Universidade do Minho
Escola de Engenharia
Departamento de Informática

Um Ambiente Computacional Distribuído para a Implementação de Sistemas Multiagente

Orlando Manuel de Oliveira Belo

Tese de Doutoramento

1997

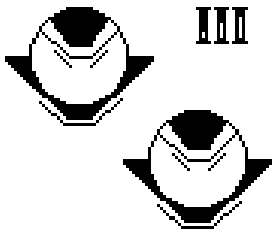
Universidade do Minho
Escola de Engenharia
Departamento de Informática

Um Ambiente Computacional Distribuído para a Implementação de Sistemas Multiagente

Orlando Manuel de Oliveira Belo

Tese submetida à Universidade do Minho
para a obtenção do grau de Doutor em Informática, Área Científica de Inteligência Artificial, elaborada sob a orientação do
Professor Doutor José Carlos Ferreira Maia Neves.

1997



Um Ambiente Computacional Distribuído para a Implementação de Sistemas Multiagente

Aos meus Pais.

Agradecimentos

Queria deixar expressos os meus agradecimentos a todos aqueles que contribuiram para a realização desta tese e trabalho subjacente ao projecto relacionado com o sistema BEABLE, em particular:

- Ao meu orientador e amigo Professor Doutor José Carlos Ferreira Maia Neves, que com a sua disponibilidade e incentivo, e através das suas críticas e sugestões atentas, tornou uma realidade a presente tese, tendo ainda contribuído significativamente para o desenvolvimento do meu espírito de iniciativa e investigação científica.
- Ao Professor Doutor Eugénio de Oliveira, pela oportunidade para apresentar o meu trabalho ao seu grupo de investigação e pela disponibilidade de diálogo sempre demonstrada.
- Ao Professor Doutor Keith Clark, pelo acolhimento no seu departamento durante a minha visita de trabalho ao Imperial College, em Londres.
- À Professora Doutora Cecília Leão, ao Professor Doutor Páscoa Machado e à Dra. Joanne Paisana pelo tempo que dispensaram em apoio aos meus processos de investigação.

- Ao meu colega e amigo António Nestor Ribeiro, pela sua atenção sempre que solicitado e pelas contribuições nos projectos que realizámos conjuntamente.

- Aos meus colegas do Departamento de Informática Luís Amaral, José Bernardo, José Pina de Miranda, Joaquim Bacelar, João Saraiva, Jorge Sousa Pinto, Jorge Portugal, José João Almeida, João Miguel Fernandes e António Costa, pelo espírito de colaboração e incentivo.

- Aos meus colegas da Área Científica de Inteligência Artificial do Departamento de Informática José Machado, Paulo Novais, César Analide, António Abelha e Paulo Azevedo pelo acréscimo de paciência e, eventualmente, de trabalho que lhes foi exigido para que a realização deste projecto fosse possível.

- À Benedita Malheiro pela generosa discussão de ideias.

- Ao Abílio Ribeiro, ao Filipe Santos, ao Vítor Alves e ao Rui Mendes pelo espírito de abertura na discussão de problemas.

- À Áurea Cardoso, à Sofia Afonso e à Madalena Costa pela disponibilidade para a leitura dos meus trabalhos.

- Aos meus alunos, porque talvez a docência e a investigação não valessem a pena sem a sua necessidade de aprender e curiosidade científica.

- À Universidade do Minho, à Escola de Engenharia e ao Departamento de Informática pelas facilidades concedidas no meu processo de valorização pedagógica, científica e técnica.

- Aos meus pais e amigos pelo apoio permanente durante o processo de realização da tese.

- À Anabela Barros pelas suas críticas, sugestões literárias e acompanhamento durante a escrita da presente tese.

Resumo

Um Ambiente Computacional Distribuído para a Implementação de Sistemas Multiagente

É comum depararem-se-nos, numa organização, situações em que os processos de resolução de problemas estão naturalmente distribuídos por entidades com responsabilidades próprias, que se assumem como conscientes e deliberativas: os especialistas. Um indivíduo pode não ter a aptidão necessária para, só por si, resolver um problema ou lidar com os diferentes tipos de recursos envolvidos na sua resolução. Contudo, numa organização, estas entidades atacam eficientemente um problema através da partilha de tarefas ou de resultados, desenvolvendo novas formas de acção, planeando e alocando recursos ou, simplesmente, trocando informação entre si. Trata-se, em síntese, de pensar, planear e agir em equipa. Com as estruturas, estratégias e recursos apropriados, esses indivíduos podem actuar como equipas efectivas. Tais características afectarão o desempenho dos diversos indivíduos, conduzindo-os a melhores e mais efectivas abordagens na resolução de problemas, o que terá, naturalmente, um impacto positivo nos custos operacionais da organização. Para se estabelecerem e manterem canais de comunicação adequados dentro desta, é de importância primordial prevenir situações de estrangulamento, de

contenção ou de bloqueio fatal. Este cenário é, com certeza, insuficiente para caracterizar adequadamente todos os aspectos e formas de comportamento existentes numa organização, no entanto, constituiu o principal estímulo para o desenvolvimento do sistema BEABLE.

