Arnoldo Uber Junior

FRAMEWORK PARA ESCALONAMENTO DISTRIBUÍDO DE PROCESSOS UTILIZANDO SISTEMA MULTIAGENTES EM SISTEMAS DE PRODUÇÃO

Florianópolis – SC

2009

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Arnoldo Uber Junior

FRAMEWORK PARA ESCALONAMENTO DISTRIBUÍDO DE PROCESSOS UTILIZANDO SISTEMA MULTIAGENTES EM SISTEMAS DE PRODUÇÃO

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Ricardo Azambuja Silveira, Dr.

Florianópolis (SC), Agosto de 2009.

FRAMEWORK PARA ESCALONAMENTO DISTRIBUÍDO DE PROCESSOS UTILIZANDO SISTEMA MULTIAGENTES EM SISTEMAS DE PRODUÇÃO

Arnoldo Uber Junior

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciências da Computação, Área Inteligência Computacional e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Frank Augusto Siqueira, Dr.
Coordenador do PPGCC

Banca Examinadora

_____

Ricardo Azambuja Silveira, Dr., UFSC
Orientador

_____

Roberto Willrich, Dr., UFSC

_____

Silvia Modesto Nassar, Dra., UFSC

_____

Andre Luis Alice Raabe, Dr., UNIVALI

“Do or do not do! There is no try.” – Jedi Master Yoda

“Whether You Think You Can or Can’t - You’re Right” – Henry Ford

“The only place where success comes before work is in the dictionary.” – Albert Einstein

AGRADECIMENTOS

Aos meus pais Anselmo e Olga, aos meus irmãos José e Jacqueline e a minha esposa querida, Luciane, por me apoiarem, motivando e ajudando durante o desenvolvimento deste trabalho.

Ao meu amigo, orientador e professor Dr. Ricardo Azambuja Silveira, pela confiança, paciência, conhecimento, dedicação e orientação empregados na elaboração deste trabalho.

Ao corpo docente e demais membros do PPGCC, pelo brilhante trabalho acadêmico e científico que realizam, fomentando a pesquisa científica e formando novas gerações de pesquisadores.

A Universidade Federal de Santa Catarina – UFSC e ao governo de nosso país, por viabilizar e estruturar o Programa de Pós-Graduação em Ciências da Computação - PPGCC.

Aos amigos mestrandos Cícero, Jocimara, Jonas, Júlia, Mathias, Natanael e Ricardo, pela amizade e companheirismo em nossa vida acadêmica.

Aos membros do grupo de pesquisa em Inteligência Artificial e Tecnologia Educacional (IATE) e do Laboratório de Conexionismo e Ciências Cognitivas (L3C) por me receberem e disponibilizarem infra-estrutura para realizar minhas pesquisas quando nas dependências da UFSC.

A Operacional Têxtil Consultoria e Sistemas Ltda por ter apoiado e disponibilizado informações importantes para a formalização do problema, projeto e testes do *framework.*

A todas as pessoas que direta ou indiretamente contribuiram para o desenvolvimento deste trabalho.

Dedico este trabalho à minha família, por todo apoio, dedicação, compreensão e amor...

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

UBER JUNIOR, Arnoldo. **Framework para Escalonamento Distribuído de Processos Utilizando Sistema Multiagentes em Sistemas de Produção**. Florianópolis, 2009. no 132. Dissertação de Mestrado (Pós-Graduação em Ciência da Computação)–Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2009.

O estudo de técnicas de escalonamento de processos remete à criação dos primeiros sistemas operacionais (SO), com os algoritmos escalonadores de processos com e sem preempção. Porém a utilização de escalonadores de processo atinge outras áreas além dos SO, afeta todos os problemas onde há um conjunto de tarefas a serem executadas e um conjunto de unidades executantes. O tempo de execução final das tarefas é diretamente afetado pela seqüência de execução adotada, como é o caso dos sistemas de produção, que necessitam de informação em tempo real, para a execução de tarefas ou para o diagnóstico de problemas, com o objetivo da rápida tomada de decisão. Desse modo é necessário precisão e agilidade no processamento, nas mudanças de prioridades, e principalmente, eficiência no gerenciamento da informação. Este trabalho propõe um *framework* para escalonamento distribuído de processos, utilizando a teoria de agentes e a técnica heurística de busca, Algoritmos Genéticos (AG). Na modelagem do *framework* e aplicação foi utilizado a metodologia MaSE (*Multi-agent System Engineering*), que especifica etapas para análise e projeto de sistemas multiagentes. O desenvolvimento do *framework* e aplicação foi integrado à plataforma JADE (Java Agent Development Framework), utilizando as ontologias desenvolvidas no editor Protégé. A fim de validar o *framework*, desenvolveu-se um estudo de caso utilizando o *framework* HIPS (*Hybrid Intelligent Process Scheduler*) e os resultados e limitações obtidos com esse estudo de caso, comparados a outro escalonamento de processos.

Palavras-chave: Sistemas Multiagentes, Escalonamento de Processos, *Job Shop,* Algoritmos Genéticos.

ABSTRACT

The study of techniques for processes schedulers is linked to the creation of the first operational systems (OS), with the process schedulers algorithms with and without preemption. However, the use of processes schedulers reaches other areas besides OS. It affects all the problems where there is a set of tasks to be executed and a set of executing units. The tasks final execution time is directly affected by the execution sequence that is adopted, as in the case of production systems, which need information in real time, for the execution of tasks or for the diagnosis of problems aiming fast decision making. Thus, precision and agility in processing, in changing priorities and mainly, efficiency in managing the information are needed. This work proposes a framework for processes distributed scheduler, using the agents theory and the Genetic Algorithms (GA) heuristic search technique. In the framework modeling and application the MaSE (Multi-agent System Engineering) was used, which specifies stages for analyses and project of multi agents systems. The framework development and application were integrated to a JADE (Java Agent Development Framework), using the ontology developed in the Protégé editor. For the framework validation, the study of a case was developed using the HIPS (Hybrid Intelligent Process Scheduler) framework and the results and limitations obtained in this study were compared to another process scheduler.

***Keywords**: Multi-agent Systems, Process Scheduling, Job Shop, Genetic Algorithms.*

LISTA DE ABREVIATURAS

ACL -	Agent Communication Language
ACC -	Agent Communication Channel
AG -	Algoritmo Genético
AID -	Agent Identifier
AMS -	Agent Management System
AOSE -	Agent Oriented Software Engineering
BT -	Busca Tabu
DF -	Directory Facilitator
ERP -	Enterprise Resource Planning
FIPA -	*Foundation for Intelligent Physical Agents*
GUI -	Graphical User Interface
GUID -	Globally Unique Identifier
HIPS -	Hybrid Intelligent Process Scheduler
HIPS TNT -	Hybrid Intelligent Process Scheduler TiNTuraria
IAD -	Inteligência Artificial Distribuída
JADE -	Java Agent Development Framework
JHIPS -	Java Hybrid Intelligent Process Scheduler
JSS -	Job Shop Scheduling
JVM -	Java Virtual Machine
KQML -	Knowledge Query and Manipulation Language
MaSE -	Multi-agent Software Engineering
MTS -	Message Transport System
PD -	Programação Dinâmica
RMA -	Remote Agent Management
SMA -	Sistema Multiagente
SO -	Sistema Operacional
UML -	Unified Modeling Language

1 INTRODUÇÃO

O estudo de técnicas de escalonamento de processos remete a criação dos primeiros sistemas operacionais (SO) com os algoritmos escalonadores de processos. Porém a utilização de escalonadores de processo atinge outras áreas além dos SO, afeta todos os problemas onde há um conjunto de tarefas a serem executadas e um conjunto de unidades executantes e o tempo de execução final das tarefas é diretamente afetado pela seqüência de execução adotada. Tais problemas são encontrados em: seqüenciamento de projetos, agendadores de tarefas, *Job-Shop Scheduling* (JSS), planejamento de produção, etc.

A demanda por informação em tempo real torna-se realidade a cada instante, onde sistemas de produção necessitam de um fluxo de informação contínuo e ágil, buscando aperfeiçoar os processos de fabricação para atender a grande variedade de configurações que seus produtos podem compor, flexibilizando, diminuindo custos, melhorando a qualidade e entregando seus produtos no prazo, na quantidade correta e conforme o solicitado.

Sacile (2005) cita que flexibilizar o processo de industrialização é a habilidade de organizar e reorganizar os recursos de produção eficientemente sobre os aspectos: preço, qualidade, tempo de respostas às mudanças do ambiente e em particular, mudanças na demanda de tecnologia.

De acordo com Soares (2002), não basta apenas trabalhar com um bom planejamento e controle da produção. Para se manter competitivo é preciso otimizar os níveis de estoque, a utilização de recursos, os custos produtivos, os tempos de trocas e evitar solicitações não atendidas. Por esta razão, várias técnicas ou paradigmas computacionais têm surgido, tentando solucionar o problema da criação de planos ou programas de produção sob o enfoque da otimização. Alguns exemplos são o uso de estratégias como: BT (Lima, 2005) (Müller, 2009), AG (Oliveira 2000, Soares 2002 e Gonçalves 2005), *Ant Colony* (Carvalho 2007), *Simulated Annealing* e buscas locais (Yamada 1996), etc.

As ferramentas que possibilitam a utilização destas técnicas de forma independentes ou em conjunto, formando novas tecnologias híbridas, permitem a solução satisfatória de problemas complexos através de outras perspectivas, tornando problemas complexos em soluções reais.

Este trabalho propõe a modelagem de um *framework* para escalonamento distribuído de processos utilizando a teoria de agentes e métodos de busca heurística, demonstrando através de um estudo de caso sua aplicação e a avaliação dos resultados obtidos.

1.1 Problematização

Através da pesquisa em trabalhos que abordam o problema de escalonamento de processos, principalmente os voltados ao estudo de escalonamento de processos na produção, foram identificados algumas limitações na modelagem do problema (geralmente focando em uma determinada situação), propostas de solução e desenvolvimento da mesma, onde esta pesquisa visa propor melhorias.

A proposta de *framework* visa atuar na modelagem do problema, identificando os recursos produtivos, sua interligação e autonomia, como também no desenvolvimento de agentes inteligentes que visam melhorar gradativamente o fluxo produtivo através de seu relacionamento cooperativo ao escalonar processos de forma satisfatória.

O resultado do projeto modelado auxiliado pela interação dos agentes é retornado ao usuário através da integração entre a aplicação desenvolvida no *framework* e o sistema que controla a produção, podendo o usuário acompanhar a execução do sistema multiagente através de recursos disponíveis na plataforma de agentes e na própria aplicação.

1.2 Objetivos

Os objetivos desta pesquisa estão detalhados nessa seção e foram utilizados como guia no projeto e desenvolvimento.

1.2.1 Objetivo Geral

Propor um *framework* para escalonamento distribuído de processos utilizando a teoria de sistemas multiagentes que permita modelar, configurar, executar e avaliar problemas do tipo JSS em sistemas de produção.

1.2.2 Objetivos Específicos

Como objetivos específicos foram definidos:

- escalonar processos de forma distribuída, dividindo o processamento em um ambiente multiagente onde a sociedade de agentes através de negociação e orientação busque a melhor eficiência com menor tempo de execução dos processos escalonados;
- pesquisar, definir e aplicar uma metodologia de modelagem de engenharia de software específica para sistema multiagente na modelagem do framework proposto;
- pesquisar, definir e implementar o framework;
- avaliar o framework proposto.

1.3 Metodologia

A metodologia inicia com a revisão dos principais conceitos teóricos de escalonamento de processos disponíveis na literatura, e também, das recentes pesquisas feitas pela comunidade científica nesse sentido, ou seja, o estado da arte o qual se encontram soluções para esse tipo de problemas.

Analisando os problemas de escalonamento de processos e as técnicas de solução existentes, foram feitas pesquisas na área de inteligência artificial, buscando técnicas que possibilitassem a solução de problemas do gênero. Neste sentindo foram revistos os conceitos teóricos de agentes e a forma como poderiam ser organizados, direcionados e modelados no sentido de atingir os objetivos o qual lhes eram propostos.

Utilizando a teoria de agentes, foram pesquisadas metodologias para o desenvolvimento de sistema multiagentes, e também, plataformas que disponibilizassem uma arquitetura para o desenvolvimento, testes e simulação dos agentes modelados, abstraindo itens como a linguagem de comunicação, protocolos e estrutura de mensagens. Concentrando os esforços na resolução do problema.

Ainda com o intuito de otimizar o processamento no escalonamento de processos, foram estudadas técnicas heurísticas de busca, onde optou-se pelo uso de Algoritmos Genéticos, através de pesquisas na literatura.

A partir desses estudos foi formalizada uma proposta do *framework* HIPS (Hybrid Intelligent Process Scheduler), utilizando uma metodologia de modelagem de sistemas multiagentes.

Com a necessidade de modelar cenários de produção, foi desenvolvida a ferramenta HIPS Architect, que juntamente aos pacotes JHIPS, possibilitam a criação de aplicações para escalonamento distribuído de processos.

A partir de um problema de escalonamento de processos real, foi desenvolvida uma aplicação com o objetivo de validar o *framework* proposto. O desenvolvimento desta aplicação foi feito integrado ao sistema que gerencia a indústria, ou seja, os resultados do escalonamento de processos gerados pelos agentes eram diretamente informados ao sistema e direcionados aos recursos produtivos.

A fim de avaliar o desempenho da aplicação HIPS TNT (TiNTuraria), foram feitas experiências no escalonamento de processos de um setor, gerando um comparativo entre os escalonamentos feitos pela aplicação HIPS TNT e os que normalmente são executados. O comparativo será feito, analisando as ordens de produção de um determinado período, observando os tempos de início e término, duração, adiantamentos e atrasos na produção, além da análise da alocação dos recursos e do tempo total gasto por cada escalonamento, descrevendo as diferenças e ganhos de cada um.

1.4 Estrutura do trabalho

Este trabalho está estruturado em nove capítulos que são:

1. Introdução: procura introduzir a idéia a ser pesquisada, o escopo, objetivos, a metodologia empregada na busca por esses objetivos e a organização do trabalho;
2. Escalonamento de Processos: aborda os conceitos de escalonamento de processos, as técnicas para solução e o estado da arte;
3. Agentes: introduz os conceitos de agentes e sistemas multiagentes, relacionando também as metodologias de desenvolvimento de sistemas multiagentes e as plataformas disponíveis;
4. Algoritmos Genéticos: aborda os conceitos principais da técnica heurística AG, empregada no desenvolvimento da aplicação HIPS TNT;

5. Framework HIPS: apresenta a proposta do framework HIPS, através de modelagem orientada a objetos e sua especificação integrada a plataforma JADE, relacionando também as tecnologias envolvidas e as limitações;
6. Validação do Framework: descreve a aplicação desenvolvida utilizando o framework, através de uma metodologia para desenvolvimento de sistemas multiagentes, relacionando as tecnologias envolvidas e limitações;
7. Resultados Obtidos: apresenta os resultados obtidos no comparativo da aplicação desenvolvida com o framework e outro escalonamento de processos;
8. Considerações Finais: expõem as conclusões obtidas nessa pesquisa;
9. Referências: são descritas todas as fontes pesquisadas e referenciadas neste trabalho.

Ao término do capítulo nove, são apresentadas as publicações obtidas durante o desenvolvimento desta pesquisa.

2 ESCALONAMENTO DE PROCESSOS

O escalonador é componente básico para garantia de bom funcionamento em sistemas computacionais, pois é responsável por decidir qual tarefa será executada quando houver mais de uma pronta para execução. O algoritmo utilizado para tal finalidade é denominado algoritmo de escalonamento, o qual segue um determinado critério para escolher uma tarefa em detrimento de outras (Cruz, 2005). Segundo Oliveira (2000) o objetivo de um escalonador é a eficaz designação dos recursos para a execução das atividades de um projeto.

Segundo Baker (1974), escalonar é a alocação de recursos em relação ao tempo, para executar um conjunto de tarefas. Escalonadores são os atribuidores de tarefas para os recursos, determinando a seqüência cronológica e satisfazendo as regras para atribuição.

Pinedo (2005) afirma que as atividades de planejamento e escalonamento são processos de tomada de decisão usados como base em muitas empresas e serviços. O planejamento e escalonamento se baseiam em técnicas matemáticas e métodos heurísticos para alocar recursos limitados em atividades a serem executadas, desta forma, aperfeiçoam seus objetivos e atingem suas metas.

Pinedo (2005) descreve como sendo recursos: máquinas em um sistema de produção, aviões em um aeroporto, operários em uma construção civil e também, processos em um ambiente computacional. Atividades seriam: operações em um sistema de produção, vôos em um sistema aéreo, estágios em uma construção civil e processos em um sistema operacional (SO). Cada atividade tem sua prioridade, tempo limite para início e término de execução. O objetivo pode se apresentar de diferentes formas, sendo desde minimizar o tempo total de todas as atividades ou minimizar o número de atividades antes de um período, dentre outras formas.

O estudo de técnicas de escalonamento de processos remete à criação dos primeiros sistemas operacionais (SO) com os algoritmos escalonadores de processos sem preempção (FCFS - primeiro a chegar primeiro a executar, SJF - primeiro o mais curto, etc.), e os algoritmos com preempção (*Round-Robin*, Prioridade, SRTF - primeiro o processo que falta menos tempo, Filas multinível, etc.). Porém a utilização de escalonadores de processo atinge outras áreas além dos SO, afeta todos os problemas onde há um conjunto de tarefas a serem executadas e um conjunto de unidades executantes e o tempo de execução final das tarefas é diretamente afetado pela seqüência

de execução adotada. Tais problemas são encontrados em: seqüenciamento de projetos, *Job-Shop Scheduling* (JSS), planejamento de produção, etc.

A eficiência do algoritmo de seqüenciamento pode ser avaliada através da análise do desempenho do resultado gerado. Analisando um escalonador de processos para SO, pode ser analisado, por exemplo: percentual de ocupação da CPU, *throughput* (taxa de saída), tempos de espera, tempos de execução médios (*turnaround time*), tempo de troca entre processos, etc. A definição de uma medida de desempenho deve considerar a necessidade ou conjunto de necessidades da situação a ser empregada.

Compartilhar os recursos ao longo do tempo, para atividades concorrentes, é uma das principais atividades do escalonador, sendo uma das áreas que obtêm uma quantidade significativa de pesquisa. Ênfase é dada principalmente para problemas de programação de máquinas onde processos representam atividades e as máquinas os recursos compartilhados, podendo executar em sua maioria, uma tarefa de cada vez.

Blazewicz (1996) classifica os problemas de escalonamento como pertencentes a uma classe ampla de problemas combinatórios designados de problemas de procura Π.

2.1 Problemas de procura II

Blazewicz (1996) define o problema de procura Π sendo um conjunto de pares (I, A), onde I designa a instância do problema, isto é, um conjunto finito de parâmetros (geralmente, números, conjuntos, funções, grafos) com valores específicos e A uma resposta ou solução para essa instância do problema.

2.2 Classificação dos métodos de escalonamento

Os métodos de escalonamento possuem sua aplicação bastante ampla, visando resolver uma grande variedade de problemas, porém cada método é geralmente desenvolvido para resolver uma classe específica de problemas, desta forma, segundo Varela (2007) é necessário classificar os métodos e enquadrá-los nas classes de problemas que resolvem.

Varela (2007) classifica os métodos de escalonamento de processos em relação: a natureza da solução, a eficiência da solução, o tipo de objetivo e o tipo de técnica utilizada para solução.

2.3 Métodos de Pesquisa

Problemas onde o número de variáveis é reduzido possibilitam determinar todo o conjunto de soluções possíveis para o problema, desta maneira, pode se escolher a melhor solução. Entretanto, problemas onde o número de soluções possíveis é obtido através da explosão combinatória, a análise de cada solução individualmente e a escolha da melhor, torna o processo geralmente, em termos práticos, impossível.

Linden (2006) utiliza o termo “intratável” para descrever está classe de problemas, descrevendo-os como sendo problemas cujo tempo necessário para resolvê-lo é considerado inaceitável para o usuário da solução, ou seja, um problema é tratável se o seu limite superior de complexidade é polinomial, e é intratável, se o limite superior de sua complexidade é exponencial (2^n, n!, etc).

Os métodos de pesquisa na busca de uma solução satisfatória para problemas de escalonamento de processos podem ser divididos em várias categorias, como por exemplo: regras de seqüenciamento (Baker, 1974) (Silva, 1999), simulação, ramificação, limite, relaxação de *Lagrange*, pesquisa em feixe e *bottleneck* (Varela, 2007), programação dinâmica e árvores de decisão (Morton, 1993), redes neuronais (Bittencourt, 2006) (Russel, 2004) e de pesquisa local e meta-heurísticas o qual sera mencionado em detalhe.

2.3.1 Métodos de pesquisa local e metas-heurísticas

Russell (2004) cita a pesquisa heurística, também como busca com informação, onde a define como sendo uma estratégia que utiliza o conhecimento específico do problema, além da definição do próprio problema.

Os métodos de busca sem informação diferem dos com informação, pois não possuem nenhuma informação adicional sobre os estados, além da definição do problema. Segundo Russell (2004), o que as buscas sem informação podem fazer é gerar sucessores e distinguir um estado objetivo de um não-objetivo, onde os métodos de busca com informação podem distinguir se um estado é mais “promissor” que outro. São exemplos de métodos de busca sem informação: em extensão, custo uniforme, em profundidade, em profundidade limitada, aprofundamento interativo, bidirecional, etc.

Os métodos heurísticos de pesquisa local (*Local Search Heuristics*) ou métodos de pesquisa na vizinhança são métodos heurísticos que têm demonstrado grande aplicabilidade nos últimos anos (Varela, 2007). Os métodos heurísticos de pesquisa local (PL) ou de pesquisa na vizinhança (PV) são muitas vezes designados meta-heurísticos ou estratégias com capacidades de engenharia do conhecimento e aprendizagem, reduzindo a incerteza, enquanto o conhecimento para os ajustes do problema é explorado e adquirido. No intuito de melhorar e acelerar o processo de pesquisa estes procedimentos tentam guiar, heuristicamente o processo de pesquisa para o melhor resultado (Morton, 1993).

> Segundo Russell (2004) métodos como: melhor escolha, A* (A estrela), recursiva pelo melhor (BRPM), A* de aprofundamento interativo (AIA*) também são métodos de buscas heurísticos, porém procuram buscar sistematicamente o espaço de busca, mantendo um ou mais caminhos na memória e registrando as alternativas exploradas em cada ponto ao longo do caminho. Quando um objetivo é encontrado, o caminho até esse objetivo também constitui uma solução do problema.

Os métodos de busca local, segundo Russell (2004) operam usando um único estado corrente e em geral se movem apenas para os vizinhos desse estado, além de possuírem a característica de não se importarem com o caminho seguido até o seu objetivo. Duas vantagens desses métodos: pouca utilização de memória e freqüentemente podem encontrar soluções razoáveis em grandes ou infinitos espaços (contínuos) de estados para os quais os métodos sem informação são inadequados.

Carvalho (2007) cita como dois conceitos importantes para o estudo de metas-heurísticas são o de intensificação e diversificação. A intensificação é a exploração mais exaustiva de uma região do espaço de busca onde se espera encontrar boas soluções, ao passo que a diversificação é a mudança do foco da busca para regiões ainda não exploradas.

2.3.1.1 Busca *Hill-climbing*

A técnica do *hill-climbing* ou também, subida de encosta é uma técnica em que se encontram melhores soluções ao explorar soluções “próximas” de uma dada solução atual ou corrente e melhor até então encontrada. Esta técnica funciona bem num espaço de procura com relativamente “poucos” altos e baixos (*hills*), isto é, em espaços de procura pouco irregulares (Varela, 2007).

Russell (2004) define este método como um laço repetitivo que se move de forma contínua no sentido do valor crescente, ou seja, encosta acima. O método termina quando alcança um “pico” em que nenhum vizinho tem valor mais alto, não mantendo árvore de busca, o método só necessita registrar o estado e o valor de sua função objetivo. Este método possui alguns problemas, como os citados por Russell (2004): máximos locais, picos e platôs.

2.3.1.2 Busca Tabu

A busca tabu (BT) é uma meta-heurística baseada no uso de técnicas baseadas em proibição e em esquemas “inteligentes” como um complemento a métodos heurísticos de base, do tipo heurístico de pesquisa local simples, com o objetivo de guiar o processo de pesquisa para fora de ótimos locais (Varela, 2007).

O nome desta busca, ou seja, “tabu” provem de uma língua da Polinésia chamada Tongan, falada pelos aborígines que moram na Ilha Tonga e indica algo que não pode ser tocado por ser sagrado (Glover, 1997). Conforme o significado do nome, o método consiste em listas com soluções não permitidas. Na sua forma mais básica, contêm os n últimos elementos visitados.

Segundo Glover (1997) a busca tabu que foi projetada para encontrar boas aproximações para a solução ótima global de qualquer problema de otimização, possui três princípios fundamentais:

- uso de uma estrutura de dados (lista) para guardar o histórico da evolução do processo de busca;
- uso de um mecanismo de controle para fazer um balanceamento entre a aceitação ou não de uma nova configuração, com base nas informações registradas na lista tabu referentes às restrições e aspirações desejadas;
- incorporação de procedimentos que alternam as estratégias de diversificação e intensificação.

Na Figura 1 é possível visualizar a interação de seus componentes.

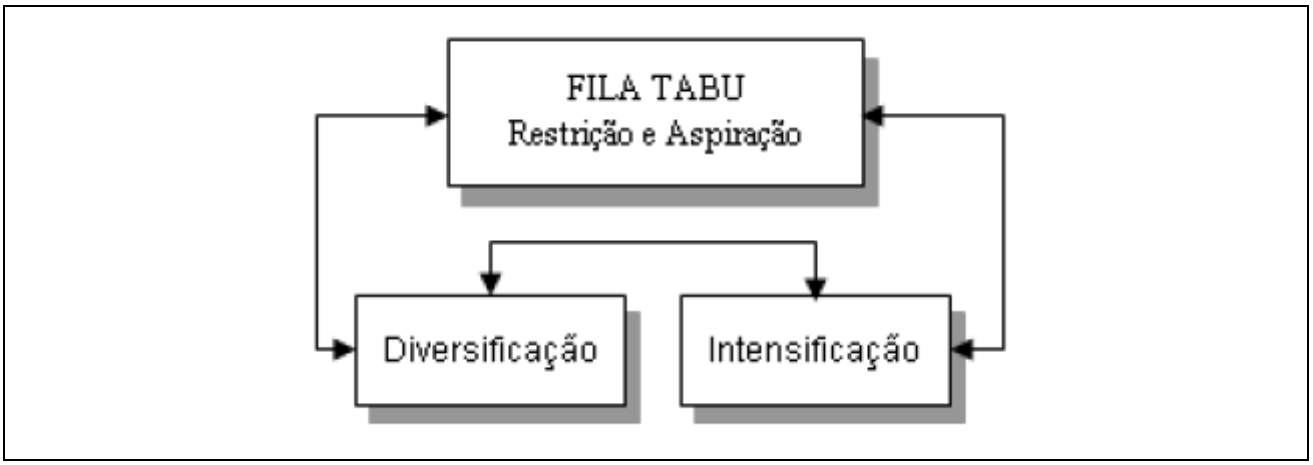

Figura 1. Estratégia de busca da BT
Fonte: Müller (2006).

O método BT procura imitar o processo de memória dos seres humanos. Não somente a memória imediata (local), mas também a memória de mais longa duração. Esta última está relacionada a uma parte da extensão do processo de exploração do algoritmo, enquanto a primeira está relacionada ao valor da solução corrente. O uso sistemático de memória é a sua característica principal (Lima, 2005).

De forma resumida, a BT começa a partir de uma solução inicial, onde a cada passo, é gerada a vizinhança da solução atual. É verificado entre os vizinhos, o que possui as melhores características, onde este vizinho passa a ser a nova solução atual e é repetido este procedimento. No intuito de prevenir repetições e guiar o algoritmo, é mantido um histórico das soluções já visitadas em memória, onde geralmente é dividido em memória de curta e longa duração. Uma das memórias de curta, entre as muitas estruturas de memória propostas (Glover, 2002), é chamada de "Lista Tabu" e tem o objetivo de não permitir que o algoritmo volte e analise uma solução pela qual tenha passado nos passos anteriores.

2.3.1.3 Simulated Anneling

Simulated Anneling ou arrefecimento simulado é uma meta-heurística que faz uma analogia com um processo térmico, chamado *annealing* ou recozimento, que é utilizado na metalurgia para obtenção de estados de baixa energia em sólidos (Varela, 2007), (Morton, 1993). O método basicamente é o arrefecimento lento de sólidos, após estes terem sido aquecidos até ao seu ponto de fusão, ou seja, o metal é aquecido a altas temperaturas, provocando um choque violento nos átomos.

Se o metal é resfriado de forma brusca, a microestrutura tende a um estado randomicamente instável. Se o metal é resfriado de forma suficientemente lenta, o sistema procurará um ponto de equilíbrio caracterizado por uma microestrutura ordenada e estável (Varela, 2007).

Segundo Petrowski (2006) o algoritmo de arrefecimento simulado substitui a solução atual por uma solução próxima (sua vizinhança no espaço de soluções), escolhida de acordo com uma função objetivo e com uma variável T (dita Temperatura, por analogia). Quanto maior for T, maior a componente aleatória que será incluída na próxima solução escolhida. À medida que o algoritmo progride, o valor de T é decrementado, começando o algoritmo a convergir para uma solução ótima, necessariamente local. Uma das principais vantagens deste algoritmo é permitir testar soluções mais distantes da solução atual e dar mais independência do ponto inicial da pesquisa.

Varela (2007) descreve que a técnica de *simulated annealing* tem sido aplicada a vários tipos de problemas de otimização combinatória, com diversos níveis de sucesso. De modo geral, Varela (2007) diz que esta técnica é um procedimento viável para usar em situações em que o conhecimento é escasso ou se aparenta difícil de aplicar algoritmicamente. Mesmo para dar soluções a problemas complexos, esta técnica é relativamente fácil de implementar e normalmente executa um procedimento do tipo *hill-climbing* com múltiplos recomeços.

2.3.1.4 Ant Colonies

A otimização por colônia de formigas, ou *Ant Colony Optimization* (ACO) faz uma analogia ao comportamento de formigas na busca de alimentos. Quando uma formiga precisa decidir para onde ir, ela usa informação proveniente de feromônio previamente depositado por outras formigas que passaram por aquele local. A direção que tiver maior depósito de feromônio será escolhida pela formiga. Por este processo de busca, formigas são capazes de encontrar o menor caminho de uma fonte de comida para o seu ninho (Carvalho, 2007).

Carvalho (2007) cita que a meta-heurística ACO está baseada em um processo de construção de solução e sua principal inovação é usar formigas artificiais que, ao percorrer um caminho, depositam certa quantidade de feromônio, que, então, irá influenciar a decisões das formigas que vierem em seguida.

Segundo Petrowski (2006) a meta-heurística ACO é mais efetiva quando é utilizada acompanhada de outro método de busca local. Estes métodos de busca local trabalharam em cima dos resultados obtidos pelo ACO, pois a as vantagens de utilizar ACO para gerar a população inicial é inegável.

2.3.1.5 Algoritmos Genéticos

O algoritmo genético (AG) é uma meta-heurística que faz uma analogia a seleção natural. Provêm de um ramo da ciência da computação chamado computação evolutiva (CE), o qual segundo Bittencourt (2006), não exige, o conhecimento prévio de uma maneira de encontrar a solução, para resolver um problema. A CE é baseada em mecanismos evolutivos encontrados na natureza, tais como a auto-organização e o comportamento adaptativo.

Os algoritmos genéticos funcionam mantendo uma população de estruturas, denominadas indivíduos ou cromossomos, que se comportam de forma semelhante à evolução das espécies (Linden, 2006). Os chamados operadores genéticos são aplicados sobre essas estruturas, onde cada indivíduo recebe uma avaliação, ou seja, passa pela função objetivo, que classifica a qualidade como solução do problema. Análogo ao processo natural, os operadores genéticos são aplicados de forma a simular a sobrevivência do indivíduo mais apto.

Segundo Bittencourt (2006) uma das primeiras aplicações propostas para os AG, seguindo o enfoque de Holland, foram os sistemas classificadores, que são sistemas de produção, onde utilizam os AG em parte do algoritmo global.

O capítulo 4 deste trabalho de pesquisa descreve de forma mais abrangente o funcionamento dos AG.

2.4 Sistemas de Escalonamento de Produção

Segundo Varela (2007) existem diferentes tipos de abordagens ao escalonamento da produção, resultando diversos tipos de sistemas, nomeadamente sistemas baseados em inteligência artificial (IA), sistemas baseados em redes neuronais e algoritmos genéticos, e ainda uma variedade de outros tipos de sistemas, sistemas de pesquisa operacional, tais como, sistemas de programação

matemática e diversos outros tipos de sistemas de apoio a decisão (SAD) no escalonamento da produção, incluindo sistemas híbridos.

O objetivo dessa seção é apresentar alguns dos principais sistemas de escalonamento do produção visando melhor contextualizar este trabalho de pesquisa, porém não abordará detalhadamente cada um dos sistemas citados, o que é realizado em Pinedo (2005), Varela (2007) e Parunak (1988).

