentes ao segundo caso mais similar (caso 13) são apresentadas no apêndice 3.

Por outro lado, as figuras de 8.39 a 8.43 apresentam as janelas gráficas correspondentes à solução sugerida e aos resultados ao terceiro casos mais similar (caso 7), pois embora apresente uma similaridade menor em função de não usar o sistema de câmara quente, este caso incluiu uma peculiaridade interessante, isto é o modo de falha "empenamento geral" está associado a outro modo de falha "ruptura da peça", o que tornou a sua análise e solução mais complexa.

A figura 8.39 destaca as causas dos dois modos de falha, pois o especialista empreendeu um processo de análise em conjunto para estes dois modos de falha.

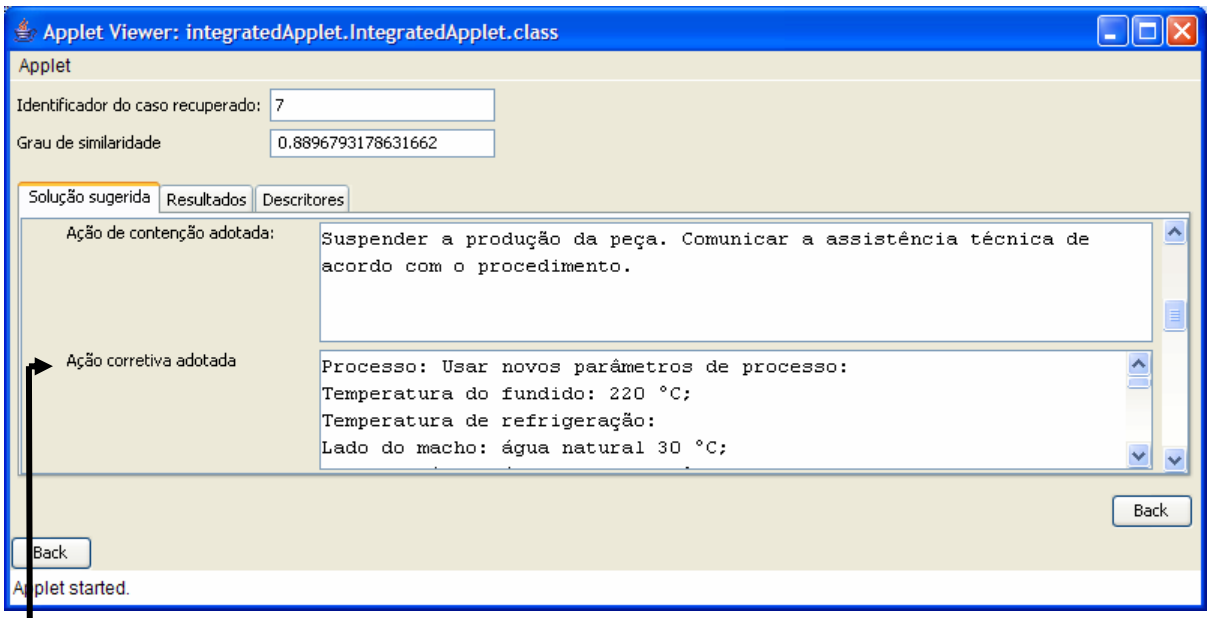


O empenamento resulta das tensões residuais desiguais da peça moldada como uma conseqüência dos diferentes níveis de contração ou contração diferencial nas diversas direções da complexa geometria da peça injetada, que possui ainda uma borda externa reforçada com 3,0 mm de espessura e paredes com espessura nominal de 1,8 mm.

Por sua vez, as rupturas nos castelos de fixação do contato metálico (Protusão: Ponto de junta mecânica) ocorreram durante o uso quando as peças eram submetidas ao carregamento. Estas rupturas ocorreram em particular nas linhas de emendas frias e com baixa resistência mecânica existentes no castelo e que foram geradas durante o processo de injeção em função da dificuldade de preenchimento do castelo e devido às baixas temperaturas na frente de fluxo de material nesta região.

Figura 8.39 – Janela gráfica do Agente de interface RBC – Solução sugerida –Parte I.

A figura 8.40 mostra que uma ação de contenção foi necessária, pois a ruptura da peça poderia ocorrer também durante o uso do produto, em especial, com a troca de baterias e, mostra ainda que a solução empregada pelo especialista da empresa envolveu diversas ações (alterações nos parâmetros de processo, necessidade de recursos de máquina e modificação do molde) em função da complexidade e extensão das causas dos modos de falha.



Processo: Usar novos parâmetros de processo:
Temperatura do fundido: 220 °C;

Temperatura de refrigeração:
Lado do macho: água natural 30 °C;
Lado da fêmea: água quente 60 °C.

Curso de dosagem: 200mm; descompressão: 4mm.
- Usar um perfil de velocidade de injeção (posição /velocidade):
pos.: 200mm – v: 120mm/s; pos.: 195mm – v: 110mm/s; pos.: 180mm – v: 110mm/s;
pos.: 170mm – v: 110mm/s; pos.: 160mm – v: 110mm/s; pos.: 150mm – v: 110mm/s;
pos.: 140mm – v: 110mm/s; pos.: 135mm – v: 110mm/s; pos.: 130mm – v: 110mm/s;
pos.: 124mm – v: 30mm/s.

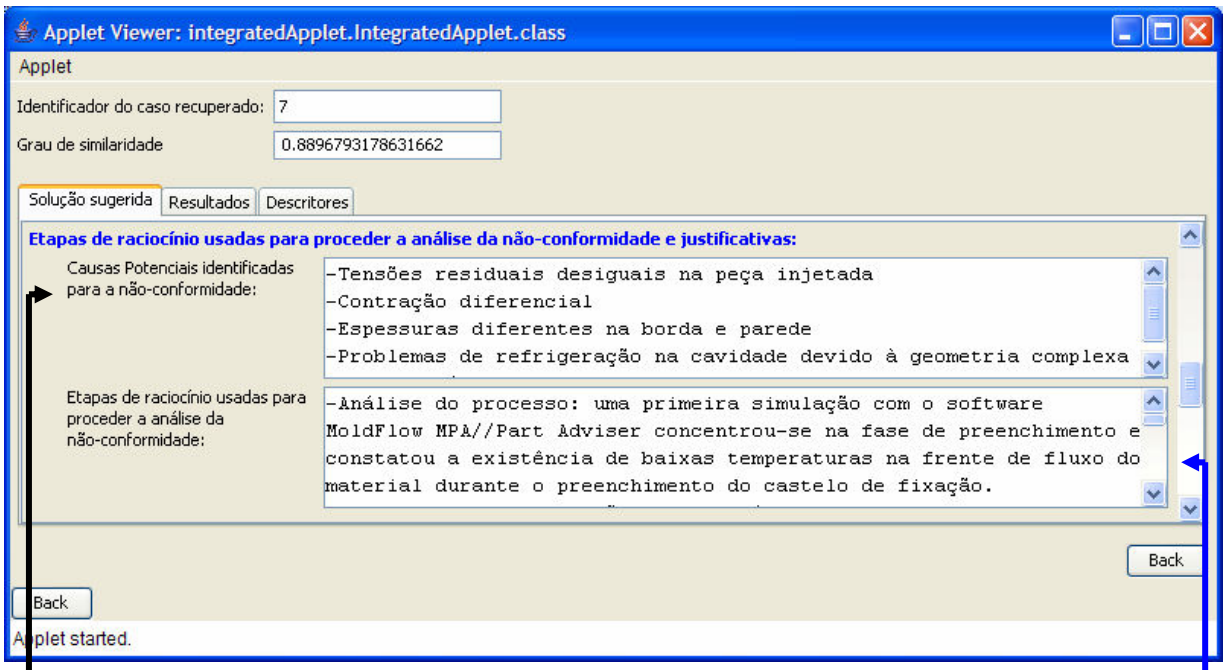
- Usar ponto de comutação em 125 mm.
- Usar um perfil da pressão de recalque (tempo/pressão) de:
t: 4,0s - p: 90bar; t: 2,0s – p: 70bar; t: 1,0s – p: 60bar; t: 1,0s – p: 60bar.; t: 4,0 s – p: 90bar.

Obs.: Para um melhor controle do ciclo de injeção da peça, especialmente na região do castelo de fixação do contato metálico foi usada uma máquina injetora com um sistema de "*close loop*" e com força de fechamento de 300 toneladas.

Molde: foi modificado o projeto do castelo de fixação do contato metálico incluindo nervuras prismáticas de reforço e aumentos nas paredes (convites de injeção), bem como raios de arredondamento em todas as arrestas para evitar a concentração de tensões no castelo.

Figura 8.40 – Janela gráfica do Agente de interface RBC – Solução sugerida –Parte II.

A figura 8.41 destaca as causas potenciais identificadas para o empenamento, em especial problemas de refrigeração na cavidade do molde, bem como a linha de raciocínio empregada pelo especialista durante a análise e solução do problema.

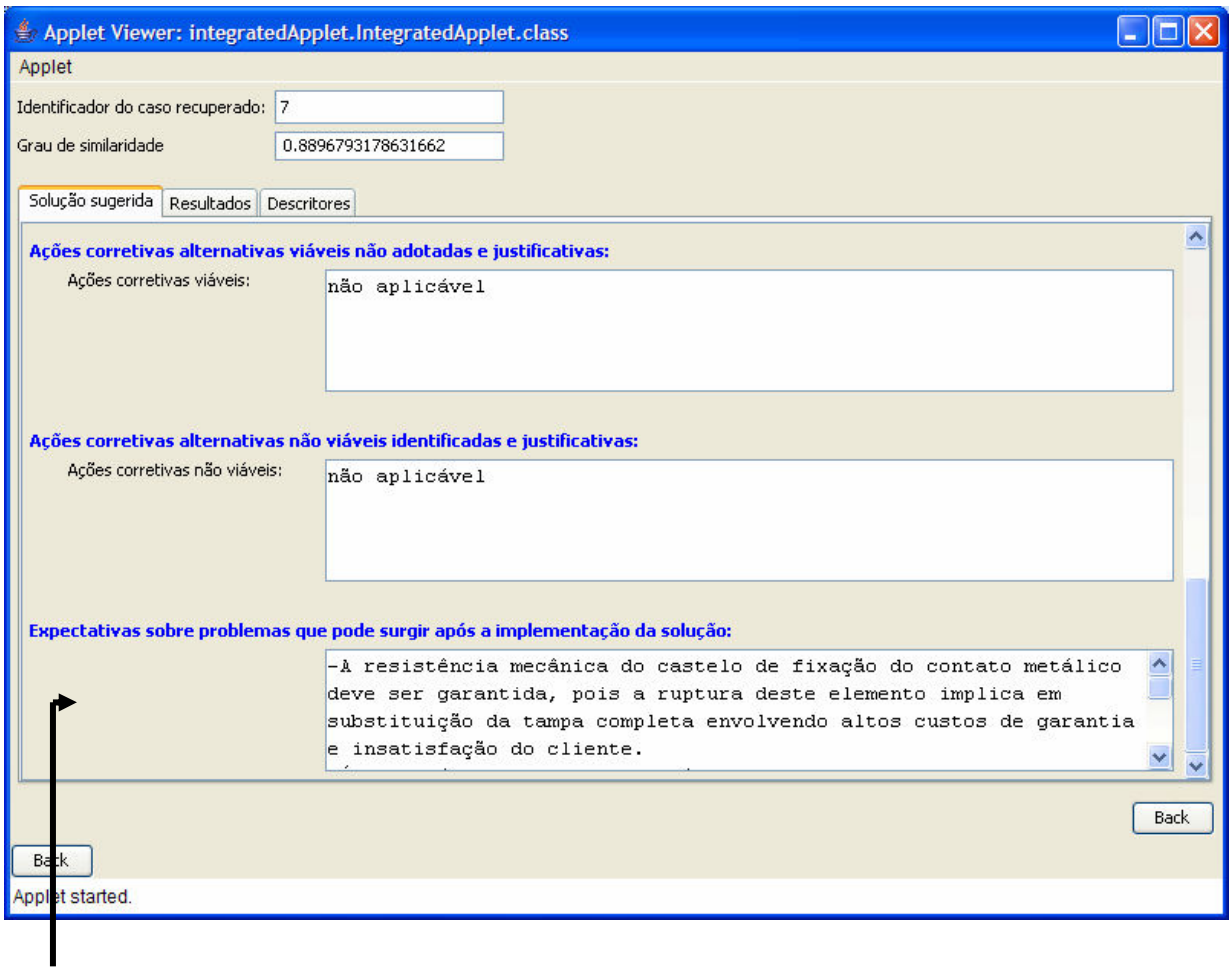


-Tensões residuais desiguais na peça injetada
-Contração diferencial
-Espessuras diferentes na borda e parede
-Problemas de refrigeração na cavidade devido à geometria complexa com reentrâncias

-Análise do processo: uma primeira simulação com o software *MoldFlow MPA//Part Adviser* concentrou-se na fase de preenchimento e constatou a existência de baixas temperaturas na frente de fluxo do material durante o preenchimento do castelo de fixação.
-A partir desta simulação novos parâmetros para o processo foram determinados (conforme definidos na ação corretiva adotada).
-Ressaltando-se o novo perfil de velocidades de injeção (posição /velocidade) e a pressão máxima, os quais exigiram o uso de uma máquina injetora com um sistema de "*close loop*" que proporciona o controle mais eficiente da velocidade de injeção e um menor tempo para atingir a velocidade de 120mm/s, além de uma força de fechamento de 300 toneladas.
-Uma segunda simulação com o software *MoldFlow MPA/Performance Adviser* foi necessária para otimizar os parâmetros de recalque, pois o castelo de fixação do contato metálico ainda apresentava ruptura quando submetido ao carregamento causada agora por tensões internas geradas na fase de recalque em função excessivo empacotamento.
-Adicionalmente, novas temperaturas para o lado interno (lado do macho) e lado externo da peça (lado da fêmea) foram determinadas experimentalmente (conforme definidos na ação corretiva adotada) de modo a compensar a contração diferencial responsável pelo empenamento ainda observado na peça injetada.
-Análise do projeto do molde: o projeto do castelo de fixação foi modificado a partir das simulações: incluindo nervuras prismáticas de reforço e aumentos nas paredes (convites de injeção) visando facilitar o preenchimento, bem como arredondamentos em todas as arestas do castelo para eliminar os pontos de concentração de tensões.

Figura 8.41 – Janela gráfica do Agente de interface RBC – Solução sugerida –Parte III.

A figura 8.42 mostra as expectativas sobre problemas que podem surgir após a implementação destacando os possíveis efeitos associados aos modos de falha identificados pelo especialista, os quais podem ser usados para realimentar futuras análises de modos de falha e efeitos (PFMEA).



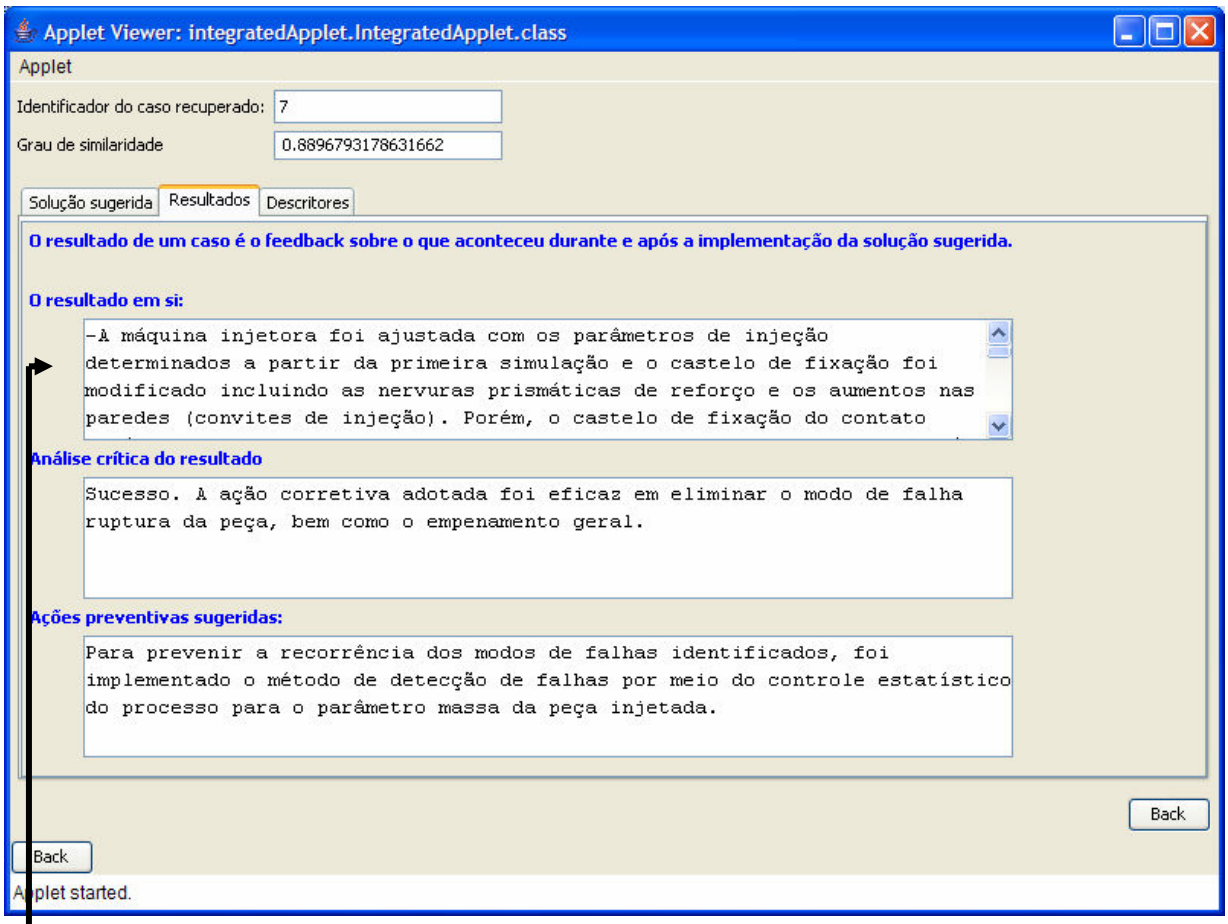
-A resistência mecânica do castelo de fixação do contato metálico deve ser garantida, pois a ruptura deste elemento implica em substituição da tampa completa envolvendo altos custos de garantia e insatisfação do cliente.

-É necessário determinar um método adequado de ensaio e teste da peça com o objetivo de determinar a sua resistência ao carregamento simulando as condições de uso.

-O aspecto dimensional desta peça é essencial, pois ela faz parte de um conjunto Tampa/Base que é montado em uma operação subseqüente, ressaltando-se ainda, que esta tampa recebe outros componentes eletrônicos em uma operação de sub-montagem.

Figura 8.42 – Janela gráfica do Agente de interface RBC – Solução sugerida –Parte IV.

Por fim, a figura 8.43 destaca o *feedback* sobre os fatos que aconteceram durante e após a implementação da solução sugerida pelo especialista, isto é o resultado em si, sua análise crítica e as ações preventivas sugeridas.



-A máquina injetora foi ajustada com os parâmetros de injeção determinados a partir da primeira simulação e o castelo de fixação foi modificado incluindo as nervuras prismáticas de reforço e os aumentos nas paredes (convites de injeção). Porém, o castelo de fixação do contato metálico ainda apresentava ruptura quando submetido ao carregamento. Após a análise da fratura do castelo constatou-se que esta ruptura era causada agora por tensões internas geradas na fase de recalque em função de excessivo empacotamento. Assim, uma nova simulação foi necessária para otimizar os parâmetros de recalque (conforme descrito nos passos de raciocínio) .

-A máquina injetora foi ajustada com os parâmetros de injeção determinados a partir da segunda simulação, isto é: com o novo perfil da pressão de recalque (tempo/pressão) e ponto de comutação (conforme descrito na solução adotada) e o molde foi modificado com o arredondamento de todas as arestas do castelo de fixação para diminuir a concentração de tensões nestas regiões. Estas modificações resolveram com sucesso o problema de ruptura do castelo de fixação.

-No tocante ao empenamento, apesar dos novos parâmetros de injeção, recalque e as modificações do molde, o empenamento ainda era observado logo após a injeção da peça. Assim, novas temperaturas de refrigeração do lado interno (lado do macho) e do lado externo da peça (lado da fêmea) foram estabelecidas de modo experimental (conforme definido na ação corretiva adotada).
Deste modo esta diferença de temperaturas de refrigeração compensa a contração diferencial. Assim, a parte externa da peça passou a resfriar mais lentamente forçando o encurvamento da tampa para fora mantendo assim o aspecto dimensional.

Figura 8.43 – Janela gráfica do Agente de interface RBC – Resultados.

Adicionalmente, outro experimento foi realizado com protótipo do modelo proposto, considerando a mesma configuração de consulta apresentada nas figuras de 8.25 a 8.33, porém convertidas para a língua inglesa, de modo a acessar casos armazenados nas bases de conhecimento dos agentes de recursos RBC neste idioma. E o caso mais similar recuperado pelos agentes relativo à análise e solução de um caso "empenamento" ou "*warpage*" relatado pelo especialista Hatch (IMM, 2007) é apresentado nas figuras 8.44 a 8.48.