O sistema BEABLE é um ambiente computacional distribuído, dirigido para a implementação de sistemas cooperativos baseados em conhecimento. Com uma arquitectura genérica, baseada em quadros negros, está particularmente orientado para aplicações reais de Inteligência Artificial Distribuída em áreas que vão desde o diagnóstico à classificação e controlo. Os agentes são entidades autónomas, interactivas e automotivadas, que transmitem um sentido de realidade, de conhecimento e de intenção nas tarefas em que se aplicam. São membros de comunidades especializadas, com aptência e capacidade de agir coerente e cooperativamente, reagindo a estímulos de forma assíncrona e concorrente. Uma aplicação é construída em torno de um ou mais agentes. Estes podem reagir oportunisticamente, usando estratégias de resolução de acordo com o estado corrente do problema e com a perícia e conhecimento que possuem. Não se trata de um sistema centralizado. Os agentes têm controlo local, bem como dos processos de interacção que possam desenvolver com outros agentes. O desenho, de características modulares, que o sistema apresenta e os seus mecanismos funcionais permitem que novos agentes possam ser integrados no sistema sem que a sua estrutura global seja afectada. Complementarmente, os agentes estão providos dos meios necessários para poderem aceder ao conhecimento e usufruir da perícia de agentes externos à comunidade. Esta circunstância permite melhorar significativamente o desempenho do sistema e expandir os horizontes da comunidade em que os agentes se integram, em direcção a uma comunidade global de comunidades.

Abstract

A Distributed Computational Environment for Multiagent Systems Implementation

Nowadays, it is very common to find a large range of situations in human organisations where problem solving activities are naturally distributed among individuals, which assume specific responsibilities in the organisation's structure and make use of the faculty of being conscious and deliberative: the organisation's experts. As an individual, he/she may have neither the expertise nor the capacity to solve a whole problem or even to deal with the resources involved. However, in an organisation with different experts, he/she may solve problems efficiently by sharing tasks or results, developing new forms of acting, planning, allocating resources, or even by simply exchanging information. In short, thinking, planning and acting as a cohesive "task force". With the appropriate structures, strategies and resources, individuals can act as teams. This will affect the individual's performance, leading to better and more effective approaches to problem solving, with a positive impact on the overall operational costs of the organisation. To establish and maintain adequate communication channels inside the organisation, it is of prime importance to prevent communication bottlenecks, contentions or deadlock

situations. This scenario is certainly insufficient to characterise all the aspects and forms of behaviour of an organisation. However, it was the prime "stimulus" for the development of the BEABLE system.

The BEABLE system is a distributed computational environment for the implementation of co-operative knowledge based systems. As a general-purpose blackboard-based architecture, it is particularly suited for real world Distributed Artificial Intelligence applications, in areas ranging from diagnosis to those of classification and control. Agents are autonomous, interactive and self-motivated entities, that bring a sense of reality, awareness and purpose to the problem solving process. They are members of specialised communities acting as coherent and co-operative teams, reacting to stimuli, being asynchronous and concurrent. The system allows the execution of several applications simultaneously, sharing different machines connected to a network. A user's application is built around one or more agents. Agents can also act opportunelly, applying their own resolution strategies according to the actual state of the problem and to their own skills and knowledge. There is no central entity to co-ordinate the system's dynamics. Each agent controls its own behaviour and any interaction with other agents of the system. The system's modular design and functional mechanisms allow the integration of new agents without affecting the system's overall structure. The agents are also provided with the means to access knowledge and to benefit from the expertise of other agents of external communities. This will significantly improve the agents' skills, expanding the horizons of a community towards a global community of communities.