Alguns dos principais sistemas de escalonamento da produção desenvolvidos nas últimas décadas, principalmente nos anos 90, segundo Varela (2007) são:

- LEKIN: Este sistema foi desenvolvido por Pinedo (2005) e é um ambiente do tipo job shop flexível. O sistema LEKIN foi especificamente projetado para escalonamento de trabalhos num conjunto de ambientes diferentes, incluindo recursos paralelos, linhas e células de produção flexíveis;
- Library of Scheduling Algorithms (LISA): Desenvolvido por Pinedo (2005) é um pacote de software para entrada, edição e resolução de problemas determinísticos de escalonamento, com particular incidência nos problemas de processador único (Varela, 2007);
- Lin e Lee: sistema para controle e programação integrada de células flexíveis de manufatura, com base em redes de Petri;
- Kazerooni: sistema de suporte à decisão, baseado em lógica difusa, para a programação da produção em sistemas flexíveis de manufatura, usando simulação digital;
- Yet Another Manufacturing System(YAMS): Desenvolvido por Parunak (1988), aborda o conceito de inteligência artificial distribuída e agentes inteligentes compostos. Trata-se de um sistema de planejamento e controle da produção para ilustrar a utilidade e aplicabilidade da inteligência artificial distribuída neste domínio. Este sistema usa um modelo em rede para simular uma implantação típica da manufatura;
- Shop-floor Contingency Rescheduling Expert (SCORE): é um sistema de reprogramação em tempo real, que interage com um sistema de planejamento de necessidades de materiais (MRP);
- PATRIARCH: é um sistema que permite resolver problemas de planejamento e programação de projetos em sistemas flexíveis de manufatura;
- MASCOT: é um sistema para programação de células de produção, que usa uma análise do tipo análise baseada em restrições;

- OPAL: Este é um sistema de programação da produção em células de fabricação que utiliza uma representação de conhecimento em forma de regras de produção;
- ISIS: é baseado em linguagem de representação por esquemas. O esquema consiste no nome do esquema e um conjunto de campos. No processo de geração da programação, várias condições são usadas para direcionar o procedimento de pesquisa. Estas condições consideram os parâmetros prazos, recursos e custos;
- NEOS Server: é um sistema de web que pode ser usado para a resolução de problemas de otimização, incluindo problemas de escalonamento da produção, com particular destaque para um tipo particular de problema, que é o problema do caixeiro viajante;
- Vishnu: O sistema Vishnu é um mecanismo poderoso de escalonamento para diversos cenários práticos reais. Possui um módulo facilmente adaptável para realizar diversas formas de escalonamento, desde otimizar operações e serviços, planejamento estratégico logístico, incluindo escalonamento de processos. O Vishnu inclui interfaces de utilizador para visualização e edição de programas e de dados.

2.5 Job Shop X Flow Shop

A maioria dos problemas de escalonamento de processos aplica-se ao ambiente de programação da produção, conhecido como *Job Shop Schedule* (JSS). Oliveira (2000) afirma que o *Job Shop* tradicional é caracterizado por permitir diferentes fluxos das ordens entre as máquinas e diferentes números de operações por ordem, que são processadas apenas uma vez em cada máquina, de acordo com

Também segundo Oliveira (2000) o problema JSS é um problema combinatório, o que chega a se tornar NP - completo em determinadas situações (enumeração explícita ou implícita de todas as alternativas possíveis para garantir a solução ótima) como exemplificado na Figura . Desta forma, algoritmos otimizantes são computacionalmente viáveis quando aplicados a problemas reais pequenos, com objetivos limitados. Para problemas de porte similar aos encontrados no ambiente real, costuma-se sacrificar a obtenção de uma solução ótima por métodos heurísticos, que resultem em uma solução subótima, com tempo computacional aceitável.

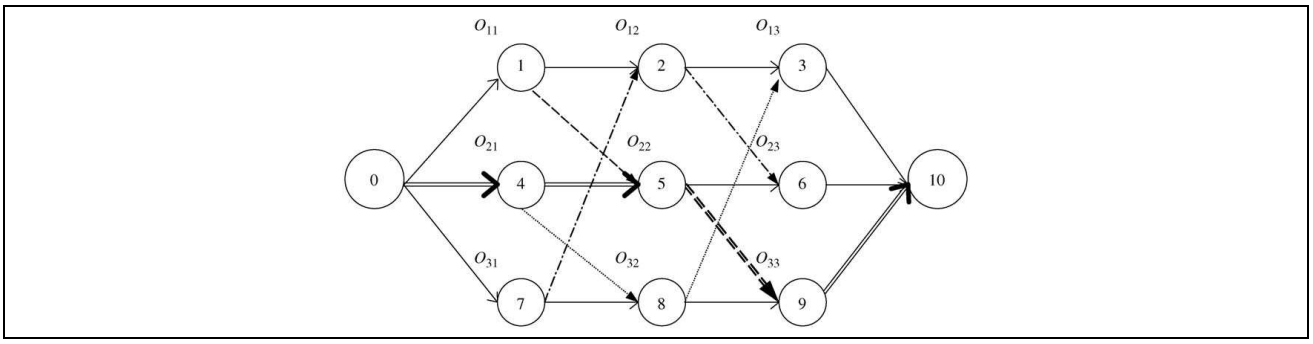

Figura 2. Esquema de Job Shop
Fonte: Aydin (2004)

A complexidade do mesmo quando o objetivo é minimizar o tempo de processo (makespan) pode ser observada em Jain (1998), onde é feito inclusive uma analogia ao problema NP - Completo do Caixeiro Viajante.

Segundo Borges (2002), em uma ambiente de fabricação, pode ser visto também um espectro contínuo de diferenciação, onde em um extremo está o *flow shop* e no outro o *job shop*.

Um ambiente do tipo *job shop* é aquele em que os materiais se deslocam na fábrica com rotas dependentes do tipo de trabalho a ser executado. Já um do tipo *flow shop* caracteriza-se pelo fato dos materiais e peças se deslocarem na fábrica com rotas constantes. As situações reais de produção que se enquadram entre esses dois tipos, ou como uma combinação de ambos (Borges, 2002).

Há problemas também, conhecidos por *open shop* e *mixed shop*, onde no primeiro, em termos práticos é um *job shop*, porém as etapas pelo qual os materiais irão passar variam durante o processo e o segundo, seria o ambiente que reúne os três ambientes citados.

Uma maneira de diferenciar melhor ambos pode ser vista através da Tabela 1.

Tabela 1. Características entre Job Shop e Flow Shop

Flow Shop	**Job Shop**
Rotas fixas	Rotas variáveis
Layout por produto	Layout por processo
Equipamento especializado	Equipamento flexível
Produção para estocar em grandes volumes	Produção por encomendas em pequenos volumes
Maiores *lead time* para aumentar a capacidade	Menores *lead time* para aumentar a capacidade
Capacidade bem definida	Capacidade difícil de definir
Intensiva em capital	Intensiva em operários
Baixos estoques intermediários	Estoques intermediários significativos
Menor coordenação de materiais	Maior coordenação de materiais
Operadores altamente especializados e treinados	Operários possuem certa habilidade em algum tipo de operação para monitorar e controlar os equipamentos de processo e/ou máquina que fabricam os produtos
Overlapping nas operações	Inexistência de *overlapping* nas operações
Falha nos equipamentos pode parar a planta	Falha nos equipamentos pode parar a produção de alguns itens
Atraso no recebimento de materiais	Atraso no recebimento de materiais e peças pode parar a planta e atrasam a produção de itens
Maior consumo de energia	Menor consumo de energia

Fonte: Adaptado de Borges (2002)

Por se tratar de um problema que tende a ser NP - Completo e possibilitar vasta aplicação, o problema JSS foi escolhido para aplicação do *framework* proposto neste trabalho, onde JSS considera-se um problema composto de características de *job shop* e *flow shop*, porém não haverá restrições do *framework* para modelagem de problemas do tipo *open shop* e *mixed shop.*

2.6 Integração com outros Sistemas

Chase (2006) afirma que a programação da produção está no coração do que é atualmente chamado de Sistemas de Execução da Manufatura (MES). O MES é um sistema de informações que programa, despacha, rastreia, monitora e controla a produção no chão-de-fábrica. Esses sistemas também fornecem ligações, em tempo real, aos sistemas MRP[1], ao planejamento do produto e dos

[1] Sistema de Planejamento das Necessidades de Materiais (MRP) são sistemas que criam programações que identificam as peças e materiais necessários para produzir os itens finais, os números exatos necessários e as datas quando os pedidos para esses materiais devem ser liberados e recebidos ou completados dentro do ciclo de produção. O propósito principal de um sistema MRP consiste em controlar os níveis de estoque, atribuir as prioridades operacionais e planejar a capacidade para carregar o sistema de produção (Chase, 2006).

processos, assim como aos sistemas que se estendem além da fábrica, incluindo a administração da cadeia de suprimentos, o ERP[2], as vendas e a administração de serviços.

Segundo Pinedo (2005) empresas modernas elaboram sistemas de informações industriais contendo redes de computadores e diversos sistemas gerenciadores de banco de dados (SGBD), onde a rede conecta computadores pessoais, estações de trabalho e terminais de entrada de dados a servidores centrais. O planejamento e escalonamento geralmente são feitos em uma dessas estações de trabalho, situada em um "ponto chave" da empresa, permitindo acesso on-line dessas informações, tais como: status das ordens, status de máquinas e níveis de estoque aos demais terminais conectados. Para este fim, geralmente as empresas recorrem aos sistemas ERPs, que controlam e coordenam as informações de todos os setores e muitas vezes, de clientes e fornecedores.

Sistemas de apoio à decisão de vários tipos diferentes podem ser ligados a um sistema ERP, permitindo que a empresa faça planejamento em longo prazo, em médio prazo e também, com tempo curto de programação (Pinedo, 2005).

A aplicação desenvolvida através do *framework* proposto nesta pesquisa utiliza como fonte e destino de informação um sistema ERP, visando o apoio a decisão no escalonamento de processos, voltado à produção, nos setores para o qual for utilizada.

2.7 Trabalhos correlatos

A solução de problemas do tipo JSS, é abordada em vários trabalhos e descrevem várias maneiras de obter uma solução satisfatória para o problema. Jain (1998) descreve o problema II de forma detalhada e cita diversas técnicas de solução empregadas ao longo dos últimos quarenta anos, onde desenvolve, praticamente, uma cronologia das soluções para este problema.

Conforme já mencionado no item 2.3.1, a aplicação de buscas locais em problemas de complexidade NP - Completo tem demonstrado grande aplicabilidade nos últimos anos (Varela,

[2] *Enterprise Resource Planning* (Planejamento de recursos empresariais) ou SIGE (Sistemas Integrados de Gestão Empresarial) são sistemas de software que integram aplicações em finanças, produção, logística, vendas, marketing, recursos humanos e outros setores de uma empresa. Esta integração é realizada através de uma base de dados compartilhada por todas as aplicações (Vollmann, 2006).

2007). As metas-heurísticas, para o problema JSS, geralmente são aplicadas de forma individual, como a BT em Lima (2005) e Müller (2009), AG em Oliveira (2000), Soares (2002) e Gonçalves (2005), *Ant Colony* em Carvalho (2007), *Simulated Annealing* e buscas locais em Yamada (1996), etc. Porém há casos onde a combinação de várias técnicas pode gerar melhores resultados (Bittencourt, 2006), conforme estudos realizados por Oliveira (2000), Cruz (2005) e Müller (2006).

O estudo de técnicas metas-heurísticas propicia consideráveis avanços na busca de soluções satisfatórias aos problemas NP - Completos, em específico o abordado nesta pesquisa. Dentre as metas-heurísticas estudadas, foi dada especial atenção ao grupo composto por: BT, AG, *Ant Colony* e *Simulated Annealing*, por serem freqüentemente citadas na literatura pesquisada (Müller (2006), Oliveira (2000), Cruz (2005), etc) e por terem trabalhos especificamente dedicados ao seu estudo, como o caso de Petrowski (2006).

Conforme mencionado por Parunak (1997) o uso da tecnologia de agentes é justificado quando as aplicações são modulares, descentralizadas, modificáveis e complexas, o que não os torna para os softwares industriais, uma panacéia. Parunak (1997) ainda define a possibilidade de uso em aplicações industriais, sobre o ponto de vista do ciclo de vida do produto, analisando três áreas em que os agentes podem ser usados efetivamente: desenvolvimento de produtos, controle de processo no planejamento de escalonamento e controle de processos em baixo nível no controle de equipamentos.

Além das técnicas metas-heurísticas, seja ela individual ou híbrida, há trabalhos que modelaram o problema JSS em um modelo que faz uso da teoria de agentes e sistemas multiagentes, como o proposto por Aydin (2004), que utiliza a proposta de Talukdar (1993), ou seja, uma arquitetura de agentes autônomos que operam de forma assíncrona em uma população compartilhada de possíveis soluções, chamada de “ATeams” e descreve uma solução para o problema JSS.

Outro problema comumente estudado sobre escalonamento de processos utilizando sistemas multiagentes é o agendamento de tarefas, tal como proposto por Wada (1997), que compara sistemas multiagentes a sistemas cliente-servidor e propõe um ambiente multiagente que administra uma agenda, tendo os agentes a tarefa de negociar com outros agentes a adequação de horários. Xin (2000) desenvolveu um trabalho semelhante, utilizando AG como meta-heurística no processo de negociação dos agentes.

Outros trabalhos também foram realizados no sentido de agregar à teoria de agentes e sistemas multiagentes o uso das técnicas heurísticas, buscando otimizar o tempo de respostas dos agentes e melhorar os algoritmos de negociação, como Iba (1996), que propôs o uso de programação genética em um sistema multiagente buscando uma cooperação emergente.

Weichhart (2004) propõem uma solução para problemas de escalonamento de processos em sistemas *Holonic* (Bongaerts, 1998), utilizando sistema multiagentes e algoritmos genéticos, descrevendo um cenário de comunicação entre clientes, empresas virtuais e recursos de produção na projeção da data de entrega. Gonçalves (2005) também faz uso das teorias de sistemas multiagentes e AG, porém utiliza uma técnica híbrida de AG e pesquisa operacional.

Conceitos de teoria dos jogos, utilização de estratégias e protocolos de negociação e barganha também são utilizados quando se utiliza sistemas multiagentes, como foi proposto por Shin (2001), que propõem um protocolo de negociação chamado MANPro (*Mobile Agent-based Negotiation Protocol*) aplicado em um sistema distribuído de produção inteligente, onde implementa um sistema de leilão, fazendo com que os agentes interajam e busquem melhorar a performance do sistema. Parsons (2002) adiciona ao uso da teoria dos jogos e sistemas multiagentes, a teoria das decisões, onde descreve o uso da teoria de jogos e decisões em um ambiente multiagente de negociação.

Solari (2007) aplica a meta-heurística AG em um sistema multiagente escalonador de processos para a indústria, com o objetivo de obter melhores resultados. Nesta proposta, menciona a criação de agentes com tarefas específicas, desde a obtenção de ordens de manufatura ao controle de estoque, trabalhando o processo de escalonamento de processos a nível gerencial e com uma situação pré-determinada de atuação.

A maioria das pesquisas que descrevem possíveis soluções para problemas do tipo JSS se baseiam em um cenário simulado de produção e geralmente abordam uma determinada técnica. A proposta de um framework para escalonamento de processos distribuídos, busca propiciar a formalização de um modelo do cenário de produção e a possibilidade de diversas soluções para o mesmo problema, variando de acordo com os parâmetros informados e a implementação de agentes originados das classes bases do framework.

3 AGENTES

Um agente de *software* é conceituado como uma entidade que funciona de forma contínua e autônoma em um ambiente em particular, geralmente habitado por outros agentes, e que seja capaz de intervir no seu ambiente, de forma flexível e inteligente, sem requerer intervenção ou orientação humana constante. De um modo ideal, um agente que funcione, continuamente, por longos períodos de tempo, pode ser capaz de aprender com a experiência e, se ele habita um ambiente com outros agentes, for capaz de comunicar-se e cooperar com eles, e ainda mover-se de um local para outro (Silveira, 2001).

Russell (2004) define genericamente um agente como uma abstração de tudo que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores.

Maes (2009) define um agente como sendo um sistema computacional que possui longo tempo de execução, possui objetivos, sensores e atuadores, decide autonomamente que ações tomar em uma determinada situação para maximizar o progresso em relação aos seus objetivos.

A Figura 3 mostra a idéia do agente interagindo com o ambiente através da percepção de seus sensores e reação através dos atuadores.

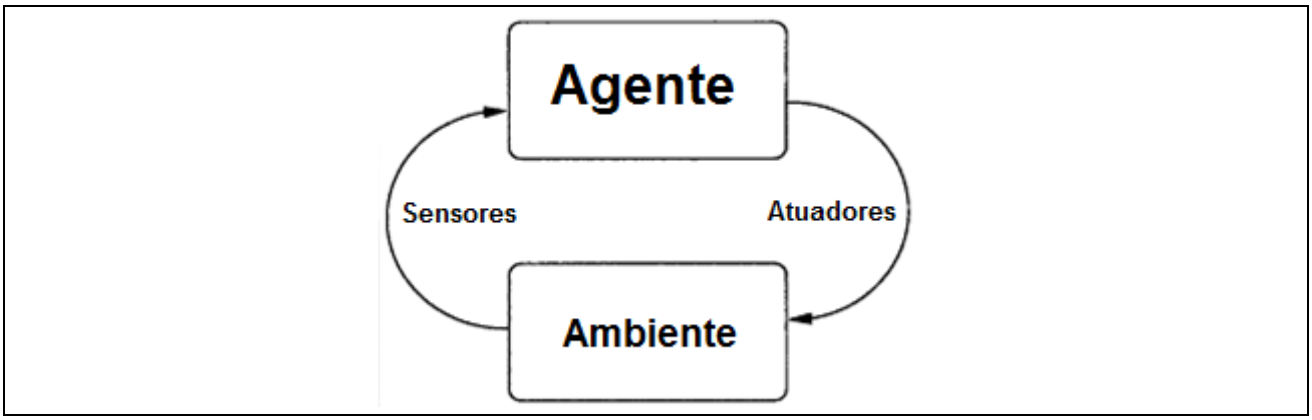

Figura 3. Agente interagindo com o ambiente
Fonte: Adaptado de Weiss (1999)

Um agente pode, de acordo com o tipo de problema com o qual está habilitado a tratar, possuir, em maior ou menor grau, os seguintes atributos (Bradshaw 1997):

- reatividade: A habilidade de perceber o ambiente de modo seletivo e manifestar um comportamento como resposta a um estímulo externo;
- autonomia: Comportamento dirigido a objetivos, pró-ativo e auto-iniciado;
- comportamento cooperativo: Trabalhar com outros agentes para atingir um objetivo comum;
- habilidade de comunicação ao nível de conhecimento: Capacidade de comunicar-se com pessoas ou outros agentes em uma linguagem de mais alto nível que um simples protocolo de comunicação programa a programa;
- capacidade de inferência: Capacidade de agir a partir de especificações abstratas de tarefas, usando conhecimentos prévios;
- continuidade temporal: Persistência de identidade por longos períodos de tempo;
- personalidade: Capacidade de demonstrar atributos de um personagem;
- adaptabilidade: Habilidade de aprender com a experiência;
- mobilidade: Habilidade de migrar de uma plataforma para outra.

Geralmente um agente não atua sozinho, mas em uma sociedade de agentes, interagindo entre si e com o ambiente através de seus sensores, que captam as alterações no ambiente e a comunicação e interação dos outros agentes e toma ou não a decisão de interagir também através de seus atuadores.

De acordo com Wooldridge (2002), só pode ser considerado um agente inteligente, os agentes que possuírem as habilidades que possibilitem atingir os objetivos previstos:

- reatividade: devem ser capazes de perceber seu ambiente e responder em tempo hábil as mudanças que ocorrem, a fim de satisfazer os objetivos definidos;
- pró-atividade: devem ser capazes de tomar a iniciativa, administrando seu comportamento;
- habilidade social: devem ser capazes de interagir com outros agentes (e possivelmente humanos).
- Weiss (1999) classifica os agentes por sua arquitetura, ou seja, as estruturas que utiliza para a solução de um problema:
- agentes baseado em lógica: onde cada tomada de decisão é baseada na dedução lógica;
- agentes reativos: onde a tomada de decisão é implementada no mapeamento direto situação para ação;
- agentes baseados em crenças-desejos-intenções: a tomada de decisão depende da manipulação de estruturas que representam as crenças, desejos e intenções do agente;

- agente em camadas: onde a tomada de decisão é feita através de várias camadas de software, sendo que cada uma é mais ou menos raciocínio explícito sobre o ambiente em diferentes níveis de abstração.

Bittencourt (2006) descreve outra classificação, dividindo os agentes em reativos e cognitivos, onde os reativos seguem a definição adotada por Weiss (1999) e os cognitivos, são baseados em organizações sociais humanas como grupos, hierarquias e mercados, possuindo uma representação explícita do ambiente e dos outros agentes, dispondo de memória e capacidade de planejar ações futuras e a comunicação entre si.

A complexidade da tomada de decisão dos agentes pode ser afetada pela complexidade das variáveis do ambiente. Russell (2004) sugeriram a seguinte classificação para os ambientes:

- acessível e inacessível: acessível é o ambiente onde o agente pode obter informações completas, precisas e atualizadas sobre o estado do ambiente;
- determinísticos e não determinísticos: um ambiente determinístico é aquele em que qualquer ação tem um único efeito garantido, não há incerteza sobre a situação que resultará da realização de uma ação;
- episódico e não-episódico: em um ambiente episódico, o desempenho de um agente é dependente de uma série de episódios discretos, sem qualquer vínculo entre o desempenho de um agente em diferentes cenários;
- estático e dinâmico: o estático é o ambiente que ele permanecerá inalterado, exceto pelas ações por parte do mandatário e um ambiente dinâmico, ele altere de acordo com a interação dos agentes e dos outros processos que operam sobre ele;
- discreto e contínuo: a distinção entre discreto e contínuo pode se aplicar ao estado do ambiente, ao modo como o tempo é tratado e as percepções e ações do agente.

3.1 Sistemas multiagentes

Ambientes de aplicações reais costumam ser compostos de um grande número de subproblemas que, por sua vez, podem ainda ser divididos em subproblemas, a fim de facilitar a análise e a implementação de soluções. Os Sistemas Multiagentes (SMA) são uma das estratégias da Inteligência Artificial Distribuída (IAD) que se mostram adequada para manipular problemas complexos. Este enfoque baseia-se no estudo e desenvolvimento de agentes autônomos em um

universo multiagente. Para os SMA, o termo autônomo designa o fato de que os agentes têm uma existência própria, independente da existência de outros agentes. Usualmente, cada agente possui um conjunto de capacidades comportamentais que definem sua competência, um conjunto de objetivos, e a autonomia necessária para utilizar suas capacidades comportamentais a fim de alcançar seus objetivos. Um agente é uma entidade computacional com um comportamento autônomo que lhe permite decidir suas próprias ações (Wooldridge 2002).

Sob o ponto de vista da constituição de uma Sociedade de Agentes, Silveira (2001) cita que a fundamentação dos SMA é baseada na interação social de indivíduos que convivem entre si e interagem mutuamente para alcançar objetivos comuns e individuais. Para tal, um agente é concebido como um indivíduo autônomo, com capacidades que lhes são inerentes para o desempenho de suas funções e o alcance dos seus objetivos.

Os SMA permitem modelar o comportamento de um conjunto de entidades inteligentes e organizadas de acordo com leis do tipo social. Estas entidades, ou agentes, dispõem de certa autonomia e estão imersos num ambiente com o qual necessitam interagir, pelo que devem possuir uma representação parcial deste ambiente e meios de percepção e comunicação (XIN, 2000).

Vlassis (2007) cita como benefícios do uso de SMA:

- velocidade e eficiência: devido à computação assíncrona e paralela;
- robustez e confiabilidade: no sentido de que todo o sistema pode sofrer uma pequena degradação quando um ou mais agentes fracassar, porém não fará com que o sistema pare ou deixe de funcionar;
- escalabilidade e flexibilidade: facilidade de adicionar novos agentes ao sistema;
- custo: admitindo-se que um agente é uma pequena parte em comparação com todo o sistema e, portanto, dar manutenção em um agente não fará com que todo o sistema seja parado e reiniciado;
- desenvolvimento e reutilização: uma vez que é mais fácil de desenvolver e manter um sistema modular ao invés de um sistema monolítico.

Os SMA podem ser subdivididos em duas abordagens principais: os denominados Sistemas Multiagentes Cognitivos, cuja característica principal é a existência de uma forma explícita de representação de conhecimento, e os Sistemas Multiagentes Reativos, cuja ênfase principal é no comportamento, sem uma preocupação maior com a representação do conhecimento.

Segundo Sacile (2005) agentes em um SMA podem decidir cooperativamente para atingir objetivos globais, onde geralmente atingem soluções melhores que um sistema centralizado com igual poder computacional. O uso de um SMA é justificado quando uma decisão distribuída é necessária.

A estrutura típica de um SMA é apresentada na Figura 4, onde podem ser vistos os agentes, a formação de relacionamentos em grupos de agentes e a interação entre eles.

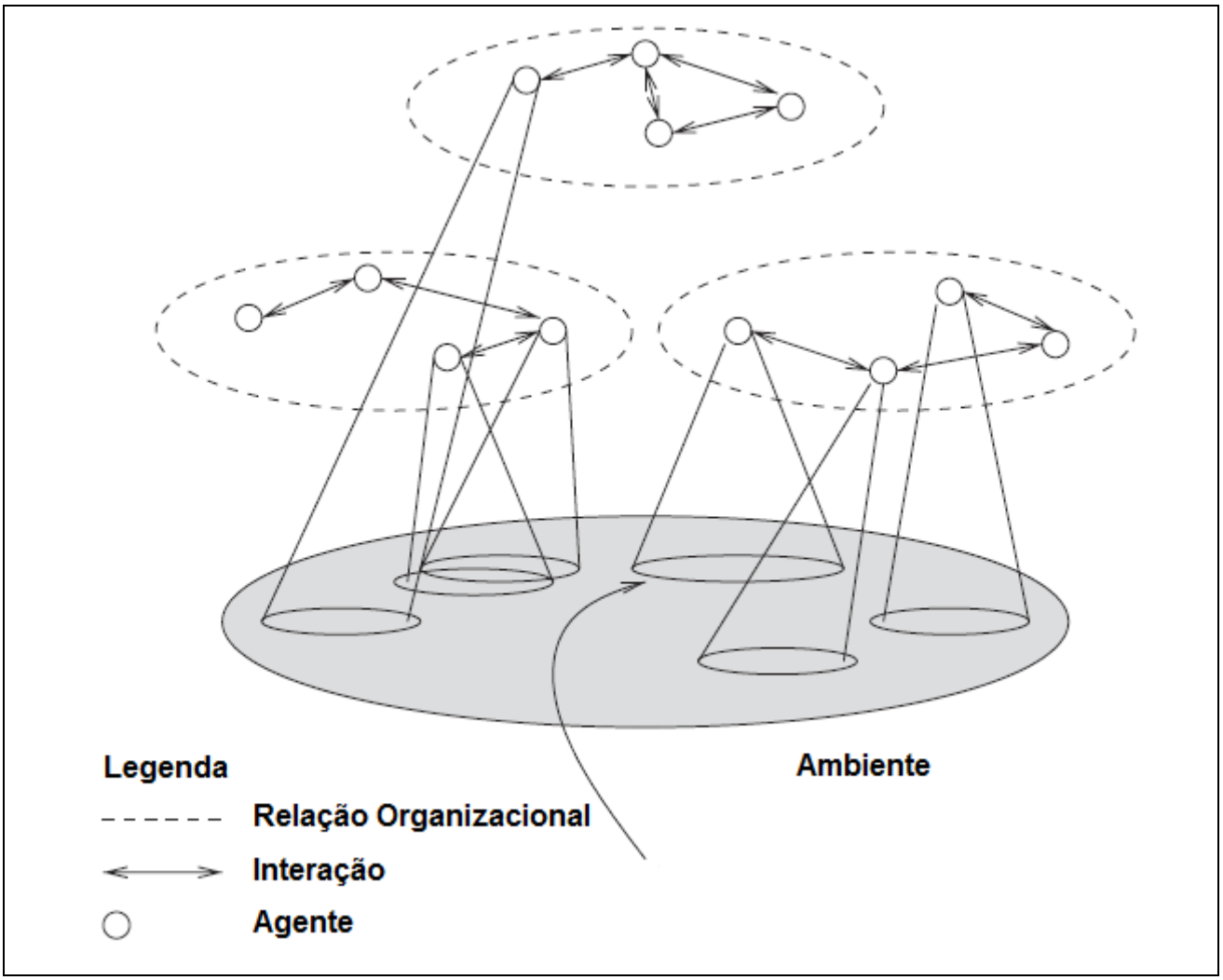

Figura 4. Estrutura típica de um SMA
Fonte: Adaptado de Bordini (2007)

A comunicação forma a base da cooperação e é constituída pelo transporte de mensagens entre os agentes através da rede, pelos protocolos de comunicação e pelos métodos de comunicação resultantes, por sua vez, os protocolos e as estratégias de cooperação, são construídos sobre os

métodos de comunicação e são de fundamental importância na definição de: quem, quando e como devem atuar os agentes na solução do problema (Silveira, 2001).

3.1.1 Coordenação

A coordenação é o processo no qual os agentes são organizados e empenhados em ajudar a garantir que uma comunidade de agentes individuais aja de uma forma coerente (Bellifemine, 2007). Segundo Silveira (2001), a coordenação é de vital importância a coordenação do comportamento dos agentes e da maneira pela qual eles compartilham seus conhecimentos, objetivos, habilidades e seus planos para, em conjunto, tomar as ações necessárias para solucionar um problema. Para que diferentes agentes autônomos possam cooperar mutuamente a fim de atingirem seus objetivos é necessário que a sociedade possua organização e comunicação. A organização diz respeito à natureza e à função da sociedade e de seus elementos constituintes e a comunicação é o principal instrumento que os agentes utilizam para desenvolver a coordenação de suas ações.

Bellifemine (2007) cita alguns motivos pelos quais os agentes devem ser coordenados, tais como: objetivos dos agentes podem gerar conflitos entre as ações que os tomam, os agentes podem ter diferentes conhecimentos e capacidades, objetivos são independentes e as metas podem ser alcançadas em tempo menor, se puderem ser trabalhadas por vários agentes ao mesmo tempo.

A coordenação de agentes pode ser abordada de várias formas, tais como: estruturação organizacional, contratação, planejamento e negociação.

A estruturação organizacional determina um *framework* de atividades e interação através da definição de regras, caminhos de comunicação e relacionamentos. Esta abordagem determina a criação de uma estrutura hierárquica, onde um exemplo dessa organização seria um sistema cliente-servidor, onde o servidor recolhe informações dos clientes e determina a tarefa que cada um irá executar. Segundo Bellefemine (2007) esta abordagem é difícil de implementar e foge a natureza de um SMA, que é a descentralização e autonomia dos agentes.

A abordagem baseada em contratação, definida por Smith (1980) através do Contract Net Protocol, é baseada em uma estrutura descentralizada, onde os agentes podem assumir dois papéis: gerente ou contratado. A premissa básica desta forma de coordenação é que se um agente não pode resolver um problema, utilizando recursos locais ou de especialização, irá decompor o problema em

subproblemas e tentar encontrar outros agentes com os recursos necessários e conhecimento para resolver estes subproblemas.

Outra abordagem consiste em ver o problema de coordenação dos agentes como um problema de planejamento, onde os agentes criam um plano multiagente com todos os detalhes de ações e interações futuras, necessárias para atingir seus objetivos. O plano é continuamente planejado e revisado.

A coordenação de agentes através de negociação, segundo Bellifemine (2007) é a mais utilizada em ambientes comerciais, pois através da comunicação de um grupo de agentes, busca alcançar uma solução mutuamente aceita. A negociação pode ser competitiva ou cooperativa, dependendo do comportamento dos agentes envolvidos. A negociação competitiva é utilizada nas situações onde os agentes possuindo seus objetivos, buscam suas metas através da interação uns com os outros, a priori, não estão dispostos a compartilhar informação, enquanto que a negociação cooperativa é utilizada em situações em que os agentes têm um objetivo em comum a atingir ou uma única tarefa a executar, neste caso, o SMA foi concebido centralmente a buscar um único objetivo global.

3.1.2 Comunicação

Segundo Bellifemine (2007), um dos componentes chaves de uma SMA é a comunicação. É necessário que os agentes tenham a capacidade de se comunicar com usuários, com recursos do sistema e também, que eles possam cooperar, colaborar e negociar com outros agentes.

Os agentes interagem com outros agentes utilizando uma linguagem formalizada, ou seja, que é compreendida pelos agentes independente da implementação dos mesmos. Está linguagem é chamada de *Agent Comunication Language* (ACL), que especifica a separação entre o conteúdo e os atos comunicativos da linguagem.

3.2 Linguagens de Comunicação de Agentes

A primeira ACL (*Agent Communication Language*) desenvolvida foi a KQML (*Knowledge Query and Manipulation Language*), na década de 90, pelo projeto ARPA do governo americano. KQML é uma linguagem de comunicação de agentes que se baseiam em atos de fala (Wooldridge,

2002), definindo um padrão de mensagem e um conjunto de performativas, tais como: *content*, *sender*, *receiver*, *language*, *ontology*, *reply-with*, *in-reply-to*, etc. Possuindo uma padronização, possibilitou a formação de diálogos entre os agentes.

Atualmente a linguagem de comunicação de agente mais estudada é a FIPA (*Foundation for Intelligent Physical Agents*) ACL (FIPA, 2009), que segundo Bellifemine (2007) incorpora muitos aspectos da KQML. As principais características da FIPA ACL são a possibilidade de utilizar diferentes linguagens de conteúdo e a gestão de interação através de protocolos predefinidos.

3.2.1 Padrão FIPA

A organização FIPA visa promover através de especificações a interoperabilidade entre agentes heterogêneos e os serviços que eles podem representar (FIPA, 2009). Propõem uma arquitetura básica para agentes inteligentes e uma linguagem de comunicação de agentes, denominada ACL FIPA.