Na figura 8.44 o especialista atribui o empenamento da peça moldada (ver figura 8.45) à falta de um detalhe de projeto na região do ponto de injeção, isto é a falta de um "*dimple*" ou o engrossamento da parede da peça no lado oposto ao ponto de injeção necessário para facilitar o fluxo de material (ver figura 8.46). O especialista adverte que ciclos mais lentos de injeção podem ser usados até que o "*dimple*" sugerido na ação corretiva seja implementado.

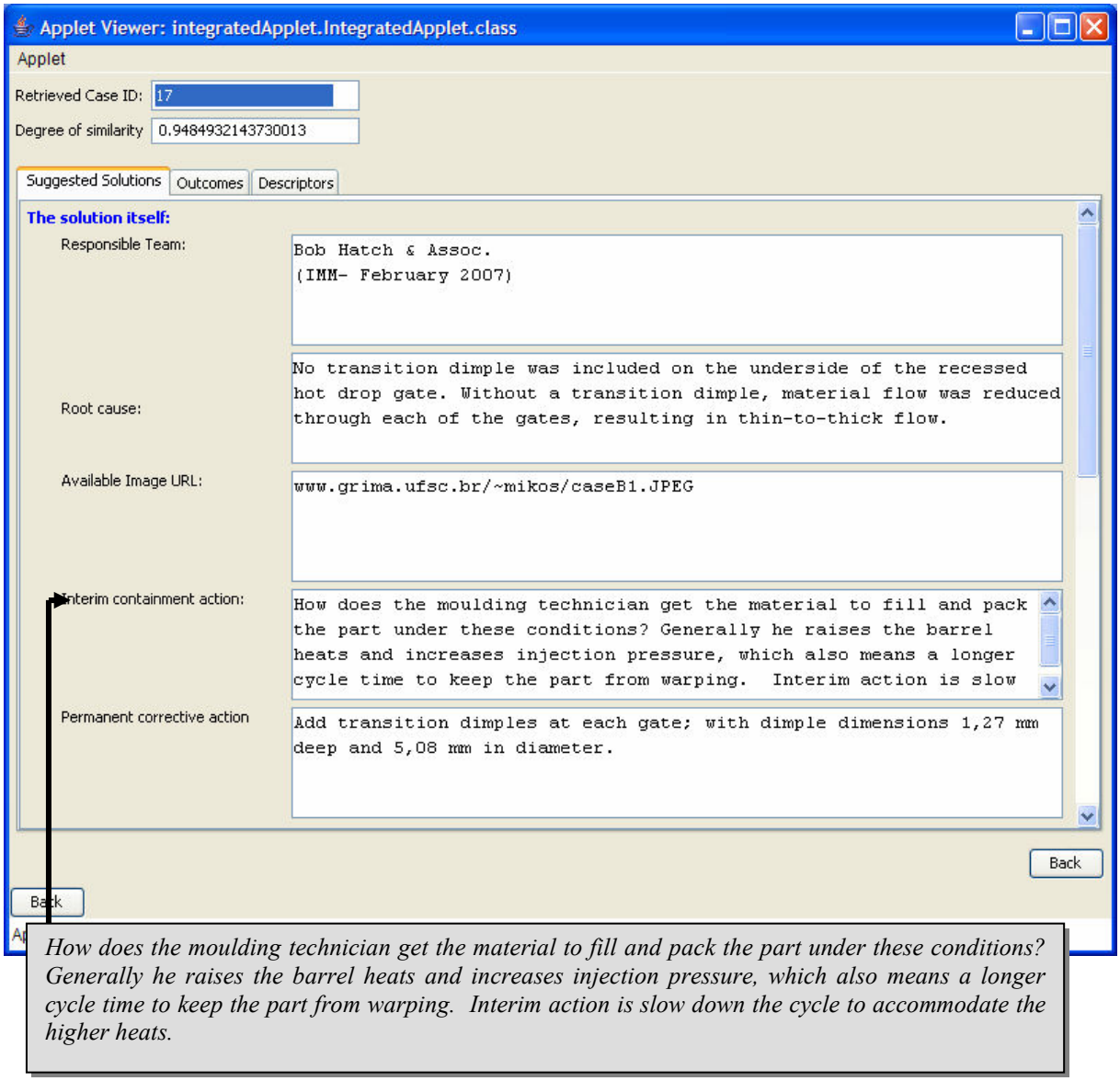


Figura 8.44 – Janela gráfica do Agente de interface RBC – Solução sugerida –Parte I.

Figura 8.45 – Empenamento da peça após a moldagem por injeção.

A figura 8.46 (A) ilustra esquematicamente o projeto da peça moldada, em particular, a região do ponto de injeção, cujo detalhe mostra que a peça foi projetada somente com um rebaixo na parede (*recessed gate*) conforme relato do especialista apresentado na figura 8.47. Enquanto a figura 8.46 (C) ilustra o "*dimple*" ou engrossamento da parede sugerido pelo especialista para o caso.

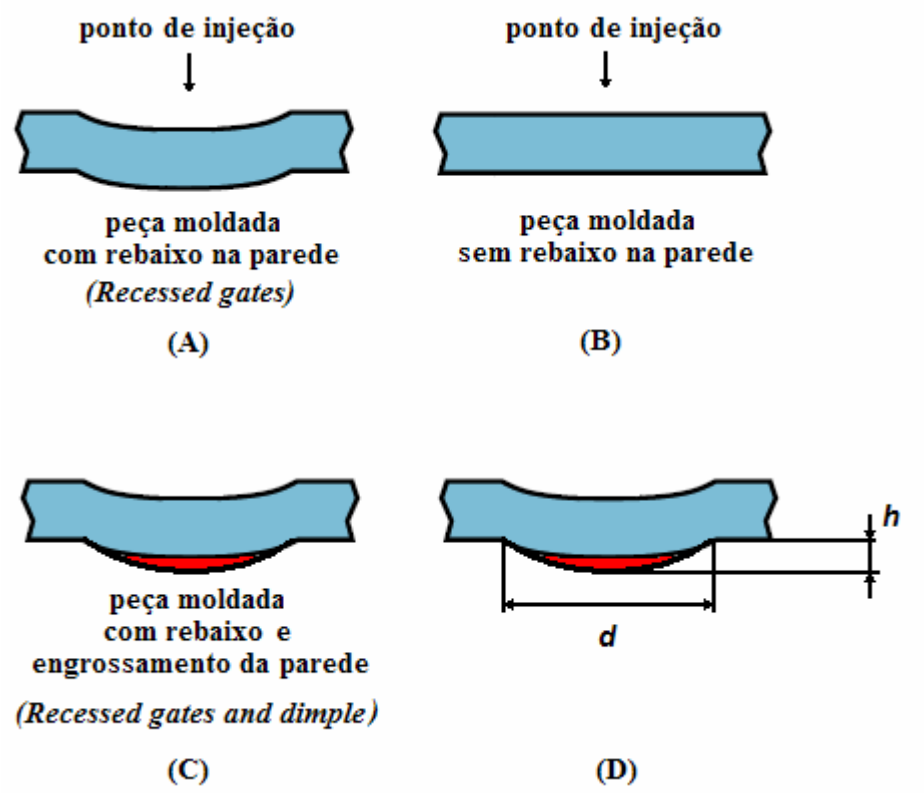
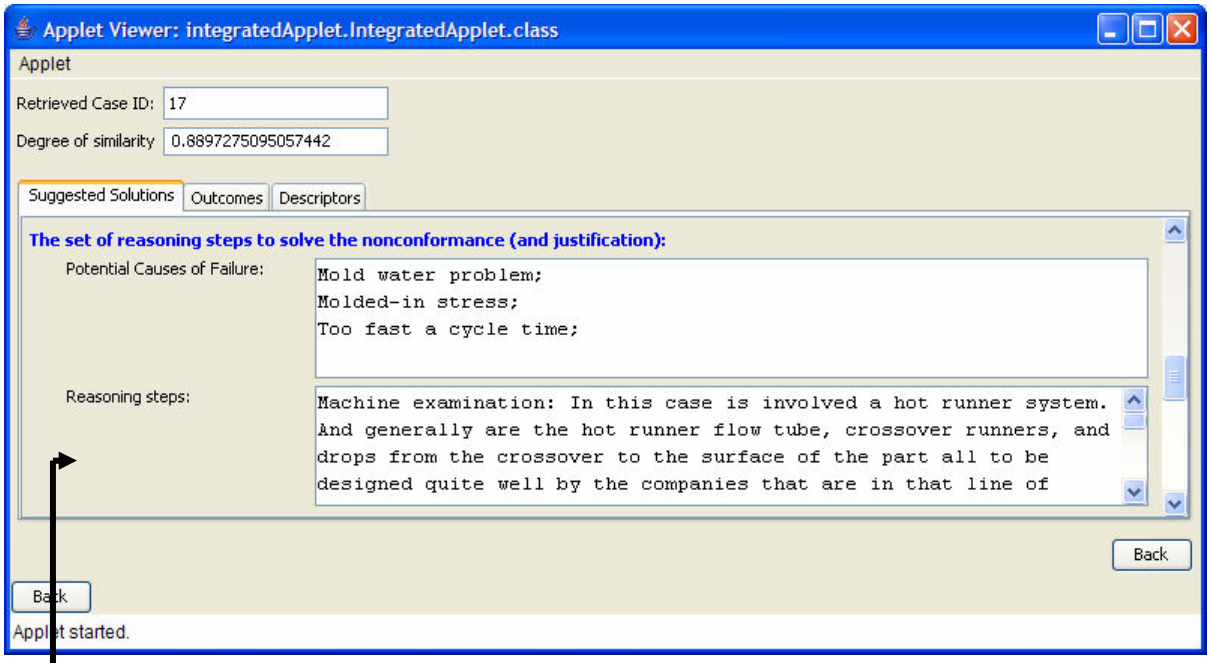


Figura 8.46 – Detalhes do ponto de injeção da peça.

O "*dimple*" é um engrossamento na parede do produto usualmente na forma de um raio esférico (figura 8.46 C e D), cujo objetivo é reduzir as restrições ao fluxo de material fundido no ponto de injeção (REES, 2002). A falta do "*dimple*" pode implicar que o material fundido deve fluir por variações de seções dotadas de cantos vivos, que criam uma restrição do fluxo

causando perda de pressão, redução da velocidade de injeção, bem como gerando tensões internas (REES, 2002).

Por sua vez, a figura 8.47 mostra a linha de raciocínio empregada pelo especialista durante a análise do problema da não-conformidade em questão, destacando a análise do projeto do ponto de injeção, a qual revelou a reduzida espessura de parede na região do ponto de injeção.



Machine examination: In this case is involved a hot runner system. And generally are the hot runner flow tube, crossover runners, and drops from the crossover to the surface of the part all to be designed quite well by the companies that are in that line of business. However, sometimes molders take these well-designed hot runner molds and try to feed them with machine nozzles that have not been drilled out to match the flow tube or bore diameter.

Gate design examination: The hot drop gate was recessed. The part's nominal wall thickness was 2,794 mm and the recess at each gate of the four gates was 1,27 mm. This reduced the wall thickness at each gate to just 1,524 mm. Without a transition dimple, material flow was reduced through each of the gates, resulting in thin-to-thick flow. We also could see that all the gate were vestige free, which told us the hot tip gate must be tapered, as they should be; otherwise, we would see evidence of vestige at some of the gate.

Figura 8.47 – Janela gráfica do Agente de interface RBC – Solução sugerida –Parte II.

A figura 8.48 apresenta uma ação corretiva alternativa sugerida pelo especialista envolvendo ou uso de outro material para a injeção, bem como as suas justificativas para esta ação alternativa.

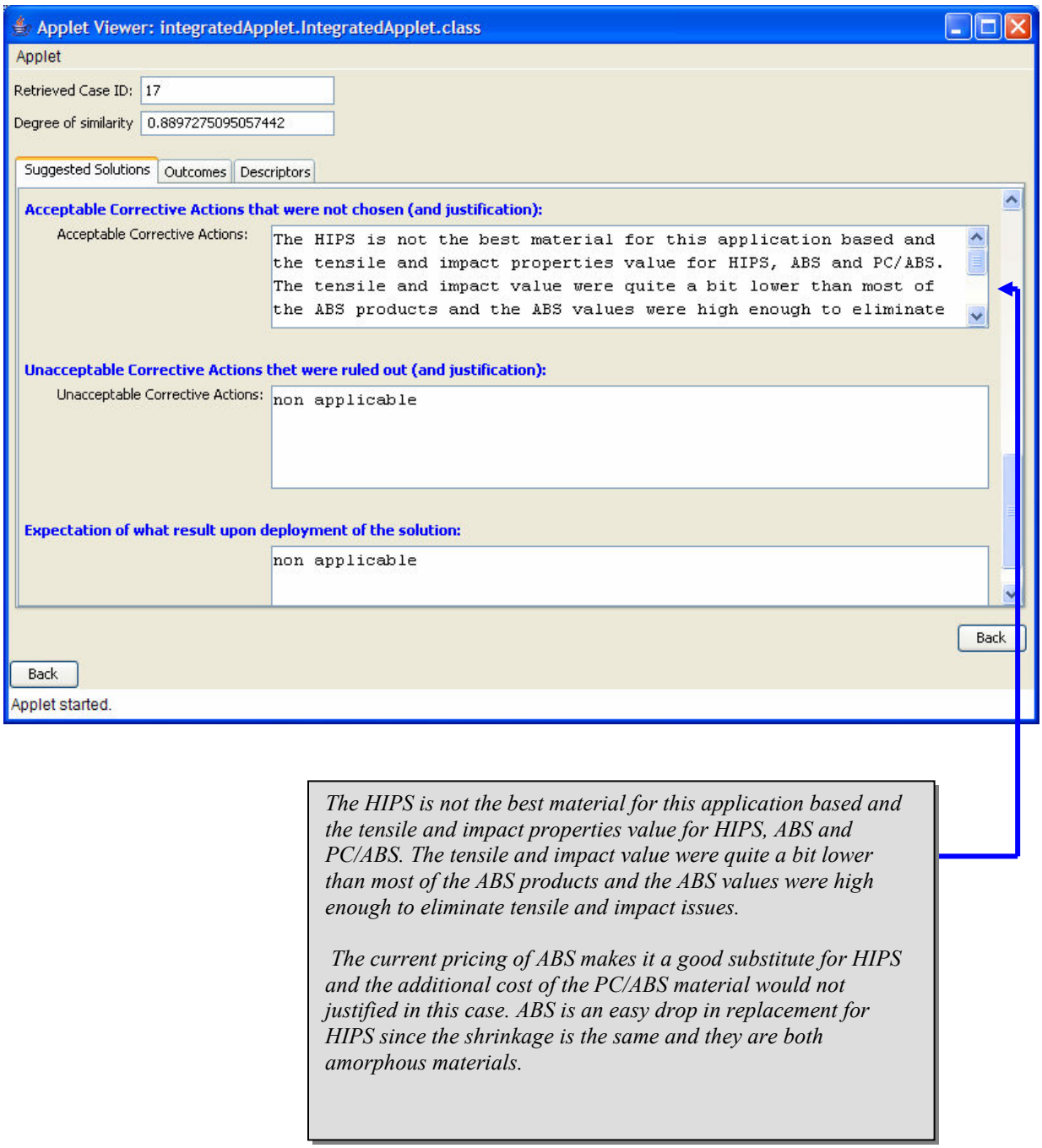


Figura 8.48 – Janela gráfica do Agente de interface RBC – Solução sugerida –Parte III.

Por fim, a figura 8.49 destaca que a solução implementada foi eficaz na solução do problema do modo de falha apresentada no relato dos resultados obtidos após a implementação da solução, bem como revela uma possível ação preventiva para evitar a recorrência deste modo de falha em peças e moldes semelhantes.

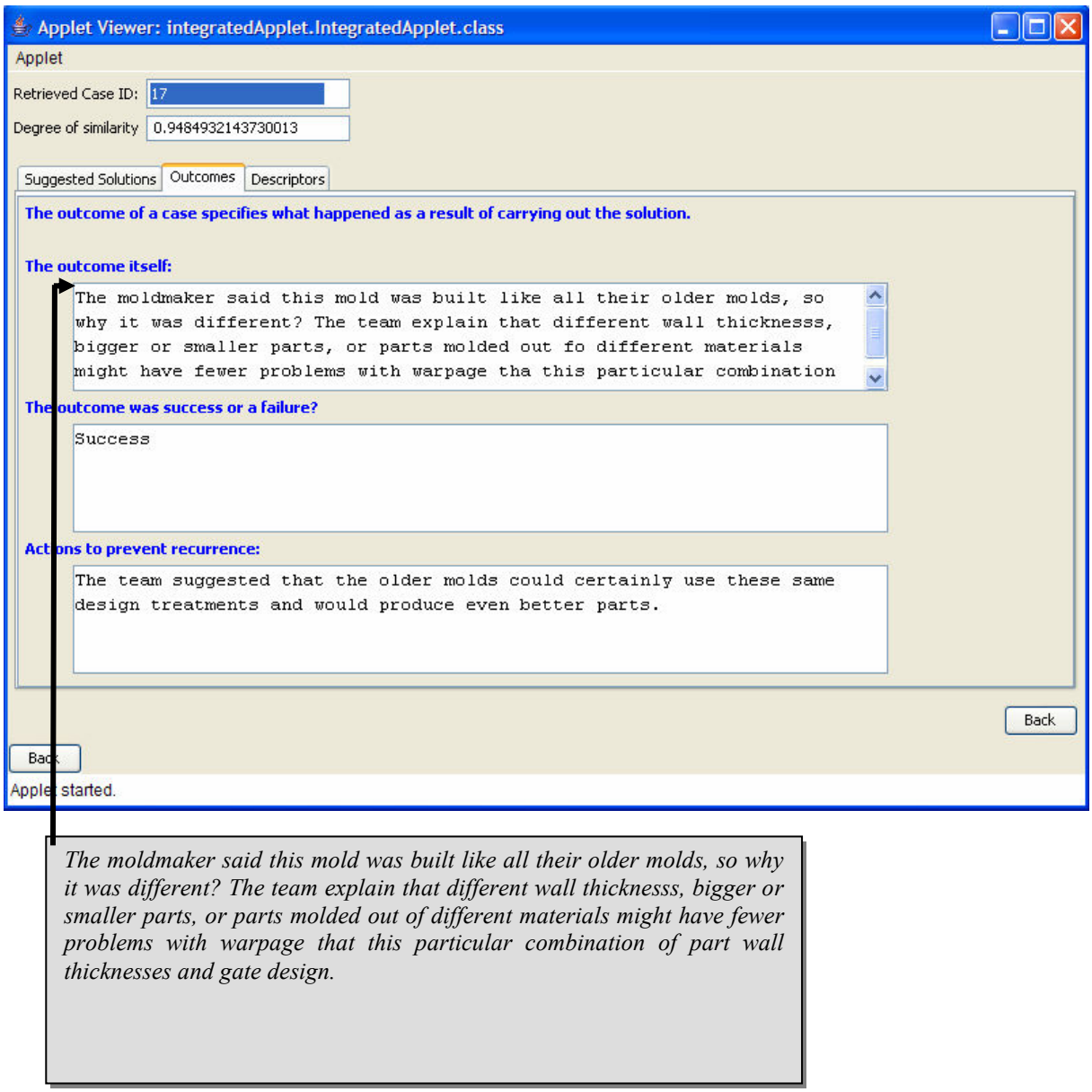


Figura 8.49 – Janela gráfica do Agente de interface RBC – Resultados.

Neste ponto, é importante destacar, que no presente trabalho de tese, o escopo do modelo proposto em relação aos agentes de recursos RBC é apoiar o processo de solução de problemas de não conformidades, fornecendo blocos de conhecimento produzidos por outros especialistas durante a análise e solução de problemas similares. Este apoio visa promover, em primeiro lugar, o entendimento sobre o problema de não-conformidade e contribuir para com os especialistas no sentido de buscar uma solução apropriada e satisfatória para este problema ou mesmo refinar o processo de análise do problema considerando um caminho construtivista como sugere Roy (1993) em seu trabalho sobre o tema apoio à decisão.

Dentro desta perspectiva, particular atenção foi dispensada também à análise da relevância dos descritores correspondentes à solução sugerida e aos resultados que compreendem as "lições aprendidas" armazenadas junto aos casos mais similares recuperados pelos agentes de

recursos RBC e disponibilizados pelo agente de interface como blocos de conhecimento organizados de acordo com a similaridade, nesta análise foram consideradas criticas do especialista consultado, bem como dados da literatura.

Em primeiro lugar, o descritor < *pessoal responsável pela solução* > (figuras 8.35, 8.39 e 8.44) visa, primariamente, indicar o grupo de especialistas envolvidos no processo de solução do problema de não-conformidade e verdadeiros responsáveis pela solução sugerida, pois o modelo proposto neste trabalho de tese contempla a noção de casos obtidos a partir de diferentes fontes e diferentes especialistas.