Résumé

Un Environement Informatique Distribué pour l'Implémentation de Systèmes Multiagent

Actuellement, il est fréquent de rencontrer dans les organisations humaines une large gamme de situations où les activités de résolution des problèmes sont naturellement distribuées entre les individus, entités qui assument un ensemble de responsabilités spécifiques dans la structure des organisations, en utilisant leur faculté de conscience et de délibération: les spécialistes. Un individu déterminé peut ne pas avoir l'aptitude ou les capacités nécessaires pour, de lui-même, résoudre un problème complet, ou même être capable de gérer les différents recours que le problème peut englober. Cependant, dans une organisation, composée de différents spécialistes, il/elle peut résoudre efficacement un problème, à travers la division des tâches ou des résultats, en développant de nouvelles formes d'action, en planifiant et en définissant l'emplacement des recours ou simplement en échangeant de l'information. Pour résumer: il s'agit de penser, planifier et agir comme une équipe de travail cohésive. Avec les structures, les stratégies et les recours appropriés, les individus d'une organisation peuvent travailler comme des équipes effectives. Ceci influencera la performance des divers individus, les conduisant à des

approches plus performantes et plus concrêtes dans la résolution des problèmes, et par conséquent, cela aura un impact positif sur les coûts opérationnels de l'organisation. Pour établir et maintenir des réseaux de communication adéquats, il est extrèmement important de prévenir d'éventuelles situations d'étranglement, de rétention et de "deadlock". Ce scénario est certainement insuffisant pour caractériser convenablement tous les aspects et tous les types de comportement existant dans une organisation. Toutefois, cela a constitué le principal motif du développement du système BEABLE.

Le système BEABLE est un environement informatique distribué pour l'implémentation de systèmes experts coopératifs. Avec une architecture générique, basée sur des "tableaux noirs", il est particulièrement orienté vers des applications réelles de l'Intelligence Artificielle Distribuée dans des domaines qui peuvent aller du diagnostic jusque à la classification et au contrôle. Les agents sont des entités autonomes, interactives et auto-motivées qui transmettent un sens de réalité, de connaissance et d'intention au processus de résolution des problèmes. Ce sont des membres de communautés spécialisées qui possedent une aptitude et une capacité à agir avec cohérence et en coopération, comme une équipe, en réagissant aux stimuli d´une forme asynchronique et concurrente. Le système permet l'exécution d'applications variées, simultanément, à travers le partage de différentes machines liées par un système de réseaux d'ordinateurs. L'application d'un utilisateur est construite autour d'un ou de plusieurs agents. Les agents peuvent agir oportunément, en appliquant leurs stratégies de résolution, en accord avec l'état actuel du problème et en accord avec l'habilité et les connaissances qu'ils possèdent. Dans le système, il n'existe pas d'entité responsable de la coordination de sa dynamique globale. Chaque agent a la capacité de contrôler son propre comportement et tout autre processus d'interaction qu'il puisse développer avec d'autres

agents. L'architecture, à caractéristiques modulaires, présentée par le système, tout comme ses mécanismes fonctionnels, permettent que de nouveaux agents puissent être intégrés dans le système sans que la structure globale en soit affectée. En outre, les agents sont pourvus des moyens nécessaires pour pouvoir accéder à la connaissance et pour bénéficier de l'habilité d'autres agents, externes à leur communauté. Cette circonstance permet d'améliorer significativement l'habilité des agents locaux et d'élargir les horizons de la communauté dans laquelle ils sont intégrés, vers une communauté globale de communautés.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nota introdutória sobre organizações humanas e suas especificidades. Algumas considerações gerais sobre a tecnologia de agentes e suas aplicações. Motivação, objectivos e organização da tese.

Não é necessária uma análise muito cuidada para se verificar que a complexidade dos problemas com que uma organização se debate no seu quotidiano cresce de dia para dia de uma forma quase exponencial. A exigência de novos métodos de trabalho, a necessidade de tratamento de maiores volumes de informação, o desenvolvimento de procedimentos adequados para a geração de conhecimento especializado e a presença do "espectro" da competitividade no espaço onde a organização está implantada são alguns dos factores a influir na definição das estratégias de uma organização.

Quanto maior for a dinâmica da organização e a sua necessidade de crescimento mais frequentes são as mutações no seu património cognitivo, mais exigentes se tornam os processos de decisão, mais necessário se afigura um melhor desempenho da sua parte e maior é a necessidade de se possuirem me-

canismos de controlo flexíveis e eficientes. A procura constante de melhores soluções, em termos de desempenho e de eficiência, e a criação de mecanismos de suporte para a redução de erros tendem, desta forma, a constituir actividades correntes no seu seio.

