Universidade Federal do Rio de Janeiro

UMA ARQUITETURA MULTI-AGENTE PARA COORDENAÇÃO ROBÓTICA

Fábio de Sousa Cardoso

2015

UMA ARQUITETURA MULTI-AGENTE PARA COORDENAÇÃO ROBÓTICA

Fábio de Sousa Cardoso

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

Orientador: Prof. Max Suell Dutra, Dr.-Ing

Rio de Janeiro

Setembro de 2015

UMA ARQUITETURA MULTI-AGENTE PARA COORDENAÇÃO ROBÓTICA

Fábio de Sousa Cardoso

TESE SUBMETIDO AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA MECÂNICA.

Aprovada por:

_____
Prof. Max Suell Dutra, Dr.-Ing.

_____
Profa. Priscila Machado Vieira Lima, D.Sc.

_____
Prof. Jules Ghislain Slama, D.Sc.

_____
Fernando Castro Pinto, D.Sc.

_____
Marco Hiroshi Naka, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
Setembro de 2015

Cardoso, de Sousa Cardoso

Uma Arquitetura Multi-agente para Coordenação Robótica – Rio de Janeiro: UFRJ/COPPE, 2015.

X, 168 p. 29,7 cm.

Orientador: Prof. Max Suell Dutra, Dr.-Ing.

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia Mecânica, 2015

Referências Bibliográficas: p. 137-138.

1. Sistemas Multi-robóticos. 2. Sistemas Multi-agentes. 3. Coordenação Robótica. I. Dutra, Max Suell. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Mecânica. III. Título.

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

UMA ARQUITETURA MULTIAGENTE PARA COORDENAÇÃO ROBÓTICA

Fábio de Sousa Cardoso

Setembro/2015

Orientador: Prof. Max Suell Dutra, Dr.-Ing.

Programa: Engenharia Mecânica

Este trabalho relata o desenvolvimento de uma arquitetura multiagente para coordenação robótica. Uma investigação inicial é apresentada sobre sistemas multiagentes e suas similaridades com aplicações multi-robóticas. A arquitetura proposta seguindo uma abordagem independente de domínio é descrita com detalhes, destacando os aspectos inovadores da pesquisa. As estratégias usadas para validação da arquitetura proposta são descritas, assim como as ferramentas utilizadas. As validações realizadas são apresentadas, juntamente com a análise crítica dos resultados. Os resultados experimentais observados, a partir das simulações realizadas, demonstraram a viabilidade da aplicabilidade da arquitetura proposta em cenário de coordenação multi-robótica, atuando especificamente na camada deliberativa de tais sistemas. Por fim, novas perspectivas e possibilidades são apresentadas para continuidade de investigações que podem contribuir para geração outras inovações nesta linha de pesquisa.

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

MULTI-AGENT ARCHITECTURE FOR ROBOTICS COORDINATION

Fábio de Sousa Cardoso

September/2015

Advisor: Prof. Max Suell Dutra, Dr.-Ing.

Department: Mechanical Engineering

This paper reports the development of a multi-agent architecture for robotics coordination. Initial investigation is presented on multi-agent systems and their similarities with multi-robotic applications. The proposed architecture following an independent approach domain and it is described in details, highlighting the innovative aspects of the research. The used strategies for validation of the proposed architecture are described, as well as the tools used. The validations are presented along with critical evaluation of the results. The experimental results observed on simulations demonstrate the feasibility of the applicability of the proposed architecture in multi-robot coordination scenarios, specifically acting in the deliberative layer of such systems. Finally, new perspectives and possibilities are presented for continuing the investigations, what may contribute to generate innovations in this research.

Sumário

Abreviaturas

BMS: São as iniciais de sistemas de produção ou de manufatura biônicos (da expressão inglesa *Bionic Manufacturing System*).

AGV: São as iniciais de veículo guiado automaticamente (da expressão inglesa *Automated Guided Vehicle*).

CIM: São as iniciais de manufatura integrada por computador (da expressão inglesa *Computer Integrated Manufacturing*).

CNC: São as iniciais de controle numérico computadorizado (da expressão inglesa *Computer Numeric Control)*.

FMC: São as iniciais de célula de manufatura flexível (da expressão inglesa *Flexible Manufacturing Cell*).

SFM: São as iniciais de sistema flexível de manufatura (da expressão inglesa *Flexible Manufacturing System*).

JADE: Plataforma de desenvolvimento de sistemas multi-agentes (*Java Agent DEvelopment framework*).

AMS: Agente da plataforma JADE (*Agent Manager System*).

DF: Agente da plataforma JADE (*Directory Facility*).

PROSA: São as iniciais da arquitetura de referência que emprega agentes recursos e produtos (da expressão inglesa *Product Resource Order Staff reference Architecture*).

SMA: Sistemas Multi-agentes ou do inglês *Multi-Agent System*.

CAD: São as iniciais de projeto auxiliado por computador (da expressão inglesa *Computer-Aided Design*).

CAM: São as iniciais de manufatura auxiliada por computador (da expressão inglesa *Computer-Aided Manufacturing*).

CAPP: São as iniciais de planejamento de processo auxiliado por computador (da expressão inglesa *Computer-Aided Process Planning*).

CAE: São as iniciais de engenharia auxiliada por computador (da expressão inglesa *Computer-Aided Engineering*).

CAQ: São as iniciais de garantia da qualidade auxiliada por computador (da expressão inglesa *Computer-Aided Quality Assurance*).

PPC: São as iniciais de planejamento e controle da produção (da expressão inglesa *Production Planning and Control*).

ERP: Termo usado para sistema de gestão empresarial (da expressão inglesa *Enterprise Resource Planning*).

VANTs: Iniciais da expressão veículos aéreos não tribulados (do inglês *Unmanned Aerial Vehicles* - UAVs).

CNP: São as iniciais do protocolo de negociação de sistemas multi-agentes (do inglês *Contract Net Protocol*).

RSFs: São as iniciais utilizadas para designar as plataformas robóticas de software (da expressão inglesa *Robotics Software Frameworks*).

MARS: São as iniciais de sistemas multi-agentes robóticos (da expressão inglesa *Multi-Agent Robotic System*).

DARS: São as iniciais de sistemas de robôs autônomos distribuídos (da expressão inglesa *Distributed Autonomous Robot Systems*).

ACL: São as iniciais de linguagem de comunicação de agentes (do inglês *Agent Communication Language*).

Capítulo 1 – Introdução

Neste capítulo são apresentados os problemas objetos desta pesquisa, os conceitos relacionados aos desafios de sistemas distribuídos com aplicações na manufatura, de sistemas distribuídos robóticos e de sistemas multi-agentes. Adicionalmente, serão apresentados os objetivos gerais e específicos do trabalho, a motivação e a organização do documento com um breve descritivo dos capítulos.

Em sistemas industriais de grande porte os gestores de sistemas de produção deparam-se com grandes desafios devido, não só ao aumento da complexidade de processos e produtos, mas também ao aumento da variedade desses mesmos produtos, que devem ser colocados no mercado de forma cada vez mais rápida devido à redução do ciclo de vida dos produtos.

Há um ambiente de expansão da rede de fornecedores e distribuidores num mercado globalizado. As empresas que desejam sobreviver têm que modernizar os seus métodos tornando-os mais ágeis e flexíveis, adaptados às mudanças de cenários onde estão inseridos. A tendência para personalizar produtos conduz à necessidade de aumentar a integração entre a indústria e os consumidores, para que toda a cadeia produtiva seja gerida em conformidade com a necessidade e procura destes consumidores. Percebe-se que as empresas estão sujeitas a uma constante pressão, no sentido de proporcionarem mais opções de escolha, reduzindo o ciclo de vida dos produtos e inovando-os. Para isso, as organizações industriais precisam mostrar-se mais inteligentes, de forma a conseguirem responder à dinâmica do mercado.

Algumas alternativas de arquiteturas de sistemas de produção surgiram a fim de responder a tais necessidades empregando um modelo distribuído de produção, com unidades cooperativas e autônomas. Estas propostas enquadram-se nos seguintes paradigmas (Tharumarajah *et al.*, 1996 e 1998), nomeadamente: Sistemas de Produção Biônicos (BMS – *Bionic Manufacturing System*); Sistemas de Produção Holônicos; Sistemas de Produção Baseados em Fractais e Sistemas Distribuídos de Produção.

Os sistemas biônicos de manufatura possuem como características a autonomia, o comportamento espontâneo e harmonia social hierárquica, semelhante aos sistemas biológicos. As células são unidades básicas que formam os sistemas biológicos. As células são semelhantes, no entanto diferenciadas por função e com capacidades de

diversas operações. As células aglomeradas formam tecidos, estruturadas em camadas hierárquicas formam os órgãos e os sistemas, até a formação de organismos complexos. A estabilidade química dos organismos é conseguida através da regulação da taxa de reações metabólicas. Nas células isto é feitos por enzimas, que agem no controle acelerando ou inibindo as reações, e por hormônios que são secretados e transportados pelos fluídos corporais para outras partes do corpo.

Os sistemas biológicos possuem similaridades aos sistemas de manufatura visto que as unidades de produção precisam de entradas para executar as operações e as saídas são devolvidas ao ambiente produtivo. Coordenadores e controladores atuam como as enzimas e os hormônios. As unidades de manufatura podem também apresentar estrutura hierárquica semelhante às células. Cada camada na hierarquia suporta e é suportada pelas camadas adjacentes.

Os fractais possuem como principal característica a auto-similaridade, visto que eles apresentam detalhes semelhantes em pequena e larga escala. Os sistemas de produção baseados em fractais são compostos de componentes menores ou entidades fractais, com características de auto-organização ou capacidade de organizar e de executar suas tarefas livremente, de adaptação às influências do meio ambiente e de similaridade de objetivos entre os fractais, além da necessidade que os fractais funcionem como um conjunto coerente.

A crescente complexidade dos sistemas de produção, em conjunto com a procura por processos de produção mais flexíveis e tolerantes a falhas, tem estimulado o desenvolvimento de sistemas de controle descentralizados, como é o caso dos sistemas de produção holônicos. Tais sistemas são baseados em unidades produtivas, denominadas *Holons*, que fazem parte de uma organização denominada holarquia. *Holons*, por sua vez, podem ser compostos por outros *Holons* (subsistemas) que formam o sistema de produção. Cada *Holon* tem características do todo, visto que pode ser visto como um sistema composto por outros *Holons*, e da parte, uma vez que tem parte da funcionalidade do sistema. Van Brussel *et al.* (1998) apresentaram aquela que é uma das mais citadas arquiteturas de referência, *Product Resource Order Staff reference Architecture* (PROSA), para os sistemas de produção holônicos, baseado em *Holons* "produto", "encomenda" e "recurso".

O paradigma holônico de manufatura busca a flexibilidade e a agilidade nos processos produtivos, tentando aplicar os benefícios que as organizações holônicas proporcionam em organismos vivos e em sociedades. Alguns benefícios deste

paradigma são a estabilidade, a adaptabilidade, a flexibilidade em face de distúrbios e o uso eficiente dos recursos disponíveis. Os sistemas holônicos tendem a preservar a estabilidade da arquitetura hierárquica juntamente com a flexibilidade e adaptabilidade da arquitetura heterárquica.

O *Holon* no contexto do sistema de manufatura, é um bloco autônomo e cooperativo que transforma, transporta, armazena e valida informação ou objetos físicos. A autonomia é a capacidade desta entidade de criar e de controlar a execução de seus planos e suas estratégias. A holarquia é um sistema composto de *Holons* que podem cooperar para atingir uma meta ou objetivo. Devem ser definidas regras a fim de permitir a cooperação dos *Holons.*

Estes paradigmas apresentam conceitos coincidentes que podem ser traduzidos em sistemas de produção caracterizados por unidades distribuídas e autônomas cooperando para gerir processos de produção. Em outras palavras, uma estrutura potencial para um sistema distribuído de produção é composta por um conglomerado de unidades autônomas, que operam em cooperação de forma integrada. Tais unidades podem ser máquinas de automação industrial, robôs manipuladores, robôs móveis ou unidades remotas microprocessadas.

Em uma visão computacional, Tanenbaum (2003) define sistema distribuído como similar a uma rede de computadores, onde cada computador tem sua própria memória privada, seus próprios sistemas de arquivos e podem possuir diferentes sistemas operacionais. Há a necessidade de uma camada de software, nomeada de *middleware*, para permitir comunicação entre os elementos da rede.

Os sistemas distribuídos de produção, muitas vezes compostos por máquinas robotizas, são concebidos com o objetivo de propiciar eficiência e racionalidade na utilização dos recursos distribuídos de produção, de modo a favorecer a fabricação de produtos, de uma forma dinâmica e rápida. As unidades de produção devem ter capacidade de responder, de maneira inteligente e eficaz, a distúrbios não previsíveis do ambiente externo, mantendo uma produção controlada e contínua.

Considerando a necessidade de sistemas de planejamento e controle da produção adaptáveis à utilização local e distribuída de recursos de produção e materiais, torna-se necessário criar um modelo para seleção, alocação e operação dos recursos de produção, para uma resposta rápida e eficaz à procura. Uma alternativa é a utilização de sistemas multi-agentes. Dada essa tendência de completa robotização dos sistemas produtivos e de suas características distribuídas, vislumbrou-se a necessidade do desenvolvimento de

uma arquitetura robótica multi-agente que pudesse atender a demanda de problemas similares a este, relacionado a planejamento e controle de sistemas produtivos distribuídos trabalhando sob encomenda, e outros mais relacionados à robótica distribuída. Desta forma, a tecnologia multi-agente será explorada neste trabalho com a finalidade da criação de uma arquitetura genérica, ou independente de domínio, de um sistema multi-robótico multi-agente.

Um agente é um sistema situado em algum ambiente e com capacidade de ações autônomas para atingir seus objetivos (Vidoni, 2011). Conforme Vidoni, no modelo multi-agente a produção responde a incertezas por meio de um controle descentralizado e, diferentemente de uma manufatura integrada por computador, não há necessidade de centralização e um processo de gerenciamento único e complexo. São necessárias as fases de comunicação e negociação entre os agentes para o uso e controle das máquinas, recursos e materiais. As principais vantagens desta tecnologia, segundo Vidoni, são:

- Decisões descentralizadas e distribuídas, onde cada agente possui certa autonomia;
- Estrutura modular com agentes independentes.

Sistemas Multi-Agentes (SMA), ou do inglês *Multi-Agent Systems* (MAS), consistem em múltiplas entidades chamadas de agentes que interagem em um ambiente compartilhado visando atingir objetivos individuais e coletivos (Uhrmacher *et al.*, 009).

Os sistemas multi-agentes consistem de grupos de agentes, que exibem um comportamento autônomo, mas ao mesmo tempo interagem com os outros agentes presentes no sistema. Um agente de software pode ser conceituado como uma entidade que funciona de forma continua 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.

A área de sistemas multi-agentes (SMA) estuda o comportamento organizado de agentes autônomas cooperativos para solução de problemas que, por vezes, estão além de suas capacidades individuais (Hübner, 2003). Para Hübner, a autonomia é a capacidade do agente de existir independentemente dos outros agentes e do problema. A organização limita o comportamento do agente a fim se buscar um comportamento coeso do grupo de agentes. O autor cita algumas características de sistemas multi-agentes:

- Permitem o desenvolvimento de sistemas adaptativos e evolutivos, eliminando e/ou incluindo novos agentes;

- Podem ser empregados em sistemas complexos e distribuídos, permitindo altos níveis de abstração;
- Aproveitando ambientes heterogêneos e distribuídos, agentes podem interagir, mesmo funcionando em diferentes plataformas.

Um possível modelo para sistema de manufatura robotizado assume que agentes de "produção" (robôs manipuladores ou móveis ou máquinas industriais) têm conhecimento suas capacidades para executar determinadas tarefas de transformação e de suas agendas. Considerando tal conhecimento, o agente obtém ordens de produção através de um sistema distribuído de planejamento e controle da produção. A informação a respeito de recursos de produção disponíveis, e que satisfaçam uma determinada procura, pode ser valiosa para o sistema. A partir de visão da capacidade e da disponibilidade de cada unidade produtiva, os sistemas devem ser capazes de enviar e controlar ordens de produção.

Os agentes obtêm suas tarefas, através de um sistema distribuído em que as tarefas são negociadas e controladas (monitoradas) num ambiente virtual. Quando alocadas aos recursos as tarefas são monitoradas quando em operação. Esse modelo, proposto nesta pesquisa, pode ser empregado em robótica colaborativa, tornando-se um arcabouço para desenvolvimento de diversas aplicações.

Para Hübner (2003) diferentes tipos de problemas podem demandar dos agentes diferentes escolhas de modelos. Há certa independência entre a concepção dos modelos e o problema. Não necessariamente os modelos são desenvolvidos para solucionar um problema em particular. Desta forma, nesta pesquisa buscou o desenvolvimento de um modelo ou arquitetura genérica a fim de atingir um maior número de problemas, abrindo um leque de opções de aplicações e futuros trabalhos.

Agentes dentro de SMA podem agir 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. Para haver a cooperação é necessário que haja comunicação entre os agentes. A comunicação ocorre através da troca de mensagens entre os agentes na rede. Protocolos de comunicação são assim estabelecidos, assim como estratégias cooperativas para solução de problemas.

A robótica tem direcionado sua evolução para área de sistemas móveis autônomos, tendo como desafios o desenvolvimento de sistemas colaborativos e autônomos.

Para Pérez (2010), os manipuladores utilizados em sistemas flexíveis de manufatura sofreram grande evolução no passado desempenhando tarefas repetitivas, no entanto, possuem desvantagens por possuírem estrutura de movimentos limitados, diferentemente dos robôs móveis que podem realizar movimentos ao longo da fábrica, em muitos casos evitando obstáculos. Os robôs móveis podem ser empregados em diferentes ambientes e envolvem várias áreas de pesquisa, tais como, mecatrônica, mecânica, sensoriamento, comunicação, navegação e aperfeiçoamento de movimentos. Conforme o autor, as aplicações dos robôs móveis vão além da indústria, chegando a áreas como exploração subaquática, oceanográfica, exploração planetária, além de aplicações militares e de logística.

A robótica tem como uma importante área a criação de robôs autônomos, com potencialidade de aceitar comandos em alto nível, sem a necessidade de intervenções humanas. A tecnologia de robôs autônomos requer o desenvolvimento do raciocínio automatizado, percepção e controle, além de outros desafios importantes como planejamento de trajetórias (Ottoni *et al.* , 2003).

Segundo Pérez (2010), os sistemas AGVs, do inglês *Automated Guided Vehicles*, são apropriados para manipulação de material em ambientes produtivos automatizados. Tais sistemas consistem em conjuntos de veículos sem condutor, visando transporte de materiais entre estações de trabalho e locais de armazenamento. Em alguns casos são utilizados caminhos predefinidos incorporados no *layout* da fábrica. O controle dos AGVs pode ser centralizado ou distribuído e a movimentação baseada em técnicas de navegação que fazem uso de sensores de diversos tipos.

Em uma pesquisa realizada por Roberto (2011) foi desenvolvido um sistema de gerenciamento de tráfego para frotas de AGVs operando em ambiente industrial. Os veículos eram utilizados para transporte de materiais em um ambiente em que havia pessoas e máquinas se movimentando. O sistema, nomeado TMS, controlava o movimento dos veículos visando evitar colisões e aperfeiçoar o tempo de transporte. O método criado é baseado na decomposição do problema em três partes: atribuição de tarefas de transporte, computação do caminho e controle de navegação de cada AVG ao longo do seu caminho. Ordens de transporte são atribuídas para os veículos disponíveis através da solução de um problema de atribuição linear fazendo uso de algoritmo de

Volgenant Jonker, a fim de minimizar a soma dos tempos requeridos de cada veículo para chegar à estação atribuída ao mesmo. O movimento dos veículos é definido de forma incremental por um controlador que computa periodicamente uma lista de segmentos de trajetórias que cada AGV pode cobrir. A sequência de segmentos é determinada de forma a minimizar o tempo nominal do percurso. Com a finalidade de definir o movimento dos veículos ao longo suas rotas, uma ferramenta chamada Diagrama de Coordenação, do inglês *Coordination Diagram* (CD), foi aplicada. O problema passa de uma dimensão de N AGVs para um problema de percursos dentro de um espaço de n dimensões com obstáculos.

Os sistemas flexíveis de manufatura, da expressão inglesa *Flexible Manufacturing System* (SFM), consistem em grupos de estações de trabalho interconectadas por meio de sistemas de armazenamento e de manipulação de material automatizados e controlados por sistemas integrados por computador (Shivanand *et al.*, 2006). A flexibilidade de tais sistemas é devido à capacidade de processar uma variedade de diferentes tipos de partes na estação de trabalho, além da quantidade de produção poder ser ajustada considerando a demanda. Os sistemas de controle computadorizados são usados para coordenar as atividades das estações de trabalho e nos sistemas de manipulação de material em SFMs. Algumas das funções destes sistemas são:

- Controle de cada estação de trabalho;
- Distribuição de instruções de controle para estações de trabalho;
- Controle da produção;
- Controle de tráfego;
- Monitoramento de desempenho do sistema e relatórios.

A flexibilidade dos sistemas SFM pode classificada da seguinte forma:

- Flexibilidades básicas:
    - Flexibilidade de máquinas para processar várias operações;
    - Flexibilidade em manipular materiais, neste caso diferentes tipos de partes podem ser transportadas e posicionadas em diferentes máquinas;
    - Flexibilidade no sequenciamento de operação para processar um item produtivo;
- Flexibilidades do sistema:

    - Flexibilidade de processamento de diferentes volumes produtivos eficientemente;
    - Flexibilidade de expansão do sistema de forma incremental;
    - Flexibilidade de rotas para cada peça no processo produtivo;
    - Flexibilidade no processo para que o sistema possa produzir um conjunto de tipos de itens ou peças sem incorrer em tempo de configuração adicional;
    - Flexibilidade de produzir um conjunto de peças com mínimo tempo de preparação ou configuração.
- Flexibilidades agregadas:
    - Flexibilidade do sistema em rodar por longos períodos sem intervenção externa;
    - Flexibilidade do sistema em produzir um conjunto de peças sem maiores investimentos de capital;
    - Flexibilidade do sistema eficientemente se adaptar as mudanças das condições e demandas do mercado.

A tendência dos SMF vai em direção a pequenas versões de SFM tradicionais, chamadas de células flexíveis de manufatura (CMF), do inglês *Flexible Manufacturing Cells* (FMC). Duas ou mais máquinas de comando numérico computadorizado (CNC) podem ser consideradas como uma CMF.

Um sistema flexível de manufatura é um sistema que possui flexibilidade a mudanças inesperadas e com capacidade de processar conjuntos de peças, sendo, em geral, formado de máquinas CNC conectadas por um sistema de movimentação de material, em muitos casos, sistemas robóticos (Shivanand *et al*., 2006). Adicionalmente, um sistema de controle computadorizado equaliza o movimento dos materiais e o andamento das operações.

Os SFMs quando bem implantados dão alta flexibilidade em gerenciar os recursos produtivos para produzir novos produtos, o que é bem aderente aos requisitos exigidos para produção de pequenos lotes de produtos ou produção sob encomenda.

Em um SFM é comum se ter um computador central controlando o conjunto de máquinas e o sistema de transporte que as interconecta, formando uma configuração conhecida como manufatura integrada por computador ou CIM (*Computer Integrated Manufacturing*). CIM é uma abordagem de fabricação em que são usados computadores

para efetuar o controle completo da produção, permitindo maior rapidez e menos erros produtivos.

Uma manufatura integrada por computador, normalmente, utilizada outros ferramentas de software, tais como: os sistemas de software para auxílio ao projeto ou CAD (*Computer-Aided Design*), sistemas de software para auxílio à manufatura ou CAM (*Computer-Aided Manufacturing*), máquinas de comando numérico ou CNC (*Computer Numerical Control*) e robôs a fim de criar células de trabalho completamente automatizadas.

As células de trabalho dos SFMs podem realizar várias operações, desde a montagem parcial de uma peça até a montagem completa do produto sem interação humana. Os robôs podem executar várias tarefas como a seleção de matéria-prima, a alimentação da máquina com a matéria-prima selecionada, a seleção da ferramenta e a operação da máquina, a remoção da peça processada, a colocação da peça em outra máquina e o armazenamento da peça acabada.

Uma manufatura integrada por computador não necessariamente é uma fábrica independente de intervenção humana, apesar de ser um grande passo nesta direção. Uma CIM aplicada a um SMF pode-se obter um sistema com flexibilidade para rapidamente produzir diferentes produtos e com volumes diferentes (Shivanand *et al.*, 2006). Além de sistemas CAD e CAM outros sistemas podem estar presentes em uma manufatura integrada por computado, tais como: CAE (*Computer-Aided Engineering)*, CAPP (*Computer-Aided Planning*), CAQ (*Computer-Aided Quality Assurance*), PPC (*Production Planning and Control*) e ERP (*Enterprise Resource Planning*).

Na arquitetura proposta, operadores humanos, representados por agentes, podem adicionar tarefas ao sistema. As tarefas são distribuídas aos agentes recursos, neste caso, robôs, AGVs, ou células produtivas, através de um protocolo pré-estabelecido, uma extensão do protocolo conhecido como *Contract Net Protocol* (CNP).

De acordo com Smith (1980), o CNP foi desenvolvido para resolver problemas de comunicação e controle para nós em um sistema distribuído. As tarefas são distribuídas a partir de um processo de negociação. Este processo facilita o controle distribuído da execução das tarefas cooperativas, ou divisão de tarefas.

Na arquitetura proposta, agentes recursos podem publicar seu estado ou responder quando solicitados. A arquitetura permite a visualização de todas as mensagens do sistema ou de um dado agente recurso. As tarefas podem variar e os agentes recursos irão responder conforme suas disponibilidades e suas habilidades. Em

caso de problemas na execução de alguma tarefa por parte de um agente recurso, suas tarefas agendadas devem ser negociadas entres os outros agentes recursos.

A arquitetura multi-agente concebida nesta pesquisa pode servir como plataforma para testar diversas aplicações, além da aplicação em manufatura flexível robotizada, como por exemplo, para coordenação de veículos aéreos não tribulados (VANTs), do inglês *Unmanned Aerial Vehicles* (UAVs). A arquitetura pode também ser expandida e reusada para testar algoritmos de controle distribuído.

1.1 Motivações

Apesar da evolução da informática e dos sistemas de gestão organizacionais industriais, ERP's (*Enterprise Resource Planning*), há considerável dificuldade de integração com outros sistemas de suporte, tais como: Sistemas CAD (Computer-Aided Design), sistemas CAM (*Computer-Aided Manufacturing*) e sistemas CAPP (*Computer-Aided Process Planning*).

Um sistema CAM deve se preocupar com o uso de sistemas computacionais para planejar, gerenciar e controlar operações de manufatura, através de interação com os recursos de produção.

A utilização de tais sistemas trás grandes benefícios, resultando em redução de custos e de tempos considerando as etapas de projeto, de planejamento do processo e da manufatura. No entanto, a integração efetiva entre estas etapas é um requisito indispensável para atingir níveis de eficiência desejáveis e comportamento ágil frente às perturbações inesperadas que podem ocorrer. A integração centrada no projeto do produto, devido ao sistema CAD, permite que haja fluxo de informação entre o sistema CAD e os sistemas CAM e CAPP a fim de permitir que, por exemplo, problemas (de materiais ou de processo) detectados no ambiente produtivo sejam realizados para fase de projeto.

Esse trabalho de pesquisa também busca o desenvolvimento de alternativas inteligentes e autônomas de integração entre esses sistemas empregados em industriais, visto que a arquitetura proposta pode ser empregada como elemento de integração entre estes sistemas: CAD, CAPP e CAM.

Há vários obstáculos para a efetiva integração entre os sistemas citados anteriormente, tais como:

- Incompatibilidade de software e de hardware entre os sistemas;

- Diferentes representações dos produtos, dos recursos e dos planos de processo;
- Os sistemas não modulares e de difícil integração;
- E os sistemas na maioria dos casos não são amigáveis e são de difícil manutenção.

Um novo direcionamento para esses sistemas é o foco na integração e o emprego de técnicas de inteligência artificial, com o uso de conceitos para buscar soluções ótimas para sequência das operações e sugestionar sequências alternativas de operações. Os sistemas distribuídos de produção baseados em agentes, que fazem parte do contexto desta pesquisa, parecem permitir a integração dos processos de planejamento e de manufatura, permitindo uma abordagem cooperativa e atendendo aos requisitos da engenharia concorrente.

Um sistema de manufatura distribuído que integra o projeto com a manufatura baseado na Web foi apresentado por Smith e Wright (2001). O sistema, chamado de Cybercut, consiste de três módulos: um módulo CAD via páginas web; um módulo CAPP; e um CNC que recebe informações de planejamento e de projeto e executa atividades de baixo nível (como por exemplo, usinagem).

Outras iniciativas de integração entre projeto, planejamento e manufatura têm surgido, e o paradigma de agentes começa a despontar como uma alternativa promissora. Assim, uma motivação para essa pesquisa é a evolução deste paradigma no contexto de sistemas flexíveis de manufatura e de aplicações diversas de robótica distribuída.

Agentes de software podem formar uma comunidade de agentes de manufatura, trabalhando para fluidez dos processos de projeto, de planejamento de processo e de manufatura, garantido qualidade do produto, agilidade no processo e informações efetivas aos clientes e envolvidos no processo.

O estudo e o emprego de sistemas multi-robóticos no contexto de sistemas flexíveis de manufatura, sistemas distribuídos de produção e sistemas multi-agentes, também são elementos motivadores desta pesquisa. Sistemas multi-robóticos cooperativos heterogêneos podem ser aplicados em várias situações, incluindo ambientes fabris. Um desafio importante nestes sistemas é a escolha adequada de qual robô deve realizar uma determinada tarefa, considerando a capacidade, a aptidão e a agenda de cada entidade robótica. A alocação e realocação de tarefas entre os robôs frente às perturbações é outra preocupação alvo de várias pesquisas.

Outro elemento motivador para essa pesquisa é no número elevado de possibilidades do uso da arquitetura proposta em diversas aplicações, tais como:

- Alocação, coordenação de robôs móveis (aéreos, terrestres ou aquáticos) em missões de exploração e de monitoramento;
- Coordenação de recursos distribuídos, como o caso de veículos para transporte de passageiros que devem atender a demanda contínua de clientes;
- Aplicações em operações de guerra em que há diversas missões sendo realizadas por recursos distribuídos;
- Missões de entrega de encomendas por VANTs - veículos aéreos não tribulados (do inglês *Unmanned Aerial Vehicles* - UAVs).

1.2 Enquadramento e Objetivos

O mundo está em constante mudança e o meio das empresas é cada vez mais turbulento. As exigências do mercado são cada vez maiores e mais difíceis de satisfazer. É necessário que surjam alternativas aos tradicionais sistemas produtivos e as aplicações distribuídas sob controle centralizado. Novos desafios são colocados a tais sistemas distribuídos, o que conduz ao levantamento de novos paradigmas de coordenação distribuída de atividades, operações ou missões. Os sistemas de recursos distribuídos deverão ser capazes de satisfazer especificações cada vez mais personalizadas de cada cliente, no menor tempo possível. O cliente quer que o seu produto tenha determinadas especificações, que podem ser diferentes de cliente para cliente.

É necessário investir em pesquisas na área dos sistemas de recursos distribuídos, para que, o quanto antes se encontre alternativas com capacidades reais de satisfazer as exigências principalmente industriais.

Os sistemas multi-agentes ou sistemas compostos por múltiplos agentes que exibem um comportamento autônomo, mas ao mesmo tempo interagem com os outros agentes, são apontados como uma possível solução para satisfazer essas atuais exigências do mercado.

Um agente de software é como uma entidade que funciona continuamente e autonomamente em um ambiente, geralmente habitado por outros agentes, e com capacidade de interação, de forma flexível e inteligente, sem requerer intervenção

humana constante. Agentes dentro de SMA podem cooperativamente ter objetivos globais. O uso de um SMA é justificado quando uma decisão distribuída é necessária. Para haver a cooperação é necessário que haja comunicação entre os agentes. A comunicação ocorre através da troca de mensagens entre os agentes. Protocolos de comunicação são assim estabelecidos, assim como estratégias cooperativas para solução de problemas diversos.

Um estudo sobre as ferramentas, os arcabouços e linguagens foi alvo desta pesquisa considerando o impacto no desenvolvimento e nos mecanismos de cooperação do SMA proposto. Os arcabouços para desenvolvimento de sistemas baseados em agentes JADE (ou *JAVA Agent DEvelopment Framework*) e MaDKit (ou *Multi-agent Development Kit*) se destacaram neste estudo. O JADE é um arcabouço de software completamente desenvolvido em linguagem Java, utilizado para o desenvolvimento de sistemas multi-agentes através de uma camada de software intermediária ou do inglês *middleware* e que atende as especificações da *Foundation for Intelligent Physical Agents* (FIPA). O JADE emprega um conjunto de ferramentas gráficas que suportam o desenvolvimento de SMAs. Um SMA baseado em JADE pode ser distribuído em diversas máquinas, mesmo com diferentes sistemas operacionais. As configurações do SMA podem ser controladas por meio de uma interface gráfica e podem ser modificadas em tempo de execução, permitindo a movimentação de um agente de uma máquina para outra. A MaDKit é uma biblioteca Java que permite o desenvolvimento e a simulação de SMA.

O desenvolvimento de um sistema que trate uma elevada quantidade de variáveis, restrições e situações, é uma tarefa de grande complexidade. A alocação automática e dinâmica dos recursos é também um desafio, sendo também foco desta pesquisa.

A robótica é uma área de pesquisa em que o paradigma multi-agentes pode ser muito útil. Os agentes podem ser robôs cooperativos e redes de sensores baseados em robôs móveis. Conforme Iñigo-Blasco *et al.* (2012), os sistemas robóticos multi-agentes possuem ainda grandes desafios. Na década passada várias plataformas robóticas, ou do inglês *Robotics Software Frameworks* (RSFs), sugiram em resposta aos problemas recorrentes na área da robótica. Algumas, tais como ROS, YARP, OROCOS, ORCA, Open-RTM e Open-RDK, possuem a infraestrutura necessária para o desenvolvimento de sistemas multi-agentes robóticos (do inglês, *Multi-agent Robotic System* – MARS).

Este trabalho enquadra-se numa área de estudo de conceitos, métodos e modelos para sistemas multi-agentes aplicados a sistemas robóticos distribuídos e que possuem como entrada encomendas de tarefas, operações ou missões. A pesquisa também abrange, considerando similaridade do modelo genérico proposto, a área de arquitetura de sistemas produtivos multi-agentes.

Há um grande número de aplicações abrangendo robôs autônomos para várias situações. Os robôs autônomos podem ser suficientes para missões para os quais foram construídos, com suas capacidades e funcionalidades desenvolvidas. No entanto, os MARSs (*Multi-Agent Robotic Systems*), através da interação dos robôs e da introdução de regras ou comportamentos cooperativos, podem expandir suas capacidades globais. Uma alternativa para desenvolvimento destes modelos cooperativos organizacionais é o desenvolvimento de sistemas de gerenciamento externos aos robôs. Tais sistemas podem decompor as tarefas, alocar as operações (resultado da decomposição das tarefas) para os robôs e monitorar a execução das tarefas alocadas. Certa autonomia dos agentes deve ser preservada com a finalidade de ser evitar uma arquitetura centralizada do sistema.

A proposta deste trabalho é a criação de uma arquitetura multi-agente para gerenciamento de sistemas multi-robôs. O sistema permite a seleção dinâmica e distribuída de recursos, com objetivo global. Pretende-se que este sistema seja baseado numa arquitetura distribuída multi-agente, onde as unidades possuem graus de autonomia.

Outro objetivo relevante deste trabalho foi abordar a tecnologia de agentes e de sistemas multi-agentes, mostrando o potencial dos arcabouços JADE e MaDKit para o desenvolvimento de sistemas multi-agente. JADE é um ambiente que propõe toda uma infraestrutura de suporte e desenvolvimento de sistemas multi-agentes, baseada em JAVA. A MaDKit é permite o rápido desenvolvimento de simulações de SMA.

O objetivo geral desta pesquisa é a criação de um modelo genérico para seleção, alocação e operação de recursos ou robôs distribuídos. Pretende-se que este modelo seja baseado numa arquitetura distribuída multi-agente e responda dinamicamente se reconfigurando em resposta a distúrbios ou desvios do planejamento, considerando os objetivos globais estabelecidos.

Os objetivos específicos estabelecidos foram os seguintes:

- Analisar os conceitos relacionados robótica, robótica móvel e colaborativa, inteligência e coordenação robótica, enxame de robôs, VANTs e aplicações;

- Analisar os conceitos relacionados sistemas multi-agentes e aplicações;
- Analisar comparativamente arquiteturas de sistemas multi-agentes;
- Especificar os requisitos para o modelo genérico;
- Desenvolver o modelo genérico que atenda aos requisitos estabelecidos, com a capacidade de se reconfigurar de forma dinâmica;
- Montar estratégias para avaliar o modelo proposto e validar o modelo.

1.3 Estrutura do Trabalho

O presente trabalho está estruturado da seguinte forma:

- O capítulo 1 tem-se a introdução desta tese, com as motivações e os objetivos da pesquisa;
- O capítulo 2 apresenta pesquisas relacionadas a sistemas multi-agentes, arquiteturas utilizadas e aplicações;
- O capítulo 3 apresenta conceitos e pesquisas relacionadas a sistemas multi-robóticos;
- O capítulo 4 apresenta a especificação e descrição da arquitetura do sistema multi-agente proposta para coordenação robótica;
- O capítulo 5 apresenta os detalhes construtivos da arquitetura do sistema multi-gente proposta para coordenação robótica;
- O capítulo 6 apresenta as simulações para validação da arquitetura proposta nas plataformas JASON, MaDKit e JADE;
- No capítulo 7 são abordadas prospecções de novas pesquisas e propostas de trabalhos futuros.
- O capítulo 8 apresenta as conclusões da pesquisa.

Capitulo 2 – Sistemas Multi-Agentes

Este capítulo apresenta conceitos e pesquisas relacionadas a sistemas multi-agentes.

2.1 Definições de Sistemas Multi-agentes

Não há um conceito único de agentes de software, várias abordagens são apresentadas por diversos autores. Genericamente, um agente (humano) pode ser uma pessoa que age no lugar da outra ou em seu nome, realizando tarefas ou ações determinadas. No caso de agentes artificiais, não humanos, ou de software há cento consenso de que são entidades com capacidade de se adaptarem ao ambiente, reagindo a sua dinâmica e modificando-o.

Conforme a FIPA ou *Foundation for Intelligent Physical Agents* (2002), um agente é processo computacional que realiza uma aplicação autônoma e com capacidade de comunicação, tipicamente usando uma ACL (*Agent Communication Language*).

A área de inteligência artificial tem atraído a atenção de pesquisadores fazendo com que sejam desenvolvidas máquinas com diferentes níveis de inteligência. A robótica é uma das áreas de inteligência artificial. A robótica tem sido muito empregada em indústrias com objetivo de aumentar a capacidade, a qualidade, a flexibilidade e a adaptabilidade produtiva. A robótica também é utilizada para realização de trabalhos em ambientes perigosos, como em plataformas de petróleo ou no espaço. Um grande desafio da área de robótica é embutir nos dispositivos inteligência, a fim de dar capacidade aos robôs de tomar decisões satisfatórias, considerando os objetivos estabelecidos, frente a várias situações do ambiente.

A robótica colaborativa é uma área também em expansão dentro da robótica. Neste contexto, a tecnologia de agentes pode ser empregada considerando as características comuns, tais como, a autonomia e a cooperação. A autonomia permite que os agentes ou robôs possam agir de acordo com seus próprios princípios independentes da interferência humana. Os agentes podem assim representar seus usuários agindo pró-ativamente. A cooperação é a capacidade dos agentes ou dos robôs de trabalharem em conjunto para realização de tarefas de interesse comum.

Para Daneshfar *et al.* (2009), as definições de agentes possuem alguns conceitos em comum, por exemplo, os agentes estão em ambientes e possuem autonomia. Um agente pode ser definido como uma entidade de *software* ou *hardware* situado em um ambiente e com capacidade de autonomamente reagir a modificações neste ambiente. O ambiente pode ser físico ou computacional e o agente por alterar o ambiente através de ações.

A autonomia dos agentes, para Wooldridge e Jennings (1995), é a capacidade de controle sobre suas ações, ou seja, a capacidade de iniciar e agendar a execução de tarefas. Os autores identificaram três classes de agentes:

- Agentes que executam tarefas simples baseadas em regras e premissas pré-definidas;
- Agentes que executam tarefas bem definidas quando solicitadas pelo usuário;
- Agentes que de forma proativa oferecem informações ou serviços ao usuário sem que tenha recebido solicitações.

Um sistema multi-agente, do inglês *multi-agent system* (MAS) é formado por uma rede de agentes computacionais que interagem uns com os outros. Os agentes podem ter apenas uma visão limitada do ambiente, assim como limitações quanto aos meios de interação com o ambiente e quanto aos meios para atingir seus objetivos. Os sistemas multi-agentes podem, por vezes, solucionar problemas que vão além dos limites da competência de um agente individual.

Conforme Lesser (1999), sistemas multi-agentes são sistemas computacionais nos quais dois ou mais agentes interagem ou trabalham em conjunto para executar um conjunto de tarefas ou para executar um conjunto de objetivos. Esses sistemas podem ser compostos por agentes homogêneos ou heterogêneos.

A autonomia de um agente está relacionada à sua habilidade de tomar suas próprias decisões a respeito de quais atividades deve realizar, quando realizá-las e como compartilhar e assimilar informações.

A autonomia do agente esta normalmente associada a sua capacidade de adaptabilidade, quanto maior a autonomia do agente maior também a adaptabilidade do agente em resolver problemas emergentes do contexto em que está situado. Os graus de autonomia e adaptabilidade do agente são comumente associados ao nível de inteligência e sofisticação que o agente possui.

2.2 Caracterização dos Sistemas Multi-Agentes e dos Agentes

Os MAS podem ser desenvolvidos com agentes competitivos, nestes:

- Os agentes podem ser projetados por distintos projetistas;
- Os agentes podem possuir motivação própria;
- Os agentes não se interessam obrigatoriamente pelo bem da comunidade, ao invés disso, possuem interesses próprios;
- Os acordos podem ocorrer através de negociação e da resolução de conflitos.

Os MAS podem ser constituídos por elementos cooperativos:

- Os agentes podem ser interdependentes;
- Os agentes são motivados pela utilidade global do sistema;
- Coordenação se dá através da cooperação e da formação de equipes.

Para integração de um conjunto de agentes constituintes de um MAS, há três conceitos importantes: comunicação, cooperação e coordenação.

A comunicação é o processo pelo qual as entidades do sistema podem interagir uns com os outros de maneira ordenada, sendo assim necessário o desenvolvimento de linguagens comuns a fim de que seja possível a cooperação entre as partes.

O objetivo da coordenação e da cooperação, neste contexto, é permitir a integração harmoniosa dos esforços individuais de entidades do sistema beneficiando um objetivo comum.

Em comunidade, o agente deve coordenar suas ações considerando os outros agentes. Modelos de coordenação necessitam de um meio e regras para o gerenciamento de interações e dependências entre os agentes. O fluxo de informação ou comunicação entre os agentes e o ambiente deve ser regulado. A comunicação pode ser indireta, via o ambiente, ou direta, através da troca de informação entre os agentes. A comunicação, para ser efetuada, necessita que haja uma linguagem definida e entendida pelas entidades do sistema.

A colaboração entre os agentes ocorre quando os mesmos se comprometem em realizar atividades que estão relacionadas com um objetivo comum. Ou seja, o objetivo comum pode ser alcançado a partir do comprometimento de todos os agentes envolvidos e necessários, num comportamento cooperativo. Há vários padrões de organizações de

sistemas multi-agentes, tais como times, coalizão, mercado, arquiteturas hierárquicas e heterárquicas.

A coordenação de um MAS é o processo em que um agente raciocina a cerca de suas ações locais e sobre as ações dos outros a fim de garantir que a comunidade atue de forma coerente.

Para Shen *et. al* (2006), a tecnologia de agentes desponta como novo paradigma para próxima geração de sistemas de manufatura. Pesquisas tentam aplicar a tecnologia em várias áreas da engenharia, com objetivo de prover a integração e a colaboração no chão de fábrica.

Segundo Monostori *et. al* (2006), agentes são importantes para engenharia industrial, pois eles possuem propriedades como autonomia, redundância, adaptabilidade e podem interagir cooperativamente num ambiente distribuído.

No estudo de Monostori *et. al* (2006) duas abstrações são apresentadas no que se refere a definição de agentes:

- Um agente é um sistema computacional situado em um ambiente dinâmico, tendo comportamento inteligente e autonomia;
- Sistemas multi-agentes são ambientes constituídos por agentes que interagem entre si, formando uma comunidade.

As propriedades básicas dos agentes, apresentadas por Monostori *et. al* (2006), são as seguintes:

- Os agentes observam o ambiente em que estão inseridos;
- Os agentes têm o seu próprio conhecimento e possuem suas crenças a respeito do ambiente;
- Os agentes possuem preferências a respeito do estado do ambiente;
- Os agentes podem iniciar e executar ações para mudar o estado dos ambientes.

Quando se trata de agentes, algumas características são compartilhadas por vários autores, tais como:

- Autonomia considerando suas próprias ações;
- Algum tipo de inteligência;
- Interação com o ambiente e com a comunidade de agentes;
- Adaptabilidade considerando o ambiente;

- Colaboração e cooperação considerando os objetivos do sistema ou objetivos compartilhados;
- E em alguns casos, mobilidade entre ambientes.

Os agentes podem ser visto como solucionadores de problemas com capacidade de interação, de sentir e de atuar no seu ambiente. Dependendo dos requisitos da aplicação ou do problema, diferentes modelos de agentes ou técnicas podem ser empregadas.

Para Wooldridge (2001), o principal desafio dos agentes é decidir que ações tomar para melhor satisfazer os objetivos estabelecidos. A complexidade destes processos de decisão depende das diferentes propriedades do ambiente em que os agentes estão inseridos.

Os ambientes podem ser, por exemplo, acessíveis ou inacessíveis, estáticos ou dinâmicos, determinísticos ou aleatórios, contínuos ou discretos e episódicos ou não episódicos. A acessibilidade é a capacidade que os sensores dos agentes possuem de perceber o estado do ambiente. Na dinâmica do ambiente verifica-se se o ambiente sofre ou não alterações enquanto o agente está processando a ação a realizar. No comportamento determinístico ou aleatório do ambiente observa-se o fato do próximo estado ser ou não determinado pelo estado atual do ambiente e pelas ações selecionadas pelo agente. Um ambiente pode ser classificado como discreto ou como contínuo avaliando a existência de um número distinto de percepções e ações ou quando as percepções e ações mudam em um espectro contínuo de valores. Avalia-se se os ambientes são episódicos ou não observando se a existência do agente é divida em etapas independentes das etapas anteriores. Os agentes podem também ser monoagentes e multi-agentes dependendo se um ou vários agentes estão presentes atuando no sistema (Artero, 2009).

Um agente inteligente estende a definição geral de um agente a partir da extensão do conceito de autonomia para uma autonomia flexível (Wooldridge, 2001). Um agente com autonomia flexível, ou agente inteligente, possui as seguintes características:

- Reatividade: Uma agente inteligente reage às mudanças ambientais.
- Pró-atividade: Um agente inteligente possui comportamento ativo ou proativo determinado pelos seus objetivos. Em um comportamento

guiado a objetivos, o agente irá dinamicamente modificar seu comportamento para atingir seus objetivos.

- Habilidade social: Agentes inteligentes possuem capacidade de interagir com outros agentes. Essa capacidade deve ir além de simplesmente passar dados entre diferentes entidades (*hardware* ou *software*), sendo a habilidade de negociar e interagir de uma maneira cooperativa.

2.3 Arquitetura dos Agentes

Wooldridge (2001) classifica a arquitetura dos agentes em quatro classes, as quais são:

- Agentes baseados em lógica: suas decisões são guiadas por deduções lógicas;
- Agentes reativos: há um mapeamento direto entre situação do ambiente e ações reativas;
- Agentes do tipo crença-desejo-intenção ou *belief-desire-intention* (BDI): as decisões dependem da manipulação de estruturas de dados representando as crenças, os desejos e as intenções dos agentes;
- Arquiteturas de camadas: as decisões são tomadas através da interação de várias camadas de software.

2.3.1 Arquitetura Baseada em Lógica

As arquiteturas baseadas em lógica usam abordagens tradicionais na construção de sistemas artificiais inteligentes, também conhecido do inglês, como A*rtificial Intelligence* (AI). Nesta abordagem o comportamento inteligente do agente pode ser gerado através de uma representação simbólica do ambiente e de seu comportamento desejado (formulação lógica), além da manipulação sintática desta representação (dedução lógica). O comportamento dos agentes nesta abordagem depende da dedução das regras estabelecidas, da informação presente na base de dados e da representação da informação que o agente tem sobre seu ambiente (Wooldridge, 2001).

Conforme Wooldridge (2001), problemas aparentemente intratáveis segundo a abordagem lógica (simbólica) levaram alguns pesquisadores questionarem tal

abordagem. Segundo os mesmos, pequenas variações na abordagem simbólica não seriam suficientes para construir agentes em ambientes com restrições de tempo. Assim, nos anos oitenta pesquisadores começaram a investigar alternativas ao paradigma da inteligência artificial com abordagem simbólica.

2.3.2 Arquitetura Reativa

A arquitetura reativa, também conhecida como *subsumption architecture* (do inglês), define duas características importantes. A primeira é que a tomada de decisão dos agentes é efetuada através de um conjunto de comportamentos voltados para realização de tarefas. Cada comportamento pode visto como uma função de ação individual, que continuamente coleta a percepção de entrada e a relaciona para a execução de uma ação. Os conjuntos de comportamentos ou ações visam à execução de tarefas. A segunda característica da arquitetura reativa é que os comportamentos podem ser iniciados simultaneamente. Deve haver então um mecanismo de seleção de ações a partir de múltiplas ações. Uma arquitetura com os comportamentos organizados em camadas pode ser empregada. As camadas inferiores na hierarquia são capazes de inibir as camadas superiores, ou seja, possuem prioridade superior. As camadas superiores podem ser representações mais abstratas de comportamento. Como exemplo, um comportamento desejado em um robô móvel é o de evitar obstáculos, assim é indicado que este comportamento tenha uma prioridade alta, consequentemente, esse comportamento deve ser inserido em camadas inferiores da arquitetura.