O conjunto completo de especificações desenvolvidas pela FIPA, incluindo os que ainda não foram normalizados, podem ser vistos através de categorias: comunicação de agentes, transporte de mensagens, gestão de agentes, arquitetura e aplicação. Destas categorias, a comunicação se destaca como sendo o centro do modelo de SMA FIPA (FIPA, 2009).

A organização destas categorias pode ser melhor observada na Figura 5.

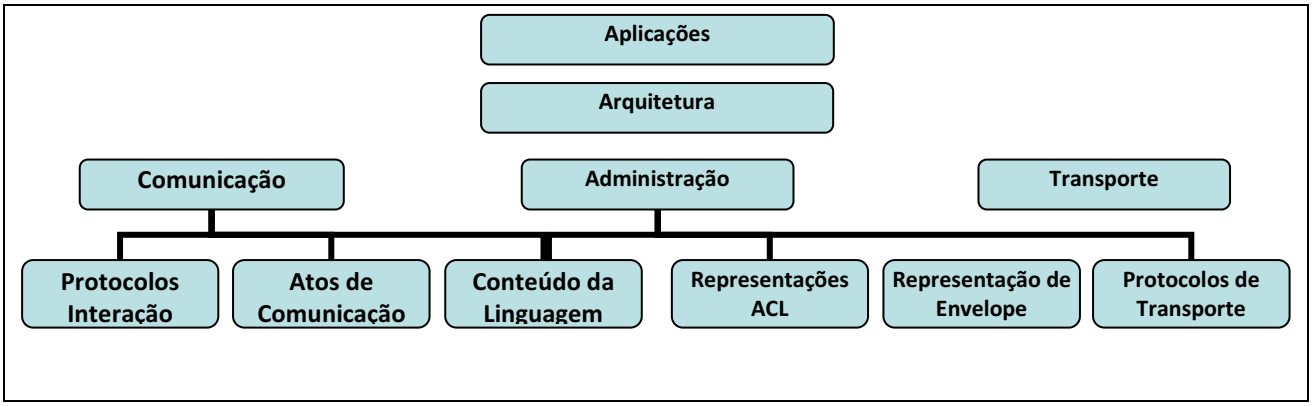

Figura 5. Categorias FIPA
Fonte: Adaptado de FIPA (2009)

As especificações FIPA propõem uma arquitetura básica para agentes inteligentes e uma linguagem de comunicação de agentes, chamada ACL FIPA, cuja semântica é definida por uma linguagem formal chamada FIPA SL (*Semantic Language*). Além da FIPA SL, há a KIF (*Knowledge Interchange Format*), CCL (*Constraint Choice Language*) e RDF (*Resource Description Framework*).

A especificação FIPA ACL *Message Structure Specification* - 00061, uma mensagem FIPA deve obrigatoriamente conter o campo ato performativo (*performative*), porém é esperado que os campos agente emissor (*sender*), agente receptor (*receiver*) e conteúdo da mensagem (*content*) sejam informados. Os demais campos são necessários dependendo da mensagem trocada. A Tabela 2 apresenta a lista de parâmetros das mensagens FIPA-ACL (FIPA, 2009).

Tabela 2. Parâmetros das mensagens FIPA-ACL

Classificação	**Parâmetro**	**Descrição**
Tipo de ato comunicativo	*Performative*	Denota o tipo de ato comunicativo da mensagem.
Participantes da comunicação	*Sender*	É o emissor da mensagem.
	Receiver	É o receptor da mensagem.
	Reply-to	Indica que as próximas mensagens dessa conversação deverão ser enviadas para o agente identificado pelo o parâmetro *reply-to*.
Conteúdo de mensagem	*Content*	É o conteúdo ou conhecimento transportado pela mensagem.
Descrição do conteúdo	*Language*	É a linguagem na qual o conhecimento está expresso.
	Encode	Aponta a codificação utilizada na expressão da linguagem de conteúdo.
	Ontology	É a ontologia utilizada para dar significado à expressão de conteúdo.
Controle de conversação	*Protocol*	É o protocolo de interação utilizado para essa mensagem pelo agente emissor.
	Conversation-id	Introduz uma expressão que será usada para identificar a conversação em andamento.
	Reply-with	Introduz uma expressão que será usada pelo agente que responderá a essa mensagem para identificá-la.
	In-reply-to	É a expressão que referencia a mensagem à qual se está respondendo.
	Reply-by	Explicita um tempo máximo durante o qual o agente emissor estará esperando por uma resposta a esta mensagem.

Fonte: Gluz e Viccari (2003)

Os atos performativos FIPA são responsáveis por determinar qual a ação que a mensagem leva ao agente e seguem a especificação FIPA Communicative Act Library Specification – 00037 (FIPA, 2009).

A ordem com que as mensagens são trocadas e os momentos que ocorrem são definidos através dos protocolos de interação, que para o padrão FIPA são onze conforme apresentado na Tabela 3.

Tabela 3. Protocolos de interação FIPA-ACL

Identificador	**Título**
SC00026	FIPA Request Interaction Protocol Specification
SC00027	FIPA Query Interaction Protocol Specification
SC00028	FIPA Request When Interaction Protocol Specification
SC00029	FIPA Contract Net Interaction Protocol Specification
SC00030	FIPA Iterated Contract Net Interaction Protocol Specification
XC00031	FIPA English Auction Interaction Protocol Specification
XC00032	FIPA Dutch Auction Interaction Protocol Specification
SC00033	FIPA Brokering Interaction Protocol Specification
SC00034	FIPA Recruiting Interaction Protocol Specification
SC00035	FIPA Subscribe Interaction Protocol Specification
SC00036	FIPA Propose Interaction Protocol Specification

Fonte: FIPA (2009).

Protocolos de interação de mensagens FIPA, como o Contract Net, especificado pela FIPA-ACL SC00029, estão implementados em plataformas de desenvolvimento de ambientes multiagentes, chamados FIPA *compliants*, como o JADE (*Jade Agent Development Framework*) e FIPA-OS.

3.3 Modelagem de SMA

A metodologia essencialmente serve como um guia, fornecendo como uma orientação para o desenvolvedor do SMA e ao mesmo tempo dá a oportunidade de adicionar ou remover componentes, tal como pretendido, com base no domínio do problema específico (Nikraz, 2006).

Existem várias metodologias para a modelagem de sistemas multiagentes conhecidas por *Agent Oriented Software Engineering* (AOSE) e outras que utilizam por base, a metodologia para desenvolvimento de software orientado a objetos, respeitando as particularidades da teoria de agentes inteligentes.

3.3.1 Engenharia de Software para SMAs

Sacile (2005) propôs uma classificação das engenharias AOSE em dois grupos: Alto Nível e projeto e linguagens, que pode ser visualizada através da Tabela 4.

Tabela 4. Classificação das AOSE

Metodologias de Alto Nível	Metodologias de projeto e linguagens
GAIA	Orientadas na UML: AIP, AUML, PASSI, …
MaSE	Desenvolvimento de Padrões (Design Patterns)
AOR	Componentes
	Teoria dos Grafos

Fonte: Adaptado de Sacile (2005)

Cada metodologia apóia mais ou menos cada uma das etapas da modelagem e desenvolvimento da aplicação multiagente, não tendo uma que englobe todas, muito menos que seja considerada na literatura como completa, porém a utilização de mais de uma como apoio para modelagem é uma prática que traz bons resultados.

Segundo Sacile (2005) dentre as AOSE de alto nível, as metodologias GAIA e MaSE, são as mais promissoras. Nesta pesquisa será dado ênfase nas metodologias de alto nível, citando algumas e focando na metodologia MaSE escolhida para a modelagem da aplicação multiagente resultante do uso do *framework* HIPS, não esquecendo as demais metodologias AOSE descritas por Wooldridge (2000), Sacile (2005), Nikraz (2006), Deloach (2006) como: AAII, Tropos, Prometheus, ROADMAP, AALAADIN, Cassiopéia, DESIRE, etc.

3.3.1.1 GAIA

Proposta por Wooldridge (2000), a metodologia GAIA, propõe uma abordagem orientada a papéis para o projeto de sistemas multiagentes. Após a identificação dos principais papéis do sistema, um modelo detalhado de papéis é construído. A metodologia GAIA requer que o relacionamento entre agentes na aplicação modelada seja estático em tempo de execução (Wooldridge, 2000).

A metodologia GAIA especifica um alto nível (sociedade de agentes) e um baixo nível (arquitetura de agentes). A sociedade de agentes é vista como uma organização computacional de vários papéis interagindo entre si.

O processo de análise da metodologia GAIA inicia pela definição de papéis no sistema e continua pela modelagem das interações entre si. Os papéis possuem quatro (4) atributos: responsabilidades (podem ser de sobrevivência e segurança), permissões, atividades e protocolos.

No processo de modelagem, o primeiro passo é mapear os papéis nos tipos de agentes e criar certo número de instâncias de cada tipo de agente. O segundo passo é determinar o modelo de serviço necessário para atribuir o papel em um ou todos os agentes. O último passo é criar o modelo de relações (familiaridades) para representação da comunicação entre os agentes.

A Figura 6 apresenta um diagrama das etapas da metodologia GAIA.

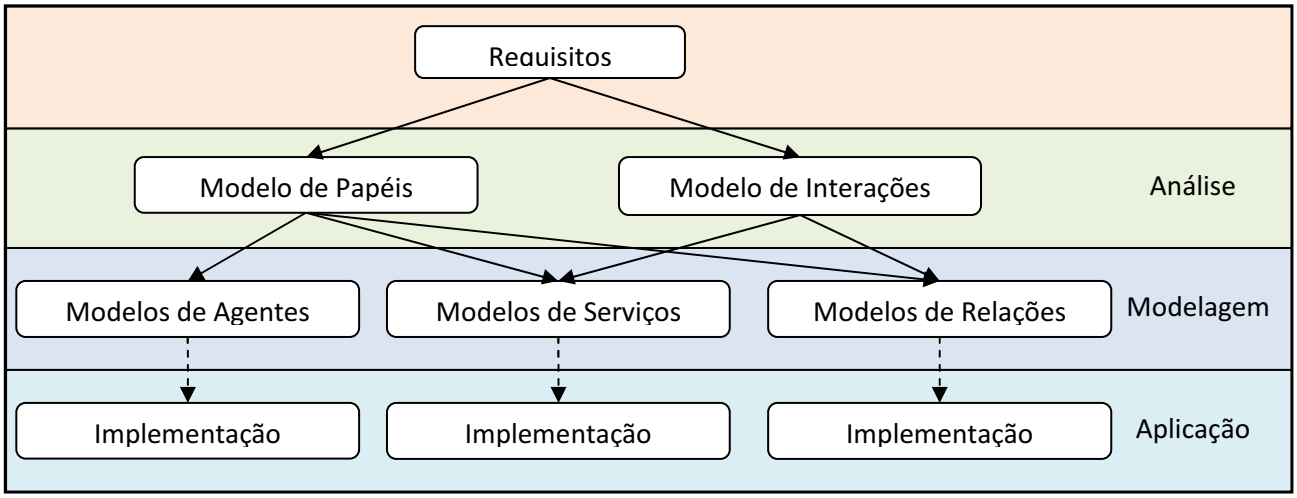

Figura 6. Metodologia GAIA
Fonte: Adaptado de Wooldridge (2000).

Os principais conceitos da metodologia GAIA, apresentados na Tabela 5, estão divididos em duas categorias:

- abstratos: utilizados durante a fase de análise para conceitualizar o sistema;
- concretos: utilizados na fase de projeto, correspondem a elementos que estarão presentes na implementação.

Tabela 5. Conceitos metodologia GAIA

Abstratos	Concretos
Papéis	Tipos de Agentes
Permissões	Serviços
Atividades	Conhecimentos
Protocolos	
Responsabilidades	
Propriedades de Segurança	
Propriedades de Sobrevivência	

3.3.1.2 MaSE

A metodologia MaSE (*Multi-agent Software Engineering*), proposta por Deloach(2001) é adotada por possibilitar a modelagem de sistemas multiagentes independente de: arquitetura do SMA, arquitetura dos agentes, linguagem de programação ou protocolo de comunicação.

Segundo Sacile (2005), o objetivo da metodologia MaSE é orientar o projetista do sistema multiagente da fase inicial de especificação até a implementação. MaSE melhora a GAIA pois permite geração de código automático através de uma ferramenta específica (AgentTool). Sacile (2005) afirma que o objetivo da metodologia MaSE é coordenar o analista desde a especificação inicial do sistema até a implementação do sistema multiagente.

A metodologia MaSE define sete etapas, divididas nas fases de análise e projeto, onde a fase de análise foi subdividida em três etapas e a fase de projeto, outras quatro etapas.

A fase de análise, segundo Deloach (2006) tem o objetivo de definir o conjunto de papéis que serão necessários para atingir os objetivos do sistema multiagente. Suas etapas são:

- levantamento de objetivos: busca extrair das necessidades os objetivos do sistema multiagente;
- aplicação de casos de uso: nesta etapa os objetivos são traduzidos em casos de uso. Os casos de uso representam o comportamento pretendido pelo sistema e a seqüência de eventos;
- refinamento de papéis: organiza os papéis em um modelo de papéis, que descreve os papéis presentes no sistema multiagente e o relacionamento entre eles.
- O objetivo da fase de projeto é pegar os papéis e as tarefas e converter em uma forma propícia para a execução, ou seja, em agentes e diálogos (Deloach, 2006). Suas etapas são:

- criação de classes de agentes: identifica as classes de agentes e seus diálogos, documentando através do diagrama de classes de agentes;
- construção dos diálogos: detalha os diálogos entre os agentes, através de um diagrama de comunicação utilizando um par finito de estados;
- montagem das classes de agentes: defini a arquitetura interna dos agentes. O diagrama de arquitetura de agentes é similar ao diagrama de classes da UML;
- projeto do sistema: define a configuração do sistema multiagente, especificando quantos agentes de cada tipo serão instanciados através do diagrama de desenvolvimento, similar ao diagrama de desenvolvimento da UML.

A representação gráfica da metodologia MaSE pode ser visualizada na Figura 7.

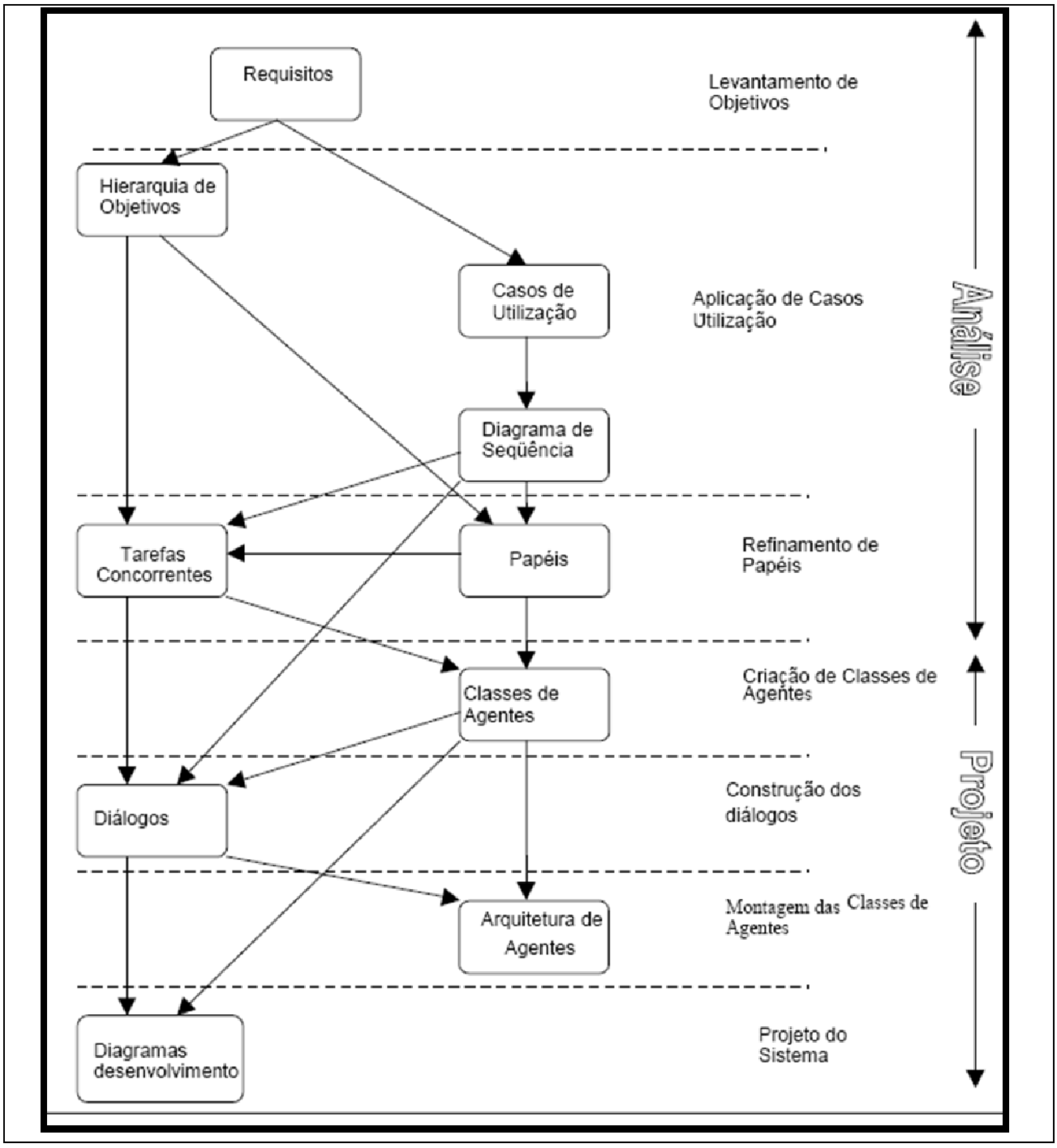

Figura 7. Metodologia MaSE

Fonte: Adaptado de Deloach (2001).

Deloach (2006) descreve a metodologia MaSE e propõem a metodologia O-MaSE, que é um complemento a metodologia MaSE. Neste complemento a fase de projeto é dividida em nível alto e baixo, adiciona um modelo da organização do sistema multiagente e recursos para modelagem da comunicação dos agentes com o próprio ambiente multiagente. Utilizamos a metodologia MaSE, pois avaliando as mudanças propostas, atingimos os objetivos da pesquisa sem

a necessidade deste complemento, porém adicionamos as recomendações para futuras pesquisas, uma avaliação da metodologia O-MaSE.

3.4 Ambientes para desenvolvimento de SMA

Os ambientes para desenvolvimento de sistemas multiagentes são ferramentas que buscam atender as necessidades projetadas pelas metodologias, no desenvolvimento de programas voltados à teoria de agentes.

Não há uma padronização no desenvolvimento, podendo ser feito em qualquer linguagem de programação ou ambiente que atenda as necessidades, porém há ferramentas que vão desde plataformas e *frameworks* para o desenvolvimento de SMA a linguagens de programação, que auxiliam na transformação do modelo projetado em uma aplicação, possibilitando a execução, testes, acompanhamento de diálogos e simulação, até ter a aplicação final depurada.

Há várias ferramentas que realizam esta função intermediária de programação, facilitando a tarefa do desenvolvimento de sistemas multiagentes, fazendo com que o analista, concentre sua atenção na modelagem e especificação final dos agentes, abstraindo o meio de comunicação, constituição do ambiente e forma de monitoração e se concentrando nos diálogos, na estrutura do agente, na interação entre eles e principalmente, na busca dos objetivos.

Silveira (2001) cita diversas ferramentas deste tipo, mencionando o trabalho de Baker (1997), que apresenta a ferramenta JAFMAS e outras como: Aglets Workbench (IBM), Concórdia, Odissey, Voyager, JATLite, InteRRaP, DMARS, Agent Talk, Telescript, AgentTCL, Swarm, Echelon, Agent Builder, etc.

Vrba (2003) cita as ferramentas JADE (CSELT), FIPA-OS (Emorphia), ZEUS (British Telecom), JACK (Agent Oriented Software), GRASSHOPPER 2(IKV++Technologies AG), ADK (Tryllian), JAS (Fujitsu,HP, IBM, Sun), AgentBuilder (IntelliOneTechnologies), MadKit (MadKit Team), ComtexAgent Plataform (Communication Technologies), Bee-gent (Toshiba) e Aglets (IBM Japan) e faz uma análise mais específica sobre desempenho, entre as plataformas JADE, FIPA-OS, ZEUS e JACK.

Destacam-se pelas referências, as ferramentas: ZEUS (Vrba, 2003 e Bigus, 2001), FIPA-OS (FIPA, 2009) e a plataforma JADE (Bellifemine, 2007 e Vrba, 2003) que será descrita em detalhe.

ZEUS é um ambiente proposto pela *British Telecom*, utiliza o padrão KQML e é composto por cinco camadas: interface, definição, organização e coordenação e comunicação. Possui três componentes principais: biblioteca de componentes de agentes (em Java), ferramentas para construção de agentes e ferramentas para visualização (Bigus, 2001).

O *framework* FIPA-OS, desenvolvido em Java, disponibiliza um conjunto de classes e um ambiente de execução com todos os recursos necessários para a comunicação entre os agentes, que utiliza o padrão de mensagem FIPA-ACL (Bigus, 2001).

3.4.1 JADE

JADE (*Java Agent Development Framework*) (JADE, 2009) é um *framework* para desenvolvimento de aplicações baseadas em agentes em conformidade com as especificações FIPA (Foundation for Intelligent Physical Agents) para sistemas multiagentes. Segundo Bellifemine (2007) JADE teve seu desenvolvimento iniciado em 1998, pela Telecom Itália, motivada pela validação das então novas especificações feitas pela FIPA, que é uma organização com objetivo de promover a tecnologia baseada na teoria de agentes e a interoperabilidade de seus padrões com outras tecnologias. A primeira distribuição de código aberto ocorreu em 2000.

O principal objetivo do *framework* JADE é simplificar o desenvolvimento de aplicações baseadas na teoria de agentes, garantindo ao mesmo tempo o cumprimento das normas especificadas, através de um conjunto de serviços e agentes, definidos pelas especificações FIPA, segundo JADE (2009): serviço de nomes (*name service*), serviço de páginas amarelas (*yellow pages service*), transporte de mensagens, serviço de codificação e decodificação de mensagens (*parsing service*) e uma biblioteca de protocolos de interação.

A plataforma JADE conforme a especificação FIPA, inclui todos os componentes obrigatórios para a administração da plataforma, tais como: *Agent Communication Channel* (ACC), *Agent Management System* (AMS) e o *Directory Facilitator* (DF). Toda a comunicação é feita através de mensagens, onde o padrão de mensagens FIPA ACL é utilizado para representá-las (JADE, 2009). Podem ser utilizadas ontologias associadas a essas mensagens para auxiliar na compreensão e validação das mensagens.

A plataforma de agentes (AP) JADE disponibiliza a infra-estrutura física onde os agentes serão executados, consistindo de servidor, sistema operacional, componentes FIPA e *softwares* adicionais.

O Agente JADE é um processo que habita a AP e disponibiliza alguns serviços padrões disponibilizados pelo *framework* JADE, seguindo a especificação FIPA *Agent Identifier* (AID), porém na aplicação, podem possuir outros, conforme necessidade do objetivo a ser atingido.

O ACC também conhecido como *Message Transport System* (MTS) é o agente responsável por prover toda a comunicação entre os agentes da plataforma. Todos os agentes incluindo os da própria plataforma (AMS, DF, etc) usam esse canal para a comunicação.

O sistema gerenciador de agentes AMS é o agente que gerencia a plataforma, o acesso a ela e sua utilização. São responsabilidades do AMS a criação e finalização dos agentes, a migração entre AP trocando de servidores, o registro dos agentes no DF e sua atualização de status, etc.

O DF é um agente que disponibiliza o serviço de páginas amarelas. Este serviço consiste em disponibilizar uma lista de agentes atualizada para que os agentes em execução acessem e possam saber da existência dos demais agentes, status, tipos, etc.

A plataforma JADE pode ser distribuída por vários servidores, onde em cada servidor, deverá haver uma máquina virtual Java ou *Java Virtual Machine* (JVM) sendo executada. Cada JVM atuará basicamente como sendo um container para os agentes e servindo como um ambiente para execução, podendo haver vários agentes executando concorrentemente na mesma JVM. Esta arquitetura pode ser visualizada através da Figura 8.

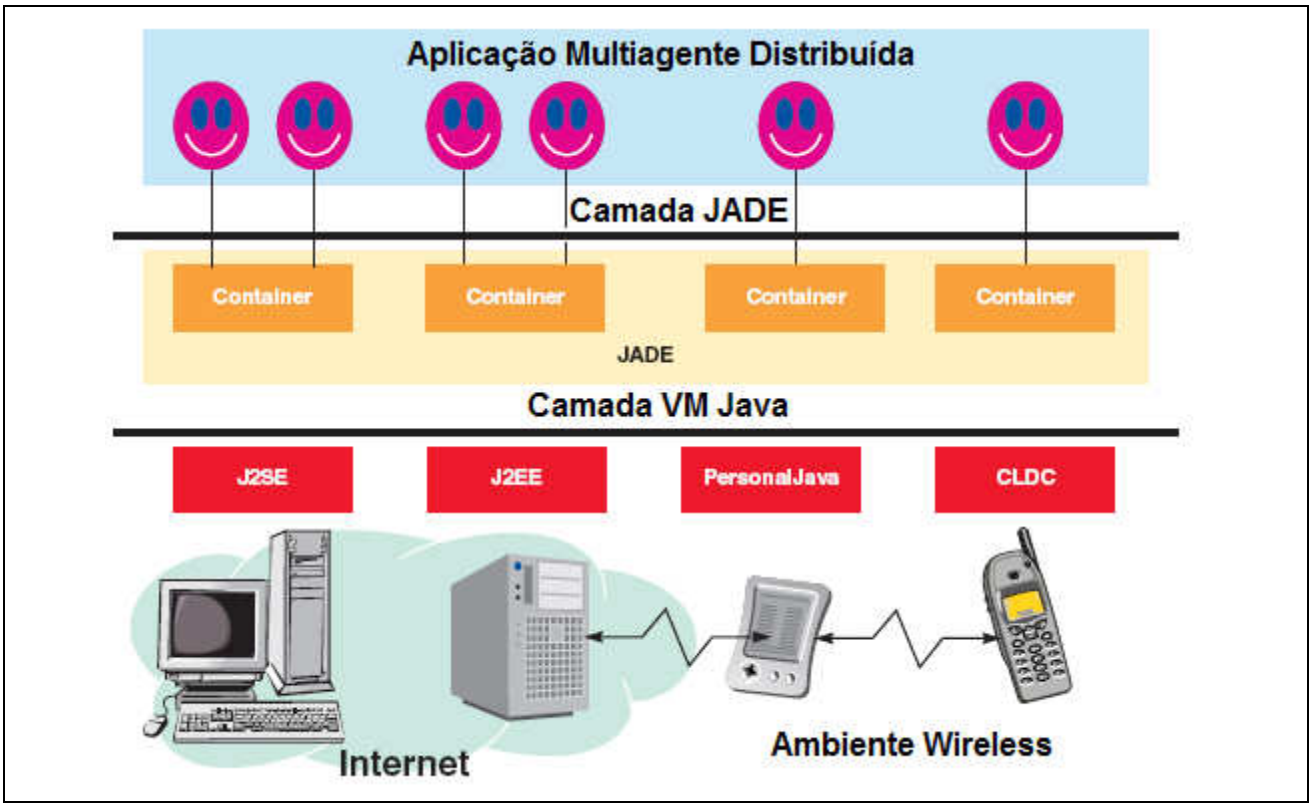

Figura 8. Arquitetura JADE

Fonte: Adaptado de JADE (2009).

A plataforma JADE é composta de agentes containeres que podem ser distribuídos na rede. Os agentes vivem nos containeres, que são processos Java que possuem todos os recursos necessários para receber e hospedar os agentes. Há um container em especial que é o *Main*, que é o ponto de partida da plataforma e todos os demais containeres, se comunicaram com ele ao iniciar, para registro. O container *Main* também é responsável pelo agentes DF e AMS. A interação entre o container *Main* e os demais pode ser visualizada na Figura 9.

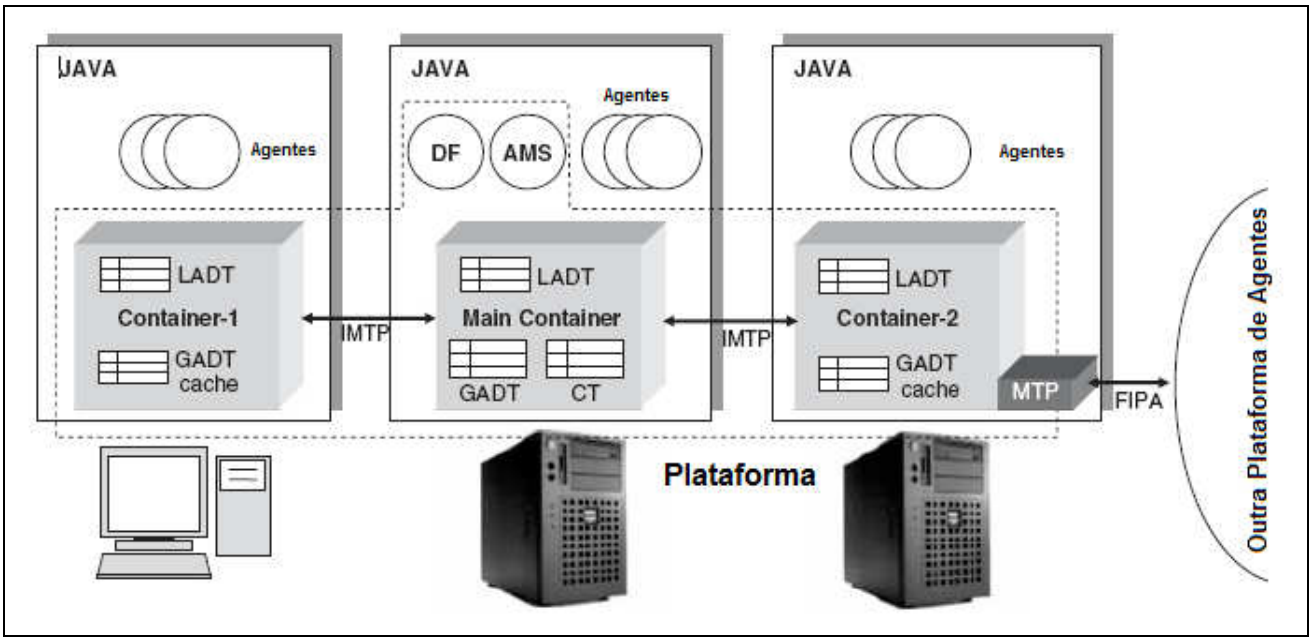

Figura 9. Relação entre containeres JADE
Fonte: JADE (2009).

Com base na avaliação feita por Vrba (2003) e nas pesquisas feitas sobre a plataforma e nas demais, foi adotado a plataforma JADE para a integração com o *framework* HIPS, servindo de base para o pacote de classes JHIPS e aplicação desenvolvida.

3.4.1.1 Ferramentas da plataforma JADE

A plataforma JADE disponibiliza uma interface gráfica para o usuário, ou *Graphical User Interface* (GUI) para administração remota, monitoramente e controle do status dos agentes (JADE, 2009). A interface remota para gerenciamento de agentes ou *Remote Agent Management* (RMA) demonstrada na Figura 10, permite, por exemplo, criar e iniciar a execução de um agente remotamente, providenciando que o container desse agente esteja em execução.

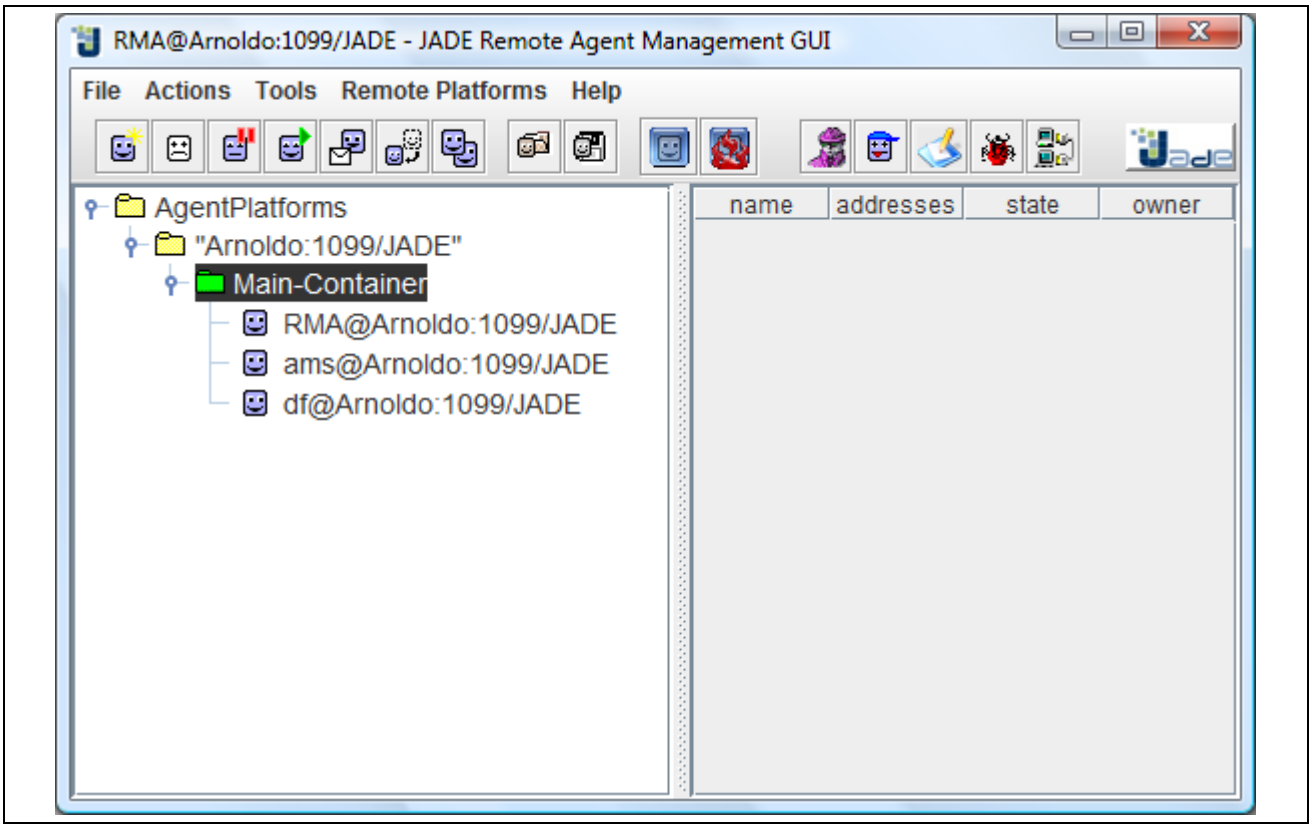

Figura 10. RMA JADE

A partir do RMA podem ser iniciadas outras ferramentas (internamente para a plataforma, cada uma delas é um agente executando) para monitoramento do sistema multiagente, tais como: DF, *Dummy*, *Sniffer*, *Introspector* e *LogManagment*.