As situações mais críticas ocorrem normalmente devido a uma deficiente conjugação das sinergias geradas pelas entidades integradas na organização que, de uma forma ou de outra, intervêm, directa ou indirectamente, nos processos de decisão. Muitas vezes, a falta de vontade política, a fraca aptência para o desenvolvimento de acções coordenadas ou a falta de disponibilidade a tempo inteiro dos especialistas, dentro de um leque mais vasto de factores críticos, constituem sérios casos de estrangulamento na eficácia e na obtenção de soluções, ou na realização de tarefas especializadas. Uma possível abordagem, por forma a atenuar esta situação, concretizar-se-ia através da integração nos domínios aplicacionais onde os problemas ocorrem de indivíduos altamente especializados, cuja perícia e conhecimento actuassem como catalizadores, quer na tomada de decisões e no consequente alcance de uma solução, quer, simplesmente, na estratégia de execução de uma dada tarefa em particular.

Complementarmente, a cooperação entre entidades será central no processo de desenvolvimento de soluções, o que lhes possibilitará alcançar maiores níveis de perícia, permitindo-lhes lidar de uma forma igualmente natural com problemas abrangendo áreas pluridisciplinares. Contudo, a conjugação de desempenhos poderá ser seriamente afectada na prática, no caso de essas entidades se situarem em locais geograficamente distintos, onde o isolamento pode ser significativo. Tal situação será, porém, meramente circunstancial, se os especialistas possuirem as estruturas e os meios de comunicação necessários para que os seus processos de interacção e resolução de problemas possam

ser executados em tempo útil e de forma efectiva.

As situações apresentadas revelam um dos diversos cenários que é possível encontrar em modelos organizacionais. Os modelos protagonizados pelo homem enquanto ser gregário e cooperativo sempre foram uma fonte de inspiração para a criação e desenvolvimento de modelos artificiais. O presente trabalho desenrola-se em torno do processo de adopção de alguns desses modelos e da análise da forma como eles podem ser traduzidos e adquiridos por ferramentas computacionais.

1.1 Considerações Gerais

A *Inteligência Artificial* (IA) (S.Russel & P.Norvig 1995) (E.Charniak & D.McDermott 1985) tem sido identificada com o estudo dos processos pelos quais os computadores podem realizar tarefas cognitivas e com o modo de representar a perícia e o conhecimento de entidades inteligentes. É simultaneamente encarada como uma disciplina científica e como uma actividade de engenharia, integrando no seu património científico uma grande diversidade de subdomínios, distribuídos por áreas que vão desde a *Robótica*, a *Representação do Conhecimento* e a *Modelação de Sistemas de Raciocínio* até aos *Sistemas de Reconhecimento* e *Tratamento de Visão*.

A perspectiva de análise orientada para uma entidade única acaba, todavia, por se revelar limitada quando a atenção recai numa aplicação onde, normalmente, um conjunto de entidades actuam conjuntamente, muitas vezes dispersas no seio do sistema no qual estão integradas. Na resolução de problemas, a perspectiva individual de uma entidade com capacidades de actuação mantém-se. Contudo, a emergência de novas situações, conhecimento e re-

cursos, associada à distribuição de perícias por diferentes entidades, impõem um realinhamento de pensamento e tornam-se factores críticos na procura de soluções. A perspectiva individual tem, assim, que ser complementada por uma perspectiva de grupo e de distribuição dos recursos envolvidos.

O cenário actual da *Inteligência Artificial Distribuída* (IAD) (V.Lesser 1990) (A.Bond & L.Gasser 1988) emerge como uma integração das aproximações clássicas da gestão e manipulação de *Sistemas Baseados em Conhecimento* (SBC) com as da área científica conhecida por "Distributed Problem Solving". A investigação em IAD realça o desenvolvimento de aproximações que permitem subdividir o modelo de trabalho real; este considera o espaço e o domínio de conhecimento da aplicação a ser desenvolvida num conjunto de subaplicações, eventualmente distribuídas por diferentes locais, de acordo com as necessidades do modelo genérico e os imperativos de coordenação dos processos que se desenvolvem em torno dessas subaplicações, em termos de partilha de conhecimento e de controlo.

A máxima "dividir para reinar" pressupõe a ideia de criar, a partir de um, componentes mais pequenos, cuja complexidade no que respeita aos processos computacionais é inferior à do original. Este tipo de abordagem disponibiliza componentes de processamento mais simples, mais robustos e com maior facilidade de alocação nas plataformas computacionais.

A análise da granularidade das entidades resultantes desse processo de "divisão" do domínio do problema e a forma como este foi subdividido, associadas a questões de hardware e de distribuição de perícias e conhecimento, constituem o domínio de aplicação para técnicas de IAD. Gradualmente, com a IAD a caminhar para a "maturidade", há que definir os princípios e as teorias que suportam a cooperação e a coordenação de processos fre-

quentemente presentes em aplicações do género. Existe praticamente uma relação directa entre a dimensão e complexidade dos sistemas e a dificuldade de implementação da própria aplicação. A IAD disponibiliza as condições para o desenvolvimento de sistemas subdivididos por diversas entidades, com capacidade de resolução autónoma de problemas (variável de entidade para entidade) e possibilidades de interacção entre si.