A abordagem reativa não apresenta a complexidade teórica da abordagem lógica e é esperado desempenho aceitável em aplicações com grande número de comportamentos. Como características da abordagem reativa pode-se citar simplicidade, economia, facilidade no trato computacional e robustez contra falhas (Wooldridge, 2001).

2.3.3 Arquitetura BDI (Belifes Desires Intentions)

Um dos modelos que permite aos agentes terem conhecimento de si mesmos e do ambiente que os cerca é o modelo nomeado BDI, ou do inglês *Belifes Desires Intentions Model*. Conforme Monostori *et. al* (2006), neste modelo o agente possui

crenças do estado do ambiente em que está inserido (*beliefs*), os estados futuros são metas ou objetivos e em longo prazo são desejos (*desires*), os compromissos que o agente faz previamente são as intenções (*intentions*). Assim, pode-se dizer que o modelo BDI é um modelo para um agente cognitivo, ou seja, um agente que modela sua decisão a respeito de um problema considerando conceitos cognitivos.

Arquiteturas do tipo crença-desejo-intenção, ou no inglês *belief-desire-intention* (BDI) tem suas raízes na filosofia tradicional de entender o raciocínio prático, ou seja, o processo de decisão de qual ação executar a cada instante para atingir os objetivos definidos.

O raciocínio prático da arquitetura BDI envolve dois importantes processos: Decidir quais objetivos se quer atingir e como esses objetivos serão alcançados. Pode-se chamar deliberação a composição destes dois processos. Nesta abordagem podem-se destacar algumas características:

- As intenções conduzem o raciocínio de deliberação dos agentes;
- As intenções restringem as deliberações futuras;
- As intenções persistem;
- As intenções dependem das crenças estabelecidas, para um comportamento racional.

Um problema no desenvolvimento da arquitetura BDI é alcançar um equilíbrio de quando e com que frequência um agente deve abandonar suas intenções estabelecidas considerando que o mesmo já não acredita que pode alcançar os objetivos. De tempo em tempo o agente deve parar para reavaliar suas intenções e crenças. No entanto, essa parada implica em custos adicionais, de tempo e de recursos computacionais. Esse dilema é essencialmente o problema do equilíbrio entre o comportamento proativo (guiado aos objetivos) e o reativo (guiado pelos eventos).

Experimentos mostraram que diferentes tipos de ambiente requerem diferentes tipos de decisões estratégicas.

Em ambientes estáticos e constantes, comportamentos proativos e guiados a objetivos são adequados, neste caso o agente pode ter um comportamento eficiente e robusto sem parar para reavaliar suas intenções.

Em ambientes mais dinâmicos, a habilidade de reação às mudanças para modificar e reavaliar as intenções são mais importantes, neste caso o agente pode ter um comportamento cauteloso (Wooldridge, 2001).

O processo do raciocínio prático dos agentes BDI é apresentado na **Erro! Fonte de referência não encontrada.**-1. Como ilustrado, um agente BDI é composto por sete módulos principais:

1. Função de revisão de crenças – função de revisar as crenças do agente a partir da percepção da entrada e de suas crenças correntes;
2. Conjunto de crenças (*beliefs*) – representando as informações que os agentes possuem a respeito do ambiente;
3. Função de geração de opções – determina as opções disponíveis para o agente ou seus desejos, com base nas suas crenças correntes sobre o ambiente e nas suas intenções correntes;
4. Conjunto de desejos ou anseios (*desires*) – representam o conjunto de desejos ou anseios do agente, também as possibilidades de ações disponibilizadas ao agente;
5. Função filtro – representa o processo de deliberação do agente, que determina as intenções do agente com base nas suas crenças, desejos e intenções correntes.
6. Conjunto de intenções (*intentions*) – representa os objetivos do agente ou o que o agente se comprometeu a fazer;
7. Função de seleção de ações – determina uma ação a ser executada com base nas intenções correntes.

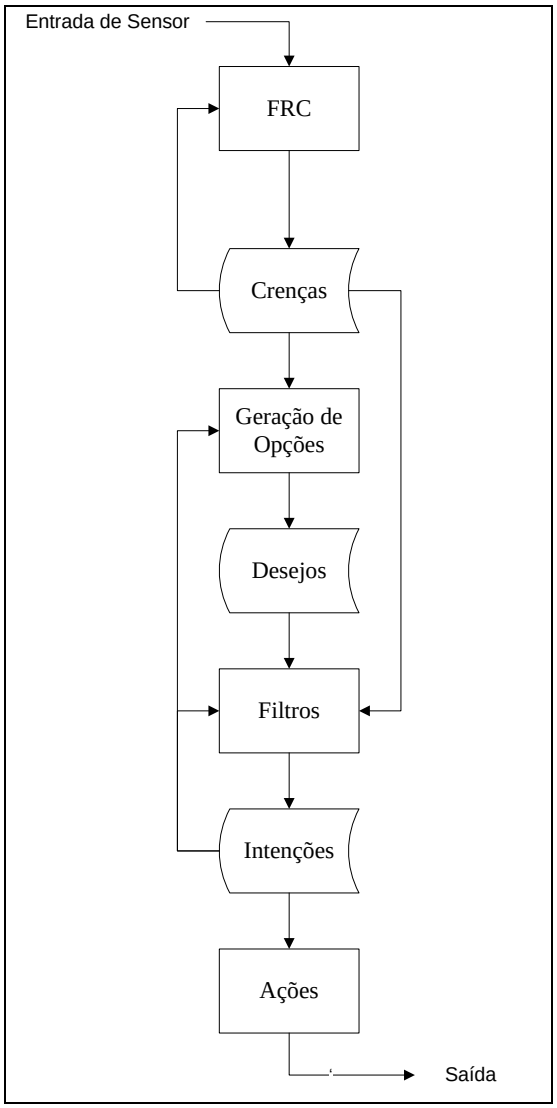

Figura 2-1 Diagrama esquemático da arquitetura genérica BDI

Para Wooldrige, na arquitetura BDI deve-se ajustar bem o processo de deliberação de acordo com seu ambiente, assegurando que em ambientes dinâmicos e em domínios altamente imprevisíveis, reconsiderações e reavaliações das intenções dos agentes devem ser relativamente mais frequentes. Enquanto, para ambientes mais estáticos, reconsiderações e reavaliações menos frequentes são satisfatórias.

O modelo BDI é atrativo, pois é intuitivo, visto que o processo de decisão do que fazer e de como fazer pode ser reconhecido, além do comum entendimento dos conceitos de crenças, anseios e intenções (*belief, desire, intention*).

2.3.4 Arquitetura em Camadas

As arquiteturas em camadas requerem que o agente seja capaz de ter comportamento reativo e proativo, e sua decomposição envolve a criação de subsistemas separados para lidar com esses comportamentos distintos. Essa abordagem leva a criação de uma arquitetura baseada em vários subsistemas organizados em camadas hierárquicas.

Essa abordagem de uma arquitetura em camada pode ser constituída ao menos de duas camadas, uma para prover os serviços relacionados ao comportamento reativo e outra relacionada ao comportamento proativo. Cada camada pode possuir dois tipos gerais de funções. A primeira é a função de reconhecimento da situação e ativação do objetivo, similar a função de "opções" do modelo BDI. Uma segunda função é responsável pelo planejamento e agendamento, ou seja, seleção de planos para execução, baseados nos planos correntes, nos objetivos e na base de conhecimento da camada.

Conforme Wooldrige as arquiteturas em camada são as mais populares arquiteturas de agentes disponíveis. As camadas representam as funcionalidades. Os comportamentos reativos, proativo e social são estruturados em camadas.

2.4 *Comunicação entre Agentes.*

A interação é uma característica importante e essencial numa comunidade de agentes. A comunicação entre agentes pode ser direta ou indireta via ambiente. A cooperação e o mecanismo de coordenação são motivadores para haver tal interação. O CNP (*Contract Net Protocol*) pode ser usado como protocolo que permite o mecanismo de coordenação ou de contratação de atividades ou serviços.

Em alguns casos, os agentes podem possuir interesses conflitantes, neste caso deve-se fazer uso de mecanismos de negociação e ou protocolos específicos.

Uma linguagem de comunicação entre agentes é necessária para colaboração, negociação e coordenação entre agentes. Atualmente, há diversas linguagens para realizar a comunicação entre agentes, tais como: AgentTalk, ACL (*Agent Communication Languages*), KIF (*Knowledge Interchange Format*) e KQML (*Knowledge Query and Manipulation Language*).

A ACL é usada pela FIPA (*Foundation for Intelligent Physical Agent*), e é uma alternativa destacável a ser empregada em sistemas multi-agentes.

No que se refere ao mecanismo de coordenação e diferentes estruturas organizacionais de sistemas multi-agentes, na maioria dos casos, o comportamento e o relacionamento humano nas organizações é tomado como referência. Em Monostori *et. al* (2006) diferentes modelos são apresentados, uns em que o mecanismo da coordenação é mais centralizado e outros em que se busca uma descentralização das responsabilidades.

O modelo comportamental humano em ambientes organizacionais pode ser tomado como base, conforme Monostori *et. al* (2006), e a tecnologia de agentes de software tem potencial para criação de modelos mais avançados. Modelos descentralizados para solução de problemas, num ambiente colaborativo, é um tema abrangente e bastante estudado por várias disciplinas e áreas do conhecimento humano.

Uma das arquiteturas mais simples de sistemas multi-agentes é a quadro negro (do inglês *backboard*), onde não há direta comunicação entre os agentes. A comunicação é realizada via a estrutura de dados central (quadro negro) compartilhada por todos os agentes. Outra abordagem oposta propõe a troca de mensagem entre os agentes exigindo a adoção de um protocolo mais elaborado de comunicação (Artero, 2009). Neste tipo de arquitetura os agentes necessitem de nomes e de endereços para permitir a troca de mensagens entre eles.

2.5 Aplicações de Sistemas Multiagentes

A tecnologia baseada em agentes de software tem se mostrado como uma alternativa promissora para implantação em sistemas distribuídos de produção.

Para Shen *et. al* (2006), as abordagem tradicionais para o problema de programação da produção, sejam analíticos, heurísticos ou meta-heurísticos (redes neurais, algoritmo genéticos, *Tabu Search*) encontram dificuldades quando aplicados em situações reais, pois usam modelos teóricos simplificados e são essencialmente computacionalmente centralizados.

Conforme Shen *et. al* (2006), na década passada vários pesquisadores têm aplicado a tecnologia de agentes na tentativa de resolver problemas de programação em aplicações como programação da produção, programação de transporte, programação de distribuição de energia, programação de recursos computacionais, agendamento de reuniões, agendamento de exame médico e gerenciamento de projetos. Pode-se verificar

que a abordagem baseada em agentes vislumbra algumas vantagens para aplicação em manufatura:

- O paradigma de agente usa computação paralela através de um largo número de processadores, os quais podem prover sistemas de programação com alta eficiência e robustez;
- É possível a interação do planejamento do processo com a programação da produção, de modo a tornar possível uma melhoria na eficiência de forma simultânea;
- Agentes recursos podem ser conectados diretamente com os dispositivos físicos representados, permitindo assim realizar a reprogramação dinâmica em tempo real, fazendo o sistema altamente tolerante a falhas;
- Pode-se criar agendas para os recursos utilizando o mecanismo usado em tratamento de negócios, ou seja, negociação em vez de simples pesquisa por recursos disponíveis;
- É possível um recurso individual abdicar de resultados locais (desempenho local) visando melhoria de resultados globais, demonstrando um comportamento cooperativo;
- Outras técnicas podem ser combinadas com a abordagem multi-agente.

Diferentes metodologias e técnicas foram empregadas no desenvolvimento de sistemas multi-agentes aplicados programação da produção.

No estudo de Maione e Naso (2003) é apresentada uma arquitetura multi-agente para criar dinamicamente rotas de itens (partes) num sistema de manufatura flexível. Tal arquitetura combina diferentes tecnologias das áreas da computação e da inteligência artificial. Conforme experimentos realizados pelos autores, a aplicação de lógica *fuzzy* ou de algoritmos *fuzzy* em arquitetura de decisão distribuída apresentou resultados interessantes.

Segundo os autores, o algoritmo nomeado *fuzzy multi-criteria algoritm*, destaca-se como uma das alternativas eficientes para o projeto de agentes inteligentes capazes de realizar avaliações quantitativas de parâmetros decisórios variados, em tempo real. No sistema proposto, cada parte ou item no sistema é associado a um agente de software inteligente com a capacidade de selecionar seu próximo destino de forma autônoma e em tempo real. O método supera os métodos ou algoritmos de negociação usuais visto

que permite que os agentes representantes das partes ou *partagents* tomem decisões não apenas sobre a operação iminente, mas também a operação subsequente. As decisões antecipadas dos *partagents* são transmitidas para outros agentes, nomeados *workstation agents*, os quais tem a função de detectar e resolver conflitos, em alguns destes casos modificando decisões do *partagents*. Neste estudo cada agente foi modelado usando a especificação chamada DEVS, do inglês *Discrete EVent System*. A partir deste formalismo os agentes podem ser descritos como sistemas sujeitos a entradas de outros agentes e com capacidade de gerar saídas para os demais agentes. Desta forma todo o sistema distribuído de manufatura multi-agente pode ser modelado, incluindo relações entre as entidades. O estudo contribuiu numa área importante que é o desenvolvimento e a aplicação de ferramentas de modelagem de sistemas multi-agentes.

A busca de padrões no uso de sistema multi-agentes, no que se refere a aplicações, modelagem e ferramentas, tem sido intensa como é o caso dos seguintes projetos: MaSE, Agent UML (AUML 2003) e do Java Agent DEvelopment (JADE 2003).

Maione e Naso (2003) compararam, através de simulações, o desempenho de Agentes de Decisões Multicritério (MCD), usando algoritmo multicritério, tais como:

- MWQ (*Minimum Work in Queue*) – A rota ou a máquina que irá receber o item é escolhida considerando a alternativa com menor quantidade de trabalho na fila;
- SPT (*Shortest Processing Time*) – Escolhe-se a máquina mais rápida a executar a operação;
- SD (*Shortest Distance*) – A escolha é realizada optando pela máquina mais próxima;
- ARDD (*Alternative Routings Dynamically Directed*) – A escolha é baseada na menor soma dos tempos de transporte, de espera e de processo.

As simulações realizadas por Maione e Naso (2003) confirmaram as vantagens do mecanismo de coordenação proposto. O trabalho também contribui para resolução de problemas clássicos da arquitetura hierárquica de manufatura, ambiente em que há completa distribuição das atividades de controle e de decisão, mantendo o desempenho do sistema.

A intervenção temporária, proposta por Maione e Naso (2003), pelos agentes de alto nível proporcionou melhoras no desempenho do sistema, uma vez que controlou o aumento no uso da banda de comunicação e do consumo de recursos computacionais.

Esse aumento pode ocorrer devido possíveis conflitos relacionados ao mecanismo de cooperação e ao protocolo de negociação empregado.

O artigo de Sluga *et. al* (2001) trata em essência da questão reconfiguração ou auto-organização dos sistemas de manufatura. O trabalho teve como objetivo principal a superação de modelos hierárquicos e convencionais por arquiteturas capazes de prover características adaptativas e dinâmicas. A proposta apresentada foi baseada no conceito de sistema de manufatura adaptativo complexo, o qual é caracterizado pela decomposição ou distribuição dos objetivos do sistema produtivo e pela alocação das atividades ou operações para as unidades de produção autônomas. O mecanismo de alocação de atividades é baseado no comportamento do mercado, provocando uma auto-organização e melhora de desempenho na manufatura a partir da seleção de recursos no sistema. O modelo do sistema distribuído de manufatura com agentes de software foi validado com simulações.

Sluga *et. al* (2001) citam os novos requisitos dos futuros sistemas de manufatura, dentre os quais se destacam:

- Arquitetura aberta;
- Capacidade de comunicação avançada;
- Estrutura de decisões descentralizada;
- Habilidade de auto-organização;
- Sistema de redefinição de trabalho ou atividades em termos de autonomia, de adaptabilidade evolucionária, de cooperação e de coordenação, de habilidade de comunicação, de poder de reconfiguração, de interatividade, de atividades e tarefas orientadas a competências, além da capacidade de aprendizado.

Na estrutura proposta por Sluga *et. al* (2001) há os chamados EWS, do inglês *Elementary Work Systems*, os quais possuem comportamentos autônomos, competências e capacidades de executar operações de manufatura. Um EWS é uma entidade física ou *hardware* necessário para executar atividades no processo produtivo. Outra entidade presente no modelo proposto é o VWS, do inglês *Virtual Work System*. O VWS foi adicionado na arquitetura proposta com a finalidade de introduzir ordens neste sistema distribuído para os EWS. O VWS é um agente de software que representa os EWS no ambiente virtual. A entidade de software VWS possui quatro elementos funcionais básicos, referenciados como: *perceptor*, *evaluator*, *effector* e o mecanismo de inferência. O *perceptor* observa o estado do ambiente de forma a perceber informações

relevantes, gerando entradas de tais percepções e inicia o processo de avaliação, cuja responsabilidade é do *evaluator* (avaliador). O mecanismo de inferência controla a avaliação considerando dados localmente armazenados e o conjunto de objetivos. A saída é provida pelo *effector*, afetando o ambiente.

Em sistemas de manufatura tradicionais, comentam Sluga *et. al* (2001), a alocação de atividades para os recursos é um problema de planejamento da produção, ou seja, define-se o plano de atividades por recursos no tempo. Em sistemas distribuídos, a questão não se limita a este problema clássico de PCP, mas surge o desafio da estruturação (ou reconfiguração) adequada e dinâmica do sistema. No modelo proposto por Sluga *et. al* (2001), este processo de reconfiguração dinâmica do sistema é baseado no mecanismo de mercado. Há a figura de agentes mediadores que representam demandas e dos VWS que representam os recursos EWS. Cada agente busca melhorar o desempenho considerando suas restrições correntes. Estruturas dinâmicas e temporárias de manufatura são construídas para atender a objetivos específicos de produção. Assim, para fabricação de uma peça uma estrutura temporária é construída, em seguida a estrutura desaparece. A arquitetura proposta por Sluga *et. al* (2001) combina propriedades de sistemas hierárquicos e de sistemas heterárquicos, assim trata-se de um modelo híbrido de sistema distribuído.

Uma forma de lidar com a necessidade de previsibilidade e de melhoria de desempenho em sistemas de controle distribuídos é buscar um equilíbrio entre estruturas hierárquicas e heterárquicas a fim de combinar as vantagens de ambas (Naso *et. al*, 2003).

Nos sistemas de manufatura holônicos de Van Brussel *et. al* (1998), um *holon* é ao mesmo tempo um agente e parte de uma outra entidade. Os hólons são organizados em uma estrutura heterárquica, mas podem temporariamente formar uma hierarquia local para realizar tarefas específicas.

No modelo holônico, que consiste de um conjunto de hólons, autônomos e cooperativos, há um controle distribuído, onde os membros podem formar federações temporais.

A PROSA (*Product, Resource, Order, Staff holons Architecture*), de Van Brussel *et. al* (1998), é uma arquitetura de controle distribuído composto dos seguintes tipos de hólons:

- O hólon produto retém o conhecimento do processo para manufatura de produtos;

- O hólon recurso controla as estações de trabalho e processa as informações associadas às mesmas;
- O hólon ordem é responsável por executar as tarefas designadas corretamente;
- Um quarto, hólon *staff* pode ser introduzido ao sistema para controlar a transição de grupos de hólons de uma estrutura hierárquica (em caso de situações estáveis do ambiente) para uma estrutura heterárquica e vice-versa (para casos em que os distúrbios são severos).

A arquitetura MetaMorph proposta por Maturana e Norrie (1999) introduziu os mediadores, ou agentes especiais que assumiram o papel de coordenadores para promoverem a cooperação entre os agentes inteligentes e aprendendo com o comportamento dos agentes. Na fábrica, quando um agente faz uma requisição, o agente mediador seleciona os receptores mais adequados e envia a solicitação a eles. A seguir o mediador deixa que ocorra a interação autônoma entre os agentes solicitantes e os agentes servidores. Os mediadores reusam as informações das ações passadas através da técnica chamada, no inglês, de *distributed case-based learning* e simulação para inspecionar o desempenho antecipadamente.

Heragu *et. al* (2002) propôs um novo modelo utilizando uma arquitetura híbrida composta por agentes do tipo parte, agentes do tipo máquina e agentes do tipo manipulação de materiais, representando os controladores de baixo nível. No alto nível, agentes do tipo célula ou departamento coordenam as ações dos agentes do tipo máquina e dos agentes manipuladores de materiais (de cada departamento). No topo da organização, um agente controla todos os agentes do tipo célula no estoque. Decisões são tomadas em cada um dos três níveis seguindo regras de negociação. Os agentes de alto nível assumem papeis de supervisão e podem estabelecer decisões, objetivos e restrições aos níveis inferiores. O planejamento de tarefas seguem algoritmos de negociação similares aos usados por Van Brussel *et. al* (1998) e por Maturana *et. al* (1999).

Capítulo 3 – Sistemas Multi-Robóticos

Neste capítulo serão abordados conceitos e pesquisas relacionadas a sistemas multi-robóticos.

3.1 Definições de Sistemas Multi-Robóticos

A área de sistemas robóticos multi-agentes, engloba vários tópicos avançados, tais como: robótica evolutiva; inteligência de enxames (*Swarm Intelligence*); algoritmos de "otimização" baseados em colônias de formigas (*Ant Colony Systems*); e estratégias de coordenação multi-robóticas.

Para Kaminka (2004), robôs são agentes, ou podem ser tratados como agentes, não só pelas definições, mas também pelas aplicações. Dois argumentos são apresentados, reforçam e suportam essa afirmação. Primeiro, em determinado nível de abstração os robôs possuem definições e desafios de desenvolvimento comuns aos agentes autônomos. O segundo argumento está relacionado ao impacto de sucesso na robótica das pesquisas da área de times de trabalho (*Teamwork*) de sistemas multi-agentes. Segundo o Kaminka, a comunidade AAMAS (*Autonomous Agents and Multi-Agent Systems*) teve um significante impacto no mundo da robótica e vice-versa.

Nos últimos anos houve considerável crescimento no número de pesquisas na área de sistemas multi-robóticos, que representam uma extensão de sistemas multi-agentes, direcionadas para aplicações capazes de realizar tarefas perigosas e fadigantes para seres humanos. Ao invés de um único robô com capacidade de realizar todas as tarefas, tem-se investido em aplicações multi-robóticas distribuídas. Há várias alternativas de projetos de sistemas multi-robóticos, como por exemplo: o emprego de robôs cooperativos ou competitivos; o uso de um modelo homogêneo ou heterogêneo; além da possibilidade de ação coordenada ou não para realização de uma tarefa.

Há várias aplicações para robôs móveis, tais como limpeza de ambientes, corte de gramado, exploração de minas, colheita na agricultura, pintura e outras mais. Nestas aplicações os robôs trabalham em áreas delimitadas e possuem capacidade de evitar obstáculos. Os robôs assumem tarefas para varrer toda a área delimitada.

O crescente interesse nesta área em que múltiplos robôs cooperam para explorar uma área é motivado pela eficiência com que esse grupo de robôs pode completar uma missão dividindo-a em tarefas e distribuindo-as aos robôs. Outro ponto importante é que tais sistemas respondem melhor a distúrbios e falhas do que um sistema composto por apenas uma entidade robótica. Se uma falha ocorre em uma aplicação com apenas um robô, certamente a tarefa de exploração ficará comprometida. Em uma solução distribuída, algoritmos multi-robóticos podem ser empregados para redistribuição de tarefas, a fim de que a missão ou a exploração não seja comprometida. No entanto, mesmo em tais aplicações ainda há exigências como a eficiência e a robustez na realização das missões estabelecidas. Em muitos casos buscando minimizar o tempo e os recursos empregados para realização da missão e eficientemente lidar com situações catastróficas com os robôs.

Os sistemas multi-agentes robóticos são compostos por robôs que possuem suas ações coordenadas e podem cooperar entre si a fim de atingir um objetivo global.

Segundo Lesser (1999), agentes pode ser caracterizados pela sua benevolência (ou comportamento cooperativo) ou seu auto-interesse. Enquanto os agentes cooperativos trabalham direcionados a objetivos comuns, os agentes de auto-interesse possuem objetivos distintos e podem interagir para atingir seus objetivos. Estes agentes, direcionados aos seus próprios interesses, apresentam assim um comportamento competitivo. Tais agentes podem interagir com outros em troca de favores e algum valor (moeda), em uma ação coordenada executando atividades para atingir seus objetivos.

Pesquisas científicas e práticas na área de sistemas multi-agentes, que no passado foram chamadas de Inteligência Artificial Distribuída ou DAI (*Distributed Artificial Intelligence*), tem como objetivo o desenvolvimento de princípios computacionais e modelos para construção e análise de padrões de interação e coordenação de sociedades de agentes (Lesser, 1999). O contexto da pesquisa com sistemas multi-agentes acaba por abranger outras disciplinas como a robótica e sistemas multi-robóticos. Basta realizar uma abstração do conceito de agente para uma entidade que pode ser um robô, que possuí certo grau de autonomia e capacidade de interação para atingir os objetivos (globais ou locais), para entender as similaridades entre as áreas de sistemas multi-robóticos e sistemas multi-agentes.

Há um crescente número de pesquisas de sistemas compostos de múltiplos robôs autônomos com comportamento cooperativo. Tais sistemas possuem subáreas de

pesquisa como a arquitetura de agrupamento, gerenciamento de conflito de recursos, aprendizagem e problemas de geometria.

Um dos grandes interesses da robótica móvel é a navegação de agentes robóticos que trabalham cooperativa na solução de problemas, tais como, de deslocamento de cargas, de entrega de materiais em ambientes produtivos, de construção de mapas em ambientes desconhecidos, de entrega de medicamentos em hospitais e de outras diversas aplicações.

As aplicações práticas da robótica móvel mostram o avanço desta área vislumbrando excelentes perspectivas futuras. Há de se esperar um aumento e popularização de aplicações domésticas, tais como aspiradores de pó e contadores de grama. Na indústria já se pode ver aplicações como transporte automatizado e veículos autônomos. Como aplicações militares pode-se destacar os sistemas de monitoramento aéreo (VANTs, veículos aéreos não tripulados), os sistemas de transporte de suprimentos, os sistemas de combate e os sistemas de controle e patrulhamento terrestres.

3.2 Caracterização de Sistemas Multi-Robóticos

Os robôs móveis autônomos possuem capacidade de locomoção e de operação de modo semi-autônomo ou completamente autônomo. Os robôs podem possuir capacidade de percepção do ambiente, através de sensores, capacidade de agir ou de atuar no ambiente, por meio de elementos atuadores e motores, e capacidade de responder a situações diversas, resolvendo tarefas por vezes complexas. A capacidade de navegação e de operação autônoma dos robôs, através técnicas de planejamento e de controle robusto, determina suas inteligências.

Os sistemas de controle de robôs móveis devem ser robustos e tolerantes a falhas, sendo necessário cuidado com diversos requisitos, tais como:

- Projeto dos elementos sensores a fim de adquirir e processar as informações relevantes a respeito do ambiente;
- Detecção e desvio de obstáculos, evitando colisões e preservando a integridade dos elementos que constituem o ambiente e o próprio robô;
- Auto-localização e mapeamento do robô no ambiente para permitir o seu deslocamento conforme uma trajetória planejada;

- Planejamento de trajetórias (de um ponto de origem ao alvo) e navegação robótica;
- Interação e comunicação, ou seja, a capacidade de interação com outros elementos do ambiente.

Os sistemas multi-robóticos ou MRS (*Multi-Robotic Systems*) ou sistemas robóticos distribuídos seguem os conceitos da área da inteligência artificial distribuída, em que grupos de robôs inteligentes são capazes de autonomamente interagir com o ambiente e uns com os outros. A área também conhecida por DARS (*Distributed Autonomous Robot Systems*) engloba temas como: enxame de robôs (*Swarm robotics*); sistemas robóticos modulares reconfiguráveis (*Reconfigurable Modular Robotic Systems*); sistemas híbridos; e rede de robôs ou, do inglês, *Network Robot Systems* (NRS).

A área de enxame de robôs é inspirada no estudo do comportamento social dos insetos. Algumas características destes sistemas são a simplicidade e a homogeneidade física dos robôs, elevada tolerância à falha e paralelismo de tarefas com foco nas missões. Há várias aplicações para estes sistemas, tais como, rede de sensores móveis, pesquisa, resgate e vigilância.

Os sistemas robóticos modulares reconfiguráveis são formados por várias unidades celulares homogêneas. As células possuem capacidade de se anexar umas as outras e reconfiguração das suas estruturas (formadas pela combinação das células) quando desejado.

Os robôs podem ser considerados como agentes, e assim pode-se usar os conceitos e metodologias usados com sucesso em sistemas multi-agentes para resolver problemas de organização, coordenação e colaboração nos DARS.

3.3 Aplicações de Sistemas Multi-Robóticos

Sistemas multi-agentes robóticos ou sistemas multi-robóticos podem ser aplicados em situações em que há um conjunto de tarefas complexas ou impossíveis de resolver por apenas um robô ou quando há benefícios de desempenho na abordagem de multi-robótica. Uma configuração com vários robôs mais simples pode proporcionar resultados mais satisfatórios do que o emprego de um único robô mais robusto,

considerando aspectos como, simplicidade de construção, flexibilidade, tolerância a falhas e custos (Cao *et. al*, 1997).

Os sistemas multi-agentes robóticos diferem dos sistemas distribuídos tradicionais em função do ambiente real envolvido no problema. Os componentes dos sistemas distribuídos tradicionais são computadores, banco de dados ou redes de computadores. Em sistemas multi-robóticos, o comportamento cooperativo da coletividade de robôs pode ser definido como uma forma de interação, baseada em comunicação, a fim de realizar tarefas colaborativamente direcionadas a objetivos e interesses comuns. Essas abordagens acabam por criar perspectivas de aplicações, tais como, a decomposição e a alocação de tarefas, ou até mesmo aplicações de inteligência artificial distribuída.

Numerosas pesquisas de robótica cooperativa começaram a surgir nos anos 80, como CEBOT, SWARM, ACTRESS e GOFER (Cao *et. al*, 1997).

CEBOT (*CEllular roBOTics System*) é uma arquitetura descentralizada e hierárquica inspirada pela organização celular de entidades biológicas (Cao *et. al*, 1997). O sistema pode ser reconfigurável dinamicamente em resposta as mudanças do ambiente.

Um SWARM é um sistema distribuído com um grande número de robôs autônomos. As pesquisas de sistemas SWARM começaram com a pesquisa de sistemas robóticos celulares de Beni (1988), onde agentes simples eram capazes de executar tarefas como a de formação de padrões e auto-organização. A *SWARM intelligence* é uma propriedade de sistemas compostos de robôs não inteligentes com comportamento coletivo inteligente. A auto-organização, conceito importante neste tipo de sistema, é a habilidade da distribuição eficiente de uma dada tarefa, como por exemplo, a formação de um padrão geométrico ou uma estrutura de organização. Assim, os SWARM apresentam arquitetura distribuída e normalmente não há diferenciação entre os membros, os robôs são do mesmo tipo. A interação se dá através da reação de cada robô ao estado de seu vizinho mais próximo (Cao *et. al*, 1997).

No sistema ACTRESS (*ACTor-based Robot and Equipments Synthetic System*) "*robotors*" (3 robôs e 3 estações de trabalho) formam um grupo heterogêneo para executar tarefas como empurrar objetos que não podem ser executadas por um *robotor* apenas (Asama *et. al*, 1989). Protocolos de comunicação foram estabelecidos com mecanismo de negociação baseado no *Contract Net* (Smith, 1980).

A arquitetura GOFER (Caloud *et. al*, 1990) foi usada para estudar a solução do problema de múltiplos robôs em um ambiente fechado usando técnicas de inteligência artificial. Um sistema central de planejamento de tarefas e programação ou agendamento (CTPS – *Central Task Planning and Scheduling System*) se comunica com todos os robôs e possui uma visão das tarefas a serem executadas e da disponibilidade dos robôs em executá-las. O CTPS gera um plano estruturado e informa todos os robôs disponíveis das metas ou objetivos pendentes e dos planos. Os robôs usam um algoritmo de alocação de tarefas semelhante ao protocolo *Contract Net* para determinar suas responsabilidades para atingir as metas estabelecidas. A arquitetura GOFER foi usada com sucesso em robôs físicos para executar três tipos de tarefas.

Heinen (2002) apresentou uma arquitetura híbrida de controle robótico, a arquitetura COHBRA (Controle Híbrido de Robôs Autônomos). Módulos de auto-localização, de planejamento (utilizando uma estratégia de controle deliberativo) e de desvio de obstáculos (com estratégia de controle reativo) foram desenvolvidos.

Uhrmacher e Weyns (2009) apresentam um caso de aplicação de um sistema de controle multi-agente no desenvolvimento de um sistema de transporte utilizando veículos guiados automaticamente (AGVs) em um ambiente de armazenamento ou estoque. Os AGVs transportavam materiais do armazém para o destino definido.

O sistema foi desenvolvido no projeto chamado de EMC_2 (*Egemin Modular Controls Concept*). Egemin N.V. é uma empresa belga que fabrica AGVs e desenvolve *softwares* de controle para aplicações de logística automatizada de armazém de materiais, usando AGVs. Esses AGVs possuem capacidade de se mover e de girar, além de pegar e de soltar cargas. Os AGVs possuem sensores para monitorar a posição e nível de bateria, plataforma computacional e módulo de comunicação sem fio. O depósito é um estoque de itens da fábrica, armazenados em prateleiras, organizadas geometricamente de forma a facilitar o acesso dos AGVs. Carregadores de bateria são posicionados em determinadas posições ao longo do depósito. Fitas magnéticas foram posicionadas em chão do deposito para guiar os AGVs ao longo do deposito. Um software controla o conjunto de AGVs, gerando e gerenciando ordens de transporte (que podem também serem introduzidas por operadores humanos). Abstraindo os sistemas de transporte tradicionais, sistemas chamados de clientes geram as solicitações de transporte, que são enviados ao sistema de controle dos AGVs, aguardando a confirmação de quando o transporte é realizado. O sistema de controle dos AGVs designa as solicitações de transporte aos AGVs apropriados e determina as rotas em que

os AGVs irão percorrer para atender as solicitações designadas (observando a dinâmica do ambiente). O sistema possui tratamento para evitar colisões dos AGVs e situações de bloqueio dos AGVs ao longo de sua trajetória.

O sistema desenvolvido no projeto EMC_2 ainda necessita um sistema de controle para o AGV individualmente. Esse sistema efetua o controle de baixo nível no AGV mantendo o mesmo na sua rota, determina continuamente a posição AGV e monitora o nível da bateria. O AGV pode realizar um conjunto de ações, tais como: pegar uma carga, movimentar-se ao longo de uma distância especificada, liberar a carga, estacionar e carregar sua bateria. Assim, os sistemas de controle dos AGVs enviam tarefas aos sistemas de controle local dos AGVs, que as executa autonomamente.

Para permitir a movimentação coordenada dos AGVs no projeto EMC_2, o *layout* do armazém é dividido em elementos lógicos: Segmentos e nós. Os segmentos são os caminhos ou rotas, de 3 a 5 metros, que um AGV pode seguir. Um segmento pode ser unidirecional ou bidirecional. Os nós estão no início ou no fim dos segmentos, sendo locais onde um AGV pode parar ou realizar ações como coletar uma carga. Cada segmento e nó possuem um identificador único. Um AGV pode parar nos nós e modificar sua direção.

Os benefícios dos sistemas multi-robóticos podem ser bem compreendidos ao se revisar suas abordagens e suas características. A descentralização do sistema permitiu que as limitações da abordagem puramente centralizada fossem superadas, como por exemplo, o problema da remoção do supervisor ou controlador (ou mesmo mestre) paralisando o sistema (arquitetura centralizada).

A centralização também demanda a necessidade de comunicação. Como boa parte dos sistemas multi-robóticos devem ser capazes de funcionar em ambientes em que a comunicação pode não ser garantida, a abordagem de controle distribuído parecer ser a alternativa mais robusta.

Capítulo 4 – Arquitetura Multi-Agente Proposta para Coordenação Robótica

Este capítulo apresenta a arquitetura multi-agente proposta para coordenação robótica.

4.1 Caracterização do Problema

Um dos problemas que essa pesquisa busca tratar é a necessidade de melhorar a comunicação e integração entre as diversas unidades de um sistema produtivo, muitas vezes formado por robôs autônomos, suas unidades gestoras e os clientes (ou entidades solicitantes dos serviços). As unidades produtivas automatizadas estão normalmente distribuídas em diversos locais numa fábrica, muitas vezes constituídas por computados, por máquinas de automação industrial e por operadores.

O cliente ao requisitar um serviço ou uma missão necessita saber em tempo real se sua solicitação poderá ser atendida ou não no prazo especificado. Desta forma, faz-se necessário uma solução que possa dar essa resposta ao cliente em um tempo satisfatório.

Num cenário em que uma missão ou um serviço é composto por outras missões, um gestor deve avaliar o escopo da solicitação, identificar o conjunto missões e as combinações de operações (ou atividades de transformação considerando um sistema produtivo) para atender a solicitação. Deve ainda, avaliar a disponibilidade de recursos (materiais, máquinas, operadores) para executar as operações relacionadas à ordem de produção ou missão. A partir desta análise o gestor pode responder ao cliente se consegue atendê-lo satisfatoriamente no tempo desejado. Caso haja uma contração do serviço pelo cliente, há a necessidade do acompanhamento eficiente da evolução das atividades relacionadas ao contrato, evitando maiores riscos para o cliente.

Num cenário em que não há uma estrutura hierárquica da missão ou da ordem de produção, da mesma forma um gestor deve avaliar o escopo da solicitação e avaliar a disponibilidade dos recursos para que o cliente seja atendido. O gestor deve, caso haja a contração do serviço por parte do cliente, acompanhar a evolução da atividade contrata ou permitir que o cliente acompanhe diretamente.

Pode-se representar um esquema genérico representando o problema de planejamento das operações ou de missões demandadas considerando os recursos disponíveis. A figura a seguir ilustra esse problema com mais detalhes.

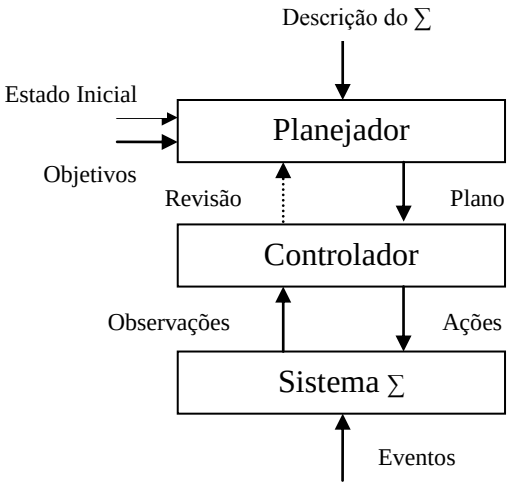

Figura 4-1 Representação de um modelo de um problema de planejamento e execução de operações.

Pode-se verificar na figura 4.1, que o módulo planejador possui como entradas a descrição dos estados do sistema ou do ambiente (∑), o estado inicial e os objetivos a serem atingidos, gerando na saída o plano de trabalho. O controlador recebe o plano, observando o estado corrente do sistema, e gera ações para a execução do plano. As ações modificam o sistema, alterando seus estados. Eventos externos podem também alterar o estado do sistema.

Um das grandes dificuldades do modelo apresentado é a diferença entre a representação do sistema (entrada do planejador) e o mundo real, visto que a representação do sistema é apenas uma abstração do ambiente ou das situações reais para fazer o planejamento possível. Esta diferença pode gerar distorções nos resultados.

Um modelo robusto deve também ter a capacidade de responder aos eventos de forma ágil. Desta forma, o controlador deve supervisionar a execução das ações comparando com o plano e em caso de distorções uma revisão do plano é solicitada ao planejador. Neste caso, o plano pode sofrer modificações pontuais ou um novo plano pode ser gerado. Tem-se desta forma um planejamento dinâmico, através de uma realimentação entre o controlador e o planejador.

Grande parte das pesquisas na área de planejamento automático tem se voltado para o desenvolvimento de planejadores ótimos considerando os recursos distribuídos

disponibilizados e problemas multicritérios, no entanto, há poucas contribuições em técnicas de reparo de planejamento em tempo real para recursos distribuídos heterogêneos. É sabido que grande parte das aplicações o planejador tem uma visão limita da realidade do sistema e não consegue antever as situações que ocorrem em tempo de execução do plano, resultando em grandes distorções do plano quando comparado com a realidade da execução. Assim, nestes cenários de incertezas uma solução robusta de replanejamento, um modelo de execução mais ágil, dando mesmo importância ao perfeito planejamento e mais a reconfigurações em tempo de execução, faz-se necessário.

Uma aplicação importante e que podemos refletir para definição dos requisitos arquitetura proposta é a gestão inteligentes dos AGVs em um SFM, considerando o transporte interno de materiais.

Pérez (2010) define navegação como a metodologia que permite guiar o curso de um robô (AGV) até um destino ao longo de um caminho desejado em um ambiente. A navegação robótica, com seus diversos esquemas, é um processo que normalmente envolve as seguintes tarefas:

- a percepção do ambiente através dos seus sensores;
- o planejamento de uma trajetória livre de obstáculos, visando o destino selecionado;
- e o comando do veículo para o destino.

Um sistema gestão inteligentes de AGVs em um SFM, no que se refere à alimentação de materiais, deve atender as demandas de transporte eficientemente e resolver conflitos entre as entidades (AGVs). Assim, no caso do controlador dos AGVs é necessário que algumas atividades sejam realizadas, como por exemplo, a programação (*scheduling*) dos AGVs, a seleção de rota e acompanhamento da execução do transporte pelo AGV.

Conforme Pérez (2010), a modelagem de ambientes é dividida em duas áreas, os interiores e os exteriores. As aplicações em ambientes exteriores requerem considerável poder computacional para realizar tarefas de controle, num ambiente dinâmico e complexo (com variedade de terrenos e variável densidade de obstáculos). O problema da localização é resolvido com utilização de sistemas de posicionamento global por satélite (GPS).

Os ambientes interiores, mais estruturados (conhecidos) e previsíveis (mediante mapas estabelecidos a priori), são mais facilmente modeláveis. Para o posicionamento

da entidade utilizam técnicas de detecção de marcas ou de referências a priori conhecidas.

Em um SFM pode-se considerar que o ambiente de navegação dos AGVs é estruturado, não havendo variação da posição das unidades produtivas e não havendo fluxo de pessoas.

Outro desafio no projeto de sistemas de AGVs é o planejamento dos caminhos. As diferentes configurações de caminhos para AGVs são (Kusiak, 1985):

- linhas simples (Figura 4.2);
- ciclo simples (Figura 4.3);
- tipo escada (Figura 4.4);
- e rede complexa (Figura 4.5).

Vários estudos têm sido realizados voltados para o problema de sistemas AGVs. No caso de ambientes internos estruturados com redes complexas a ferramenta primária é baseada geralmente em simulação mediante redes Petri (Li *et al.*, 2008).

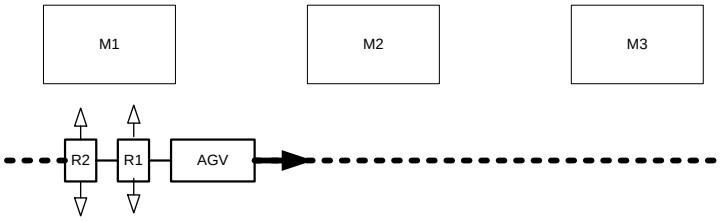

Figura 4.2 Sistema AGV em linha simples (baixa flexibilidade). Fonte: Pérez, 2010

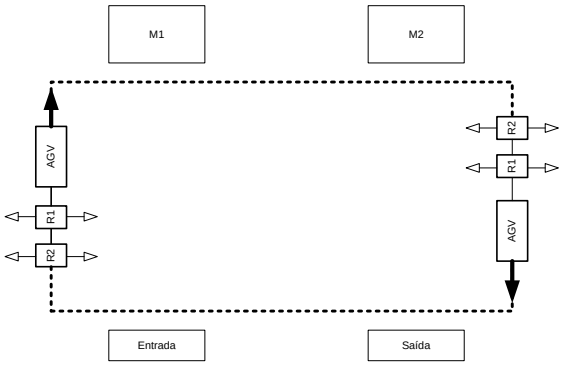

Figura 4-3 Sistema AGV em ciclo simples. Fonte: Pérez, 2010

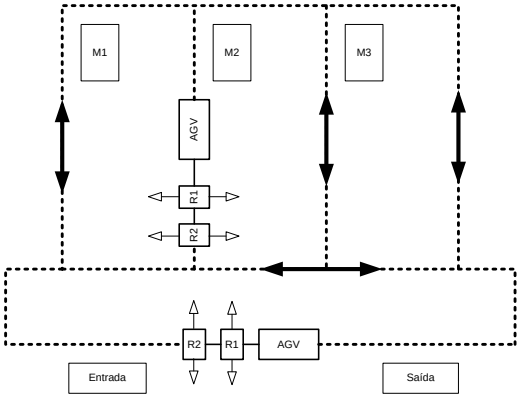

Figura 4-4 Sistema AGV em tipo escada (ladder). Fonte: Pérez, 2010

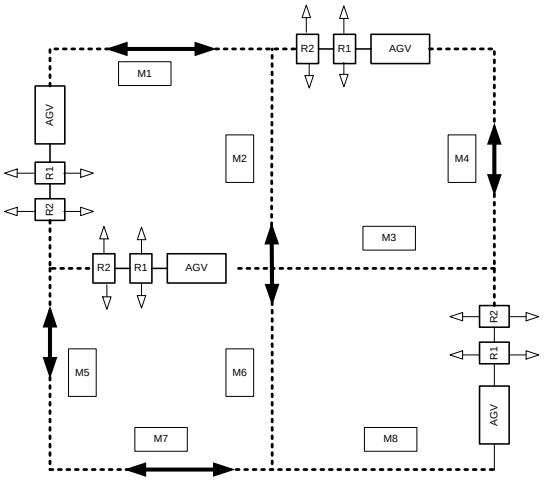

Figura 4-5 Sistema AGV em rede complexa. Fonte: Pérez, 2010

Considerando uma aplicação *Job Shop* ou produção sob encomenda, cada tipo de produto pode ter uma rota diferente. Essas rotas individuais podem ser determinadas por uma entidade planejadora (agente) de processos. A atenção ao atendimento deste requisito é justificável visto que um SFM deve ter capacidade de produzir uma variedade de tipos de produtos ao mesmo tempo em pequenos lotes.

O problema *Job Shop* está relacionado à programação de *J* trabalhos (*Jobs*) ou tarefas para *M* máquinas ou recursos produtivos. Em determinado instante de tempo pode haver uma demanda de *J* trabalhos requerendo o processamento de sequências de máquinas, com seus respectivos tempos de processamento. Assim, busca-se usar os recursos de forma mais eficiente possível para o atendimento da demanda de trabalhos. O objetivo pode ser diminuir o tempo total de execução de todas as operação ou transformações associadas a cada um dos *Jobs*.

Em sistemas *Job Shop*, as seguintes restrições podem ser consideradas e aplicadas ao modelo proposto (Pérez, 2010):

- as operações não podem ser interrompidas e cada máquina pode processar apenas uma operação por vez;
- cada operação só pode ser processada em uma única máquina por vez;
- cada *job* é fabricado em uma seqüência de operações;
- não existe restrição de precedência entre operações de diferentes *jobs*;
- e não existe relação de precedência entre as operações executadas por uma mesma máquina.

Pode-se neste momento destacar alguns aspectos importantes que são requisitos para o sistema proposta nesta pesquisa:

- A necessidade de uma solução que permita a alocação automática dos recursos distribuídos disponíveis para execução das missões demandadas;
- O sistema deve permitir o monitoramente das atividades contratadas a fim de que o cliente possa se sentir confortável quanto a execução das missões;
- O sistema deve permitir no mínimo o reparo do plano estabelecido inicialmente sem qualquer interferência por parte do cliente;
- A arquitetura deve prover uma solução que permita a cooperação entre os agentes distribuídos de modo a não aumentar a sobrecarga de mensagens entre os agentes;
- O protocolo a ser estabelecido deve prover um mecanismo para haja aprendizagem distribuído, a fim de permitir a evolução do sistema como um todo;
- Arquitetura proposta deve ser escalável para aplicações empregando robôs móveis (terrestres ou aéreos) e sistemas *Job Shop*.

4.2 Arquitetura Multi-agente Proposta para Coordenação Robótica

A fim de atender aos requisitos descritos anteriormente apresentados é proposta uma arquitetura de sistema multi-agente para coordenação robótica.

Os agentes podem representar unidades produtivas, células compostas por máquinas, robôs manipuladores ou robôs móveis (AGVs).

A arquitetura proposta permite que sejam construídos sistemas multi-robôs. As tarefas especificadas (ou missões) pelos usuários do sistema podem ser decompostas em operações atribuídas aos robôs individualmente. Essa decomposição é responsabilidade dos clientes do sistema. O sistema aloca as missões demandas pelos clientes. Após a designação das tarefas aos robôs, o sistema monitora a execução das tarefas, a fim de informar ao usuário o estado das missões ou o estado das ordens de produção. No caso de um sistema flexível devem-se identificar situações de falhas para, caso necessário, iniciar um processo de realocação de tarefas (reconfiguração do sistema) para outros robôs.

A arquitetura proposta tem como funcionalidade principal o gerenciamento das missões. Todas as entidades físicas, sejam elas máquinas de automação industrial ou AGVs (robôs), são representadas por agentes virtuais. As entidades físicas devem se comunicar com os agentes, que os representam, de forma individual.