Todavia, outro aspecto importante a destacar em relação ao descritor < *pessoal responsável pela solução* > diz respeito à atitude frente ao compartilhamento do conhecimento, pois o conhecimento gera valor dentro da cadeia produtiva, e de acordo Stewart (1997) deve ser reconhecido como um valioso capital intelectual. Portanto, o objetivo secundário deste descritor está relacionado à propriedade intelectual do conhecimento gerado nos processos de solução de não-conformidades, isto é, assegurar a referência às pessoas que o geram, ou como denominou Davenport (2005), trabalhadores do conhecimento. Assim, além do modelo proposto servir adequadamente como uma possível estrutura para o compartilhamento do conhecimento, Stewart (1997) adverte que a forma de avaliação e reconhecimento dos recursos humanos que decorrem da cultura das organizações envolvidas podem influenciar significativamente a atitude do indivíduo ou grupo frente ao compartilhamento deste valioso conhecimento.

Por sua vez, o descritor < *causa raiz verificada* > (figuras 8.35, 8.39 e 8.44), que sintetiza grande parte do esforço empreendido pelos especialistas durante o processo de solução de uma não-conformidade. Este descritor em conjunto com o descritor < *imagens disponíveis* > (figuras 8.35, 8.39 e 8.40), podem esclarecer detalhes fundamentais da análise realizada mediante fotos, desenhos entre outros, as quais podem ser acessadas via internet usando-se os navegadores padrão disponíveis, o que minimiza os recursos computacionais necessários.

Por outro lado, o descritor < *ação de contenção adotada* > (figuras 8.36, 8.40 e 8.44) pode expressar os detalhes de procedimentos complexos que visam bloquear a propagação de produtos não-conforme e seus efeitos, os quais são requisitos obrigatórios nos sistemas de gestão da qualidade, meio ambiente e segurança do trabalho. Este descritor pode ser de grande valia quando ações de contenção necessárias devem ser tomadas com urgência, e onde uma decisão baseada em informações incompletas pode trazer graves conseqüências econômicas para as organizações envolvidas.

Em especial, o descritor < *ação corretiva adotada* > (figuras 8.36, 8.40 e 8.44) representa a solução sugerida nos termos do ciclo de raciocínio baseado em casos, a qual pode ser transferida para a situação da não-conformidade corrente caracterizando a etapa de reuso do ciclo RBC.

Assim, com relação ao ciclo RBC, apresenta uma elevada complexidade, pois em geral a solução recuperada deve ser adaptada para que a mesma satisfaça completamente os requisitos da situação problema. A literatura revela que os mais variados graus de modificação podem ser empregados, utilizando-se diferentes técnicas de adaptação, que podem envolver desde a cópia da solução até adaptações realizadas de acordo com regras complexas que refletem um modelo de conhecimento refinado sobre o domínio de aplicação (von WANGENHEIMHEIM e von WANGENHEIM, 2003).

Contudo, para guiar a etapa de reutilização da < *ação corretiva adotada* >, o descritor < *etapas de raciocínio para proceder à análise da não-conformidade* > (figuras 8.37, 8.41 e 8.47) tem um papel essencial, pois pode descrever a linha de raciocínio ou metodologia adotada na solução prévia e, portanto, pode ser repetida e usada como um guia para o processo de adaptação da solução.

Adicionalmente, este descritor pode permitir ao usuário rever o seu próprio processo ou método de análise e solução de problema de não-conformidades tendo em vista o processo ou método adotado por outros especialistas em situações similares, bem como verificar se a não-conformidade relacionada ao processo de manufatura em questão pode estar associada às causas expressas pelos descritores < *causa raiz verificada* > (figuras 8.35, 8.39 e 8.44) ou mesmo < *causas potenciais identificadas para a não-conformidade* > (figuras 8.37, 8.41 e 8.47).

Adicionalmente, o processo de adaptação pode lançar mão dos descritores < *ações corretivas alternativas viáveis* > (figuras 8.42 e 8.48) e < *ações corretivas não viáveis* > (figuras 8.42 e 8.48) que podem englobar um conhecimento valioso sobre ações alternativas já estudadas.

Uma vez que a ação corretiva tenha sido adaptada pelo usuário na etapa de reuso do ciclo RBC, inicia-se a etapa de revisão, que consiste em validar a ação corretiva adaptada, e quando necessário corrigi-la. Esta etapa representa uma parte fundamental do processo e envolve a confirmação que a ação é plausível para a não-conformidade considerada.

A literatura sugere que os critérios para a revisão da ação corretiva adaptada pode ser a própria qualidade da solução, bem como outros critérios específicos definidos para a aplicação em questão, como a facilidade de compreensão da solução por parte do usuário ou a

quantidade de esforço para implementá-la (von WANGENHEIM e von WANGENHEIM, 2003).

Todavia, a literatura revela também que a verificação da ação adaptada é, em muitos casos, realizada durante a sua aplicação prática, embora possa ser substituída por simulações computacionais, por exemplo, mediante o software Moldflow® amplamente usado na área de moldagem por injeção quando a solução indicar modificações nos parâmetros de processo, desenho do molde ou do componente entre outros fatores.

Com esta finalidade os descritores < *expectativas sobre problemas que podem surgir após a implementação da solução* > (figuras 8.42 e 8.48), < *o resultado em si* > (figuras 8.38, 8.43 e 8.49) e < *análise critica do resultado* > (figuras 8.38, 8.43 e 8.49) podem contribuir significativamente para esta etapa, pois representam em última análise o *feedback* sobre o estado do processo após a implementação da solução.

Por fim, o descritor < *ações preventivas sugeridas* > (figuras 8.38, 8.43 e 8.49) pode incluir definições de ações para eliminar as causas de não-conformidades potenciais de forma a evitar a sua recorrência. Ou ainda indicar as medidas necessárias para o controle do processo de manufatura na forma de planos de controle requeridos pelos sistemas de gestão da qualidade.

Então, fechando o ciclo RBC, toda vez que uma não-conformidade for solucionada, a nova experiência deve ser retida e integrada à base de conhecimento, assegurando um aprendizado sustentado, e que este novo conhecimento esteja disponível para apoiar novos usuários.

No presente trabalho de tese, a retenção automática de novas experiências não faz parte do escopo do modelo, mas pode ser realizada pela inclusão do conteúdo dos descritores diretamente nas tabelas do sistema de gerenciamento de banco de dados relacionais, como mostrado no Capítulo 7.

Quanto a este aspecto, a literatura revela que algumas aplicações de RBC condicionam a retenção de novos casos ao atendimento de políticas de retenção e manutenção das bases de casos próprias de cada organização, tendo em vista o aspecto de confiabilidade dos casos relatados (von WANGENHEIM e von WANGENHEIM, 2003).

Ainda quanto à relevância do comportamento colaborativo do modelo com respeito ao seu propósito geral, o especialista consultado durante a pesquisa de campo destaca que o acesso às "lições aprendidas" pode contribuir significativamente para a redução da sobrecarga de trabalho do especialista no tocante ao processo de análise e solução de problemas de não-conformidades, além de facilitar o tratamento inicial de problemas de não-conformidade por outros técnicos no processo.

Uma análise crítica adicional por parte do especialista revelou também que, para facilitar a configuração das consultas, os detalhes dos moldes, materiais e máquinas poderiam ser previamente cadastrados, permitindo-se desta forma minimizar o esforço dedicado ao preenchimento dos descritores relativos a estes elementos durante as consultas, bastando indicar o seu número de cadastro. E sugere também estudar a possibilidade de interligar os agentes de interface do modelo com os controladores da própria máquina, de modo a obter os parâmetros de processo diretamente da máquina.

Por outro lado, no tocante ao processo de aquisição de conhecimento empreendido durante a pesquisa de campo, foi possível comprovar que o conjunto de descritores propostos na estrutura conceitual do caso de não-conformidade ajustou-se adequadamente ao formato dos relatos das experiências de solução de não-conformidade feitos pelo especialista da empresa, pois em essência estes relatos foram consistentes com as linhas observadas na literatura.

Além disso, foi possível comprovar também que este conjunto de descritores mostrou-se adequado em relação ao propósito de guiar a aquisição de conhecimento relevante, considerando-se tanto a perspectiva dos descritores usados para configurar uma consulta quanto a perspectiva daqueles que descrevem a solução sugerida e os respectivos resultados esperados, que são recuperados com os casos mais similares. Este conjunto indica claramente quais aspectos devem ser explorados e quais informações necessitam ser obtidas de acordo com a estratégia de aquisição adotada.

8.3.1 Validação operacional do modelo sob o prisma dos agentes de recursos PFMEA

Com o objetivo de analisar a validade operacional do modelo proposto, mas agora sob prisma dos agentes de recursos de recurso PFMEA dois experimentos foram realizados. Neste sentido, a figura 8.50 mostra a interface gráfica destinada a capturar as informações primárias relacionadas à ocorrência da não-conformidade, bem como identificar a perspectiva de apoio à solução de problemas de não-conformidades requerida pelo usuário.

Nestes experimentos as consultas foram direcionadas aos agentes de recursos PFMEA, cujas bases de conhecimento são preenchidas com os conhecimentos decorrentes da aplicação do método de análise de falhas e efeitos em processos de manufatura (Botão 2). A janela gráfica indica que a língua inglesa foi selecionada para a configuração das consultas, pois as bases de conhecimentos dos agentes de recursos PFMEA foram preenchidas com conhecimentos codificados neste idioma.

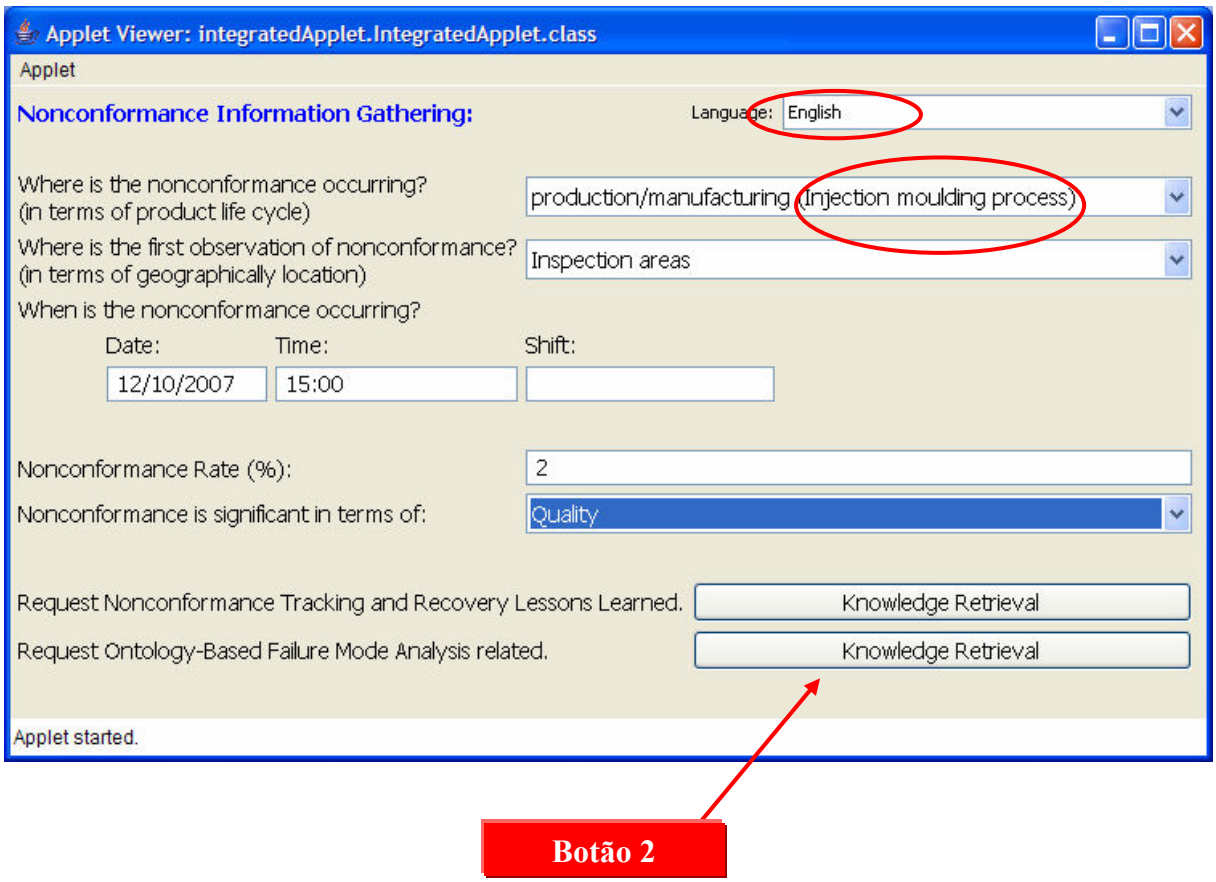
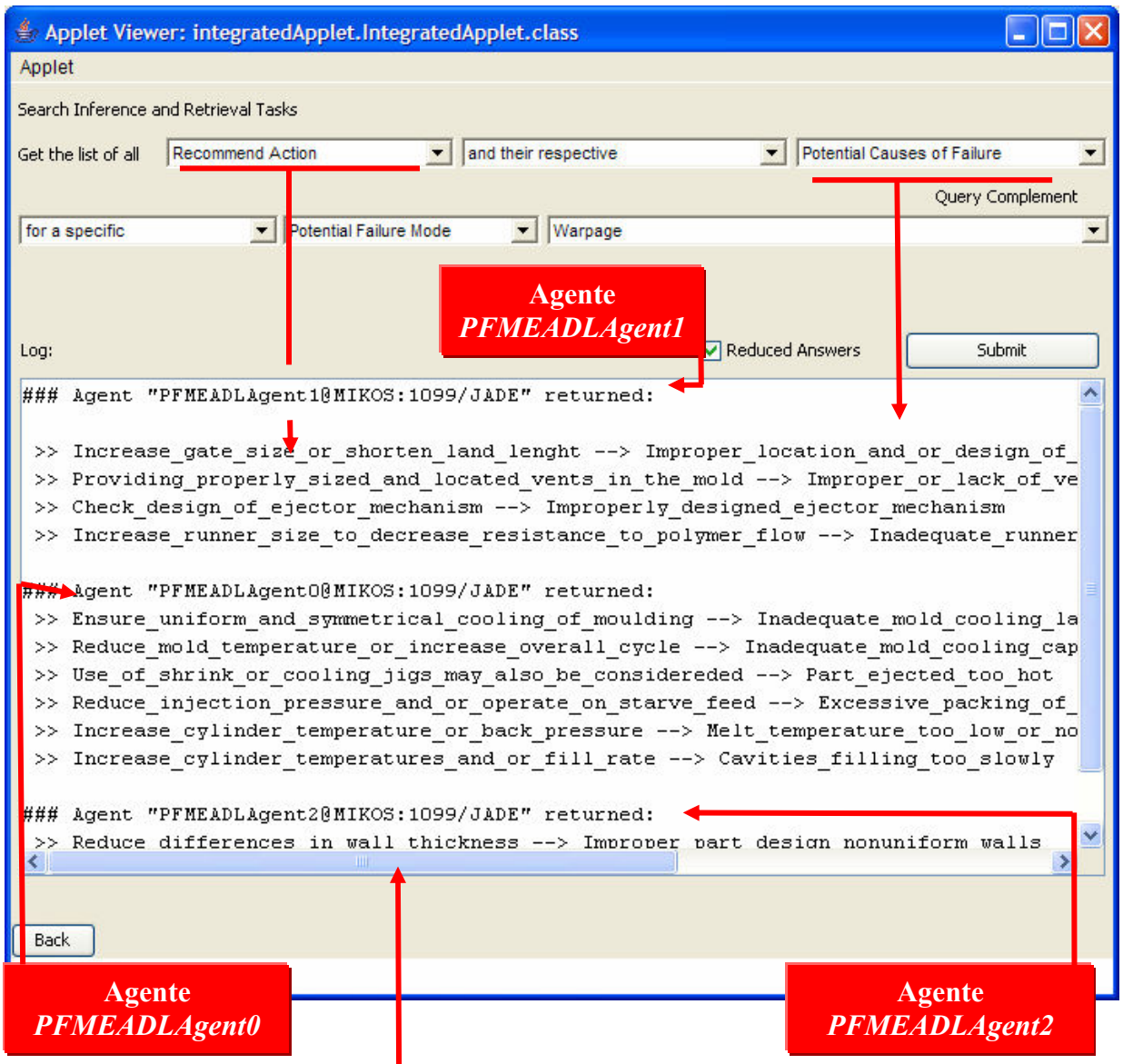


Figura 8.50 – Janela de interface gráfica do protótipo apresentada ao usuário.

A figura 8.51 mostra a janela gráfica do Agente de Interface PFMEA preparada para consultas na língua inglesa. No primeiro experimento realizado, a consulta configurada junto ao agente de interface PFMEA tem por objetivo recuperar todas as ações recomendadas (*Recommended Actions*) para as respectivas causas em potencial (*Potential Causes of Failure*) de um modo de falha específico (*Potential Failure Mode*), no caso o modo de falha "empenamento" ou "*warpage*" disponíveis nas bases de conhecimentos dos agentes de recursos PFMEA ativos na organização multiagente.

Neste sentido, é possível observar na figura 8.51 os resultados enviados pelos agentes de recursos de conhecimento PFMEA, destacando-se que três diferentes agentes de recursos PFMEA que se encontravam ativos na organização multiagente forneceram os resultados da consulta submetida pelo agente de interface, os agentes *PFMEADLAgent0*, *PFMEADLAgent1* e *PFMEADLAgent2*.



Agent "PFMEADLAgent1@MIKOS:1099/JADE" returned:
>> Increase_gate_size_or_shorten_land_lenght --> Improper_location_and_or_design_of_gates
>> Providing_properly_sized_and_located_vents_in_the_mold --> Improper_or_lack_of_venting
>> Check_design_of_ejector_mechanism --> Improperly_designed_ejector_mechanism
>> Increase_runner_size_to_decrease_resistance_to_polymer_flow --> Inadequate_runner_system

Agent "PFMEADLAgent0@MIKOS:1099/JADE" returned:
>> Ensure_uniform_and_symmetrical_cooling_of_moulding -->
Inadequate_mold_cooling_lack_of_balance_in_mold_halves_differential_coolingl
>> Reduce_mold_temperature_or_increase_overall_cycle -->
Inadequate_mold_cooling_capacity_of_system
>> Use_of_shrink_or_cooling_jigs_may_also_be_considereded --> Part_ejected_too_hot
>> Reduce_injection_pressure_and_or_operate_on_starve_feed --> Excessive_packing_of_cavity
>> Increase_cylinder_temperature_or_back_pressure --> Melt_temperature_too_low_or_nonhomogeneus
>> Increase_cylinder_temperatures_and_or_fill_rate --> Cavities_filling_too_slowly

Agent "PFMEADLAgent2@MIKOS:1099/JADE" returned:
>> Reduce_differences_in_wall_thickness --> Improper_part_design_nonuniform_walls
>> Undercuts_should_not_have_sharp_angles --> Improper_design_of_undercuts_ribs_bosses_threads

Figura 8.51 – Interface gráfica do agente de interface PFMEA.

É importante notar, que o experimento apresentado na figura 8.51 busca demonstrar as potencialidades para o emprego colaborativo de diferentes agentes de recursos PFMEA, no qual os métodos de inferência e recuperação de conhecimento são capazes de acessar de bases que representam os conhecimentos decorrentes da aplicação do método de análise do modo de falha e efeitos de diferentes do ponto de vista. Isto é, análise de modos de falha associados: à construção do molde, à operação do processo de moldagem em si e associados ao projeto da peça moldada, respectivamente.

Neste contexto, o agente *PFMEADLAgent1* forneceu respostas à consulta submetida identificando as possíveis causas relacionadas ao projeto do sistema de alimentação, localização e dimensionamento dos pontos de injeção, ventilação da cavidade e extratores do molde.

Por sua vez, o agente *PFMEADLAgent0* apresentou as possíveis causas que foram atribuídas a operação do processo de moldagem, tais como: refrigeração das cavidades, preenchimento e empacotamento da cavidade, temperatura do material e temperatura de extração da peça. E o agente *PFMEADLAgent2* apresentou as possíveis causas que podem ser atribuídas ao projeto da peça moldada.

A figura 8.52 apresenta a configuração de uma consulta que tem por objetivo recuperar todos os possíveis modos de falha (*Potential Failure Mode*) e as suas respectivas causas (*Potential Causes of Failure*) que foram atribuídos a uma *feature* em particular durante a aplicação do método PFMEA, no caso à uma parede paralela à linha de partição do molde "*Wall_parallel_ to_the_ parting_line*".

Este experimento visa demonstrar entre outras a potencialidade do uso do conceito *feature* modelado na ontologia PFMEA DL e usada nas bases de conhecimento dos agentes de recursos PFMEA. O uso deste conceito, suas relações e instâncias permitem que consultas possam ser configuradas diretamente a partir das *features* de moldabilidade de uma peça, sem a necessidade de usar a indicação de um nome ou referência de componente específico para refinar a consulta, tornando assim as consultas às bases de conhecimento mais abrangentes.

É importante destacar também que o processo de aquisição de conhecimento empreendido permitiu comprovar que o componente *intensional* TBox formalizado na ontologia PFMEA DL é passível de reutilização, considerando a operacionalização de conhecimento de referência em dois diferentes domínios ou processos de produção dentro de uma cadeia produtiva (ver figura 8.23).