A ferramenta DF permite visualizar as características dos agentes registrados, registrar e remover os registros de agentes, modificarem atributos dos agentes existentes e também pesquisar por determinadas características dos agentes. O RMA permite também a associação com outros DF, desta forma, criando uma rede de domínios e subdomínios de páginas amarelas. Qualquer DF, mesmo que não seja pertencente à plataforma JADE, pode ser controlada pela mesma interface gráfica e as mesmas operações básicas: ver, registrar, cancelar, modificar e pesquisar pode ser executado pelo DF remoto (JADE, 2009).

O *DummyAgent* é uma ferramenta para efetuar testes e monitoração, pois permite enviar mensagens ACL para os demais agentes e verificar as respostas, desta forma, validando validar um agente antes de integrar ele ao sistema multiagente.

O *Sniffer é uma ferramenta para* interceptar as mensagens trocadas entre os agentes, permitindo verificar seu conteúdo e a seqüência cronológica em que ocorreram (Bellefemine, 2007) (JADE, 2009).

A ferramenta *Introspector* tem a funcionalidade de depurar o comportamento de um agente, se diferenciando do *Sniffer***,** que monitora as mensagens trocadas na plataforma. A ferramenta permite monitorar o ciclo de vida do agente, as filas de mensagens de entrada e saída, verificando as mensagens trocadas em cada comportamento (Bellefemine, 2007) (JADE, 2009).

O *LogManager* simplifica a gerência dinâmica e distribuída do *Log*, proporcionando uma interface gráfica que permite o registro dos níveis de cada componente da plataforma JADE que pode sofrer alteração durante o tempo de execução.

3.4.2 Ontologias

Segundo Russell (2004), a palavra ontologia representa uma teoria específica sobre a natureza de ser ou existir. A ontologia determina os tipos de itens que existem, mas não determina suas propriedades específicas e seus inter-relacionamentos.

O conceito de ontologia em ciências da computação denota um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes (Nikraz, 2006). Na área de inteligência artificial, é utilizada como uma forma de representação do conhecimento sobre o mundo ou de parte dele. Uma ontologia é utilizada para realizar inferência sobre os objetos do domínio (Weiss, 1999).

Protégé (2009) descreve ontologia como sendo os conceitos e relacionamentos que são importantes em um domínio específico, fornecendo um vocabulário para esse domínio, bem como uma especificação do significado dos termos utilizados no vocabulário.

Uma ontologia é formada principalmente, segundo Protégé (2009) por:

- classes de objetos (conceitos): definem quais são as informações necessárias para sua identificação e classificação dentro de um determinado domínio e são organizados em hierarquia;
- relações: ligações entre conceitos que fogem à hierarquia;

- atributos: são relações com tipos de dados pré-definidos;
- instâncias: são ocorrências concretas de conceitos abstratos;
- axiomas: são regras válidas dentro do domínio modelado.

Há ferramentas específicas para ontologias, que através da interface gráfica possibilitam formar a estrutura da Ontologia, desta forma facilitando a criação, manutenção e reutilização. Uma das ferramentas mais utilizadas na literatura para esse fim e que permitiu através de um *plugin* à integração posterior as bibliotecas que o JADE fornece foi o editor de ontologias Protégé.

3.4.2.1 Editor Protégé

Protégé é uma plataforma de código aberto com um conjunto de ferramentas para construir modelos de domínios e bases de conhecimento para aplicações através de ontologias. Protégé implementa um conjunto de estruturas para modelar o conhecimento e ferramentas para a criação, visualização e manipulação das ontologias em várias formas de representação (Protege, 2009).

As ontologias criadas no editor Protégé podem ser utilizadas pelos agentes JADE através de classes. Essas classes são geradas através de um plugin adicionado ao editor Protégé, chamado *Bean Generator.*

3.4.2.2 Bean Generator

Ontology Bean Generator é um *plugin* que pode ser adicionado como uma aba na interface do editor de ontologias Protégé. Este *plug*in gera classes Java utilizando as classes disponibilizadas pelo *framework* JADE, desta forma, gerando ontologias compatíveis com as especificações FIPA, podendo ser utilizadas por agentes JADE (BEAN GENERATOR, 2009).

4 ALGORITMOS GENÉTICOS

Darwin propôs a teoria da evolução das espécies que possui como idéia central que as variações (mutações) ocorrem na reprodução e serão preservadas em gerações sucessivas, em proporção aproximada a seu efeito sobre a adaptação reprodutiva (Russel, 2004).

De acordo com Linden (2006), na natureza todos os indivíduos dentro de um ecossistema competem entre si por recursos limitados, tais como água e comida. Aqueles dentre os indivíduos de uma mesma espécie que não obtém êxito tendem a ter uma prole menor e esta descendência reduzida faz a probabilidade de ter seus genes propagados ao longo de sucessivas gerações seja menor, processo que é denominado seleção natural. A combinação entre as características dos indivíduos que sobrevivem pode produzir um novo indivíduo muito mais bem adaptado as características de seu meio ambiente ao mesclar características positivas de cada um dos reprodutores. Este processo implica nos descendentes de indivíduos serem variações dos seus pais, podendo ter características positivas e negativas.

Submeter os indivíduos a reprodução constante não fará termos “super-descendentes”, pois a evolução natural não é um processo dirigido, podendo haver casos onde pais fortes e saudáveis possam gerar descendentes fracos e doentes.

A evolução é um processo no qual os seres vivos são alterados por um conjunto de modificações que eles sofrem através dos tempos, podendo ser explicada por alguns fatores como mutação gênica, recombinação gênica, seleção natural e isolamentos (Linden, 2006).

Na década de 60, Holland e outros começaram a estudar os chamados sistemas adaptativos, que foram modelados como sistemas de aprendizagem de máquina. Tais modelos, conhecidos por algoritmos genéticos (AG) (Holland, 1975), implementavam populações de indivíduos contendo um genótipo, formado por cromossomos (representados por cadeias de bits) aos quais se aplicavam operadores de seleção, recombinação e mutação. Havia um quarto operador, o de inversão, porém não chegou a ser amplamente utilizado (Bittencourt, 2006).

Uma sucinta definição de AG seria a proposta por Linden (2006) onde descreve os AG como uma técnica de busca baseada numa metáfora do processo biológico de evolução natural.

Complementando esta definição, Linden (2006) cita que os AG são técnicas heurísticas de otimização global, que se opõe aos métodos como gradiente[3] (*hill climbing*).

Russell (2004) define os AG como sendo uma técnica na qual os estados sucessores são gerados pela combinação de dois estados pais, em vez de serem gerados pela modificação de um único estado.

Os AG são técnicas probabilísticas e não determinísticas, ou seja, com uma mesma população inicial e parâmetros, pode ser encontradas soluções distintas a cada execução.

Quando não é possível aplicar métodos otimizantes, de tempo polinomial, para a resolução dos problemas, devido ao fato de se tratar de problemas da classe NP, recorrer-se ao uso de métodos de aproximação ou heurísticos que, portanto, não garantindo que se encontrem soluções ótimas para a maioria dos problemas, poderão conduzir a soluções consideradas boas e, muitas vezes, soluções próximas da ótima ou sub-ótimas (Varela, 2007).

4.1 Conceitos e Esquema

De acordo com Russell (2004), os AGs ocorrem como a busca em feixe, onde começam com um conjunto de k estados gerados aleatoriamente, chamado de população. Cada estado, ou indivíduo, é representado como uma cadeia sobre o alfabeto finito. A produção da próxima geração de estados é feita através da função de avaliação (*fitness*), que deve representar através de seu resultado, valores altos para estados melhores e valores baixos para estados piores.

O esquema básico de um AG é demonstrado através da Figura 11.

[3] Segue a derivada de uma função de forma a encontrar o máximo de uma função, ficando facilmente retido em máximos locais. Os AG não possuem essa dependência tão forte dos valores iniciais.

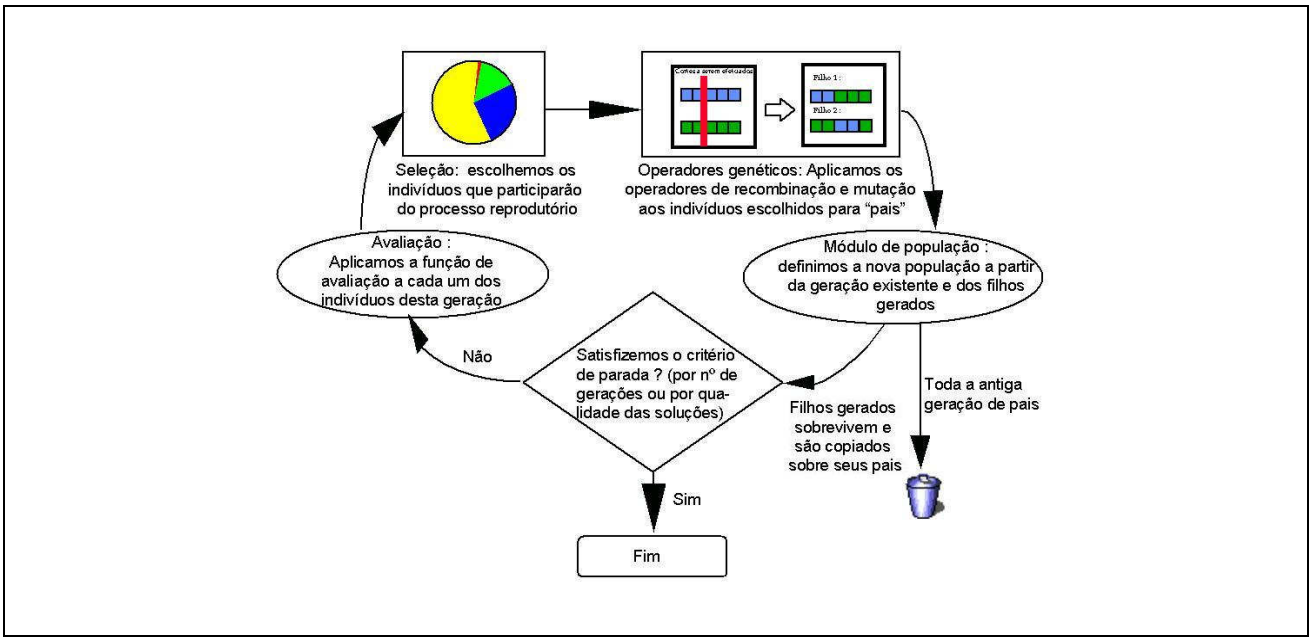

Figura 11. Esquema de funcionamento do Algoritmo Genético
Fonte: Linden (2006).

Para reprodução, são escolhidos aleatoriamente dois indivíduos, onde também neles é escolhido um ponto de corte. Posteriormente são gerados dois novos indivíduos através da troca de elementos entre os dois pais, no ponto de corte, conhecido por *crossover* ou recombinação**.**

O operador de mutação ocorre incidindo sobre cada posição que forma o indivíduo, podendo gerar uma mutação aleatória, com pequena probabilidade de ocorrer.

4.1.1 Função de avaliação

A função de avaliação, de acordo com Linden (2006) é a maneira utilizada pelos AGs para determinar a qualidade de um indivíduo como solução do problema. A função de avaliação só julga a qualidade da solução que está sendo apresentada por aquele indivíduo, sem armazenar qualquer tipo de informação sobre as técnicas de resolução do problema. Isto leva a conclusão de que o mesmo AG pode ser usado para descobrir o máximo de toda e qualquer função de n variáveis, sem nenhuma alteração das estruturas de dados e procedimentos adotados, alterando-se, apenas, a função de avaliação.

4.2 Operador de seleção

O método de seleção de pais deve simular o mecanismo de seleção natural que atua sobre as espécies biológicas, em que os pais mais capazes geram mais filhos, ao mesmo tempo em que os pais menos aptos também podem gerar descendentes. Conseqüentemente é necessário privilegiar os indivíduos com função de avaliação alta, sem desprezar completamente aqueles indivíduos com função de avaliação extremamente baixa. Linden (2006) afirma que esta decisão é razoável, pois até indivíduos com péssima avaliação podem ter características genéticas que sejam favoráveis à criação de um indivíduo que seja a melhor solução para o problema que está sendo abordado. Características estas que podem não estar presentes em nenhum outro cromossomo de nossa população.

Na operação de seleção, se deixar apenas os melhores indivíduos se reproduzirem, a população tenderá a ser composta de indivíduos cada vez mais semelhantes e faltará diversidade a esta população, que segundo Linden (2006) este efeito é denominado convergência genética.

A convergência genética pode ser evitada, ou ao menos amenizada, selecionando-se de forma justa os indivíduos menos aptos da população. Fazendo uma analogia a natureza, os indivíduos mais fracos também geram prole, apesar de fazê-lo com menos freqüência do que os mais aptos.

4.2.1.1 Seleção Aleatória

A seleção aleatória escolhe os indivíduos da população que vão passar pelo processo de reprodução por sorteio, sem levar em consideração a melhor ou pior aptidão resultante da função de avaliação.

4.2.1.2 Método de *Culling*

O método de *Culling* de acordo com Russell (2004), todos os indivíduos abaixo de um determinado limiar são descartados. Com esta medida tende a convergir com maior rapidez que a versão aleatória.

4.2.1.3 Roleta viciada

No método da roleta viciada é criada uma roleta de forma virtual na qual cada cromossomo recebe um intervalo proporcional à sua avaliação. A soma dos intervalos corresponde à totalidade da roleta.

O funcionamento consiste no ato de rodar a roleta, onde o número sorteado pode corresponder a um valor do intervalo de 0 a 100, 0 a 360 ou 0 à somatória de todos os intervalos. O valor sorteado na roleta corresponde a um intervalo, desta forma selecionando o indivíduo.

4.2.2 Operador de *Crossover* ou recombinação

O operador de *crossover* pode ser realizado a partir da escolha de um ou *n* pontos de corte. O ponto de corte constitui uma posição entre dois genes de um cromossomo. Cada indivíduo de n genes contém n-1 pontos de corte, e este ponto de corte é o ponto de separação entre cada um dos genes que compõem o material genético de cada pai (Linden, 2006).

Na Figura 12 abaixo pode ser visto um exemplo de cromossomo, com seus genes e os pontos de corte possíveis.

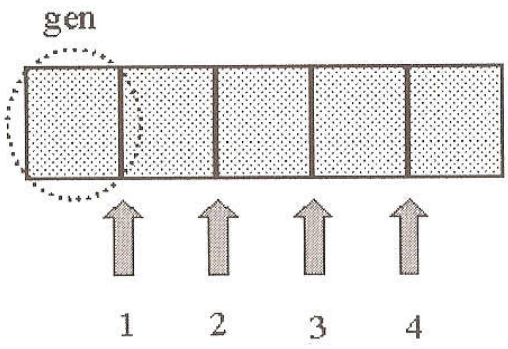

Figura 12. Exemplo de pontos de corte
Fonte: Linden (2006).

4.2.2.1 *Crossover* com Um Ponto

Sorteado o ponto de corte, os pais são separados em duas partes, onde uma delas posteriormente será trocada pela parte do outro pai.

Este processo, de acordo com Linden (2006) é parecido com o que acontece na natureza durante a formação cromossomial de um indivíduo pertencente a uma espécie que adota a reprodução sexuada. A diferença é que na natureza não é adotado apenas um ponto de corte.

4.2.2.2 *Crossover* de Dois Pontos

O operador de *Crossover* pode ser feito também se selecionando mais de um ponto de corte, ou seja, o *crossover* de dois pontos, onde a parte a ser trocada entre os dois indivíduos é a parte compreendida entre os dois pontos de corte.

Na Figura 13 é apresentado o funcionamento do operador de crossover de dois pontos.

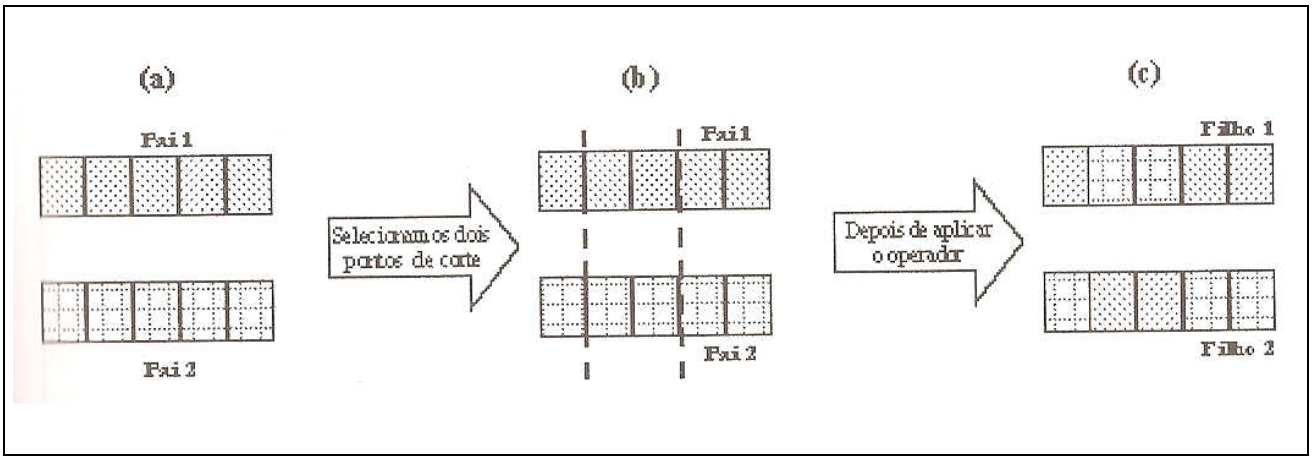

Figura 13. Funcionamento crossover de dois pontos
Fonte: Linden (2006).

A operação de *crossover* de dois pontos é ligeiramente mais complexa do que a operação do seu equivalente de um só ponto, porém a diferença de desempenho conseguida, em geral, faz com que o custo extra seja válido. Linden (2006) afirma que o número de esquemas que podem ser efetivamente transferidos aos descendentes usando-se este operador aumenta de forma considerável, mas o operador de *crossover* conhecido como uniforme é ainda maior.

4.2.2.3 *Crossover* Uniforme

O *crossover* uniforme foi desenvolvido para minimizar o efeito causado pelo *crossover* de dois pontos em separar esquemas de maior comprimento, pois devido ao fato de possuir dois pontos de corte, possuía maior probabilidade de separar esquemas de maior comprimento.

O funcionamento do *crossover* uniforme se baseia no sorteio de um número zero ou um para cada gene do pai. Caso o número sorteado for igual a um, o primeiro filho recebe o gene do primeiro pai e o segundo filho, o gene do segundo pai, se for zero, é invertido essa ordem.

Na Figura 14 é apresentado o funcionamento do operador de *crossover* de dois pontos.

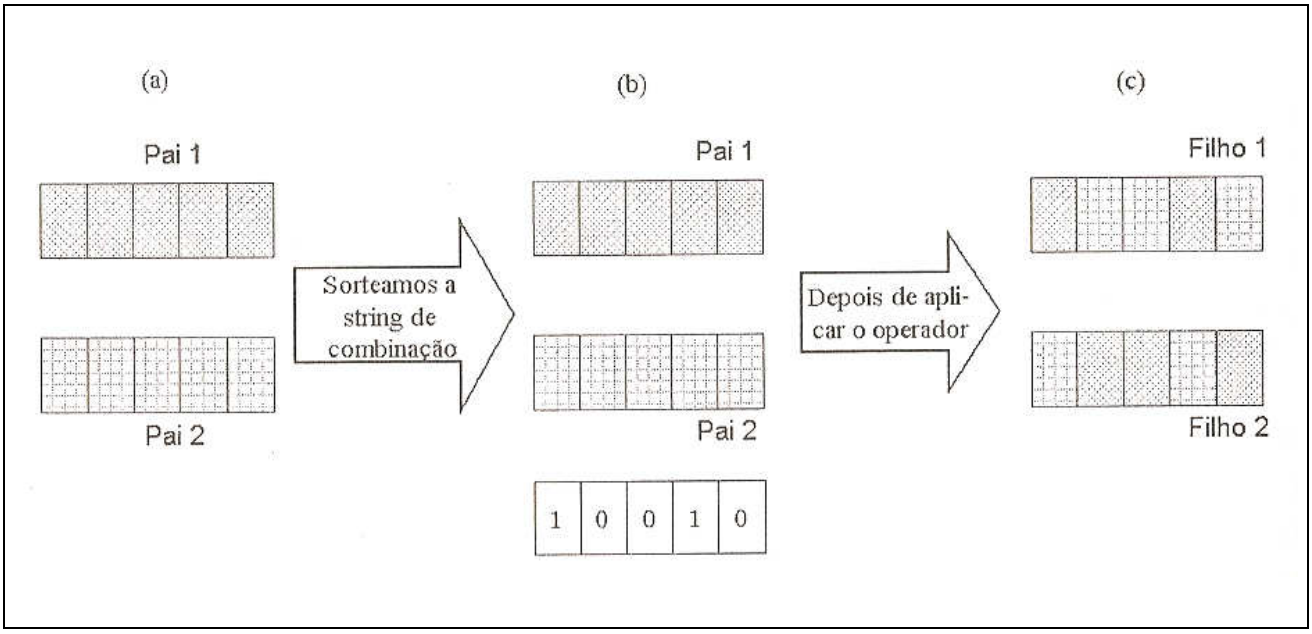

Figura 14. Funcionamento crossover uniforme
Fonte: Linden (2006).

4.2.3 Operador de Mutação

O operador de mutação é responsável pela introdução e manutenção da diversidade genética na população. Ele trabalha alterando arbitrariamente, logo após o cruzamento, um ou mais componentes de uma estrutura escolhida entre a descendência, fornecendo dessa forma meios para a introdução de novos elementos na população. O operador de mutação é aplicado aos indivíduos com uma probabilidade dada por uma taxa de mutação (Holland, 1975).

A taxa de mutação tende a ser um valor baixo, evitando que a cada geração tenha uma renovação completa na população, mas o necessário para que introduza indivíduos com características novos no processo de reprodução.

A Figura 15 demonstra o operador de mutação atuando sobre um indivíduo, após passar pelo operador de *crossover*.

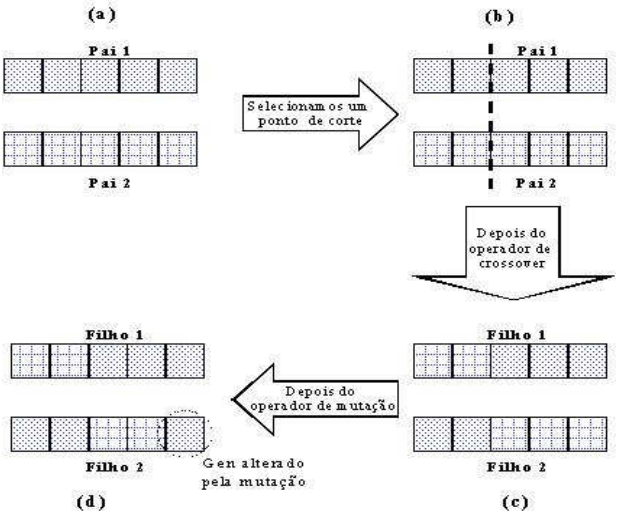

Figura 15. Operador de mutação

Fonte: Linden (2006).

4.3 Características

São características dos AG de acordo com Linden (2006):

- paralelismo: no sentido da população de soluções, onde os indivíduos são analisados simultaneamente, sem criar grupos restritos;
- global: não utilizam apenas informação local, ou seja, não necessariamente ficam presos em máximos locais;
- não totalmente aleatórios: baseia-se na passagem de informação de uma geração para a outra;
- não afetado por descontinuidades: não usam informações de derivadas na sua evolução nem necessitam de informação sobre o seu entorno para poder efetuar sua busca;
- lida com funções discretas e contínuas: pode trabalhar com funções reais, discretas, booleanas e até categóricas (não-numéricas);
- espaço de busca intratável: por ser uma busca direcionada e inteligente, pode trabalhar em situações onde o espaço de busca é grande, muitas vezes não eficiente para outros métodos.

4.4 Limitações

Deve ficar claro, que o uso de AG não busca a solução ótima para o problema, mas sim uma solução satisfatória, de acordo com uma função ou um tempo de processamento estabelecido. Durante o processamento serão encontrados máximos e mínimos locais, próximos ou não do máximo global.

Os algoritmos genéticos, sendo uma técnica de busca e otimização, permitem atingir objetivos obedecendo a uma regra de satisfação e evita a análise de todas as combinações possíveis, sendo desta maneira uma técnica eficiente para aplicações em problemas onde o número de combinações é grande e a análise de todo o conjunto de opções é uma tarefa que demandaria muito tempo de processamento, tornando inviável a sua realização em virtude do tempo de resposta, fundamental para a resolução do problema descrito nesta pesquisa.

5 *FRAMEWORK* HIPS

A proposta *Hybrid Intelligent Process Scheduler* (HIPS) é criar um *framework* que permita modelar ambientes de produção, permitindo o desenvolvimento de soluções para o problema de escalonamento de processos nos recursos produtivos.

O *framework Hybrid Intelligent Process Scheduler* (HIPS) permite modelar problemas de escalonamento de processos em cenários de produção que atendem as especificações de problemas do tipo JSS e também executar e monitorar o sistema multiagentes utilizado.

Neste capítulo, será descrito o *framework*, limitações e tecnologias empregadas no seu desenvolvimento.

5.1 Estrutura

O *framework* HIPS é composto pelo ambiente de modelagem, chamado de Hips *Architect* e por três pacotes de classes que compõem o JHIPS (Java HIPS).

O HIPS *Architect* é a ferramenta responsável pela modelagem do problema JSS, criando o cenário de produção desejado, permitindo configurar os layouts de produção e definir os parâmetros e forma de otimização, desta forma, compondo o ambiente multiagente. No Hips *Architect* pode ser executado o cenário projetado, integrado a plataforma JADE.

O pacote de classes JHIPS, é composto de classes bases para representar o cenário de produção modelado no Hips *Architect*, estruturando e carregando as definições configuradas no modelo.

A forma de interação entre a produção, sistema ERP e o *framework* HIPS é representado na Figura 16.

Figura 16. Arquitetura HIPS

Na Figura 16 há o ambiente de produção, gerenciado pelo sistema ERP que utiliza o banco de dados para armazenar informações. O *framework* HIPS é integrado ao sistema ERP através do banco de dados, lendo informações necessárias para a modelagem do ambiente de produção e permitindo o desenvolvimento das classes da aplicação, utilizando o pacote de classes JHIPS.

5.2 HIPS Architect

A ferramenta HIPS *Architect* foi projetada para possibilitar a modelagem de cenários de produção e sua execução. O cenário modelado na ferramenta pode ser configurado de acordo com as necessidades do usuário e cada agente modelado pode ser associado a uma classe distinta de agente JADE.

5.2.1 Modelagem

A modelagem da ferramenta HIPS *Architect*, por se tratar de uma aplicação orientada a objetos, foi feita utilizando UML[4] 2.1, através da ferramenta de modelagem *Enterprise Architech* 6.5, cujo fornecedor é *Sparx Systems Corporation*.

Serão apresentados os diagramas principais da modelagem, que compreende o caso de uso principal, diagrama de atividades e de classes. Também será descrito o modelo de dados utilizado para armazenamento das estruturas especificadas pelo *framework*.

5.2.1.1 Diagrama de Caso de uso principal

O diagrama de caso de uso demonstrado na Figura 17 mostra a interação dos atores:

- especialista em Ciências da Computação: responsável pela configuração da ferramenta HIPS Architect e também, pelo desenvolvimento das classes JADE;
- especialista Engenheiro de Produção: responsável por definir o layout de produção, a interação dos recursos de produção e da sua sistemática;
- usuário: responsável pela supervisão do ambiente.

A interação do Usuário restringiu-se em configurar os parâmetros dos agentes de fase e recurso e executar o ambiente. O engenheiro de produção pode efetuar as mesmas operações do usuário, porém pode manter todo o ambiente. O especialista em ciências da computação administra

[4] Segundo Deboni (2003) a UML é essencialmente uma linguagem de modelagem, definida por uma metalinguagem, isto é, a partir da UML podem ser geradas outras linguagens coerentes com o objetivo original de descrever um sistema orientado a objetos. A notação da UML é padronizada e deve ser assim para poder descrever com precisão cada parte do software, de forma única e de um modo onde qualquer um, que conheça UML, possa ler o mesmo software. A notação da UML é uma notação gráfica, na qual a descrição de um software é feita com símbolos gráficos padronizados que se relacionam formando diagramas. Cada diagrama apresenta uma visão parcial do sistema de software e a união dos diagramas deve representar um único sistema de software. Os diagramas são descritos por nomes e termos padronizados, que compõem um glossário próprio de termos que também faz parte da UML.

toda a ferramenta, com as mesmas atribuições do engenheiro de produção, porém especifica e desenvolve as classes Java e configura o ambiente.

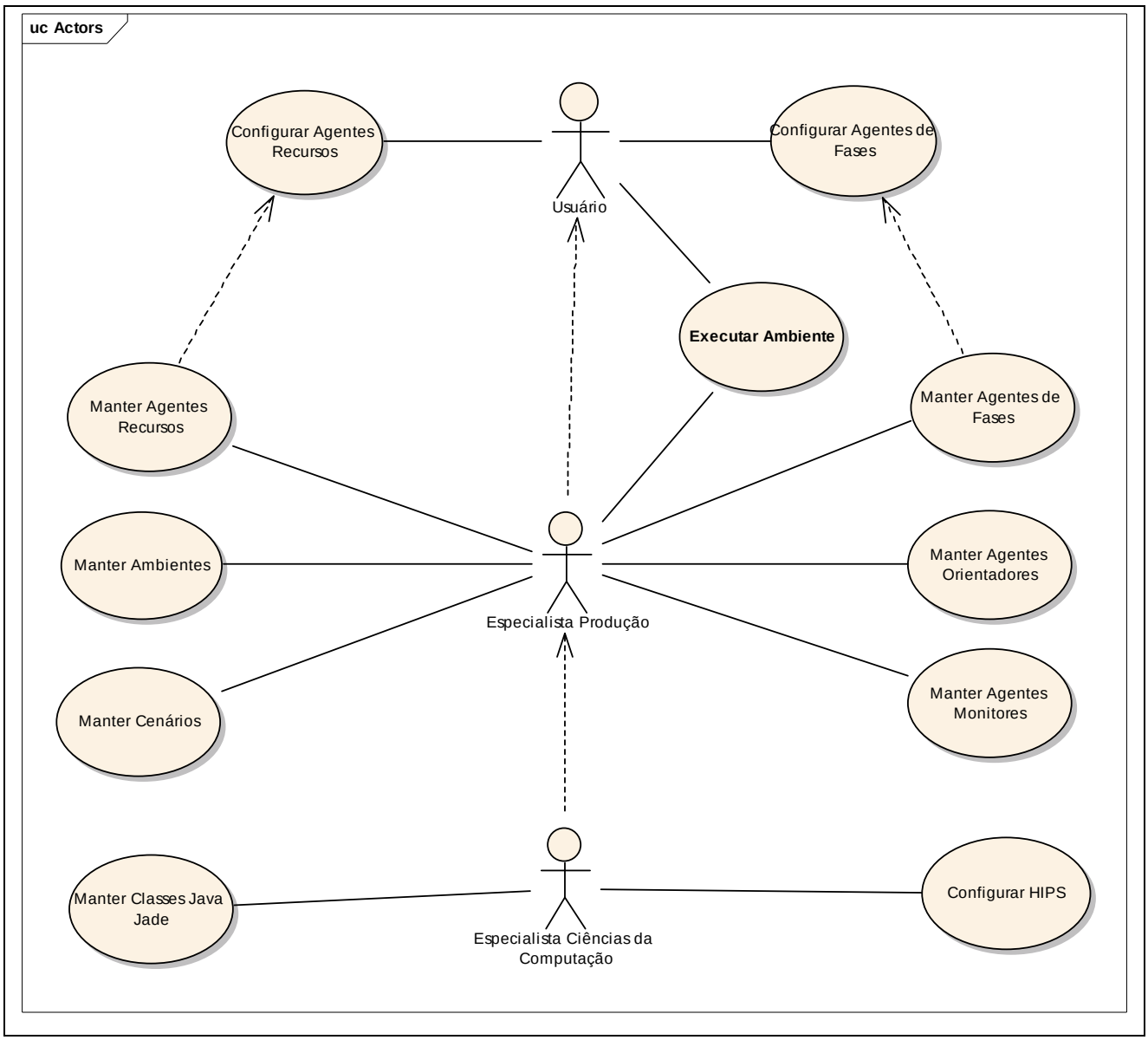

Figura 17. Caso de Uso Principal

5.2.1.2 Diagrama de Classes

O diagrama de classes especificado para o desenvolvimento da ferramenta HIPS Architect pode ser visualizado na Figura 18.