Cada entidade física, juntamente com o agente virtual que o representa, forma um componente autônomo. No caso de robôs (ou AGVs), por exemplo, o agente robô, juntamente com seu *firmware*, formam um componente autônomo. O agente, que representa a entidade física, monitora as alocações e tarefas em execução da mesma. No caso de robôs, parâmetros como nível de bateria, falhas e localização podem ser monitorados. No caso de outras máquinas outros parâmetros também podem ser monitorados, como taxa de falhas de montagem e taxa de paradas de produção. No entanto, o principal parâmetro a ser monitorado é o acompanhamento da execução da missão.

As entidades físicas não precisam se preocupar com as interações diretas com outros robôs ou com algum mecanismo de alocação de tarefas, isto é responsabilidade do seu agente virtual. Esse arranjo arquitetural faz com que as entidades físicas tenham seu comportamento mais centrado no seu controle reativo inteligente e automação local.

Os usuários também são representados por agentes na arquitetura proposta. Os agentes clientes recebem a requisição de tarefas ou ordens de produção do usuário. As ordens de produção ou missões são enviadas aos agentes gerentes ou gestores.

Os agentes gestores devem identificam a missão a ser executada, direcioná-las aos agentes que representam as entidades físicas, ou recursos do sistema, para que um processo que negociação seja estabelecido, e missões sejam designadas a tais entidades. Em um cenário em que a tarefa é composta por operações, uma possibilidade é que um agente possa decompor a tarefa em operações e essas operações possam então ser

designadas para os agentes recursos, que podem executar de forma concorrente caso não haja problemas de precedência.

As operações designadas às entidades físicas devem ser realocadas em caso de falhas de execução. O agente recurso que detectar que sua missão irá falhar ou falhou pode e deve iniciar um processo de realocação, assumindo um papel de cliente temporariamente.

O protocolo de negociação *Contract Net* é empregado na alocação e na realocação de missões ou operações (aos agentes do tipo recurso). De modo geral, quando há uma oferta de uma missão aos agentes representantes das entidades físicas, estes respondem com uma proposta para realizar aquela missão, considerando suas habilidades e suas disponibilidades. O objetivo não estimular a competição, neste sentido a decisão pode ser situacional considerando as melhores possibilidades de maximizar os resultados globais. Os agentes do tipo gerente avaliam as propostas e confirmam a alocação da missão para um dos agentes do tipo recurso. Após a alocação da missão, o agente recurso deve informar a entidade física sobre a tarefa alocada. Durante a execução da tarefa pela entidade física, o respectivo agente do tipo recurso informa ao agente do tipo cliente sobre andamento da tarefa contratada. Em caso de falhas ou riscos elevados devido ao atraso na execução de tarefas, o agente recurso pode abortar a missão corrente a fim de iniciar a renegociação da tarefa em risco, atribuindo-a a outro agente do tipo recurso.

Cada agente da arquitetura proposta pode possuir uma interface para interação com um usuário humano. Neste caso, o agente cliente pode acompanhar a evolução das tarefas em processo de contratação e em execução.

Um agente do tipo cliente ao solicitar a execução de uma tarefa de entrega de materiais de um AVG, por exemplo, poderia ser informado quanto ao progresso da tarefa. Os agentes recursos necessitam de constantes interações com suas respectivas entidades físicas para deliberar sobre as missões negociadas e receber informações sobre a evolução das tarefas contratadas.

Caso o agente gerente não consiga recurso para realizar uma missão, esta pode ficar numa fila até que a tarefa seja alocada. A ordem de produção ou missão pode também ter sua data expirada, caso não seja resolvida essa questão de falta de recurso.

Apesar de não fazerem parte do escopo desta pesquisa, vislumbra-se que outros agentes podem ser introduzidos no sistema com o objetivo de fornecer serviços interessantes:

- Agente planejador de rotas – Esse agente possui o registro de todas as rotas e distâncias no ambiente.
- Agente compartilhador de informações (ou agente banco de dados - BD) – Possui informações (proposta dos agentes, estado das ordens já alocadas, ordens pendentes de alocação) que podem ser compartilhadas entre os agentes.

Agentes que representa recursos devem, em certos casos, assumir o papel de agentes clientes, solicitando serviços de outros recursos. Um agente representando uma estação de trabalho poderia contratar um agente AGV para lhe garantir a entrega de materiais para montagem. Assim, esse agente estação de trabalho assume o papel de cliente e negocia a realização de uma tarefa com agentes AGVs. Apesar de um ser uma possibilidade na arquitetura proposta, as subcontratação só foram desenvolvidas para o caso de situações em que o agente recurso não conseguiu finalizar sua tarefa.

A figura a seguir apresenta o modelo de negociação de tarefas entre os agentes do tipo gerente e os agentes do tipo recurso. O agente gerente envia solicitações de propostas aos agentes do tipo recurso. Os agentes do tipo recurso, com conhecimento sobre suas capacidades, podem retornar propostas para execução da tarefa ao agente gerente. Este avalia as propostas recebidas e contrata a melhor considerando os objetivos globais, enviando uma confirmação de alocação ao agente recurso escolhido. Na fase de execução o agente recurso deve informar o agente cliente ou agente gerente sobre o andamento do serviço e sua finalização.

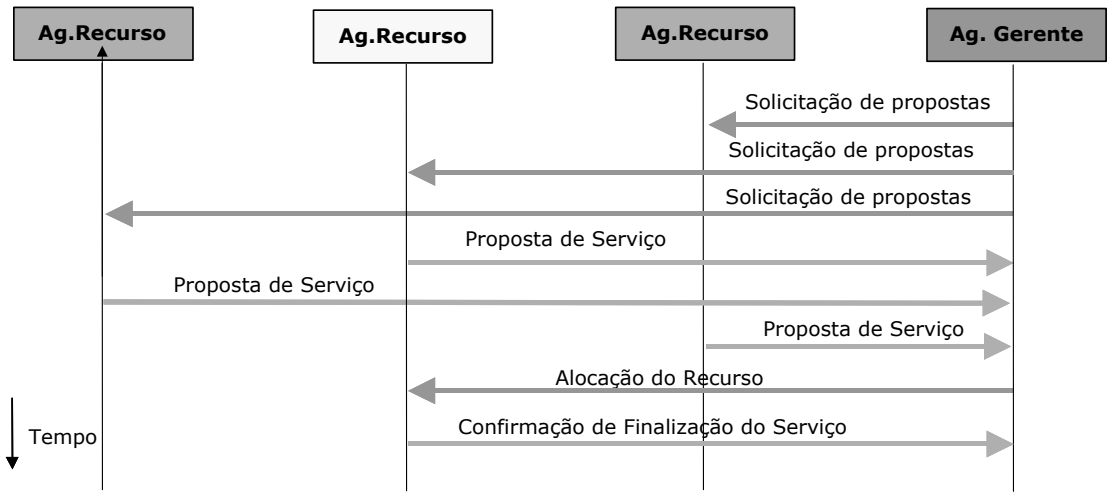

Figura 4-6 Contract Net Protocol (CNP)

Uma visão simplificada da arquitetura é apresentada na figura 4-7. Observa-se que o usuário interage com o agente cliente passando missões ou ordens de serviço. Em seguida, essas tarefas são negociadas em um segundo nível entre os agentes do tipo gerente e os agentes do tipo recurso. Pode haver vários agentes do tipo cliente, agentes do tipo gerente e agentes do tipo recurso. Os agentes recurso devem atualizar os agentes cliente sobre o *status* de suas ordens de serviços. Os agentes recurso interagem com as entidades físicas enviando comandos e recebendo informações sobre o andamento das tarefas.

A arquitetura promove uma separação entre o meio lógico e o meio físico, no entanto, pode ser necessário especificar ou determinar uma camada entre os dois meios, ou *middleware*. A função desta camada é prover um meio de comunicação entre o sistema multi-agente e as entidades físicas. Outra abordagem é que cada agente recurso cuide, de forma descentraliza e individual, da interface com sua respectiva entidade física.

Os agentes recursos devem monitorar suas respectivas entidades físicas, em caso de problemas de comunicação ou falhas nas entidades, o agente recurso deve informar o agente cliente relacionado com a ordem de serviço em processo e outras que por ventura tenham sido agendadas.

O principal objetivo da arquitetura proposta é o gerenciamento de missões ou encomendas, conforme demanda, de forma automática, considerando a capacidade e a disponibilidade de recursos.

O modelo de micro-economia de auto-organização tem sido adotado para controle e planejamento de produção, em alguns vários trabalhos como o de Sousa P. e Ramos, C. (1997).

O comportamento flexível e guiado pela demanda é o requisito mestre dos sistemas de manufatura modernos para o atendimento as necessidades dos mercados globalizados. A tecnologia baseada em agentes, empregando modelo distribuído, flexível e autônomo, pode agilmente responder a distúrbios e mudanças inesperadas no ambiente. Algumas características são altamente desejáveis nestes sistemas modernos de manufatura flexíveis:

- Arquitetura multi-nível aberta;
- Avançada capacidade de comunicação;
- Tomadas de decisão descentralizada;

- Habilidades de auto-organização;
- Sistema reconfigurável, adaptável com certos níveis de autonomia;
- Mecanismos de coordenação, cooperação e interatividade orientadas para realização de tarefas (vinculadas a competência).

A arquitetura proposta foi desenhada para atender a estes requisitos sistemas modernos de manufatura flexíveis citados.

Abordando um pouco mais acerca da figura 4-7, pode-se verificar que realmente os agentes são entidades do mundo virtual que representam as entidades físicas sendo elas unidades produtivas, processos automatizados, máquinas, robôs ou até mesmo humanos. Os agentes gerente de processo, controle de materiais e controle de qualidade, são agentes de suporte ao agente gerente. Apesar de representados neste esquema não fazem parte do modelo genérico especificado e não foram desenvolvidas regras de negócios ou simulações para os mesmos, representam apenas possibilidades de serviços a serem disponibilizados no caso de um sistema multi-agente empregado em um ambiente de manufatura.

O modelo proposto deve permitir a criação de várias instâncias do agente gerente.

No caso de um cenário de manufatura sob encomenda. O agente cliente demanda uma solicitação de produção ao agente gerente, ou este é automaticamente criado a partir da criação da solicitação. Ao receber a solicitação o agente gerente deve confirmar (com agente gerente de processo) se há lista de material (BOM ou *Build Of Material*) e plano de processo (com a precedência entre as tarefas ou operações). Caso haja BOM e plano de processo, o gerente deve confirmar (com o agente controle de materiais) a disponibilidade dos itens da lista de materiais do produto em questão. O agente gerente pode em seguida iniciar a negociação das operações (ou tarefas) vinculadas à ordem de produção, com suas respectivas precedências, com os agentes recursos. O agente gerente receberá as propostas de execução das tarefas dos agentes recursos, selecionando a melhor combinação que atende aos requisitos apresentados pelo agente cliente (por exemplo, o atendimento mais rápido ou menor tempo de entrega). O agente gerente deve informar e reservar os recursos selecionados. Em seguida, o agente gerente pode fazer uma proposta de entrega do produto ao agente cliente. Após a confirmação do agente cliente, o agente gerente pode contratar definitivamente os agentes recursos selecionados e reservados.

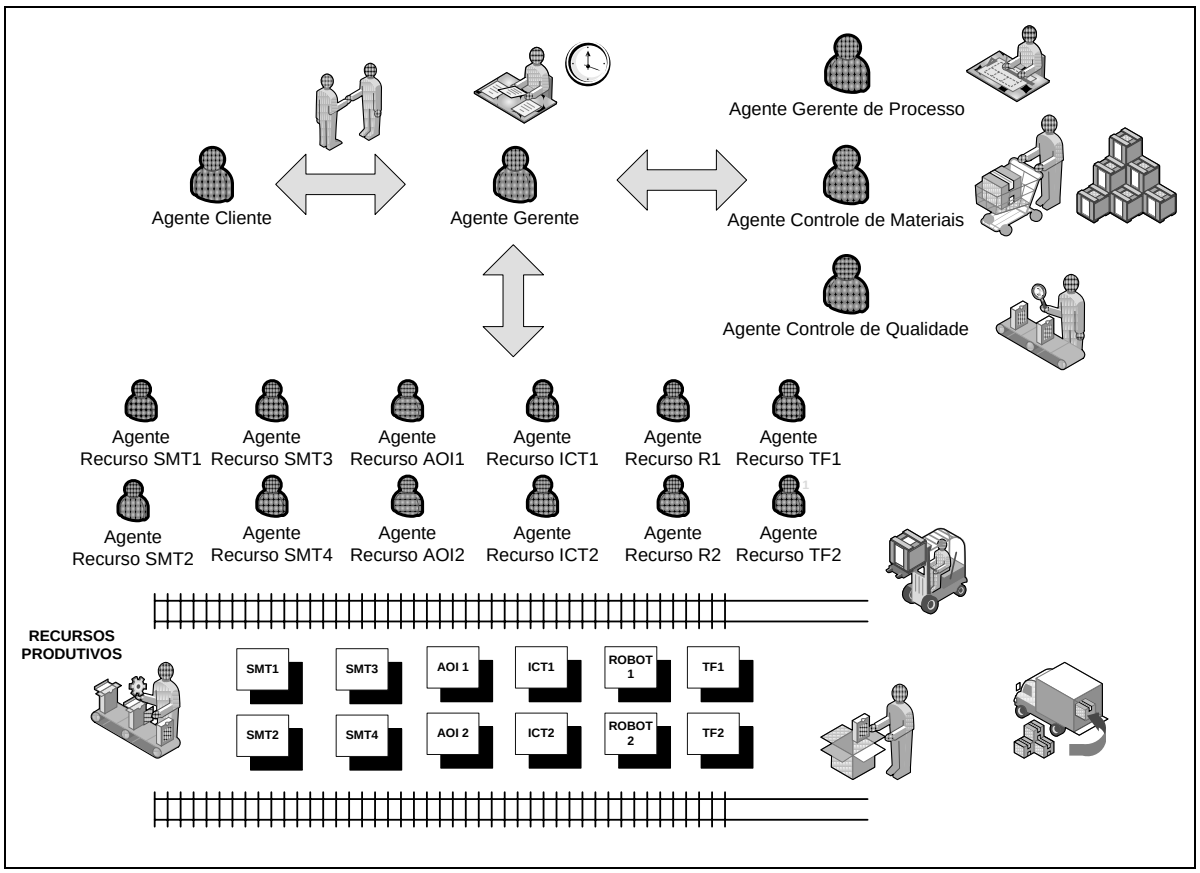

Figura 4-7 Visão de alto-nível da arquitetura proposta

A seguir são apresentadas algumas características dos agentes da arquitetura proposta:

- Agente Cliente – O agente do cliente fornece a interface do usuário ao sistema. Ele gerencia e anuncia as ordens ou missões, e lida com mudanças dinâmicas nas condições das ordens.
- Agente Recurso – O agente que representa as unidades físicas, máquinas, robôs e etc. O agente tem conhecimento sobre a agenda da unidade, a competência ou habilidades da unidade, da capacidade e etc.
- Agente Gerente – Representa o gestor da missão que recebe dos agentes clientes e realizando assim a contratação dos recursos para executar as missões demandadas.
- Agente Gerente de Processo (opcional e específico para aplicações de manufatura) – Fornece a lista de materiais e os planos de processo

relacionado aos produtos de trabalho. Esse agente deve receber a notificação do plano estabelecido para a ordem de produção, juntamente com os recursos alocados, a fim de alimentar os postos com instruções de trabalho, caso necessário, e acompanhar o processo produtivo.

- Agente Controle de Materiais (opcional e específico para aplicações de manufatura) – Esse agente deve informar ao gerente a disponibilidade de materiais, deve ainda estimar a chegada dos itens, gerenciar o estoque produtivo. O agente controle de materiais deve ser contratado pelo agente para realizar o transporte de materiais relacionados à ordem de produção.
- Agente Controle de Qualidade (opcional e específico para aplicações de manufatura) – Define os critérios de qualidade e acompanha a evolução de execução tarefas monitorando os indicadores de qualidade. O agente controle de qualidade pode interromper o processo produtivo considerando os indicadores de qualidade do processo e do produto, o que pode resultar em uma reconfiguração do sistema.

A estratégia adotada para o caso de falhas das missões por parte dos agentes recursos foi a subcontratação de outro agente do tipo recurso para execução da missão. Ou seja, o recurso que inicialmente foi contratado para executar uma missão para um cliente, e falhou na sua missão, passa agora a exercer o papel de cliente. Este subcontrata um novo agente do tipo recurso, para isso faz nova solicitação de contratação para um agente do tipo gerente. Uma vez subcontratado, o novo agente do tipo recurso será responsável pela execução da missão. Finalizada a missão com sucesso, o novo recurso contratado pode compartilhar seu conhecimento com o agente do tipo recurso que falhou na missão, a fim de prover um mecanismo de aprendizado distribuído para o sistema.

A figura a seguir ilustra a topologia de arquitetura proposta. Os agentes do tipo cliente estão no nível mais alto da arquitetura e os agentes do tipo gerente em um nível intermediário. Observa-se que os agentes do tipo recurso então em níveis baixo da arquitetura, representando as entidades do nível físico. Os agentes do tipo recurso podem interagir com as entidades físicas através de comunicação sem fio ou através de interfaces disponíveis em equipamento de automação industrial. As plataformas JADE e MaDKit foram utilizadas no desenvolvimento dos protótipos para validação da arquitetura proposta. Assim, os agentes podem ser construídos sobre essas plataformas.

A plataforma JADE parece ser mais indica para o desenvolvimento final do sistema, enquanto a MaDKit parece mais apropriada para o desenvolvimento dos protocolos de negociação e simulação de projetos multi-agentes.

Outras possibilidades de desenvolvimento, mas que não foram empregadas nesta pesquisa são a plataforma Mobile-C e a ferramenta Labview. A plataforma Mobile-C pode ser empregada, pois permite uma interoperabilidade com a plataforma JADE, facilitando a interação com elementos da camada física, visto que suporta C e C++. A ferramenta Labview, da National Instruments, também pode ser empregada como de interface com elementos da camada física (máquinas e equipamentos de automação industrial).

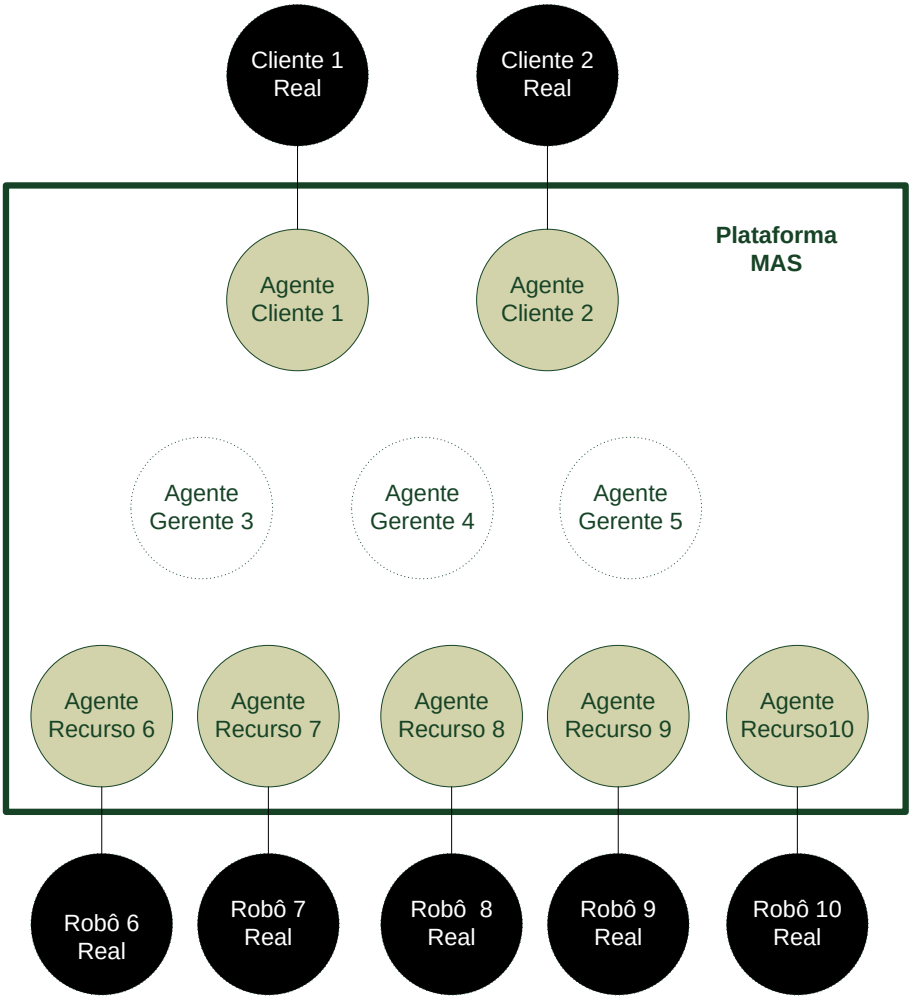

Figura 4-8 Topologia da arquitetura proposta.

A figura a seguir mostra o protocolo de negociação *Contract Net* modificado empregado na arquitetura proposta. Pode-se destacar alguns pontos de inovação

considerando as abordagens apresentadas por outras pesquisas que utilizaram similar abordagem:

- A conexão direta entre clientes e recursos após o processo de contratação ter sido finalizado, fazendo com que o agente gerente tenha um papel apenas no processo de seleção inicial dos recursos;
- O cliente pode ser informando sobre o andamento da missão visto que há uma conexão direta com o recurso responsável;
- Em caso de falha na execução, o agente recurso responsável solicita ao agente gerente a subcontratação de outro recurso para finalização da missão, inicialmente sem informar ao cliente a falha ou abandonar a missão;
- Caso haja subcontratação o agente cliente não é afetado e a missão pode ser finalizada com sucesso;
- Caso o recurso subcontratado finalize a missão com sucesso, ele poderá compartilhar seu conhecimento com recurso que falhou e o contratou, fazendo com que o sistema multi-agente como um todo evolua;
- O conhecimento compartilhado pode estar relacionado a uma melhor configuração de uma máquina, ao melhor conhecimento do ambiente de atuação de um robô, a melhor configuração dos parâmetros de controle do robô e etc.

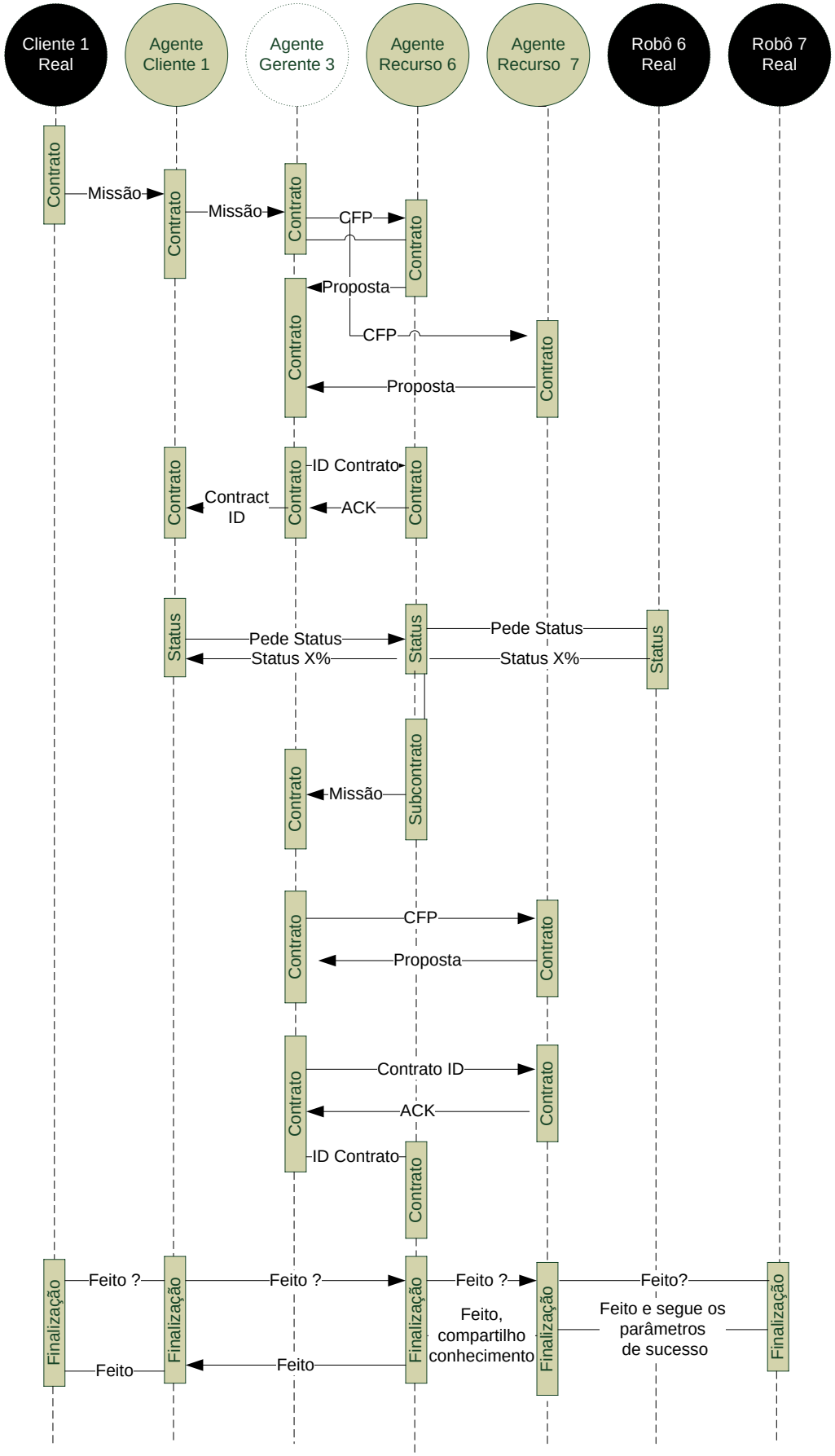

Figura 4-9 Contract Net Modificado Empregado na Arquitetura Proposta

Capítulo 5 – Detalhes Construtivos da Arquitetura Proposta

Neste capítulo são apresentados os detalhes construtivos da arquitetura proposta. Primeiramente, o protocolo *Contract Net* é apresentado como mais detalhes, visto que uma proposta de sua extensão é um ponto de inovação desta pesquisa. Em seguida, são abordadas algumas plataformas que foram utilizadas no desenvolvimento da arquitetura proposta. Por fim, são apresentados detalhes construtivos dos protótipos desenvolvidos para validar a arquitetura proposta.

5.1 Contract Net Protocol

Para Vokřínek *et. al* (2007), vários mecanismos de negociação são usados para buscar um acordo entre agentes a respeito de uma solução de cooperação mutua vantajosa. As negociações podem ocorrer em paralelo (1:m – "um" para "m" negociações), normalmente associadas a protocolos de negociação baseados em leilões. Essas negociações podem ser interativas (como as conhecidas, do inglês como *English auction* e como *Dutch auction*) ou de um lance (como as conhecidas, do inglês como *First-price-sealed-bid auction* e *Vickrey auction*).

Um dos mais conhecidos protocolos de negociação é o *Contract Net Protocol* ou CNP (Smith, 1980), como os outros anteriormente citados, também é padronizado pela FIPA (*Foundations of Intelligent Physical Agents*). Estes protocolos tratam da fase de negociação não cobrindo a fase de execução. A conclusão com sucesso das tarefas alocadas é o elemento motivador para um comportamento cooperativo almejado. No entanto, deve-se buscar uma possível evolução na atitude dos agentes, como por exemplo, a desistência das tarefas considerando o não atendimento aos requisitos de qualidade acordados (Vokřínek *et. al*, 2007). Esta necessidade se aplica em várias situações no cotidiano de indústrias em função da elevada dependência de terceiros (para fornecimento de insumos e componentes para o processo produtivo). Assim, Vokřínek *et. al* (2007), propôs um protocolo de negociação interativo que foi designado para prover meios para contratações flexíveis e robustas em um ambiente competitivo, protocolo este inspirado no CNP.

Uma cooperação entre agentes é definida como uma relação entre provedor-cliente com condições definidas (qualidade, preço, data de vencimento, penalidades e outras). Os ambientes e as regras de interação destes agentes podem ser colaborativas ou competitivas.

Em um cenário colaborativo, os agentes normalmente possuem um objetivo comum que eles tentam cooperativamente cumprir. Em outros casos, por exemplo, os agentes podem possuir diferentes objetivos, mas sua motivação primária é maximizar seu bem estar (prosperidade) social, somatório dos lucros ou resultados individuais (Vokřínek *et. al*, 2007).

No caso de agentes competitivos, a motivação principal é a maximização dos seus resultados individuais, não interessando a prosperidade social da comunidade. Os agentes, com interesses próprios, podem estabelecer um processo de cooperação para atingir um objetivo comum da comunidade, somente se houver uma contribuição para maximização dos seus resultados ou interesses próprios.

Em ambientes colaborativos, os agentes mantêm seus contratos quando os resultados da comunidade são maximizados. Quando os resultados globais são diminutos ou uma oportunidade de colaboração melhor chega, os agentes podem livremente liberar seus contratos ou um processo de reconfiguração de tarefas ou alocações pode ocorrer. O comportamento colaborativo dos agentes, com reconfigurações e liberação de contratos, assegura a maximização da prosperidade social da comunidade. Em ambientes competitivos, o contrato pode ser mantido com penalidade atribuída ao agente em caso de não atendimento do compromisso estabelecido.

O CNP é um protocolo empregado em uma rede multi-agentes composta por um número ilimitado de nós (ou agentes). Há dois tipos de agentes, os gerentes e os contratados (ou fornecedores). Os gerentes possuem habilidades de processar dados, analisar informações e designar tarefas aos agentes fornecedores (ou recursos) com maior poder de execução. Para um dado problema, os agentes fornecedores fazem proposta para um agente gerente considerando sua solução. O agente gerente escolhe a melhor proposta, contratando o agente fornecedor selecionado.

Um dos desafios do CNP é a questão de que um agente gerente pode contratar um agente fornecedor que não é o mais adequado para a tarefa, visto que este pode já estar alocado ou parcialmente alocado para outra tarefa. Outro desafio, no caso para

sistemas multi-robóticos em que a comunicação pode não ser confiável ou garantida, é a dependência de constante interação entre as entidades distribuídas.

A figura a seguir mostra a dinâmica de mensagens no CNP. O agente *Initator* solicita propostas aos outros agentes lançando uma mensagem CFP (*Call for Proposals*) com a especificação da tarefa e das condições de execução. Os agentes *Participants* recebem a mensagem de CFP, sendo assim potenciais contratantes da proposta. Estes agentes *Participants* podem submeter suas propostas para executar a tarefa, incluindo pré-condições de execução como preço, tempo ou data em que a tarefa será finalizada e outras pré-condições. Outros agentes *Participants* podem recusar a chamada por propostas. Passado o período do estado de proposta definido (*propose state)*, o agente *Initiator* começa a avaliar as propostas recebidas e selecionar os agentes para executar a proposta. Pode ser escolhida uma proposta, várias propostas ou nenhuma, caso as propostas recebidas não sejam satisfatórias. Em seguida, na etapa de contratação (*contract state*), o agente ou os agentes selecionados recebem uma notificação de aceite e os demais agentes recebem notificação de rejeição. Quando há o recebimento da notificação de aceite da proposta, o agente *Participant* está assim comprometido com a execução da tarefa. Ao final da execução da tarefa o agente *Participant* deve enviar uma mensagem ao agente *Initiator* informando a finalização da tarefa. Em caso de falhas na execução o agente *Initiator* também deve ser informado.

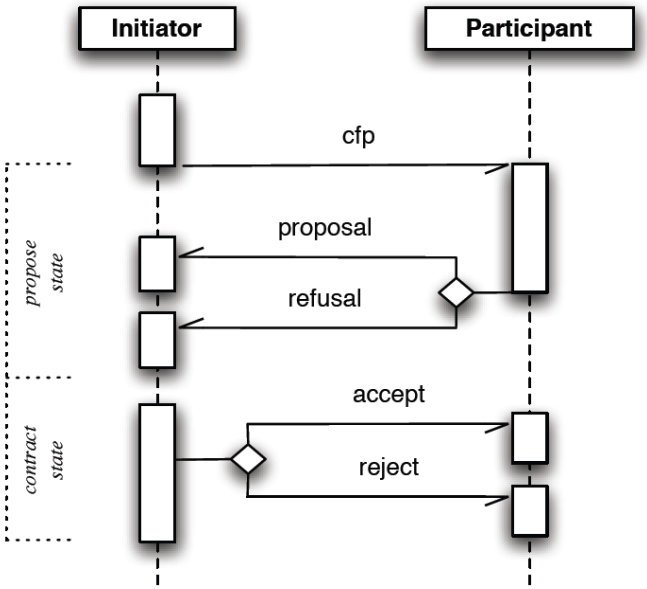

Figura 5-1 Contract Net Protocol. Fonte: JADE (http://jade.tilab.com)

O CNP foi padronizado pela FIPA em 2002, sendo largamente empregado em várias aplicações com algumas modificações.

Os protótipos desenvolvidos e as simulações realizadas nesta pesquisa tiveram como ponto de partida o protocolo *Contract Net* apresentado neste tópico.

5.2 Plataformas de Sistemas Multi-Agentes Utilizadas

Há diversas opções de linguagens e plataformas para desenvolvimento de sistemas multi-agentes. O emprego de uma plataforma ou *framework* facilita o desenvolvimento, pois permite dar maior foco no problema a ser resolvido com a abordagem multi-agentes, em vez de maiores preocupações construtivas.

5.2.1 MaDKit

A plataforma MaDKit foi uma das escolhidas para prototipação da arquitetura proposta, considerando sua simplicidade para o desenvolvimento de protótipos de sistemas multi-agentes e também a simplicidade do desenvolvimento de simulações.

MaDKit é uma plataforma de desenvolvimento baseada em sistema multi-agentes escrita em Java. MaDKit foi desenvolvida para permitir essa fácil construção de aplicações distribuídas e simulações, usando paradigma multi-agentes.

A plataforma possui as seguintes características:

- Permite a criação de agentes artificiais e o gerenciamento do seu ciclo de vida;
- Possui uma infraestrutura eficiente para comunicação entre agentes;
- Possui uma elevada heterogeneidade na arquitetura dos agentes, sem um modelo pré-definido;
- Possui uma infraestrutura de simulação com ferramentas de monitoração e fácil desenvolvimento;
- Facilidade no desenvolvimento de aplicações distribuídas.

Em contraste com as abordagens tradicionais que são muito centradas na arquitetura dos agentes, a MaDKit é uma plataforma centrada na organização, ou do inglês *organization-centered approach* (OCMAS), de modo que não há modelo pré-

definido de agentes. MaDKit é construído sobre um modelo organizacional AGR (Agent/Group/Role), em que os agentes desempenham papéis em grupos e, assim, criam sociedades artificiais.

Um agente na plataforma MaDKit é apenas uma entidade com capacidade de comunicação ativa e que exerce papéis dentro de grupos específicos. Os grupos são definidos a partir da agregação de agentes. Ou seja, cada agente é parte de um ou mais grupos. Em sua forma mais básica, o grupo é apenas uma maneira de marcar um conjunto de agentes. Os grupos possuem as seguintes características:

- Um agente pode ser membro de n grupos ao mesmo tempo;
- Os grupos pode se sobrepor livremente;
- Um grupo pode ser fundado por um agente, e um agente pode solicitar a admissão em qualquer grupo;
- Um grupo pode estar concentrado em um local ou distribuídos em diversas máquinas.

O papel de um agente é uma representação abstrata da função de um agente dentro de um grupo. Cada agente pode possuir diversos papéis ou funções. Os papéis devem ser solicitados pelos agentes candidatos. Esses papéis guiam o processo de construção do sistema multi-agentes através da modelagem de um esquema de comunicação abstrato.

A plataforma está disponível no endereço http://www.madkit.org.

5.2.2 JADE

A plataforma JADE (*Java Agent DEvelopment framework*) também foi escolhida para prototipação da arquitetura proposta. JADE é uma plataforma ou um arcabouço para o desenvolvimento em JAVA de sistemas multi-agentes. Os agentes podem estar locais ou máquinas diferentes (utilizando até diferentes sistemas operacionais). O arcabouço JADE tem se destacado como plataforma multi-agente e vem sendo utilizada por diversos pesquisadores, e segue o padrão FIPA (*Foundation for Intelligent Physical Agents*). JADE simplifica o desenvolvimento de sistemas multi-agentes através de uma camada de software ou *middleware*, de um conjunto de ferramentas gráficas, de várias bibliotecas de desenvolvimento e diversos serviços como o de páginas amarelas.

Para comunicação entre os agentes, a plataforma utiliza o padrão de protocolos FIPA.

Cada instância do ambiente de execução é chamada de contêiner (*Container*). A plataforma JADE pode ser constituída de vários contêineres. Cada contêiner, por sua vez, pode possuir diversos agentes. Há um contêiner principal (*Main container*) que deve estar sempre ativo na plataforma, sendo o primeiro a ser iniciado. Os outros contêineres devem assim ser referenciados ao contêiner principal. Os contêineres devem ser informados do endereço do *Main container* na rede e sua porta.

O *Main container* é responsável pelo sistema de gerenciamento de agentes e pelo serviço de páginas amarelas, através do agente AMS (*Agent Manager System*) e do agente DF (*Directory Facility*). Estes dois agentes estão presentes apenas nos *Main containers* e são iniciados automaticamente quando os *Main containers* são iniciados.

O AMS provê o serviço de nomes, assegurando que cada agente possui um nome único na plataforma, e possui certa autoridade, como a de criar e de eliminar agentes dos contêineres da plataforma.

A figura a seguir ilustra o arranjo ou topologia de contêineres empregado na plataforma JADE. Pode-se observar que a plataforma 1 (um) é composta por três contêineres (um *Main conteiner* e dois contêineres). A plataforma 2 (dois) é formada por um contêiner, o *Main contêiner*. Os agentes são identificados por um nome único, podendo se diferenciados até em plataformas distintas, como os agentes A4 e o A5.

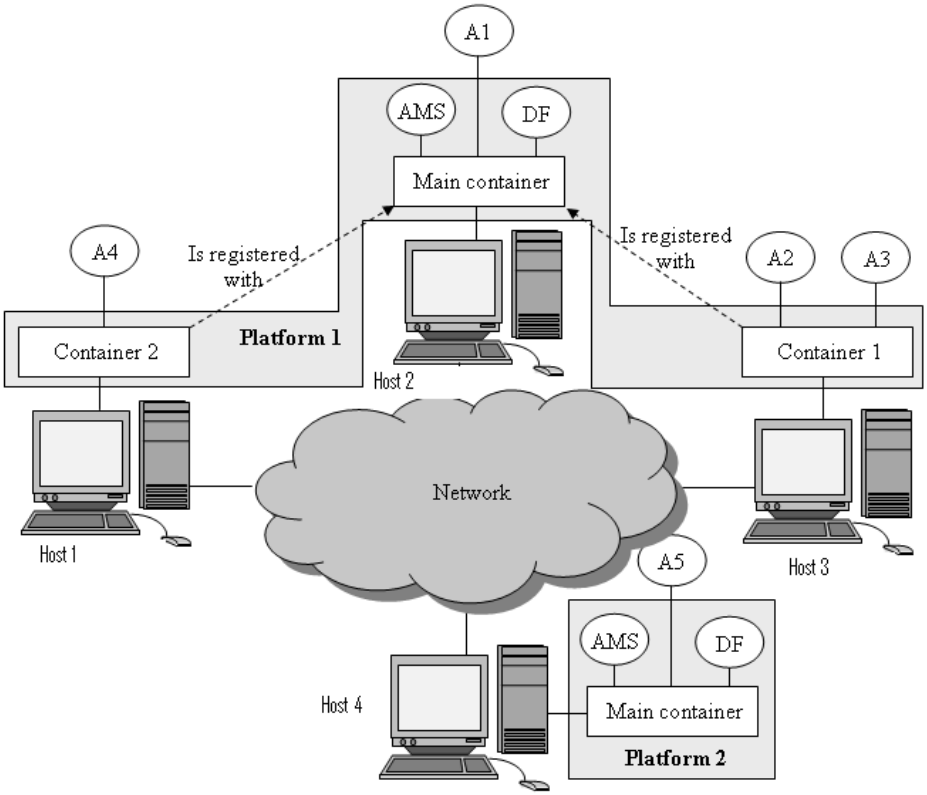

Figura 5-2 Exemplo de arranjo de contêineres empregados na plataforma. Fonte: JADE (http://jade.tilab.com)

A plataforma JADE comporta o desenvolvimento de sistemas distribuídos em diferentes maquinas (mesmo com diferentes sistemas operacionais). Trata-se de um *software* livre (*open source*) distribuído pela Telecom Itália, nos termos da licença LGPL (*Lesser General Public License Version 2*).

A FIPA é uma organização, que faz parte da IEEE *Computer Society* a partir de 2005, e promove a padronização da tecnologia baseada em agentes e sistemas multi-agentes, além da interoperabilidade com outras tecnologias.

Os agentes FIPA, como processos de software, possuem um ciclo de vida que deve ser gerenciado pela plataforma empregada, neste caso a JADE.

Conforme especificado pela FIPA (*Foundation for Intelligent Physical Agents*, 2004), o ciclo de vida de um agente apresenta os seguintes estados conforme figura a seguir:

- Iniciado (*Initiated*) – O agente está construído mais não registrado no AMS, não possui nome e nem endereço, não podendo interagir com outros agentes;

- Ativo (*Active*) – O agente é registrado no AMS, possui nome e endereço, podendo acessar as funcionalidades da plataforma;
- Suspenso (*Suspended*) – O agente está suspenso e nenhuma ação pode ser executada;
- Aguardando (*Waiting*) – O agente está bloqueado, aguardando alguma ação ou alguma mensagem;
- Excluído (*Deleted*) – O agente é finalizado juntamente com seu registro no AMS;
- Trânsito (*Transit*) – Um agente móvel entra neste estado quando migra para outra localidade, armazenando mensagens que serão enviadas a sua nova localização.

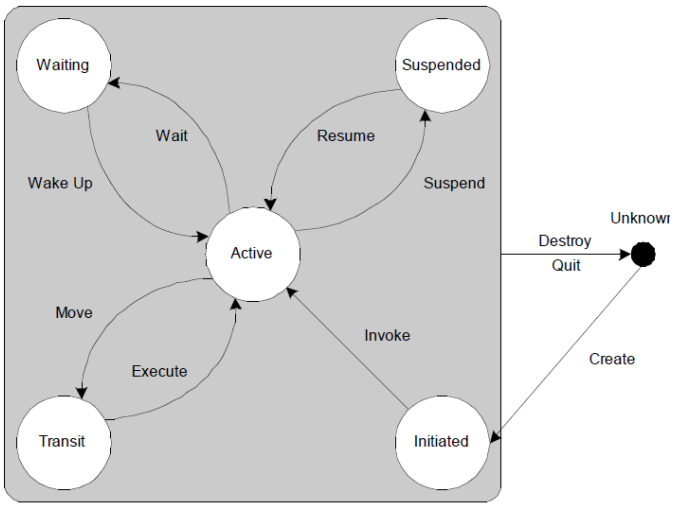

Figura 5-3 Ciclo de vida de agentes FIPA (*Foundation for Intelligent Physical Agents*, 2004).

Uma das vantagens dos sistemas distribuídos e do uso de *frameworks* como JADE, é que a carga de processamento fica distribuída em diferentes agentes. Cada agente deve possuir um algoritmo que determinará seu comportamento e permitirá a interação com outros agentes, buscando satisfazer os objetivos globais.

5.2.3 Jason

Jason é uma plataforma de desenvolvimento de sistemas multi-agentes. Jason é uma extensão da linguagem de programação orientada a agente chamada AgentSpeak, linguagem usada para programar o comportamento individual de agentes. (Bordini *et.*

al, 2007). A plataforma Jason é desenvolvida em Java (multi-plataforma), permite a customização dos agentes e sistemas multi-agentes, e usa diferentes infraestruturas como JADE e SACI, que atuam como *middleware* de sistemas distribuídos baseados em agentes. Jason é um software código aberto e é distribuído sob licença LGPL.

A Jason foi escolhida nesta pesquisa como plataforma de simulação dos protocolos de negociação definidos na arquitetura proposta.

5.2.4 Mobile-C

A plataforma Mobile-C não foi empregada nesta pesquisa, mas é aqui apresentada de forma sucinta visto a possibilidade de pesquisas futuras, considerando as características desta plataforma.

A Mobile-C foi desenvolvido como uma plataforma para desenvolvimento de sistemas multi-agentes móveis de acordo com o padrão FIPA, com foco em aplicações mecatrônicas e de sistemas embarcados. Como esses sistemas utilizam normalmente linguagem C/C++, Mobile-C escolheu C/C++ como linguagem, para o desenvolvimento de aplicações com interface com programas de controle de baixo nível. A plataforma Mobile-C usa Ch (interpretador C/C++) como suporte a execução dos agentes móveis C/C++.

Os componentes da plataforma Mobile-C incluem: AMS (*Agent Management System*), ACC (*Agent Communication Channel*), ASM (*Agent Security Manager*), DF (*Directory Facilitator*) e AEE (*Agent Execution Engine*). O AMS gerencia o ciclo de vida dos agentes. O ACC é responsável pelo roteamento de mensagens entre entidades locais e remotas. As mensagens usam linguagem de comunicação de agentes (ACL). O ASM é responsável pela formação da política de segurança da plataforma. O DF provê o serviço de páginas amarelas. Os agentes do sistema podem registrar seus serviços no DF para prover para a comunidade. Os agentes podem também buscar serviços no DF. O AEE serve como força de trabalho para os agentes móveis. Os agentes móveis tem que estar vinculados a um AEE para serem executados. O agente móvel é empacotado em uma mensagem de agentes (FIPA ACL) em XML (*extensible markup language*). A mensagem também contém outros dados (especificando estados, itinerário e tarefas) encapsulados.

5.3 Detalhes construtivos dos protótipos desenvolvidos

A prototipagem foi a forma escolhida para validação da arquitetura multi-agente para coordenação robótica proposta. O foco dos protótipos foi exclusivamente a validação da camada deliberativa da arquitetura proposta, visto que o comportamento reativo individualizado da cada agente recurso extrapola os objetivos estabelecidos desta pesquisa.

Apesar de dois protótipos terem sido desenvolvidos, o protótipo empregando a plataforma MaDKit validou praticamente todos os aspectos inovadores propostos da pesquisa. Os principais aspectos inovadores podem ser relembrados a seguir:

- O cliente deve ser informado sobre o andamento da missão;
- Em caso de falha na execução o agente recurso responsável pode subcontratar outro agente recurso;
- Caso haja subcontratação o agente cliente não é afetado e a missão pode ser finalizada com sucesso;
- Caso o recurso subcontratado finalize a missão com sucesso, ele poderá compartilhar seu conhecimento com o agente recurso que falhou e o contratou.

Simulações utilizando a plataforma JASON sobre a JADE também foram realizadas com a finalidade de avaliar conjuntamente a robustez da JADE, o uso da JASON no desenvolvimento e a simulação de sistemas multi-agentes e o protocolo *Contract Net* sobre as plataformas JADE e JASON.

Um segundo protótipo foi desenvolvido sobre a plataforma JADE, buscando a obtenção uma analise mais apurada das dificuldades do desenvolvimento de um sistema multi-agentes, considerando a possibilidade de uma estrutura mais complexa de tarefas.

5.3.1 Protótipo da Arquitetura Proposta usando Plataforma MaDKit

Conforme definido na arquitetura multi-agente para coordenação robótica proposta no capítulo 4, no protótipo utilizando a plataforma MaDKit foram criados três tipos de agentes: agente cliente, agente gerente e agente recurso.

A estrutura organizacional do sistema, disponibilizada como interface em linguagem Java, apresentada a seguir (ver trecho de código) é formada por uma

comunidade, denominada de COMMUNITY="fabrica" e por dois grupos agentes CLIENTES_GROUP="fabrica-clientes" e RECURSOS_GROUP = "fabrica-recursos". Os papéis definidos na arquitetura são de CLIENTE_ROLE = "cliente", de GERENTE_ROLE = "gerente" e de RECURSO_ROLE = "recurso".

```
public interface MarketOrganization {
        public static final StringCOMMUNITY = "fabrica";
        public static final StringCLIENTES_GROUP     = "fabrica-clientes";
        public static final StringRECURSOS_GROUP = "fabrica-recursos";
        public static final StringCLIENTE_ROLE        = "cliente";
        public static final StringGERENTE_ROLE        = "gerente";
        public static final StringRECURSO_ROLE        = "recurso";
}
```

Conforme trecho de código apresentado a seguir os agentes do protótipo foram iniciados a partir de um método main(String[] args). Neste caso, foram instanciados três agentes do tipo gerente, dois agentes do tipo cliente e vinte agentes do tipo recurso. O agente console permite a utilização de vários recursos disponibilizados na plataforma.

```
public static void main(String[] args) {

    Gerente.nbDeClientesCriados = 0;
    Gerente.nbDeGerenteCriados = 0;

    new Madkit(Option.launchAgents.toString(),
        MySchedule.class.getName()+",true;"
        + Gerente.class.getName() + ", true,3;"
        + Cliente.class.getName() + ", true,2;"
        + Recurso.class.getName() + ",true,20;"
        + ConsoleAgent.class.getName());
      };
  }
```

O protótipo permite que sejam inicializados diferentes números de agentes clientes, gerentes e recursos. Para evitar um acumulo de agentes do tipo cliente sem respostas é sugerido que sempre haja um número maior de agentes do tipo gerente do que agentes do tipo cliente. Além da necessidade de haver um número considerável de recursos para o atendimento das missões solicitadas pelos gerentes.

Na figura a seguir é apresentado o diagrama de classes do protótipo desenvolvido para validação da arquitetura proposta para coordenação robótica. Em cada uma das classes principais do sistema pode-se verificar a presença dos métodos *void activate(), void live()* e *void end()*. O método *void activate()* é chamado na criação

do agente. Em seguida, o método *void live()* é chamado, representando que o agente está ativo. Quando o agente é destruído ou finalizado o método *void end()* é chamado.

A classe *MyShedule* é uma classe que estende a classe *Scheduler* da plataforma MaDKit. A *Scheduler* estende a classe *Agent* da plataforma e fornece uma infraestrutura para execução de simulações na plataforma. Através desta classe é possível iniciar e parar uma simulação, configurar um tempo de execução para simulação, além da geração de arquivos de saída.