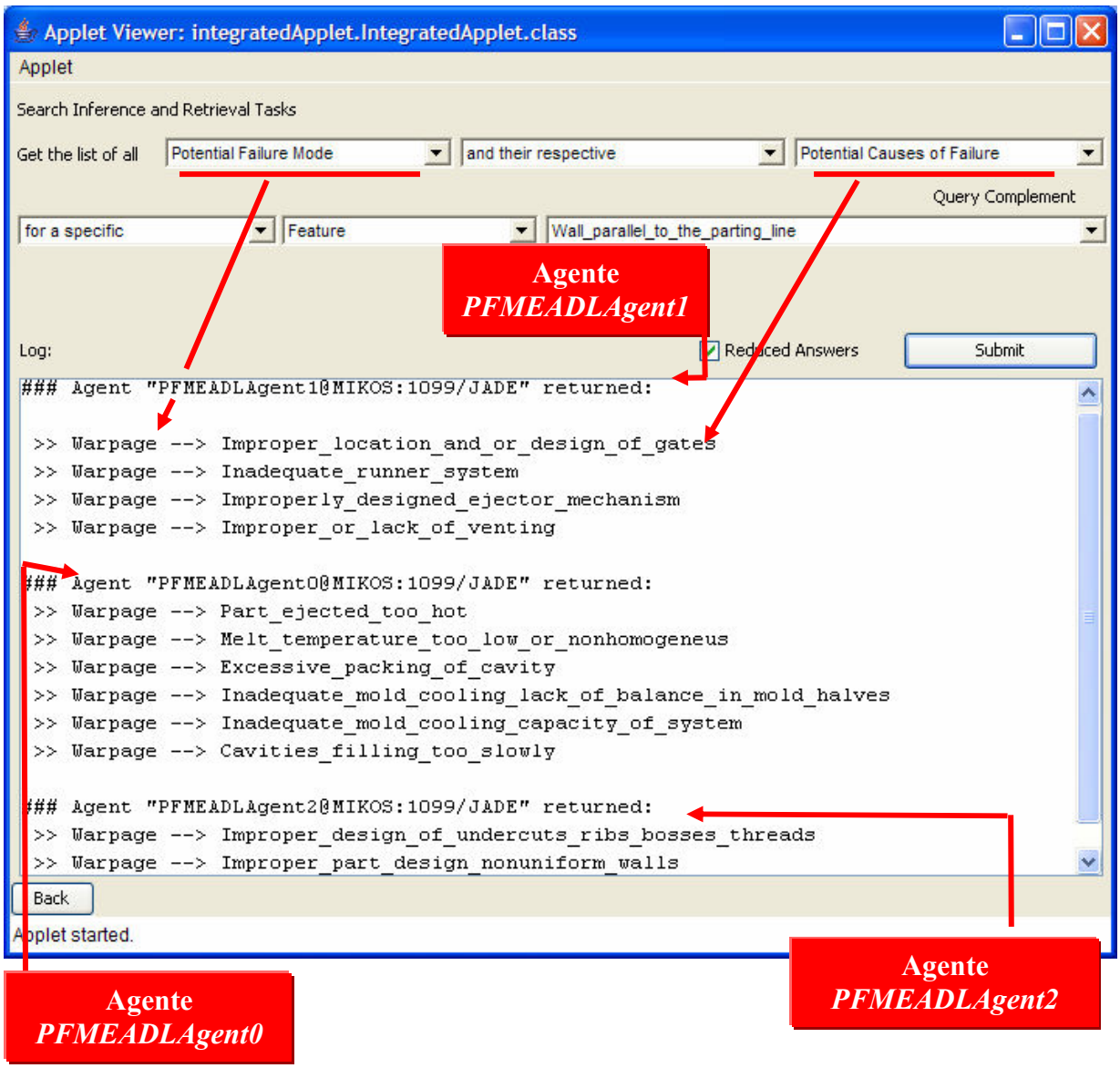


Figura 8.52 – Interface gráfica do agente de interface PFMEA.

Por fim, a análise critica identificou-se uma oportunidade de melhoria no que diz respeito à aquisição de conhecimento, em especial, para aquelas organizações que já dispõem de sistemas de bancos de dados relacionais próprios para armazenamento de dados do método de análise de modos de falha e efeitos. Assim, é possível estudar a forma de automatizar o mapeamento de instâncias das classes OWL DL diretamente de tabelas de bancos de dados já disponíveis.

8.4 SÍNTESE DO CAPÍTULO

Este capítulo foi dedicado a apresentar e discutir o processo de verificação, validação conceitual e operacional do modelo baseado em agentes proposto pelo autor. 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 conseqüência de sua base de conhecimento e, apenas de forma secundária, conseqüência 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.

CAPÍTULO 9
CONCLUSÕES E RECOMENDAÇÕES

A pesquisa descrita nesta tese explorou o potencial 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 estabelecidas no modelo proposto representam conceitualmente abstrações que visam superar as barreiras impostas pela complexidade inerente tanto às perspectivas ou cenários de apoio quanto à distribuição espacial e organizacional das fontes. Além da própria diversidade das características envolvidas em uma aplicação desta natureza, tais como: diferenças entre *hardwares*, sistemas operacionais e tipos de redes.

O modelo proposto foi implementado, na forma de um protótipo de laboratório, de modo a permitir a validação do trabalho de pesquisa empreendido nesta tese.

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

9.1 DISCUSSÃO

O modelo baseado em agentes proposto neste trabalho de 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 particularmente 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, respectivamente. 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, sair 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 guiar o usuário durante a configuração de consultas personalizadas.

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

E 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 proposto 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 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.

Constatou-se ainda que o componente *intensional* TBox (*terminological component*) formalizado na ontologia PFMEA é passível de reutilização em diferentes processos de manufatura considerando a operacionalização de conhecimentos de referência em dois diferentes domínios.

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 (***J**ava **A**gent **DE**velopment Framework*) desenvolvido em conformidade às 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-TOMÁS 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.

E o sistema RacerPro Server (*Renamed ABox and Concept Expression Reasoner Professional*), uma nova versão do sistema Racer desenvolvido por Haarslev e Möller (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 foi implementada 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 CONCLUSÕES

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

1. O modelo proposto pode ser considerado como uma abordagem viável e promissora para o compartilhamento e reuso de conhecimentos entre especialistas e agentes computacionais no domínio de soluções de problemas de não-conformidades, tendo em vista os objetivos geral e específicos estabelecidos no capítulo introdutório.

2. O modelo proposto permite também um suporte adequado às ações de raciocínio e a recuperação de conhecimentos no domínio pretendido.

3. As bases de conhecimentos operacionalizadas pelos agentes de recursos de conhecimento podem representar de forma adequada o conhecimento decorrente 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 complexos.

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 permite ao usuário a possibilidade de ajustar 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.

Entretanto, cumpre observar que a implementação do modelo proposto neste trabalho de tese está limitada ao desenvolvimento de um protótipo de laboratório, sendo considerado o processo de moldagem por injeção de termoplásticos.

Todavia, o protótipo desenvolvido foi capaz de demonstrar 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.

9.3 CONTRIBUIÇÕES

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

1. O próprio modelo conceitual e a especificação de projeto da organização multiagente em apoio à solução a solução de problemas de não-conformidades de processos em ambientes de manufatura com recursos distribuídos. → Observando-se que estes foram desenvolvidos a partir da questão norteadora da pesquisa e em consonância aos objetivos geral e específicos estabelecidos no capítulo introdutório.

2. A definição e a formalização por meio de vetores atributo-valor da estrutura conceitual que descreve os casos de não-conformidade, a qual envolve os descritores da não-conformidade *per se*, a solução sugerida e os resultados obtidos após a implementação solução. → Ressaltando-se que a estrutura proposta representa o cerne da base de conhecimento que será acessada e manipulada pelos agentes de recursos de

conhecimento RBC previstos no domínio do processo de moldagem por injeção de termoplásticos.

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 cerne da base de conhecimento que será acessada e manipulada pelos agentes de recursos de conhecimento PFMEA.

4. O próprio desenvolvimento do protótipo a partir das especificações de projeto do modelo. → Onde os detalhes formais da implementação do modelo de comportamento de cada uma das classes de agentes propostos tem por objetivo indicar, em termos metodológicos, a forma de implementação visando permitir a construção de novas aplicações.

9.4 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

As recomendações para trabalhos futuros têm por objetivo indicar aspectos que demandam novas pesquisas e desenvolvimentos.

1. A pesquisa de campo revelou que para facilitar a configuração das consultas, os detalhes dos moldes, materiais e máquinas poderiam ser previamente cadastrados. Desta forma, seria possível minimizar o esforço dedicado à escolha dos descritores relativos a estes elementos.
2. Estudar a possibilidade de interligar os agentes de interface do modelo com os controladores da própria máquina, de modo a obter os parâmetros de processo diretamente da máquina.

3. Explorar e desenvolver novos agentes de interface e de recursos para outros processos de manufatura complexos estendendo os modelos conceituais propostos.

4. Investigar métodos para automatizar o mapeamento de instâncias das classes OWL DL diretamente de tabelas de 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.

5. Investigar métodos para automatizar o mapeamento do conhecimento factual dos 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.

REFERÊNCIAS BIBLIOGRÁFICAS

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

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

ABECKER, Andreas; BERNARDI, Ansgar; van ELST, Ludger. Agent Technology for Distributed Organizational Memories: The Frodo Project. In: International Conference on Enterprise Information Systems, 5th, Angers, France, 2003. **Proceedings...** France: Artificial Intelligence and Decision Support Systems, 2003, v. II, p. 3-10.

ADAMANTIOS, M.; ROCK, D. Intellectual Capital: Utilizing the web for knowledge management and data utilization in reliability engineering. In: International Symposium on Product Quality & Integrity, 2002, Seatle, USA. **Proceedings…** USA: Annual Reliability and Maintainability Symposium, IEEE, 2002, p. 379-385.

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 junho 2006.

AGENTBUILDER LITE. An Integrated Toolkit for Constructing Intelligent Software Agents – Release 2004. Disponível em: <http://www.agentbuilder.com/>. Acesso em: 10 maio 2006.

ALAVI, M.; LEIDNER, D. Review: knowledge management and knowledge management systems: conceptual foundations and research issues. **MIS Quarterly,** v. 25, n. 1, p. 107-136, 2001.

ALAVI, Maryam. **KPMG Peat Marwick U.S.: One Giant Brain** – Study Case 9-397-108, Boston: Harvard Business School Publishing, 1997.

AMERICAN SOCIETY FOR TESTING AND MATERIALS. **ASTM D 1238:** rev. C Standard Test Method for Melt Flow Rates of Thermoplastics by Extrusion Plastometer. West Conshohocken, PA, 2004.

AMERICAN SOCIETY FOR TESTING AND MATERIALS. **ASTM D 792**: Standard Test Methods for Density and Specific Gravity (Relative Density) of Plastics by Displacement. West Conshohocken, PA, 2000.

AMERICAN SOCIETY FOR TESTING AND MATERIALS. **ASTM D1600-07:** Standard Terminology for Abbreviated Terms Relating to Plastics. West Conshohocken, PA, 2007. 10 p.

ARPÍREZ, Julio C.; CORCHO, Oscar; FERNÁNDEZ-LÓPEZ, Mariano; GÓMEZ-PÉREZ. Asunción. **WebODE: A scalable workbench for ontological engineering**. In: International Conference on Knowledge Capture, First, 2001, Canada. **Proceedings…** Canada: ACM Press, 2001. p. 6-13.

ARUNAJADAI, S. G.; STONE, R. B.; TUMER, I. Y. A framework for creating a function-based design tool for failure mode identification. In: ASME Design Theory and Methodology Conference, Montreal, Canada. **Proceedings…** DTEC2002/DTM-34018, 2002.

ASSOCIAÇÃO BRASILEIRA DA INDÚSTRIA DO PLÁSTICO. **Perfil da Indústria Brasileira de Transformação de Material Plástico 2006**. São Paulo, 2006.

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

ASSOCIACÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 9000**: Sistemas de gestão da qualidade – Fundamentos e vocabulário. Rio de Janeiro, 2000. 26p.

ASSOCIACÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 9000**: Sistemas de gestão da qualidade – Fundamentos e vocabulário. Rio de Janeiro, 2005. 35 p.

ASSOCIACÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 9001**: Sistemas de gestão da qualidade – Requisitos. Rio de Janeiro, 2000. 21 p.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 9004**: Sistemas de gestão da qualidade – Diretrizes para melhorias de desempenho. Rio de Janeiro, 2000. 48p.

ASSOCIATION OF PLASTICS MANUFACTURERS – PLASTICSEUROPE. **Plastic and Economy.** Disponível em: <http://www.plasticseurope.org/>. Acesso em: 20 setembro 2007.

ATAMER, A. Comparison of FMEA and Field-Experience for a Turbofan Engine with Application to Case-Based Reasoning. In: Aerospace Conference, 2004, Big Sky, MT, USA **Proceedings...** USA: IEEE Aerospace Conference, 2004. v. 5, p. 3354-3360.

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.

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.

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.

BANDINI, S.; COLOMBO, E.; SARTORI, F.; VIZZARI, G. Case-Based reasoning and production process design: The case of P-Truck curing. In: European Conference on Case-Based Reasoning ECCBR, 7th, 2004, Madrid, Spain. **Proceedings...** Germany: Advances in Case-Based Reasoning – Lectures Notes in Computer Science, 2004, v. 3155, p. 504–517.

BARIANI, P. F.; SALVADOR, M.; LUCCHETTA, G. Development of a test method for the rheological characterization of polymers under the injection molding process conditions. **Journal of Materials Processing Technology,** v. 191, p. 119–122, 2007.

BASF PLASTICSPORTAL EUROPE, Technical Resource, 2007, Injection Moulding Troubleshooter. Disponível em: <http://worldaccount.basf.com>. Acesso em: 10 julho 2007.

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.

BAYER AG., BAYER POLYMERS, **Bayer Material Science, Resources, 2007b, Troubleshooting - Injection Molding**. Disponível em: <http://bayermaterialsciencenafta. com/resources/>. Acesso em: 10 outubro 2007.

BAYER AG., BAYER POLYMERS, **The Injection Molding of Quality Molded Parts. ATI 1146e Increasing Productivity through Process Optimization**, Leverkusen, Germany, 2007a. 26 p.

BECHHOFER, S.; HORROCKS, I.; GOBLE, C.; STEVENS, R. Oiled: a reasonable ontology editor for the semantic web. In: International Description Logics Workshop (DL-2001), 2001, Stanford, USA. **Proceedings…** Stanford: Working Notes, CEURS-WS online, v. 49. Disponível em: <http://CEUR-WS.org/Vol-49/>. Acesso em: 17 julho 2006.

BECHHOFER, Sean; MÖLLER, Ralf; CROWTHER, Peter. The DIG Description Logic Interface. In: International Workshop on Description Logics (DL2003), 2003, Italy. **Proceedings…** Italy: CEUR-WS, 2003. p. 1-8.

BECKMAN, T. A methodology for knowledge management. In: International Conference on AI and Soft Computing - IASTED, 1997, 16th, Calgary. **Proceedings...** Calgary: ACTA Press, 1997, p. 29–32.

BELLIFEMINE, Fabio; CAIRE, Giovanni; TRUCCO, Tiziana; RIMASSA, Giovanni. **JADE Programmers guide** – last update: 21 August 2006. JADE 3.4. Disponivel em: <http://jade.tilab.com/>. Acesso em: 10 maio 2006a.

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: 10 maio 2006b.

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.

BERGENTI, F.; POGGI, A.; RIMASSA, G.; TURCI, P. CoMMA: a multi-agent system for corporate memory management. In: International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3, 1th, 2002, Bologna, Italy. **Proceedings...** New York: ACM, 2002.

BINDER, M.; CLEGG, B. Enterprise management: A new frontier for organizations. **International Journal of Production Economics,** v. 106, n. 2, p. 409-430, 2007.

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).

BORDINI, R.H.; MOREIRA, A.F. Proving BDI properties of agent-oriented programming languages. **Annals of Mathematics and Artificial Intelligence.** v. 0, p.1-30, 2004.

BORST, Willem Nico. **Construction of Engineering Ontologies.** 1997. 227 f.. Ph. D-thesis series n. 97-14. Centre for Telematica and Information Technology, University of Tweenty Enschede, The Netherlands, 1997.

BRACHMAN, Ronald J.; LEVESQUE, Hector J. **Knowledge Representation and Reasoning.** San Francisco, CA: Morgan Kaufmann Publishers, 2004. 381 p.

BRATMAN, M.E.; ISRAEL, D.J.; POLLACK, M.E. Plans and resource-bounded practical reasoning. **Computational Intelligence**, v. 4, p. 349–355, 1988.

BRENNER, Walter; ZARNEKOW, Rudiger; WITTIG, Harmut. Intelligent Software Agents: Foundations and Applications. New York: Springer-Verlag, 1998. 326 p.

BRESCIANI, P.; PERINI, A.; GIORGIONI, P; GIUNCHIGLIA, F.; MYLOPOULOS, J. Tropos: An Agent-Oriented Software Development Methodology. **Journal of Autonomous Agents and Multi-Angent Systems.** v. 8, p.203-236, 2004.

BRYCE, Douglas M. **Troubleshooting: A guide for injection molders – eBook version.** USA: Internet Publications Library and Service, 2003.

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.

BUSSMANN, Stefan; JENNINGS, Nicholas R.; WOOLDRIDGE, Michael. **Multiagent Systems for Manufacturing Control: A Design Methodology.** Heidelberg: Springer-Verlag, 2004. 288 p.

CAIRE, G.; LEAL, F. **Message: Methodology for Engineering Systems of Software Agents – Recommendations on supporting tools**. Eurescom Technical Information, 2001. Disponível em: <http://www.eurescom.de/~pub-deliverables/P900-series/P907/TI2/>. Acesso em: 28 janeiro 2005.

CAMARINHA-MATOS, L.M.; AFSARMANESH, H. The virtual enterprise concept. In: Working Conference on Infrastructures for Virtual Enterprises: Networking Industrial Enterprises, IFIP TC5 WG5.3 / Prodnet Working, 1999, Porto, Portugal. **Proceedings…** Boston: Kluwer Academic, 1999. p. 3 –14.

CAMARINHA-MATOS, L.M.; PANTOJA-LIMA, C. Cooperation coordination in virtual entrerprises. **Journal of Intelligent Manufacturing,** v. 12, n. 2, p. 133-150, 2001.

CAMPAGNARO, Carlos Alberto. **Proposição de uma estrutura referencial para tratamento de não conformidades em componentes produtivos do setor automotivo.** 2007. 187 f.. Dissertação (Mestrado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção e Sistemas, Pontifícia Universidade Católica do Paraná, Curitiba, 2007.

CANCICLIERI JUNIOR, Osiris. Product Model Based Translation Mechanism to Support Multiple Viewpoints in the Design for Manufacture of Injection Moulded Products. 1999. 369 f.. Doctoral Thesis - Department of Manufacturing Engineering, Loughborough University, 1999.

CANCICLIERI JUNIOR, Osiris.; YOUNG, Robert I. M. Information sharing in multiviewpoint injection moulding design and manufacturing. **International Journal of Production Research,** v. 41, n. 7, p. 1565-1586.

CAPRA, Fritjof. **Teia da Vida.** Sao Paulo: Cultrix, 1996. 256 p.

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

CECIL, J.; DAVIDSON, S.; MUTHAYAN, A. A distributed internet-based framework for manufacturing planning. **International Journal Advanced Manufacturing Technology,** v. 27, n. 5-6, p. 619-624, 2006.

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.

CHEN, Mu-Yen; CHEN, An-Pin. Knowledge management performance evaluation: a decade review from 1995 to 2004. **Journal of Information Science,** v. 32, n. 1, p. 17-38, 2006.

CHEN, Mu-Yen; TSAI, M. J.; WU, H. R. The research of KM operation module in science and technology industry – case study of TSMC. In: International Conference of Information Management, 12th, Taipei, 2001. **Proceedings...** Taipei: CSIM Press, 2001, A-75.

CHEN, Z.; TURNG, L-S. A review of current developments in process and quality control for injection molding. **Advances in Polymer Technology**, v. 24, n. 3, p. 165-182, 2005.

CHIN, K.-S.; DUAN, G.; TANG, X.-Q. A computer-integrated framework for global quality chain management. **International Journal of Advanced Manufacturing Technology,** v. 27, n. 5-6, p. 547-560, 2006.

CHRYSLER CORPORATION, FORD MOTOR COMPANY E GENERAL MOTORS CORPORATION. **Advanced Product Quality Planning (APQP) and Control Plan**. 1. ed. IQA, 1997. 101 p.

CO-ODE. **Collaborative Open Ontology Development Environment- About the Project**. Disponível em: <http://www.co-ode.org/about/>. Acesso em: 15 setembro 2006.

COOK, C.; COOK, M. **The Convergence of Knowledge Management and Business Intelligence.** New York: Auerbach Publications, 2000. Disponível em: <www.brint.com/ members/online/20080108/intelligence/> Acesso em: 25 setembro 2005.

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.

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.