A estrutura de classes especificada se divide em três grupos:

- classes fornecidas pela ferramenta de desenvolvimento: (CodeGear RAD Studio 2009, ou também popularmente conhecido como “Delphi 2009”), representadas no diagrama de classes através do estereótipo “<<utility>>”. São classes básicas ao desenvolvimento, sua estrutura não foi representada por fazerem parte da ferramenta de desenvolvimento;
- classes base: representadas pelo estereótipo “<<class>>”, foram definidas para representar os agentes e suas características e são utilizadas pelas classes de formulários como base para gerenciar os objetos em tela e também, para armazená-los no banco de dados;
- formulários: são as interfaces da ferramenta com os atores, representadas pelo estereótipo “<<form>>” e são responsáveis pela entrada e saída de dados, permitindo aos atores a manutenção e interação.

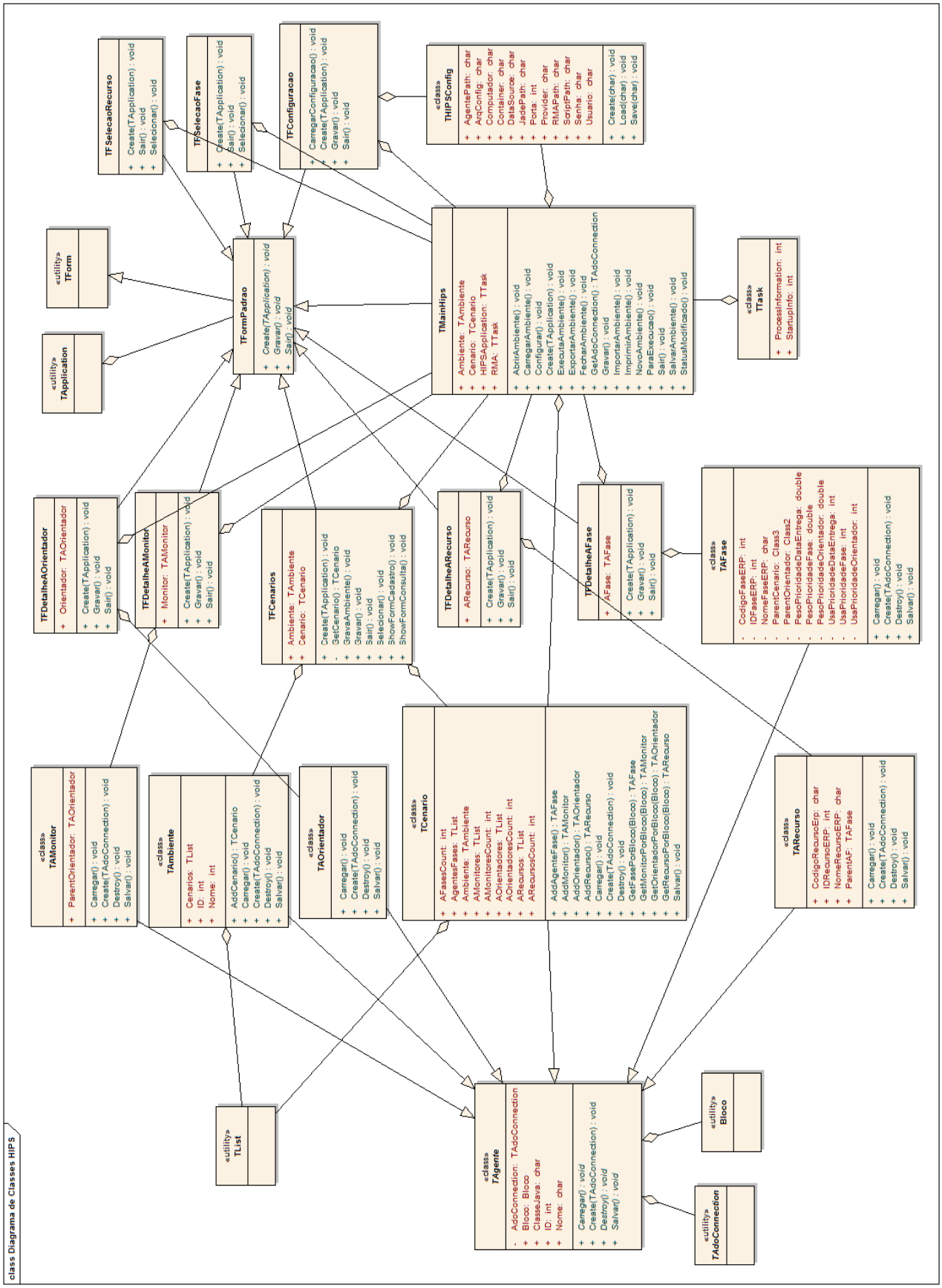

Figura 18. Diagrama de classes da ferramenta HIPS architect

5.2.1.3 Diagrama de Atividades

O diagrama de atividades representado na Figura 19 demonstra o fluxo da informação do caso de uso manter ambiente. Este diagrama de atividades foi escolhido por ser o principal diagrama de atividades da ferramenta HIPS *Architect.*

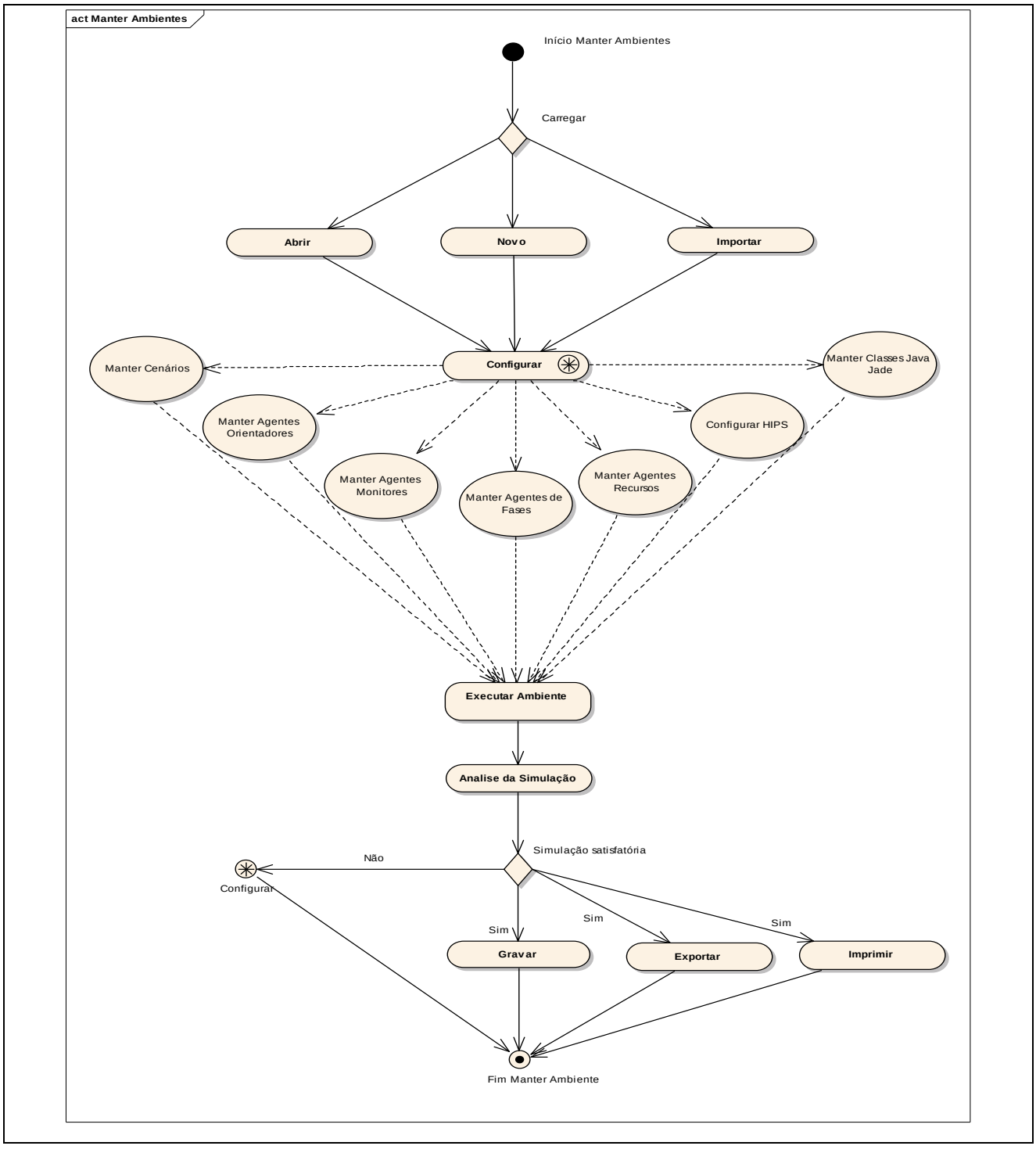

Figura 19. Diagrama de atividades

5.2.1.4 Modelo de Dados

O modelo de dados utilizado para armazenar as especificações da *framework* HIPS é demonstrado na Figura 20. Este modelo de dados contém além das definições da tabelas, também a estrutura requerida de *Views*[5], geradas a partir da integração com o sistema ERP.

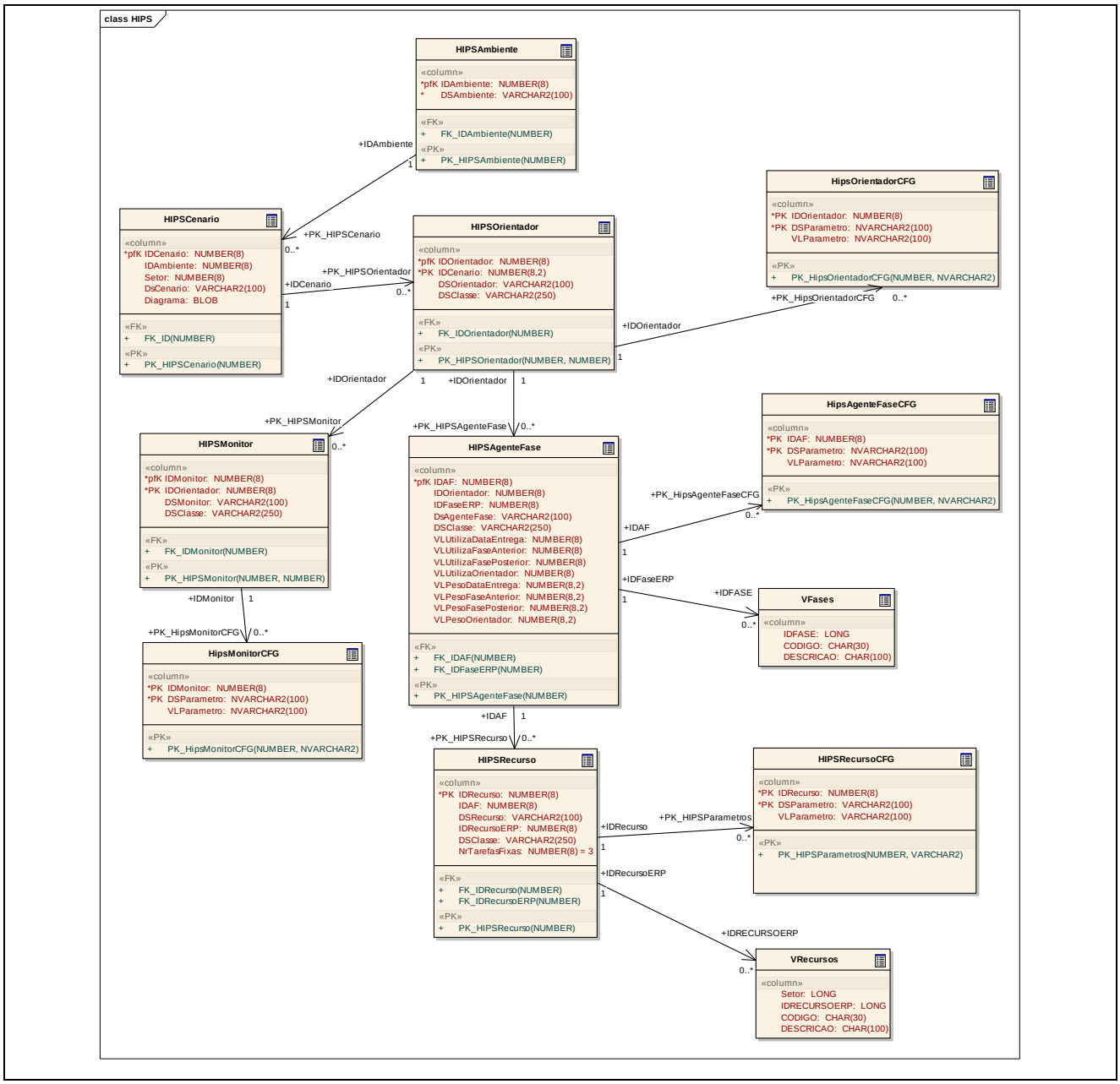

Figura 20. Modelo de dados

[5] Objeto de banco de dados gerado a partir de uma consulta sobre as demais tabelas e estruturas do banco. A forma de acesso as informações é semelhante ao acesso de uma tabela.

5.2.2 Descrição da Interface

A tela principal da ferramenta HIPS *Architect* é apresentada na Figura 21, que mostra as principais divisões em destaque através dos itens representados. Na parte superior (item 1) temos as principais funções quanto à manutenção dos ambientes. Os recursos para edição e visualização (item 2), estão divididos em cinco barras de ferramentas, onde na primeiro linha estão Fonte, *Zoom*, Execução e na segunda linha Agentes e Desenho. Na região central (item 3) é o espaço reservado para criação do ambiente de produção, ou seja, o editor e embaixo (item 4), uma barra de status do ambiente atual.

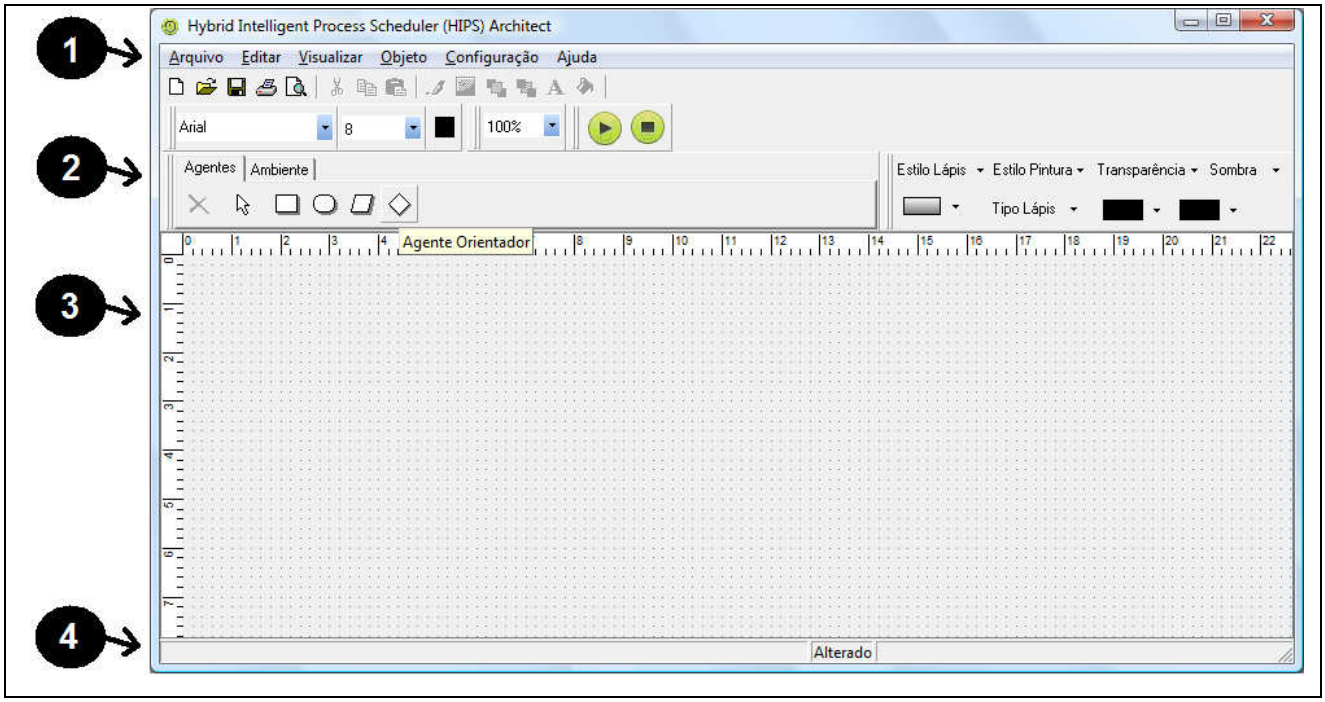

Figura 21. Tela principal HIPS Architect

O usuário da ferramenta para modelar um novo ambiente, deverá selecionar a barra de ferramentas Agente e selecionar dentre as quatro opções disponíveis: Recurso, Fase, Monitor e Orientador. Selecionando a desejada, deverá clicar no editor. O novo agente aparecerá representado, desta forma podendo modelar todo o seu ambiente de produção.

Cada agente pode ser configurado, para isso, o usuário deverá executar um duplo clique no agente desejado e um formulário de configuração desse agente será exibido, conforme demonstrado na Figura 22.

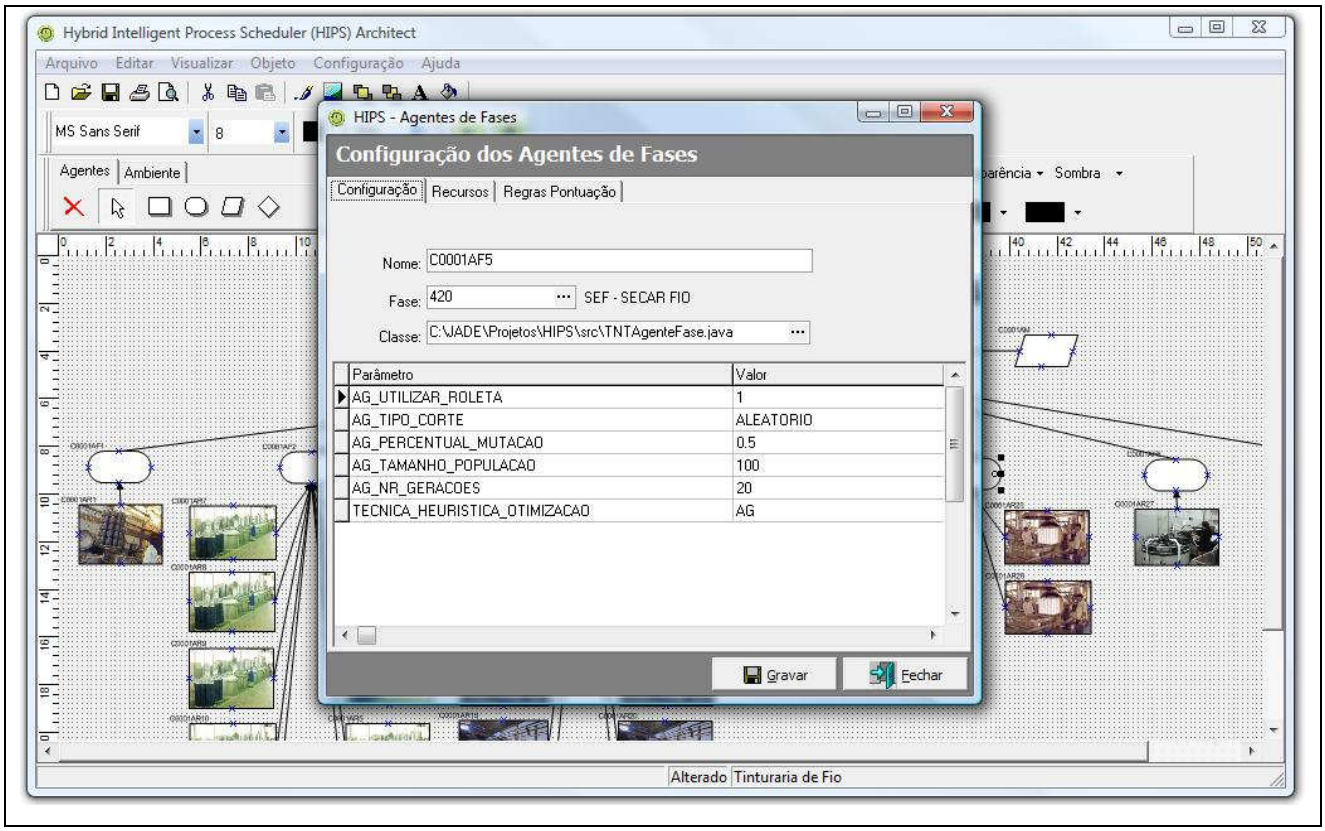

Figura 22. HIPS Architect configurando agente de fase

Recursos visuais como imagens, formatação de texto, mensagens, setas, etc para uma melhor apresentação do ambiente foram incorporados ao HIPS *Architect*, como pode ser visualizado na Figura 21.

Após o ambiente modelado e configurado, o usuário poderá utilizar o próprio ambiente do HIPS *Architect* para efetuar a execução e avaliação, bastando para isso utilizar os recursos da barra de ferramentas Execução.

As configurações disponíveis no HIPS *Architect* permitem informar os parâmetros necessários para integrar a *framework* JADE. Desta forma, podem ser utilizados os recursos para depurar e visualizar os agentes desenvolvidos em Java e toda a comunicação entre os agentes modelados. Um exemplo da execução de um ambiente pode ser visualizado na Figura 23.

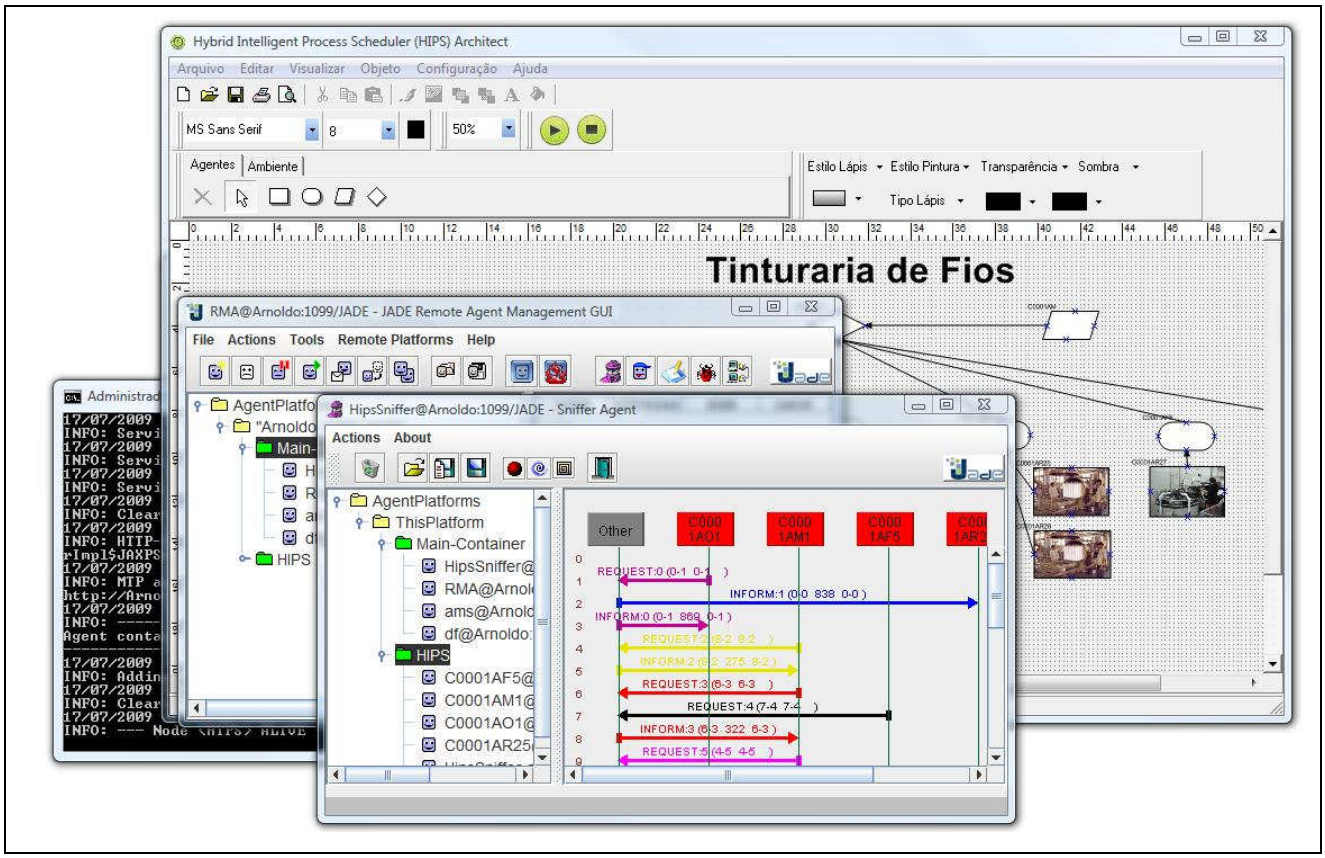

Figura 23. HIPS Architect executando ambiente.

5.3 JHIPS

Os pacotes de classes JHIPS são classes Java que utilizam a *framework* JADE e são divididos em três grupos:

- JHIPS Ontology: são as classes que compõem a ontologia do sistema multiagente;
- JHIPS Base: são os agentes base HIPS, possuem as estruturas básicas para execução do sistema multiagente para representar um cenário de produção modelado no HIPS Architect;
- JHIPS Tools: são classes acessórias fornecidas pelo framework para acesso a dados e AG.

5.3.1 JHIPS Ontology

JHIPS *Ontology* é o pacote de classes padrão de ontologias disponibilizadas pelo *framework*, utilizadas pelos agentes para a troca de mensagens de forma padronizada.

A classes principal deste pacote é a HIPSOntology, que define a instância da ontologia, o vocabulário, predicados e os conceitos com seus atributos. Gerada pelo *plugin Bean Generator*, no editor Protégé, está integrada a plataforma JADE, herdando a classe jade.content.onto.Ontology.

As demais classes, são classes conceitos, herdando da plataforma JADE, a classe jade.content.Concept, que são:

- HIPSPrioridade: classe conceito disponível para controle das prioridades entre os agentes;
- HIPSProduto: conceito estabelece a manutenção de produtos;
- HIPSPedido: conceito base para manter os pedidos;
- HIPSOrdem: conceito de ordem a ser escalonada;
- HIPSFase: estabelece a necessidade da divisão da ordem em fases e disponibiliza a manutenção;
- HIPSTarefa: classe conceito para determinar a janela de tempo necessária em um determinado recurso por uma determinada ordem.

O diagrama de classes do pacote JHIPS *Ontology*, que segue os conceitos citados acima, pode ser visualizado na Figura 24.

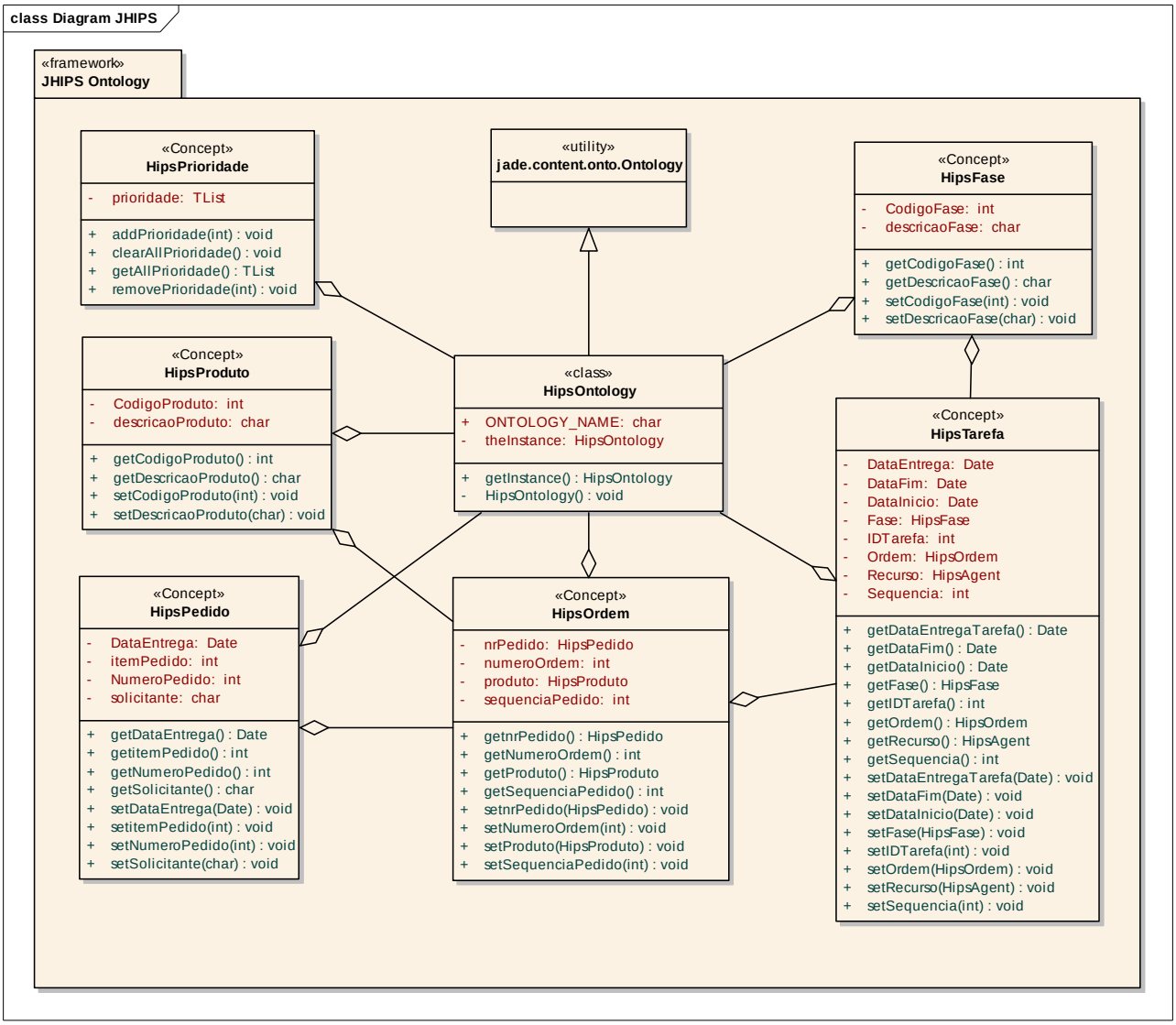

Figura 24. Diagrama de classes do pacote JHIPS Ontology.

5.3.2 JHIPS Base

O pacote de classes JHIPS Base é o conjunto de agentes HIPS, que implementam as estruturas básicas para execução do sistema multiagente, capaz de representar um cenário de produção modelado na ferramenta HIPS *Architect*.

A classe de agente principal é HIPSAgent, integrada a plataforma JADE, herdando a classe jade.core.AID. Está classe serve de base para os demais agentes, possuindo os métodos básicos de configuração ao iniciar Setup() e ao finalizar takeDown().

As demais classes de agentes que herdam as características do HIPSAgent são:

- HIPSAgenteRecurso: responsáveis por gerenciar o recurso de produção;
- HIPSAgenteFase: administram a fase de produção, escalonando os processos entre os recursos disponíveis;
- HIPSAgenteMonitor: responsável por informar a situação atual do sistema multiagente, trazendo os status de cada agente iniciado e sua situação perante a produção;
- HIPSAgenteOrientador: coordena as prioridades de produção, definindo a ordem em que as tarefas serão executadas.

O diagrama de classes de agentes do pacote JHIPS Base pode ser visualizado na Figura 25.