Os agentes dos tipos cliente, gerente e recurso são iniciados com uma interface gráfica. O método *public void setupFrame(JFrame frame)*, presente em cada um dos agentes, permite que essa interface padrão da plataforma seja customizada.

O agente cliente quando iniciado, a partir do método *void activate()*, cria um grupo de Clientes (caso este ainda não tenha sido criado), através do método *createGroupIfAbsent*, e requisita o papel de cliente dentro deste grupo através do método *requestRole*. O atributo *produto* relacionado à missão é definido randomicamente para cada agente cliente criado. Os produtos da missão estão relacionados diretamente com os tipos agentes recursos disponíveis, neste caso: *drone1*, *drone2* e *robot3*. Apesar das simulações ficarem restritas a esses três tipos de recursos, a arquitetura proposta permite uma maior diversidade.

Figura 5-4 Diagrama de Classe do Protótipo Desenvolvido para Validar a Arquitetura Proposta

A seguir é mostrado o método *void live()* do agente cliente, que é chamado quando este estiver ativo. Está destacado em negrito o método *sendMessageWithRoleAndWaitForReply* que envia uma mensagem aos participantes do *CLIENTES_GROUP*, particularmente para os agentes com papel *GERENTE_ROLE*, passando como parâmetros também o conteúdo da mensagem, neste caso o produto requisitado, e papel do remetente da mensagem, o *CLIENTE_ROLE*, e o tempo de espera da mensagem. Os agentes gerentes presentes no sistema irão receber essa

mensagem e tratar internamente. Caso haja resposta por parte de um dos agentes gerentes, os métodos destacados em negrito fora do laço *while* são chamados.

```
protected void live() {
      boolean foiFinalizada = false;
      int tentativasSemRespostas = 0;
      int tentativasComFalhas = 0;
      while (!foiFinalizada) {
         Message respostaGerente = null;
         while (respostaGerente == null) {
            respostaGerente = sendMessageWithRoleAndWaitForReply(
                  MarketOrganization.COMMUNITY,
                  MarketOrganization.CLIENTES_GROUP,
                  MarketOrganization.GERENTE_ROLE,
                  new StringMessage(product),
                  MarketOrganization.CLIENTE_ROLE,
                  1000);
            if (logger != null && respostaGerente == null) {
               logger.info("CLIENTE - Sem resposta de Gerente por enquanto - :(");
               Gerente.noVezQueClienteFicouSemRespostasDeGerentes++;
               tentativasSemRespostas++;
               if (tentativasSemRespostas == 5) {
                  logger.info("CLIENTE - Não fui atendido após 5 tentativas sem resposta :(");
                  end();
               }
            }
            pause(200);
         }
         logFindGerente(respostaGerente);
         statusDaMissao((StringMessage) respostaGerente);
         foiFinalizada = finalizaMissao((StringMessage) respostaGerente);
         pause(500);
         if (foiFinalizada == false) {
            tentativasComFalhas++;
            if (tentativasComFalhas == 5) {
               logger.info("CLIENTE - Não fui atendido após 5 falhas - :(");
               end();
           }  }  }  }
```

O método *logFindGerente(respostaGerente)* do agente cliente imprime no console de saída ou arquivo de saída (*log*) o conteúdo da resposta do agente gerente. A chamada do método *logger*.info, destacado em negrito, imprime na saída o nome do gerente que enviou a resposta, que neste caso está tratando da solicitação feita pelo agente cliente. A interface gráfica do agente cliente é pintada de amarelo através do método *blinkPanel.setBackground(Color.YELLOW)* para indicar que o agente cliente já solicitou um pedido de missão para o sistema. Este pedido está sendo tratado por parte de agente gerente que irá alocar um recurso para executar a missão.

```
private void logFindGerente(Message respostaGerente) {
      if (logger != null) {
        logger.info("CLIENTE – Encontrei um Gerente: " + respostaGerente.getSender());
      }
      if (blinkPanel != null) {
         blinkPanel.setBackground(Color.YELLOW);
         pause(2000);
      }
   }
```

O método *statusDaMissao((StringMessage) respostaGerente)* do agente cliente é chamado para questionar o status ou andamento da missão ao agente recurso selecionado para realizá-la (ver trecho de código mostrado a seguir). Como parâmetro deste método, é passada a mensagem de resposta do agente gerente que tratou esse pedido. A partir desta mensagem, do tipo *StringMessage*, é possível extrair o *contractGroupID*. O *contractGroupID* é o código que identifica unicamente o grupo relacionado com a missão contratada ou mesmo identifica unicamente a missão contratada, sendo utilizada pelo agente cliente (solicitante da missão), pelo agente gerente (gestor da missão) e pelo agente recurso (executor da missão). Para que a mensagem seja enviada ao agente recurso, o agente cliente necessita requisitar seu papel no grupo *contractGroupID*. Esse grupo foi criando pelo agente recurso selecionado para executar a missão. Esse grupo é restrito aos agentes que possuem o código *contractGroupID*. Em seguida, a mensagem *sendMessageAndWaitForReply* é enviada ao agente recurso contratado solicitando uma reposta quanto ao status da missão. Essa mensagem tem um tempo máximo de espera de 4000 ms. Caso haja resposta por parte do agente recurso dentro do tempo estabelecido, uma mensagem é impressa na saída, identificando o recurso que enviou a mensagem, através da chamada *missao.getSender()*, e o percentual de realização da missão, através da chamada *missao.getContent()*.

```
   private void statusDaMissao(StringMessage respostaGerente) {

      String contractGroupID = respostaGerente.getContent();

      requestRole(MarketOrganization.COMMUNITY, contractGroupID,
MarketOrganization.CLIENTE_ROLE);

      StringMessage missao = (StringMessage)
sendMessageAndWaitForReply(MarketOrganization.COMMUNITY, contractGroupID,
MarketOrganization.RECURSO_ROLE, new StringMessage("Status Missao"), 4000);

      if (missao != null) {
         if (logger != null) {
            logger.info("CLIENTE - Missao em Andamento pelo Recurso" +
missao.getSender() + " :) - " + missao.getContent() + " PORCENTO");
         }
         if (hasGUI()) {
            blinkPanel.setBackground(Color.YELLOW);
         }
      }
   }
```

O método *finalizaMissao((StringMessage) respostaGerente)* do agente cliente é chamado para verificar se a missão foi finalizada com sucesso ou não por parte do agente recurso contrato (ver código a seguir).

```
   private boolean finalizaMissao(StringMessage respostaGerente) {

      String contractGroupID = respostaGerente.getContent();
      requestRole(MarketOrganization.COMMUNITY, contractGroupID,
MarketOrganization.CLIENTE_ROLE);
      Message missao =
sendMessageAndWaitForReply(MarketOrganization.COMMUNITY, contractGroupID,
MarketOrganization.RECURSO_ROLE, new StringMessage("Missao foi Finalizada?"),
15000);

      if (missao != null) {
         if ((((StringMessage) missao).getContent() == "Done")) {
            if (logger != null) {
               logger.info("CLIENTE - Sim: Missao FINALIZADA :) ");
            }
            if (hasGUI()) {
               blinkPanel.setBackground(Color.GREEN);
            }
            leaveGroup(MarketOrganization.COMMUNITY,
MarketOrganization.CLIENTES_GROUP);
            pause((int) (1000 + Math.random() * 2000));
            return true;
```

```
 } else if (((StringMessage) missao).getContent() == "Fail") {
            if (logger != null) {
               logger.info("CLIENTE - Missao FALHOU :)  ");
            }
            if (hasGUI()) {
               blinkPanel.setBackground(Color.RED);
            }
            // leaveGroup(MarketOrganization.COMMUNITY,
MarketOrganization.CLIENTES_GROUP);
            pause((int) (1000 + Math.random() * 2000));
            return false;
         }
      }
      return false;
   }
```

Como no método *statusDaMissao*, o agente cliente necessita requisitar o papel *CLIENTE_ROLE* dentro do grupo *contractGroupID*. Em seguida, uma mensagem é enviada ao recurso que foi contratado para realizar a missão, questionando se a missão foi finalizada ou não. O tempo máximo da resposta foi definido em 15000 ms. Caso haja resposta neste tempo, verifica-se o conteúdo desta através da chamada *((StringMessage) missao).getContent()*. A missão pode ter sido finalizada com sucesso caso o conteúdo desta chamada seja *"Done"* ou pode ter falhado, caso o conteúdo da mensagem seja *"Fail"*. Pode-se observar o uso do *blinkPanel.setBackground* para pintar a interface gráfica do agente cliente de cor verde, caso o missão tenha sido finalizada com sucesso, ou de cor vermelha, em caso de falha na execução da missão.

O método *void end()* é chamado para encerrar a execução o agente cliente. Uma mensagem é impressa na saída indicando que o agente está saindo e novos outros agentes do tipo cliente serão iniciados na plataforma.

```
   @Override
   protected void end() {
      if (logger != null) {
         logger.info("CLIENTE - Estou saindo agora, mas vou chamar outro cliente !");
      }
      pause((int) (Math.random() * 2000 + 5000));
      launchAgent(new Cliente(), 2, true);
   }
```

Apesar do agente cliente nesta simulação ter apenas o papel de cliente, ele poderia assumir o papel de recurso na arquitetura proposta. Neste caso ele receberia demanda de outros clientes e poderia ser contratado para realizar missões. Ele teria que

subcontratar outros agentes do tipo recurso para realizada as missões por ele contratadas.

Os agentes do tipo gerente cuidam basicamente da seleção e contratação de agentes recursos para a execução das missões ou tarefas. A seguir é apresentado o método *activate()* do agente gerente. Pode-se observa que o agente gerente participa a princípio de dois grupos *CLIENTES_GROUP* e o *CLIENTES_GROUP*. Assim são requisitados o papel de *GERENTE_ROLE* para estes dois grupos (ver trecho de código a seguir).

```
protected void activate()
        {
        createGroupIfAbsent(MarketOrganization.COMMUNITY,
MarketOrganization.CLIENTES_GROUP,true,null);
        createGroupIfAbsent(MarketOrganization.COMMUNITY,
MarketOrganization.RECURSOS_GROUP,true,null);
        requestRole(MarketOrganization.COMMUNITY,
MarketOrganization.CLIENTES_GROUP, MarketOrganization.GERENTE_ROLE,null);
        requestRole(MarketOrganization.COMMUNITY,
MarketOrganization.RECURSOS_GROUP, MarketOrganization.GERENTE_ROLE,null);
    if(logger != null)
                logger.info("GERENTE - Eu nasci e sou " + Gerente.this);
      }
```

No método *void live()* do agente gerente há o método *purgeMailbox()* que faz com que a ultima mensagem recebida seja tratada e a caixa de entrada seja limpa, apagando mensagens anteriores. Caso não haja mensagem na caixa de entrada a variável *m* será nula e o método *waitNextMessage()* será chamado. Se a mensagem recebida for proveniente de um agente com papel de cliente ou *CLIENTE_ROLE*, o método *handleClienteRequest((StringMessage)* será chamado.

```
        protected void live()
        {
                while (true)
                {
                        Message m = purgeMailbox();
                        if (m == null) {
                        m = waitNextMessage();
                        }
                        String role = m.getSender().getRole();
                        if (role.equals(MarketOrganization.CLIENTE_ROLE)) {
                    noDeMensagenRecebidasDeClientes++;
                                handleClienteRequest((StringMessage) m);
                        }
                }
        }
```

Uma simplificação do método *handleClienteRequest(StringMessage request)* do agente gerente é apresentado a seguir e são destacados em negrito as partes mais importantes para seu entendimento. O agente gerente, a partir da requisição de recursos feita por um agente cliente envia uma chamada de proposta aos recursos com a competência desejada, e isso é feito através da chamada do método *broadcastMessageWithRoleAndWaitForReplies*. A especificação da competência ou produto desejado é feito através do terceiro parâmetro (o parâmetro *String Role*) passado na chamada deste método. Neste caso, a seguinte *String* é passada - *request.getContent() + "-" + MarketOrganization.RECURSO_ROLE*. Isso faz com que apenas os agentes recurso que possuem tal competência recebam essa mensagem. O conteúdo da mensagem enviada é "fazer-proposta-por-favor". Em caso de nenhuma resposta ser recebida, uma mensagem é enviada ao terminal de saída e a tela da interface de saída do agente gerente fica de cor cinza.

No caso do agente gerente receber propostas por parte dos recursos, este irá tratar a lista de ofertas recebidas, selecionando a menor, como a melhor proposta utilizando para isso o método *ObjectMessage.min(offers).* A melhor oferta e o agente recurso selecionado são então impressos no terminal de saída. Em seguida, o *contractGroupId* é criado e enviado para o recurso selecionado através do método *sendMessageWithRoleAndWaitForReply*. Como resposta para essa mensagem é esperada uma confirmação ou um *ack* por parte do agente recurso selecionado num tempo de 1000 ms. Em caso de confirmação por parte do agente recurso, o agente gerente envia uma mensagem de resposta ao agente cliente, já com número do contrato através do método *sendReply(request, new StringMessage(contractGroupId)).* Uma mensagem também é impressa no terminal de saída informando que um recurso foi contratado e o contrato foi enviado ao agente cliente solicitante da missão.

```
   private void handleClienteRequest(StringMessage request) {
      if (logger != null) {
         logger.info("GERENTE - Eu recebi uma requisicao para " + request.getContent() + " do
" + request.getSender());
      }
      List<Message> propostas = broadcastMessageWithRoleAndWaitForReplies(
            MarketOrganization.COMMUNITY,
            MarketOrganization.RECURSOS_GROUP,
            request.getContent() + "-" + MarketOrganization.RECURSO_ROLE,
            new StringMessage("fazer-proposta-por-favor"),
            MarketOrganization.GERENTE_ROLE,   900);
```

```
        noDeContratosNegociadosComRecursos++;

      if (propostas == null) {
            noNegociacoesSemPropostaRecursos++;

             if (logger != null) {
                logger.info("GERENTE - Sem propostas por enquanto: Ninguém está disponível - " +
request.getContent().toUpperCase() + " !! Precisamos de novos recursos !");
            }
            if (hasGUI()) {
                blinkPanel.setBackground(Color.LIGHT_GRAY);
            }
            return;
        }

    // seleciona melhor  oferta

        final List<IntegerMessage> offers = Arrays.asList(propostas.toArray(new
IntegerMessage[0]));
        IntegerMessage best = ObjectMessage.min(offers);
        if (logger != null) {
            logger.info("GERENTE - A melhor oferta veio de" + best.getSender() + " " +
best.getContent());
        }
        String contractGroupId = "" + System.nanoTime();     //cria-se um contrato Id


      // envia o contrato para recurso

        Message ack = sendMessageWithRoleAndWaitForReply(
                best.getSender(),
                new StringMessage(contractGroupId),
                MarketOrganization.GERENTE_ROLE,
                1000);

        if (ack != null) {
            if (logger != null) {
                logger.info("GERENTE - O Recurso foi contratado! Enviando o número do
contrato para o Cliente - o solicitante foi" + request.getSender());
            }
            sendReply(request, new StringMessage(contractGroupId));
            noDeRecursosContratadosComSucesso++;

        } else {
            noNegociacoesSemRespostaRecursosAposSelecionado++;
            if (logger != null) {
                logger.info("GERENTE - Recurso desapareceu, sem repostas !!!!");
            }
        }
        if (hasGUI()) {
            blinkPanel.setBackground(Color.LIGHT_GRAY);
        }

}
```

O tipo de agente recurso é definido randomicamente na sua criação. Nas simulações as opções definidas foram as seguintes: "drone1", "drone2" ou "robot3". Os agentes recurso, para efeito de simulação, são iniciados randomicamente com um índice de sucesso, este valor será usado para definir se agente recurso irá finalizar as missões por ele contratadas.

```
public class Recurso extends Agent {

   public static List<String> availableTransports = Arrays.asList("drone1", "drone2", "robot3");
   private int indiceSucesso = ((int) (Math.random() * 100));
   private String competence;

   public Recurso() {
         competence=Recurso.availableTransports.get((int)(Math.random()*Recurso.availableTr
         ansports.size()));
   }
```

No método *void activate()* do agente recurso tem-se a criação do grupo de recursos, caso ainda não tenha sido criado, e a requisição de um papel para o grupo de recursos de sua competência.

```
public void activate() {

         createGroupIfAbsent(MarketOrganization.COMMUNITY,MarketOrganization.RECUR
SOS_GROUP, true, null);

         requestRole(MarketOrganization.COMMUNITY,MarketOrganization.RECURSOS_GR
         OUP,  competence + "-" + MarketOrganization.RECURSO_ROLE, null);

          if (logger != null) {
                   logger.info("RECURSO - Eu nasci - Minha competência é " + competence + " e
sou " + Recurso.this);
                   logger.info("RECURSO - Valor - Índice de Sucesso " + indiceSucesso);
         }
   }
```

Uma versão simplificada do método *void live()* do agente recurso é mostrada a seguir. O agente fica aguardando uma mensagem através do método *waitNextMessage()*. Caso uma mensagem seja recebida, ela é tratada conforme seu conteúdo e o papel do agente que enviou. Se a mensagem foi enviada por um agente com papel de gerente, o método *handleGerenteMessage((StringMessage) m)* é chamado. No caso do conteúdo da mensagem ser igual a "Status Missao", o método *passaStatus(Sm)* é chamado, e caso o conteúdo da mensagem seja "Quero Aprender", o método chamado será o *passaAprendizado(Sm)*. Por fim, se nenhum dos casos descritos

ocorrer, o método *finalizeContract(Sm)* será chamado. Essa mensagem foi enviada certamente pelo agente cliente que gerou a solicitação de missão (ver trecho de código a seguir).

```
public void live() {

while (true) {

        Message m = waitNextMessage();
         StringMessage Sm = null;
         Sm = (StringMessage) m;

       if ((m.getSender().getRole().equals(MarketOrganization.GERENTE_ROLE))) {
          handleGerenteMessage((StringMessage) m);
       } else if ((Sm = (StringMessage) m).getContent().equals("Status Missao")) {
          passaStatus(Sm);
       } else if ((Sm = (StringMessage) m).getContent().equals("Quero Aprender")) {
          passaAprendizado(Sm);
       } else {
          finalizeContract(Sm);
        }
    }
  }
```

No método *void handleGerenteMessage(StringMessage m),* o agente recurso pode tratar a mensagem enviada pelo agente gerente. A mensagem pode trata-se de chamada por proposta, assim o método *sendReply (m, new IntegerMessage((int) (Math.random() * 500)))* passa como parâmetro um valor randômico como proposta. Outra possibilidade é que a mensagem já seja uma confirmação de seleção por parte do agente gerente, neste caso o método *iHaveBeenSelected(m)* é chamado (ver trecho de código apresentado a seguir).

```
private void handleGerenteMessage(StringMessage m) {

    if (m.getContent().equals("fazer-proposta-por-favor")) {

       if (logger != null) {
          logger.info("RECURSO - Eu recebi uma chamada de proposta de " + m.getSender());
       }

       sendReply(m, new IntegerMessage((int) (Math.random() * 500)));

    } else {
       iHaveBeenSelected(m);
    }
  }
```

O método *void iHaveBeenSelected(m)* manipula a mensagem enviada por parte do agente gerente para o agente recurso confirmando a sua seleção para execução de uma missão. A interface de usuário do agente recurso é pintada de amarelo, indicando que o agente recurso foi selecionado para uma missão. O *contractGroupID* passado como conteúdo da mensagem é coletado. O agente recurso cria o grupo relacionado ao *contractGroupID* e requisita seu papel de recurso ou *RECURSO_ROLE* neste grupo. Por fim, o agente recurso envia uma mensagem confirmando que recebeu o contrato através da mensagem *sendReply(m, new Message()).*

```
  private void iHaveBeenSelected(StringMessage m) {

     if (hasGUI()) {
        blinkPanel.setBackground(Color.YELLOW);
     }

     if (logger != null) {
        logger.info("RECURSO - Eu fui selecionado para missão :)");
     }

     String contractGroup = m.getContent();

     createGroup(MarketOrganization.COMMUNITY, contractGroup, true);
     requestRole(MarketOrganization.COMMUNITY, contractGroup,
MarketOrganization.RECURSO_ROLE);

     if (logger != null) {
        logger.info("RECURSO - Este é o No do Contrato= " + contractGroup);
     }

     sendReply(m, new Message());
  }
```

Através do método *void passaStatus(StringMessage m)* apresentado a seguir, o agente recurso passa o status sobre o andamento da execução da missão ao agente cliente. Como na simulação realizada a camada reativa não foi considerada, o percentual de execução foi gerando randomicamente. Numa situação real o agente recurso deve requisitar do seu robô real o status da missão para em seguida enviá-la ao agente cliente. O status da missão é enviado ao agente cliente através do método *sendReply (m, new StringMessage(String.valueOf(status))).*

```
private void passaStatus(StringMessage m) {

      int status = ((int) (Math.random() * 100));

      if (hasGUI()) {
         blinkPanel.setBackground(Color.YELLOW);
      }

      if (logger != null) {
         logger.info("RECURSO - Eu - " + Recurso.this + " competência -  " + competence + "
realizei " + status + " PORCENTO - STATUS - Em Andamento");
      }

      sendReply(m, new StringMessage(String.valueOf(status)));
      pause((int) (Math.random() * 2000 + 1000));
   }
```

O método *void finalizeContract(StringMessage m)* realiza finalização do contrato. Uma variável chamada de índice de sucesso é avaliada e considerada neste ponto do programa apenas por se tratar de uma simulação. Assim, agentes do tipo recurso com índice de sucesso acima de 20 (ou 40 em alguns simulações) sempre finalizam suas missões com sucesso, enquanto agentes recurso com índice de sucesso menores ou iguais a 20 falham. Em caso de sucesso, a interface de usuário do agente recurso é pintada de verde e uma mensagem confirmando a realização da missão é enviada para o agente cliente. Em seguida, o agente recurso deixa o grupo identificado pelo *contractGroupID.*

Ainda no método *void finalizeContract(StringMessage m),* se o índice de sucesso for menor ou igual a 20, o agente recurso irá falhar na execução de sua missão e sua tela gráfica será pintada de vermelho. Como o agente recurso se comprometeu na execução da missão, ele ficará responsável por buscar novo agente recurso para executar a missão, neste caso será realizada uma subcontratação de outro agente recurso, para isso o método *subcontratacao()* é chamado. Se a subcontratação foi realizada com sucesso, a mensagem *sendReply(m, new StringMessage("Done"))* é enviada ao agente cliente informando que a missão foi realizada com sucesso e o painel do agente recurso é pintado de verde. Caso contrário uma mensagem informando a falha é enviada ao agente cliente que contratou a missão e o painel do agente tipo recurso é pintado de vermelho.

A subcontratação dentro da arquitetura proposta é uma das inovações desta pesquisa. Como há a comunicação e a negociação entre agentes recursos neste processo

para busca de um reparo do plano estabelecido inicial, pode-se dizer que os agentes recursos trabalham cooperativamente para atender toda demanda de solicitações de clientes. Assim, os agentes do tipo recurso buscam conjuntamente a satisfação de um objetivo global comum.

```
private void finalizeContract(StringMessage m) {

      boolean subContractFinalizou = false;
      if (indiceSucesso > 20) { //Se indiceSucesso >20 nao ha falha
         if (hasGUI()) {
            blinkPanel.setBackground(Color.GREEN);
         }
         if (logger != null) {
            logger.info("RECURSO - Eu - " + Recurso.this + " competência -  " + competence + "
realizei a missao ! - sendReply para " + m.getSender());
         }

         sendReply(m, new StringMessage("Done"));
         pause((int) (Math.random() * 2000 + 1000));

         leaveGroup(MarketOrganization.COMMUNITY, m.getSender().getGroup());

         if (hasGUI()) {
            blinkPanel.setBackground(Color.LIGHT_GRAY);
         }

      } else {
         if (hasGUI()) {
            blinkPanel.setBackground(Color.RED);
         }
         subContractFinalizou = subcontratacao();
         if (subContractFinalizou) {
            sendReply(m, new StringMessage("Done"));
            pause((int) (Math.random() * 2000 + 1000));
            leaveGroup(MarketOrganization.COMMUNITY, m.getSender().getGroup());
            if (hasGUI()) {
               blinkPanel.setBackground(Color.GREEN);
            }
         } else {
            if (logger != null) {
               logger.info("RECURSO - Eu - " + Recurso.this + " competência -  " + competence
+ " FALHEI Na missao ! - sendReply para " + m.getSender());
            }
            sendReply(m, new StringMessage("Fail"));
            pause((int) (Math.random() * 2000 + 1000));
            leaveGroup(MarketOrganization.COMMUNITY, m.getSender().getGroup());
            if (hasGUI()) {
               blinkPanel.setBackground(Color.RED);
            }
        }
      }
   }
```

O método *boolean subcontratacao()* apresentado de forma resumida a seguir. O agente recurso agora fará o papel de cliente: requisitando o papel *CLIENTE_ROLE* no grupo de clientes; enviando solicitação de missão para agentes tipo gerente e aguardando resposta com a confirmação da contratação; chamando o método *finalizaMissao((StringMessage) respostaGerente)* para ser informado sobre a finalização com sucesso ou não da missão por parte o agente recurso contratado; e retornando ao papel de recurso, caso a sua missão como cliente tenha sido finalizada com sucesso pelo agente recurso subcontratado.

```
 private boolean subcontratacao() {

     createGroupIfAbsent(MarketOrganization.COMMUNITY,
MarketOrganization.CLIENTES_GROUP, true, null);

     requestRole(MarketOrganization.COMMUNITY,
MarketOrganization.CLIENTES_GROUP, MarketOrganization.CLIENTE_ROLE, null);

boolean missaoFinalizada = false;
     int tentativasSemRespostas = 0;
     int tentativasComFalhas = 0;

     while (!missaoFinalizada) {

        Message respostaGerente = null;

        while (respostaGerente == null) {

           respostaGerente = sendMessageWithRoleAndWaitForReply(
                 MarketOrganization.COMMUNITY,
                 MarketOrganization.CLIENTES_GROUP,
                 MarketOrganization.GERENTE_ROLE,
                 new StringMessage(competence),
                 MarketOrganization.CLIENTE_ROLE, 1000);

           if (logger != null && respostaGerente == null) {

              logger.info("RECURSO - CLIENTE - Sem resposta de gerente por enquanto - :(");
              tentativasSemRespostas++;

              if (tentativasSemRespostas == 5) {

                 logger.info("RECURSO - CLIENTE – Não fui atendido após 5 tentativas - :(");
                 return false;
              }
           }
        }

        logFindGerente(respostaGerente);
```

```
        missaoFinalizada = finalizaMissao((StringMessage) respostaGerente);
        if (missaoFinalizada) {
           createGroupIfAbsent(MarketOrganization.COMMUNITY,
MarketOrganization.RECURSOS_GROUP, true, null);
           requestRole(MarketOrganization.COMMUNITY,
MarketOrganization.RECURSOS_GROUP, competence + "-" +
MarketOrganization.RECURSO_ROLE, null);
           return true;
        } else {
           tentativasComFalhas++;
           if (tentativasComFalhas == 5) {
              logger.info("RECURSO - CLIENTE – Não fui atendido após 5 falhas - :(");
              return false;
           }
        }
     }
     return false;
  }
```

O método *boolean finalizaMissao (StringMessage respostaGerente)* do agente recurso, mostrado a seguir numa versão simplificada, é chamado a partir do processo de subcontratação, visto que agora o agente do tipo recurso está assumindo um papel de cliente. Pode-se verificar que o envio de uma mensagem ao agente recurso subcontratado questionando-o a respeito da finalização da missão.

Caso a missão tenha sido finalizada com sucesso, pressupõe-se que tenha havido um compartilhamento de informação entre os agentes recursos envolvidos, através de qualquer meio, até mesmo troca de mensagens. O agente recurso subcontratado pode agir como tutor do agente recurso contratante, visto que ele falhou em uma missão, que foi realizada com sucesso pelo recurso subcontratado.

Com a finalidade de simular um aprendizado, o índice de sucesso do agente recurso contratante que falhou, no entanto aprendeu a partir do compartilhamento de informação, é acrescido de 10. O processo de aprendizado distribuído, mesmo apenas simulado neste contexto, representa uma inovação, pois permite que os agentes distribuídos possam evoluir num processo controlado de troca de mensagens. Protocolos específicos podem ser estabelecidos para que essas duas entidades envolvidas possam realmente compartilhar informações e um aprendizado efetivo ocorra.

O painel do agende recurso que está atuando como cliente é pintado de cor verde, representando o sucesso da missão. Caso haja falha, o painel de interface do agente recurso contratante é pintado de cor vermelha.

```
private boolean finalizaMissao(StringMessage respostaGerente) {

   String contractGroupID = respostaGerente.getContent();
   requestRole(MarketOrganization.COMMUNITY, contractGroupID,
MarketOrganization.CLIENTE_ROLE);

   Message missao =
sendMessageAndWaitForReply(MarketOrganization.COMMUNITY, contractGroupID,
MarketOrganization.RECURSO_ROLE, new StringMessage("Missao foi Finalizada? "),
4000);

  if (missao != null) {

        if (((StringMessage) missao).getContent().equals("Done")) {
           indiceSucesso = indiceSucesso + 10;
           if (logger != null) {
              logger.info("RECURSO - CLIENTE - Sim: Missao FINALIZADA :) com
COMPARTILHAMENTO DE APRENDIZADO - indiceSucesso = " + indiceSucesso);
           }

           if (hasGUI()) {
              blinkPanel.setBackground(Color.GREEN);
           }

           leaveGroup(MarketOrganization.COMMUNITY,
MarketOrganization.CLIENTES_GROUP);
           pause((int) (1000 + Math.random() * 2000));
           return true;

        } else if (((StringMessage) missao).getContent().equals("Fail")) {

           if (logger != null) {
              logger.info("RECURSO - CLIENTE - Missao FALHOU :)  ");
           }

           if (hasGUI()) {
              blinkPanel.setBackground(Color.RED);
           }

           leaveGroup(MarketOrganization.COMMUNITY,
MarketOrganization.CLIENTES_GROUP);
           pause((int) (1000 + Math.random() * 2000));
           return false;
        }
     }
     return false;
  }
```

5.3.2 Contract Net usando Plataforma JASON

Neste tópico são apresentados detalhes do emprego do protocolo Contract Net no ambiente JASON utilizando a infraestrutura multi-agente JADE.

A aplicação define um agente contratante ou gerente (c) e agentes participantes ou agentes recursos. O contratante oferece uma tarefa aos participantes que lançam suas propostas. O participante "pr" recursa a proposta e o participante "pn" apenas não responde. Os demais participantes "p" participam da negociação ativamente, no entanto, apenas um é contratado.

A estrutura a seguir apresentada é utilizada para iniciar os agentes da aplicação. Nota-se que a infraestrutura JADE é definida.

```
MAS cnp {
   infrastructure: Jade
   agents:
      c;    // O agente que inicia o protocolo Contract Net
      p #3; // Os participantes (3) que oferecem os serviços
      pr;   // O participante que sempre recursa os serviços
      pn;   // O participante nunca responde às solicitações de serviços
}
```

A seguir é apresentado o código do agente contratante ou gerente. Os comentários estão destacados em negrito.

```
/* Crenças e regras iniciais */
all_proposals_received(CNPId)
  :- .count(introduction(participant,_),NP) & // contagem do número de participantes
     .count(propose(CNPId,_), NO) &   // contagem do número de propostas recebidas
     .count(refuse(CNPId), NR) &      // contagem do número de propostas recusadas
     NP = NO + NR.

/* Objetivos Iniciais */
!startCNP(1,fix(computer)).
```

```
/* Planos */
// Início do CNP a tarefa “fix computer” é passada para função
+!startCNP(Id,Task)
   <- .print("Waiting participants...");
      .wait(2000);  // espera a introdução de participantes
      +cnp_state(Id,propose);   // relembra o estado do CNP
      .findall(Name,introduction(participant,Name),LP);
      .print("Sending CFP to ",LP);
      .send(LP,tell,cfp(Id,Task)); // envia a proposta para lista de participantes
      // o tempo de término do CNP é configurado para 4 segundos
      // em seguida o evento  +!contract(Id) é gerado
      .at("now +4 seconds", { +!contract(Id) }).

// recebe as propostas
// se todas as propostas foram recebidas, não aguarda o término do tempo de
// propostas
@r1 +propose(CNPId,_Offer)
   :  cnp_state(CNPId,propose) & all_proposals_received(CNPId)
   <- !contract(CNPId).
// recebe as recusas
@r2 +refuse(CNPId)
   :  cnp_state(CNPId,propose) & all_proposals_received(CNPId)
   <- !contract(CNPId).
// início da fase de contratação
@lc1[atomic]
+!contract(CNPId)
   :  cnp_state(CNPId,propose)
   <- -+cnp_state(CNPId,contract);
      .findall(offer(O,A),propose(CNPId,O)[source(A)],L);
      .print("Offers are ",L);
      L \== []; // limita o plano de execução a apenas uma oferta
      .min(L,offer(WOf,WAg)); // ordena as ofertas a partir da melhor
      .print("Winner is ",WAg," with ",WOf); // anuncia o ganhador
      !announce_result(CNPId,L,WAg);
      -+cnp_state(CNPId,finished).
```

```
// não faz nada, a fase corrente não é proposta
@lc2 +!contract(_).
-!contract(CNPId)
  <- .print("CNP ",CNPId," has failed!").

+!announce_result(_,[],_).

// anuncia o ganhador
+!announce_result(CNPId,[offer(_,WAg)|T],WAg)
  <- .send(WAg,tell,accept_proposal(CNPId));
     !announce_result(CNPId,T,WAg).

// anuncia aos outros o ganhador rejeitando as propostas
+!announce_result(CNPId,[offer(_,LAg)|T],WAg)
  <- .send(LAg,tell,reject_proposal(CNPId));
     !announce_result(CNPId,T,WAg).
```

A seguir é apresentado o código do agente participante que envia propostas ao agente contratante em resposta as tarefas anunciadas. Os comentários estão destacados em negrito.

```
// retorna o preço para o produto de uma função randômica
// o valor randômico entre 100 e 110.
price(_Service,X) :- .random(R) & X = (10*R)+100.
plays(initiator,c).
/* Plano */
// envia uma mensagem a plataforma se apresentando como participante
+plays(initiator,In)
  :  .my_name(Me)
  <- .send(In,tell,introduction(participant,Me)).

// responde a uma chamada por proposta (Call For Proposal)
@c1 +cfp(CNPId,Task)[source(A)]
  :  plays(initiator,A) & price(Task,Offer)
  <- +proposal(CNPId,Task,Offer);
     .send(A,tell,propose(CNPId,Offer)). // envia a proposta
```

```
@r1 +accept_proposal(CNPId)
  :  proposal(CNPId,Task,Offer)
  <- .print("My proposal '",Offer,"' won CNP ",CNPId,
          " for ",Task,"!").
    // recebe  notificação de que ganhou a proposta

@r2 +reject_proposal(CNPId)
  <- .print("I lost CNP ",CNPId, ".");
    -proposal(CNPId,_,_). // tratamento da rejeição de proposta
```

A seguir é apresentado o código do agente participante que não responde as propostas realizadas pelo agente contratante. Os comentários estão destacados em negrito.

```
// envia uma mensagem a plataforma se apresentando como participante
plays(initiator,c).
+plays(initiator,In)
  :  .my_name(Me)
  <- .send(In,tell,introduction(participant,Me)).
```

O código do agente participante é apresentado a seguir. Este sempre recusa as propostas realizadas pelo agente contratante. Os comentários estão destacados em negrito.

```
plays(initiator,c).
+plays(initiator,In)
  :  .my_name(Me)
  <- .send(In,tell,introduction(participant,Me)).

// envia a resposta de recusa da proposta realizada
+cfp(CNPId,_Service)[source(A)]
  :   plays(initiator,A)
  <- .send(A,tell,refuse(CNPId)).
```

A figura a seguir ilustra o funcionamento da plataforma JADE, com o Main-Container, e os agentes inicializados. Todos os agentes, para efeitos de simulação, são executados na mesma máquina. Essa visualização é provida pelo agente RMA(*Remote Agent Management*) da plataforma JADE. Pode-se observar que foram criados três agentes participantes ativos (p1, p2, p2), um agente que sempre rejeita as propostas (pr), um agente que não responde as propostas (pn) e um agente contratante (c). Os demais agentes são próprios da plataforma JADE.

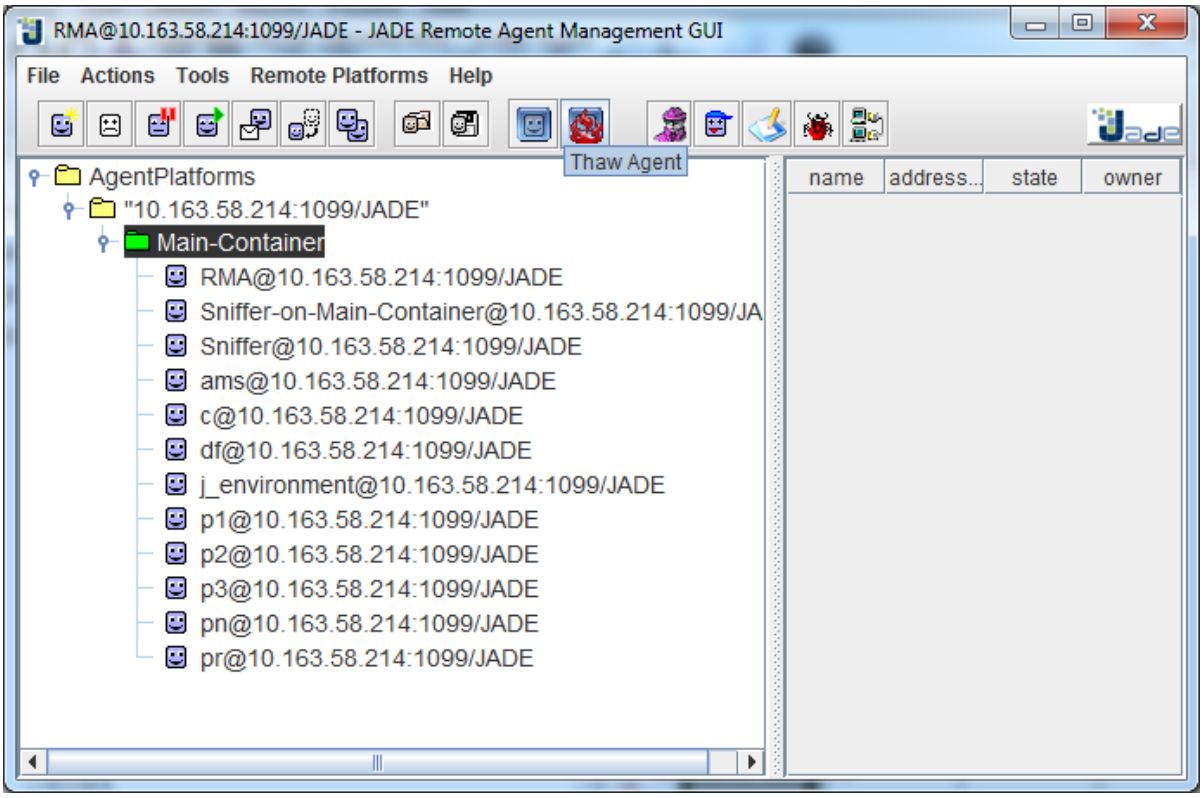

Figura 5-5 Visualização dos agentes presentes na aplicação com seus respectivos endereços através do agente RMA (Remote Agent Management) da plataforma JADE.

Com a inicialização do agente Sniffer da plataforma JADE é possível visualizar a interação, ou troca de mensagens entre os agentes durante a negociação das tarefas. A figura apresentada a seguir ilustra essa troca de mensagens entre os agentes participantes da negociação.

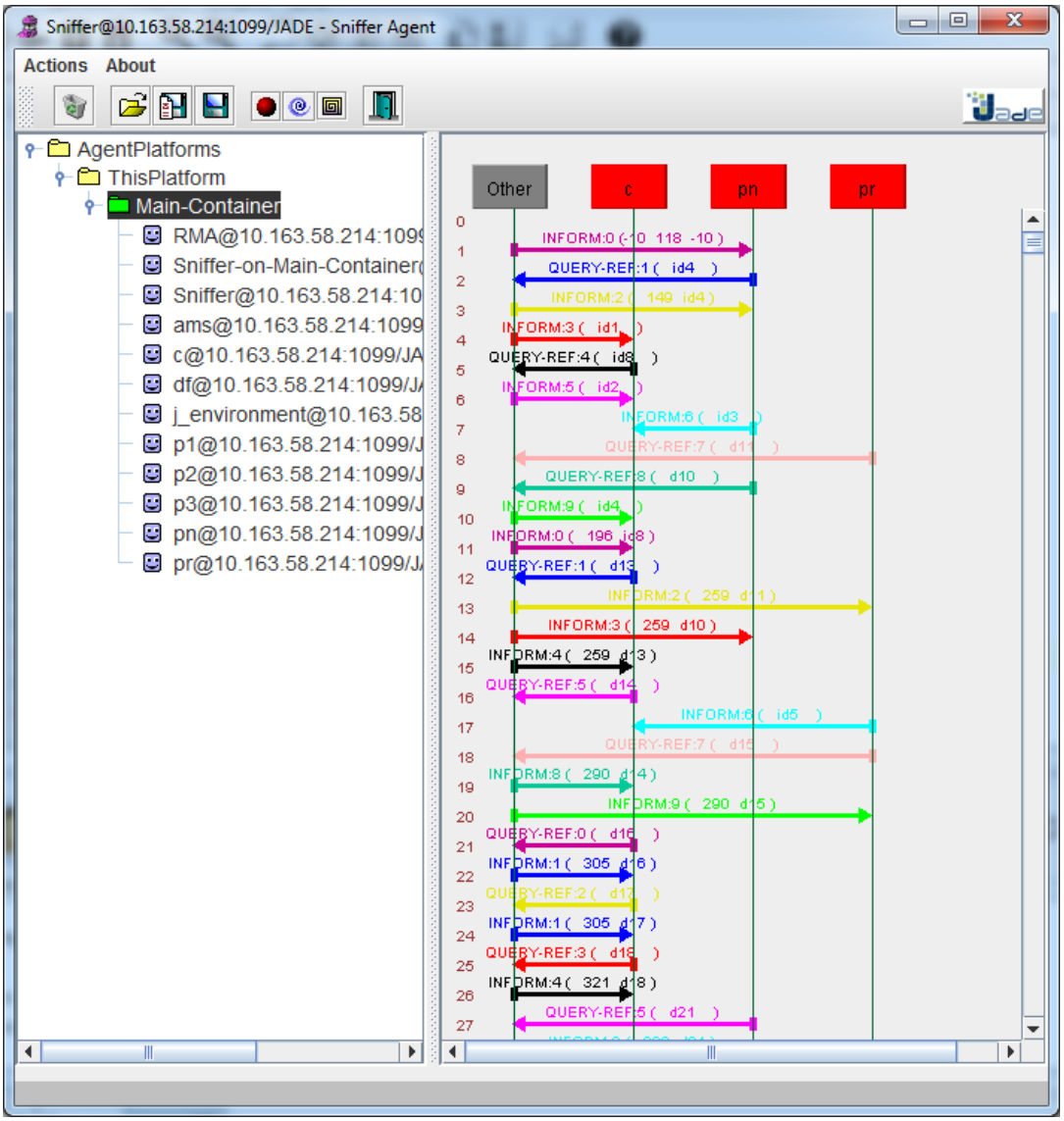

Figura 5-6 Visualização da interação entre os agentes pelo agente Sniffer (da plataforma JADE).

A plataforma JASON fornece um console para acompanhamento da execução da aplicação, conforme apresentado na figura a seguir.

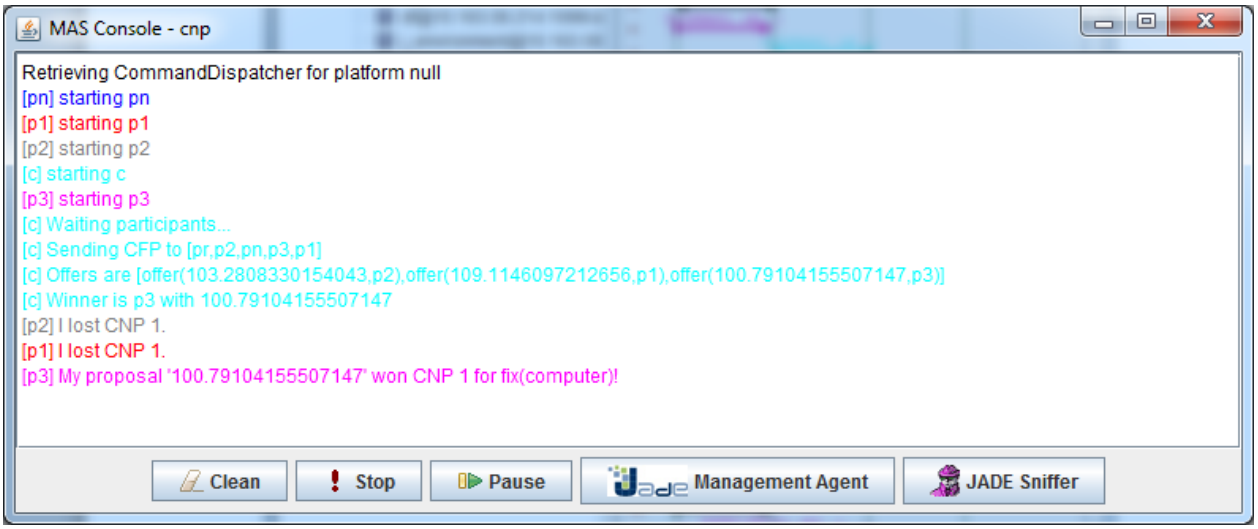

Figura 5-7 Console para visualização da evolução da interação entre os agentes do ambiente JASON.

5.3.3 Protótipo da Arquitetura Proposta usando JADE

Um segundo protótipo foi desenvolvido usando a plataforma JADE, com o objetivo de avaliar eu emprego como uma opção para o desenvolvimento da arquitetura proposta, considerando diversas possibilidades de aplicações.

Neste protótipo há quatro tipos de agentes: Agente Cliente, que cria a demanda de encomenda; Agente Gerente que tem a função de se comunicar com agentes clientes e criar uma instância de um Agente Gestor de Encomenda para cada encomenda solicitada e suas subencomendas; Agente Gestor de Encomenda, que irá tratar as encomendas e contratar Agentes Recurso para executá-las; e Agentes Recurso que irão se candidatar para executar as solicitações e quando selecionados processarão os pedidos. O protocolo *Contract Net* é utilizado para as contrações das encomendas.

No trecho de código a seguir o agente cliente recebe um produto e sua quantidade do usuário e envia o Agente Gerente, para depois receber uma confirmação de alocação.

```
System.out.print("CLIENTE - Produto: ");
String produto = br.readLine();
System.out.print("CLIENTE - Quantidade: ");
int qtd = Integer.parseInt(br.readLine());
msgXML = parser.criaXMLEncomenda(myAgent.getLocalName(),"", produto, qtd);
enviaEncomenda("Gerente", msgXML);
```

O Agente Gerente recebe a mensagem envia pelo Agente Cliente e cria um Agente Gestor de Encomendas para cada encomenda recebida, conforme trecho de código a seguir.

```
dados[0] = enc.criaXMLEncomenda();
ac = cc.createNewAgent(GE, "agentes.GestorEncomendas", dados);
ac.start();
```

O Agente Gestor de Encomendas criado recebe a encomenda que lhe foi passada na sua criação pelo agente gerente, busca os agentes recurso da plataforma, processa o pedido de encomenda, escolhe os melhores recursos para tarefa, e avisa ao gerente a confirmação da encomenda como alocada.

```
public void setup() {
    enc = new Encomenda((String) myAgent.getArguments()[0]);
    recursos = searchDF("Recurso");
    pedidos(enc.getProduto(), Integer.toString(enc.getQuantidade()));
    myAgent.addBehaviour(sequencia);
    recebeMsg();
    }
```

O agente recurso, conforme trecho de código a seguir, recebe mensagem do agente gestor de encomenda solicitando uma proposta, envia uma mensagem de resposta com uma proposta, neste caso seu tempo mínimo para atender a tarefa. Pode-se verificar que o agente recurso pode receber mensagem para alocação em definitivo de sua tarefa ou para desalocar uma tarefa.

```
public void action() {
   ACLMessage msg = myAgent.receive();
   if (msg != null) {
      if (msg.getOntology().equals("msgPede")) {
         Encomenda enc = new Encomenda(msg.getContent(), 1);
         // tempo que leva para processar o produto
         int tempo = getTempo(enc.getProduto());
         int tempo_fim = alocarTemp(tempo, msg.getSender().getLocalName(),
         enc.getIdEncomenda(), enc.getProduto(),
          enc.getTempoMinimoInicio());
          String m = Integer.toString(tempo_fim);
          enviaResposta(msg, m);
    } else {
         if (msg.getOntology().equals("msgAloca")) {
            alocaDefinitivo(Integer.parseInt(msg.getContent()));

         } else {
            if (msg.getOntology().equals("msgDesaloca")) {
               desalocaEnc(msg.getContent(),
                     msg.getReplyWith());
            }
         }
    }
  }
```

Capítulo 6 – Simulações e Discussão dos Resultados

O objetivo deste capítulo é apresentar as simulações realizadas nos protótipos desenvolvidos e discutir sobre os resultados encontrados. Como mencionado anteriormente, a prototipagem da arquitetura multi-agente para coordenação robótica proposta foi a forma escolhida para sua validação. Considerando que o foco dado à pesquisa foi o desenvolvimento de uma solução para camada deliberativa de um sistema multi-robótico, os protótipos desenvolvidos ficaram restritos a essa camada. O comportamento reativo individualizado de cada robô não foi alvo das simulações.

As simulações foram divididas conforme o ambiente e plataforma utilizada. Assim, foram realizadas simulações utilizando as plataformas MaDKit, Jason e JADE.

6.1 Simulações e Resultados do Protótipo da Arquitetura usando Plataforma MaDKit

Neste tópico são apresentadas as simulações e os resultados do protótipo desenvolvido, usando a plataforma MaDKit, para validar a arquitetura multi-agente para coordenação robótica proposta.