COSSENTINO, M.; POTTS, C. PASSI: **A process for specifying and implementing multi-agent systems using UML**. 2001. Disponível em: <http://citeseer.ist.psu.edu/668818.html> . Acesso em: 20 setembro 2007.

CROSBY, P. B. **Quality Without Tears: The Art of Hassle-free Management**. New York: McGraw-Hill, 1984. 205 p.

D'INVERNO, M.; LUCK, M.; GEORGEFF, M.; WOOLDRIDGE, M. The dMARS Architecture: a Specification of the Distributed Multi-Agent Reasoning System. **Journal of Autonomous Agents and Multi-Agent Systems**. v. 9, p.5-53, 2004.

DAHLBERG, Ingetraut. Teoria do conceito. **Ciência da Informação**, Rio de Janeiro, v. 7, n. 2, p. 101-107, 1978.

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 junho 2006.

DAM, K. H.; WINIKOFF, M. Comparing Agent-Oriented Methodologies. In: International Bi-Conference Workshop on Agent-Oriented Information Systems (AOSIS), 5th, 2003, Melbourne, Australia. **Proceedings…** Germany: Lectures Notes in Computer Science, Springer, v. 3030, 2003. p. 78-93.

DAVENPORT, T. H.; LONG, D. W.; BEERS, M.C. Successful knowledge management projects. **Sloan Management Review,** v. 39, n. 2, p. 43–57, 1998.

DAVENPORT, Thomas. Pense Fora do Quadrado. Rio de Janeiro: Campus, 2005. 224 p.

DEITEL, H. M.; DEITEL, P. J. **JAVA como programar**. 4. ed. Porto Alegre: Bookman, 2003. 1386 p.

DELOACH, S. A. 2001. Specifying agent behavior as concurrent tasks. In: International Conference on Autonomous Agents, 5th, 2001, Montreal, Quebec, Canada. **Proceedings…** New York: AGENTS'01, ACM Press, p. 102-103, 2001.

DELOACH, S. A.; WOOD, M. F.; SPARKMAN, C. H. Multiagent Systems Engineering. **The International Journal of Software Engineering and Knowledge Engineering,** v. 11, n. 3, p. 231—258, 2001.

DEMING, William. 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: 29 setembro 2007.

DÉRY, R.; LANDRY, M.; BANVILLE, C. Revisiting the issue of model validation in OR: an epistemological view. **European Journal of Operational Research,** v. 66, n. 2, p. 168-183, 1993.

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.

DIXON, John R.; POLI, Corrado. **Engineering Design and Design for Manufacturing: Structured Approach.** First ed. Conway, MA, USA: Field Stone Publishers, 1999. 600 p.

DOMINGUE, John; MOTTA, Enrico; CORCHO-GARCIA, Oscar. **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: 24 julho 2006.

DOW CHEMICAL. Processing. Injection Moulding. Disponível em: <http://plastics.dow.com/plastics/ap/fab/molding/injection.htm>. Acesso em: 10 setembro 2007.

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

DUPONT. Surlyn Resin. **Troubleshooting Guide and Mould Preparation Checklist**. Disponível em: < http://www2.dupont.com/Elvax/en_US/assets/downloads/elvax_molding_guide.pdf >. Acesso em: 21 outubro 2007.

EMORPHIA. **FIPA-OS – version 2002**. Disponível em: <http://www.emorphia.com/research/about.htm>. Acesso em: 12 junho 2006.

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.

ERMINE, Jean-Louis. Challenges and Approaches for Knowledge Management. In: European Conference on Principles and Practice of Knowledge Discovery in Databases, 4th, 2000, Lyon, France. **Proceedings…** France: Lecture Notes in Artificial Intelligence – PKDD, Springer, 2000, p. 5-11.

EUROPEAN COMMISION, Manufuture High-Level Group. **Report of High-Level Group: Assuring the Future of Manufacturing in Europe.** Belgium, 2004.

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 março 2006.

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

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

FEIGENBAUM, Edward. Some Challenges and Grand Challenges for Computational Intelligence. **Journal of the Association for Computing Machinery,** v. 50, n. 1, p. 32–40, 2003.

FENG, S.C. Preliminary design and manufacturing planning integration using web-based intelligent agents. **Journal of Intelligent Manufacturing**. v. 16, p.423-437, 2005.

FERBER, Jacques; GUTKNECHT, Olivier; MICHEL, Fabien. From agents to organizations: An organizational view of multi-agent systems. In: International Workshop in Agent-Oriented Software Engineering, 4th, Melbourne, Australia. **Proceedings…** Germany: AOSE-Lecture Notes in Computer Science, 2003, v. 2935. p. 214-230.

FERNÁNDEZ-LÓPEZ, M.; GÓMEZ-PÉREZ, A.; JURISTO, N. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In: Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, p. 33–40, 1997.

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.

FININ, T.; LABROU, Y.; MAYFIELD J. KQML as an agent communication language In: J. Bradshaw (Ed.), **Software Agents**, MIT Press: Cambridge, MA, USA, 1997. p. 291-316.

FLEURY, Afonso; FLEURY, Maria T. Capacitação Competitiva da Indústria de Transformação de Plástico. **Polímeros: Ciência e Tecnologia**, v. 10, n. 3, p. E4-E10, 2000.

FOLEY, K.J. From quality management to organisation excellence: further thoughts on the contemporary business. In: International Conference on Organisational Excellence, 2th, 2001, Versailles. **Proceedings...** Versailles: Multinational Alliance for the Advancement of Organisational, 2001, p.197-214.

FOLEY, K.J. From quality management to organizational excellence: don't throw the baby out with the bath water. In: International Research Conference on Organisational Excellence, 1st , 2000, USA, CO, Fort Collins Estes Park. **Proceedings...** USA: Multinational Alliance for the Advancement of Organisational, 2000.

FÖRSTER, H.; WARNECKE, G.; KLONARIS, P.; PFEIFER, T. Der Regelkreis ist noch nicht geschlossen. **Qualität und Zuverlässigkeit,** v. 41, n. 10, 1996.

FOSTER, David; JONKER, Jan. Third generation quality management: the role of stakeholders in integrating business into society. **Managerial Auditing Journal**, v. 18, n. 4, p. 323-328, 2003.

FOSTER, David; JONKER, Jan. Towards a third generation of quality management: Searching for a theoretical re-conceptualisation of contemporary organisations based on the notions of stakeholders and transactivity. **International Journal of Quality & Reliability Management**, v. 24, n. 7, p. 683-703, 2007.

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: 10 fevereiro 2006.

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: 10 fevereiro 2006.

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: 10 fevereiro 2006.

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

GAIA. **jCOLIBRI Selected Publications of Group for Artificial Intelligence Applications**. Disponível em: < http://gaia.fdi.ucm.es/grupo/projects/jcolibri/jcolibri.html>. Acesso em: 11 março 2007.

GANDON, Fabien. **Distributed Artificial Intelligence and Knowledge Management: Ontologies and multi-agent systems for a corporate semantic web.** 2002. 483 f.. PhD Thesis – School of Science and Technologies of Information and Communication, University of Nice – Sophia Antipolis, France, 2002.

GANGEMI, A.; CATENACCI, C.; CIARAMITA, M.; LEHMANN, J. Modelling Ontology Evaluation and Validation. In: European Semantic Web Conference - The Semantic Web: Research and Applications (ESWC 2006), 3rd, 2006, Budva, Montenegro. **Proceedings …** Germany: Lecture Notes in Computer Science, v. 4011, Springer 2006, p. 140-154.

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

GARCIA, Ana C. B.; VAREJÃO, F.M.; FERRAZ, I. N. Aquisição de Conhecimento. In: REZENDE, S. O. (Org.) **Sistemas Inteligentes: fundamentos e aplicações**. São Paulo: Editora Manole, 2005. p. 51-87.

GARDINER, Tom; TSARKOV, Dmitry; HORROCKS, Ian. Framework for an Automated Comparison of Description Logic Reasoners. In: International Semantic Web Conference, 5th , 2006, Athens, GA, USA. **Proceedings…** Athens: International Semantic Web Conference Lecture Notes in Computer Science, Springer, v. 4273, p. 654-667.

GARITA, C.; AFSARMANESH, H.; HERTZBERGER, L. O. The PRODNET cooperative information management for industrial virtual enterpreises. **Journal of Intelligent Manufacturing,** v. 12, n. 2, p. 151-170, 2001.

GASS, S. I. Model accreditation: A rationale and process for determining a numerical rating. **European Journal of Operational Research,** v. 66, n. 2, p. 250-257, 1993.

GASSER, L. Perspectives on organizations in multi-agent systems. In: J. G. CARBONELL and J. SIEKMANN (Eds), **Mutli-Agents Systems and Applications**. New York: Springer Verlag, 2001, p. 1-16.

GASSER, L.; HULHAGE, I.; LEVERICH, B.; LIEB, J.; MAJCHRZAK, A. Organization as complex dynamic design problems. In: Portuguese Conference on Artificial Intelligence, EPIA'93, 6th, Porto, Portugal. **Proceedings…** Germany: Lecture Notes in Artificial Intelligence: Progress in Artificial Intelligence, Springer-Verlag, v. 727, 1993. p. 1-12.

GE PLASTICS SOUTH AMERICA. Pesquisa Técnica. **Guia de Problemas e Soluções**. Disponível em: < http://www.geplastics.com.br/resins/techsolution/troubleshooting.html >. Acesso em: 10 agosto 2007.

GOODSHIP, Vannessa. Rapra Review Report. **Report 172: Troubleshooting in Injection Moulding.** v. 15, n. 4, 2004. 50 p.

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. Disponivel em: <http://ksi.cpsc.ucalgary.ca/KAW/>. Acesso em: 37 março 2006.

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

GRÜNINGER, M.; FOX, M. S. Methodology for the Design and Evaluation of Ontologies. In: International Joint Conference on Artificial Intelligence, 14., 1995, Montreal, Quebec, Canada. **IJCAI Proceedings 1995**. Morgan Kaufmann, 1995. p. 6.1–6.10.

GUARINO, Nicola. Formal Ontology in Information systems. In: International Conference on Formal Ontology in Information Systems, 1th, 1998, Trento-Italy. **Proceedings…** Amsterdam: IOS Press, 1998, p. 3-15.

GUESSOUM, Z.; BRIOT, J.-P. From Active Object to Autonomous Agents. **IEEE Concurrency,** v. 7, n. 3, p. 68-78, 1999.

GUNENDRAN, Anantharajah; 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.

GUPTA, Jatinder N. D.; SHARMA, Sushil K. Intelligent Enterprise of the 21st Century. USA: Idea Group, 2003. 438 p.

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.

HARADA, Júlio. Moldes para injeção de termoplásticos: projetos e princípios básicos. São Paulo: ArteLiber Editora, 2004. 308 p.

HAVILAND, Paul R. Analytical Problem Solving. In: Annual Quality Congress, 2004, Toronto, Canada. **Proceedings...** ASQ – American Society for Quality, 2004, v. 58, 2004, p. 273-280.

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.

HOPKINS, J. F.; FISHWICK, P. A. A Three-dimensional human agent metaphor for modeling and simulation. **Proceedings of the IEEE**, v. 89, n. 2, 2001. p. 131-147.

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.

HORROCKS, Ian. OWL: A Description Logic Based Ontology Language. In: International Conference on Principles and Practice of Constraint Programming (CP 2005), 2005a, 11th, Sitges, Spain. **Proceedings…** Germany: Lecture Notes in Computer Science, Springer, vol. 3709, p. 5-8, 2005a.

HORROCKS, Ian. Application of Description Logics: State of Art and Research Challenges. In: 13th International Conference on Conceptual Structures (ICCS 2005), Kassel, Germany, July 17-22, 2005. **Proceedings**… Lectures Notes in Artificial Intelligence - Springer, 2005, vol. 3596, p. 78-90. 2005b.

HORROCKS, Ian. The FaCT system. In: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux'98), 1998, Oisterwijk, Netherlands. **Proceedings…** Netherlands: Lecture Notes in Artificial Intelligence, Springer-Verlag, n. 1397, 1998, p. 307–312.

HORROCKS, Ian; PATEL-SCHNEIDER, Peter F.; van HARMELEN, Frank. From SHIQ and RDF to OWL: The making of a web ontology language. **Journal of Web Semantics**, v. 1, n. 1, p. 7–26, 2003.

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 2006.

IGLESIAS, C. A.; GARIJO, M.; GONZÁLEZ, J. C.; VELASCO, J. R. Analysis and design of multiagent systems using MAS-CommonKADS. In: International Workshop on Intelligent Agents IV, Agent Theories, Architectures, and Languages, 4th, 1997. **Proceedings…** London: M. P. SINGH, A. S. RAO, and M. WOOLDRIDGE (Eds.), Lecture Notes In Computer Science, v. 1365, Springer-Verlag, 1997, p. 313-327.

IGLESIAS, C.A.; GARIJO, M. The Agent-Oriented Methodology MAS-CommonKADS. In: HENDERSON-SELLERS, B.; GIORGINI, P. Agent-oriented Methodologies. Hershey, PA: Idea Group Publishing, 2005.

INJECTION MOLDING MAGAZINE NETWORKING - IMMNET. Disponível em: <http://www.immnet.com/search>. Acesso em: 10 agosto 2007.

INTERNATIONAL ELECTROTECHNICAL COMMISSION. **IEC 60812:** Analysis techniques for system reliability: Procedure for failure mode and effect analysis (FMEA). Geneva, Switzerland, 2006. 7 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. 34 p.

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

ISHIKAWA Kaoru. **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. GEORGIEVSKI. 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 março 2006.

JAVA AGENT SERVICES. **JSR 87: JAVA SPECIFICATION REQUESTS** – Review 2002. Disponível em: <http://www.jcp.org/en/jsr/detail?id=87>. Acesso em: 15 julho 2006.

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.

JHA, S.; CHALASANI, P.; SHEHORY, O.; SYCARA, K. A formal treatment of distributed matchmaking. In: International Conference on Autonomous Agents. 2nd, 1998, Minneapolis, Minnesota, USA. **Proceedings…** New York: K. P. Sycara and M. Wooldridge (Eds.) AGENTS '98, ACM, NY, p. 457-428. 1998.

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

KEPNER, Charles Higgins; TREGOE, Benjamin B. **O novo administrador racional.** São Paulo: McGraw-Hill, 1986. 215 p.

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

KITAMURA, Y.; MIZOGUCHI, R. Ontology-based systematization of functional knowledge. **Journal of Engineering Design,** v. 15, n. 4, p. 327-351, 2004.

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

KLUSCH, M. Information agent technology for the internet: A survey. **Data and Knowledge Engineering,** v. 36, n. 3, p. 337-372, 2001.

KLUSCH, M. Information agent technology for the internet: A survey. **Data and Knowledge Engineering.** v. 36, n. 3, p. 337-372, 2001.

KNOWLEDGE BASED SYSTEMS INC. – KBSI. Integrated Definition Methods - IDEF. Disponível em: < http://www.idef.com/>. Acesso em: 23 outubro 2007.

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

KUME, Hitoshi. Métodos Estatísticos para Melhoria da Qualidade. São Paulo: Editora Gente – Aots, 1993. 241 p.

LANDRY, M., ORAL, M. In search of a valid view of model validation for operations research. **European Journal of Operational Research,** v. 66, n. 2, p. 161-167, 1993.

LANDRY, M.; MALOUIN, J.; ORAL, M. Model validation in operations research. **European Journal of Operational Research,** v. 14, p. 207-220, 1983.

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.

LARI, A. An integrated information system for quality management. **Business Process Management Journal**. v. 8, n. 2, p. 169-182, 2002.

LARROYO, Francisco. Lógica y metodología de las ciencias. México: D. F. Porrua, 1975.

LEAKE, D. B. CBR in context: The Present and Future. In : D. LEAKE (Ed) **Case-Based Reasoning Experiences: Lessons Learning & Future Directions,** 1996. Disponível em: <http://www.cs.indiana.edu/~leake/papers/p-96-01.pdf>. Acesso em: 15 julho 2004.

LEE, B. H. Using FMEA models and ontologies to build diagnostic models. **Artificial Intelligence for Engineering Design, Analysis and Manufacturing**, v. 15, p. 281-293, 2001.

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.

LENNARTSSON, M.; VANHATALO, E. **Evaluation of possible SixSigma implementation including a DMAIC project: A case study at the cage factory**, SKF Sverige AB. Master Thesis, Lulea University of Technology, 2004.

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.

LIEBIG, Thorsten. **Reasoning with OWL: System Support and Insights**. Munich, Germany: Computer Science Faculty, Ulm University, 2006. 49 p. (Technical report 2006-04).

LIEBOWITZ, J. Key ingredients to the success of an organization's knowledge management strategy. **Knowledge and Process Management**, v. 6, n. 1, p. 37–40, 1999.

LIKER, J. **The Toyota Way**. 1. ed. New York: McGraw-Hill, 2004. 330 p.

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.

LUEG, Christopher. Knowledge management and information technology: relationship and perspectives. **The European Journal for the Informatics Professional,** v. 3, n. 1, p. 2-6, 2002.

MACKENZIE, Garth R.; SCHULMEYER, G. G.; YLMAZ, Levent. Verification Technology Potential with Different Modeling and Simulation Development and Implementation Paradigms. In: Foundations for V&V in the 21st Century Workshop (Foundations '02), 2002, Laurel, Maryland, USA. **Proceedings…** USA: The Johns Hopkins University Applied Physics Laboratory, 2002. p. A2-A40.

MALEK, M.; TOITGANS, M-P.; WYBO, J-L.; VINCENT, M. An operator support system based on case-based reasoning for the plastic moulding injection process. In: European Workshop on Case–Based Reasoning, 4th, 1998, Dublin, Ireland. **Proceedings...** Germany: Lecture Notes in Computer Science, 1998, v. 1488, p. 402-413.

MANRICH, Silvio. **Processamento de Termoplásticos: rosca única, extrusão e matrizes, injeção e moldes.** ”São Paulo: Artliber Editora Ltda., 2005. 431 p.

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.

MARTINS, C. A. **Uma abordagem para pré-processamento de dados textuais em algoritmos de aprendizado**. 2003. 90 f.. Tese (Doutorado em Ciências da Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação – ICMC-USP, USP- São Carlos, Agosto, 2003.

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.

MIKA, Peter; IOSIF, Victor; SURE, York; AKKERMANS, Hans. Ontology-based Content Management in a Virtual Organization. In: **Handbook on Ontologies**. Germany, 2004. p. 455-476.

MINDSWAP GROUP. **MINDSWAP Projects - Maryland Information and Network Dynamics Lab Semantic Web Agents Project.** Disponível em: <http://www.mindswap.org/projects/>. Acesso em: 4 maio 2006.

MISER, H. J. Foundational concept of science appropriate for validation in operational research. **European Journal of Operational Research**, v. 66, n. 2, p. 204-215, 1993.

MOK, S.L.; KWONG, C.K.; LAU, W.S. An intelligent hybrid system for initial process parameter setting of injection moulding. **International Journal of Production Research,** v. 38, n. 17, p. 4565-4576, 2000.

MORAÏTIS, Pavlos; PETRAKI, Eleftheria; SPANOUDAKI, Nikolaos I. Engineering JADE Agents with the Gaia Methodology. In: Agent Technologies, Infrastructures, Tools, and Applications for E-Services: NODe 2002 Agent-Related Workshops, 2002, Erfurt, Germany. **Proceedings...** Heidelberg: Lectures Notes in Computer Science, Springer Verlag, v. 2592, 2002, p. 77-91.

MySQL. **About MySQL AB**. Disponível em: <http://www.mysql.com/company/>. Acesso em: 10 março 2006.

NARDI, D.; BRACHMAN, R. J. An Introduction to Description Logics. In: BAADER, F. (Editor). **Description Logic Handbook: Theory, Implementation and Applications**. West Nvack, New York, USA: Cambridge University Press, 2003, p. 1-40.

NATIONAL RESEARCH COUNCIL, Committee in Visionary Manufacturing Challenges, Board on Manufacturing and Engineering Design, Comission on Engineering and Technical Systems. **Visionary Manufacturing Challenges for 2020**. Washington, DC, USA: National Academy Press, 1998.

NECHES, R.; FIKES, R. E.; FININ, T.; GRUBER, T.R.; SENATOR, T.; SWARRTOUT, W. R. Enabling technology for knowledge sharing. **AI Magazine**, v. 12, n. 3, p. 36-56, 1991.

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.

NONAKA, Ikujiro; TAKEUCHI, Hirotaka. The Knowledge Creating Company. New York: Oxford University Press, 1995. 304 p.

NOVEON. Processing and Design. **Troubleshooting General Injection Moulding**. Disponível em: <http://www.tempritecpvc.com/processing-design/troubleshooting-general. asp >. Acesso em: 10 setembro 2007.

NOY, F. N.; GUINNESS, D. L. **Ontology development 101: a guide to create your first ontology.** Stanford University, USA, 2002. Disponível em: <http://ksl.stanford.edu /people/dlm/papers/ontology-tutorial-noy-mcguinness.doc>. Acesso em: 12 setembro 2006.

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 2006.

ORAL, M., KETTANI, O. The facets of the modelling and validation process in operational research. European Journal of Operational Research, v. 66, n. 2, p. 216-234, 1993.