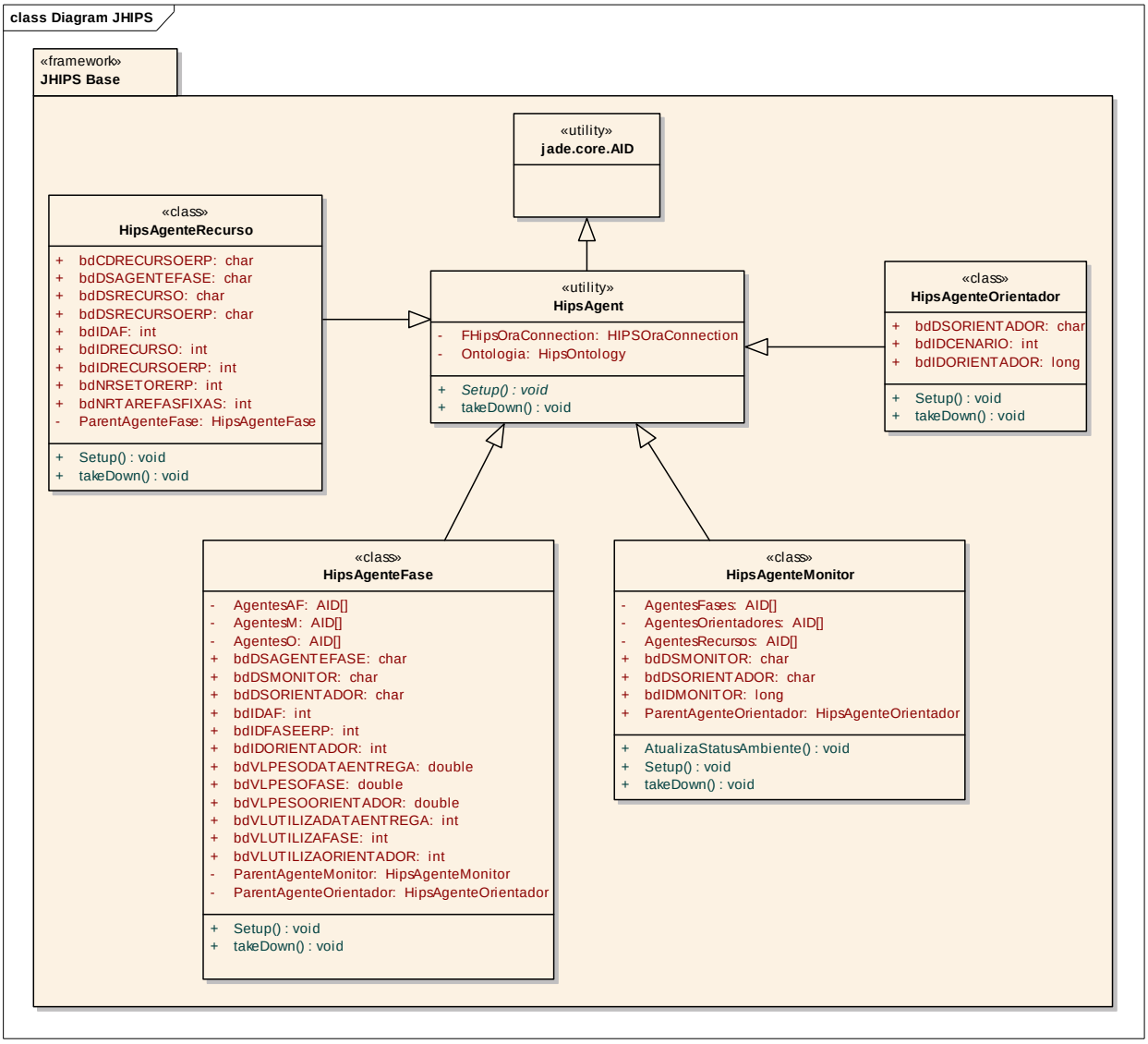

Figura 25. Diagrama de classes do pacote JHIPS Base.

As classes de agentes do pacote JHIPS Base não executam qualquer ação no cenário de produção, sua função é inicializar os agentes, carregar as configurações definidas no HIPS Architect e iniciar a ontologia HipsOntology.

5.3.3 JHIPS Tools

As classes disponíveis no pacote JHIPS *Tools* são classes acessórias para o desenvolvimento de aplicações HIPS. São fornecidas duas classes: HIPSOraConnection e HIPSAG.

A classe HIPSOraConnection tem finalidade de fazer a comunicação com o banco de dados Oracle (Oracle, 2009), permitindo consultar e manter informações.

HIPSAG é uma classe base para utilização de AG. Foi desenvolvida durante o desenvolvimento da aplicação e posteriormente incorporada ao pacote de classes acessórias do *framework*.

O diagrama de classes do pacote JHIPS Tools pode ser visualizado na Figura 26.

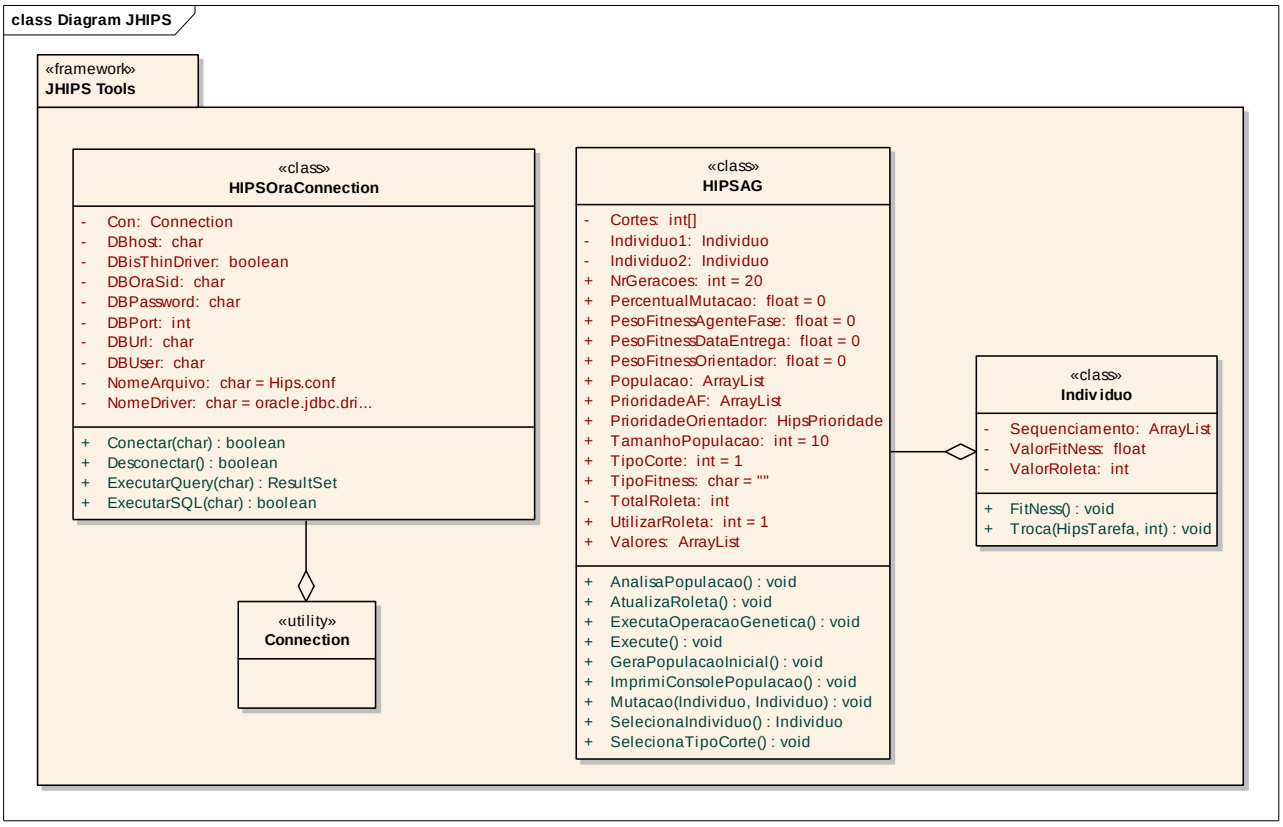

Figura 26. Diagrama de classes do pacote JHIPS Tools.

5.4 Tecnologias Relacionadas

O *framework* HIPS utiliza o banco de dados Oracle 11g, da empresa Oracle Corporation (Oracle, 2009), como fonte de dados, onde na mesma instância do banco de dados, havia os esquemas do sistema ERP, HIPS e aplicação HIPS.

A ferramenta HIPS *Architect* foi desenvolvida utilizando a linguagem de programação Delphi, no ambiente Codegear RAD Studio 2009 e possui integração com a plataforma JADE.

O pacote JHIPS foi desenvolvido na linguagem de programação Java, da Sun Microsystems, utilizando a Java *Development Kit* (JDK) na versão 6, update 12 (1.6.0_12b64), 64bits. O ambiente utilizado foi o Eclipse Plataform (Eclipse, 2009), versão 3.4.1 Build 1700.

O grupo de classes JHIPS *Ontology*, foi gerado pelo *plugin* Ontology *Bean Generator*, inserido no editor Protégé, para Java, também integrado a plataforma JADE.

A ferramenta de modelagem *Enterprise Architect* (EA), desenvolvida pela empresa Sparx Systems Co, na versão 6.5, foi utilizada para modelar parte do *framework*. Esta modelagem continua na aplicação, onde serão mencionados os agentes herdados a partir do *framework*.

5.5 Limitações

O *framework* HIPS possui algumas limitações que foram definidas para se enquadrar ao escopo dessa pesquisa, porém trabalhos futuros de pesquisa podem retirar essas limitações e tratá-las, complementando as estruturas propostas.

As limitações definidas são:

- pode ser modelado apenas um cenário por ambiente, ou seja, a comunicação entre dois cenários ainda não é permitida;
- somente um agente orientador e um monitor poderão ser definidos por cenário;
- os recursos de produção só podem se associar a um agente de fases;
- os agentes de fases só podem ter um orientador;
- o monitor só pode ser associado a um orientador;
- HIPS Architect só pode executar um ambiente por vez;

- JHIPS Base possuir agentes para o próprio ambiente e o cenário, permitindo a interação agentes X cenário X ambiente;
- JHIPS Tools fornecer novas classes de acesso a dados para outros bancos e também, outras classes para busca eurística além do AG.

6 VALIDAÇÃO DO FRAMEWORK

Com o intuito de validar o *framework* proposto HIPS, foi desenvolvido um estudo de caso, através de uma aplicação. Para melhor descrever a aplicação, primeiramente será relatado um contexto de problema para escalonamento de processos, posteriormente as etapas de modelagem e desenvolvimento, relatando ao final, os resultados obtidos.

Esta aplicação foi proposta, tendo como base um cenário de produção real, que atende as especificações de um problema JSS, para escalonamento de processos distribuídos. Foi utilizada a metodologia MaSE para a modelagem da aplicação.

6.1 Problema exemplo

Conforme descrito no item anterior, o problema utilizado para modelagem no *framework* é do tipo JSS. Portanto as etapas anteriores ao estado de possuir as ordens para seqüenciamento da produção e posteriores a fabricação, mencionadas na Figura não serão relatadas e fazem parte das atividades desempenhadas pelo sistema ERP, que gerencia a indústria e servirá como fonte de informação através de integração para o *framework* proposto.

Na Figura 27, em destaque, está onde o *framework* atua dentro do processo de fabricação.

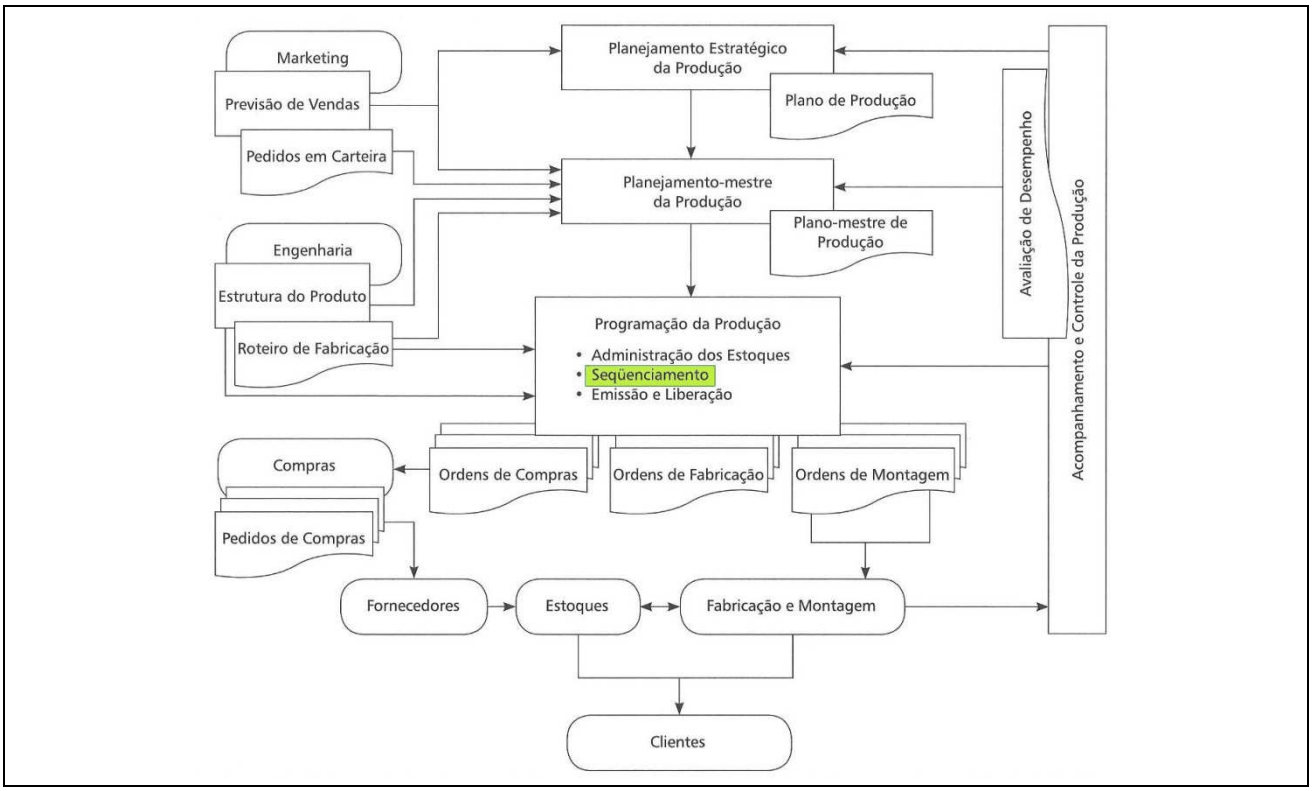

Figura 27. Fluxo de Informações

Fonte: Adaptado de Tubino (2007).

O cenário de produção real segue o modelo apresentado na Figura 28.

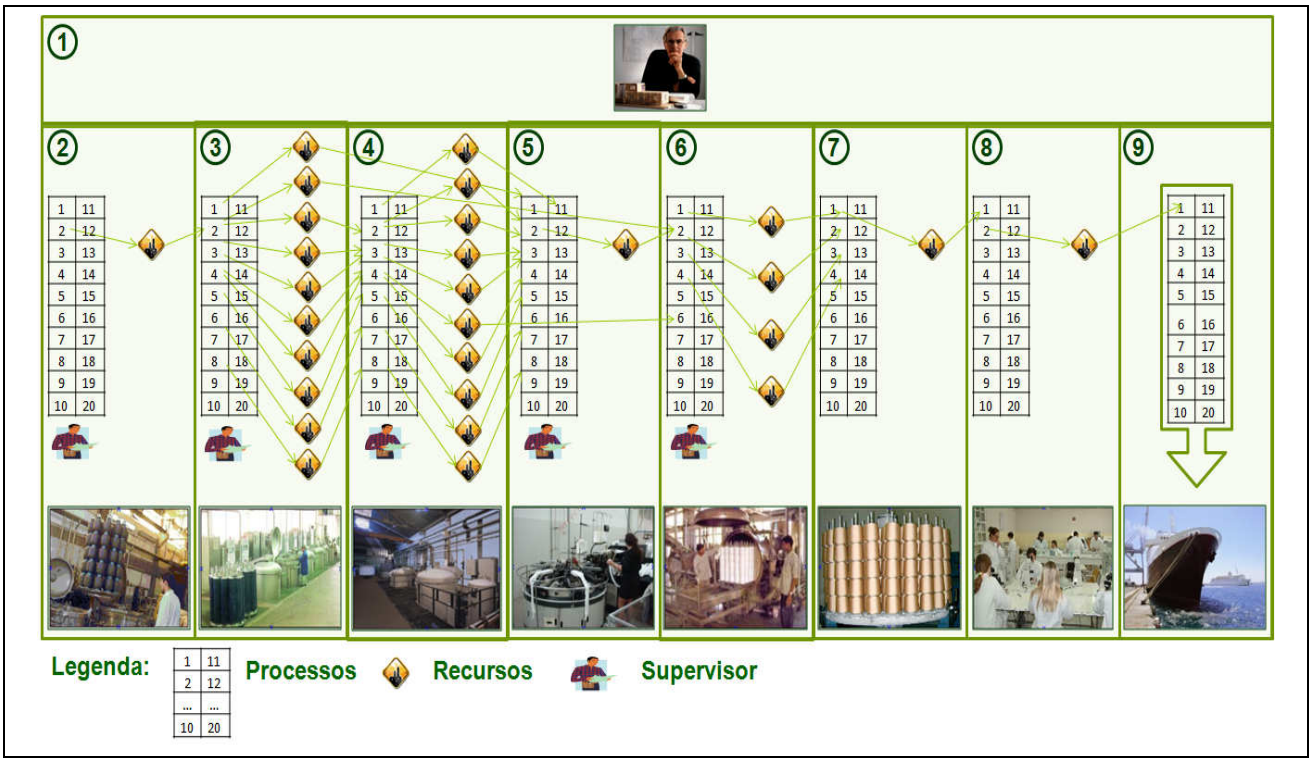

Figura 28. Cenário de Produção

O modelo representa o setor de tinturaria de fios (Vieira, 1988) (Araújo, 1986), que possui um fluxo de produção composto por sete fases distintas (numeradas de 2 a 8 no modelo), onde há processo de transformação de material em todas elas, porém não necessariamente em seqüência. O modelo apresenta ainda o Planejamento, Programação e Controle da Produção (PPCP) com o número um e o estoque final de produto, com o número nove, que dependendo do setor a ser escalonado, pode ser o estoque de produto acabado ou um estoque intermediário de outro setor, dando início a um novo modelo.

Cada fase de produção possui um conjunto de processos distintos, possuindo recursos específicos para a realização daquela etapa na produção do produto. Estoque intermediário pode ocorrer no processo, em virtude dos recursos de produção não atenderem a demanda ou problemas durante a execução. Uma vez executado o processo no recurso de produção, o produto intermediário resultante alimenta outro estoque intermediário, que irá abastecer os recursos produtivos da fase seguinte, que pode ou não, estar na seqüência apresentada.

Dependendo do tamanho deste estoque intermediário, o número de alternativas para escolha do processo a executar e da seqüência que será executada pode se tornar uma operação que exige um tempo considerável do supervisor de produção. Esta situação é multiplicada um número considerável de vezes, se levarmos em consideração que a escolha local da fase, ocasionará uma mudança de seqüenciamento em todas as fases subseqüentes, podendo causar adiantamento ou atrasos nas entregas dos produtos, no final do fluxo.

O seqüenciamento de uma fase ou recurso determinado pela imposição de outro, gera comumente uma perda de eficiência local, fazendo com que os objetivos sejam atingidos em detrimento da eficiência global do sistema. Parunak (1997) definiu como possíveis aplicações industriais eficientes de agentes: desenvolvimento de produto, processos de planejamento e escalonamento e operações de controle de equipamentos.

Portanto há necessidade de uma ferramenta que possibilite a comunicação entre as fases de produção, buscando atingir um equilíbrio entre eficiência local da fase e a eficiência global do setor, através da comunicação, cooperação e auto-organização dos recursos, o que se pretende atingir através da modelagem e desenvolvimento da aplicação TNT (TiNTuraria) implementada utilizando o *framework* HIPS.

6.2 Aplicação

Com o objetivo de verificar a aplicabilidade do *framework* HIPS proposto, foi desenvolvida a aplicação HIPS TNT. Inicialmente foi feita a modelagem dessa aplicação através da metodologia MaSE, definindo os objetivos, utilizando os agentes propostos no *framework* HIPS para construção dos diálogos e classes. Posteriormente o modelo construído foi sendo desenvolvido utilizando o *framework* HIPS, através do HIPS *Architect* e dos pacotes JHIPS para construção das classes de agentes.

6.2.1 Modelagem

A aplicação HIPS TNT proposta foi modelada utilizando a metodologia MaSE proposta por DeLoach (2001).

6.2.1.1 Metodologia MaSE

Conforme descrito no item 3.3.1.2, sobre a metodologia MaSE, ela é dividida em duas etapas: análise e modelagem, gerando a aplicação ao final desse processo.

6.2.1.2 Definição de Objetivos (*Capturing Goals*)

Segundo DeLoach (2001) a fase de definição de objetivos pode ser dividida em dois passos: identificação e estruturação. Identificados os objetivos principais e secundários, eles foram estruturados conforme o diagrama abaixo mostrado na Figura 29.

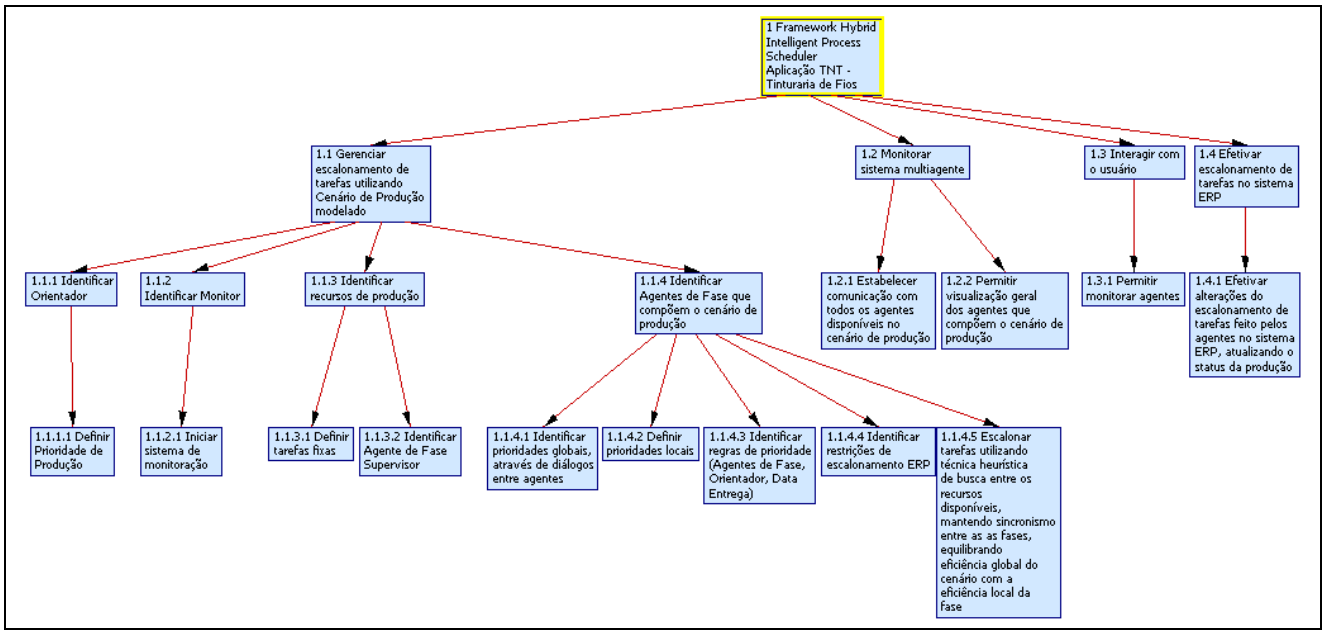

Figura 29. Diagrama de objetivos

6.2.1.3 Aplicação dos Casos de uso (*Applying Use Cases*)

A fase de aplicação dos casos de uso é onde são representadas as conversas entre os agentes. A representação dessas conversas se dá através de diagramas de seqüência (DeLoach, 2001). O diagrama de seqüência é utilizado para determinar o conjunto mínimo de mensagens que devem ser trocados entre os papéis definidos para os agentes.

A Figura 30 mostra a representação da troca de mensagens entre os agentes de fase, TNTAgenteFase e o agente orientador, TNTAgenteOrientador. Nas Figuras 30, 31, 32 e 33, optou-se por representar já o agente, facilitando a compreensão, porém poderiam ser apresentados também através dos papéis que serão exercidos pelos agentes, onde posteriormente serão descritos cada tipo de agente e sua função dentro do sistema multiagente.

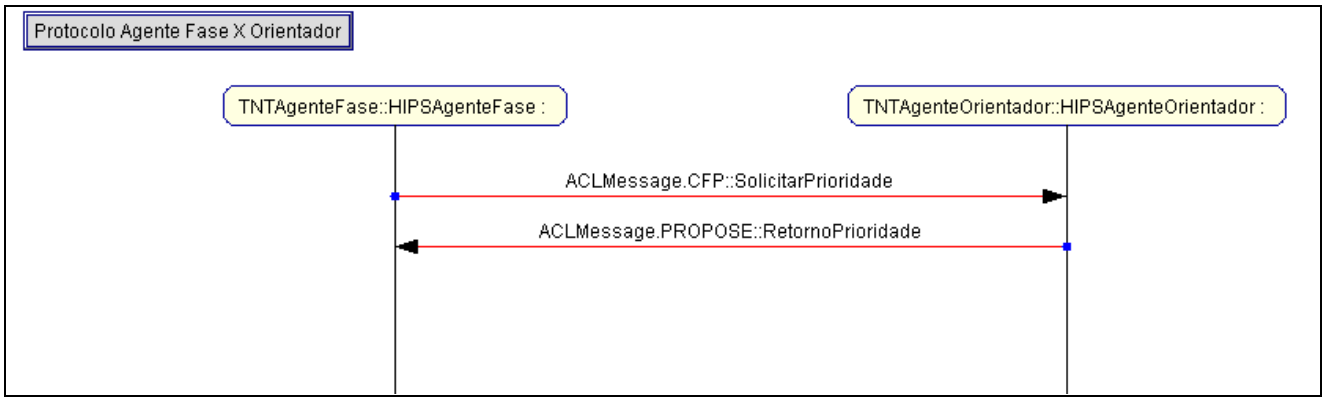

Figura 30. Protocolo Agente Fase x Orientador

As mensagens trocadas entre um agente de fase, com os demais, podem ser visualizadas na Figura 31. Foram adotados os nomes TNTAgenteFasePré e TNTAgenteFasePós para representar outros agentes de fases que podem fazer parte do cenário de produção.

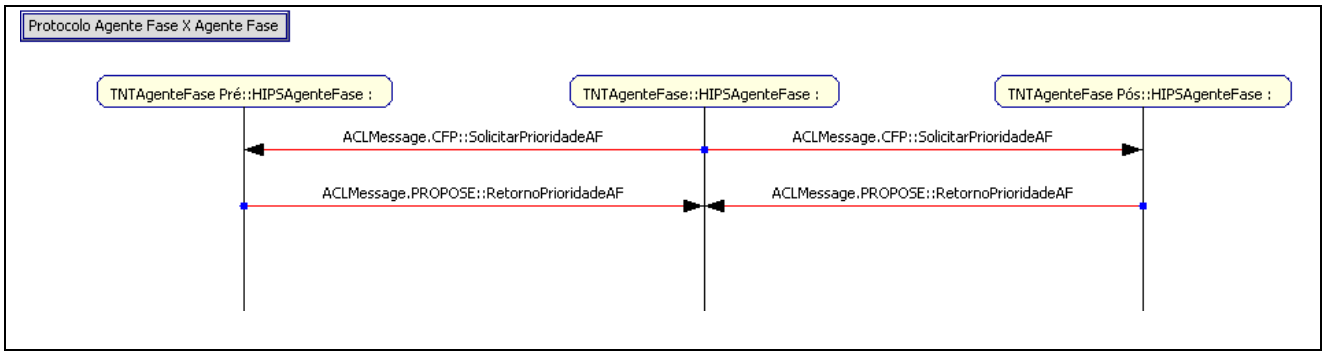

Figura 31. Protocolo Agente Fase x Agente Fase

A comunicação através de mensagens, entre os agentes recursos, TNTAgenteRecurso e o agentes de fases, TNTAgenteFase, podem serem visualizadas na Figura 32.

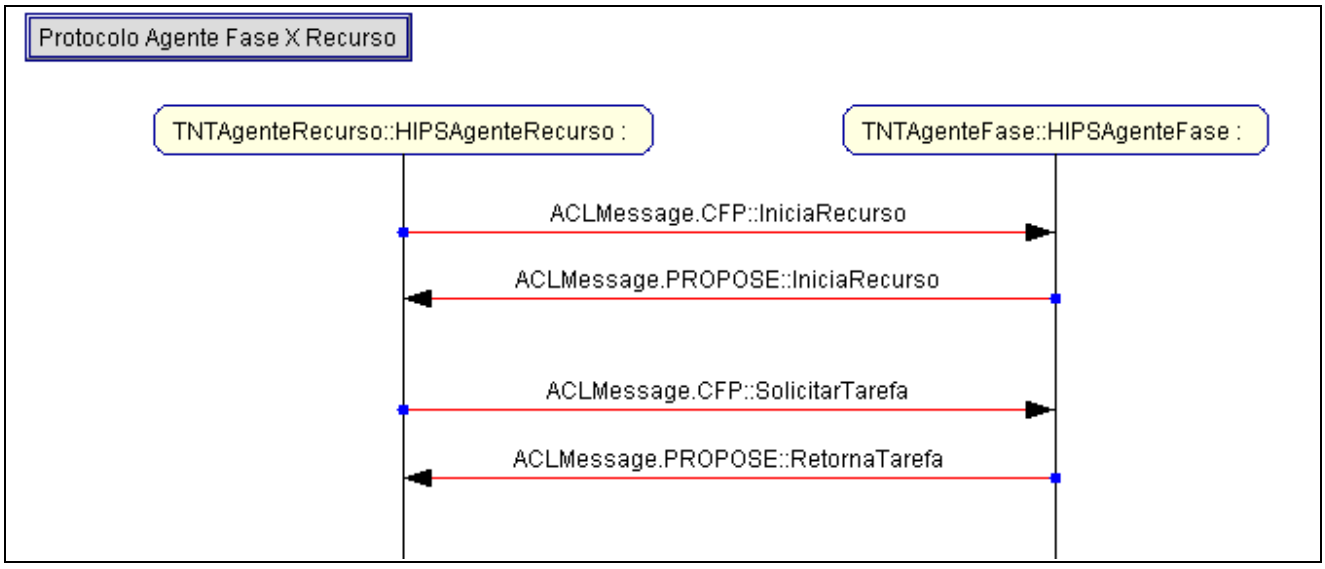

Figura 32. Protocolo Agente Fase x Recurso

A comunicação para atualização de status do cenário de produção, apresentado pelo agente monitor, TNTAgenteMonitor é apresentada na Figura 33.

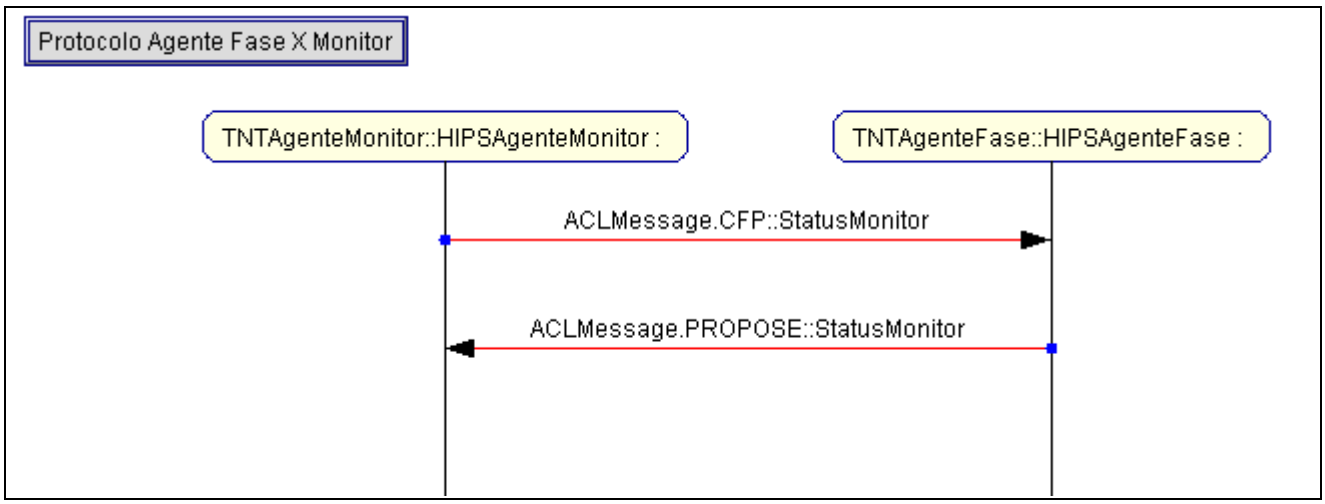

Figura 33. Protocolo Agente Fase x Monitor

Outras mensagens são trocadas entre os agentes que fazem parte da aplicação HIPS TNT e do *framework*, porém foram mencionadas apenas as principais.

6.2.1.4 Refinamentos dos papéis (*Refining Roles*)

A terceira fase da metodologia MaSE é transformar os objetivos especificados no diagrama de objetivos em uma forma mais útil para a construção de sistemas multiagentes, que são os papéis (Deloach, 2001).

De acordo com Deloach (2001) os papéis são blocos usados para definir a classe do agente e capturar os objetivos do sistema durante a fase de design. Todo objetivo está associado a um papel e todo papel é responsabilidade de um agente.

O diagrama de papéis apresentado na Figura 34 mostra a interação entre o agente orientador, agente monitor, agentes de fases com outros agentes de fases e também, com os agentes recursos.