Como apresentado no capítulo anterior, para iniciar os agentes é usado o seguinte trecho de código. Assim, as simulações foram realizadas seguindo essa configuração: dois agentes do tipo cliente são iniciados para gerar seus pedidos e são em seguida finalizados e novos outros dois agentes são criados; três agentes do tipo gerente são criados com a finalidade de processar os pedidos dos agentes cliente; e vinte agentes recurso são criados para executar as missões criadas pelos agentes do tipo cliente. Essa configuração foi a utilizada em praticamente todas as simulações para permitir a comparação de resultados.

```
public static void main(String[] args) {
    Gerente.nbDeClientesCriados = 0;
    Gerente.nbDeGerenteCriados = 0;
    new Madkit(Option.launchAgents.toString(),
        MySchedule.class.getName()+",true;"
```

```
            + Gerente.class.getName() + ",true,3;"
            + Cliente.class.getName() + ",true,2;"
            + Recurso.class.getName() + ",true,20;");
           // + ConsoleAgent.class.getName()
          };
    }
```

Os agentes do tipo recurso iniciados eram, basicamente, de três tipos: "drone1", "drone2" e "robot3". Outros tipos podem ser criados na arquitetura deixando a rede mais heterogênea. No entanto, para dar equilíbrio e estabilidade maior a simulação definiu-se limitar esse número em três. Caso necessitássemos aumentar o número de tipos seria indicado também aumentar o número de agentes do tipo recurso, com o objetivo de buscar o atendimento mais eficiente das missões, diminuindo assim o número de falhas.

Foram realizadas simulações considerando três grupos de cenários, com o objetivo compreender melhor o comportamento da solução e permitir a comparação com uma aplicação desenvolvida usando o protocolo *Contract Net* na sua versão pura, ou seja, sem modificações. Assim, os conjuntos de simulações foram os seguintes:

- Simulações utilizando uma aplicação desenvolvida com o *contrac net* sem modificações, versão chamada de CNP pura do protocolo.
- Simulações utilizando uma versão da aplicação que permite a subcontratação de outros agentes do tipo recurso em caso de falha. Essa subcontratação é realizada pelo agente recurso que falhou em sua missão, neste caso, assumindo temporariamente o papel de agente cliente.
- Simulações utilizando uma versão da aplicação que permite além da subcontratação, uma espécie de aprendizagem caso o agente subcontratado realize a missão, simulando assim um compartilhamento de informações com o agente que falhou. Neste caso, o índice de sucesso em missões do agente que falhou, mas absorveu conhecimento é incrementado de dez pontos. Desta forma, espera-se que o número de falhas nas missões diminua com o tempo, considerando uma aprendizagem distribuída.

Um aspecto que vale a pena um destaque é sobre o chamado índice de sucesso dos agentes do tipo recurso. Cada agente recurso é iniciado com um índice de sucesso,

que lhe é atribuído randomicamente pelo método apresentado a seguir. Como são vinte agentes do tipo recurso, cada um terá um índice de sucesso em missões.

```
private int indiceSucesso = ((int) (Math.random() * 100));
```

Simulações foram realizadas considerando que os agentes recurso com índice de sucesso iguais ou menores a vinte sempre falhavam em suas missões contratadas. Foram realizadas também várias simulações considerando um cenário um pouco mais crítico, em que os agentes do tipo recurso com índice de sucesso iguais ou menores que quarenta sempre falhavam em suas missões. Desta forma, uma vez que essas atribuições são randômicas, em algumas simulações havia um cenário em que vários agentes tinham índices de sucesso menores do que os limiares definidos, isso aumentava bastando o número de falhas. Em algumas simulações o número de agentes que falhavam, devido ao índice de sucesso baixo que lhe era atribuído, era menor, configurando assim um cenário melhor, com menos falhas nas missões.

As simulações foram realizadas em apenas uma máquina para permitir a comparação dos resultados. Assim, todos os agentes foram criados e trocavam mensagem nessa máquina. A configuração da máquina empregada nas simulações foi a seguinte: Processador Intel Pentium Dual CPU T2310 – 1.46GHz, memória RAM instalada de 2GB, sistema operacional Windows 7 de 32 bits. Os códigos foram desenvolvidos utilizando a biblioteca, a documentação e os exemplos da plataforma MaDKit. Como ambiente de desenvolvimento foi utilizado NetBeans IDE 8.0.2 (Build 201411181905) com compilador Java 1.7.0_01.

O agente gerente foi utilizado para coletar informações sobre o andamento das contratações. Variáveis foram distribuídas em algumas partes do código a fim de extrair as estatísticas das simulações. Foram definidas as seguintes variáveis para coletar informações do sistema:

- *noDeClientesCriados* para acompanhar o número de agentes clientes criados nas simulações;
- *noDeContratosFinalizadosComSucesso* para contagem do número de contratos realizados com sucesso ou missão concluída pelos agentes recurso;
- *noDeContratosFinalizadosSemSucesso* para contagem do número de contratos realizados sem sucesso ou número de missões que falharam realizadas pelos agentes recurso;

- *noVezQueClienteFicouSemRespostasDeGerentesSaiuSemAtendimento* para contagem do número de vezes em que agentes clientes não conseguiram sequer respostas de agentes gerente após um número estabelecido de tentativas;
- *noDeContratosNegociadosRecursos* para contagem do número de negociações realizadas, ou seja, chamadas de propostas (*call for proposal - cfp*) feitas pelos agentes gerente (através de mensagens do tipo *broadcast*);
- *noDeRecursosContratadosComSucesso* para contabilizar o número de vezes em que os agentes recurso foram contratados com sucesso, ou seja, foram selecionados para uma missão;
- *noDeMensagensTrocasComRecursos* para contagem do número de mensagens recebidas pelos agentes recursos, que podem ser por vários motivos, como por exemplo, solicitação de contrato feita pelo gerente, confirmação de que o agente recurso foi selecionado pelo agente gerente, solicitação de finalização do contrato feita pelo agente cliente;
- *noDeMensagensRecebidasDeClientes* para contagem do número de mensagens que os agentes recursos receberam de agentes clientes;
- *noNegociacoesSemPropostaRecursos* para contabilizar as vezes em que negociações foram realizadas, mas não houveram propostas feitas por agentes do tipo recurso;
- *noNegociacoesSemRespostaRecursosAposSelecionado* para contabilizar as vezes em que negociações foram realizadas, e um agente recurso foi selecionado, mas em seguida não houveram respostas por parte do agente recurso selecionado;
- *noVezQueClienteFicouSemRespostasDeGerentes* para contabilizar o número de vezes em que os clientes firam sem respostas de gerentes;
- *noDeSubContratosFinalizadosComSucesso* para contagem do número de subcontratações finalizadas com sucesso;
- *noDeSubContratosFinalizadosSemSucesso* para contagem do número de subcontratações finalizadas sem sucesso ou falharam.

A partir de cada simulação realizada foi gerado um arquivo com o resultado das contagens descritas anteriormente. Foi possível extrair deste arquivo informações a respeito do comportamento do sistema ao longo do tempo. As simulações tiveram um tempo máximo estabelecido de 20 minutos.

Um exemplo do arquivo de saída gerado é mostrado a seguir, registrando a inicialização de sistema. Verifica-se que os agentes do sistema são logo iniciados. Pode-se observar que cada agente recurso possui um índice de sucesso e uma competência (ou produto relacionado). Os agentes do tipo cliente também são iniciados com uma necessidade por produto (ou missão).

```
        ----------------------------------------
                MaDKit
            version: 5.0.5.2
          build-id: 20150430-2053
         MaDKit Team (c) 1997-2015
        ----------------------------------------
[Gerente-3] Informações : GERENTE - Eu nasci e sou Gerente-3 (ACTIVATED)
[Gerente-4] Informações : GERENTE - Eu nasci e sou Gerente-4 (ACTIVATED)
[Gerente-8] Informações : GERENTE - Eu nasci e sou Gerente-8 (ACTIVATED)
[Recurso-9] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-9 (ACTIVATED)
[Recurso-9] Informações : RECURSO - Valor - Indice de Sucesso 52
[Recurso-11] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-11 (ACTIVATED)
[Recurso-11] Informações : RECURSO - Valor - Indice de Sucesso 32
[Recurso-10] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-10 (ACTIVATED)
[Recurso-10] Informações : RECURSO - Valor - Indice de Sucesso 95
[Cliente-7] Informações : CLIENTE - Eu nasci e sou Cliente-7 (ACTIVATED)
[Cliente-7] Informações : CLIENTE - Necessito de um drone em robot3 in 1640 ms !
[Recurso-26] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-26 (ACTIVATED)
[Recurso-26] Informações : RECURSO - Valor - Indice de Sucesso 51
[Recurso-22] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-22 (ACTIVATED)
[Recurso-22] Informações : RECURSO - Valor - Indice de Sucesso 25
[Cliente-5] Informações : CLIENTE - Eu nasci e sou Cliente-5 (ACTIVATED)
[Cliente-5] Informações : CLIENTE - Necessito de um drone em drone2 in 1695 ms !
[Recurso-25] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-25 (ACTIVATED)
[Recurso-25] Informações : RECURSO - Valor - Indice de Sucesso 42
[Recurso-23] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-23 (ACTIVATED)
[Recurso-23] Informações : RECURSO - Valor - Indice de Sucesso 94
[Recurso-17] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-17 (ACTIVATED)
[Recurso-17] Informações : RECURSO - Valor - Indice de Sucesso 21
[Recurso-15] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-15 (ACTIVATED)
[Recurso-15] Informações : RECURSO - Valor - Indice de Sucesso 66
[Recurso-13] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-13 (ACTIVATED)
[Recurso-13] Informações : RECURSO - Valor - Indice de Sucesso 99
[Recurso-20] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-20 (ACTIVATED)
[Recurso-20] Informações : RECURSO - Valor - Indice de Sucesso 78
[Recurso-19] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-19 (ACTIVATED)
[Recurso-19] Informações : RECURSO - Valor - Indice de Sucesso 45
[Recurso-21] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-21 (ACTIVATED)
[Recurso-21] Informações : RECURSO - Valor - Indice de Sucesso 63
[Recurso-18] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-18 (ACTIVATED)
[Recurso-18] Informações : RECURSO - Valor - Indice de Sucesso 79
[Recurso-16] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-16 (ACTIVATED)
[Recurso-16] Informações : RECURSO - Valor - Indice de Sucesso 33
[Recurso-14] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-14 (ACTIVATED)
[Recurso-14] Informações : RECURSO - Valor - Indice de Sucesso 47
[Recurso-12] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-12 (ACTIVATED)
[Recurso-12] Informações : RECURSO - Valor - Indice de Sucesso 14
[Recurso-28] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-28 (ACTIVATED)
[Recurso-28] Informações : RECURSO - Valor - Indice de Sucesso 61
[Recurso-27] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-27 (ACTIVATED)
[Recurso-27] Informações : RECURSO - Valor - Indice de Sucesso 38
[Recurso-24] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-24 (ACTIVATED)
[Recurso-24] Informações : RECURSO - Valor - Indice de Sucesso 0
```

Figura 6-1 Arquivo de saída (*log*) gerado registrando a inicialização de sistema.

Na figura apresentada pode-se observar a inicialização dos agentes gerente 3, 4 e 9. O agente recurso 9 é um tipo "drone1" (sua competência) e tem índice de sucesso de 52 em suas missões. O agente recurso 11 iniciado em seguida é do tipo "drone1" e tem índice de sucesso de 32. O agente recurso 10 é do tipo "drone1" e tem índice de sucesso de 95. Pode-se também observar a inicialização do agente cliente 7 com necessidade de robot3.

Pode-se verificar que foram iniciados vinte agentes recurso. Lembrando que nessa simulação os agentes recurso com índice de sucesso menores ou iguais a quarenta irão sempre falhar. Dos vintes agentes recurso, sete irão sempre falhar, dos quais três são do tipo "drone1", três do "robot3" e um do tipo "drone2" (ver lista a seguir).

[Recurso-11] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-11
[Recurso-11] Informações : RECURSO - Valor - Indice de Sucesso 32
[Recurso-27] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-27
[Recurso-27] Informações : RECURSO - Valor - Indice de Sucesso 38
[Recurso-16] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-16
[Recurso-16] Informações : RECURSO - Valor - Indice de Sucesso 33

[Recurso-22] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-22
[Recurso-22] Informações : RECURSO - Valor - Indice de Sucesso 25
[Recurso-17] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-17
[Recurso-17] Informações : RECURSO - Valor - Indice de Sucesso 21
[Recurso-12] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-12
[Recurso-12] Informações : RECURSO - Valor - Indice de Sucesso 14

[Recurso-24] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-24
[Recurso-24] Informações : RECURSO - Valor - Indice de Sucesso 0

Os outros treze irão sempre finalizar com sucesso suas missões. Considerando que as subcontratações estão habilitadas nessa simulação, esse cenário é razoavelmente bom, como será visto, e o sistema deve reagir bem com um número elevado de missões finalizadas com sucesso. Dos treze agentes recursos que sempre finalizarão com sucesso suas missões, quatro são do tipo "drone1", quatro são do tipo "robot3" e cinco são do tipo "drone2" (ver lista a seguir), o que representa uma boa distribuição, para o caso do sucesso das subcontratações.

[Recurso-9] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-9
[Recurso-9] Informações : RECURSO - Valor - Indice de Sucesso 52
[Recurso-10] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-10
[Recurso-10] Informações : RECURSO - Valor - Indice de Sucesso 95
[Recurso-23] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-23
[Recurso-23] Informações : RECURSO - Valor - Indice de Sucesso 94
[Recurso-14] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-14
[Recurso-14] Informações : RECURSO - Valor - Indice de Sucesso 47

[Recurso-26] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-26
[Recurso-26] Informações : RECURSO - Valor - Indice de Sucesso 51
[Recurso-15] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-15
[Recurso-15] Informações : RECURSO - Valor - Indice de Sucesso 66
[Recurso-13] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-13
[Recurso-13] Informações : RECURSO - Valor - Indice de Sucesso 99
[Recurso-18] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-18
[Recurso-18] Informações : RECURSO - Valor - Indice de Sucesso 79

[Recurso-25] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-25
[Recurso-25] Informações : RECURSO - Valor - Indice de Sucesso 42
[Recurso-20] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-20
[Recurso-20] Informações : RECURSO - Valor - Indice de Sucesso 78
[Recurso-19] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-19
[Recurso-19] Informações : RECURSO - Valor - Indice de Sucesso 45
[Recurso-21] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-21
[Recurso-21] Informações : RECURSO - Valor - Indice de Sucesso 63
[Recurso-28] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-28
[Recurso-28] Informações : RECURSO - Valor - Indice de Sucesso 61

Com o sistema iniciado com seus agentes é possível acompanhar as negociações das missões e seus resultados em tempo real pela interface gráfica e através dos terminais de saída do Netbeans ou do agente Console da plataforma MaDKit.

A figura mostrada a seguir (figura 6.2) ilustra a interface gráfica criada na simulação. Pode-se observar os agentes cliente na parte superior da tela, agentes gerente na parte intermediária e os agentes recurso na parte inferior da tela. Há uma mudança de cor nos agentes conforme o andamento das negociações e missões: indo de cinza, agente ainda inativo ou solicitando serviço; para amarelo, quando está sendo atendido ou em execução de um pedido; e, por fim, para verde ou vermelho, indicando o sucesso ou a falha na missão ou pedido.

Na figura 6.3 pode-se observar o ambiente de desenvolvimento Netbeans e o terminal de saída da execução da simulação. Na parte inferior desta figura pode-se observar a geração das contagens descritas anteriormente.

A figura 6.4 ilustra o agente Console da plataforma MaDKit com o terminal de saída registrando as interações com os agentes clientes. Através deste terminal é possível também extrair outras medidas na simulação, bastando selecionar a opção Jconsole Monitoring (do agente Console). As figuras de 6.5 a 6.8 ilustram essas medidas para monitoramento da aplicação Java MySchedule, que inicia os outros agentes do sistemas. Pode-se acompanhar o uso memória, a quantidade de classes e as *threards* carregadas, além do uso de CPU da aplicação.

Figura 6-2 Interface gráfica dos agentes para acompanhamento das simulações.

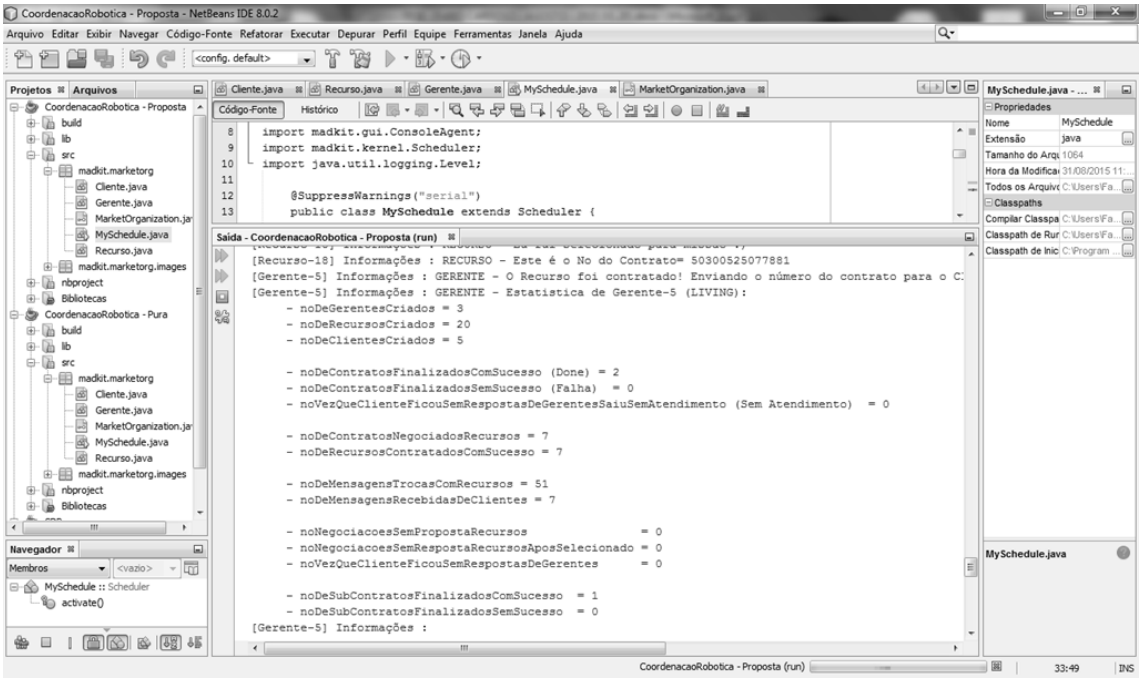

Figura 6-3 Ambiente de desenvolvimento Netbeans e terminal de saída da execução da simulação.

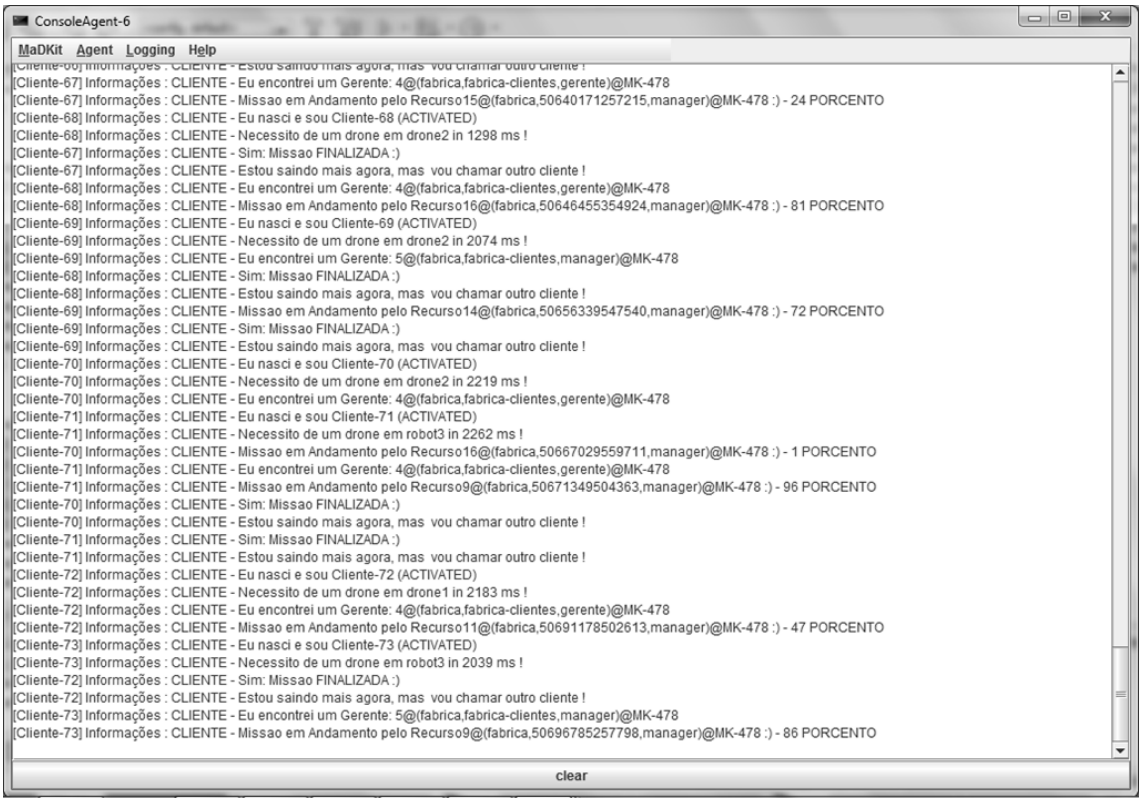

Figura 6-4 Agente Console da plataforma MaDKit e com log de saída registrando as interações dos agentes clientes.

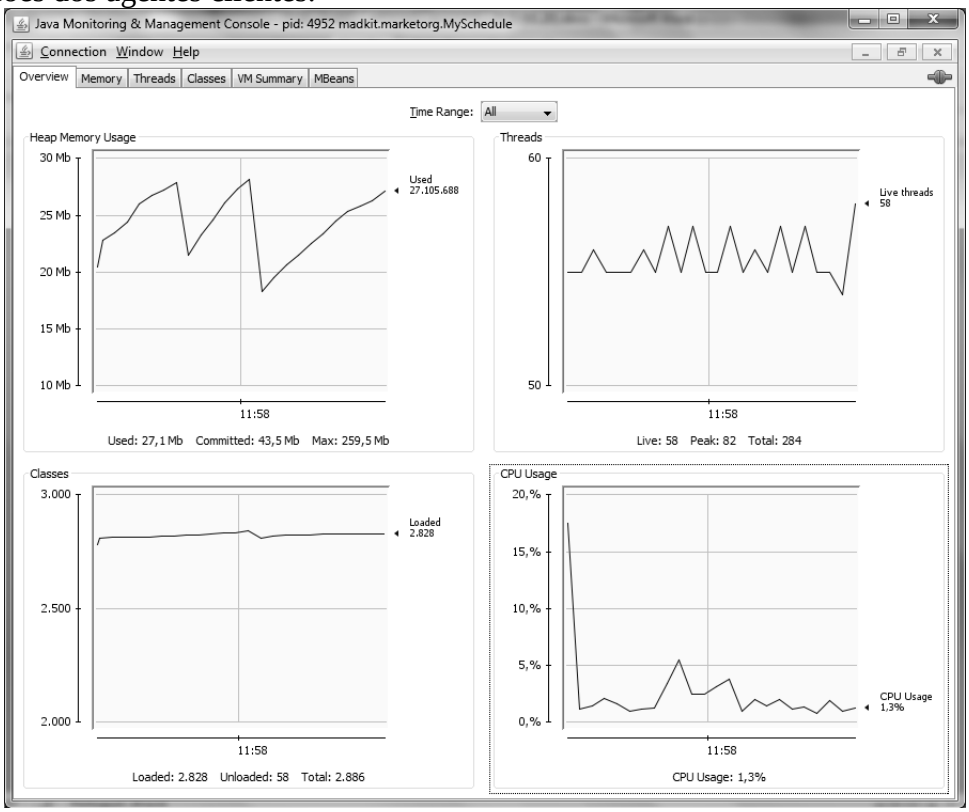

Figura 6-5 Console de Monitoramento e Gerenciamento do Agente Console da plataforma MaDKit com visão geral do uso de memória, de classes, de *threads* e uso da CPU da simulação iniciada a partir de MySchedule.

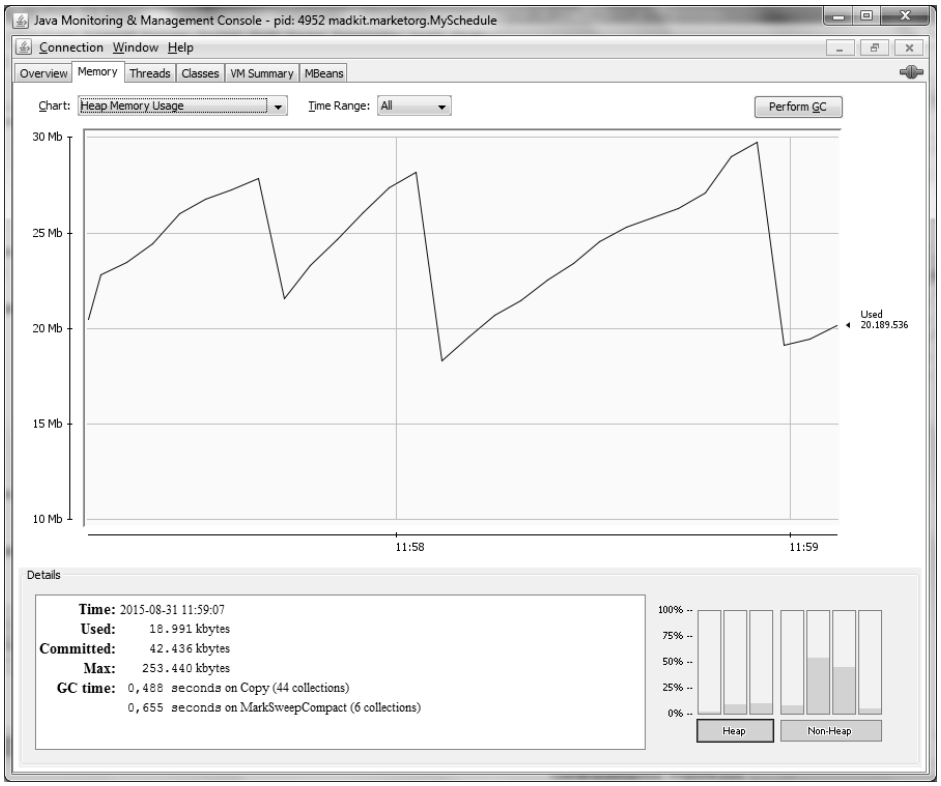

Figura 6-6 Console de Monitoramento e Gerenciamento do Agente Console da plataforma MaDKit mostrando uso de memória da aplicação iniciada a partir do arquivo Java MySchedule.

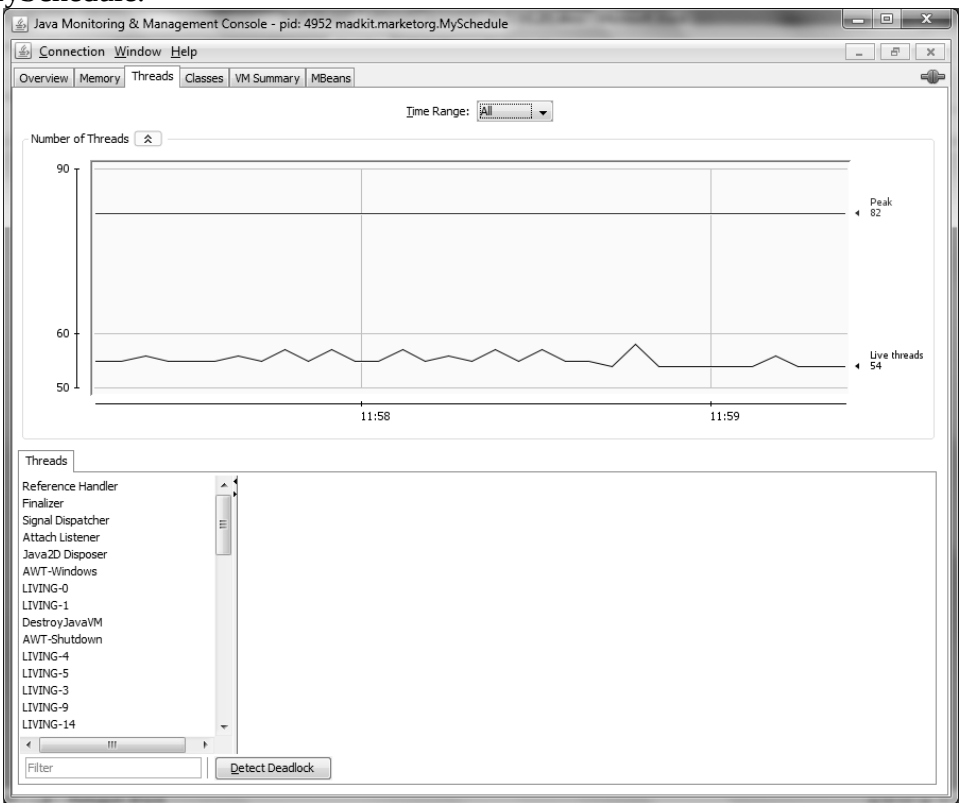

Figura 6-7 Console de Monitoramento e Gerenciamento do Agente Console da plataforma MaDKit mostrando número de *Threads* da aplicação iniciada a partir do arquivo Java MySchedule.

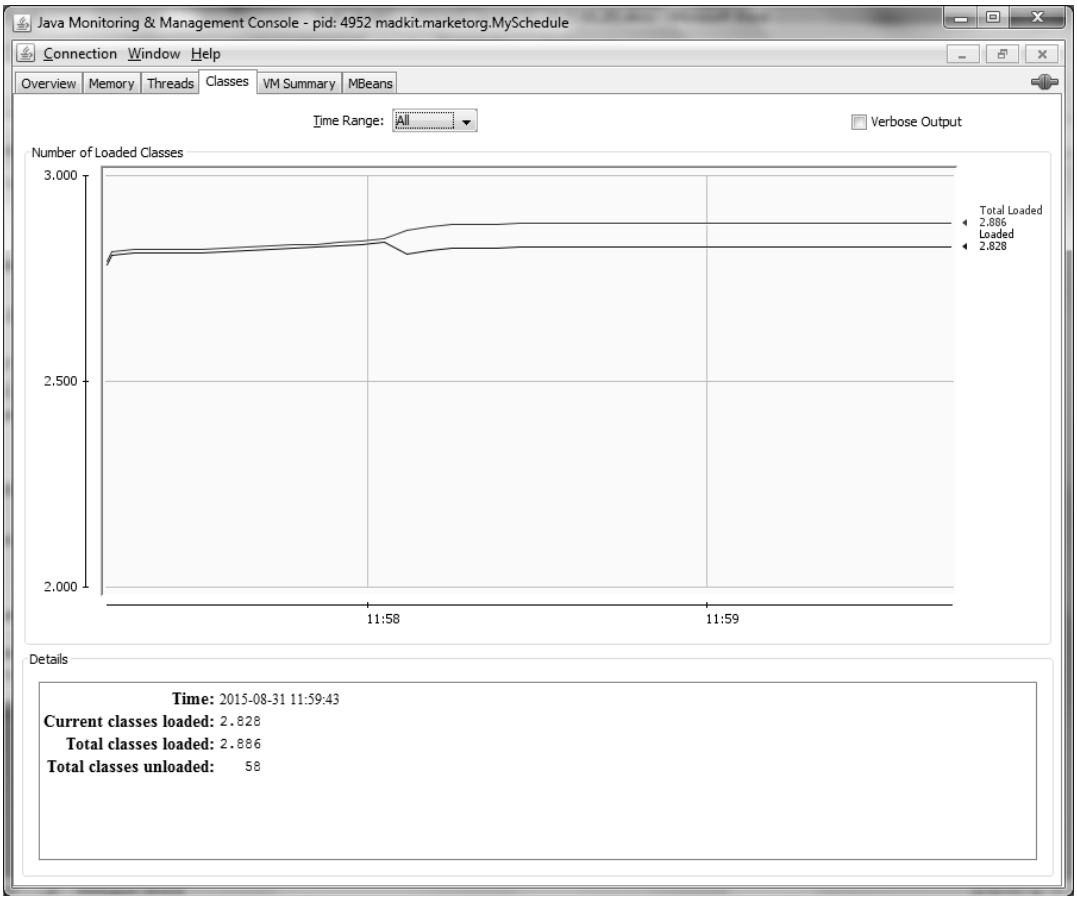

Figura 6-8 Console de Monitoramento e Gerenciamento do Agente Console da plataforma MaDKit mostrando número de classes carregadas da aplicação iniciada a partir do arquivo Java MySchedule.

O Anexo II mostra um arquivo de simulação gerado com tempo limitado de 44 segundos. Ao final desta simulação pode-se observar os seguintes resultados no arquivo gerado:

- noDeGerentesCriados = 3
- noDeRecursosCriados = 20
- noDeClientesCriados = 7

- noDeContratosFinalizadosComSucesso (Done) = 4
- noDeContratosFinalizadosSemSucesso (Falha) = 1
- noVezQueClienteFicouSemRespostasDeGerentesSaiuSemAtendimento = 0

- noDeContratosNegociadosRecursos = 8
- noDeRecursosContratadosComSucesso = 8

- noDeMensagensTrocasComRecursos = 65
- noDeMensagensRecebidasDeClientes = 8

- noNegociacoesSemPropostaRecursos = 0
- noNegociacoesSemRespostaRecursosAposSelecionado = 0
- noVezQueClienteFicouSemRespostasDeGerentes = 0

- noDeSubContratosFinalizadosComSucesso = 1
- noDeSubContratosFinalizadosSemSucesso = 1

Os resultados mostram que foram criados sete agentes do tipo cliente com seus pedidos, destes quatro contratos ou missões foram finalizados com sucesso e houve uma falha. Houve um total de oito negociações e também por oito vezes os contratos foram estabelecidos. Foram trocadas sessenta e cinto mensagens com os agentes recursos e foram recebidas oito mensagens de agentes do tipo cliente. O número de negociações sem propostas de recurso foi de zero, assim como não houveram negociações sem respostas de recursos após seleciados. Também não houveram situações em que um agente cliente ficou sem respostas de agentes do tipo gerente. O número de subcontratações finalizadas com sucesso foi de um e o número de subcontratações com falha foi de um também.

6.1.1 Resultados do Protótipo da Arquitetura usando Plataforma MaDKit

A seguir são apresentados os resultados das simulações utilizando uma versão da aplicação desenvolvida com protocolo *contract net* conforme foi definido por Smith em 1980, no seu trabalho intitulado "*The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver*".

Os resultados de uma simulação serão detalhados aqui para se ter uma maior compreensão de todo o processo. Os resultados apresentados nas tabelas e gráficos a seguir foram extraídos de uma simulação com CNP (*contract net protocol*) Puro. Definiu-se que os agentes com índice de sucesso maior que vinte sempre teriam sucesso em suas missões. Na simulação apresentada a seguir, apenas doze agentes sempre tinham sucesso ou outros oito, com índice de sucesso menores ou igual a vinte, sempre falhavam quando selecionados. Nesta versão de *contract net* não se utilizou nem mecanismos de subcontratação e nem de aprendizado.

A tabela 6-1 mostra o resultado da simulação utilizando a versão da aplicação chamada de CNP Puro com índice de sucesso (IS) superior a vinte para que os agentes do tipo recurso tenham sucesso em suas missões. As informações foram coletadas minuto a minuto até 10 minutos. Extraiu-se também informações em 20 minutos de simulação.

A tabela 6-2 trás o resultado percentual, que foi detalhado na tabela 6-1, da contagem do número de sucessos e falhas nas missões ou contratos, utilizando a versão CNP Puro e IS>20 para sucesso em missões por parte dos agentes do tipo recurso.

A figura 6-9 apresenta o resultado gráfico da simulação, minuto a minuto, detalhando o número de clientes criados, o número de contratos finalizados com sucesso e o número de contratos que falharam para o CNP Puro e IS >20 para missões bem sucedidas.

A figura 6.10 apresenta o gráfico da simulação detalhando o número de contratos negociados, o número de vezes que recursos foram contratados, o número de mensagens trocadas com agentes do tipo recurso e a quantidade de mensagens recebidas de clientes pelos agentes do tipo gerente, utilizando aplicação nomeada de CNP Puro com IS>20 para missões bem sucedidas.

A figura 6-11 trás um gráfico da simulação detalhando o número de vezes que os clientes ficaram sem resposta de agentes gerentes, o número de negociações sem resposta após seleção dos recursos e o número de negociações sem propostas por parte dos agentes do tipo recurso (com versão CNP Puro com IS>20 para sucesso em missões).

Tabela 6-1 Resultado de uma simulação empregando CNP Puro e sucesso nas missões para agentes com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	10	18	27	35	43	52	60	69	78	88	175
Contratos Finalizados Sucesso	6	12	19	26	33	40	46	52	60	70	151
Contratos que Falharam	2	4	5	5	7	8	10	14	14	14	21
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	18	27	34	46	55	64	73	83	92	184
Recursos Contratados	10	18	27	34	44	53	62	71	81	90	179
Mensagens Trocas c/ Recursos	64	110	154	183	221	253	289	327	361	396	712
Mens. Recebidas de Clientes	10	18	27	34	46	55	64	73	83	93	184
Negociações Sem Propostas	0	0	0	0	1	1	1	1	1	1	4
Neg. SemResp. Após Seleção	0	0	0	0	1	1	1	1	1	1	1
Cliente Ficou SemResp.Gerente	0	0	0	0	3	3	4	4	5	5	9

Tabela 6-2 Resultado percentual de sucesso e falha nos contratos, utilizando versão CNP Puro e IS>20

Contratos Finalizados com Sucesso	86%
Contratos que Falharam	12%
Cliente Saiu Sem Resposta	0%

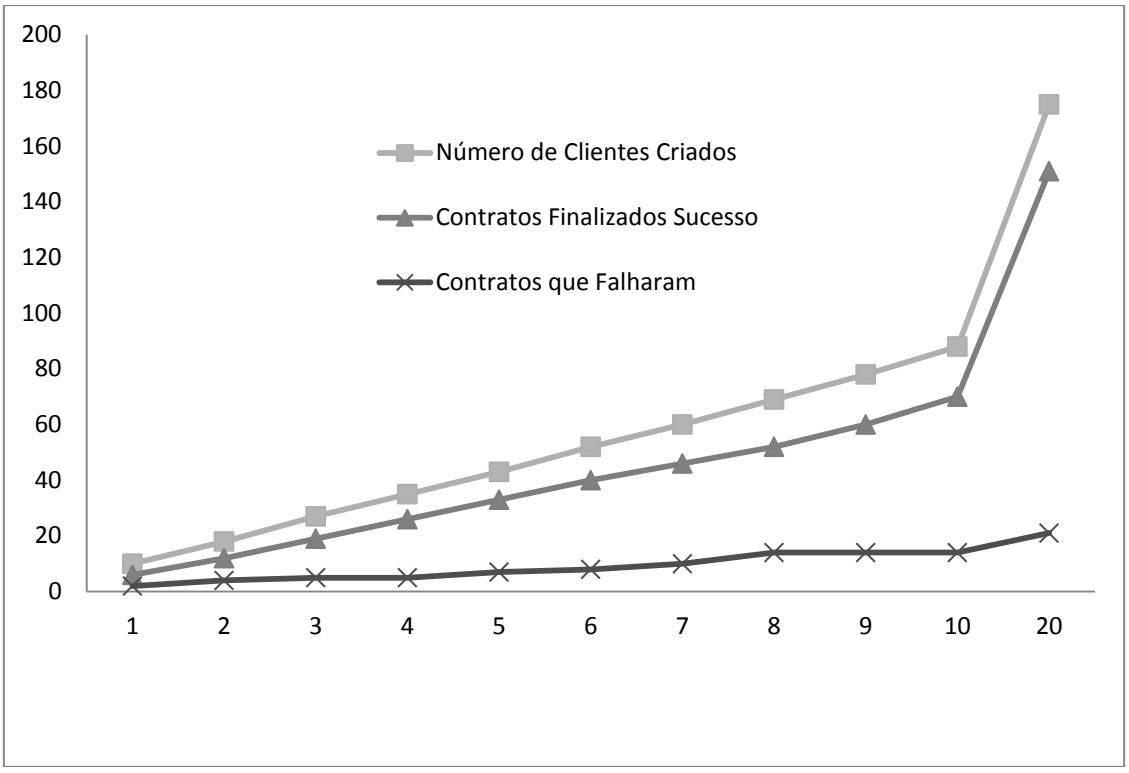

Figura 6-9 Resultado gráfico da simulação, minuto a minuto, detalhando número de clientes criados, contratos finalizados com sucesso e contratos que falharam para o CNP Puro e índice de sucesso (IS) >20.

.

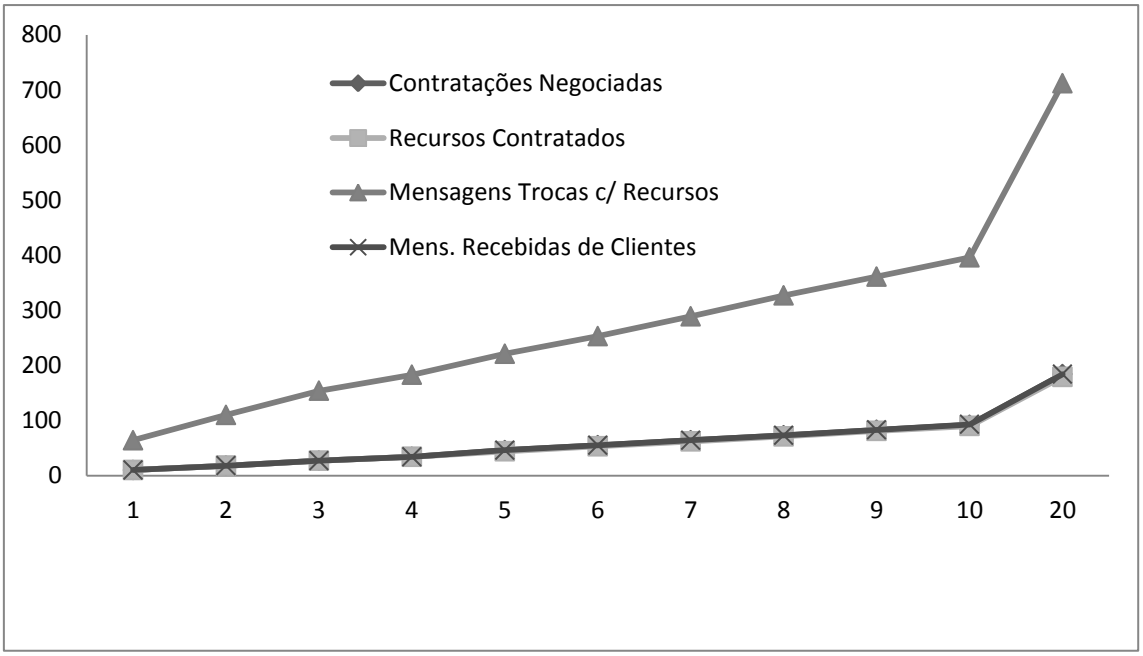

Figura 6-10 Resultado gráfico da simulação, minuto a minuto, detalhando número de contratos negociados, número de vezes que recursos foram contratados, mensagem trocas com agentes do tipo recurso e mensagens recebidas de agentes clientes pelos agentes do tipo gerente, utilizando versão da aplicação - CNP Puro com índice de sucesso (IS) >20.

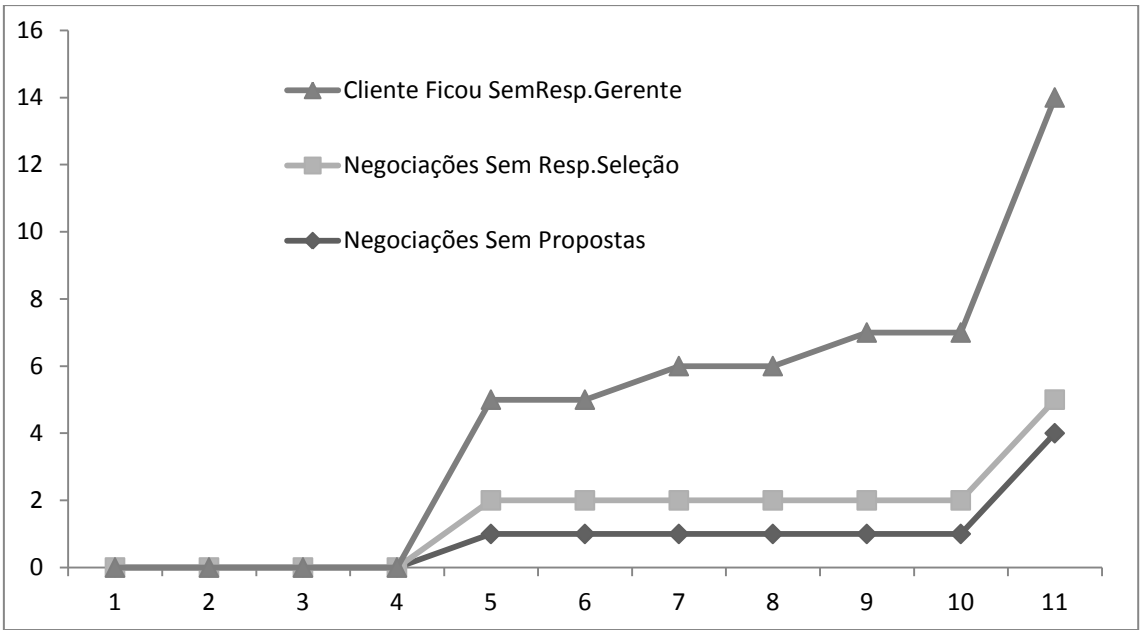

Figura 6-11 Resultado gráfico da simulação, minuto a minuto, detalhando número de vezes que os clientes ficaram sem resposta de agentes gerentes, número de negociações sem resposta após seleção dos recursos e número de negociações sem propostas por parte dos agentes do tipo recurso (CNP Puro com IS>20 para sucesso em missões).

A seguir estão apresentados os resultados das simulações realizadas com a versão dita CNP com IS>20 para sucesso em missões por parte dos agentes do tipo recurso. Foram feitas cinco simulações para esse caso específico. As tabelas de 6-3 a 6-7 trazem o resultado detalhado de cada simulação realizada, usando versão da aplicação CNP Puro com IS>20 para definir o sucesso das missões por parte dos agentes do tipo recurso.

Tabela 6-3 Resultado da primeira simulação empregando CNP Puro e IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	10	18	27	35	44	54	63	71	80	89	177
Contratos Finalizados Sucesso	7	12	15	20	27	33	39	44	50	54	96
Contratos que Falharam	1	4	9	13	15	19	21	25	28	32	79
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	19	28	37	46	56	64	73	82	91	185
Recursos Contratados	10	18	26	35	44	54	62	71	80	89	181
Mensagens Trocas c/ Recursos	70	136	184	218	258	301	335	370	407	442	807
Mens. Recebidas de Clientes	10	19	28	37	46	56	65	73	82	91	185
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	2
Negociações Sem Resp.Seleção	0	1	2	2	2	2	2	2	2	2	2
Cliente Ficou SemResp.Gerente	0	1	2	2	2	2	2	2	2	2	8

Tabela 6-4 Resultado da segunda simulação empregando CNP Puro e IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	11	18	26	34	43	53	61	70	78	87	177
Contratos Finalizados Sucesso	9	14	20	29	35	43	50	58	65	71	153
Contratos que Falharam	1	1	2	3	5	7	8	9	10	13	20
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	12	18	31	41	49	60	68	77	86	95	187
Recursos Contratados	11	18	27	36	44	54	62	71	79	88	178
Mensagens Trocas c/ Recursos	95	134	186	218	252	285	313	344	379	420	755
Mens. Recebidas de Clientes	12	19	31	41	49	60	68	77	86	95	187
Negociações Sem Propostas	0	0	0	1	1	2	2	2	3	3	5
Negociações Sem Resp.Seleção	1	1	4	4	4	4	4	4	4	4	4
Cliente Ficou SemResp.Gerente	1	1	5	6	6	7	7	7	8	8	11

Tabela 6-5 Resultado da terceira simulação empregando CNP Puro e IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	10	19	27	36	45	54	62	71	80	89	176
Contratos Finalizados Sucesso	8	17	24	34	42	49	57	64	73	82	156
Contratos que Falharam	0	0	1	1	2	3	4	6	6	6	19
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	13	23	31	40	50	59	68	77	86	96	190
Recursos Contratados	12	21	29	38	47	56	64	73	82	91	183
Mensagens Trocas c/ Recursos	97	140	171	206	251	287	320	354	392	425	786
Mens. Recebidas de Clientes	13	23	31	40	50	59	68	77	86	96	190
Negociações Sem Propostas	0	0	0	0	0	0	1	1	1	1	2
Negociações Sem Resp.Seleção	1	2	2	2	3	3	3	3	3	4	5
Cliente Ficou SemResp.Gerente	3	4	4	4	5	5	6	6	6	7	14

Tabela 6-6 Resultado da quarta simulação empregando CNP Puro e IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	11	19	27	36	45	54	62	71	80	88	178
Contratos Finalizados Sucesso	8	15	22	32	40	50	58	67	76	84	174
Contratos que Falharam	2	2	2	2	2	2	2	2	2	2	2
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	12	23	31	42	51	60	68	79	88	97	193
Recursos Contratados	11	20	28	39	48	57	0	75	84	92	186
Mensagens Trocas c/ Recursos	95	149	180	223	254	290	3	355	386	417	771
Mens. Recebidas de Clientes	12	23	31	42	51	60	6	79	88	97	193
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	1	1
Negociações Sem Resp.Seleção	1	3	3	3	3	3	3	4	4	4	6
Cliente Ficou SemResp.Gerente	1	4	4	6	6	6	6	8	8	9	15

Tabela 6-7 Resultado da quinta simulação empregando CNP Puro e IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	11	19	27	36	45	54	62	71	80	88	178
Contratos Finalizados Sucesso	9	16	24	34	42	52	61	70	79	87	176
Contratos que Falharam	1	1	1	1	1	1	1	1	1	1	1
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	11	20	31	40	49	59	68	77	86	94	186
Recursos Contratados	11	20	29	38	47	57	66	75	84	92	181
Mensagens Trocas c/ Recursos	92	140	194	228	26-	299	333	366	399	429	764
Mens. Recebidas de Clientes	11	20	31	40	49	59	68	77	86	94	186
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	3
Negociações Sem Resp.Seleção	0	0	2	2	2	2	2	2	2	2	2
Cliente Ficou SemResp.Gerente	0	1	3	3	4	4	4	4	4	4	7

As tabelas 6-8 e 6-9 apresentam os resultados finais após apurar a média de cada simulação realizada (CNP Puro com IS>20). Verificou-se que em média 85% das missões são finalizadas e 14% falham nessa configuração.