PADGHAM, L.; WINIKOFF, M. Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents. In: Annual ACM Conference on Objected-Oriented Programming, Systems, Languages, and applications, 17th, Seatle, WA, USA. **Proceedings…** USA: ACM, Proceedings of Workshop on Agent Oriented Methodologies, 2002. p. 97-108.

PALA, Özge; VENNIX, Jac A.M.; KLEIJNEN, Jack P. C. Validation in soft or, hard or and system dynamics: a critical comparison and contribution to the debate. In: International Conference of the System Dynamics Society, 17th, 1999, Wellington, New Zealand. **Proceedings…** Wellington: Systems thinking for the millennium. 1999. Disponível em: < http://www.systemdynamics.org/conferences/1999/HOME.PDF >. Acesso em: 10 setembro 2007.

PAOLUCCI, Massimo; SACILE, Roberto. **Agent-Based Manufacturing and Control Systems**: **New Agile Manufacturing Solutions for Achieving Peak Performance.** Florida: CRC Press, 2005. 288 p.

PATEL-SCHNEIDER, Peter; HAYES, Patrick; HORROCKS, Ian. **W3C Recommendation: OWL Web Ontology Language Semantics and Abstract Syntax**. 2004. Disponível em: <http://www.w3.org/TR/owl-semantics/>. Acesso em: 15 junho 2006.

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. **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. PROTIS – Aus Fehlern lernen. **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.

PINTO, Ubiratan Schuch. **Avaliação de critério para a determinação de contratipos de termoplásticos aplicáveis em simulação da moldagem por injeção.** 2002. 123 f.. Dissertação (Mestrado em Ciência e Engenharia de Materiais) – Programa de Pós-Graduação em Ciência e Engenharia de Materiais, Universidade Federal de Santa Catarina, 2002.

POLIMOLD INDUSTRIAL. Câmara Quente. Disponível em: < http://www.polimold.com. br>. Acesso em: 10 julho 2007.

RACER SYSTEMS GMBH & Co. KG, RacerPro Server, **User's Guide Version 1.9,** 2005. Disponível em: <http://www.racer-systems.com.>. Acesso em: 12 junho 2007.

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.

RAO, A.S.; GEORGEFF, M. P. BDI Agents: From Theory to Practice. In: International Conference on Multi-Agent Systems – ICMA, 1th, 1995, San Francisco, CA. **Proceedings…** San Francisco, USA: V. LESSER (Ed.), AAAI Press, 1995. p. 312-319.

RECIO-GARCÍA, Juan A.; SÁNCHEZ, Antonio; DÍAZ-AGUDO, Maria B.; GONZÁLEZ-CALERO, 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.

REES, Herbrt. **Mold Engineering**. 2nd ed. New York: Carl Hanser Verlag, 2002. 621 p.

ROY, B. Decision science or decision-aid science?" European Journal of Operational Research, v. 66, n. 2, p. 184-203, 1993.

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência Artificial.** Tradução 2. ed. Rio de Janeiro: Elservier/Campus, 2004. 1056 p.

RYMER, John.R. The Muddle in the Middle. **Byte.com**, v. 21, n. 4, p. 67-70, April 1996.

SANCHO, Andreu. Análisis, Interpretación y Corrección de Defectos en Piezas Inyectadas con Termoplástico. In: Seminario Internacional en Inyección de Termoplásticos, 1., 2005, Joinville. **Anais …** Joinville, SC: ASCAMM - Asociación Catalana de Empresas de Moldes y Matrices - Technology Centre - SOCIESC – Sociedade Educacional de Santa Catarina, Joinville, 2005.

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.

SHAH, J., MÄNTYLA, M. **Parametric and Featured-Based CAD/CAM: concepts, techniques and application.** New York: John Wiley and Songs Inc., 1995.

SHANK, Roger. **Dynamic memory: A theory of learning in computers and people.** New York: Cambridge University Press, 1982. 234 p.

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.371-281, 2005.

SHEN, W.; NORRIE, D.H. Agent-based systems for intelligent manufacturing: a state-of-the-art survey. **Knowledge and Information Systems**. v. 2, n 1, p.129-156, 1999.

SHEN, Weiming; NORRIE, Douglas H.; BARTHÈS. Jean-Paul. **Multi-agent systems for concurrent intelligent design and manufacturing.** New York: Taylor & Francis, 2001. 383 p.

SHENOY, A. V.; SAINI, D. R. Thermoplastic Melt Rheology and Processing. **Plastic Engineering,** v. 37, p. 312-391, 1996.

SHIU, Simon C. K.; PAL, Sankar K. Case-Based Reasoning: Concepts, Features and Soft Computing. **Applied Intelligence,** v. 21, n. 3, p. 233-238, 2004.

SHOHAM, Yoav. Agent oriented programming: An overview of the framework and summary of recent research. Lectures Notes in Computer Science, v. 808, p. 123-129, 1994.

SILVA, Edna Lúcia da.; MENEZES, Estera Muszkat. **Metodologia da pesquisa e elaboração de dissertação**. Florianópolis: UFSC/PPGEP/LED, 2000, 118 p.

SIPPER, D.; BULFIN, R. Production: planning, control and integration. 1. ed. Singapore: McGraw-Hill, 1997. 630 p.

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, Nigel; CHAMBERS, Stuart; JOHNSTON, Robert. **Administração da Produção**. São Paulo: Editora Atlas, 2002. 754 p.

SMITH J. H. Modeling muddles: Validation beyond numbers. **European Journal of Operational Research,** v. 66, n. 2, p. 235-249, 1993.

SOARES, A. L.; AZEVEDO, A. L.; SOUSA, J. P. Distributed planning and control systems for the virtual enterpreise: organization requirements and development life-cycle. **Journal of Intelligent Manufacturing,** v. 11, n. 3, p. 253-270, 2000.

SOCIETY OF AUTOMOTIVE ENGINEERS INTERNATIONAL. **SAE J1739**: Potential Failure Mode and Effects Analysis. USA, PA, 2002. 60 p.

SOCIETY OF THE PLASTICS INDUSTRY. **Economic Statistics - Global business trends.** Disponível em: <http://www.socplas.org/industry/global.htm>. Acesso em: 20 setembro 2007.

SORS, L.; BARDÓCZ, L.; RADNÓTI, I. Plásticos Moldes e Matrizes. São Paulo: Editora Hemus, 2002. 498 p.

SOWA, John F. Top-layer ontological categories. **International Journal of Human-Computer Studies,** v. 43, n. 5-6, p. 669-685, 1995.

STAAB, S.; SCHNURR, H. P.; STUDER, R.; SURE, Y. Knowledge Processes and Ontologies. **IEEE Intelligent Systems,** v. 16, n. 1, p. 26–34, 2001.

STAMATIS, D. H. **Failure Mode and effect analysis: FMEA from theory to execution.** 2nd ed. Milwaukee, Wisconsin: ASQ Quality Press, 2003. 455 p.

STEIL, F. G. Injection Molds for Thermoplastics In: Geng, H. (Editor) **Manufacturing Engineering Handbook**. New York: McGraw-Hill Companies Inc., 2004. p. 40.1-40.11.

STEWART, T. A. Capital intelectual: a nova vantagem competitiva das empresas. Rio de Janeiro: Campus, 1997. 264 p.

STUDER R.; BENJAMINS, V. R.; FENSEL, D. Knowledge Engineering: Principles and Methods. **IEEE Transactions on Data and Knowledge Engineering**, v. 25, n. 1-2, p. 161-197, 1998.

SU, Kuan-Hua; LIN, Jia-Horng; LIN, Chih-Ching. Influence of reprocessing on the mechanical properties and structure of polyamide 6. **Journal of Materials Processing Technology,** v. 192–193, p. 532–538, 2007.

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.

SUN DEVELOPER NETWORK. **JAVA SE Overview**. Disponível em: <http://java.sun.com/javase/technologies/index.jsp#overview>. Acesso em: 14 março 2007.

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).

TACLA, Cesar. A.; BARTHÈS Jean-Paul. A multi-agent system for acquiring and sharing lessons learned. **Computers in Industry,** v. 52, n. 1, p. 5-16, 2003.

TAGUCHI, G.; WU, Y. **Introduction to Off-line Quality Control**. Nagoya: Japan Quality Control Organization, 1980. 111 p.

TANG, X.-Q; LU, Q.-L. Intranet/Extranet/Internet-based quality information management system in expanded enterprises. **International Journal of Advanced Manufacturing Technology,** v. 20, n. 11, p. 853-858, 2002.

TECINNO. **CBRWorks 4 - Compendium**, 1999.

TEIJIN KASEI AMERICA. Technical Help Troubleshooting. **General Purpose Guide**. Disponível em: < http://www.teijinkasei.com/general_troubleshooting.asp>. Acesso em: 10 setembro 2007.

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 2006.

TRANTINA, G. G. Design with Plastics. In: **Characterization and failure analysis of plastics**. USA: ASM International – The Materials Information Society, 2003. p. 56-63.

TSARKOV, D., HORROCKS, I. Optimised Classification for Taxonomic Knowledge Bases. In: International Workshop on Description Logics (DL2005), 2005, Edinburgh, Scotland, UK. **Proceedings…** Germany: Ian Horrocks, Ulrike Sattler, Frank Wolter (Eds.), CEUR Workshop Proceedings, v. 147, 2005.

TSARKOV, Dmitry; HORROCKS, Ian. FaCT++ description logic reasoner: System description. In: International Joint Conference on Automated Reasoning (IJCAR 2006), 3rd, 2006, Seattle, USA. **Proceedings…** Seattle: Lecture Notes in Artificial Intelligence, Springer, v. 4130, 2006. p. 292-297.

TSUNG, F. Impact of information sharing on statistical quality control. **IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans.** v. 30, n. 2, p. 211-216, 2000.

TUMER, I. Y., STONE, R. B., BELL, D. G. Requirements for a failure mode taxonomy for use in conceptual design. In: **International Conference on Engineering Design**, Stockholm, Sweden, August 2003. Paper No. 1612, 2003.

UMG ABS LTD. Home Solutions. **Troubleshooting**. Disponível em: <http://www. umgabs.co.jp/en/solution/trouble/index.htm >. Acesso em: 10 setembro 2007.

USCHOLD, M.; KING, M. Towards a Methodology for Building Ontologies. In: International Joint Conference on Artificial Intelligence. 1995, Montréal, Québec, Canada. **Proceedings…** IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing. Springer, 1995.

USCHOLD, Mike; JASPER, Robert. A Framework for Understanding and Classifying Ontology Applications. In: Workshop on Ontologies and Problem-Solving Methods (KRR5): Lessons Learned and Future Trends, 1999, Stockholm, Sweden. **Proceedings...** Amsterdam: CEUR-WS, 1999, v. 18, p. 11.1-11.12.

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 ENGERS, Tom Maarten. **Knowledge Management: The role of mental models in business systems design.** 2001. 227 f.. PhD Thesis – Department of Computer Science, Faculty Mathematics an Information Sciences, VU University Amsterdam, Amsterdam, 2001.

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 2006.

WATERMAN, Donald A. A guide to expert systems. Massachusetts: Addison-Wesley Publishing Company, 1986. 419 p.

WATSON, I. **Applying Case-Based Reasoning: Techniques for Enterpreise Systems**. San Francisco, CA: Morgan Kaufmann Publishers Inc., 1997. 289 p.

WATSON, Ian. **Applyig Knowledge Management: Techniques for building Organizational Memories.** San Francisco, CA: Morgan Kaufmann Publishers Inc., 2003. 252 p.

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.

WIENDAHL, H. -P.; LUTZ, S. Production in Networks. **CIRP Annals – Manufacturing Technology,** v. 51, n. 2, p. 573-586, 2002.

WIIG, K. Knowledge management: where did it come from and where will it go? **Expert Systems with Applications**, v. 13, n. 1, p. 1–14, 1997

WILKE, W.; BERGMANN, R. Techniques and Knowledge used for Adaptation during CBR Problem Solving. In: International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert System, 11th, 1998, Castellon, Espanha. **Proceedings …** disponível em: <http://citeseer.ist.psu.edu>. Acesso em: 15 julho 2004.

WINDRUM, Paul; FAGIOLO, Giorgio; MONETA, Alessio. Empirical Validation of Agent-Based Models: Alternatives and Prospects. **Journal of Artificial Societies and Social Simulation**, vol. 10, no. 2, 2007. Disponível em: <http://jasss.soc.surrey.ac.uk/10/2/8.html>. Acesso em: 20 setembro 2007.

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. A methodology for agent-oriented analysis and design. In: International Conference on Autonomous Agents, 3th, 1999, Seattle, WA, USA. **Proceedings…** Autonomous Agents Online Proceedings, p. 69-76. 1999. Disponível em: <http://sigart.acm.org/proceedings/agents97/>. Acesso em: 12 setembro 2007.

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, Michael. **Introduction to Multiagent Systems**. London: John Wiley & Sons Ltd, 2002. 256 p.

WOOLDRIDGE, Michael; JENNINGS, Nicholas R. Intelligent agents: Theory and Practice. **The Knowledge Engineering Review,** v. 10, n. 2, p. 115-152, 1995.

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.

ZAIDAT, A.; BOUCHER, X.; VINCENT, L. A framework for organization network engineering and Integration. **Robotics and Computer-Integrated Manufacturing,** v. 21, n. 3, p. 259-271, 2005.

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.

ZAMBONELLI, F.; PARUNAK, H. Signs of a revolution in computer science and software engineering. In: **P. Petta, R. Tolksdorf and F. Zambonelli (eds.), Engineering Societies in the Agent World (ESAW),** 3th International Workshop, 2002, Madrid, Spain. **Proceedings...** Berlin, Germany: Lectures Notes in Computer Science - Springer, v. 2577, 2003, p. 13-28.

ZÚÑIGA, Gloria. 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.

APÊNDICE 1
MODELOS INTERAÇÃO

Neste apêndice são apresentados os modelos de interação correspondentes ao capítulo 4.

PROPOSITO: *DefinirProcessoDeManufatura*		
INICIADOR: USUÁRIO	**RESPONDEDOR:** ASSISTENTE	**Entrada:**
PROCESSAMENTO: Selecionar o processo de manufatura desejado entre os processos modelados disponíveis no modelo.		**Saída:** *PerfilDoProcessoDeManufatura*

Figura 1 – Modelo de interações Usuário/Assistente – *DefinirProcessoDeManufatura.*

PROPOSITO: *DefinirPerspectivaDaConsulta*		
INICIADOR: USUÁRIO	**RESPONDEDOR:** ASSISTENTE	**Entrada:**
PROCESSAMENTO: Selecionar a perspectiva da consulta entre casos mais similares ou análise de modos de falha.		**Saída:** *PerspectivaDaConsulta*

Figura 2 – Modelo de interações Usuário/Assistente – *DefinirPerspectivaDaConsulta.*

PROPOSITO: *ConfigurarConsultaDeAcordoComEstruturaDoCaso*		
INICIADOR: USUÁRIO	**RESPONDEDOR:** ASSISTENTE	**Entrada:**
PROCESSAMENTO: Configurar uma consulta aos raciocinadores com base em atributos que descrevem a não-conformidade.		**Saída:** *ConfiguraçãoDaConsultaDeAcordo: ComEstruturaDoCaso; ComConceitosOntológicos.*

Figura 3 – Modelo de interações Usuário/Assistente – *ConfigurarConsultaDeAcordoCom_ EstruturaDoCaso.*

PROPOSITO: *GuiarAConfiguraçãoDaConsulta*		
INICIADOR: ASSISTENTE	**RESPONDEDOR:** USUÁRIO	**Entrada:** *PerfilDoProcessoDeManufatura* *PerspectivaDaConsulta*
PROCESSAMENTO: Guiar o usuário na definição dos atributos da não-conformidade mediante interfaces gráficas apropriadas.		**Saída:** *ConfiguraçãoDaConsultaDeAcordo:* *ComEstruturaDoCaso;* *ComConceitosOntológicos.*

Figura 4 – Modelo de interações Usuário/Assistente – *GuiarAConfiguraçãoDaConsulta.*

PROPOSITO: *ObterAConsultaConfigurada*		
INICIADOR: ASSISTENTE	**RESPONDEDOR:** USUÁRIO	**Entrada:** *ConfiguraçãoDaConsultaDeAcordo:* *ComEstruturaDoCaso;* *ComConceitosOntológicos.*
PROCESSAMENTO: Obter a consulta configurada pelo usuário.		**Saída:** *ConsultaConfigurada*

Figura 5 – Modelo de interações Assistente/Usuário – *ObterAConsultaConfigurada.*

PROPOSITO: *EncapsularAConsultaConvertidaNoConteúdoDaMensagem*		
INICIADOR: ASSISTENTE	**RESPONDEDOR:** USUÁRIO	**Entrada:** *ConsultaConvertida*
PROCESSAMENTO: Converter a consulta para a sintaxe apropriada e encapsular como conteúdo de uma mensagem.		Saída: *MensagemComConsulta_* *Encapsulda*

Figura 6 – Modelo de interações Assistente/Usuário – *EncapsularAConsultaConvertida_ NoConteúdoDaMensagem.*

PROPOSITO: *AguardarRespostaDaConsulta*		
INICIADOR: ASSISTENTE	**RESPONDEDOR:** RACIOCINADORES	**Entrada:** *MensagemComResposta_ Encapsulda*
PROCESSAMENTO: Configurar uma consulta aos raciocinadores com base em atributos que descrevem a não-conformidade.		Saída:

Figura 7 – Modelo de interações Assistente/Raciocinadores – *AguardarResposta_ DaConsulta.*

PROPOSITO: *ApresentarRespostaConvertidaAoUsuário*		
INICIADOR: ASSISTENTE	**RESPONDEDOR:** USUÁRIO	**Entrada:**
PROCESSAMENTO: Apresentar a resposta convertida ao usuário.		**Saída:** *RespostaConvertida*

Figura 8 – Modelo de interações Assistente/Usuário – *ApresentarResposta_ ConvertidaAoUsuário.*

PROPOSITO: *AguardarContinuamenteMensagensDoAssistente*		
INICIADOR: RACIOCINADOR	**RESPONDEDOR:** ASSISTENTE	**Entrada:** *MensagemComConsulta_ EncapsuldaEnviada*
PROCESSAMENTO: Aguardar continuamente por mensagens do assistente.		**Saída:**

Figura 9 – Modelo de interações Raciocinadores/Assistente – *AguardarContinuamente_ MensagensDoAssistente.*

PROPOSITO: *ObterAConsultaEncapsuladaNoConteúdoDaMensagem*		
INICIADOR: RACIOCINADOR	**RESPONDEDOR:** ASSISTENTE	**Entrada:** *MensagemComConsulta_ Encapsulda*
PROCESSAMENTO: Obter a consulta e converte-las para um formato adequado aos métodos de raciocínio.		**Saída:** *ConsultaConvertida*

Figura 10 – Modelo de interações Raciocinadores/Assistente – *ObterAConsultaEncapsulada_ NoConteúdoDaMensagem.*

PROPOSITO: *EncapsularARespostaConvertidaNoConteúdoDaMensagem*		
INICIADOR: RACIOCINADOR	**RESPONDEDOR:** ASSISTENTE	**Entrada:** *RespostaDoRaciocínioParaASin taxeInicial*
PROCESSAMENTO: Encapsular as respostas dos métodos de raciocínio de acordo com a sintaxe inicial.		**Saída:** *MensagemComResposta_ Encapsulda*

Figura 11 – Modelo de interações Raciocinadores/Assistente – *EncapsularAResposta_ ConvertidaNoConteúdoDaMensagem.*

PROPOSITO: *EnviarAMensagemAoAssistente*		
INICIADOR: RACIOCINADOR	**RESPONDEDOR:** ASSISTENTE	**Entrada:** *MensagemComResposta_ Encapsulda*
PROCESSAMENTO: Enviar a mensagem com a resposta dos métodos de raciocínio convertidas e encapsuladas na mensagem.		**Saída:** *MensagemComResposta_ EncapsuldaeEnviada*

Figura 12 – Modelo de interações Raciocinadores/Assistente – *EnviarAMensagem_ AoAssistente.*

PROPOSITO: *AguardarSolicitaçãoDeIdentificãoDeRaciocinadores*		
INICIADOR: IDENTITICADOR	**RESPONDEDOR:** ASSISTENTE	**Entrada:** *SolicitaçãoDeLocalizaçãoE_CapacidadesDosRaciocinadores_Ativos*
PROCESSAMENTO: Aguardar as solicitações de identificação de raciocinadores ativos e capazes de responder uma consulta desejada.		**Saída:**

Figura 13 – Modelo de interações Raciocinadores/Assistente – *AguardarSolicitaçãoDe_IdentificãoDeRaciocinadores.*

APÊNDICE 2
IMPLEMENTAÇAO DA ONTOLOGIA PFMEA

Neste apêndice são apresentadas as definições de classes da ontologia PFMEA preparadas no editor PROTÉGÉ OWL.