Nos últimos tempos, tanto os aspectos metodológicos como os aspectos tecnológicos relacionados com a actividade de investigação em IA, particularmente em IAD, criaram uma parceria entre os investigadores de IA e os dos *Sistemas Multiagente* (SM). Da combinação da IA e dos SM surge um novo potencial que permite a utilização de ferramentas poderosas e sofisticadas na resolução de problemas, melhora a robustez dos sistemas perante situações de falha e de ruptura e gera o suporte necessário para aumentar a velocidade de execução destes (através da paralelização de programas), permitindo integrar metodologias, combinar múltiplos paradigmas computacionais e fontes de conhecimento e aumentar significativamente a modularidade dos sistemas (A.Bond & L.Gasser 1988) (N.Jennings, L.Varga, R.Aarnts, J.Fuchs & P.Skarek 1993).

Uma análise atenta a esta conjugação de domínios de conhecimento revela-nos uma perspectiva promissora quando aplicada no domínio da *Engenharia do Conhecimento* (EC). De facto, o desenvolvimento de SBC tem que lidar com vários e diferentes especialistas, em distintos domínios de aplicação, o que envolve frequentemente problemas relacionados com modelos de descrição, decomposição, comunicação, coordenação e coerência.

1.2 Aplicações Baseadas em Agentes

No início do presente capítulo foram apresentadas algumas questões que, em termos gerais, podem ser encontradas em cenários aplicacionais da vida real. A sua capacidade para o acolhimento de SBC distribuídos é significativa. Se acrescentarmos a necessidade destes sistemas desenvolverem acções cooperativas, então dirigimo-nos para modelos nos quais a conjugação dos aspectos científico-tecnológicos da IA e dos SM tem real aplicabilidade e assento.

Os SBC podem ser encarados como agentes (ou sistemas de agentes) no panorama dos SM. Um grupo de SBC actuando em conjunto, partilhando recursos, estratégias de decisão e conhecimento, possibilita a criação de um cenário típico para que se possa identificar, numa perspectiva global, como um SM.

Os SM estão relacionados com o desenvolvimento de soluções para problemas que envolvam entidades artificiais (os agentes) e com o modo como estas entidades se comportam, interactuam, evitam conflitos e desenvolvem formas de comportamento cooperativo, quer através da coordenação de planos de acção, quer pela partilha de conhecimento.

Em IA os objectivos são estados possíveis dos domínios de actividade dos agentes, reais ou abstractos, que permitem activar os seus mecanismos de raciocínio, conduzindo-os de forma a alcançarem os seus objectivos, através da tomada de decisões ou do planeamento de acções (A.Gadomski & J.Zytkow 1994). Em SM o alcance dos objectivos não é da exclusividade de um só agente, mas sim de uma comunidade de agentes. Nesta perspectiva, a necessidade de se criar e manter um meio para a interacção entre os agentes,

uma arquitectura robusta e fiável para o seu acolhimento, de se desenvolverem mecanismos de raciocínio, cooperação, resolução de conflitos e revisão de conhecimento é condição imprescendível para a criação e implementação de um SM. Na maioria dos SM, os elementos possuem perícias de resolução de problemas inter-relacionadas (mas distintas) e frequentemente têm que ser coordenados durante os seus processos de actuação. Este tipo de interacções é vital, devido às dependências entre as acções dos agentes, à necessidade de obedecerem a restrições de operacionalidade global e ao facto de, na generalidade dos casos, nenhum agente possuir a competência, o conhecimento ou os recursos suficientes para resolver o problema proposto (N.Jennings 1993*a*).

Vários motivos têm sido apresentados para a adopção da tecnologia baseada em agentes, sendo de realçar os seguintes (N.Jennings 1995*a*):

- **Abertura**.

  Os componentes dos sistemas não são conhecidos previamente, podendo sofrer modificações ao longo do tempo e apresentando grande heterogeneidade.

- **Complexidade**.

  O domínio de aplicação é muito vasto, sofisticado ou imprevisível, e a única maneira razoável de o endereçar é através do desenvolvimento de componentes modulares especializados, capazes de tratarem partes do domínio.

- **Distribuição de dados, controlo, perícia e recursos**.