Tabela 6-8 Resultado final apurando a média das simulações (de 1 a 5) empregando CNP Puro e IS>20.

Minutos	20	20	20	20	20	**Média**
Número de Clientes Criados	177	177	176	178	178	**177**
Contratos Finalizados Sucesso	96	153	156	174	176	**151**
Contratos que Falharam	79	20	19	2	1	**24**
Cliente Saiu Sem Resposta	0	0	0	0	0	**0**
Contratações Negociadas	185	187	190	193	186	**188**
Recursos Contratados	181	178	183	186	181	**182**
Mensagens Trocas c/ Recursos	807	755	786	771	764	**777**
Mens. Recebidas de Clientes	185	187	190	193	186	**188**
Negociações Sem Propostas	2	5	2	1	3	**2,6**
Negociações Sem Resp.Seleção	2	4	5	6	2	**3,8**
Cliente Ficou SemResp.Gerente	8	11	14	15	7	**11**

Tabela 6-9 Resultado final percentual, após apurar a média das simulações (de 1 a 5), empregando CNP Puro e IS>20.

Resultado Final da Simulação - CNP Puro com IS>20	
Contratos Finalizados com Sucesso	85%
Contratos que Falharam	14%
Cliente Saiu Sem Resposta	0%

A seguir estão apresentados os resultados das simulações realizadas com a versão dita CNP Puro com IS>40 para sucesso em missões por parte dos agentes do tipo recurso. As tabelas de 6-3 a 6-7 trazem o resultado detalhado de cada simulação realizada, empregando versão da aplicação CNP Puro com IS>40 para definir o sucesso das missões por parte dos agentes do tipo recurso.

Tabela 6-10 Resultado da primeira simulação empregando CNP Puro e IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	11	20	27	36	44	53	62	71	80	89	178
Contratos Finalizados Sucesso	7	11	13	17	21	25	28	32	35	38	69
Contratos que Falharam	3	7	12	17	21	26	32	36	42	49	107
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	12	23	32	43	51	60	69	78	87	96	189
Recursos Contratados	11	20	28	38	46	55	64	73	82	91	180
Mensagens Trocas c/ Recursos	84	139	174	219	245	282	320	353	388	423	744
Mens. Recebidas de Clientes	12	23	32	43	51	60	69	78	87	96	189
Negociações Sem Propostas	0	2	2	2	2	2	2	2	2	2	6
Negociações Sem Resp.Seleção	1	1	2	3	3	3	3	3	3	3	3
Cliente Ficou SemResp.Gerente	1	3	5	7	7	7	7	7	7	7	11

Tabela 6-11 Resultado da segunda simulação empregando CNP Puro e IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	10	16	24	31	38	44	51	60	63	68	136
Contratos Finalizados Sucesso	6	7	9	13	16	18	20	23	24	26	52
Contratos que Falharam	2	7	14	16	20	24	30	35	37	40	80
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	9	16	25	31	39	49	57	65	68	73	146
Recursos Contratados	9	16	25	31	38	48	56	64	67	72	144
Mensagens Trocas c/ Recursos	74	117	167	199	238	286	325	369	382	405	810
Mens. Recebidas de Clientes	9	16	25	31	39	48	57	65	68	73	146
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	0
Negociações Sem Resp.Seleção	0	0	0	0	1	1	1	1	1	1	2
Cliente Ficou SemResp.Gerente	7	28	38	56	86	96	117	129	141	149	298

Tabela 6-12 Resultado da terceira simulação empregando CNP Puro e IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	10	18	24	34	42	50	59	67	76	85	169
Contratos Finalizados Sucesso	6	10	11	15	20	24	27	31	36	40	79
Contratos que Falharam	2	6	9	15	19	23	29	33	37	41	87
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	11	21	28	37	46	54	63	71	80	88	175
Recursos Contratados	10	18	25	34	43	51	60	68	77	85	172
Mensagens Trocas c/ Recursos	96	166	200	238	278	312	350	328	425	459	836
Mens. Recebidas de Clientes	11	21	28	37	46	54	63	71	80	88	175
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	0
Negociações Sem Resp.Seleção	1	3	3	3	3	3	3	3	3	3	3
Cliente Ficou SemResp.Gerente	11	21	22	25	26	30	34	36	37	38	84

Tabela 6-13 Resultado da quarta simulação empregando CNP Puro e IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	11	18	27	36	45	54	63	72	81	90	180
Contratos Finalizados Sucesso	6	9	13	19	26	33	38	43	50	55	115
Contratos que Falharam	3	7	11	15	16	19	23	27	29	33	62
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	11	19	31	40	49	58	68	77	86	95	189
Recursos Contratados	11	18	28	37	46	55	64	73	82	91	184
Mensagens Trocas c/ Recursos	93	132	184	220	254	288	327	366	400	436	796
Mens. Recebidas de Clientes	11	19	31	40	49	58	68	77	86	95	189
Negociações Sem Propostas	0	0	1	1	1	1	2	2	2	2	3
Negociações Sem Resp.Seleção	0	1	2	2	2	2	2	2	2	2	2
Cliente Ficou SemResp.Gerente	0	1	4	4	4	4	5	5	5	5	10

Tabela 6-14 Resultado da quinta simulação empregando CNP Puro e IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	11	18	27	34	42	51	60	68	78	86	171
Contratos Finalizados Sucesso	5	9	17	23	28	36	42	49	58	64	136
Contratos que Falharam	3	7	7	8	11	13	15	17	17	19	33
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	19	27	36	44	53	62	70	80	89	180
Recursos Contratados	9	18	26	34	42	51	60	68	78	87	177
Mensagens Trocas c/ Recursos	87	141	172	209	243	279	315	346	382	418	781
Mens. Recebidas de Clientes	11	19	27	36	44	53	62	70	80	89	180
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	0
Negociações Sem Resp.Seleção	1	1	1	2	2	2	2	2	2	2	3
Cliente Ficou SemResp.Gerente	3	7	16	22	26	27	29	32	39	40	70

As tabelas 6-15 e 6-16 apresentam os resultados finais após apurar a média de cada simulação realizada (CNP Puro com IS>40). Verificou-se que em média 54% das missões são finalizadas e 44% falham nessa configuração. O resultado é previsto considerando que agora apenas os agentes do tipo recurso com índice de sucesso acima de 40 não falham nas missões.

Tabela 6-15 Resultado final apurando a média das simulações (de 1 a 5) empregando CNP Puro e IS>40.

Minutos	20	20	20	20	20	**Média**
Número de Clientes Criados	178	136	169	180	171	**167**
Contratos Finalizados Sucesso	69	52	79	115	136	**90**
Contratos que Falharam	107	80	87	62	33	**74**
Cliente Saiu Sem Resposta	0	0	0	0	0	**0**
Contratações Negociadas	189	146	175	189	180	**176**
Recursos Contratados	180	144	172	184	177	**171**
Mensagens Trocas c/ Recursos	744	810	836	796	781	**793**
Mens. Recebidas de Clientes	189	146	175	189	180	**176**
Negociações Sem Propostas	6	0	0	3	0	**1,8**
Negociações Sem Resp.Seleção	3	2	3	2	3	**2,6**
Cliente Ficou SemResp.Gerente	11	298	84	10	70	**95**

Tabela 6-16 Resultado final percentual, após apurar a média das simulações (de 1 a 5), empregando CNP Puro e IS>40.

Resultado Final da Simulação - CNP Puro com IS>40	
Contratos Finalizados com Sucesso	54%
Contratos que Falharam	44%
Cliente Saiu Sem Resposta	0%

A seguir partiu-se para as simulações utilizando as funcionalidades propostas na arquitetura, ou seja, subcontratações e melhora continuada no índice de sucesso dos agentes do tipo recurso que falharam, subcontrataram outro agente do tipo recurso que finalizou com sucesso sua missão e compartilhou seu conhecimento.

A seguir estão apresentados os resultados das simulações realizadas com a versão CNP com subcontratações e com IS>20 e IS>40 para sucesso em missões por parte dos agentes do tipo recurso. As tabelas de 6-17 a 6-21 trazem o resultado detalhado de cada simulação realizada, empregando versão da aplicação CNP com subcontratações com IS>20. As tabelas de 6-22 a 6-26 trazem o resultado detalhado de cada simulação realizada, empregando versão da aplicação CNP com subcontratações com IS>40 para definir o sucesso das missões por parte dos agentes do tipo recurso.

Tabela 6-17 Resultado da primeira simulação empregando CNP com subcontratações e com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	14	21	27	34	41	49	55	61	68	140
Contratos Finalizados Sucesso	6	12	17	24	31	37	45	51	56	63	135
Contratos que Falharam	0	0	1	1	1	1	1	1	2	3	3
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	17	26	39	48	55	63	71	82	90	173
Recursos Contratados	10	16	24	35	44	51	59	67	77	84	162
Mensagens Trocas c/ Recursos	86	145	293	274	328	359	409	450	508	548	930
Mens. Recebidas de Clientes	10	17	26	39	48	55	63	71	82	90	173
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	5
Negociações Sem Resp.Seleção	0	1	2	4	4	4	4	4	5	6	6
Cliente Ficou SemResp.Gerente	4	13	21	31	36	42	46	53	58	63	100
SubContrFinalizadosComSucesso	2	2	2	4	5	5	6	7	8	8	12
SubContrFinalizadosSemSucesso	0	0	1	1	1	1	1	1	2	3	3

Tabela 6-18 Resultado da segunda simulação empregando CNP com subcontratações e com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	15	21	29	36	43	48	56	61	68	138
Contratos Finalizados Sucesso	7	14	18	23	29	34	40	44	50	54	98
Contratos que Falharam	0	0	1	3	4	6	6	9	9	12	38
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	9	19	30	40	51	62	70	83	92	103	214
Recursos Contratados	9	19	28	36	46	55	63	73	82	90	172
Mensagens Trocas c/ Recursos	76	133	183	225	276	322	359	406	450	490	908
Mens. Recebidas de Clientes	9	19	30	40	51	63	70	83	92	103	214
Negociações Sem Propostas	0	0	1	3	4	6	6	9	9	12	39
Negociações Sem Resp.Seleção	0	0	1	1	1	1	1	1	1	1	3
Cliente Ficou SemResp.Gerente	0	0	2	2	3	3	3	3	3	3	9
SubContrFinalizadosComSucesso	1	1	5	6	8	10	12	15	18	19	28
SubContrFinalizadosSemSucesso	0	0	1	3	4	6	6	9	9	12	37

Tabela 6-19 Resultado da terceira simulação empregando CNP com subcontratações e com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	14	21	29	36	44	50	57	63	70	138
Contratos Finalizados Sucesso	7	11	17	25	32	40	44	51	57	64	131
Contratos que Falharam	0	0	0	0	0	0	2	2	2	2	3
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	11	19	29	42	52	60	71	83	93	102	197
Recursos Contratados	11	18	27	35	42	50	61	70	79	88	177
Mensagens Trocas c/ Recursos	86	124	172	215	250	287	368	414	459	507	974
Mens. Recebidas de Clientes	11	19	29	42	52	60	72	84	94	103	199
Negociações Sem Propostas	0	0	0	4	7	7	7	10	11	11	17
Negociações Sem Resp.Seleção	0	1	2	3	3	3	3	3	3	3	3
Cliente Ficou SemResp.Gerente	0	2	5	10	13	13	13	16	17	17	25
SubContrFinalizadosComSucesso	3	3	3	3	3	3	6	8	10	13	31
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	2	2	2	2	3

Tabela 6-20 Resultado da quarta simulação empregando CNP com subcontratações e com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	7	15	21	27	34	42	48	56	63	69	142
Contratos Finalizados Sucesso	4	11	16	23	29	37	43	52	58	64	137
Contratos que Falharam	0	0	0	0	0	0	0	0	0	0	0
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	8	18	25	37	45	53	63	72	79	89	177
Recursos Contratados	8	16	23	33	41	49	59	68	75	84	168
Mensagens Trocas c/ Recursos	75	148	190	248	287	326	369	416	452	495	912
Mens. Recebidas de Clientes	8	18	25	37	45	53	63	72	80	89	178
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	2
Negociações Sem Resp.Seleção	0	2	2	4	4	4	4	4	4	5	7
Cliente Ficou SemResp.Gerente	0	2	2	4	4	5	6	6	6	7	13
SubContrFinalizadosComSucesso	1	2	3	6	7	7	8	10	10	12	22
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	0	0	0	0	0

Tabela 6-21 Resultado da quinta simulação empregando CNP com subcontratações e com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	9	14	19	25	33	39	46	54	60	66	134
Contratos Finalizados Sucesso	8	11	15	21	27	33	40	46	52	58	122
Contratos que Falharam	0	0	0	1	2	2	3	4	5	5	9
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	22	31	42	52	61	74	88	98	106	209
Recursos Contratados	10	20	27	38	48	57	66	77	86	94	191
Mensagens Trocas c/ Recursos	86	174	215	274	324	362	409	469	516	555	1110
Mens. Recebidas de Clientes	10	22	31	42	52	61	74	88	98	106	210
Negociações Sem Propostas	0	0	0	0	0	0	3	5	5	5	11
Negociações Sem Resp.Seleção	0	2	4	4	4	4	5	6	7	7	7
Cliente Ficou SemResp.Gerente	0	2	4	4	4	4	8	11	11	11	20
SubContrFinalizadosComSucesso	1	3	5	9	12	13	14	18	20	22	43
SubContrFinalizadosSemSucesso	0	1	1	2	3	3	4	5	6	6	10

Tabela 6-22 Resultado da primeira simulação empregando CNP com subcontratações e IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	13	17	23	31	36	43	49	55	61	119
Contratos Finalizados Sucesso	6	7	7	10	17	22	26	31	35	40	83
Contratos que Falharam	0	2	4	8	8	9	11	13	14	15	28
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	11	23	36	55	66	76	89	102	115	131	245
Recursos Contratados	11	22	34	46	57	67	80	92	105	120	229
Mensagens Trocas c/ Recursos	82	176	239	310	379	432	495	553	622	702	1263
Mens. Recebidas de Clientes	11	23	36	55	67	77	90	103	116	132	246
Negociações Sem Propostas	0	0	1	6	6	6	6	7	7	7	10
Negociações Sem Resp.Seleção	0	1	1	3	3	3	3	3	3	4	6
Cliente Ficou SemResp.Gerente	0	0	1	5	5	5	5	5	5	5	10
SubContrFinalizadosComSucesso	3	5	9	13	16	21	26	28	35	39	77
SubContrFinalizadosSemSucesso	0	3	5	11	11	12	14	16	18	22	37

Tabela 6-23 Resultado da segunda simulação empregando CNP com subcontratações e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	13	19	26	33	41	47	53	59	65	130
Contratos Finalizados Sucesso	6	9	14	22	29	36	42	47	52	56	119
Contratos que Falharam	0	0	0	0	0	0	1	2	3	4	7
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	11	20	29	39	48	57	69	81	92	105	211
Recursos Contratados	10	19	25	33	42	51	63	75	86	99	198
Mensagens Trocas c/ Recursos	81	141	175	210	256	300	360	423	476	542	1038
Mens. Recebidas de Clientes	11	20	30	39	48	57	69	81	92	105	211
Negociações Sem Propostas	0	0	0	2	2	2	2	2	2	2	8
Negociações Sem Resp.Seleção	1	1	4	4	4	4	4	4	4	4	5
Cliente Ficou SemResp.Gerente	1	1	5	7	7	7	7	7	7	7	14
SubContrFinalizadosComSucesso	2	4	5	5	7	9	13	17	19	24	50
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	1	3	4	6	11

Tabela 6-24 Resultado da terceira simulação empregando CNP com subcontratações e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	9	16	22	29	36	43	50	57	65	71	140
Contratos Finalizados Sucesso	6	12	18	25	32	38	46	53	61	66	136
Contratos que Falharam	1	1	1	1	1	1	1	1	1	1	1
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	11	20	31	42	52	59	68	77	90	102	194
Recursos Contratados	11	20	30	40	50	57	66	73	82	93	180
Mensagens Trocas c/ Recursos	93	158	223	285	337	373	418	454	500	560	1029
Mens. Recebidas de Clientes	11	20	31	42	52	59	68	77	90	102	194
Negociações Sem Propostas	0	0	0	0	0	0	0	2	5	5	8
Negociações Sem Resp.Seleção	0	0	1	2	2	2	2	2	3	4	6
Cliente Ficou SemResp.Gerente	0	0	1	1	2	3	3	5	9	10	17
SubContrFinalizadosComSucesso	1	4	6	8	10	10	11	11	12	15	31
SubContrFinalizadosSemSucesso	1	1	1	1	1	1	1	1	1	1	1

Tabela 6-25 Resultado da quarta simulação empregando CNP com subcontratações e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	7	14	22	30	36	43	51	57	65	72	146
Contratos Finalizados Sucesso	4	11	17	25	31	37	45	51	59	67	141
Contratos que Falharam	0	0	0	0	0	1	1	1	1	1	1
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	8	17	25	34	41	51	59	67	74	83	172
Recursos Contratados	8	17	25	33	40	50	58	66	73	82	167
Mensagens Trocas c/ Recursos	67	135	181	230	265	321	365	406	439	484	896
Mens. Recebidas de Clientes	8	17	25	34	41	51	59	67	74	83	172
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	4
Negociações Sem Resp.Seleção	0	0	0	1	1	1	1	1	1	1	1
Cliente Ficou SemResp.Gerente	0	0	0	1	1	1	1	1	1	1	8
SubContrFinalizadosComSucesso	1	2	3	3	3	4	6	7	7	8	16
SubContrFinalizadosSemSucesso	0	0	0	0	0	1	1	1	1	1	1

Tabela 6-26 Resultado da quinta simulação empregando CNP com subcontratações e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	9	14	19	25	33	39	46	54	60	66	134
Contratos Finalizados Sucesso	8	11	15	21	27	33	40	46	52	58	122
Contratos que Falharam	0	0	0	1	2	2	3	4	5	5	9
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	22	31	42	52	61	74	88	98	106	209
Recursos Contratados	10	20	27	38	48	57	66	77	86	94	191
Mensagens Trocas c/ Recursos	86	174	215	274	324	362	409	469	516	555	1110
Mens. Recebidas de Clientes	10	22	31	42	52	61	74	88	98	106	210
Negociações Sem Propostas	0	0	0	0	0	0	3	5	5	5	11
Negociações Sem Resp.Seleção	0	2	4	4	4	4	5	6	7	7	7
Cliente Ficou SemResp.Gerente	0	2	4	4	4	4	8	11	11	11	20
SubContrFinalizadosComSucesso	1	3	5	9	12	13	14	18	20	22	43
SubContrFinalizadosSemSucesso	0	1	1	2	3	3	4	5	6	6	10

As tabelas 6-27 e 6-28 apresentam os resultados finais após apurar a média de cada simulação realizada (CNP com subcontratações e com IS>20). Verificou-se que em média 90% das missões são finalizadas e 8% falham nessa configuração.

Tabela 6-27 Resultado final apurando a média das simulações (de 1 a 5) empregando CNP com subcontratações e com IS>20.

Minutos	20	20	20	20	20	**Med**
Número de Clientes Criados	140	138	138	142	134	**138,4**
Contratos Finalizados Sucesso	135	98	131	137	122	**124,6**
Contratos que Falharam	3	38	3	0	9	**10,6**
Cliente Saiu Sem Resposta	0	0	0	0	0	**0**
Contratações Negociadas	173	214	197	177	209	**194**
Recursos Contratados	162	172	177	168	191	**174**
Mensagens Trocas c/ Recursos	930	908	974	912	1110	**966,8**
Mens. Recebidas de Clientes	173	214	199	178	210	**194,8**
Negociações Sem Propostas	5	39	17	2	11	**14,8**
Negociações Sem Resp.Seleção	6	3	3	7	7	**5,2**
Cliente Ficou SemResp.Gerente	100	9	25	13	20	**33,4**
SubContrFinalizadosComSucesso	12	28	31	22	43	**27,2**
SubContrFinalizadosSemSucesso	3	11	3	0	10	**5,4**

Tabela 6-28 Resultado final percentual, após apurar a média das simulações (de 1 a 5), empregando CNP com subcontratações e com IS>20.

Resultado Final da Simulação - CNP com Subcontratações e com IS>20	
Contratos Finalizados com Sucesso	90%
Contratos que Falharam	8%
Cliente Saiu Sem Resposta	0%

As tabelas 6-28 e 6-29 apresentam os resultados finais após apurar a média de cada simulação realizada (CNP com subcontratações e com IS>40). Verificou-se que em média 90% das missões são finalizadas e 7% falham nessa configuração.

Tabela 6-28 Resultado final apurando a média das simulações (de 1 a 5) empregando CNP com subcontratações e com IS>40.

Minutos	20	20	20	20	20	**Med**
Número de Clientes Criados	119	130	140	146	134	**133,8**
Contratos Finalizados Sucesso	83	119	136	141	122	**120,2**
Contratos que Falharam	28	7	1	1	9	**9,2**
Cliente Saiu Sem Resposta	0	0	0	0	0	**0**
Contratações Negociadas	245	211	194	172	209	**206,2**
Recursos Contratados	229	198	180	167	191	**193**
Mensagens Trocas c/ Recursos	1263	1038	1029	896	1110	**1067**
Mens. Recebidas de Clientes	246	211	194	172	210	**206,6**
Negociações Sem Propostas	10	8	8	4	11	**8,2**
Negociações Sem Resp. Seleção	6	5	6	1	7	**5**
Cliente Ficou SemResp. Gerente	10	14	17	8	20	**13,8**
SubContrFinalizadosComSucesso	77	50	31	16	43	**43,4**
SubContrFinalizadosSemSucesso	37	11	1	1	10	**12**

Tabela 6-29 Resultado final percentual, após apurar a média das simulações (de 1 a 5), empregando CNP com subcontratações e com IS>40.

Resultado Final da Simulação - CNP com Subcontratações e com IS>40	
Contratos Finalizados com Sucesso	90%
Contratos que Falharam	7%
Cliente Saiu Sem Resposta	0%

A seguir estão apresentados os resultados das simulações realizadas com a versão CNP com subcontratações e com simulação de aprendizagem, no caso de subcontratação finalizada com sucesso da missão por parte do agente do tipo recurso subcontratado. As simulações foram realizadas considerando IS>20 e IS>40. As tabelas de 6-30 a 6-34 trazem o resultado detalhado de cada simulação realizada, empregando versão da aplicação CNP com subcontratações e aprendizado e com IS>20. As tabelas de 6-35 a 6-44 trazem o resultado detalhado de cada simulação realizada, empregando a versão da aplicação CNP com subcontratações e aprendizado e com IS>40. Para esse caso foi realizado um total de dez simulações.

Tabela 6-30 Resultado da primeira simulação empregando CNP com subcontratações e aprendizado e com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	9	15	21	27	35	42	51	59	64	70	147
Contratos Finalizados Sucesso	8	12	16	23	31	38	46	54	60	66	142
Contratos que Falharam	0	0	0	0	0	0	0	0	0	0	0
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	20	27	35	44	52	62	72	82	93	179
Recursos Contratados	10	19	26	34	43	51	61	71	81	92	171
Mensagens Trocas c/ Recursos	85	150	184	223	264	304	348	390	431	479	867
Mens. Recebidas de Clientes	10	20	27	35	44	52	62	72	82	93	179
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	5
Negociações Sem Resp.Seleção	0	1	1	1	1	1	1	1	1	1	3
Cliente Ficou SemResp.Gerente	0	1	1	3	3	4	7	8	12	17	26
SubContrFinalizadosComSucesso	1	1	4	4	4	4	4	4	4	4	4
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	0	0	0	0	0

Tabela 6-31 Resultado da segunda simulação empregando CNP com subcontratações e aprendizado e com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	9	15	21	28	35	42	51	59	67	73	149
Contratos Finalizados Sucesso	6	11	15	22	29	36	44	52	60	67	143
Contratos que Falharam	0	0	0	0	0	0	0	0	0	0	0
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	8	15	21	30	37	45	53	61	69	76	155
Recursos Contratados	8	15	21	29	36	43	51	59	67	74	151
Mensagens Trocas c/ Recursos	59	101	148	191	228	269	308	348	387	420	804
Mens. Recebidas de Clientes	8	15	22	30	37	45	53	61	69	76	155
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	0
Negociações Sem Resp.Seleção	0	0	0	1	1	2	2	2	2	2	4
Cliente Ficou SemResp.Gerente	0	0	0	1	1	2	2	2	2	2	5
SubContrFinalizadosComSucesso	0	0	1	1	1	1	1	1	1	1	1
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	0	0	0	0	0

Tabela 6-32 Resultado da terceira simulação empregando CNP com subcontratações e aprendizado e com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	10	15	22	29	36	45	51	58	67	73	145
Contratos Finalizados Sucesso	8	12	20	27	34	42	48	56	64	70	143
Contratos que Falharam	0	0	0	0	0	0	0	0	0	0	0
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	21	29	37	44	52	59	67	75	82	161
Recursos Contratados	10	20	28	36	43	51	58	66	74	81	157
Mensagens Trocas c/ Recursos	79	161	212	249	285	325	359	398	443	479	889
Mens. Recebidas de Clientes	10	21	29	37	44	52	59	67	76	82	161
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	2
Negociações Sem Resp.Seleção	0	1	1	1	1	1	1	1	1	1	2
Cliente Ficou SemResp.Gerente	5	10	12	17	21	23	27	32	36	39	89
SubContrFinalizadosComSucesso	1	1	2	2	2	2	2	2	2	2	2
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	0	0	0	0	0

Tabela 6-33 Resultado da quarta simulação empregando CNP com subcontratações e aprendizado e com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	13	20	27	35	42	50	58	66	73	149
Contratos Finalizados Sucesso	7	10	17	24	32	39	47	55	63	69	146
Contratos que Falharam	0	0	0	0	0	0	0	0	0	0	0
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	21	28	39	49	57	66	74	83	93	173
Recursos Contratados	10	19	26	35	43	50	58	66	74	82	158
Mensagens Trocas c/ Recursos	85	157	196	249	289	322	362	404	446	496	881
Mens. Recebidas de Clientes	10	21	28	39	49	57	66	74	83	93	173
Negociações Sem Propostas	0	0	0	0	2	3	4	4	5	5	8
Negociações Sem Resp.Seleção	0	2	2	4	4	4	4	4	4	6	7
Cliente Ficou SemResp.Gerente	0	0	0	4	6	7	8	8	9	12	16
SubContrFinalizadosComSucesso	2	4	5	5	5	5	5	5	5	5	5
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	0	0	0	0	0

Tabela 6-34 Resultado da quinta simulação empregando CNP com subcontratações e aprendizado e com IS>20.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	9	15	22	29	37	44	52	59	68	76	151
Contratos Finalizados Sucesso	7	12	19	25	33	40	48	55	63	71	147
Contratos que Falharam	0	0	0	0	0	0	0	0	0	0	0
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	9	16	23	30	41	48	57	64	72	80	164
Recursos Contratados	9	16	23	29	39	45	54	61	69	77	153
Mensagens Trocas c/ Recursos	73	119	160	196	242	284	324	359	399	434	790
Mens. Recebidas de Clientes	9	16	23	30	41	49	57	64	73	80	164
Negociações Sem Propostas	0	0	0	0	1	1	1	1	1	1	8
Negociações Sem Resp.Seleção	0	0	0	1	1	2	2	2	2	2	3
Cliente Ficou SemResp.Gerente	0	0	0	1	3	4	4	4	4	4	12
SubContrFinalizadosComSucesso	0	1	1	1	1	1	1	1	1	1	1
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	0	0	0	0	0

Tabela 6-35 Resultado da primeira simulação empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	7	14	22	28	34	43	49	58	63	70	142
Contratos Finalizados Sucesso	4	9	15	21	26	32	39	46	52	59	131
Contratos que Falharam	0	0	1	2	3	5	5	6	6	6	6
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	19	28	37	45	53	60	70	77	86	159
Recursos Contratados	10	19	27	36	43	51	58	67	74	82	154
Mensagens Trocas c/ Recursos	70	132	183	227	264	306	345	396	428	471	857
Mens. Recebidas de Clientes	10	19	28	37	45	53	60	71	77	86	159
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	0
Negociações Sem Resp.Seleção	0	0	1	1	2	2	2	3	3	4	5
Cliente Ficou SemResp.Gerente	4	5	11	13	14	19	22	28	28	33	78
SubContrFinalizadosComSucesso	1	2	2	3	3	4	4	4	5	5	5
SubContrFinalizadosSemSucesso	0	0	1	2	3	5	5	6	6	6	6

Tabela 6-36 Resultado da segunda simulação empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	16	22	29	36	43	51	59	67	74	148
Contratos Finalizados Sucesso	7	15	20	25	33	39	48	56	64	71	145
Contratos que Falharam	0	0	0	1	1	1	1	1	1	1	1
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	14	23	31	42	51	59	68	76	84	93	186
Recursos Contratados	10	18	25	36	45	53	61	69	77	84	154
Mensagens Trocas c/ Recursos	98	147	200	248	296	337	379	419	455	491	857
Mens. Recebidas de Clientes	14	23	32	42	51	59	68	76	84	93	186
Negociações Sem Propostas	2	3	3	3	3	3	3	3	3	5	13
Negociações Sem Resp.Seleção	2	2	3	3	3	3	4	4	4	4	5
Cliente Ficou SemResp.Gerente	4	5	6	6	6	7	8	8	8	10	20
SubContrFinalizadosComSucesso	2	2	4	6	8	8	8	8	8	8	9
SubContrFinalizadosSemSucesso	0	0	0	1	1	1	1	1	1	1	1

Tabela 6-37 Resultado da terceira simulação empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	10	17	24	30	36	44	52	60	68	75	152
Contratos Finalizados Sucesso	8	14	20	25	32	39	47	55	64	71	147
Contratos que Falharam	0	0	0	0	0	0	0	0	0	0	0
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	10	18	26	35	44	52	61	69	77	84	167
Recursos Contratados	10	18	25	33	42	50	59	67	75	82	160
Mensagens Trocas c/ Recursos	77	132	176	222	261	297	336	374	411	443	814
Mens. Recebidas de Clientes	10	18	27	36	44	52	61	69	77	84	167
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	4
Negociações Sem Resp.Seleção	0	0	1	2	2	2	2	2	2	2	3
Cliente Ficou SemResp.Gerente	0	0	1	3	4	4	5	5	5	5	11
SubContrFinalizadosComSucesso	1	1	1	2	3	3	3	3	3	3	3
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	0	0	0	0	0

Tabela 6-38 Resultado da quarta simulação empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	6	7	14	18	22	25	28	32	35	39	71
Contratos Finalizados Sucesso	2	3	6	9	11	13	16	19	22	26	49
Contratos que Falharam	2	2	4	5	5	5	5	5	5	5	10
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	13	27	49	82	97	110	113	127	131	135	257
Recursos Contratados	12	21	30	35	40	43	46	50	54	58	90
Mensagens Trocas c/ Recursos	113	176	244	295	332	363	375	407	431	459	713
Mens. Recebidas de Clientes	15	28	50	83	98	111	114	128	132	136	258
Negociações Sem Propostas	0	4	16	44	54	64	64	74	74	74	164
Negociações Sem Resp.Seleção	1	2	3	3	3	3	3	3	3	3	3
Cliente Ficou SemResp.Gerente	0	2	7	15	15	15	15	15	15	15	15
SubContrFinalizadosComSucesso	1	3	4	4	5	5	5	5	6	6	6
SubContrFinalizadosSemSucesso	2	5	8	10	11	12	12	13	13	13	22

Tabela 6-39 Resultado da quinta simulação empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	6	13	19	24	31	37	44	52	58	66	134
Contratos Finalizados Sucesso	3	8	12	16	22	28	33	39	46	54	114
Contratos que Falharam	0	1	2	2	3	3	4	6	6	6	13
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	12	23	34	47	56	70	81	99	109	117	227
Recursos Contratados	10	21	32	44	53	66	77	95	104	112	216
Mensagens Trocas c/ Recursos	96	166	224	298	346	418	475	572	616	651	1177
Mens. Recebidas de Clientes	12	23	34	47	56	70	81	100	109	117	227
Negociações Sem Propostas	0	0	0	1	1	1	1	1	2	2	6
Negociações Sem Resp.Seleção	2	2	2	2	2	3	3	3	3	3	5
Cliente Ficou SemResp.Gerente	2	2	4	5	5	6	6	6	7	7	13
SubContrFinalizadosComSucesso	2	6	7	11	12	16	19	24	26	27	51
SubContrFinalizadosSemSucesso	1	2	3	5	6	8	9	13	13	13	23

Tabela 6-40 Resultado da sexta simulação empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	14	21	26	33	41	47	55	61	69	135
Contratos Finalizados Sucesso	5	10	18	20	28	36	41	49	55	62	129
Contratos que Falharam	0	0	0	0	0	0	0	0	1	1	1
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	13	23	33	43	52	63	72	80	89	97	196
Recursos Contratados	13	22	32	40	49	59	68	76	85	93	187
Mensagens Trocas c/ Recursos	94	142	205	254	299	348	388	429	472	511	959
Mens. Recebidas de Clientes	13	24	33	43	52	63	72	80	89	97	196
Negociações Sem Propostas	0	0	0	0	0	1	1	1	1	1	6
Negociações Sem Resp.Seleção	0	1	1	3	3	3	3	3	3	3	3
Cliente Ficou SemResp.Gerente	0	2	2	4	4	6	6	6	6	6	9
SubContrFinalizadosComSucesso	3	5	8	8	10	12	14	15	17	17	42
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	0	0	1	1	1

Tabela 6-41 Resultado da sétima simulação empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	10	16	24	31	39	46	54	62	69	78	154
Contratos Finalizados Sucesso	8	13	21	28	36	42	50	59	66	75	151
Contratos que Falharam	0	0	0	0	0	0	0	0	0	0	0
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	12	20	28	40	49	56	64	72	79	88	166
Recursos Contratados	12	20	27	37	45	52	60	68	75	84	162
Mensagens Trocas c/ Recursos	67	137	202	255	298	332	371	412	447	492	875
Mens. Recebidas de Clientes	12	20	29	40	49	56	64	72	79	88	166
Negociações Sem Propostas	0	0	0	0	0	0	0	0	0	0	0
Negociações Sem Resp.Seleção	0	0	1	3	4	4	4	4	4	4	4
Cliente Ficou SemResp.Gerente	1	1	2	6	7	7	7	7	7	7	9
SubContrFinalizadosComSucesso	0	1	1	1	1	1	1	1	1	1	1
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	0	0	0	0	0

Tabela 6-42 Resultado da oitava simulação empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	15	22	30	40	46	53	60	67	74	151
Contratos Finalizados Sucesso	7	12	19	27	36	42	50	57	64	71	147
Contratos que Falharam	0	0	0	0	0	0	0	0	0	0	0
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	9	17	25	34	44	51	60	68	77	86	172
Recursos Contratados	9	17	24	32	42	49	56	64	73	82	166
Mensagens Trocas c/ Recursos	84	140	185	231	278	309	344	385	427	472	878
Mens. Recebidas de Clientes	9	17	25	34	45	51	60	68	77	86	172
Negociações Sem Propostas	0	0	0	0	0	0	2	2	2	2	3
Negociações Sem Resp.Seleção	0	0	1	2	2	2	2	2	2	2	3
Cliente Ficou SemResp.Gerente	0	0	2	3	3	3	5	6	6	6	11
SubContrFinalizadosComSucesso	1	1	1	1	2	2	2	3	4	6	11
SubContrFinalizadosSemSucesso	0	0	0	0	0	0	0	0	0	0	0

Tabela 6-43 Resultado da nona simulação empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	14	20	26	34	41	49	56	64	71	149
Contratos Finalizados Sucesso	7	12	17	21	29	37	45	52	60	67	145
Contratos que Falharam	0	0	0	0	0	0	0	0	0	0	0
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	11	20	29	37	45	52	60	71	79	86	168
Recursos Contratados	11	20	29	37	45	52	60	67	75	82	160
Mensagens Trocas c/ Recursos	87	133	174	209	244	278	316	351	390	423	785
Mens. Recebidas de Clientes	11	20	29	37	45	52	60	71	79	86	168
Negociações Sem Propostas	0	0	0	0	0	0	0	4	4	4	8
Negociações Sem Resp.Seleção	0	0	0	0	0	0	0	0	0	0	0
Cliente Ficou SemResp.Gerente	0	0	0	0	0	0	0	4	4	4	8
SubContrFinalizadosComSucesso	2	4	7	8	9	9	9	9	9	9	9
SubContrFinalizadosSemSucesso	0	1	1	1	1	1	1	1	1	1	1

Tabela 6-44 Resultado da décima simulação empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	8	14	21	28	36	44	51	59	67	75	150
Contratos Finalizados Sucesso	6	10	16	24	32	40	47	55	63	71	146
Contratos que Falharam	0	1	1	1	1	1	1	1	1	1	1
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	0
Contratações Negociadas	12	23	31	40	48	57	64	73	81	89	168
Recursos Contratados	12	21	28	37	45	54	61	70	78	86	164
Mensagens Trocas c/ Recursos	99	155	193	232	267	309	343	383	421	460	821
Mens. Recebidas de Clientes	12	23	31	40	48	57	64	73	81	89	168
Negociações Sem Propostas	0	2	2	2	2	2	2	2	2	2	3
Negociações Sem Resp.Seleção	0	0	1	1	1	1	1	1	1	1	1
Cliente Ficou SemResp.Gerente	4	4	5	5	5	6	6	7	7	7	11
SubContrFinalizadosComSucesso	0	0	1	2	2	2	2	2	2	2	2
SubContrFinalizadosSemSucesso	0	1	1	1	1	1	1	1	1	1	1

As tabelas 6-45 e 6-46 apresentam os resultados finais após apurar a média de cada simulação realizada (CNP com subcontratações e aprendizado e com IS>20). Verificou-se que em média 97% das missões são finalizadas e 0% falham nessa configuração.

Tabela 6-45 Resultado final apurando a média das simulações (de 1 a 5) empregando CNP com subcontratações e aprendizado e com IS>20.

Minutos	20	20	20	20	20	**Med**
Número de Clientes Criados	147	149	145	149	151	**148,2**
Contratos Finalizados Sucesso	142	143	143	146	147	**144,2**
Contratos que Falharam	0	0	0	0	0	**0**
Cliente Saiu Sem Resposta	0	0	0	0	0	**0**
Contratações Negociadas	179	155	161	173	164	**166,4**
Recursos Contratados	171	151	157	158	153	**158**
Mensagens Trocas c/ Recursos	867	804	889	881	790	**846,2**
Mens. Recebidas de Clientes	179	155	161	173	164	**166,4**
Negociações Sem Propostas	5	0	2	8	8	**4,6**
Negociações Sem Resp.Seleção	3	4	2	7	3	**3,8**
Cliente Ficou SemResp.Gerente	26	5	89	16	12	**29,6**
SubContrFinalizadosComSucesso	4	1	2	5	1	**2,6**
SubContrFinalizadosSemSucesso	0	11	0	0	0	**2,2**

Tabela 6-46 Resultado final percentual, após apurar a média das simulações (de 1 a 5), empregando CNP com subcontratações e aprendizado e com IS>20.

Resultado Final da Simulação - CNP com Subcontratações e Aprendizado e com IS>20	
Contratos Finalizados com Sucesso	97%
Contratos que Falharam	0%
Cliente Saiu Sem Resposta	0%

As tabelas 6-47 e 6-48 apresentam os resultados finais após apurar a média de cada simulação realizada (CNP com subcontratações e aprendizado e com IS>40). Verificou-se que em média 91% das missões são finalizadas e 5% falham nessa configuração.

Tabela 6-47 Resultado final apurando a média das simulações (de 1 a 10) empregando CNP com subcontratações e aprendizado e com IS>40.

Minutos	20	20	20	20	20	20	20	20	20	20	**Med**
Número de Clientes Criados	142	148	152	71	134	135	154	151	149	150	**129,4**
Contratos Finalizados Sucesso	131	145	147	49	114	129	151	147	145	146	**117,2**
Contratos que Falharam	6	1	0	10	13	1	0	0	0	1	**6**
Cliente Saiu Sem Resposta	0	0	0	0	0	0	0	0	0	0	**0**
Contratações Negociadas	159	186	167	257	227	196	166	172	168	168	**199,2**
Recursos Contratados	154	154	160	90	216	187	162	166	160	164	**154,8**
Mensagens Trocas c/ Recursos	857	857	814	713	1177	959	875	878	785	821	**883,6**
Mens. Recebidas de Clientes	159	186	167	258	227	196	166	172	168	168	**199,4**
Negociações Sem Propostas	0	13	4	164	6	6	0	3	8	3	**37,4**
Negociações Sem Resp.Seleção	5	5	3	3	5	3	4	3	0	1	**4,2**
Cliente Ficou SemResp.Gerente	78	20	11	15	13	9	9	11	8	11	**27,4**
SubContrFinalizadosComSucesso	5	9	3	6	51	42	1	11	9	2	**14,8**
SubContrFinalizadosSemSucesso	6	11	0	22	23	1	0	0	1	1	**12,4**

Tabela 6-48 Resultado final percentual, após apurar a média das simulações (de 1 a 10), empregando CNP com subcontratações e aprendizado e com IS>40.

Resultado Final da Simulação - CNP com Subcontratações e Aprendizado e com IS>40	
Contratos Finalizados com Sucesso	91%
Contratos que Falharam	5%
Cliente Saiu Sem Resposta	0%

As tabelas 6-49 e 6-50 apresentam o resumo dos resultados finais percentuais dos cenários testados. A tabela 6-49 mostra a comparação entre os resultados finais percentuais das simulações considerando IS>20. Já a tabela 6-50 mostra a comparação

entre os resultados finais percentuais das simulações, considerando IS>40. A tabela 6-51 e a figura 6-12 apresentam os resultados percentuais de sucessos e falhas nas missões para IS>20 dos três cenários testados.

Tabela 6-49 Resultado final percentual apurado das simulações utilizando IS>20.

IS >20	CNP Subcontr. e Aprend.	CNP com Subcontratações	CNP Puro
Contr. Finaliz. com Sucesso	97%	90%	85%
Contratos que Falharam	0%	8%	14%
Cliente Saiu Sem Resposta	0%	0%	0%

Tabela 6-50 Resultado final percentual apurado das simulações com IS>40.

IS >40	CNP Subcontr. e Aprend.	CNP com Subcontratações	CNP Puro
Contr. Finaliz. com Sucesso	97%	90%	54%
Contratos que Falharam	1%	7%	44%
Cliente Saiu Sem Resposta	0%	0%	0%

Tabela 6-51 Resultado final percentual de sucessos e de falhas nas missões calculado a cada minuto (a partir da média calculada dos resultados) para simulações com IS>20.

Configuração Testada	Minutos	1	2	3	4	5	6	7	8	9	10	20
CNP	% Sucesso	77%	80%	78%	84%	84%	84%	85%	86%	86%	86%	85%
	% Falha	9%	9%	11%	11%	11%	12%	12%	12%	12%	12%	14%
CNP Subcontract	% Sucesso	80%	82%	81%	85%	86%	87%	88%	88%	89%	89%	90%
	% Falha	0%	0%	2%	4%	4%	4%	5%	6%	6%	6%	8%
CNP Sub. e Compartilhamento	% Sucesso	80%	78%	82%	86%	89%	91%	91%	93%	93%	94%	97%
	% Falha	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

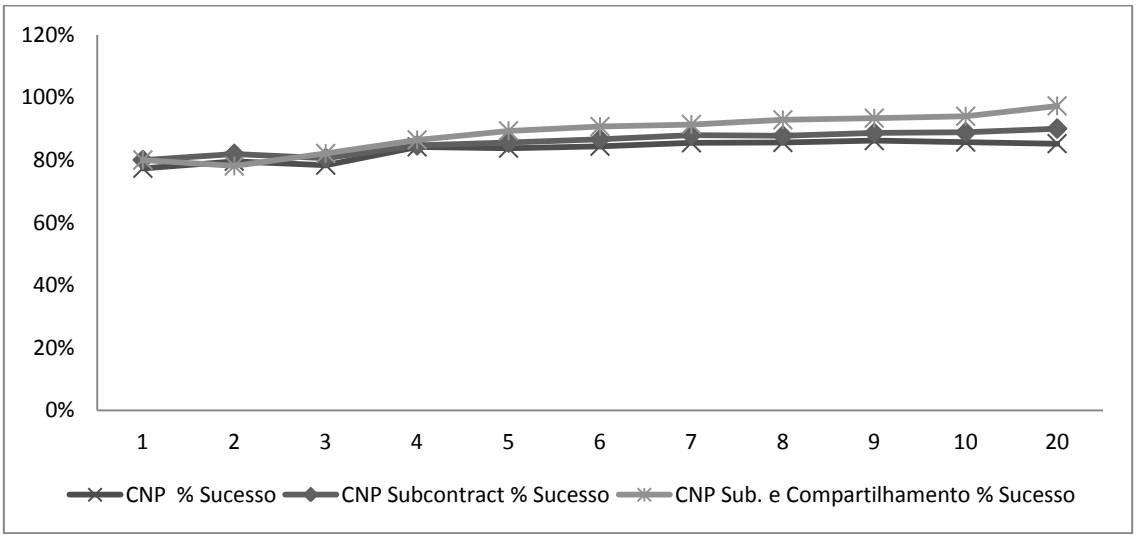

Figura 6-12 Resultado gráfico percentual de sucessos e de falhas nas missões calculado a cada minuto (a partir da média calculada dos resultados) para simulações com IS>20.

A tabela 6-52 e a figura 6-13 apresentam os resultados percentuais de sucessos e falhas nas missões para IS>40 dos três cenários testados.

Tabela 6-52 Resultado final percentual de sucessos e de falhas nas missões calculado a cada minuto (a partir da média calculada dos resultados) para simulações com IS>40.

Configuração Testada	Minutos	1	2	3	4	5	6	7	8	9	10	20
CNP	% Sucesso	57%	51%	49%	51%	53%	54%	53%	53%	54%	53%	54%
	% Falha	25%	38%	41%	42%	41%	42%	44%	44%	43%	44%	44%
CNP Subcontract	% Sucesso	73%	71%	72%	77%	80%	82%	84%	84%	85%	86%	90%
	% Falha	2%	4%	5%	8%	7%	6%	7%	8%	8%	8%	7%
CNP Sub. e Compartilhamento	% Sucesso	72%	76%	78%	80%	84%	85%	87%	88%	90%	91%	94%
	% Falha	3%	3%	4%	4%	4%	4%	3%	3%	3%	3%	2%

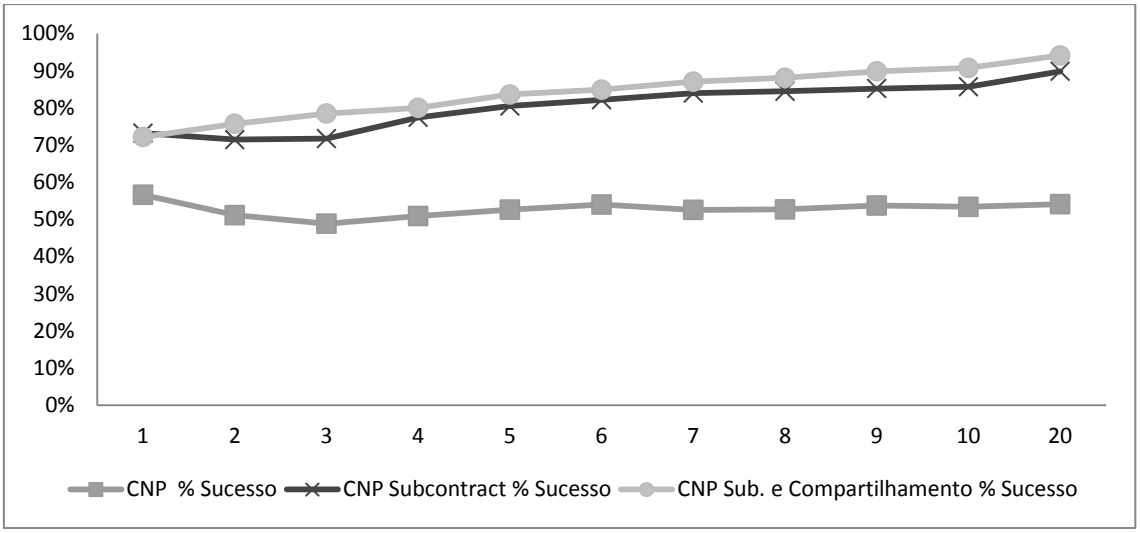

Figura 6-13 Resultado gráfico percentual de sucessos e de falhas nas missões calculado a cada minuto (a partir da média calculada dos resultados) para simulações com IS>40.

Ao final dos ciclos de simulações pôde-se verificar que as subcontratações realizadas por agentes recursos que falharam em suas missões tiveram um impacto considerável nos resultados finais percentuais, proporcionando um aumento no número de contratos finalizados para ambos os cenários testados. A proposta de compartilhamento de aprendizado também contribuiu para melhora e estabilização dos resultados, fazendo com que os agentes do tipo recurso com IS baixo melhorassem ao longo do tempo, após vários ciclos de subcontratações. A abordagem das subcontratações implicou também em um aumento do número de troca de mensagens na plataforma e uma sobrecarga de agentes gerentes em certas situações, o que requer certa atenção na configuração inicial do sistema com número adequado de agentes gerentes para atendimento e de agentes do tipo recurso distribuídos. Se houver vários agentes

recurso com baixo índice de sucesso, como estes temporariamente serão agentes clientes, pode haver escassez de agentes recurso de um determinado tipo, fato que poderá gerar sucessivas e recursivas chamadas por propostas sem respostas de recursos.