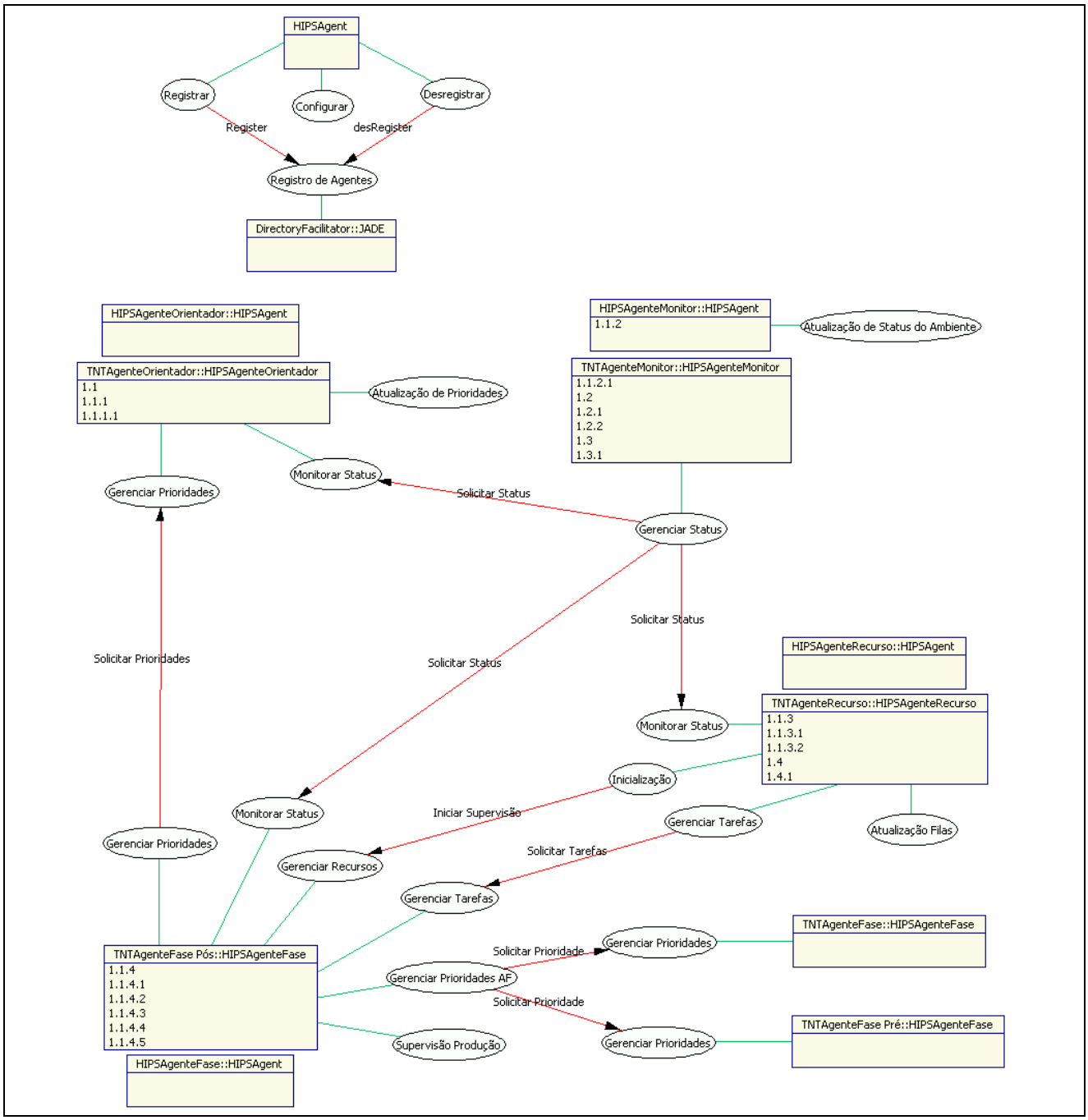

Figura 34. Diagrama de papéis

Foram definidos quatro papéis para efetuar o escalonamento de processos: Orientador, Monitor, Agente de Fase e Recurso.

6.2.1.5 Criando Classes de Agentes (*Creating agent class*)

Na fase de criação das classes de agentes da metodologia MaSE, as classes de agentes são identificadas a partir dos papéis (Deloach, 2001).

O resultado desta fase é o diagrama das classes de agentes, que deve demonstrar as classes de agentes e as mensagens trocadas entre elas. O diagrama das classes de agentes definido para a aplicação HIPS TNT é apresentado na Figura 35. Foi definido um agente JADE para representar a comunicação com o DF (*Directory Facilitator*), disponível na plataforma JADE.

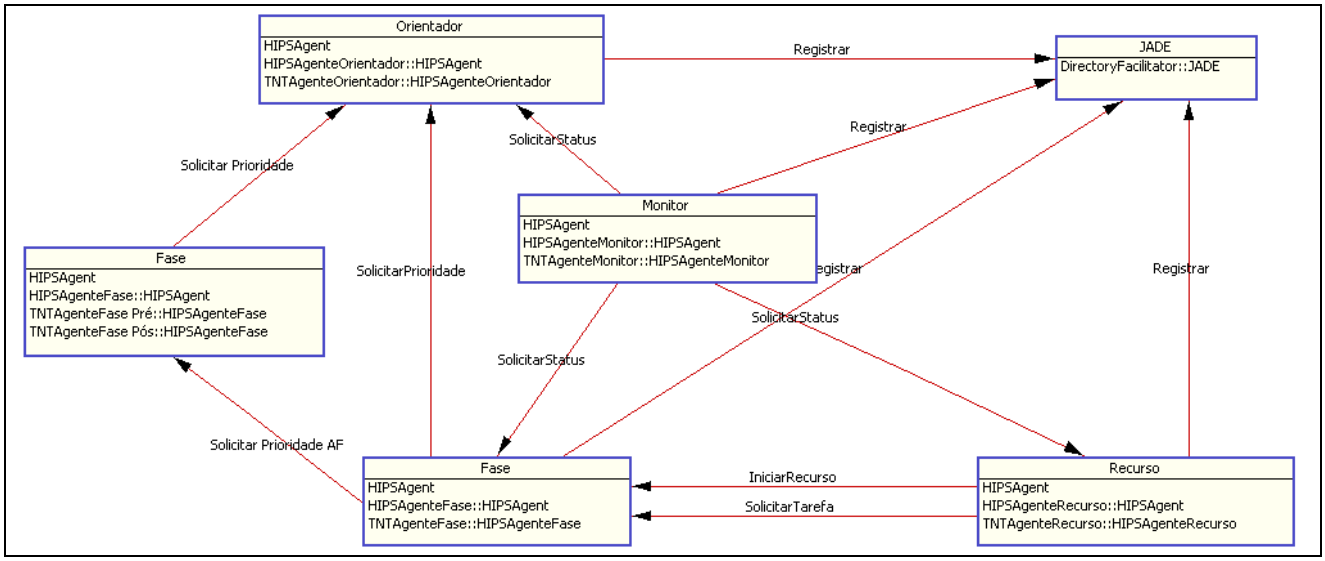

Figura 35. Diagrama de agentes

6.2.1.6 Construções de Conversas (*Construting conversations*)

A fase de construção de conversas da metodologia MaSE define um protocolo de coordenação entre dois agentes. Especificamente é a troca de mensagens entre duas classes de agentes, onde uma pergunta e a outra responde. Um diagrama de classe de comunicação é um par finito de estados de máquina que define os estados da conversa entre duas classes de agentes participantes (Deloach, 2001).

Os diagramas de classe de comunicação, que representam os estados da conversa apresentada na Figura 32, com a mensagem ACLMESSAGE.CPF::IniciarRecurso, entre o agente de fase, TNTAgenteFase e o agente recurso, TNTAgenteRecurso são apresentados nas Figuras 36 (pergunta) e Figura 37 (resposta).

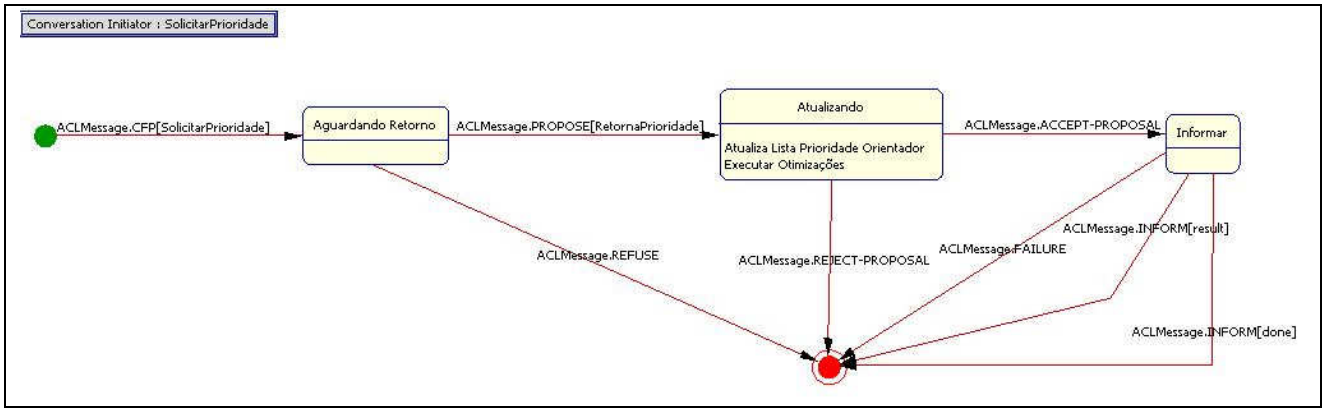

Figura 36. Diagrama de classe de comunicação protocolo Agente Fase x Recurso – Pergunta

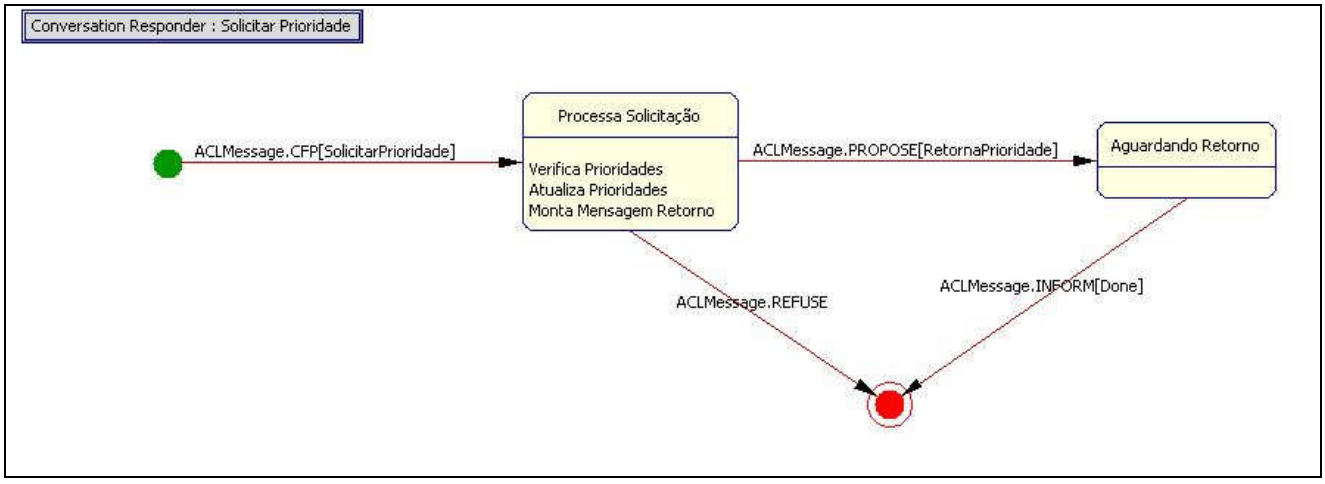

Figura 37. Diagrama de classe de comunicação protocolo Agente Fase x Orientador - Resposta

6.2.1.7 Montando as Classes dos Agentes (*Assembling agent class*)

Nesta fase, de acordo com a metodologia MaSE, são definidas as estruturas internas dos agentes. Na Figura 38 é apresentada a estrutura dos agentes propostos para a aplicação HIPS TNT, através de um diagrama de classes.

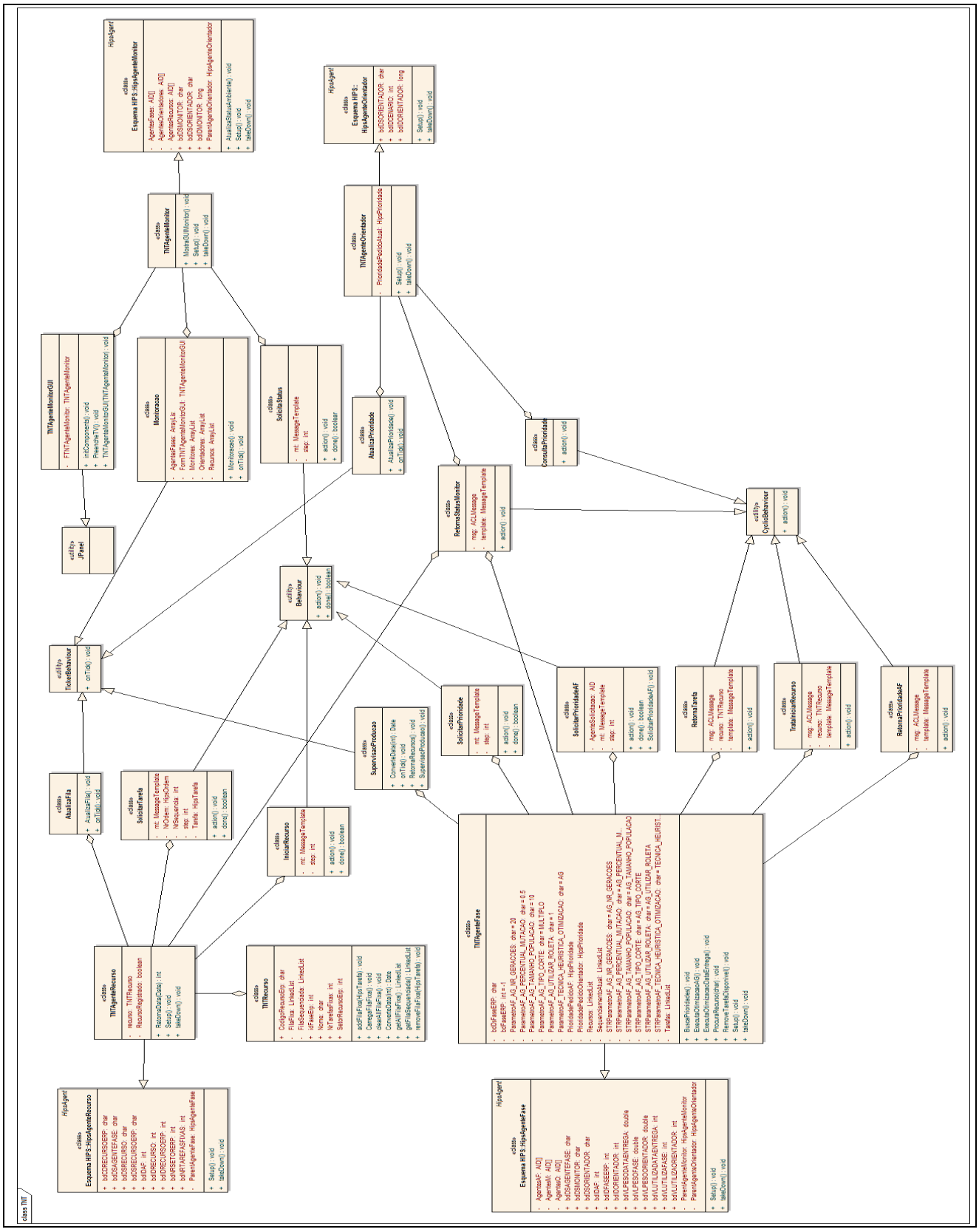

Figura 38. Diagrama da estrutura dos agentes

No diagrama da estrutura dos agentes, são apresentados também os agentes provenientes do *framework* HIPS, que servem de base para os agentes da aplicação HIPS TNT.

6.2.1.8 Design do Sistema (*System design*)

De acordo com a metodologia MaSE, a etapa de design do sistema utiliza o diagrama de implantação (*deployment diagram*) para definir parâmetros para o SMA, tais como: número de agentes de cada tipo, localizações dos agentes dentro do sistema, instâncias dos agentes ativas e suas respectivas classes, visão geral das conversas entre os agentes.

A aplicação HIPS TNT é executada através do HIPS Architect, portanto o número de instâncias dos agentes TNTAgenteFase e TNTAgenteRecurso varia de acordo com o cenário definido, porém foi representado no diagrama de implantação na Figura 39, um agente de cada tipo, exceto o TNTAgenteFase, que apresenta outra instância com o objetivo de demonstrar a comunicação com os demais agentes de fase do cenário.

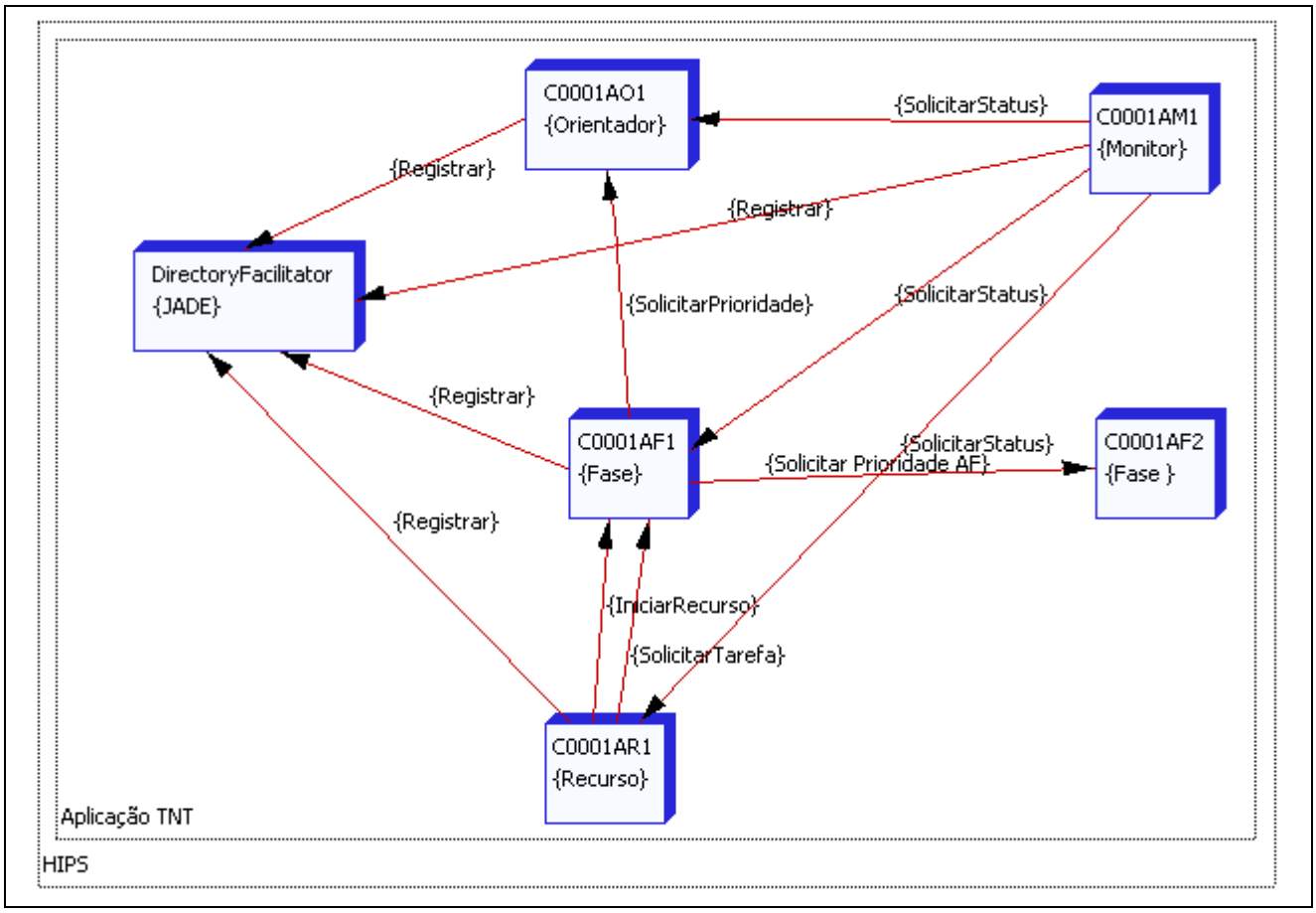

Figura 39. Diagrama de agentes

6.2.1.9 Ciclo de vida da aplicação HIPS TNT (*Life cycle*)

O ciclo de vida definido para a aplicação HIPS TNT é iniciado com a instanciação do agente Orientador, TNTAgenteOrientador e do agente monitor, TNTAgenteMonitor. Posteriormente são iniciados os agentes de fases, TNTAgenteFase para cada uma das fases quem compõem o cenário de produção e por fim, a instanciação dos agentes recursos, TNTAgenteRecurso.

O TNTAgenteOrientador inicializa buscando as prioridades de produção, que foram definidas como sendo prioridades do agente orientador, as datas de entregas dos pedidos em produção, ou seja, tem maior prioridade os pedidos com data de entrega mais próxima, que possuam alguma ordem pendente de produção no cenário. A atualização das prioridades de produção é feita com periodicidade de um minuto, através de um comportamento próprio do agente orientador. Os pedidos e ordens mencionados são disponibilizados através de integração, pelo sistema ERP.

O agente TNTAgenteOrientador possui ainda outros dois comportamentos, que são informar o seu status ao agente TNTAgenteMonitor e retornar as prioridades de produção aos agentes de fases TNTAgenteFase.

O agente responsável por informar o usuário qual o status de cada agente no cenário de produção é o TNTAgenteMonitor. Seu comportamento é baseado na monitoração do cenário de produção, buscando novos agentes registrados na plataforma JADE, através de constantes consultas ao DF. Também é atribuição do agente monitor, solicitar aos agentes de fases, recursos e orientador os seus status atuais, apresentando ao usuário através de uma interface, as informações de cada agente registrado na plataforma.

Agentes de fase supervisionam a fase de produção definida para eles no HIPS *Architect*. Passam a exercer a monitoração dos recursos de produção, assim que os agentes recursos, TNTAgenteRecurso, são iniciados e buscam a sua inicialização junto ao agente de fase o qual são alocados.

O TNTAgenteFase aguarda a inicialização de seus recursos de produção. Assim que recebe a requisição de início dos agentes TNTAgenteRecursos, passam a buscar os processos a serem escalonados nos recursos disponíveis inicializados, que não fazem parte da lista fixa de processos de cada recurso, definida também através do HIPS *Architect*.

Carregados todos os processos a serem escalonados, o TNTAgenteFase executa o procedimento de escalonamento, que verifica os parâmetros passados através do HIPS *Architect* na definição da técnica de otimização. No caso da aplicação HIPS TNT, foi definido o uso de AG como técnica heurística de otimização, porém outras técnicas podem ser implementadas.

A otimização através de AG é executada com periodicidade de um minuto, ou seja, executa a otimização, aguarda um minuto e executa a otimização novamente. Cada execução atualiza a lista de processos pendentes de produção, verificado se há novos recursos, verifica a lista fixa de cada recurso, consultas as prioridades do agente orientador, TNTAgenteOrientador e dos agentes de fase vizinhos, ou seja, todos os agentes de fase, o qual as ordens a serem escalonadas possuam processos posteriores a serem escalonados. Através dessa consulta, é possível determinar as prioridades de produção do setor e também de cada fase de produção posterior, buscando otimizar o escalonamento nesse sentido.

Cada indíviduo que compõem a população utilizada no AG, é uma lista de processos seqüenciados. Para o calculo da pontuação do seqüenciamento, ou seja, o *fitness* do indivíduo, ainda são considerados os pesos definidos no HIPS *Architect* ao configurar o agente de fase, para cada uma das regras de pontuação.

Os agentes do tipo TNTAgenteFase ainda possuem os comportamentos de retornar o status ao TNTAgenteMonitor, retornar ordens de produção seqüenciadas aos agentes TNTAgenteRecurso e retornar aos demais agentes de fase a sua prioridade de produção, também calculada através do critério da data de entrega, porém difere do agente orientador, pois nesse caso analisa somente as ordens pertinentes a fase.

Os recursos de produção representados através do agente TNTAgenteRecurso inicializam buscando as informações do recurso de produção que controlam, posteriormente integrando ao sistema ERP, carregando os processos a serem executados que já estão fixos, ou seja, compõem a fila fixa de processos que não podem ser alterados através de escalonamento do TNTAgenteFase.

Após estarem iniciados e com as informações carregadas, buscam seu agente de fase, através das configurações definidas no HIPS *Architect*, informando o recurso de produção que estão controlando e o número de processos em sua fila fixa. Uma vez inicializados perante o seu agente de fase, passam a ter o comportamento de monitorar o recurso de produção através de integração com o sistema ERP. Notando a confirmação do término de algum dos processos, solicitam ao

agente de fase a próxima ordem a ser adicionada a sua fila fixa, que trabalha da forma primeiro a entrar, primeiro a sair, ou seja, *First In, First Out* (FIFO).

Na Figura 40 é apresentado o cenário de produção desenvolvido na ferramenta HIPS *Architect*, seguindo o modelo do problema exposto na Figura 28.

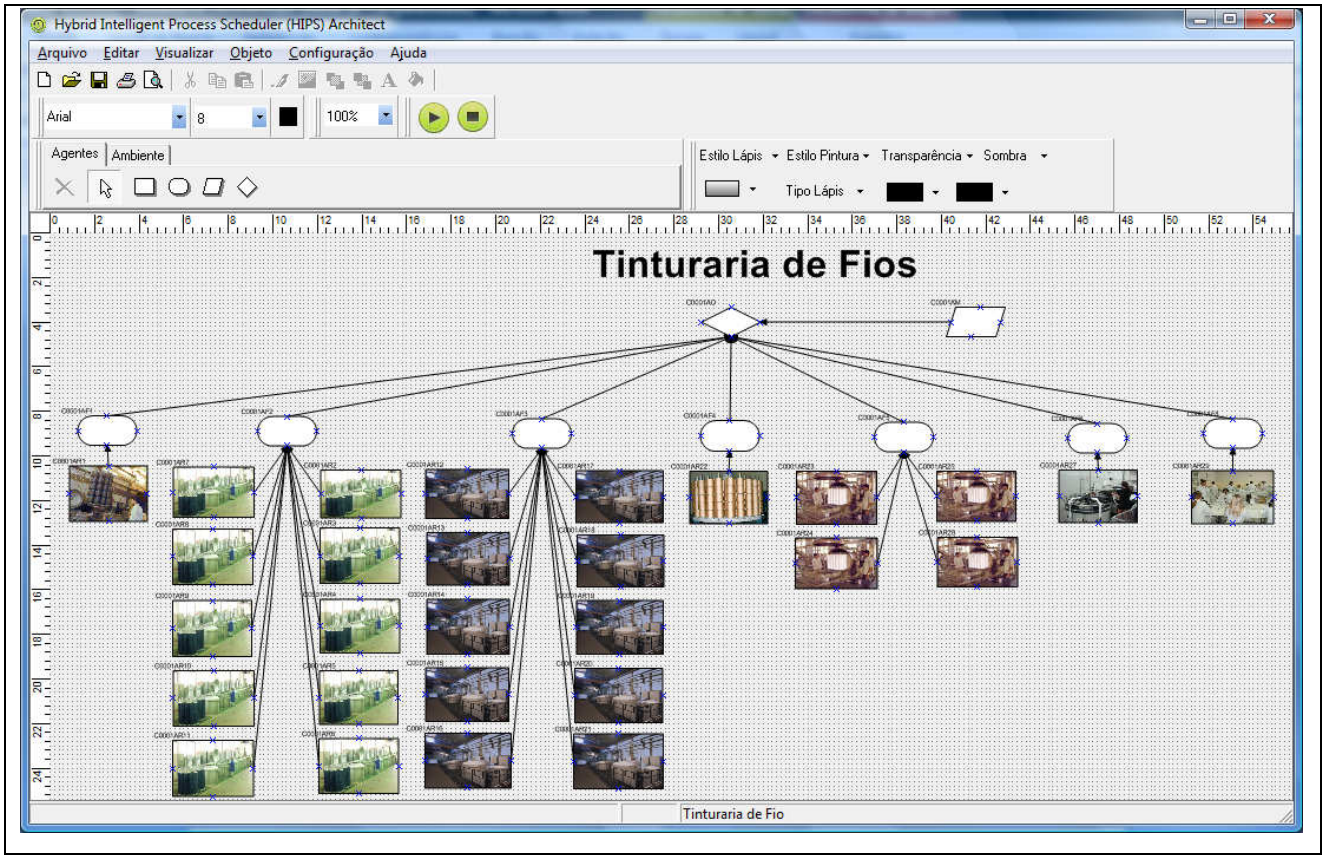

Figura 40. Cenário de produção modelado no HIPS Architect

A execução desse cenário de produção, ou seja, a aplicação HIPS TNT pode ser visualizada na Figura 41, onde foram utilizados também, alguns recursos da plataforma JADE para visualização dos agentes criados e dos diálogos sendo trocados naquele instante.

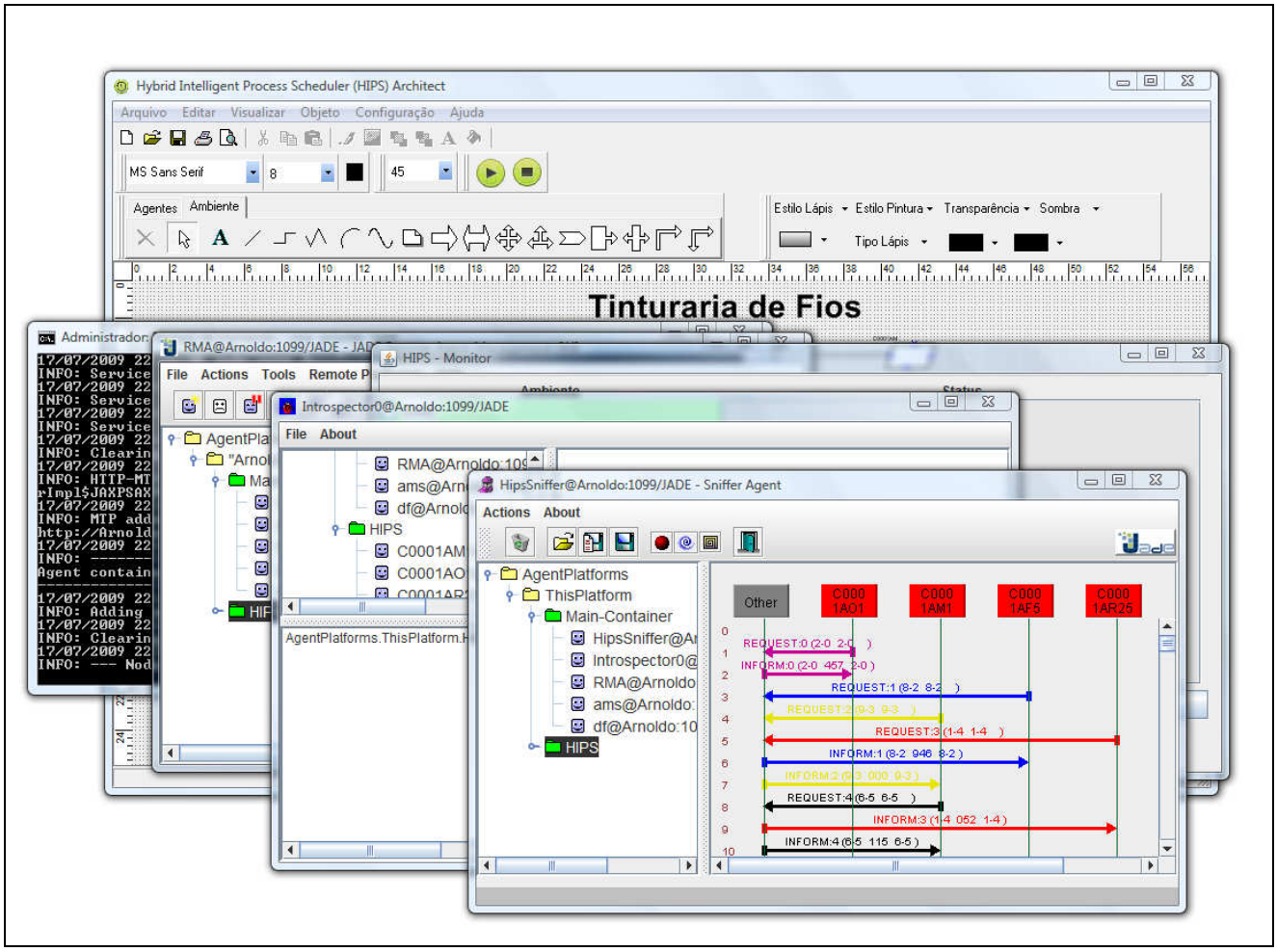

Figura 41. HIPS TNT em execução

6.3 Tecnologias Relacionadas

Adicionando as tecnologias relacionadas para o desenvolvimento do *framework*, relatadas no item 5.4, para a aplicação ainda foi necessário a utilização de uma ferramenta para otimizar o desenvolvimento da interface com o usuário, do agente monitor da aplicação. Para essa necessidade foi escolhida a ferramenta JFormDesigner versão 4.0 da empresa *FormDev Software* (FormDev, 2009).

A ferramenta AgentTool 2.0 foi utilizada para o desenvolvimento dos diagramas propostos pela metodologia MaSE, sendo fornecida pelo próprio autor da metodologia Deloach(2001).

6.4 Limitações

Devido à diversidade de problemas que podem ser modelados no *framework* e executados, não há um passo-a-passo para a criação de um escalonamento de processos distribuídos, que seja satisfatório em todas as situações. Portanto a obtenção do melhor aproveitamento do *framework* depende da análise fundamentalmente dos especialistas em engenharia da produção e ciências da computação ao modelar o cenário de produção e ao especificar os agentes, posteriormente necessitando de simulações a serem feitas pelo usuário no cenário de produção criado.

7 Resultados Obtidos

Com o objetivo de avaliar a aplicação HIPS TNT, será feito um comparativo entre o escalonamento de processos existente em uma das fases de produção e a mesma fase de produção, com seus recursos sendo escalonados pela aplicação, durante um determinado período.