  O domínio da aplicação implica a existência de um número distinto de entidades para a resolução do problema, estando estas física ou logicamente distribuídas e necessitando de interactuar umas com as outras

no sentido de resolverem os seus problemas e/ou executarem as suas tarefas.

- **Constituição de uma "metáfora natural"**.
  A noção de agente autónomo poderá ser a forma mais simples de se conceptualizarem funcionalidades em termos de software, podendo estas ser personalizadas de acordo com as preferências dos seus utilizadores.

- **Existência de sistemas de software antigo** ("legacy systems").
  É comum encontrar numa organização sistemas de software que vêm do passado em utilização corrente. Por forma a acompanhar as necessidades das organizações em que estão inseridos, estes sistemas de software sofrem alterações, alcançando, por vezes, custos de manutenção proibitivos. A longo prazo, como solução para este tipo de situações, poder-se-ão incorporar esses sistemas em ambientes comunitários cooperativos, tornando-os, assim, passíveis de serem explorados, eventualmente, por outras entidades para aí deslocadas.

O desenvolvimento de estratégias de coordenação, a exploração de novos modelos de cooperação e de novas técnicas para a resolução de conflitos, o estabelecimento de novos protocolos de comunicação e a investigação de novos mecanismos para suportar acções de verificação de consistência são apenas alguns dos tópicos que se poderiam considerar nesta linha de investigação e de aplicação tecnológica.

Esta emergência de múltiplas actividades relacionadas com a tecnologia de agentes demonstra claramente que algo de diferente está a suceder no panorama das Ciências da Computação. As futuras aplicações envolverão sistemas distribuídos complexos, sistemas de informação baseados em sistemas de re-

de, sistemas de interface homem-máquina, SBC cooperativos, entre outros, que constituirão, sem dúvida, durante os próximos anos, áreas científico-tecnológicas emergentes, nas quais a tecnologia baseada em agentes assumirá um papel central.

Iremos assistir, nos próximos tempos, a uma actividade fervilhante, em que as contribuições científicas e técnicas vão surgir em massa, abordando inúmeros temas relacionados com esta tecnologia. De realçar, a título de exemplo, áreas como as seguintes:

- arquitecturas de agentes;
- vida artificial;
- técnicas e meios de comunicação;
- coordenação, cooperação e resolução de conflitos entre agentes;
- aprendizagem e adaptabilidade em SM;
- organização e representação de conhecimento;
- aplicações práticas de SM;
- avaliação de desempenhos de SM;
- alocação de recursos em SM;
- sistemas de interface inteligentes.

Neste rápido processo de emergência de uma nova ciência e tecnologia, as opiniões dividem-se, naturalmente, no que respeita à melhor maneira de encarar a nova realidade e à criação das suas bases. A máxima "dividir para

reinar" é também abordada por B.Hayes-Roth (1995) como uma estratégia quase destinada ao insucesso neste processo, já que, ao desenvolverem-se componentes independentes, serão tomadas decisões implícita ou explicitamente imcompatíveis, as quais, aliadas ao pouco cuidado no planeamento e implementação das infra-estruturas partilhadas, afectarão assim o grau de interoperacionalidade dos componentes do sistema. Nesse trabalho, propõe-se que as estratégias de investigação sejam direccionadas para o desenvolvimento de arquitecturas compreensivas de agentes e para o suporte à integração e interoperacionalidade de componentes.

As aplicações baseadas em SM podem ser divididas em três grandes áreas (V.Lesser 1995):

- **Aplicações distribuídas para a avaliação de situações**.
  Abordam a forma como os agentes com diferentes "esferas" de conhecimento e mecanismos de controlo partilham as interpretações de uma situação, no sentido de chegarem a respostas e a explicações mutuamente consistentes e compreensivas.

- **Aplicações distribuídas para o planeamento e alocação de recursos**.
  Abordam a forma como os agentes devem coordenar as suas acções de planeamento, de modo a evitarem e resolverem eventuais conflitos sobre a alocação de recursos, e por forma a maximizarem os resultados do próprio sistema.

- **Aplicações de *Sistemas Periciais* (SP) distribuídos**.
  Abordam a forma como os agentes se inter-relacionam, contribuindo para a resolução conjunta de problemas, com as suas perícias e critérios.

O último ponto exposto faz referência a um dos principais objectivos a alcançar com a presente tese: a concepção e o desenvolvimento de um ambiente computacional com a capacidade de integrar um conjunto de SBC distribuídos, susceptíveis de cooperarem na resolução de problemas.

1.3 Motivação e Objectivos da Tese

A actividade do homem é eminentemente cooperativa. Em domínios que envolvam situações de diagnóstico, planeamento, monitorização e controlo é frequente emular idêntico comportamento na resolução de problemas, materializado pelos especialistas nos domínios de conhecimento requeridos. Este tipo de situações também ocorre no domínio da EC.