A tabela 6-53 apresenta os resultados de um cenário a mais testado. Neste caso foram iniciados dois agentes cliente, cinco agentes gerente e trinta e cinco agentes recurso. Foi considerado o IS>50 para o sucesso em missões por parte dos agentes recurso. Foram realizados cinco ciclos de simulação com a versão CNP com subcontratações e aprendizado. A tabela 6-53 mostra a média das contagens apuradas realizadas em cinco ciclos de simulação realizados.

A tabela 6-54 e a figura 6-14 apresentam os resultados percentuais de sucessos e de falhas nas missões para IS>50 empregando CNP com subcontratações e aprendizado, com configuração inicial de dois agentes cliente, cinco agentes gerente e trinta e cinco agentes recurso.

Tabela 6-53 Resultado final apurando a média das simulações (de 1 a 5) empregando CNP com subcontratações e aprendizado e com IS>50 (configuração inicial de 2 agentes cliente, 5 agentes gerente e 35 agentes recurso)

Minutos	1	2	3	4	5	6	7	8	9	10	20
Número de Clientes Criados	7,6	12,6	18,6	26,4	34,8	42,8	52	61,4	70,6	80,2	167,4
Contratos Finalizados Sucesso	4,6	7,6	12,4	18,8	26,6	33	42	50,8	59,2	68,2	149,8
Contratos que Falharam	0,2	0,2	0,4	0,8	1,2	2	2,8	3,2	3,8	4,2	8,8
Cliente Saiu Sem Resposta	0	0	0,2	0,4	0,6	0,8	1	1	1,2	1,2	2,2
Contratações Negociadas	16,6	28,6	42,4	55,4	68,8	84	97,8	110,2	123,4	135,8	259,4
Recursos Contratados	15	24,2	35,8	46,6	56,2	65,4	75,2	84,6	93,6	103	191,6
Mensagens Trocas c/ Recursos	176,2	266	339,2	399	452	501,6	554	604,2	653,2	702,4	1177
Mens. Recebidas de Clientes	16,6	28,8	42,4	55,6	68,8	84	97,8	110,4	123,6	136	259,4
Negociações Sem Propostas	0,2	2	3	5	8,4	14,2	18,2	21,2	25,4	28,2	61
Negociações Sem Resp.Seleção	1,4	2,4	3,6	3,8	4,2	4,4	4,4	4,4	4,4	4,6	6,8
Cliente Ficou SemResp.Gerente	0,8	4	8,2	11,6	19,6	26,2	30,8	35	40,4	45	91
SubContrFinalizadosComSucesso	1,8	3,6	6	9	10	10,8	11,4	11,6	11,6	11,6	11,6
SubContrFinalizadosSemSucesso	0,4	0,6	0,8	1	1,6	2,4	3	3,6	4,2	4,6	9,2

Tabela 6-54 Resultado final percentual de sucessos e de falhas nas missões calculado a cada minuto para simulação empregando CNP com subcontratações e aprendizado e com IS>50 (configuração inicial de 2 agentes cliente, 5 agentes gerente e 35 agentes recurso)

Configuração Testada	Minutos	1	2	3	4	5	6	7	8	9	10	20
CNP Sub. e Compartilhamento	% Sucesso	61%	60%	67%	71%	76%	77%	81%	83%	84%	85%	89%
	% Falha	3%	2%	2%	3%	3%	5%	5%	5%	5%	5%	5%

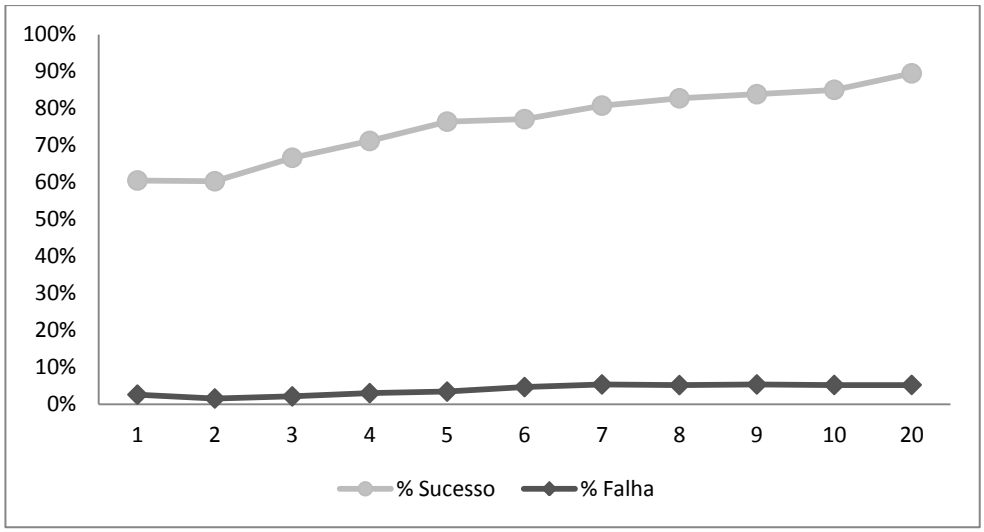

Figura 6-14 Resultado gráfico percentual de sucessos e de falhas nas missões calculado a cada minuto para simulação empregando CNP com subcontratações e aprendizado e com IS>50 (configuração inicial de 2 agentes cliente, 5 agentes gerente e 35 agentes recurso).

Para esse cenário testado, usando IS>50 e configuração inicial de dois agentes cliente, cinco agentes gerente e trinta e cinco agentes recurso, era de se esperar um elevado número de missões com falha, no entanto, o sistema respondeu bem, indo de uma taxa inicial de sucesso em missões de 61% para, ao final de vinte minutos de simulação, um valor de 89%. Observou-se também um elevado número de subcontratações, o que era esperado em função do IS elevado.

Observa-se também que os objetivos estabelecidos para arquitetura proposta foram validados, os quais são:

- A solução permite a alocação automática dos recursos distribuídos disponíveis para execução das missões demandadas;
- O sistema permite o monitoramento das atividades contratadas;
- Há o reparo do plano estabelecido inicialmente sem qualquer interferência por parte do cliente através das subcontratações;
- A arquitetura permite a cooperação entre os agentes distribuídos de modo a não aumentar a sobrecarga de mensagens entre os agentes, também através das subcontratações;
- O protocolo permite que haja um compartilhamento de informações ao fim do processo de subcontratações, sendo este um possível mecanismo para uma aprendizagem distribuída, a fim de permitir a evolução do sistema como um todo;

- Arquitetura é escalável para aplicações empregando robôs móveis (terrestres ou aéreos) e sistemas *Job Shop*.

6.2 Simulações usando Jason sobre JADE

Foram realizadas algumas simulações para avaliação da plataforma Jason usando a infraestrutura JADE. Assim pôde-se também avaliar a plataforma JADE. Foi utilizado um pequeno programa para geração dos três tipos de agentes (descrito no capítulo anterior).

```
MAS cnp {
   infrastructure: Jade
   agents:
      c;    // O agente que inicia o protocolo Contract Net
      p #3; // Os participantes (3) que oferecem os serviços
      pr;   // O participante que sempre recursa os serviços
      pn;   // O participante nunca responde às solicitações de serviços
}
```

A figura a seguir apresenta um cenário de intensa negociação com cinco agentes participantes fazendo propostas para contratar uma tarefa.

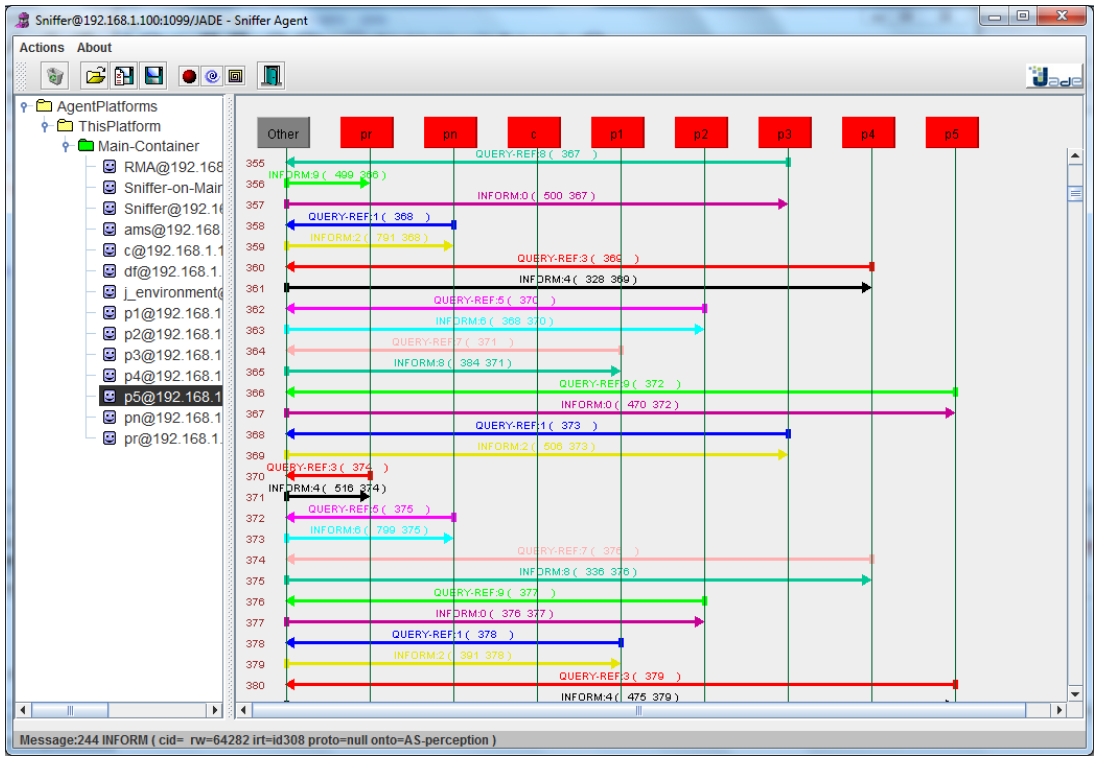

Figura 6-15 Visualização da interação entre os agentes, utilizando o agente Sniffer (da plataforma JADE), com intensa negociação entre cinco agentes participantes.

A figura a seguir mostra quinze agentes participantes negociando com o agente gerente a contratação de uma tarefa.

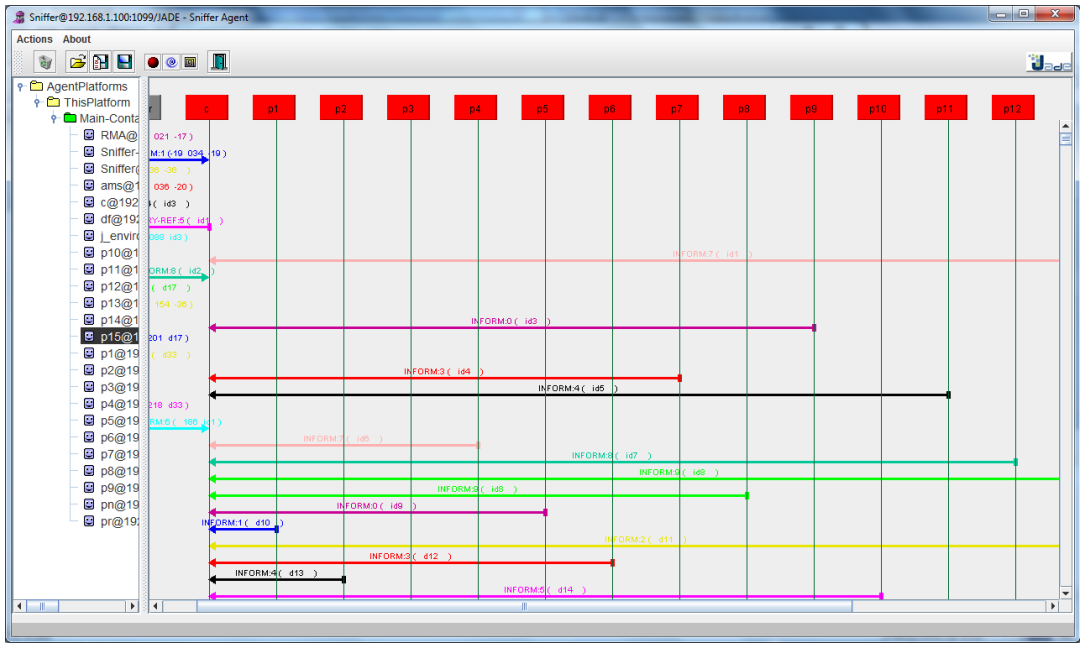

Figura 6-16 Visualização da interação entre quinze agentes participantes.

A partir destas simulações do CNP no ambiente Jason, usando infraestrutura JADE como *middleware* (camada de software intermediária), e utilizando projetos

exemplos disponibilizados, foi possível confirmar a potencialidade desta plataforma no desenvolvimento de sistemas multi-agentes e de agentes autônomos cognitivos. O Jason é uma plataforma para o desenvolvimento sistemas multi-agentes que usa uma extensão da linguagem de programação orientada a agente conhecida como AgentSpeak. Jason é uma plataforma *open source* (código aberto) sendo distribuída sobre licença GNU LGPL.

A linguagem AgentSpeak é uma linguagem de programação orientada a agente, sendo baseada em lógica de programação e arquitetura BDI para agentes autônomos (cognitivos).

A plataforma JADE também apresentou boa robustez, com uma infraestrutura completa para o desenvolvimento de sistemas multi-agentes distribuídos utilizando de linguagem programação Java.

6.3 Simulações com um protótipo usando plataforma JADE

O Netbeans também foi utilizado para desenvolvimento e para simulação do protótipo usando biblioteca JADE. Pode-se observar na figura 6-17 uma imagem do Netbeans com o console de saída executando a aplicação protótipo.

O Agente Gerente é iniciado a partir do Neatbeans passando os parâmetros corretos para inicialização do JADE conforme sua documentação disponível. Os outros agentes podem ser inicializados a partir do JADE Remote Agent Management GUI. A figura 6-18 mostra o JADE Remote Agent Management GUI da plataforma JADE que foi utilizado para iniciar os agentes do protótipo. Um agente cliente, um agente gerente e quatro agentes do tipo recurso foram iniciados, como pode ser visualizado na imagem.

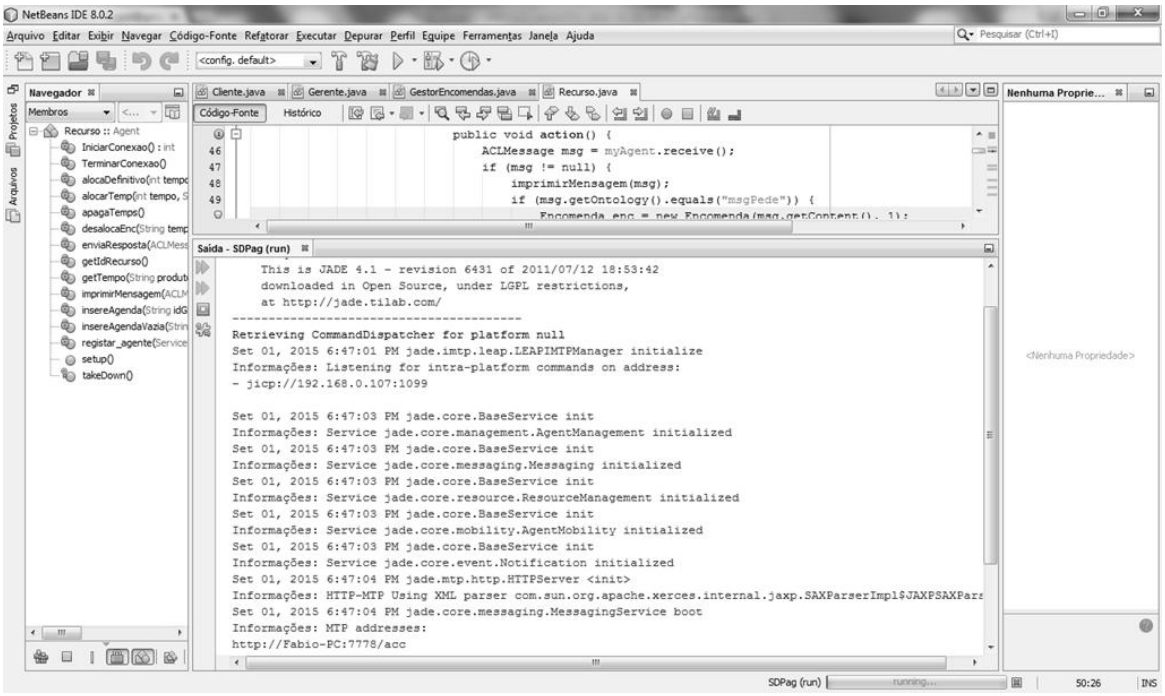

Figura 6-17 Ambiente de desenvolvimento do Netbeans sendo utilizado para executar uma aplicação multi-agente

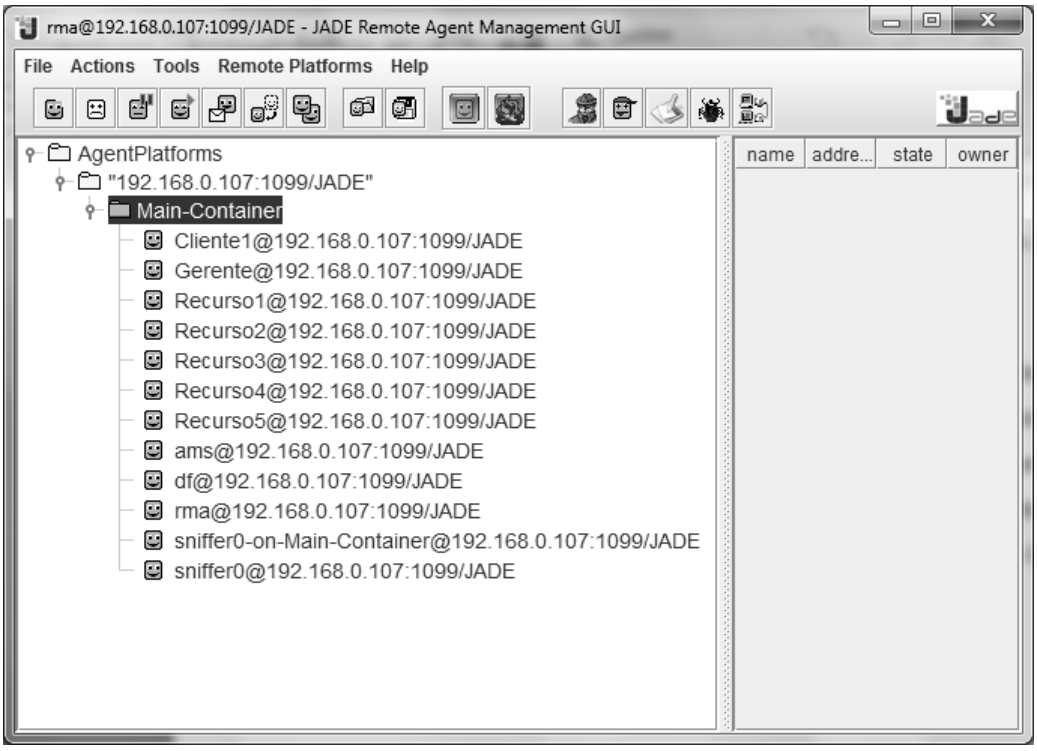

Figura 6-18 JADE Remote Agent Management GUI usando para iniciar os agentes e outros fins.

A figura a seguir ilustra a comunicação intensa para negociação entre os agentes do tipo recurso e um agente gestor de encomenda.

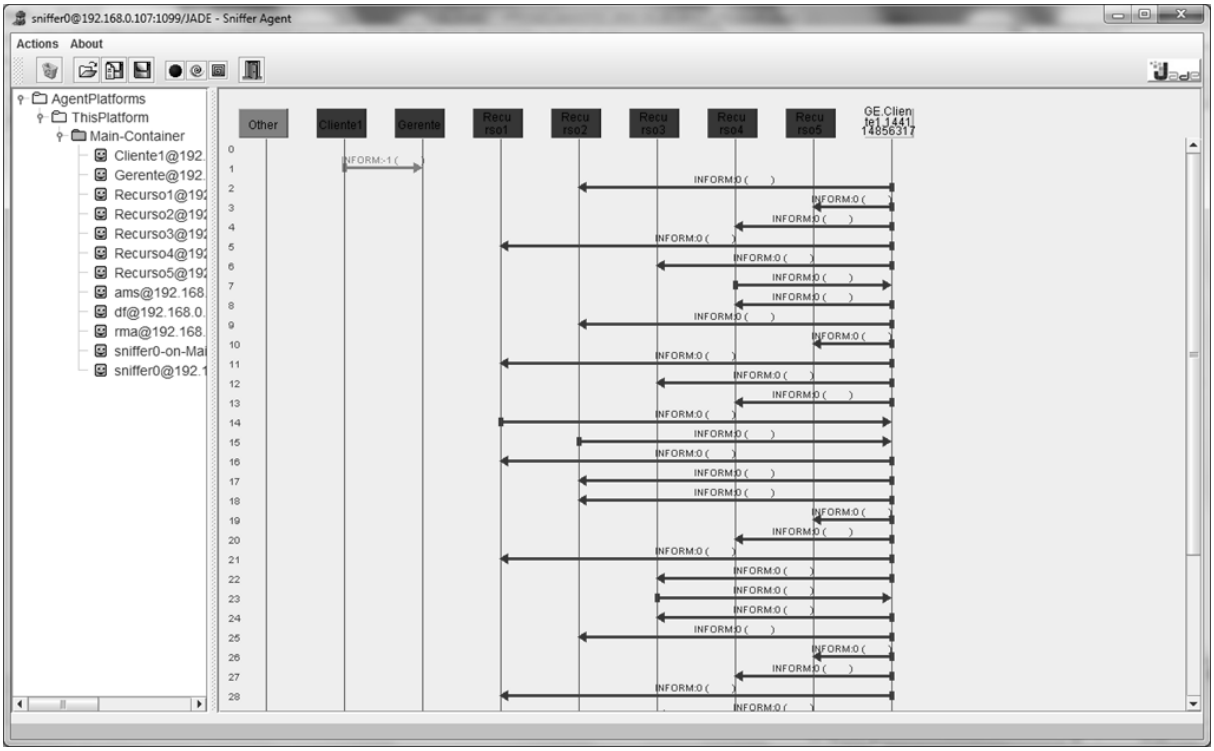

Figura 6-18 Agente Sniffer do JADE usando para monitora a comunicação entre os agentes

Algumas mensagens foram impressas no console de saída para registrar as negociações entre os agentes do sistema.

```
=========RECURSO - Nova Mensagem Recebida========
Emissor: GE.Cliente1.1441148563172
Receptor: Recurso2
Mensagem: <encomenda>
        <nomeCliente>Cliente1</nomeCliente>
        <idEncomenda>Cliente1.1441148562770</idEncomenda>
        <produto>P2.1</produto>
        <quantidade>1</quantidade>
        <tempoMinimoInicio>0</tempoMinimoInicio>
</encomenda>
Tipo de mensagem: msgPede
=========RECURSO - Nova Mensagem Recebida========
Emissor: GE.Cliente1.1441148563172
Receptor: Recurso5
Mensagem: <encomenda>
        <nomeCliente>Cliente1</nomeCliente>
        <idEncomenda>Cliente1.1441148562770</idEncomenda>
        <produto>P2.1</produto>
        <quantidade>1</quantidade>
        <tempoMinimoInicio>0</tempoMinimoInicio>
</encomenda>
Tipo de mensagem: msgPede
================================================
```

O desenvolvimento do protótipo serviu para demonstrar que a plataforma JADE é uma ótima ferramenta para o desenvolvimento de soluções e projetos de sistemas multi-agentes. A plataforma certamente suporta a arquitetura proposta neste trabalhado, sendo indicada para o desenvolvimento de aplicações diversas, incluindo robótica distribuída.

O protótipo permitiu testar o cenário em que uma missão é comporta por outras missões ou uma combinação de operações. O agente gestor de encomenda pôde avaliar o escopo da solicitação que lhe foi feita, identificando o conjunto de operações que faziam parte da encomenda solicitada, negociou com agentes do tipo recurso a alocação das operações e tratou questões de precedências. Assim, pode-se dizer que a encomenda foi decomposta em operações e essas operações foram designadas para os agentes recursos.

Capítulo 7 – Proposta para Trabalhos Futuros

Tendo a compreensão de que esta pesquisa não se esgota neste trabalho, é possível prospectar novas fronteiras a serem desbravadas, fazendo-nos refletir sobre a real contribuição desta pesquisa.

Além inovar propondo uma arquitetura para coordenação robótica distribuída empregando sistemas multi-agentes e uma extensão do protocolo *Contract Net*, com acompanhamento de tarefas e com subcontratações, este trabalho abriu novas janelas de pesquisa, como a oportunidade de se compartilhar conhecimento entre os agentes. Técnicas de aprendizado de máquina, mais especificamente aprendizado por reforço estão cada vez mais sendo empregadas no controle reativo robótico. Os agentes recurso podem compartilhar suas experiências positivas ou suas políticas de ações com os agentes que falharam. Distribuindo assim suas aprendizagens por reforço experimentadas. Essa área é conhecida também como aprendizado por reforço multiagente. Várias estratégias estão sendo pesquisadas, indo da *Whitebox*, que busca máximar os resultados pela ação conjunta dos agentes, até *Blackbox*, que dá maior foco ao aprendizado individual. Recompensas e punições poderiam ser empregadas aos agentes, mas atenção deve ser tomada para não tornar o ambiente competitivo, o que não é foco da arquitetura proposta, que tem como premissas a simplicidade, a colaboração e a descentralização.

O planejamento é uma parte fundamental de sistemas inteligentes. O Planejamento Automático é uma área de pesquisa da IA com foco na geração automática da sequência de ações para resolver um problema. Assim, uma possibilidade que necessitaria ser investiga é o uso de planejadores automáticos como suporte aos agentes do sistema, tanto na camada reativa como na camada deliberativa. Há várias técnicas disponíveis, para dada uma situação inicial, um conjunto de ações e uma situação final desejada, encontrar uma sequência de ações que soluciona o problema. Para o desenvolvimento de entidades inteligentes, como robôs autônomos, temos o desafio de desenvolver softwares que permitam o planejamento efetivo de ações em ambientes dinâmicos.

Outra possibilidade que foi pesquisada neste trabalho, mas que necessita de maior investigação seria o emprego de técnicas de computação evolutiva em alguns agentes da plataforma. A computação evolutiva é o ramo da computação que se baseia no estudo de mecanismos evolutivos encontrados na natureza para desenvolver soluções computacionais. A Robótica Evolutiva trata de metodologias que usam a Computação Evolutiva para desenvolver robôs autônomos. Os controladores robóticos são constituídos na maioria das vezes por redes neurais artificiais e a evolução dos controladores ocorre através da modificação na intensidade das conexões entre os neurônios da rede neural. O projeto de robôs autônomos e seus controladores é uma tarefa complexa e a dificuldade está em prever todas as situações do mundo real. Os Algoritmos Evolucionários têm sido utilizados para desenvolver os sistemas robóticos nestes ambientes dinâmicos, criando assim Robôs Evolucionários. O emprego destas técnicas evolucionárias e o compartilhamento eficiente de parâmetros, como proposto nesta pesquisa, podem contribuir para que um conjunto de robôs possa mapear mais rapidamente uma ambiente e possa prever mais situações do mundo real.

Os sistemas evolucionários, tais como os algoritmos genéticos, são baseados no principio darwiniano de reprodução do mais apto. Os indivíduos mais aptos de cada geração são selecionados para criar a próxima geração até que se tenha um controle efetivo da tarefa.

Sistemas evolutivos podem ser desenvolvidos a partir do modelo proposto através da eliminação de agentes do tipo recurso que não melhorem sua desempenho após vários ciclos de interações de subcontratações, e novos agentes recursos podem ser incluídos no sistema. Pode-se usar abordagem semelhante a empregada por Maturana e Norrie (1999) na arquitetura MetaMorph, na qual agentes mediadores reusam as informações das ações passadas (através da técnica chamada, no inglês, *distributed case-based learning*) e simulam o desempenho antecipadamente antes das tomadas de decisão.

Certa atenção deve ser dada no comportamento reativo dos robôs, neste aspecto um dos assuntos que vem recebendo a atenção de vários pesquisadores é a navegação robótica, visto a importância de robôs chegarem a lugares que humanos não podem ou tem dificuldade, dada alguma periculosidade. O campo da inteligência cognitiva vem atraindo atenção de pesquisadores como uma alternativa para incorporar inteligência a estes robôs. A principal dificuldade se apresenta no desenvolvimento de uma

capacidade do robô em tomar decisões inteligentes quanto à geração de trajetórias frente a perturbações para atingir seu objetivo.

Capítulo 8 – Conclusões

Essa pesquisa aplica conceitos de inteligência artificial no planejamento de operações. Planejamento é um processo explícito de deliberação que escolhe e organiza ações antecipadamente, buscando atingir os objetivos previamente estabelecidos.

Um dos objetivos científicos da inteligência artificial é o entendimento da inteligência e o planejamento é um componente do comportamento racional (inteligente), sendo assim parte da inteligência.

O estudo da inteligência artificial também tem como objetivo a construção de entidades inteligentes. No caso desta pesquisa o objetivo foi o desenvolvimento de arquitetura de software para seleção e organização de ações para máquinas (robôs) autônomas inteligentes.

Como descrito anteriormente, há certa independência entre a concepção dos modelos e o problema, ou seja, não necessariamente os modelos são desenvolvidos para solucionar um problema em particular. Nesta pesquisa buscou-se uma abordagem mais genérica do problema de planejamento (*domain-independent planning*), a fim de comportar casos mais genéricos deste campo de estudo. No entanto, adaptações para domínios mais específicos do planejamento são possíveis (*domain-specific planning*).

A proposição estabelecida inicialmente foi validada com sucesso, ou seja, verificou-se que os agentes obtiveram suas tarefas, através de um sistema distribuído em que as tarefas foram negociadas e foram controladas (monitoradas) num ambiente virtual. Uma vez alocadas aos recursos, as tarefas foram monitoradas na fase de operação. Além disso, o modelo proposto nesta pesquisa pode ser empregado em robótica colaborativa.

Os sistemas multi-agentes são ferramentas muito poderosas, principalmente quando combinados com sistemas multi-robóticos para o tratamento de problemas complexos em que uma abordagem distribuída é possível.

A comunicação entre agentes heterogêneos é uma das áreas mais complexas de sistemas multi-agentes, no entanto, a combinação dos agentes num sistema eleva bastante sua capacidade (especialização).

O desenvolvimento de sistemas multi-agentes pode ser uma tarefa muito complexa. No entanto, há várias ferramentas e plataformas de desenvolvimento para acelerar esse processo.

O emprego de sistemas multi-agentes busca o ganho do coletivo (grupo de agentes homogêneos ou heterogêneos) para realizar tarefas, por vezes complexas para apenas uma entidade, e assim atingir objetivos globais. A maximização de resultados e a flexibilidade podem ser resultantes de comportamentos colaborativos e da distribuição de tarefas entre as entidades do sistema. Há assim a agregação de capacidade, de conhecimento e força de trabalho que contribuem para atingir os objetivos estabelecidos para o grupo.

A utilização de sistemas multiagentes é uma estratégia adequada para manipular problemas de demanda de operações versus recursos distribuídos e compartilhados, permitindo a definição de objetivos, papéis dos agentes e a forma como se relacionam para atingir os objetivos. O uso de uma metodologia para análise é de fundamental importância ao modelar um problema utilizando sistemas multiagentes, pois permite relacionar claramente os objetivos e a estrutura do ambiente, modelando a sociedade de agentes e guiando o pesquisador até seu desenvolvimento.

Em sistemas multi-robóticos equipes de robôs realizam tarefas de forma coordenada através de comunicação implícita, por meio de sensores que capturam informações do ambiente, e/ou comunicação explícita, através da troca de mensagens entre os mesmos.

O modelo de sistema multi-agente proposto nesta pesquisa teve como foco a criação de um processo ágil e eficiente de utilização dos recursos robóticos distribuídos, mas pode ser usado em outras aplicações como em sistemas de produção distribuídos.

A abordagem apresentada busca dar para as unidades robóticas autonomia e capacidade de reação eficiente frente a distúrbios imprevisíveis.

O modelo genérico mostrado nesta pesquisa permite a seleção, a alocação e a operação de recursos distribuídos robóticos, acelerando o processo de planejamento, dando maior visibilidade a operação e robustez frente a distúrbios, através de um processo de reconfiguração do sistema, através de subcontratações.

Os resultados desta pesquisa contribuem para o desenvolvimento de soluções robustas de replanejamento buscando um modelo de execução mais ágil e simples, mais distribuído e dando mesmo importância ao perfeito planejamento e mais a reconfigurações em tempo de execução e a aspectos colaborativos.

Referências Bibliográficas

FIPA (Foundation for Intelligent Physical Agents). (2002, 12 03). *FIPA Abstract Architecture Specification.* Retrieved 02 10, 2013, from http://www.fipa.org/

Artero, A. O. (2009). *Inteligência artificial: teória e prática.* São Paulo: Livraria da Física.

Asama, H.; Matsumoto, A.; Ishida, Y. (1989, Sep 4-6). Design Of An Autonomous And Distributed Robot System: Actress. *Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop*, pp. .283-290.

Beni, G. (1988, Aug). The concept of cellular robotic system. *Intelligent Control, 1988.Proceedings., IEEE International Symposium on*, pp. 57-62.

Bordini, Rafael H., Hübner, Jomi F. . (2007). *Jason: A Java-based interpreter for anextended version of AgentSpeak.*

Caloud, P.; Wonyun Choi; Latombe, J.-C.; Le Pape, C.; Yim, M. (1990, Jul 3-6). Indoor automation with many mobile robots. *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop, 1*, pp. 67-72.

Cao, Y. Uny, Fukunaga, Alex S., Kahng, Andrew B. (1997). Cooperative Mobile Robotics: Antecedents and Directions. (K. Academic, Ed.) *Autonomous Robots*, pp. 7-27.

Daneshfar, F. , Bevrani, H. (2009). Multi-Agent Systems in Control Engineering: A Survey. *Journal of Control Science and Engineering*. doi:10.1155/2009/531080

Foundation for Intelligent Physical Agents. (2004, 03 18). *FIPA Agent Management Specification.* Retrieved 02 10, 2013, from http://www.fipa.org

Heragu, S. S., Graves, R. J., Kim, B.-I., St. Onge, A. (2002). Intelligent agent based framework for manufacturing systems control. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans,*, pp. 560–573.

Hübner, Jomi F., Bordini, Rafael H. . (n.d.). *Jason a Java-based interpreter for an extended version of AgentSpeak*. Retrieved Fevereiro 17, 2013, from Jason: http://jason.sourceforge.net/wp/

Hübner, J. F. (2003). *Um Modelo de Reorganização de Sistemas Multiagentes.* Universidade de São Paulo, Escola Politécnica, São Paulo.

Iñigo-Blasco, P., Río, F.D.D., Romero-Ternero, M.C., Cagigas-Muñiz, D., Diaz, S.V. (2012, June). Robotics software frameworks for multi-agent robotic systems development. *Robotics and Autonomous Systems, 60*(6), pp. 803-821.

kaminka, G. A. (2004). Robots are Agents, Too! . *AgentLink News*, pp. 16-17.

Lesser, V. R. (1999, January). Cooperative Multi-Agent Systems: a personal view of the state of the art. (I. C. Society, Ed.) *IEEE Transactions on Knowledge and Data Engineering, 11*, pp. 133-142.

Maione, G., Naso, D. (2003, December). A soft computing approach for task contracting in multi-agent manufacturing control. *Computers in Industry, 52*, 199 - 219.

Maturana, F., Shen, W., Norrie, D. H. (1999). MetaMorph: an adaptive agent-based architecture for intelligent manufacturing. *International Journal of Production Research*, pp. 2159–2173.

Monostori, L. , Váneza, J., Kumara, S.R.T. (2006). Agent-Based Systems for Manufacturing. *Annals of the CIRP, 55*, pp. 697-719.

Naso, D., Turchiano, B. (2003, October). An improved projection algorithm for direct adaptive fuzzy control. *IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN & CYBERNETICS, 3*, pp. 2126-2131.

Ottoni, Guilherme de Lima, Lagesy, Walter Fette. (2003, Outubro, Novembro e Dezembro). Navegação de Robôs Móveis em Ambientes Desconhecidos. *Revista Controle & Automação*.

Pérez, O. L. (2010). *Uma Arquitetura Mecatrônica de Navegação para Veículos com Reboques Guiados Automaticamente em Ambientes de Sistemas Flexíveis de Manufatura.* Rio de Janeiro: UFRJ/COPPE.

Roberto, O. (2011). *Traffic Management of Automated Guided Vehicles in Flexible Manufacturing Systems.* Università degli Studi di Ferrara.

Shen, W., Wang, L., Hao, Q. (2006). Agent-Based Distributed Manufacturing Process Planning and Scheduling: a state-of-art survey. *IEEE Transactions on Systems, Man and Cybernetics*, pp. 563-571.

Shivanand, H. K., Koti, V., Koti, V. . (2006). *Flexible Manufacturing System.* New Age International (P) Ltd., Publishers.

Sluga, A., Butala, P., Peklenik, J. (2001). Self-organization in a distributed manufacturing system based on constraint logic programming. *CIRP Annals-Manufacturing Technology, 10*(1), pp. 323-326.

Smith, C. S., Wright, P. K. (2001). Cybercut: An Internet-based CAD/CAM System. *ASME Journal of Computing and Information Science in Engineering, 1*, 1-33.

Smith, R. G. (1980, December). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers, C-29*.

Tanenbaum, A. S. (2003). *Computer Networks.* (Fourth ed.). Prentice Hall.ISBN: 0-13-066102-3.

Tharumarajah A., Wells A. J., Nemes L.CSIRO Manufacturing Science & Technology, Preston, Victoria, Australia. (1998). Comparison of Emerging Manufacturing Concepts. *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, (pp. 325 - 331 vol.1).

Tharumarajah, A. W. (1996). Comparison of the Bionic, Fractal and Holonic Manufacturing System Concepts. *International Journal of Computer Integrated Manufacturing*, 9(3): 217-226.

Uhrmacher, Adelinde M. , Weyns, Danny. (2009). *Multi-agent systems: Simulation and applications. Computational Analysis, Synthesis, and Design of Dynamic Models Series.* CRC Press.

Van Brussel, Hendrik., Wyns, Jo., Valckenaers, Pau., Bongaerts, Luc., Peeters, Patrick. (1998). *Reference Architecture for Holonic Manufacturing Systems: PROSA.* Computers in Industry.

Vidoni, R. (2011). Multi Agent Robotic System For Simulation. *ACTA TECHNICA CORVINIENSIS – BULLETIN of ENGINEERING*. Retrieved from http://acta.fih.upt.ro

Vokřínek, J., Bíba, J., Hodík, J., Vybíhal , J., Pěchouček, M. (2007). Competitive Contract Net Protocol. *SOFSEM*, pp. 656-668.

Wooldridge , M., Jennings, N. R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review, 10*(2), 115–152,.

Wooldridge, M. (2001). *Intelligent Agents: The Key Concepts.*

Anexo I – Código Fonte do Protótipo Utilizando Plataforma MaDKit

MarketOrganization.java

```
package madkit.marketorg;

public interface MarketOrganization {

    public static final String COMMUNITY = "fabrica";
    public static final String CLIENTES_GROUP = "fabrica-clientes";
    public static final String RECURSOS_GROUP = "fabrica-recursos";
    public static final String CLIENTE_ROLE = "cliente";
    public static final String GERENTE_ROLE = "gerente";
    public static final String RECURSO_ROLE = "recurso";
}
```

MySchedule.java

```
package madkit.marketorg;

import madkit.kernel.Madkit;
import madkit.kernel.Madkit.Option;
import madkit.gui.ConsoleAgent;
import madkit.kernel.Scheduler;
import java.util.logging.Level;

    @SuppressWarnings("serial")
    public class MySchedule extends Scheduler {

    @Override
            protected void activate() {
                    setDelay(300);
              this.getLogger().createLogFile();
                    setLogLevel(Level.FINE);
                    setSimulationDuration(120);
              setSimulationState(SimulationState.RUNNING);
                    super.activate();
       }

    public static void main(String[] args) {
       Gerente.nbDeClientesCriados = 0;
       Gerente.nbDeGerenteCriados = 0;
       new Madkit(Option.launchAgents.toString(),
             MySchedule.class.getName()+",true;"
             + Gerente.class.getName() + ",true,3;"
             + Cliente.class.getName() + ",true,2;"
             + Recurso.class.getName() + ",true,20;"
             + ConsoleAgent.class.getName());
            };
    }
```

Cliente.java

```
package madkit.marketorg;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Image;
import java.awt.Toolkit;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import madkit.gui.OutputPanel;
import madkit.kernel.Agent;
import madkit.kernel.Madkit;
import madkit.kernel.Madkit.Option;
import madkit.kernel.Message;
import madkit.message.StringMessage;

@SuppressWarnings("serial")
public class Cliente extends Agent {

   static int nbOfClientsOnScreen = 0;

   private JPanel blinkPanel;
   private static ImageIcon clientImage = new ImageIcon(new
ImageIcon(Cliente.class.getResource("images/client.png")).getImage().getScaledInstance(70, 70,
Image.SCALE_SMOOTH));
   private String product = Recurso.availableTransports.get((int) (Math.random() *
Recurso.availableTransports.size()));

   @Override
   protected void activate() {
      Gerente.nbDeClientesCriados++;
      createGroupIfAbsent(MarketOrganization.COMMUNITY, MarketOrganization.CLIENTES_GROUP,
true, null);
      requestRole(MarketOrganization.COMMUNITY, MarketOrganization.CLIENTES_GROUP,
MarketOrganization.CLIENTE_ROLE, null);
      int pause = 1000 + (int) (Math.random() * 2000);

      if (logger != null) {
         logger.info("CLIENTE - Eu nasci e sou " + Cliente.this);
      }
      if (logger != null) {
         logger.info("CLIENTE - Necessito de um drone em " + product + " in " + pause + " ms !");
      }

      pause(pause);
   }
```

```
@Override
protected void live() {
    boolean foiFinalizada = false;
    int tentativasSemRespostas = 0;
    int tentativasComFalhas = 0;
    while (!foiFinalizada) {
        Message respostaGerente = null;
        while (respostaGerente == null) {
            respostaGerente = sendMessageWithRoleAndWaitForReply(
                    MarketOrganization.COMMUNITY,
                    MarketOrganization.CLIENTES_GROUP,
                    MarketOrganization.GERENTE_ROLE,
                    new StringMessage(product),
                    MarketOrganization.CLIENTE_ROLE,
                    1000);
            if (logger != null && respostaGerente == null) {
                logger.info("CLIENTE - Sem resposta de Gerente por enquanto - :(");
                Gerente.noVezQueClienteFicouSemRespostasDeGerentes++;
                tentativasSemRespostas++;
                if (tentativasSemRespostas == 5) {
                    logger.info("CLIENTE - Não fui atendido apos 5 tentativas sem resposta :(");
                    end();
                }
            }
            pause(200);

        }
        logFindGerente(respostaGerente);
        statusDaMissao((StringMessage) respostaGerente);
        foiFinalizada = finalizaMissao((StringMessage) respostaGerente);
        pause(500);
        if (foiFinalizada == false) {
            tentativasComFalhas++;
            if (tentativasComFalhas == 5) {
                logger.info("CLIENTE - Não fui atendido apos 5 falhas - :(");
                end();
            }

        } //else
        //  end();
    }
}
@Override
protected void end() {
    if (logger != null) {
        logger.info("CLIENTE - Estou saindo mais agora, mas  vou chamar outro cliente !");
    }

    pause((int) (Math.random() * 2000 + 5000));
    //launchAgent(new Cliente(),4,true);
    launchAgent(new Cliente(), 2, true);
}
```

```
   private void logFindGerente(Message respostaGerente) {

      if (logger != null) {
         logger.info("CLIENTE - Eu encontrei um Gerente: " + respostaGerente.getSender());
      }
      if (blinkPanel != null) {
         blinkPanel.setBackground(Color.YELLOW);
         pause(2000);
      }
   }



 private boolean finalizaMissao(StringMessage respostaGerente) {
      String contractGroupID = respostaGerente.getContent();
      requestRole(MarketOrganization.COMMUNITY, contractGroupID,
MarketOrganization.CLIENTE_ROLE);
      Message missao = sendMessageAndWaitForReply(MarketOrganization.COMMUNITY,
contractGroupID, MarketOrganization.RECURSO_ROLE, new StringMessage("Missao foi Finalizada?"),
15000);

      if (missao != null) {
         if ((((StringMessage) missao).getContent() == "Done")) {
            if (logger != null) {
               logger.info("CLIENTE - Sim: Missao FINALIZADA :) ");
            }
            if (hasGUI()) {
               blinkPanel.setBackground(Color.GREEN);
            }
            leaveGroup(MarketOrganization.COMMUNITY, MarketOrganization.CLIENTES_GROUP);
            pause((int) (1000 + Math.random() * 2000));
            return true;
         } else if (((StringMessage) missao).getContent() == "Fail") {
            if (logger != null) {
               logger.info("CLIENTE - Missao FALHOU :)  ");
            }
            if (hasGUI()) {
               blinkPanel.setBackground(Color.RED);
            }
            // leaveGroup(MarketOrganization.COMMUNITY, MarketOrganization.CLIENTES_GROUP);
            pause((int) (1000 + Math.random() * 2000));
            return false;
         }
      }
      return false;
   }
```

```
   private void statusDaMissao(StringMessage respostaGerente) {

      String contractGroupID = respostaGerente.getContent();
      requestRole(MarketOrganization.COMMUNITY, contractGroupID,
MarketOrganization.CLIENTE_ROLE);
      StringMessage missao = (StringMessage)
sendMessageAndWaitForReply(MarketOrganization.COMMUNITY, contractGroupID,
MarketOrganization.RECURSO_ROLE, new StringMessage("Status Missao"), 4000);
      if (missao != null) {
         if (logger != null) {
            logger.info("CLIENTE - Missao em Andamento pelo Recurso" + missao.getSender() + " :) - " +
missao.getContent() + " PORCENTO");
         }
         if (hasGUI()) {
            blinkPanel.setBackground(Color.YELLOW);
         }
      }

   }

   @Override
   public void setupFrame(JFrame frame) {
      JPanel p = new JPanel(new BorderLayout());
      p.add(new OutputPanel(this), BorderLayout.CENTER);
      blinkPanel = new JPanel();
      blinkPanel.add(new JLabel(clientImage));
      blinkPanel.add(new JLabel(new ImageIcon(getClass().getResource("images/" + product + ".png"))));
      p.add(blinkPanel, BorderLayout.NORTH);
      p.validate();
      frame.add(p);
      int xLocation = nbOfClientsOnScreen++ * 390;
      if (xLocation + 390 > Toolkit.getDefaultToolkit().getScreenSize().getWidth()) {
         nbOfClientsOnScreen = 0;
      }
      frame.setLocation(xLocation, 0);
      frame.setSize(390, 300);
   }
}
```