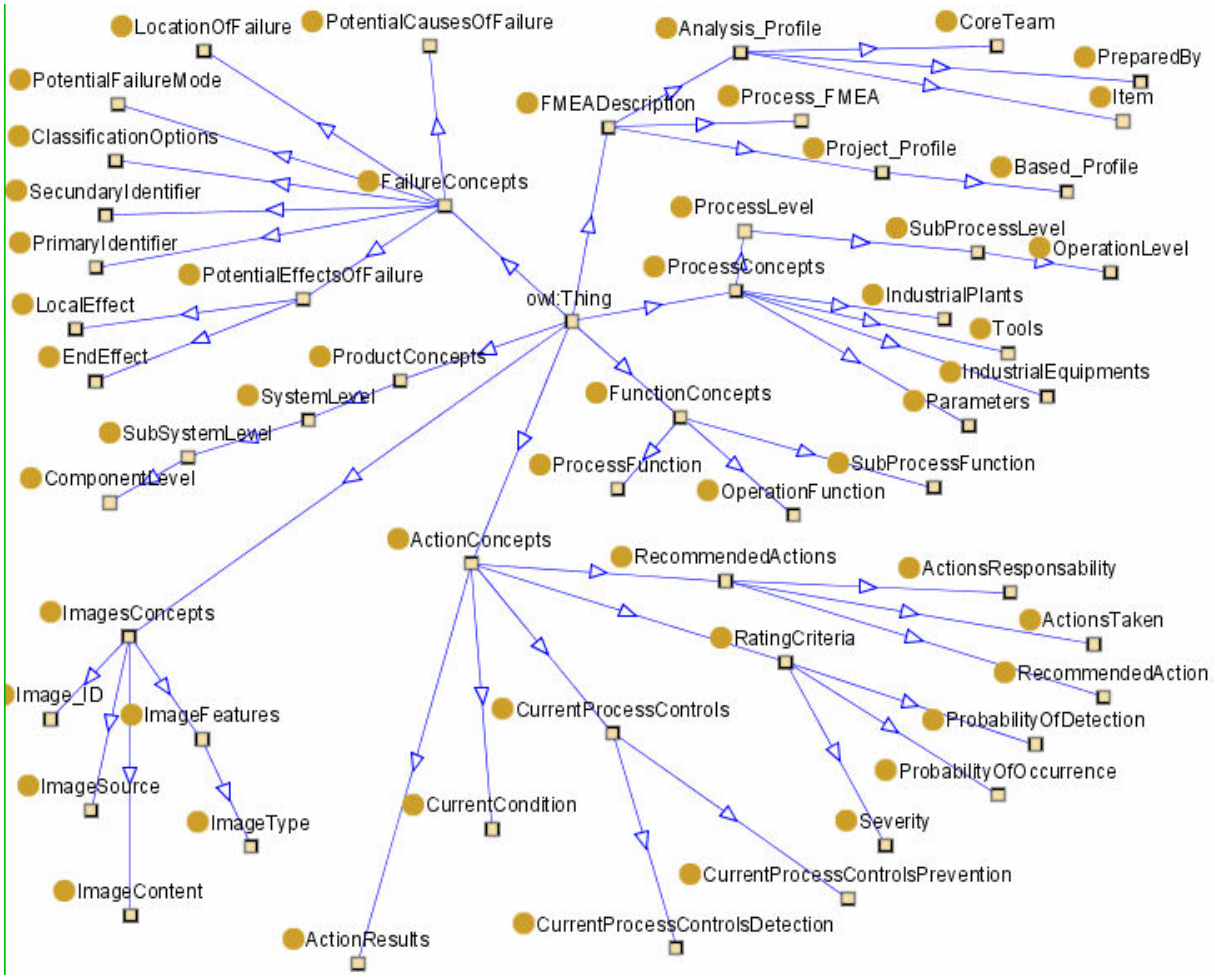


Figura 1 – Representação gráfica da Taxonomia de classes OWL-DL da ontologia PFMEA.

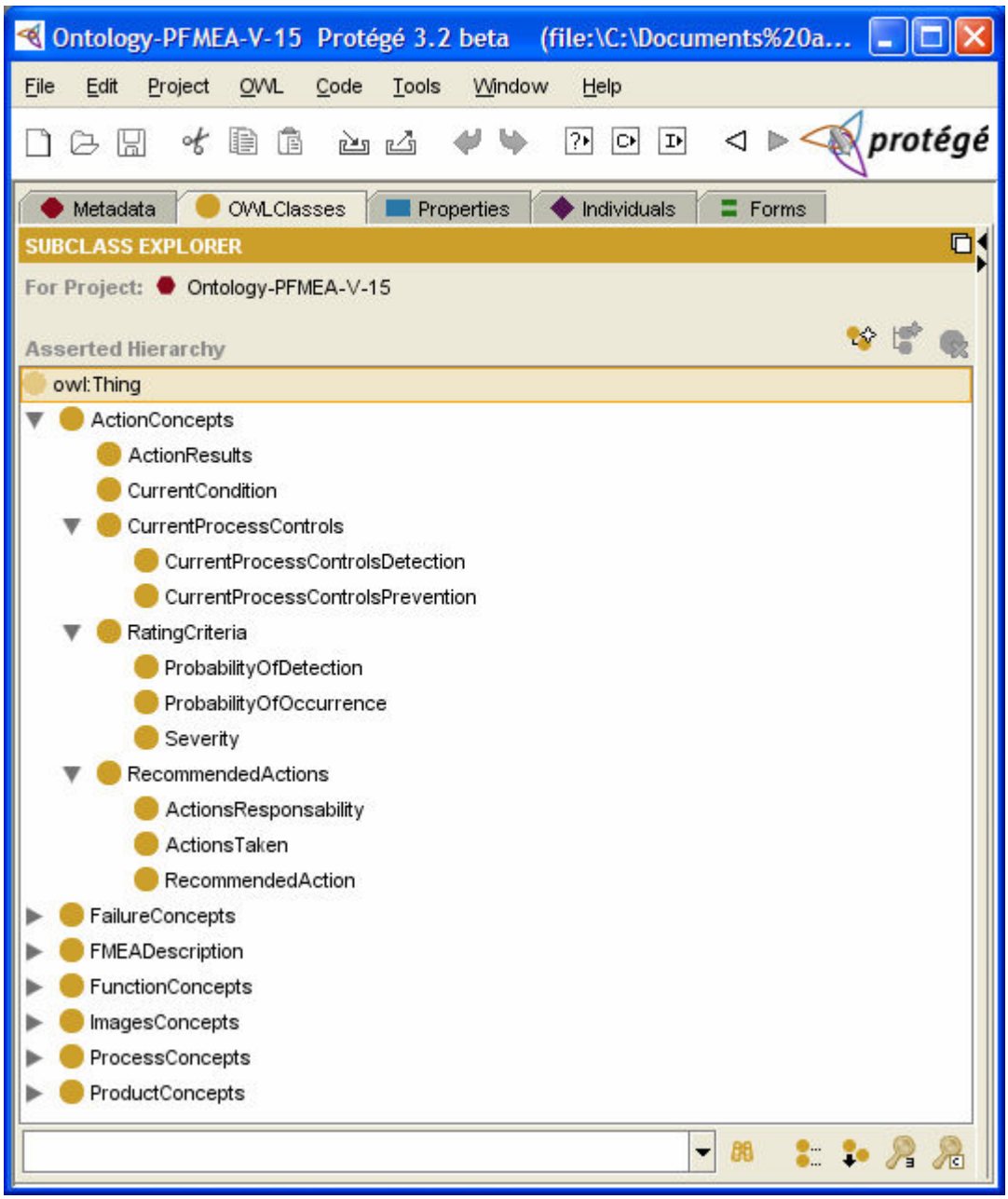


Figura 2 – Definições de classes OWL-DL para a ontologia PFMEA – eixo de *ActionConcepts*.

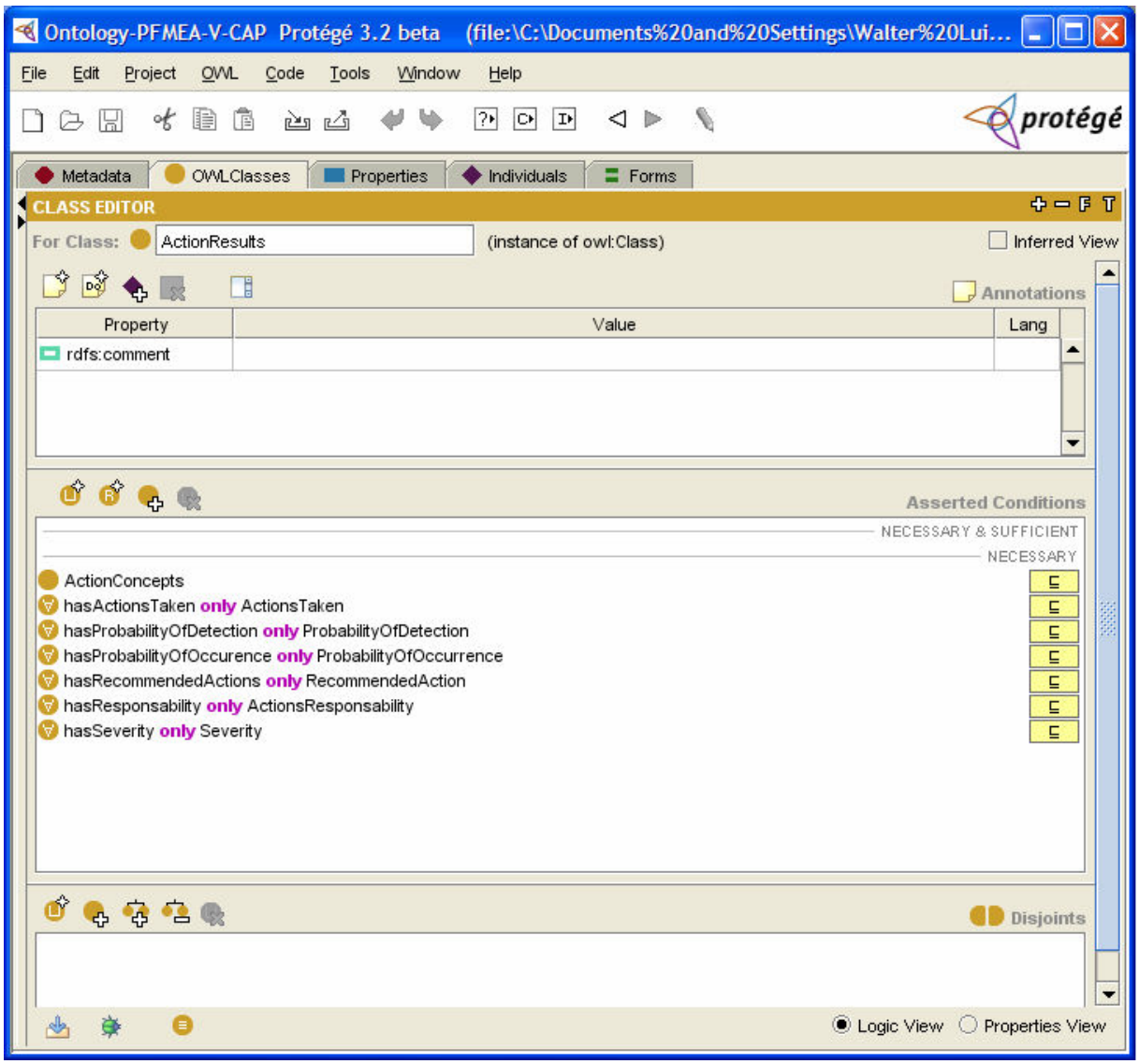


Figura 3 – Aplicação das restrições em propriedades da linguagem OWL-DL para descrever a subclasse OWL-DL "*ActionResults*".

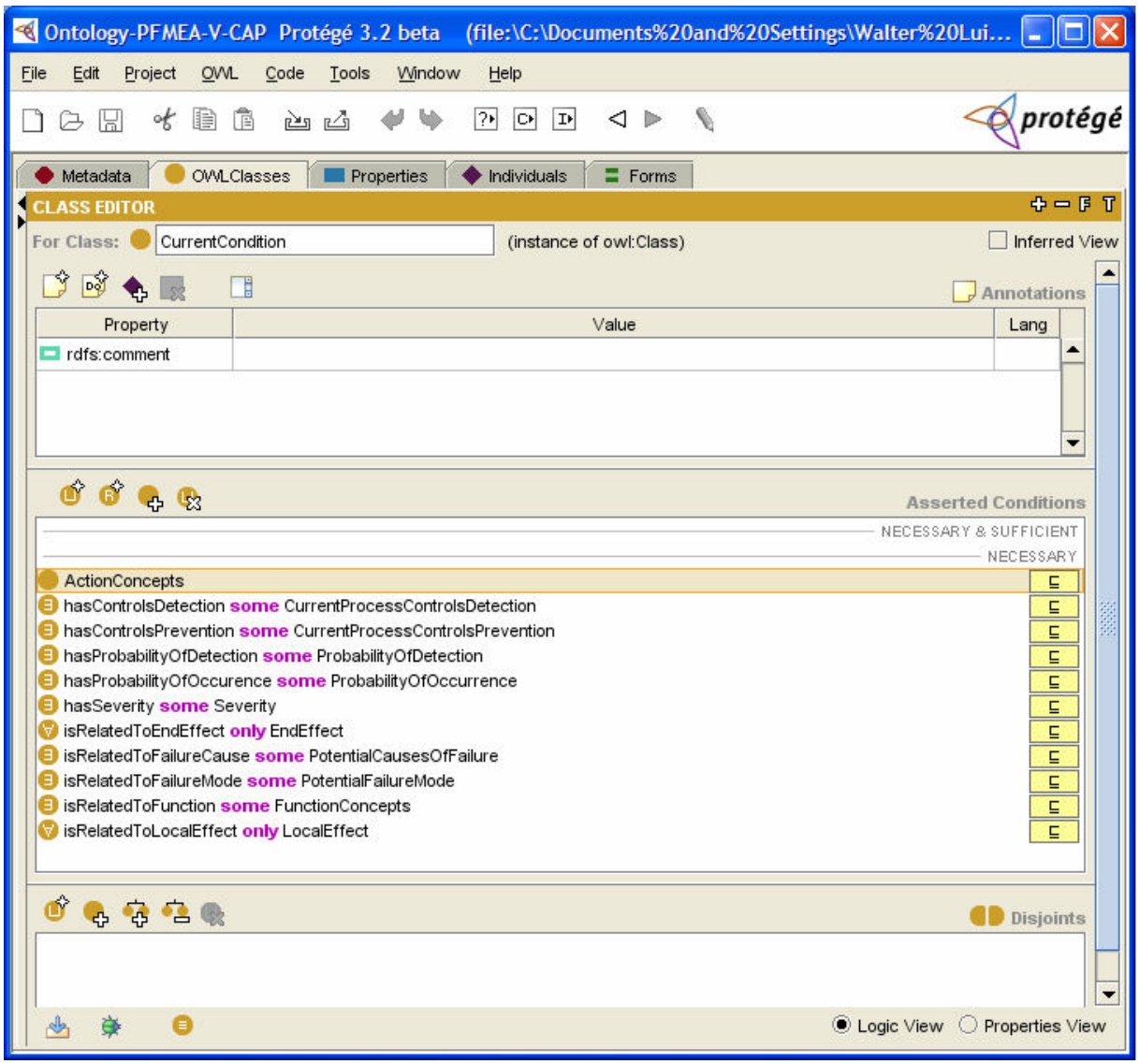


Figura 4 – Aplicação das restrições em propriedades da linguagem OWL-DL para descrever a subclasse OWL-DL "*CurrentCondition*".

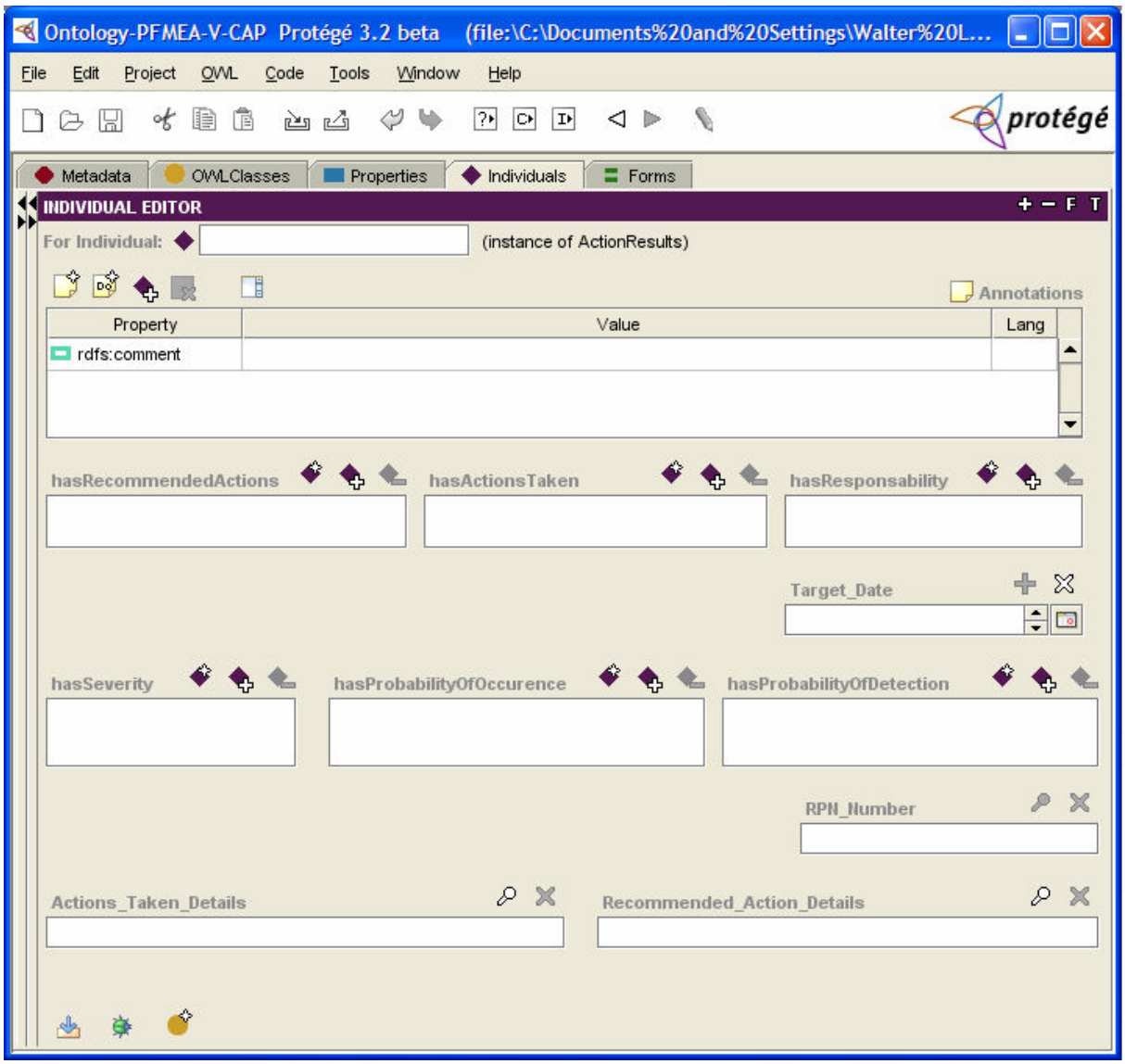
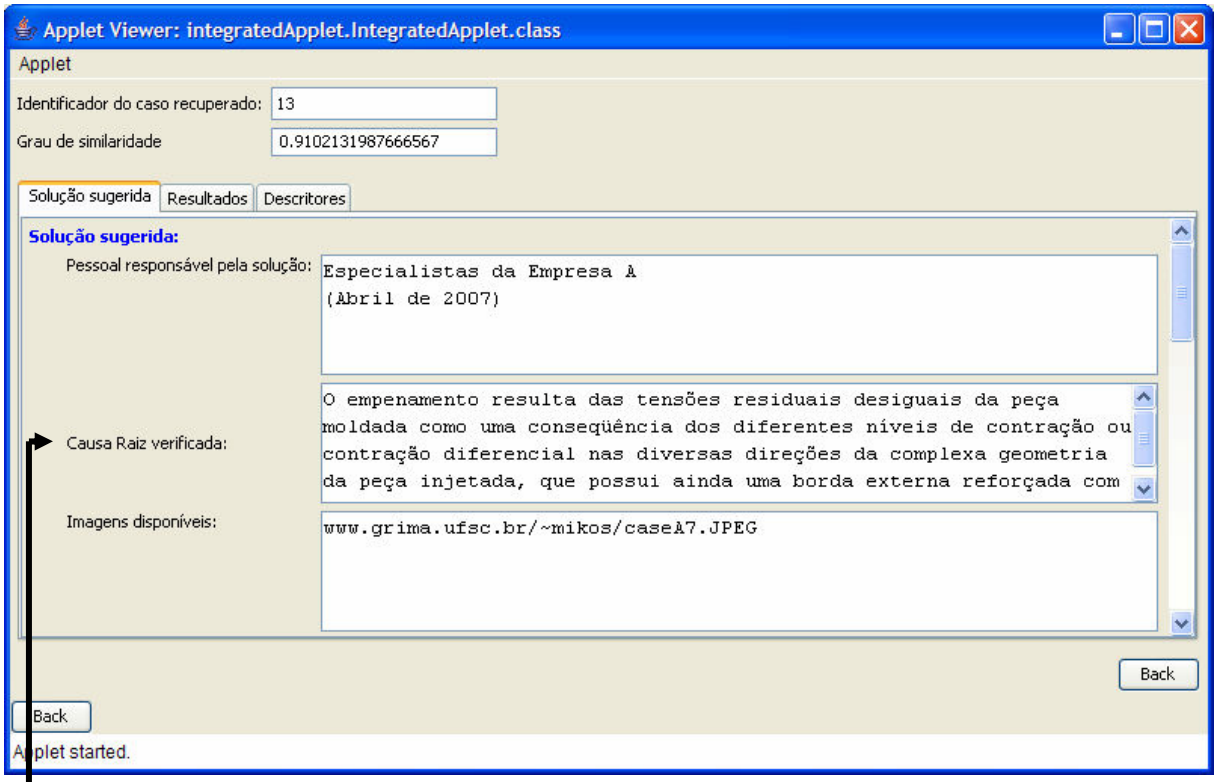


Figura 5 – Janela gráfica configurada no editor PROTÉGÉ OWL para a especificação de indivíduos da subclasse "*ActionResults*".