No comparativo que será descrito, foram utilizadas informações reais de produção do setor de tinturaria de fios, onde constam todos os recursos produtivos deste setor e as ordens pendentes de produção, provenientes do sistema ERP.

Não é objetivo desta pesquisa garantir a eficiência da aplicação desenvolvida, utilizando o *framework* HIPS, porém foi analisada a eficiência da aplicação HIPS TNT, visando adequar funcionalidades do *framework* às necessidades da aplicação.

7.1 Definições e Variáveis

A fase escolhida para análise foi à quinta fase do cenário de produção apresentado na Figura 48, ou seja, a fase de secagem dos fios após a operação de tingimento. No cenário de produção real, esta fase possui seis recursos produtivos, porém a aplicação foi modelada com quatro recursos apenas, pois dois estavam em manutenção.

O período para análise escolhido foi de uma semana, ou seja, sete dias contados a partir de uma segunda-feira, definindo a variável $n_{dias}= 7$.

O número de ordens produção aguardando serem escalonadas nos recursos de produção, ou seja, disponíveis na fase para o agente de fase, é representado pela variável n_{OP} , cujo valor inicial é $n_{OP} = 74$.

Outras duas variáveis também são utilizadas durante o comparativo, o número de dias adiantados e o número de dias atrasados, representados por n_{ad} e n_{at} , onde:

$$n_{ad}= \sum \text{dias adiantados de todas as ordens agendadas}$$

$$n_{at}= \sum \text{dias atrasados de todas as ordens agendadas}$$

O número de dias adiantados, seria a quantidade de dias em que as ordens foram agendadas antecipadamente e o número de dias atrasados, a quantidade de dias em que as orden foram agendadas tardiamente. Um exemplo disso seria o escalonamento de uma ordem para a data atual, no qual, sua data de entrega seja dois dias posteriores. Esta ordem está dois dias adiantados, da mesma forma o atraso, no caso de ser escalonada para a data atual e sua data de entrega ser agendada para dois dias anteriores.

7.2 Comparativo

A execução inicia com n_{OP} ordens aguardando serem escalonadas na fase entre seus recursos, porém três ordens em cada recurso obedecem ao escalonamento já existente, obedecendo aos parâmetros definidos no HIPS *Architect*. Apenas no final do dia, os agentes TNTAgenteRecursos solicitaram novas ordens ao agente TNTAgenteFase.

Somente a partir da primeira solicitação de ordem, que podem ser notadas diferenças no escalonamento feito pela aplicação HIPS TNT e a otimização feita pelo AG. Essa situação também pode ser notada, na Figura 42, onde há o gráfico comparando a execução da aplicação HIPS TNT e o escalonamento normal.

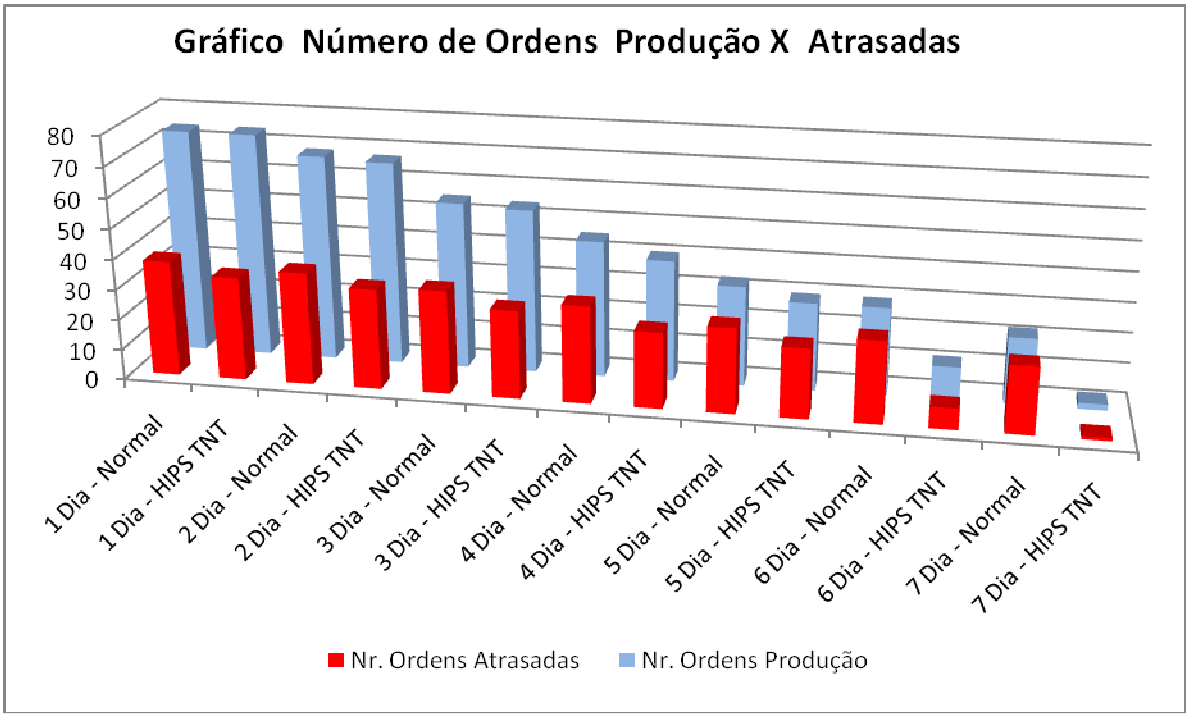

Figura 42. Gráfico Número de Ordens Produção x Atrasadas

Conforme observado, no primeiro dia de execução, praticamente não há diferenças entre o escalonamento normal e o feito pelo HIPS TNT, apenas no final do período, houve uma diferença em relação às ordens atrasadas, onde das n_{OP} iniciais para ambos, havia trinta e oito para o escalonamento normal em atraso e trinta e quatro para o escalonamento HIPS TNT.

A disparidade entre os dois escalonamento continuou crescendo até o último dia, onde pode ser notado um decrescimento contínuo no número de ordens em produção à medida que as entregas são feitas e novas ordens não são adicionadas ao setor, acompanhado da eficiência do escalonamento HIPS TNT, perante o outro, em diminuir as ordens atrasadas, escalonando ordens com prioridade maior, por primeiro.

Outro comparativo feito foi em relação ao n_{ad} e ao n_{at}, durante o período. A Figura 43 apresenta o gráfico gerado a partir dos escalonamentos normal e HIPS TNT.

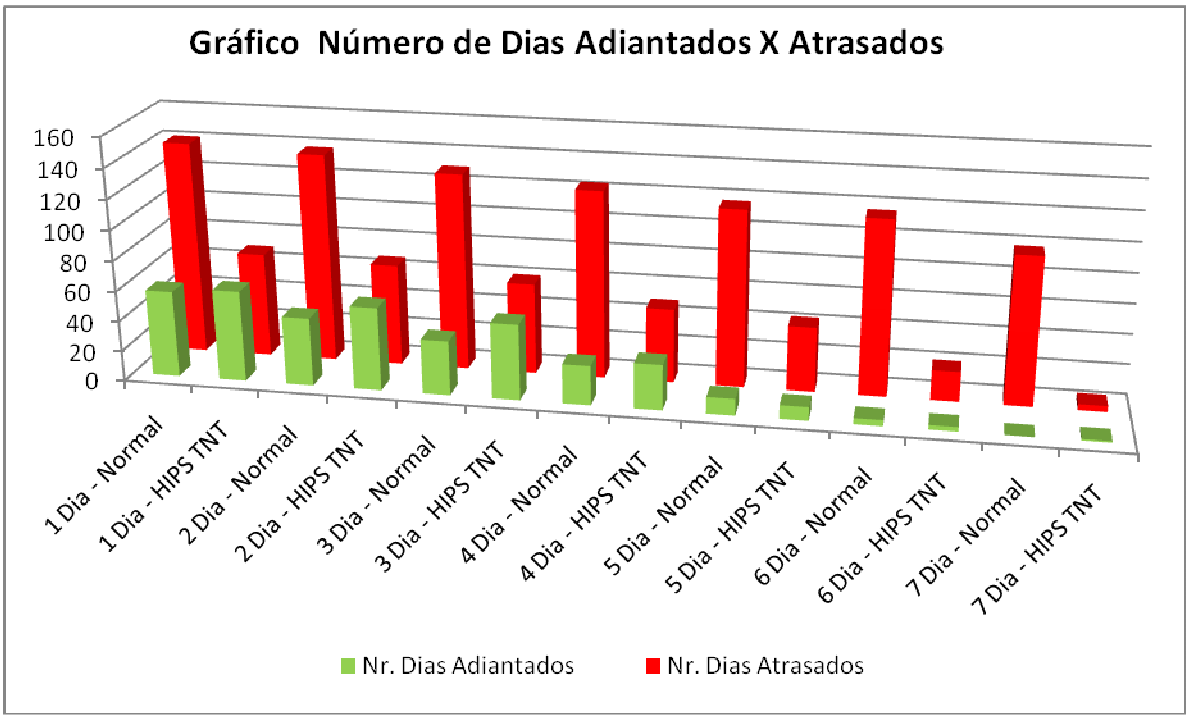

Figura 43. Gráfico Número de Dias Adiantados x Atrasados

A situação do comparativo anterior não se repete, os atrasos causados pela permanência das três ordens fixas do escalonamento atual, não interfere com igual intensidade neste, pois otimizando as demais ordens do escalonamento, foram obtidos resultados melhores de imediato, diminuindo os dias em atrasos a menos que a metade, inclusive conseguindo um aumento nos dias adiantados.

No escalonamento normal, os dias em atraso e os dias adiantados seguem uma linha regular de decréscimo, já no escalonamento HIPS TNT, pode ser notado já na metade do período, que há uma queda acentuada nos dias adiantados, buscando evitar os adiantamentos e diminuindo a quantidade de atrasos.

Quanto à alocação de tempo dos recursos, foi feito um comparativo entre os dois escalonamentos para indicar qual alocou melhor os recursos de produção disponíveis.

Determinando em tempo quase real a distribuição dos processos aos recursos de produção, o escalonamento HIPS TNT conseguiu um aproveitamento melhor de tempo, evitando paradas de processo para configuração de equipamentos. Esta diferença em relação ao escalonamento normal pode ser verificada na Figura 44, que apresenta os gráficos comparativos.

Figura 44. Gráficos do Tempo de Alocação dos Recursos

Os gráficos da Figura 44 apresentam sob duas perspectivas o tempo de alocação dos recursos de produção. Em três dos quatro recursos disponíveis, o escalonamento HIPS TNT ocupou

menos tempo, onde no recurso que alocou mais tempo, esta situação é justificada pela melhor distribuição dos processos, procurando alocar todos os recursos, o menor tempo possível, conforme definição. Poderá haver situações, onde será mais vantajoso, alocar um recurso por mais tempo do que alocar todos por menos tempo, porém as regras de pontuação devem representar está situação.

Outra análise leva em consideração o tempo total utilizado pelos escalonamentos, ou seja, a soma dos tempos de todos os recursos dos dois escalonamentos. A Figura 45 apresenta o gráfico desta comparação e destaca o excedente.

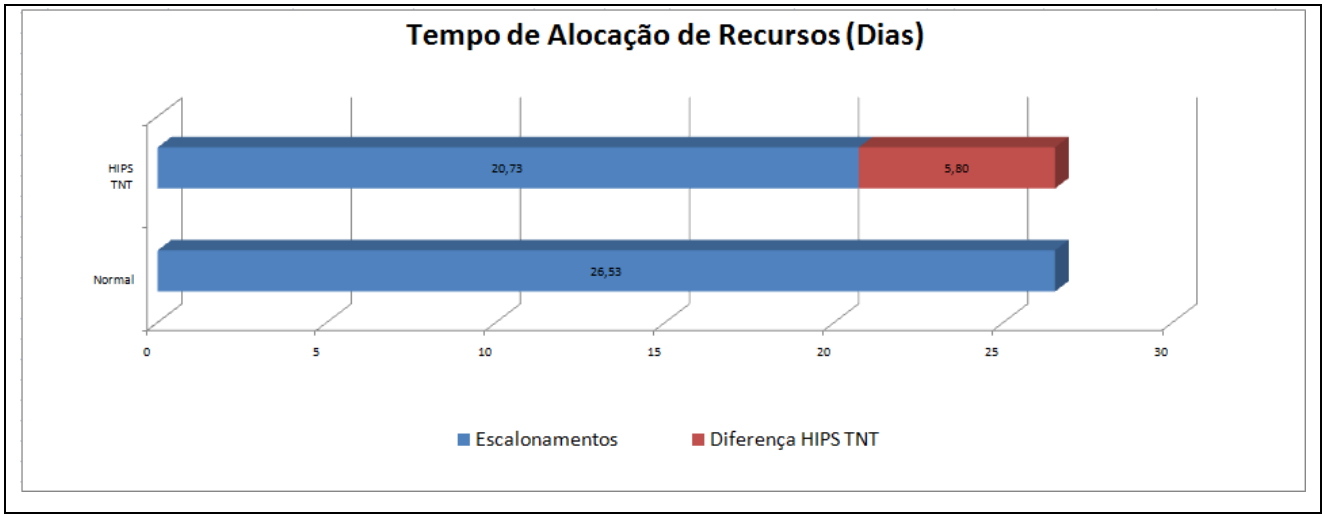

Figura 45. Gráficos do Tempo de Alocação dos Recursos Total

No gráfico da Figura 55, podem ser observadas as somatórias dos tempos de todos os recursos de cada escalonamento e em destaque, o número de dias gasto a mais pelo escalonamento normal.

Comparando diretamente o escalonamento normal e HIPS TNT, temos que o escalonamento HIPS TNT obteve um tempo total de 20,73 dias, com uma economia de tempo de 5,8 dias, o que representa um ganho de 27,98% em relação ao escalonamento normal de 26,73 dias.

Considerando que as n_{OP} ordens em produção, estavam movimentando 12.943,97Kg de fios, em termos práticos, utilizando o escalonamento HIPS TNT, poderiam ter sido secos mais 3621,72 Kg no mesmo período.

Através destes comparativos, foi possível verificar a quantidade de tempo utilizada a mais pelo escalonamento normal e a melhor distribuição dos processos entre os recursos disponíveis,

realizado pela aplicação HIPS TNT. Benefícios que o escalonamento, quase que em tempo real, com técnicas heurísticas de otimização, pode proporcionar.

Importante ressaltar que o escalonamento normal não sofreu alterações desde o início do processo, ou seja, o planejado no início do período de análise foi considerado como válido, até o fim. Intervenções por parte da supervisão da fase, ou da gerência de produção poderiam auxiliar na obtenção de valores melhores de eficiência do escalonamento normal, porém tomadas de decisão deveriam ocorrer para essa situação e ações deveriam ser executadas no sistema ERP. Contudo esse processo não é *on-line* e o cenário de produção está em constante mudança.

8 CONSIDERAÇÕES FINAIS

8.1 Conclusões

Problemas complexos geralmente costumam ser compostos de um grande número de subproblemas e variáveis, que por sua vez, podem ainda ser divididos em subproblemas, a fim de facilitar a análise e a implementação de soluções satisfatórias.

Compartilhar recursos ao longo do tempo, entre recursos concorrentes, obedecendo a regras de seqüenciamento, gerando agendamento dos processos, evitando atrasos são algumas das principais tarefas do escalonador, que dependendo do número de recursos, processos e variáveis, torna-se um problema complexo, de difícil análise, requerendo um tempo considerável para solução.

A utilização de sistemas multiagentes é uma estratégia adequada para manipular estes problemas, permitindo a definição de objetivos, papéis dos agentes e a forma como se relacionaram para atingir os objetivos.

O uso de uma metodologia para análise é de fundamental importância, ao modelar um problema utilizando sistemas multiagente, pois permite relacionar claramente os objetivos e a estrutura do ambiente, modelando a sociedade de agentes e guiando o pesquisador até o seu desenvolvimento.

Ao desenvolver o *framework* HIPS e a aplicação HIPS TNT, foi possível comprovar que a utilização de um *framework* para desenvolvimento dos agentes modelados, no caso, o JADE, possibilitou que os esforços fossem concentrados na análise do problema a ser resolvido e no planejamento de sua implementação, abstraindo-se toda a implementação da linguagem de comunicação, protocolos e mensagens e focando no desenvolvimento da solução para o problema.

O uso de Algoritmos Genéticos pelos agentes de fase permitiu ganhos na escolha de uma solução satisfatória local, evitando a análise de todo o conjunto de possibilidades ao seqüenciar os processos pendentes, que mudavam continuamente durante o ciclo de vida do SMA, sendo finalizados pelos agentes recursos ou sendo adicionados a fase. Outra vantagem importante dos AG é o tempo de processamento, possibilitando retornar um seqüenciamento satisfatório, não necessitando de uma grande quantidade de tempo, conforme necessidade dos agentes de fase.

A proposta do *framework* HIPS atingiu seus objetivos, onde ao desenvolver a aplicação HIPS TNT, foi constatado a aplicabilidade do *framework*, permitindo dividir o problema em partes, otimizando cada parte de maneira específica e satisfatória ao problema, resultando em benefícios decorrentes da interação dos agentes, na formação do escalonamento de processos ao ambiente de produção.

8.2 Trabalhos Futuros

A aplicação desenvolvida baseia-se em um sistema multiagente cooperativo, ou seja, onde todos os agentes colaboram com os demais em busca de um objetivo comum, porém outras estratégias podem ser adotadas para a busca do mesmo objetivo. Uma dessas estratégias seria a criação de um cenário de competição entre os agentes, forçando a formulação de estratégias, tomadas de decisão em proveito próprio e/ou de um grupo, por um agente em detrimento dos demais, onde os agentes passariam a ser jogadores.

Definindo os agentes como jogadores, poderia ser utilizada a Teoria dos Jogos (Fiani, 2006) para modelar o comportamento dos agentes, o que poderia mudar a forma de negociação dos agentes de fases e dos recursos e a própria dinâmica do sistema multiagente.

Utilizando a Teoria dos Jogos, uma implementação a ser analisada no escalonamento distribuído de processos, é o uso dos jogos de informação incompleta, como os leilões.

Fazendo uma analogia entre leilões e o *framework* HIPS, o conceito agente de fase poderia desempenhar o papel do leiloeiro e os agentes recursos, os arrematadores. Dentro do sistema multiagentes, em cada agente de fase, seriam feitos constantes leilões, arrematando os processos a serem executados, sendo os vencedores dos leilões, os recursos que oferecerem os melhores lances seguindo critérios do tipo: menor tempo de processamento, menor tempo de início, menor data de entrega, etc.

A aplicação HIPS TNT obteve resultados melhores que o escalonamento normalmente feito, porém novas análises podem ser feitas com outros períodos, setores, com parâmetros de escalonamento diferentes, podendo ter resultados melhores que os obtidos.

A utilização de AG como busca heurística atendeu as expectativas, porém outras técnicas tais como: *simulated annealing*, Tabu, *Ant Colony* e métodos híbridos, podem ser implementadas, permitindo um comparativo. Desta forma, o usuário do HIPS *Architect* teria opções para simular e escolher através de parâmetros, a que melhor se adéqua ao cenário de produção.

Nesta pesquisa foram citadas limitações do *framework* e da aplicação, que poderiam ser analisadas em trabalhos futuros, como por exemplo, a criação de cenários de produção com mais de um orientador, possibilitando a construção de diálogos entre os orientadores, desta forma, integrando os setores de produção, com o objetivo de melhorar a definição das prioridades de cada setor, considerando sempre, o objetivo global.

9 REFERÊNCIAS

ARAÚJO, Mário, E.M. de Melo e Castro. **Manual de Engenharia Têxtil.** Vol. 1. – Lisboa:Fundação Calouste Gulbenkian, 1986.

AZEVEDO, Israel Belo de. O prazer da produção científica: diretrizes para a elaboração de trabalhos acadêmicos. 7.ed. Piracicaba: Ed. da Unimep, 1999. 208p.

AYDIN, M. Emin, Fogarty, Terence C.. Teams of autonomous agents for job-shop scheduling problems: an experimental study. **Journal of Intelligent Manufacturing**, Londres, v.15. p.455-459, 2004. ISSN: 0956-5515.

BAKER, K. **Introduction to Sequencing and Scheduling**. Durman, USA, 1974. 318p.

BAKER, Albert. **JAFMAS - a java-based agent framework for multiagent systems.** Development and Implementation. Cincinnati: Department of Electrical & Computer Engineering and Computer Science. University of Cincinnati, 1997. Doctoral thesis.

BEAN GENERATOR. **Ontology Bean Generator**. Disponível em: < http://protege.cim3.net/cgi-bin/wiki.pl?OntologyBeanGenerator>. Acesso em: 27 jun. 2009.

BELLIFEMINE, Fábio, CAIRE, Giovanni, GREEWOOD, Dominic. **Devoloping multiagent systems with JADE**. New York: Wiley, 2007. 300p.

BIGUS, Joseph P. BIGUS Jennifer. **Constructing intelligent agents using Java**. 2. ed. New York: Wiley Computer Publishing, 2001, 432p.

BITTENCOURT, Guilherme. **Inteligência artificial**: ferramentas e teorias. 3.ed. Florianópolis: Editora da UFSC, 2006. 371 p.

BLAZEWICZ, J., Ecker, K., H., Pesch, E., Schmidt, G., Weglarz, J. “Scheduling Computer and Manufacturing Processes”, Springer-Verlag, 1996.

BONGAERTS L. **Integration of scheduling and control in holonic manufacturing systems**. Ph.D. Thesis: Katholieke Universiteit, Leuven, Belgium. 1998.

BORDINI, Rafael H.. HÜBNER, Jomi Fred, WOOLDRIDGE, Michael. **Programming multiagent Systems in AgentSpeak using Jason**. Sussex: Wiley, 2007. 292p.

BORGES, Flávio H., DALCOL, Paulo R. T**.. Indústria de processos: comparações e caracterizações.** ENEGEP. XXII Encontro Nacional de Engenharia de Produção. 2002.

BRADSHAW, J. M. **An introduction to software agents**. Ed. Software Agents. Massachusetts: MIT Press, p. 3 - 46, 1997. ISBN:0-262-52234-9

CARVALHO, Eros Moreira, RAMOS, Gabriel Silva. **Otimização por Colônia de Formigas**. Departamento de Informática, Universidade Federal do Paraná, Curitiba/PR, 2007.

CHASE, Richar B., JACOBS, Robert F., AQUILANO, Nicholas J.. **Administração da produção para a vantagem competitiva**. 10 . ed. Porto Alegre: Bookman, 2006. 724p.

CRUZ, Gisélia Magalhães ; LIMA, George . **Simulador de escalonamento para sistemas de tempo real**. In: IV WTICG - ERBASE 2006, 2006, Aracajú. IV, 2006. 111p.

DELOACH, S. A.; Wood, M. **Developing multiagent systems with agentTool**. In: Intelligent Agents VII. Agent Theories Architectures and Languages, 7th International Workshop (ATAL 2000). Vol. 1986. Springer LNCS – Verlag, Berlin, 2001.

DELOACH, S. Multiagent **Systems engineering organization-based multiagent systems**. In: 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'05). Springer LNCS, St. Louis, MO, 2006, 109-125p.

DEBONI, José Eduardo Zindel. **Modelagem orientada a objetos com a UML**. 1. ed. São Paulo: Futura, 2003. 219p.

FIANI, Ronaldo. **Teoria dos jogos com aplicações em economia, administração e ciências sociais**. Rio de Janeiro: Elsevier, 2006. 388p.

FIPA. **Foundation for intelligent physical agents**. Disponível em: < http://www.fipa.org/>. Acesso em: 01 jul. 2009.

FORMDEV. **JFormDesigner Java swing GUI designer**. Disponível em: <http://www.formdev.com/>. Acesso em: 02 jul. 2009.

GONÇALVES, J. F., Mendes, M. J. J., RESENDE, M. G. C. A hybrid genetic algorithm for the job shop scheduling problem. **European Journal of Operational Research**, 167:77–95, 2005. ISSN: 0377-2217.

GLOVER, F. LAGUNA, M. **Tabu Search**. Kluwer Academic Publishers, page 382, 1997.

GLOVER 2, F., KOCHENBERGER, G.A. **Handbook of metaheuristics**. Kluwer Academic Publishers, 2002. 574p.

GLUZ, João Carlos; VICCARI, Rosa Maria. **Linguagens de comunicação entre agentes**: fundamentos padrões e perspectivas. In: JORNADA DE MINI-CURSOS DE INTELIGÊNCIA ARTIFICIAL, n. 3, ago. 2003, Campinas. Anais do Congresso da Sociedade Brasileira de Computação, n. 23, ago. 2003, Campinas.

HOLLAND, J. H. **Adaptation in natural and artificial systems**. University of Michigan Press, 1975. 211p.

IBA, Hitoshi. **Emergent cooperation for multiple agents using genetic programming**, Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, p.32-41, September 22-26, 1996.

JAIN, A.S, MEERAN, S**. Deterministic Job-Shop scheduling**: past, present and future**. European Journal of Operational Research**. V.113, i.2, p:390-434, 1998.

JORDAN, Carsten. **Batching and scheduling**: models and methods for several problem classes.. Lecture Notes in Economics and Mathematical Systems, Springer, New York, 1996.

JADE. **Java agent development framework**. Disponível em: <http://jade.tilab.com>. Acesso em: 21 jun. 2009.

LIMA, Marcelo de Oliveira. **Usando a meta-heurística Tabu search para o planejamento de redes ópticas de telecomunicações.** Departamento de Informática – Universidade Federal do Espírito Santo (UFES), Vitória/ES, 2005.

LINDEN, Ricardo. **Algoritmos Genéticos**: uma importante ferramenta da Inteligência Computacional. Rio de Janeiro: Brasport, 2006. 372p.

MAES, Pattie. **General tutorial on software agents**. Disponível em: < http://web.media.mit.edu/~pattie/CHI97>. Acesso em: 01 jul. 2009.

MORTON, T.; PENTICO, D. W.. **Heuristic scheduling systems:** with applications to production systems and project management. John Wiley & Sons Inc., EUA, 1993. 720p.

MÜLLER, Gilberto Irajá; GOMES, Arthur Torgo. **Estratégias de escalonamento em um ambiente de Job-shop**. Disponível em: <http://www.inf.furb.br/seminco/2006/artigos/25042.pdf>. Acesso em: 27 jun. 2009.

MÜLLER, Gilberto Irajá; GOMEZ, Arthur Tórgo. **Utilização da busca Tabu para a geração de um modelo aplicado ao Job-shop scheduling problem considerando um sistema de manufatura fexível**. Universidade do Vale do Rio dos Sinos, São Leopoldo, 2006. 10p.

NIKRAZ, Magid; CAIRE, Giovanni; BAHRI, Parisa. A methodology for the analysis and design of multiagent systems using JADE. **International Journal of Computer Systems Science & Engineering special issue on Software Engineering for Multiagent Systems**. Murdoch University, Austrália, 2006.

OLIVEIRA, Ronald Lopes; WALTER, Cláudio. **Escalonamento de um Job-Shop**: um algoritmo com regras heurísticas. UFRGS, 2000.

ORACLE. **Oracle Software Company**. Disponível em: < http://www.oracle.com/index.html>. Acesso em: 02 jul. 2009.

PARSONS, Simon; WOOLDRIDGE, Michael. **Game theory and decision theory in multiagent systems.** Autonomous Agents and Multiagent Systems, 5, 243–254, 2002. Kluwer Academic Publishers, 243-254, 2002.

PARUNAK, H.V.D. **Distributed artificial intelligence systems.** Artificial Intelligence Implications for CIM. Springer-Verlag, Berlim, 225-254, 1988.

PARUNAK, H.V.D., Sauter, J., Clark, S.J.. **Toward the specification and design of industrial synthetic ecosystems**. Springer-Verlag, Berlim, 1997.

PETROWSKI, J. Dréo A.; TAILLARD, P; SIARRY, E.. **Metaheuristics for hard optimization:** Simulated Annealing, Tabu Search, Evolutionary and Genetic Algorithms, Ant Colonies,… - Methods and Case Studies. Springer: Berlin, Alemanha, 2006. ISBN 9783540309666

PINEDO, Michel L.. **Planning and scheduling in manufacturing and services**. New York, Springer, 2005.

PROTEGE. The **Protégé ontology editor and knowledge acquisition system**. Disponível em: <http://protege.stanford.edu>. Acesso em: 27 jun. 2009.

RUSSELL, Stuart; Norvig, Peter. **Inteligência artificial:** tradução 2.ed. Rio de Janeiro: Editora Campus, 2004. 1040p.

SACILE, Roberto; PAOLUCCI, Massimo. **Agent-based manufacturing and control systems**. Flórida, CRC Press LLC, 2005. 288p.

SILVA, S. C.; VARELA, M. L. R. **Classificação e descrição formal de sistemas de produção orientados ao produto na óptica do escalonamento da produção**. Publicação Interna do Departamento de Produção e Sistemas, Universidade do Minho, Portugal, 1999.

SILVEIRA, Ricardo Azambuja. **Modelagem orientada a agentes aplicada a ambientes inteligentes distribuídos de ensino:** JADE - Java agent framework for distance learning environments. Tese. Porto Alegre: Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, 2001. 126f.:il.

SHIN, Moonsoo; RYU, Kwangyeol; JUNG, Mooyoung. **A novel negotiation protocol for agent-based control architecture.** The 5th International Conference on Engineering Design & Automation, Las Vegas, 2001.

SMITH, R.; DAVIS, R. **The contract net protocol**: high level communication and control in a distributed problem solver. IEEE Transactions on Computers, v. 29, n. 12, p. 1104-1113, 1980.

SOARES, Marcio Morelli,; et. al. **Otimização do planejamento mestre da produção através de algoritmos genéticos.** XXII ENEPGEP - Encontro Nacional de Engenharia de Produção. Curitiba, 2002.

SOLARI, María de los Ángeles; OCAMPO, Ernesto. **Application of genetic algorithms to a manufacturing industry scheduling multiagent system.** Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications. Springer Netherlands, 2007. p. 263-268. ISBN 978-1-4020-6265-0.

TALUKDAR, S. **Asynchronous teams**. Proceedings of the 4th International Symposium on Expert Systems Applications to Power Systems, Latrobe University, Melbourne, Austrália, 1993.

TUBINO, Dálvio Ferrari. **Planejamento e controle da produção**: teoria e prática. São Paulo: Ed. Ática, 2007. 196p.

VARELA, Maria Leonilde Rocha. **Uma contribuição para o escalonamento da produção baseado em métodos globalmente distribuídos**. Tese. Braga: Universidade do Minho. Programa de Pós-Graduação em Produção e Sistemas, 2007. 224f.:il.

VIEIRA, Oacyr Feijó. **Controle de qualidade na indústria de fiação e tecelagem.** Vol. II. Rio de Janeiro: SENAI/CETIQT, 1988.

VLASSIS, Nikos. A **concise introduction to multiagent systems and distributed artificial intelligence**. 1. ed. Grécia: Ed. Morgan & Claypool, 2007. 71p.

VOLLMANN, Thomas E.; et. al. **Sistemas de planejamento & controle da produção para o gerenciamento da cadeia de suprimentos**. 5.ed. – Porto Alegre: Bookman, 2006. 648p.

VRBA, P. **JAVA-based agent platform evaluation**. In: HoloMAS 2003, LNAI 2744, 2003, p.47–58.

WADA, Yuji; SHIOUCHI, Masatoshi; TAKADA, Yuji. **A multiagent approach to distributed schedule management system**. Fujitsu Sci Tech.v. 33,, n.2, Japão, 1997. p.196-210.

WEICHHART, Georg. et. al. **Modelling of an agent-based schedule optimization system**. Institute of Systems Theory and Simulation. Universidade Johannes Kepler Linz, Austria, 2004.

WEISS, Gerhard. M**ultiagent systems:** a modern approach to distributed modern approach to artificial intelligence. 1. ed., Mit Press, Cambridge, EUA, 1999. 648p.

WOOLDRIDGE, Michael; JENNINGS, N. R.; KINNY, D. **The GAIA methodology for agent oriented analysis and design.** Autonomous Agents and MultiAgent Systems. Springer Netherlands.: 2000. p. 285-312.

WOOLDRIDGE, Michael. **An introduction to multiagent systems**. John Wiley & Sons, 2002. 348p.

XIN, Li. **MARSiMA - marcação automática de reuniões usando um sistema multiagente**. Mestrado em Inteligência Artificial e Computação – Universidade do Porto, 2000.

YAMADA, Takeshi; NAKANO, Ryohei. **Job shop scheduling by simulated annealing combined with deterministic local search**. Kluwer academic publishers, MA, USA, 1996. p. 237-248.

ARTIGOS ACEITOS PARA PUBLICAÇÃO

Artigos Publicados em Anais de Congressos ou Periódicos (completo)

Uber Júnior., Arnoldo ; SILVEIRA, R. A. . Using Multiagent Systems And Genetic Algorithms To Deal Staggering Problem. In: 7th International Conference on Practical Applications of Agents and MultiAgent Systems (PAAMS'09), 2009, Salamanca. 7th International Conference on Practical Applications of Agents and MultiAgent Systems (PAAMS'09) Series: Advances in Intelligent and Soft Computing .. Massachussets : Springer, 2009. v. 55. p. 567-577.

Uber Júnior., Arnoldo ; SILVEIRA, R. A. . Modelagem de um framework para escalonamento de processos distribuídos utilizando sistemas multiagentes e algoritmos genéticos. In: XV SIMPEP Simpósio de Engenharia da Produção, 2008, Baurú. Anais. Bauru : UNESP, 2008.