Gerente.java

```
package madkit.marketorg;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Image;
import java.awt.Toolkit;
import java.util.Arrays;
import java.util.List;
import java.util.Date;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import madkit.action.AgentAction;
import madkit.gui.OutputPanel;
import madkit.kernel.Agent;
import madkit.kernel.Message;
import madkit.message.IntegerMessage;
import madkit.message.ObjectMessage;
import madkit.message.StringMessage;

/**
 * @author Fabio de Sousa Cardoso
 * @version 1.0
 */
public class Gerente extends Agent
{

	private static final long serialVersionUID = 12179089771001083 96L;
	static int nbDeGerenteCriados=0;
	static int nbDeRecursosCriados=0;
	static int nbDeClientesCriados=0;
	public static Date agora = new java.util.Date();

	static int noDeMensagenTrocasComRecursos=0;
	//Sao as mensagens recebidas pelos recursos que pode ser por varios motivos
	//1. Solicitacao de contrato feita pelo gerente
	//2. Confirmacao de que foi selecionado pelo gerente
	//3. Solicitacao de fechamento do contrato feita pelo cliente
	//4. Compartilhamento de conhecimento feita por outro agente recurso - falta desenvolver
	static int noDeContratosFinalizadosComSucesso=0;
	//Sao contados o numero de contratos realizados com sucesso ou missao concluida pelos recursos
	//incluindo subcontracoes (neste caso as subcontracoe	static int
noDeContratosFinalizadosSemSucesso=0;
	//Sao contados o numero de contratos realizados sem sucesso ou missao nao concluida pelos recursoss
nao sao contadas aqui)
	static int noDeContratosFinalizadosSemSucesso=0;
	//Sao contados o numero de contratos realizados sem sucesso ou missao nao concluida pelos recursos
	//neste caso ate as subcontracoes falharam
	static int noDeSubContratosFinalizadosComSucesso=0;
	//Sao contados o numero de subcontratos realizados com sucesso ou missao concluida pelos recursos
```

```
      static int noDeSubContratosFinalizadosSemSucesso=0;
       //Sao contados o numero de Subcontratos realizados sem sucesso ou missao nao concluida pelos
recursos
       static int noDeRecursosContratadosComSucesso=0;
       //Sao contados o numero de Contratados com sucesso ou selecionados para cada missao
       static int noDeContratosNegociadosComRecursos=0;
       //Sao contados o numero de Contratados Negociacoes que iniciaram (broadcast o cfp - chamada por
proposta)
       static int noDeMensagenRecebidasDeClientes=0;
       //Sao contados o numero de mensagens recebidas pelo gerente vidas dos clientes com pedido
       static int noNegociacoesSemPropostaRecursos=0;
        //Sao contados o numero de vezes que nao houveram proposta para uma dada negociacao
       static int noNegociacoesSemRespostaRecursosAposSelecionado=0;
       //Sao contados o numero de vezes em que recursos foram selecionados mas nao responderam
       static int noVezQueClienteFicouSemRespostasDeGerentes=0;
       //Sao contados o numero de vezes em que o cliente ficou sem resposta do gerente

            private static ImageIcon gerenteImage= new ImageIcon(new
ImageIcon(Gerente.class.getResource("images/gerente.png")).getImage().getScaledInstance(70, 70,
Image.SCALE_SMOOTH));
            private JPanel blinkPanel;

            protected void activate() //Gerente é iniciado
            {
            createGroupIfAbsent(MarketOrganization.COMMUNITY,MarketOrganization.CLIENTES_GRO
UP,true,null);
            createGroupIfAbsent(MarketOrganization.COMMUNITY,MarketOrganization.RECURSOS_GRO
UP,true,null);
            requestRole(MarketOrganization.COMMUNITY,MarketOrganization.CLIENTES_GROUP,
MarketOrganization.GERENTE_ROLE,null);
            requestRole(MarketOrganization.COMMUNITY,MarketOrganization.RECURSOS_GROUP,
MarketOrganization.GERENTE_ROLE,null);
                  if(logger != null)
                              logger.info("GERENTE - Eu nasci e sou " + Gerente.this);
          }
            protected void live() //Gerente em estado ativo
            {
                        while (true)
                        {
                                    Message m = purgeMailbox();//para sempre tratar a ultima requisiçao
                                    if (m == null) {
                         m = waitNextMessage();//Espera uma messagem
                                    }
                                    String role = m.getSender().getRole();
                                    if(role.equals(MarketOrganization.CLIENTE_ROLE)) {
                           noDeMensagenRecebidasDeClientes++;
                                    handleClienteRequest((StringMessage) m);
                                    }
                        }
            }

            @Override
            protected void end() { //Gerente sendo finalizado
                        AgentAction.RELOAD.getActionFor(this).actionPerformed(null);
            }
```

```
private void handleClienteRequest(StringMessage request) {
      if (!checkAgentAddress(request.getSender())) //o Cliente ainda esta lá?
      {
         return;
      }
      if (hasGUI()) { // iniciando o contract net protocol
         blinkPanel.setBackground(Color.YELLOW);
      }
      if (logger != null) {
         logger.info("GERENTE - Eu recebi um requisicao para " + request.getContent() + " do " +
request.getSender());
      }
      List<Message> propostas = broadcastMessageWithRoleAndWaitForReplies(//espera todas as respostas
            MarketOrganization.COMMUNITY, //community
            MarketOrganization.RECURSOS_GROUP, //group
            request.getContent() + "-" + MarketOrganization.RECURSO_ROLE, //role
            new StringMessage("fazer-proposta-por-favor"), //pergunda por uma proposta
            MarketOrganization.GERENTE_ROLE, //Eu sou um gerente
            900); // Posso esperar um tempo

      noDeContratosNegociadosComRecursos++;

      if (propostas == null) { // no reply

         noNegociacoesSemPropostaRecursos++;

         if (logger != null) {
            logger.info("GERENTE - Sem propostas por enquanto: Ninguem esta disponivel - " +
request.getContent().toUpperCase() + " !! Precisamos de novamos recursos !a");
         }
         if (hasGUI()) {
            blinkPanel.setBackground(Color.LIGHT_GRAY);
         }
         return;
      }

      // seleciona melhor oferta
      final List<IntegerMessage> offers = Arrays.asList(propostas.toArray(new
IntegerMessage[0]));//casting
      IntegerMessage best = ObjectMessage.min(offers);
      if (logger != null) {
         logger.info("GERENTE - A melhor oferta veio de" + best.getSender() + " " + best.getContent());
      }

      String contractGroupId = "" + System.nanoTime();  // envia o contrato para recurso
      Message ack = sendMessageWithRoleAndWaitForReply(
            best.getSender(),
            new StringMessage(contractGroupId),
            MarketOrganization.GERENTE_ROLE,
            1000); /
```

```
        if (ack != null) {
            if (logger != null) {
                logger.info("GERENTE - O Recurso foi contratado! Enviando o número do contrato para o
Cliente - o solicitante foi" + request.getSender());
            }
            sendReply(request, new StringMessage(contractGroupId));
            noDeRecursosContratadosComSucesso++;

        } else {
            noNegociacoesSemRespostaRecursosAposSelecionado++;
            if (logger != null) {
                logger.info("GERENTE - Recurso desapareceu, sem repostas !!!!");
            }
        }
        if (hasGUI()) {
            blinkPanel.setBackground(Color.LIGHT_GRAY);
        }
        if (logger != null) {
            logger.info("GERENTE - Estatistica de " + Gerente.this + ":\n"
        + "     - noDeGerentesCriados = " + nbDeGerenteCriados
        + "\n     - noDeRecursosCriados = " + nbDeRecursosCriados
        + "\n     - noDeClientesCriados = " + nbDeClientesCriados
        + "\n     - noDeContratosFinalizadosComSucesso = " + noDeContratosFinalizadosComSucesso
         + "\n     - noDeRecursosContratadosComSucesso = " + noDeRecursosContratadosComSucesso
         + "\n     - noDeMensagensTrocasComRecursos = " + noDeMensagenTrocasComRecursos
         + "\n     - noDeMensagensRecebidasDeClientes = " + noDeMensagenRecebidasDeClientes
         + "\n     - noDeContratosNegociadosRecursos = " + noDeContratosNegociadosComRecursos
         + "\n     - noNegociacoesSemPropostaRecursos = " + noNegociacoesSemPropostaRecursos
         + "\n     - noNegociacoesSemRespostaRecursosAposSelecionado = " +
noNegociacoesSemRespostaRecursosAposSelecionado
+ "\n  - noVezQueClienteFicouSemRespostasDeGerentes = " +
noVezQueClienteFicouSemRespostasDeGerentes
        + "\n - noDeContratosFinalizadosSemSucesso(Falha)  = " + noDeContratosFinalizadosSemSucesso
        + "\n  - noDeSubContratosFinalizadosComSucesso  = " + noDeSubContratosFinalizadosComSucesso
        + "\n - noDeSubContratosFinalizadosSemSucesso  = " + noDeSubContratosFinalizadosSemSucesso);
            logger.info("GERENTE -- " + agora.toString());
        }
    }
            @Override
            public void setupFrame(JFrame frame) {
                    JPanel p = new JPanel(new BorderLayout());
                    //customizing but still using the OutputPanel from MaDKit GUI
                    p.add(new OutputPanel(this),BorderLayout.CENTER);
                    blinkPanel = new JPanel();
                    blinkPanel.add(new JLabel(gerenteImage));
                    p.add(blinkPanel,BorderLayout.NORTH);
                    blinkPanel.setBackground(Color.LIGHT_GRAY);
                    p.validate();
                    frame.add(p);
                    int xLocation = nbDeGerenteCriados++*390;
                    if(xLocation+390 > Toolkit.getDefaultToolkit().getScreenSize().getWidth())
                            nbDeGerenteCriados=0;
                    frame.setLocation(xLocation, 320);
                    frame.setSize(390, 300);
            }
}
```

Recurso.java

```
package madkit.marketorg;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Toolkit;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Date;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import madkit.gui.OutputPanel;
import madkit.kernel.Agent;
import madkit.kernel.Message;
import madkit.message.IntegerMessage;
import madkit.message.ObjectMessage;
import madkit.message.StringMessage;

/**
 * @author Fabio de Sousa Cardoso, Prof. Max Suell Dutra
 * @version 1.0
 */
public class Recurso extends Agent {

    private static final long serialVersionUID = 9125493540160734521L;
    public static List<String> availableTransports = Arrays.asList("drone1", "drone2", "robot3");
    private int indiceSucesso = ((int) (Math.random() * 100));  // indice de sucesso de 20%
    public static Date agora = new java.util.Date();
    final private static Map<String, ImageIcon> icons = new HashMap<String, ImageIcon>();

    static {
        for (String competence : availableTransports) {
            icons.put(competence, new ImageIcon(Recurso.class.getResource("images/" + competence +
".png")));
        }
    }
    static private int nbOfRecursosNaTela = 0;
    private String competence;
    private JPanel blinkPanel;

    private int noDeContratosFinalizadosComSucessoPorMim = 0;
    //Sao contados o numero de subcontratos realizados com sucesso ou missao concluida pelos recursos
    private int noDeContratosFinalizadosSemSucessoPorMim = 0;
    //Sao contados o numero de Subcontratos realizados sem sucesso ou missao nao concluida pelos recursos
    private int noDeSubContratosFinalizadosComSucessoGerenciadosPorMim = 0;
    //Sao contados o numero de subcontratos realizados com sucesso ou missao concluida pelos recursos
    private int noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim = 0;
    //Sao contados o numero de Subcontratos realizados sem sucesso ou missao nao concluida pelos recursos
```

```
  public Recurso() {
      competence = Recurso.availableTransports.get((int) (Math.random() *
Recurso.availableTransports.size()));
   }

   public void activate() {

      Gerente.nbDeRecursosCriados++;
      createGroupIfAbsent(MarketOrganization.COMMUNITY,
MarketOrganization.RECURSOS_GROUP, true, null);
      requestRole(MarketOrganization.COMMUNITY, MarketOrganization.RECURSOS_GROUP,
competence + "-" + MarketOrganization.RECURSO_ROLE, null);
      if (logger != null) {
         logger.info("RECURSO - Eu nasci - Minha competência é " + competence + " e sou " +
Recurso.this);
         logger.info("RECURSO - Valor - Indice de Sucesso " + indiceSucesso);
      }
   }

   public void live() {
      while (true) {
         Message m = waitNextMessage();
         StringMessage Sm = null;
         Gerente.noDeMensagenTrocasComRecursos++;
         Sm = (StringMessage) m;

         if ((m.getSender().getRole().equals(MarketOrganization.GERENTE_ROLE))) {
            handleGerenteMessage((StringMessage) m);
         } else if ((Sm = (StringMessage) m).getContent().equals("Status Missao")) {
            passaStatus(Sm);
         } else if ((Sm = (StringMessage) m).getContent().equals("Quero Aprender")) {
            passaAprendizado(Sm);
         } else {
            finalizeContract(Sm);
         }
      }
   }

   private void handleGerenteMessage(StringMessage m) {

      if (m.getContent().equals("fazer-proposta-por-favor")) {
         if (logger != null) {
            logger.info("RECURSO - Eu recebi uma chamada de proposta de " + m.getSender());
         }
         sendReply(m, new IntegerMessage((int) (Math.random() * 500)));
      } else {
         iHaveBeenSelected(m);
      }
   }
```

```
  private void iHaveBeenSelected(StringMessage m) {
      if (hasGUI()) {
         blinkPanel.setBackground(Color.YELLOW);
      }
      if (logger != null) {
         logger.info("RECURSO - Eu fui selecionado para missão :)");
      }
      String contractGroup = m.getContent();
      createGroup(MarketOrganization.COMMUNITY, contractGroup, true);
      requestRole(MarketOrganization.COMMUNITY, contractGroup,
MarketOrganization.RECURSO_ROLE);

      if (logger != null) {
         logger.info("RECURSO - Este é o No do Contrato= " + contractGroup);
      }

      sendReply(m, new Message()); // just an acknowledgment
   }

   private void finalizeContract(StringMessage m) {

      boolean subContractFinalizou = false;
      if (indiceSucesso > 20) { // indice de Falha de 20% - Assim se indiceFalha>20 nao houve falha
         if (hasGUI()) {
            blinkPanel.setBackground(Color.GREEN);
         }
         if (logger != null) {
            logger.info("RECURSO - Eu - " + Recurso.this + " competência -  " + competence + " realizei a
missao ! - sendReply para " + m.getSender());
         }

         sendReply(m, new StringMessage("Done"));
         pause((int) (Math.random() * 2000 + 1000));

         leaveGroup(MarketOrganization.COMMUNITY, m.getSender().getGroup());

         if (hasGUI()) {
            blinkPanel.setBackground(Color.LIGHT_GRAY);
         }
         Gerente.noDeContratosFinalizadosComSucesso++;
         noDeContratosFinalizadosComSucessoPorMim++;

      } else {
         if (hasGUI()) {
            blinkPanel.setBackground(Color.RED);
         }
         noDeContratosFinalizadosSemSucessoPorMim++;

         subContractFinalizou = subcontratacao();
         if (subContractFinalizou) {
            sendReply(m, new StringMessage("Done"));
            pause((int) (Math.random() * 2000 + 1000));//let us celebrate !!
            leaveGroup(MarketOrganization.COMMUNITY, m.getSender().getGroup());
            if (hasGUI()) {
               blinkPanel.setBackground(Color.GREEN);
            }
```

```
            Gerente.noDeSubContratosFinalizadosComSucesso++;
            Gerente.noDeContratosFinalizadosComSucesso++;
            noDeSubContratosFinalizadosComSucessoGerenciadosPorMim++;
         } else {
            if (logger != null) {
               logger.info("RECURSO - Eu - " + Recurso.this + " competência -  " + competence + " FALHEI
Na missao ! - sendReply para " + m.getSender());
            }
            sendReply(m, new StringMessage("Fail"));
            pause((int) (Math.random() * 2000 + 1000));//let us celebrate !!
            leaveGroup(MarketOrganization.COMMUNITY, m.getSender().getGroup());
            if (hasGUI()) {
               blinkPanel.setBackground(Color.RED);
            }
            Gerente.noDeSubContratosFinalizadosSemSucesso++;
            noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim++;
         }
      }
      logger.info("RECURSO - " + Recurso.this + " com competência de " + competence + "- Estatisticas:\n"
            + "     - noDeContratosFinalizadosComSucessoPorMim = " +
noDeContratosFinalizadosComSucessoPorMim
            + "\n     - noDeContratosFinalizadosSemSucessoPorMim = " +
noDeContratosFinalizadosSemSucessoPorMim
            + "\n     - noDeSubContratosFinalizadosComSucessoGerenciadosPorMim = " +
noDeSubContratosFinalizadosComSucessoGerenciadosPorMim
            + "\n     - noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim = " +
noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim);
      logger.info("RECURSO - " + Recurso.this + " com competência de " + competence + "- meu índice de
Acertos atual é :" + indiceSucesso);
      logger.info("RECURSO - " + agora.toString());
   }

   private void passaStatus(StringMessage m) {
      int status = ((int) (Math.random() * 100));  // indice de Falha de 20%
      if (hasGUI()) {
         blinkPanel.setBackground(Color.YELLOW);
      }
      if (logger != null) {
         logger.info("RECURSO - Eu - " + Recurso.this + " competência -  " + competence + " realizei " +
status + " PORCENTO - STATUS - Em Andamento");
      }
      sendReply(m, new StringMessage(String.valueOf(status)));
      pause((int) (Math.random() * 2000 + 1000));//let us celebrate !!
   }
```

```
  private void passaAprendizado(StringMessage m) {
      if (hasGUI()) {
         blinkPanel.setBackground(Color.BLACK);
      }
      if (logger != null) {
         logger.info("RECURSO - Eu - " + Recurso.this + " competência -  " + competence + " VOU
COMPARTILHAR ENSINAMENTO");
         logger.info("RECURSO - Eu - VOU COMPARTILHAR ENSINAMENTO para getSender = " +
m.getSender() + " getSender().getRole() = " + m.getSender().getRole());
      }
      sendReplyWithRole(m, new StringMessage("Compartilhado"), m.getSender().getRole());
   }
   private boolean subcontratacao() {
      createGroupIfAbsent(MarketOrganization.COMMUNITY, MarketOrganization.CLIENTES_GROUP,
true, null);
      requestRole(MarketOrganization.COMMUNITY, MarketOrganization.CLIENTES_GROUP,
MarketOrganization.CLIENTE_ROLE, null);
      boolean missaoFinalizada = false;
      int tentativasSemRespostas = 0;
      int tentativasComFalhas = 0;
      while (!missaoFinalizada) {
         Message respostaGerente = null;
         while (respostaGerente == null) {
            respostaGerente = sendMessageWithRoleAndWaitForReply(
                  MarketOrganization.COMMUNITY,
                  MarketOrganization.CLIENTES_GROUP,
                  MarketOrganization.GERENTE_ROLE,
                  new StringMessage(competence),
                  MarketOrganization.CLIENTE_ROLE, 1000);
            if (logger != null && respostaGerente == null) {
               logger.info("RECURSO - CLIENTE - Sem resposta de Gerente por enquanto - :(");
               tentativasSemRespostas++;
               if (tentativasSemRespostas == 5) {
                  logger.info("RECURSO - CLIENTE – Não fui atendido após 5 tentativas sem resposta  :(");
                  return false;
               }
            }
         }
         logFindGerente(respostaGerente);
         missaoFinalizada = finalizaMissao((StringMessage) respostaGerente);
         if (missaoFinalizada) {
            createGroupIfAbsent(MarketOrganization.COMMUNITY,
MarketOrganization.RECURSOS_GROUP, true, null);
            requestRole(MarketOrganization.COMMUNITY, MarketOrganization.RECURSOS_GROUP,
competence + "-" + MarketOrganization.RECURSO_ROLE, null);
            return true;
         } else {
            tentativasComFalhas++;
            if (tentativasComFalhas == 5) {
               logger.info("RECURSO - CLIENTE - NAO FUI ATENTIDO APOS 5 FALHAS - :(");
               return false;
            }
         }
      }   return false;
   }
```

```
  private void logFindGerente(Message respostaGerente) {
      if (logger != null) {
         logger.info("RECURSO - CLIENTE - Eu encontrei um Gerente: " + respostaGerente.getSender());
      }
      if (blinkPanel != null) {
         blinkPanel.setBackground(Color.YELLOW);
         pause(2000);
      }
   }

   private boolean finalizaMissao(StringMessage respostaGerente) {
      String contractGroupID = respostaGerente.getContent();
      requestRole(MarketOrganization.COMMUNITY, contractGroupID,
MarketOrganization.CLIENTE_ROLE);

      Message missao = sendMessageAndWaitForReply(MarketOrganization.COMMUNITY,
contractGroupID, MarketOrganization.RECURSO_ROLE, new StringMessage("Missao foi Finalizada?"),
4000);
      pause((int) (1000 + Math.random() * 2000));
      if (missao != null) {

         if (((StringMessage) missao).getContent().equals("Done")) {

            indiceSucesso = indiceSucesso + 10; // Indice de Sucesso foi aumentado simulado um
aprendizado

            if (logger != null) {
               logger.info("RECURSO - CLIENTE - Sim: Missao FINALIZADA :) com
COMPARTILHAMENTO DE APRENDIZADO - indiceSucesso = " + indiceSucesso);
            }
            if (hasGUI()) {
               blinkPanel.setBackground(Color.GREEN);
            }
            Message aprendizagem = sendMessageAndWaitForReply(MarketOrganization.COMMUNITY,
contractGroupID, MarketOrganization.RECURSO_ROLE, new StringMessage("Quero Aprender"), 4000);
            pause((int) (1000 + Math.random() * 2000));
            if (aprendizagem != null) {
               if (((StringMessage) aprendizagem).getContent().equals("Compartilhado")) {
                  indiceSucesso = indiceSucesso + 10; // Indice de Sucesso foi aumentado simulando um
aprendizado
                  if (logger != null) {
                     logger.info("RECURSO - CLIENTE - Missao FINALIZADA - APRENDIZADO
COMPARTILHADO  - indiceSucesso = " + indiceSucesso);
                  }

               } else if (((StringMessage) aprendizagem).getContent().equals("Nao Compartilhado")) {
                  if (logger != null) {
                     logger.info("RECURSO - CLIENTE - Missao FINALIZADA - Aprendizado NAO
Compatilhado  - indiceSucesso = " + indiceSucesso);
                  }
               }
```

```
            } else {
                logger.info("RECURSO - CLIENTE - Missao FINALIZADA - Mensagem de aprendizagem  -
NULL = " + aprendizagem);
            }
            leaveGroup(MarketOrganization.COMMUNITY, MarketOrganization.CLIENTES_GROUP);
            pause((int) (1000 + Math.random() * 2000));
            return true;
         } else if (((StringMessage) missao).getContent().equals("Fail")) {
            if (logger != null) {
               logger.info("RECURSO - CLIENTE - Missao FALHOU :)  ");
            }
            if (hasGUI()) {
               blinkPanel.setBackground(Color.RED);
            }
            leaveGroup(MarketOrganization.COMMUNITY, MarketOrganization.CLIENTES_GROUP);
            pause((int) (1000 + Math.random() * 2000));
            return false;
         }
      }
      return false;
   }

   @Override
   public void setupFrame(JFrame frame) {
      JPanel p = new JPanel(new BorderLayout());
      p.add(new OutputPanel(this), BorderLayout.CENTER);
      blinkPanel = new JPanel();
      blinkPanel.add(new JLabel(icons.get(competence)));
      p.add(blinkPanel, BorderLayout.NORTH);
      blinkPanel.setBackground(Color.LIGHT_GRAY);
      p.validate();
      frame.add(p);
      int xLocation = nbOfRecursosNaTela++ * 300;
      if (xLocation + 300 > Toolkit.getDefaultToolkit().getScreenSize().getWidth()) {
         nbOfRecursosNaTela = 0;
      }
      frame.setLocation(xLocation, 640);
      frame.setSize(300, 300);
   }

}
```

Anexo II – Arquivo gerado a partir de uma simulação de 44 segundos realizada.

```
                ----------------------------------------
                MaDKit
              version: 5.0.5.2
            build-id: 20150430-2053
           MaDKit Team (c) 1997-2015
         ---------------------------------------

[Gerente-3] Informações : GERENTE - Eu nasci e sou Gerente-3 (ACTIVATED)
[Gerente-5] Informações : GERENTE - Eu nasci e sou Gerente-5 (ACTIVATED)
[Gerente-4] Informações : GERENTE - Eu nasci e sou Gerente-4 (ACTIVATED)
[Cliente-13] Informações : CLIENTE - Eu nasci e sou Cliente-13 (ACTIVATED)
[Cliente-13] Informações : CLIENTE - Necessito de um drone em drone2 in 1049 ms !
[Recurso-7] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-7 (ACTIVATED)
[Recurso-7] Informações : RECURSO - Valor - Indice de Sucesso 85
[Recurso-12] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-12 (ACTIVATED)
[Recurso-12] Informações : RECURSO - Valor - Indice de Sucesso 75
[Recurso-10] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-10 (ACTIVATED)
[Recurso-10] Informações : RECURSO - Valor - Indice de Sucesso 87
[Cliente-6] Informações : CLIENTE - Eu nasci e sou Cliente-6 (ACTIVATED)
[Cliente-6] Informações : CLIENTE - Necessito de um drone em drone1 in 2019 ms !
[Recurso-8] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-8 (ACTIVATED)
[Recurso-8] Informações : RECURSO - Valor - Indice de Sucesso 54
[Recurso-9] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-9 (ACTIVATED)
[Recurso-9] Informações : RECURSO - Valor - Indice de Sucesso 18
[Recurso-14] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-14 (ACTIVATED)
[Recurso-14] Informações : RECURSO - Valor - Indice de Sucesso 49
[Recurso-21] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-21 (ACTIVATED)
[Recurso-21] Informações : RECURSO - Valor - Indice de Sucesso 85
[Recurso-17] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-17 (ACTIVATED)
[Recurso-17] Informações : RECURSO - Valor - Indice de Sucesso 54
[Recurso-15] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-15 (ACTIVATED)
[Recurso-15] Informações : RECURSO - Valor - Indice de Sucesso 51
[Gerente-3] Informações : GERENTE - Eu recebi um requisicao para drone2 do 13@(fabrica,fabrica-clientes,cliente)@MK-479
[Recurso-14] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-10] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Gerente-3] Informações : GERENTE - A melhor oferta veio de10@(fabrica,fabrica-recursos,drone2-recurso)@MK-479 100
[Recurso-10] Informações : RECURSO - Eu fui selecionado para missão :)
[Recurso-10] Informações : RECURSO - Este é o No do Contrato= 55721519508751
[Gerente-3] Informações : GERENTE - O Recurso foi contratado! Enviando o número do contrato para o Cliente - o solicitante
foi13@(fabrica,fabrica-clientes,cliente)@MK-479
[Gerente-3] Informações : GERENTE - Estatistica de Gerente-3 (LIVING):
    - noDeGerentesCriados = 3
    - noDeRecursosCriados = 9
    - noDeClientesCriados = 2

    - noDeContratosFinalizadosComSucesso (Done) = 0
    - noDeContratosFinalizadosSemSucesso (Falha)  = 0
    - noVezQueClienteFicouSemRespostasDeGerentesSaiuSemAtendimento (Sem Atendimento)  = 0

    - noDeContratosNegociadosRecursos = 1
    - noDeRecursosContratadosComSucesso = 1
```

```
  - noDeMensagensTrocasComRecursos = 3
    - noDeMensagensRecebidasDeClientes = 1
    - noNegociacoesSemPropostaRecursos              = 0
    - noNegociacoesSemRespostaRecursosAposSelecionado = 0
    - noVezQueClienteFicouSemRespostasDeGerentes      = 0
    - noDeSubContratosFinalizadosComSucesso  = 0
    - noDeSubContratosFinalizadosSemSucesso  = 0
[Gerente-3] Informações :

GERENTE - 31/08/2015 13:12:11

[Recurso-22] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-22 (ACTIVATED)
[Recurso-22] Informações : RECURSO - Valor - Indice de Sucesso 81
[Recurso-23] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-23 (ACTIVATED)
[Recurso-23] Informações : RECURSO - Valor - Indice de Sucesso 13
[Recurso-20] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-20 (ACTIVATED)
[Recurso-20] Informações : RECURSO - Valor - Indice de Sucesso 66
[Recurso-18] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-18 (ACTIVATED)
[Recurso-18] Informações : RECURSO - Valor - Indice de Sucesso 55
[Cliente-13] Informações : CLIENTE - Eu encontrei um Gerente: 3@(fabrica,fabrica-clientes,manager)@MK-479
[Recurso-16] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-16 (ACTIVATED)
[Recurso-16] Informações : RECURSO - Valor - Indice de Sucesso 13
[Recurso-26] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-26 (ACTIVATED)
[Recurso-26] Informações : RECURSO - Valor - Indice de Sucesso 1
[Recurso-25] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-25 (ACTIVATED)
[Recurso-25] Informações : RECURSO - Valor - Indice de Sucesso 12
[Recurso-24] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-24 (ACTIVATED)
[Recurso-24] Informações : RECURSO - Valor - Indice de Sucesso 86
[Recurso-11] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-11 (ACTIVATED)
[Recurso-11] Informações : RECURSO - Valor - Indice de Sucesso 38
[Recurso-19] Informações : RECURSO - Eu nasci - Minha competência é drone2 e sou Recurso-19 (ACTIVATED)
[Recurso-19] Informações : RECURSO - Valor - Indice de Sucesso 50
[Recurso-27] Informações : RECURSO - Eu nasci - Minha competência é robot3 e sou Recurso-27 (ACTIVATED)
[Recurso-27] Informações : RECURSO - Valor - Indice de Sucesso 50
[Gerente-3] Informações : GERENTE - Eu recebi um requisicao para drone1 do 6@(fabrica,fabrica-clientes,cliente)@MK-479
[Recurso-16] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-18] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-21] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-8] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-7] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-9] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-12] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-15] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Gerente-3] Informações : GERENTE - A melhor oferta veio de16@(fabrica,fabrica-recursos,drone1-recurso)@MK-479 1
[Recurso-16] Informações : RECURSO - Eu fui selecionado para missão :)
[Recurso-16] Informações : RECURSO - Este é o No do Contrato= 55722961145573
[Gerente-3] Informações : GERENTE - O Recurso foi contratado! Enviando o número do contrato para o Cliente - o solicitante
foi6@(fabrica,fabrica-clientes,cliente)@MK-479
[Gerente-3] Informações : GERENTE - Estatistica de Gerente-3 (LIVING):
    - noDeGerentesCriados = 3
    - noDeRecursosCriados = 20
    - noDeClientesCriados = 2
    - noDeContratosFinalizadosComSucesso (Done) = 0
    - noDeContratosFinalizadosSemSucesso (Falha)  = 0
    - noVezQueClienteFicouSemRespostasDeGerentesSaiuSemAtendimento (Sem Atendimento)  = 0
    - noDeContratosNegociadosRecursos = 2
    - noDeRecursosContratadosComSucesso = 2
    - noDeMensagensTrocasComRecursos = 12
    - noDeMensagensRecebidasDeClientes = 2
    - noNegociacoesSemPropostaRecursos              = 0
    - noNegociacoesSemRespostaRecursosAposSelecionado = 0
    - noVezQueClienteFicouSemRespostasDeGerentes      = 0
    - noDeSubContratosFinalizadosComSucesso  = 0
    - noDeSubContratosFinalizadosSemSucesso  = 0
[Gerente-3] Informações :
GERENTE - 31/08/2015 13:12:13
```

[Cliente-6] Informações : CLIENTE - Eu encontrei um Gerente: 3@(fabrica,fabrica-clientes,manager)@MK-479
[Recurso-10] Informações : RECURSO - Eu - Recurso-10 (LIVING) competência - drone2 realizei 21 PORCENTO - STATUS - Em Andamento
[Cliente-13] Informações : CLIENTE - Missao em Andamento pelo Recurso10@(fabrica,55721519508751,manager)@MK-479 :) - 21 PORCENTO
[Recurso-16] Informações : RECURSO - Eu - Recurso-16 (LIVING) competência - drone1 realizei 60 PORCENTO - STATUS - Em Andamento
[Cliente-6] Informações : CLIENTE - Missao em Andamento pelo Recurso16@(fabrica,55722961145573,manager)@MK-479 :) - 60 PORCENTO
[Recurso-10] Informações : RECURSO - Eu - Recurso-10 (LIVING) competência - drone2 realizei a missao ! - sendReply para 13@(fabrica,55721519508751,cliente)@MK-479
[Cliente-13] Informações : CLIENTE - Sim: Missao FINALIZADA :)
[Cliente-13] Informações : CLIENTE - Estou saindo mais agora, mas vou chamar outro cliente !
[Gerente-3] Informações : GERENTE - Eu recebi um requisicao para drone1 do 16@(fabrica,fabrica-clientes,cliente)@MK-479
[Recurso-7] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-18] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-21] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-9] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-8] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-12] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-15] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-10] Informações : RECURSO - Recurso-10 (LIVING) com competência de drone2- Estatisticas:
- noDeContratosFinalizadosComSucessoPorMim = 1
- noDeContratosFinalizadosSemSucessoPorMim = 0
- noDeSubContratosFinalizadosComSucessoGerenciadosPorMim = 0
- noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim = 0
[Recurso-10] Informações : RECURSO - Recurso-10 (LIVING) com competência de drone2- meu índice de Acertos atual é :87
[Recurso-10] Informações : RECURSO - Mon Aug 31 13:12:09 GMT-04:00 2015
[Gerente-3] Informações : GERENTE - A melhor oferta veio de9@(fabrica,fabrica-recursos,drone1-recurso)@MK-479 91
[Recurso-9] Informações : RECURSO - Eu fui selecionado para missão :)
[Recurso-9] Informações : RECURSO - Este é o No do Contrato= 55728627513614
[Gerente-3] Informações : GERENTE - O Recurso foi contratado! Enviando o número do contrato para o Cliente - o solicitante foi16@(fabrica,fabrica-clientes,cliente)@MK-479
[Gerente-3] Informações : GERENTE - Estatistica de Gerente-3 (LIVING):
- noDeGerentesCriados = 3
- noDeRecursosCriados = 20
- noDeClientesCriados = 2
- noDeContratosFinalizadosComSucesso (Done) = 1
- noDeContratosFinalizadosSemSucesso (Falha) = 0
- noVezQueClienteFicouSemRespostasDeGerentesSaiuSemAtendimento (Sem Atendimento) = 0
- noDeContratosNegociadosRecursos = 3
- noDeRecursosContratadosComSucesso = 3
- noDeMensagensTrocasComRecursos = 24
- noDeMensagensRecebidasDeClientes = 3
- noNegociacoesSemPropostaRecursos = 0
- noNegociacoesSemRespostaRecursosAposSelecionado = 0
- noVezQueClienteFicouSemRespostasDeGerentes = 0
- noDeSubContratosFinalizadosComSucesso = 0
- noDeSubContratosFinalizadosSemSucesso = 0
[Gerente-3] Informações :
GERENTE - 31/08/2015 13:12:19
[Recurso-16] Informações : RECURSO - CLIENTE - Eu encontrei um Gerente: 3@(fabrica,fabrica-clientes,manager)@MK-479
[Gerente-4] Informações : GERENTE - Eu recebi um requisicao para drone1 do 9@(fabrica,fabrica-clientes,cliente)@MK-479
[Recurso-18] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-7] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-12] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-15] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-21] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-8] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Gerente-4] Informações : GERENTE - A melhor oferta veio de18@(fabrica,fabrica-recursos,drone1-recurso)@MK-479 88
[Recurso-18] Informações : RECURSO - Eu fui selecionado para missão :)
[Recurso-18] Informações : RECURSO - Este é o No do Contrato= 55731836675487
[Gerente-4] Informações : GERENTE - O Recurso foi contratado! Enviando o número do contrato para o Cliente - o solicitante foi9@(fabrica,fabrica-clientes,cliente)@MK-479
[Gerente-4] Informações : GERENTE - Estatistica de Gerente-4 (LIVING):

- noDeGerentesCriados = 3
- noDeRecursosCriados = 20
- noDeClientesCriados = 2

- noDeContratosFinalizadosComSucesso (Done) = 1
- noDeContratosFinalizadosSemSucesso (Falha) = 0
- noVezQueClienteFicouSemRespostasDeGerentesSaiuSemAtendimento (Sem Atendimento) = 0

- noDeContratosNegociadosRecursos = 4
- noDeRecursosContratadosComSucesso = 4

- noDeMensagensTrocasComRecursos = 32
- noDeMensagensRecebidasDeClientes = 4

- noNegociacoesSemPropostaRecursos = 0
- noNegociacoesSemRespostaRecursosAposSelecionado = 0
- noVezQueClienteFicouSemRespostasDeGerentes = 0

- noDeSubContratosFinalizadosComSucesso = 0
- noDeSubContratosFinalizadosSemSucesso = 0

[Gerente-4] Informações :

GERENTE - 31/08/2015 13:12:22

[Recurso-9] Informações : RECURSO - CLIENTE - Eu encontrei um Gerente: 4@(fabrica,fabrica-clientes,gerente)@MK-479
[Cliente-28] Informações : CLIENTE - Eu nasci e sou Cliente-28 (ACTIVATED)
[Cliente-28] Informações : CLIENTE - Necessito de um drone em drone1 in 1486 ms !
[Recurso-18] Informações : RECURSO - Eu - Recurso-18 (LIVING) competência - drone1 realizei a missao ! - sendReply para 9@(fabrica,55731836675487,cliente)@MK-479
[Gerente-5] Informações : GERENTE - Eu recebi um requisicao para drone1 do 28@(fabrica,fabrica-clientes,cliente)@MK-479
[Recurso-21] Informações : RECURSO - Eu recebi uma chamada de proposta de 5@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-8] Informações : RECURSO - Eu recebi uma chamada de proposta de 5@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-15] Informações : RECURSO - Eu recebi uma chamada de proposta de 5@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-12] Informações : RECURSO - Eu recebi uma chamada de proposta de 5@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-7] Informações : RECURSO - Eu recebi uma chamada de proposta de 5@(fabrica,fabrica-recursos,gerente)@MK-479
[Gerente-5] Informações : GERENTE - A melhor oferta veio de8@(fabrica,fabrica-recursos,drone1-recurso)@MK-479 87
[Recurso-8] Informações : RECURSO - Eu fui selecionado para missão :)
[Recurso-8] Informações : RECURSO - Este é o No do Contrato= 55735397708096
[Gerente-5] Informações : GERENTE - O Recurso foi contratado! Enviando o número do contrato para o Cliente - o solicitante foi28@(fabrica,fabrica-clientes,cliente)@MK-479
[Gerente-5] Informações : GERENTE - Estatistica de Gerente-5 (LIVING):
- noDeGerentesCriados = 3
- noDeRecursosCriados = 20
- noDeClientesCriados = 3

- noDeContratosFinalizadosComSucesso (Done) = 1
- noDeContratosFinalizadosSemSucesso (Falha) = 0
- noVezQueClienteFicouSemRespostasDeGerentesSaiuSemAtendimento (Sem Atendimento) = 0

- noDeContratosNegociadosRecursos = 5
- noDeRecursosContratadosComSucesso = 5

- noDeMensagensTrocasComRecursos = 39
- noDeMensagensRecebidasDeClientes = 5

- noNegociacoesSemPropostaRecursos = 0
- noNegociacoesSemRespostaRecursosAposSelecionado = 0
- noVezQueClienteFicouSemRespostasDeGerentes = 0

- noDeSubContratosFinalizadosComSucesso = 0
- noDeSubContratosFinalizadosSemSucesso = 0

[Gerente-5] Informações :

GERENTE - 31/08/2015 13:12:25

[Cliente-28] Informações : CLIENTE - Eu encontrei um Gerente: 5@(fabrica,fabrica-clientes,gerente)@MK-479
[Recurso-18] Informações : RECURSO - Recurso-18 (LIVING) com competência de drone1- Estatisticas:
- noDeContratosFinalizadosComSucessoPorMim = 1
- noDeContratosFinalizadosSemSucessoPorMim = 0
- noDeSubContratosFinalizadosComSucessoGerenciadosPorMim = 0
- noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim = 0

[Recurso-9] Informações : RECURSO - CLIENTE - Sim: Missao FINALIZADA :) com COMPARTILHAMENTO DE APRENDIZADO - indiceSucesso = 18
[Recurso-16] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-16 (LIVING)
[Recurso-16] Informações : RECURSO - Valor - Indice de Sucesso 13
[Recurso-16] Informações : RECURSO - CLIENTE - NAO FUI ATENTIDO - FALHAS - :(
[Recurso-16] Informações : RECURSO - Eu - Recurso-16 (LIVING) competência - drone1 FALHEI Na missao ! - sendReply para 6@(fabrica,55722961145573,cliente)@MK-479
[Cliente-6] Informações : CLIENTE - Missao FALHOU :)
[Recurso-8] Informações : RECURSO - Eu - Recurso-8 (LIVING) competência - drone1 realizei 49 PORCENTO - STATUS - Em Andamento
[Cliente-28] Informações : CLIENTE - Missao em Andamento pelo Recurso8@(fabrica,55735397708096,manager)@MK-479 :) - 49 PORCENTO
[Cliente-6] Informações : CLIENTE - Não fui atendido apos 1 falha - :(
[Cliente-6] Informações : CLIENTE - Estou saindo mais agora, mas vou chamar outro cliente !
[Recurso-16] Informações : RECURSO - Recurso-16 (LIVING) com competência de drone1- Estatisticas:
- noDeContratosFinalizadosComSucessoPorMim = 0
- noDeContratosFinalizadosSemSucessoPorMim = 1
- noDeSubContratosFinalizadosComSucessoGerenciadosPorMim = 0
- noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim = 1

[Recurso-9] Informações : RECURSO - Eu nasci - Minha competência é drone1 e sou Recurso-9 (LIVING)
[Recurso-9] Informações : RECURSO - Valor - Indice de Sucesso 18
[Recurso-8] Informações : RECURSO - Eu - Recurso-8 (LIVING) competência - drone1 realizei a missao ! - sendReply para 28@(fabrica,55735397708096,cliente)@MK-479
[Cliente-28] Informações : CLIENTE - Sim: Missao FINALIZADA :)
[Recurso-9] Informações : RECURSO - Recurso-9 (LIVING) com competência de drone1- Estatisticas:
- noDeContratosFinalizadosComSucessoPorMim = 0
- noDeContratosFinalizadosSemSucessoPorMim = 1
- noDeSubContratosFinalizadosComSucessoGerenciadosPorMim = 1
- noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim = 0

[Recurso-9] Informações : RECURSO - Recurso-9 (LIVING) com competência de drone1- meu índice de Acertos atual é :18
[Recurso-9] Informações : RECURSO - Mon Aug 31 13:12:09 GMT-04:00 2015
[Recurso-9] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-9] Informações : RECURSO - Eu recebi uma chamada de proposta de 5@(fabrica,fabrica-recursos,gerente)@MK-479
[Cliente-28] Informações : CLIENTE - Estou saindo mais agora, mas vou chamar outro cliente !
[Recurso-8] Informações : RECURSO - Recurso-8 (LIVING) com competência de drone1- Estatisticas:
- noDeContratosFinalizadosComSucessoPorMim = 1
- noDeContratosFinalizadosSemSucessoPorMim = 0
- noDeSubContratosFinalizadosComSucessoGerenciadosPorMim = 0
- noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim = 0

[Recurso-8] Informações : RECURSO - Recurso-8 (LIVING) com competência de drone1- meu índice de Acertos atual é :54
[Recurso-8] Informações : RECURSO - Mon Aug 31 13:12:09 GMT-04:00 2015
[Cliente-29] Informações : CLIENTE - Eu nasci e sou Cliente-29 (ACTIVATED)
[Cliente-29] Informações : CLIENTE - Necessito de um drone em robot3 in 1811 ms !
[Gerente-5] Informações : GERENTE - Eu recebi um requisicao para robot3 do 29@(fabrica,fabrica-clientes,cliente)@MK-479
[Recurso-17] Informações : RECURSO - Eu recebi uma chamada de proposta de 5@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-23] Informações : RECURSO - Eu recebi uma chamada de proposta de 5@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-24] Informações : RECURSO - Eu recebi uma chamada de proposta de 5@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-27] Informações : RECURSO - Eu recebi uma chamada de proposta de 5@(fabrica,fabrica-recursos,gerente)@MK-479
[Gerente-5] Informações : GERENTE - A melhor oferta veio de24@(fabrica,fabrica-recursos,robot3-recurso)@MK-479 10
[Recurso-24] Informações : RECURSO - Eu fui selecionado para missão :)
[Recurso-24] Informações : RECURSO - Este é o No do Contrato= 55746810671984
[Gerente-5] Informações : GERENTE - O Recurso foi contratado! Enviando o número do contrato para o Cliente - o solicitante foi29@(fabrica,fabrica-clientes,cliente)@MK-479
[Gerente-5] Informações : GERENTE - Estatistica de Gerente-5 (LIVING):
- noDeGerentesCriados = 3
- noDeRecursosCriados = 20
- noDeClientesCriados = 4
- noDeContratosFinalizadosComSucesso (Done) = 2
- noDeContratosFinalizadosSemSucesso (Falha) = 1
- noVezQueClienteFicouSemRespostasDeGerentesSaiuSemAtendimento (Sem Atendimento) = 0

- noDeContratosNegociadosRecursos = 6
- noDeRecursosContratadosComSucesso = 6

- noDeMensagensTrocasComRecursos = 48
- noDeMensagensRecebidasDeClientes = 6

- noNegociacoesSemPropostaRecursos = 0
- noNegociacoesSemRespostaRecursosAposSelecionado = 0
- noVezQueClienteFicouSemRespostasDeGerentes = 0

- noDeSubContratosFinalizadosComSucesso = 1
- noDeSubContratosFinalizadosSemSucesso = 1

[Gerente-5] Informações :

GERENTE - 31/08/2015 13:12:37

[Cliente-29] Informações : CLIENTE - Eu encontrei um Gerente: 5@(fabrica,fabrica-clientes,gerente)@MK-479
[Cliente-30] Informações : CLIENTE - Eu nasci e sou Cliente-30 (ACTIVATED)
[Cliente-30] Informações : CLIENTE - Necessito de um drone em drone2 in 2308 ms !
[Recurso-24] Informações : RECURSO - Eu - Recurso-24 (LIVING) competência - robot3 realizei 21 PORCENTO - STATUS - Em Andamento
[Cliente-29] Informações : CLIENTE - Missao em Andamento pelo Recurso24@(fabrica,55746810671984,manager)@MK-479 :) - 21 PORCENTO
[Gerente-4] Informações : GERENTE - Eu recebi um requisicao para drone2 do 30@(fabrica,fabrica-clientes,cliente)@MK-479
[Recurso-20] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-14] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-26] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-25] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-22] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-10] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-11] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-19] Informações : RECURSO - Eu recebi uma chamada de proposta de 4@(fabrica,fabrica-recursos,gerente)@MK-479
[Gerente-4] Informações : GERENTE - A melhor oferta veio de19@(fabrica,fabrica-recursos,drone2-recurso)@MK-479 14
[Recurso-19] Informações : RECURSO - Eu fui selecionado para missão :)
[Recurso-19] Informações : RECURSO - Este é o No do Contrato= 55750728891794
[Gerente-4] Informações : GERENTE - O Recurso foi contratado! Enviando o número do contrato para o Cliente - o solicitante foi30@(fabrica,fabrica-clientes,cliente)@MK-479
[Gerente-4] Informações : GERENTE - Estatistica de Gerente-4 (LIVING):
- noDeGerentesCriados = 3
- noDeRecursosCriados = 20
- noDeClientesCriados = 5

- noDeContratosFinalizadosComSucesso (Done) = 2
- noDeContratosFinalizadosSemSucesso (Falha) = 1
- noVezQueClienteFicouSemRespostasDeGerentesSaiuSemAtendimento (Sem Atendimento) = 0

- noDeContratosNegociadosRecursos = 7
- noDeRecursosContratadosComSucesso = 7

- noDeMensagensTrocasComRecursos = 58
- noDeMensagensRecebidasDeClientes = 7

- noNegociacoesSemPropostaRecursos = 0
- noNegociacoesSemRespostaRecursosAposSelecionado = 0
- noVezQueClienteFicouSemRespostasDeGerentes = 0

- noDeSubContratosFinalizadosComSucesso = 1
- noDeSubContratosFinalizadosSemSucesso = 1
[Gerente-4] Informações :

GERENTE - 31/08/2015 13:12:41

[Cliente-30] Informações : CLIENTE - Eu encontrei um Gerente: 4@(fabrica,fabrica-clientes,gerente)@MK-479
[Recurso-24] Informações : RECURSO - Eu - Recurso-24 (LIVING) competência - robot3 realizei a missao ! - sendReply para 29@(fabrica,55746810671984,cliente)@MK-479
[Cliente-29] Informações : CLIENTE - Sim: Missao FINALIZADA :)
[Recurso-19] Informações : RECURSO - Eu - Recurso-19 (LIVING) competência - drone2 realizei 51 PORCENTO - STATUS - Em Andamento
[Cliente-30] Informações : CLIENTE - Missao em Andamento pelo Recurso19@(fabrica,55750728891794,manager)@MK-479 :) - 51 PORCENTO
[Recurso-24] Informações : RECURSO - Recurso-24 (LIVING) com competência de robot3- Estatisticas:
- noDeContratosFinalizadosComSucessoPorMim = 1
- noDeContratosFinalizadosSemSucessoPorMim = 0
- noDeSubContratosFinalizadosComSucessoGerenciadosPorMim = 0
- noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim = 0
[Recurso-24] Informações : RECURSO - Recurso-24 (LIVING) com competência de robot3- meu índice de Acertos atual é :86
[Cliente-29] Informações : CLIENTE - Estou saindo mais agora, mas vou chamar outro cliente !
[Recurso-19] Informações : RECURSO - Eu - Recurso-19 (LIVING) competência - drone2 realizei a missao ! - sendReply para 30@(fabrica,55750728891794,cliente)@MK-479
[Cliente-30] Informações : CLIENTE - Sim: Missao FINALIZADA :)
[Recurso-19] Informações : RECURSO - Recurso-19 (LIVING) com competência de drone2- Estatisticas:
- noDeContratosFinalizadosComSucessoPorMim = 1
- noDeContratosFinalizadosSemSucessoPorMim = 0
- noDeSubContratosFinalizadosComSucessoGerenciadosPorMim = 0
- noDeSubContratosFinalizadosSemSucessoGerenciadosPorMim = 0
[Recurso-19] Informações : RECURSO - Recurso-19 (LIVING) com competência de drone2- meu índice de Acertos atual é :50
[Cliente-30] Informações : CLIENTE - Estou saindo mais agora, mas vou chamar outro cliente !
[Cliente-31] Informações : CLIENTE - Eu nasci e sou Cliente-31 (ACTIVATED)
[Cliente-31] Informações : CLIENTE - Necessito de um drone em robot3 in 2301 ms !
[Cliente-32] Informações : CLIENTE - Eu nasci e sou Cliente-32 (ACTIVATED)
[Cliente-32] Informações : CLIENTE - Necessito de um drone em robot3 in 2274 ms !
[Gerente-3] Informações : GERENTE - Eu recebi um requisicao para robot3 do 31@(fabrica,fabrica-clientes,cliente)@MK-479
[Recurso-17] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-23] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Recurso-27] Informações : RECURSO - Eu recebi uma chamada de proposta de 3@(fabrica,fabrica-recursos,gerente)@MK-479
[Gerente-3] Informações : GERENTE - A melhor oferta veio de17@(fabrica,fabrica-recursos,robot3-recurso)@MK-479 28
[Recurso-17] Informações : RECURSO - Eu fui selecionado para missão :)
[Recurso-17] Informações : RECURSO - Este é o No do Contrato= 55764646529984
[Gerente-3] Informações : GERENTE - O Recurso foi contratado! Enviando o número do contrato para o Cliente - o solicitante foi31@(fabrica,fabrica-clientes,cliente)@MK-479
[Gerente-3] Informações : GERENTE - Estatistica de Gerente-3 (LIVING):
- noDeGerentesCriados = 3
- noDeRecursosCriados = 20
- noDeClientesCriados = 7

- noDeContratosFinalizadosComSucesso (Done) = 4
- noDeContratosFinalizadosSemSucesso (Falha) = 1
- noVezQueClienteFicouSemRespostasDeGerentesSaiuSemAtendimento (Sem Atendimento) = 0

- noDeContratosNegociadosRecursos = 8
- noDeRecursosContratadosComSucesso = 8

- noDeMensagensTrocasComRecursos = 65
- noDeMensagensRecebidasDeClientes = 8

- noNegociacoesSemPropostaRecursos = 0
- noNegociacoesSemRespostaRecursosAposSelecionado = 0
- noVezQueClienteFicouSemRespostasDeGerentes = 0

- noDeSubContratosFinalizadosComSucesso = 1
- noDeSubContratosFinalizadosSemSucesso = 1
[Gerente-3] Informações :

GERENTE - 31/08/2015 13:12:55