APÊNDICE 3

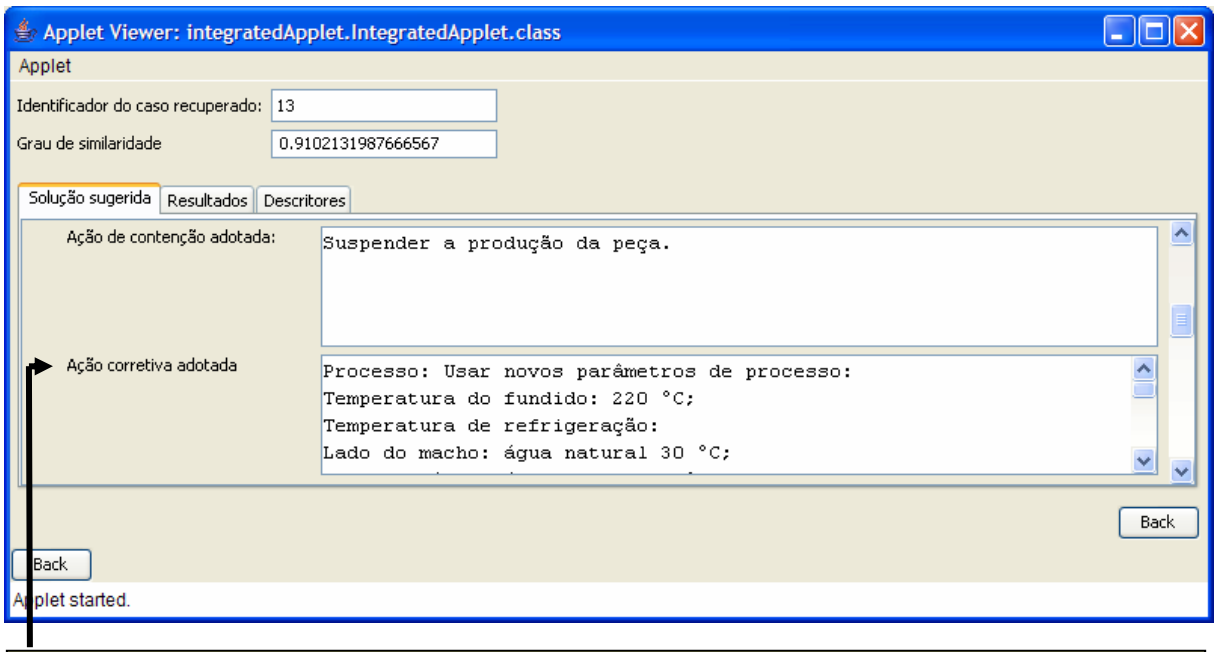
JANELAS GRÁFICAS DO AGENTE DE INTERFACE E EXEMPLO DE CASO

Neste apêndice são apresentadas as janelas gráficas do agente de interface RBC dos estudos de casos correspondentes ao capítulo 8.



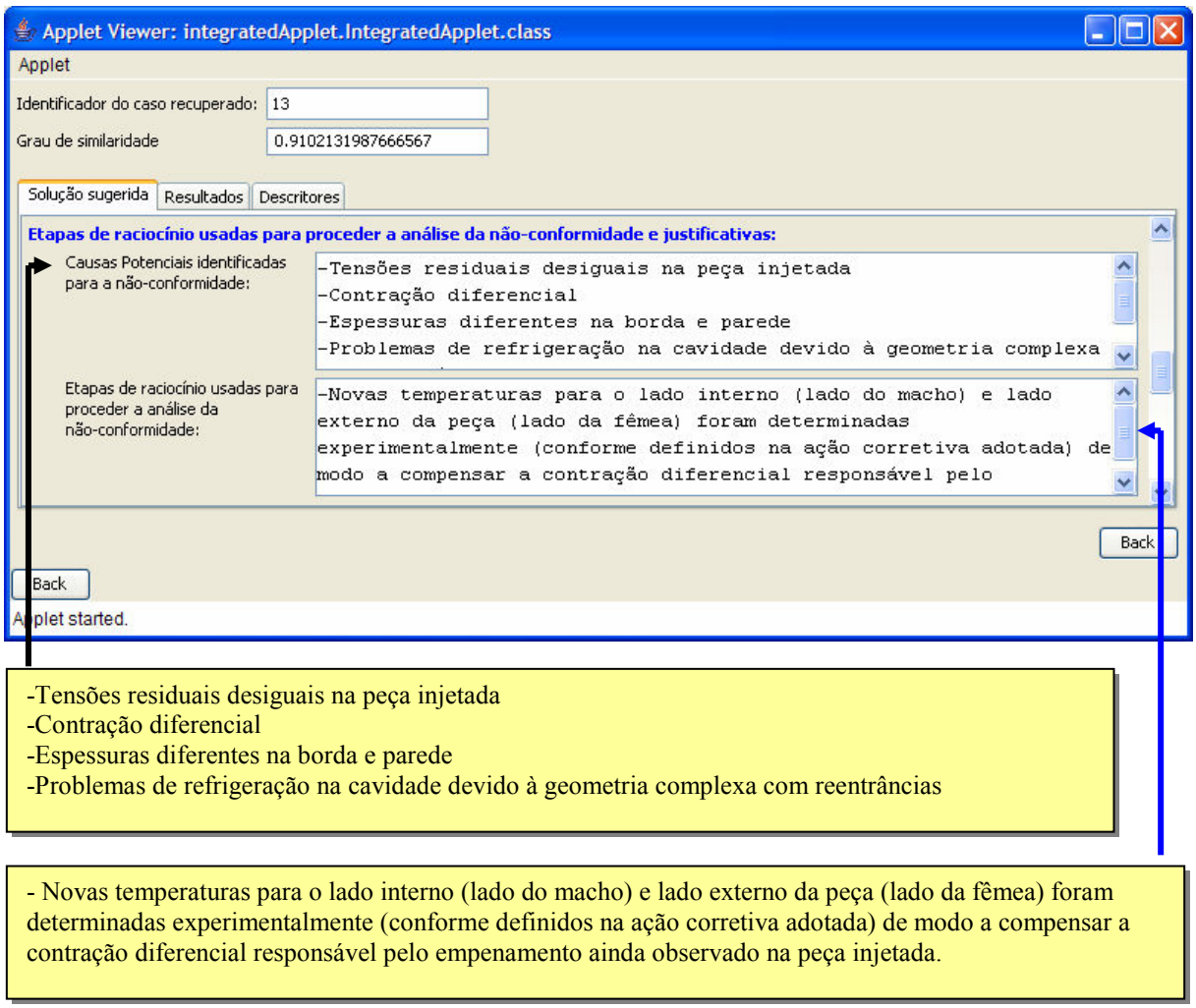
O empenamento resulta das tensões residuais desiguais da peça moldada como uma conseqüência dos diferentes níveis de contração ou contração diferencial nas diversas direções da complexa geometria da peça injetada, que possui ainda uma borda externa reforçada com 3,0 mm de espessura e paredes com espessura nominal de 1,8 mm.

Figura 1 – Janela gráfica do Agente de interface RBC – Descritores da solução sugerida – Parte I.



Processo: Usar novos parâmetros de processo:
Temperatura do fundido: 220 °C;
Temperatura de refrigeração:
Lado do macho: água natural 30 °C;
Lado da fêmea: água quente 60 °C.
Curso de dosagem: 200mm; descompressão: 4mm.

Figura 2 – Janela gráfica do Agente de interface RBC – Descritores da solução sugerida – Parte II.



-Tensões residuais desiguais na peça injetada
-Contração diferencial
-Espessuras diferentes na borda e parede
-Problemas de refrigeração na cavidade devido à geometria complexa com reentrâncias

- Novas temperaturas para o lado interno (lado do macho) e lado externo da peça (lado da fêmea) foram determinadas experimentalmente (conforme definidos na ação corretiva adotada) de modo a compensar a contração diferencial responsável pelo empenamento ainda observado na peça injetada.

Figura 3 – Janela gráfica do Agente de interface RBC – Descritores da solução sugerida – Parte III.

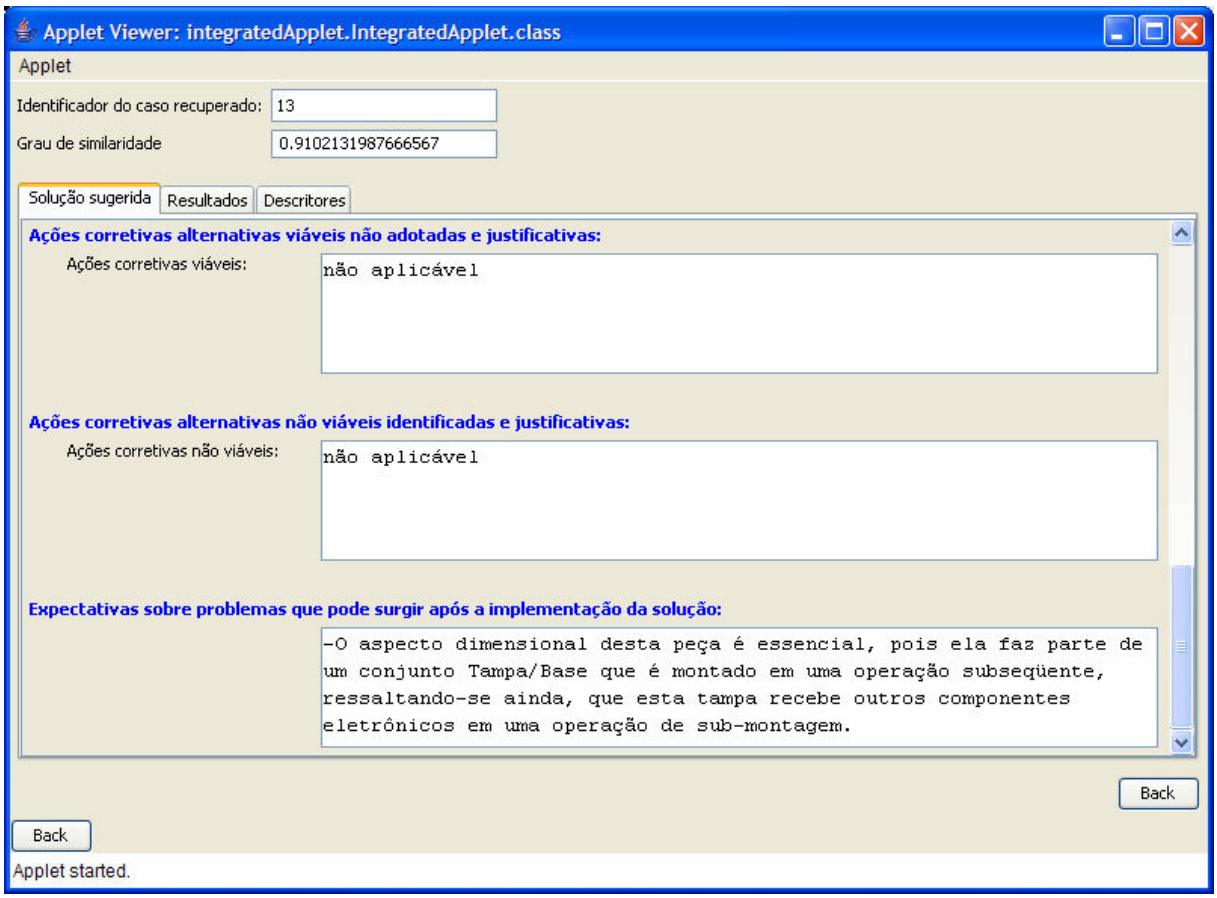
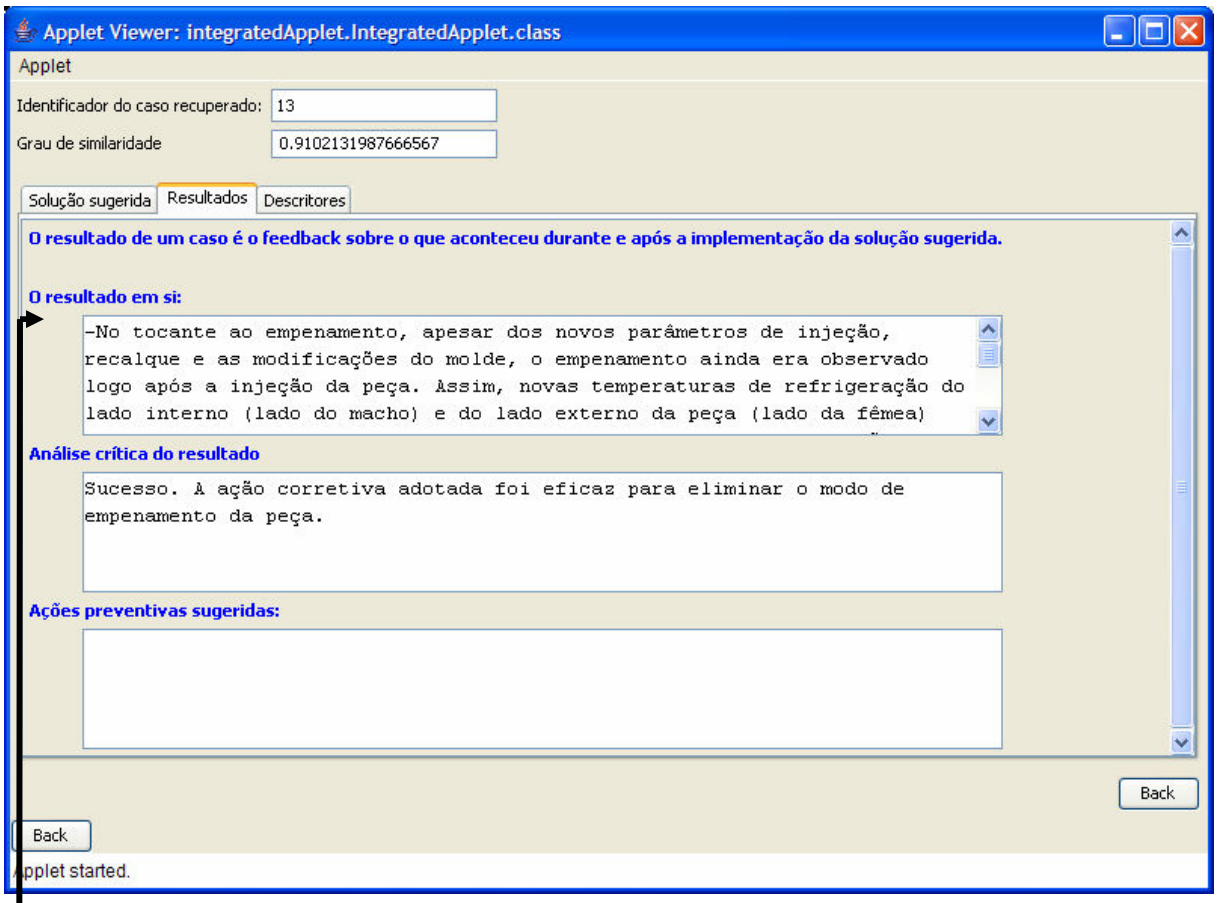


Figura 4 – Janela gráfica do Agente de interface RBC – Descritores da solução sugerida – Parte IV.



-No tocante ao empenamento, apesar dos novos parâmetros de injeção, recalque e as modificações do molde, o empenamento ainda era observado logo após a injeção da peça. Assim, novas temperaturas de refrigeração do lado interno (lado do macho) e do lado externo da peça (lado da fêmea) foram estabelecidas de modo experimental (conforme definido na ação corretiva adotada).
Deste modo esta diferença de temperaturas de refrigeração compensa a contração diferencial, pois, a parte externa da peça passou a resfriar mais lentamente compensando o encurvamento da tampa.

Figura 5 – Janela gráfica do Agente de interface RBC – Resultados.

ANEXO 1

Potential Failure Mode and Effects Analysis (Process FMEA)

System | Process: | PFMEA Number : 01-2004

Subsystem | Medical device manufacturing – Injection molding | Page: 1/5 - 2/5

Component : Nonintrathecal device
Dry Medical device | Process Responsibility Joyce Hansen | Prepared By : Peter S. Lee

Model Year(s) | Key Date : January 2004 | PFMEA Date: January 2004

Core Team : Peter S. Lee, Bryan Plumlee, Terri Rymer, Robert Schwabe, and Joyce Hansen | PFMEA (Rev) Date

Item	Potential Failure Mode	Potential Effect(s) of Failure	Sev	Potential Cause(s) Mechanisms of Failure	Occur	Current Process Controls Prevention	Detec	RPN	Recommended Action(s)	Responsibility and Target Completion Date	Action Results				
Process Function/ Requirements											Actions Taken	Sev	Occ	det	RPN
- Transfer of molded component to inspections sites	Bacterial endotoxin cross-contamination of molded components during transfer to inspection sites.	Bacterial endotoxin cross-contamination onto molded components	3	Bacterial endotoxin transferred from equipmen surfaces	1	Equipment cleaned and dried at start-up and as needed, a mimimum of once per shift per local standard operating procedure (SOPS) for general cleaning for controlled environment areas and for injection molding procedure. Routine testing of environment surfaces and air per local SOPs for microbiological air testing.	3	9	None needed						
Inspection sites	Manual handling of molded components	Bacterial endotoxin cross-contamination onto molded components	3	Improper personnel cleanliness (wet hands0	1	Cleaning of inspection station at start-up and a minimum of once per shift per local standard operating procedure (SOPS) for injection molding procedure. Handwahing, growning, and entry requirement per local SPOs for dress code procedure. Routine testing of environment surfaces and air per local SOPs for microbiological sampling procedure and microbiological air testing.	3	9	None needed						

Transfer and storage	Bacterial endotoxin cross-contamination on the outside of packing from plant environment	Bacterial endotoxin cross-contamination onto molded components	1	Water from plant environment	1	Use of secondary barrier cardboard boxes. Molded components are placed in polybags per local SOPs on injection molding procedure and appropriate material specifications. Proper transfer and storage procedure per loca SPOs on disposable set of component boxes and bags for water or water damage and particulate matter to local SOPs on disposable set manufacturing procedure.	2	2	None needed						

Fonte: LEE, Peter S.; PLUMLEE, Bryan; RYMER, Terri; SCHWABE, Robert; HANSEN, Joyce. Using FMEA to Devolp Alternatives to Batch Testing In: MDDI Medical Device and Diagnostic Industry. Disponível em: < http://www.devicelink.com/mddi/archive/04/01/018.html>. Acesso em: 20 setembro 2007.

Análise do Modo de Falha e Efeitos (FMEA de processo)

Sistema	Processo:	PFMEA Número : 01-2007
Subsistema	Moldagem por injeção de termoplásticos (High Impact PS)	Página: 1/5 - 2/5
Componente : Tampa	Responsabilidade : Especialista A	Preparado por :
Molde: TP01	Data : 2007	PFMEA Data: Junho 2007
Time : Especialistas empresa “A”.		PFMEA (Rev) Data

Item	Modo Falha Potencial	Efeito Potencial da Falha	Sev	Causa(s) ou Mecanismos Potenciais da Falha	Occur	Controles Atuais do Processo	Detec	RPN	Ações recomendadas	Responsabi-lidades e prazos	Resultados das Ações				
Função do Processo / Requisitos											Ações tomadas	Sev	Occ	det	RPN
- Fase de Recalque - ocorre assim que a cavidade acaba de ser preenchida e pressurizada	Empenamento geral	Problema de montagem - no acoplamento da tampa com a base do equipamento.	9	Tensões residuais desiguais na peça moldada. Decorrente da menor contração na região do ponto de injeção (centralizado no lado externo) causado pelo maior empacotamento desta região e uma maior contração nas regiões afastadas do ponto de injeção, em particular nas bordas reforçadas.	5	Inspeção durante o start-up. Inspeção visual a cada 10 peças.	5	225	Usar os perfis de alta velocidade de injeção. Adotar um controle mais rigoroso do início do empacotamento por meio ponto de comutação e usar um perfil de recalque adequado à geometria do componente.	Imediata	Alteração de parâmetros de recalque	9	1	5	45