Existem casos em que os problemas apresentados a um SBC não são por este resolvidos. Os motivos do insucesso estão, não raras vezes, associados ao facto de o sistema, só por si, não possuir o conhecimento e a perícia necessários à sua resolução. A solução do problema pode implicar a utilização de conhecimento de diferentes áreas, o que só uma equipa pluridisciplinar pode disponibilizar. Uma eventual forma de se ultrapassar tal situação poderia passar pela criação das estruturas necessárias ao desenvolvimento de um meio cuja natureza permitisse acolher os SBC nas diferentes áreas de conhecimento em jogo, e onde estes pudessem interactuar. Desta forma, os problemas para os quais não se conseguisse uma solução enquanto sujeitos à intervenção de um único sistema poderiam ser resolvidos pela actuação conjunta de vários tipos de sistemas.

Com base neste tipo de situações, surgiu a ideia principal do presente projecto, o desenvolvimento do sistema BeAble. Assentava, basicamente, na

criação de um ambiente computacional distribuído, onde fosse possível criar e manter as estruturas necessárias para reflectir o comportamento associado a essas entidades cooperativas. Em termos gerais, tal ambiente computacional deveria apresentar:

- características adequadas para acolher problemas oriundos de áreas de diagnóstico, controlo, monitorização e classificação;
- um meio onde fosse possível integrar diversos SBC, com diferentes competências, no qual estes pudessem desenvolver processos de cooperação entre si;
- facilidades na distribuição desses mesmos sistemas por diferentes plataformas computacionais situadas em locais geograficamente distintos;
- uma arquitectura facilmente compreensível, de princípios operacionais simples, na qual não tivessem sido descurados os aspectos relacionados com as infra-estruturas de comunicação e partilha de recursos;
- um ambiente capaz de suportar a resolução simultânea de vários problemas propostos pelos utilizadores e que exijam a conjugação de conhecimento pluridisciplinar.

Complementarmente, o estudo de toda a problemática associada à criação de ambientes comunitários para agentes de natureza heterogénea constitui um motivo bastante forte para o desenvolvimento de uma acção desta natureza. Assim, o estudo e a investigação de aspectos científicos e tecnológicos directamente relacionados com aplicações de agentes inteligentes, envolvendo questões como comunicação, coordenação, cooperação, resolução de conflitos

e verificação de consistência, foram também despoletados de imediato por tais acções.

Os objectivos deste trabalho, já parcialmente enunciados, visam a implementação de um novo ambiente computacional que permita integrar um grupo de SBC distribuídos, de características heterogéneas, capazes de desenvolver, assincronamente, acções de cooperação com vista à resolução conjunta de problemas: o sistema BeAble.

Adicionalmente, foi definido para o projecto o seguinte conjunto de objectivos específicos:

- planear, desenhar e desenvolver uma ferramenta computacional de aplicação genérica, um ambiente computacional distribuído para a implementação de comunidades de agentes;
- criar uma filosofia de cooperação entre agentes, não numa perspectiva convencional, mas sim entre SM, desenvolvendo e explorando as diversas formas de interacção entre "nichos" especialistas de agentes cognitivos, a fim de defender a definição e distribuição de competência e graus de especialização em comunidades de agentes;
- alargar o horizonte de consulta através da conceptualização e desenvolvimento de uma "ponte" entre a arquitectura multicomunidade do sistema e as actuais ferramentas computacionais de "browsing" da Internet;
- criar mecanismos de adaptabilidade para os agentes do sistema, por forma a que estes acompanhem efectivamente a dinâmica do meio e da situação com que no momento estiverem a lidar;

- flexibilizar o nível de intervenção dos agentes, permitindo-lhes lidar com mais do que um processo de cada vez, não os vinculando a situações monoprocessuais, dando-lhes a capacidade de atender simultaneamente mais do que um utilizador e rentabilizando-lhes o desempenho.

1.4 Organização da Tese

Além do actual capítulo, que inclui uma pequena introdução sobre modelos organizacionais e algumas considerações gerais sobre a tecnologia de SM, a presente tese apresenta um conjunto de sete capítulos, organizados da seguinte forma:

- **Capítulo 2**

  Computação Baseada em Agentes.

  Apresentação do actual panorama da *Computação Baseada em Agentes* (CBA). Caracterização e comportamento de um agente genérico. Conceitos e características dos SM.

- **Capítulo 3**

  Sistemas de Agentes Baseados em Conhecimento.

  Abordagem da nova geração de SBC e análise da sua correspondência com os sistemas orientados por agentes. O conhecimento e sua representação segundo a perspectiva da nova geração de SBC. Acções de coordenação, cooperação, resolução de conflitos e verificação de consistência em SM. *Sistemas Baseados em Quadros Negros* (SBQN) e sua influência no modelo adoptado para o sistema.

- **Capítulo 4**

  O Modelo Computacional do Sistema.

  Apresentação do modelo computacional do sistema, características e funcionalidades principais, arquitectura, ambiente distribuído e definição das classes de agentes que podem ser integradas no ambiente do sistema. Aspectos de implementação e instalação do sistema. Descrição das actividades de administração e controlo do sistema. Apresentação das características, arquitectura e funcionalidades dos agentes de supervisão, monitorização e de suporte operacional.

- **Capítulo 5**

  A Resolução de Problemas.

  Apresentação, descrição e caracterização das classes de agentes envolvidas directamente nos processos de resolução de problemas. Modelos de coordenação, cooperação e aprendizagem dos agentes cognitivos.

- **Capítulo 6**

  Multicomunidades de Agentes.

  Ampliação do modelo do sistema para obtenção de um ambiente multicomunidade de agentes. Caracterização e descrição das funcionalidades do agente de encaminhamento. Análise dos modelos de comunicação intercomunidades.

- **Capítulo 7**

  Aplicações.

  Apresentação dos domínios de aplicação do sistema e dos processos relacionados com o desenvolvimento de quatro aplicações, para análise e verificação do comportamento do sistema em áreas de conhecimento relacionadas com problemas de classificação, controlo e diagnóstico.

- **Capítulo 8**

  Conclusões e Trabalho Futuro.

  Síntese dos pontos mais relevantes do trabalho desenvolvido durante a realização do projecto do sistema BEABLE. Análise do estado actual do sistema, em termos de características funcionais, arquitectura e aplicações, e indicação de potenciais linhas de investigação futura, por forma a dar continuidade ao trabalho realizado até ao momento.

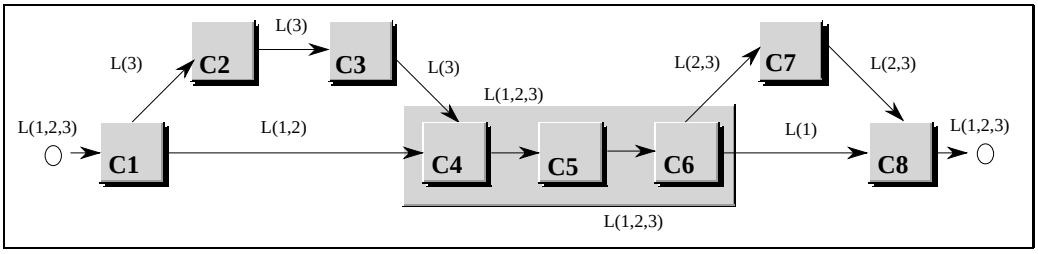


Figura 1.1: Linhas de Orientação de Leitura da Tese.

Três linhas de orientação podem ser adoptadas na leitura desta tese (Figura 1.1). A primeira, constituída pelos Capítulos 1, 4, 5, 6 e 8, disponibiliza uma visão orientada exclusivamente para os aspectos científicos e técnicos do sistema BEABLE, não apresentando, contudo, os conceitos e os fundamentos de base que suportam as características e funcionalidades do próprio sistema. A segunda, que considera todos os capítulos da anterior linha, é complementada com o Capítulo 7, visando, não só apresentar os aspectos relacionados com a concepção, desenvolvimento e caracterização do sistema, mas também revelar as suas potenciais áreas de aplicação, bem como apresentar algumas das aplicações desenvolvidas para teste e validação do sistema BEABLE. Por fim, a terceira linha de orientação engloba todos os capítulos da tese, permitindo ao leitor dominar a sua génese: os motivos e objectivos da sua realização; uma perspectiva actual da tecnologia de agentes e dos SBC; os conceitos, os

fundamentos e os modelos que suportam o ambiente computacional do sistema, assim como os aspectos disponibilizados pelas outras linhas de orientação de leitura.

Capítulo 2

Computação Baseada em Agentes

Apresentação do actual panorama da Computação Baseada em Agentes. Caracterização e comportamento de um agente genérico. Conceitos e características dos Sistemas Multiagente.

Nos últimos anos, a IA reforçou significativamente o seu impacto nas mais diversas áreas ou domínios de conhecimento, quer em termos da geração de novos conceitos e metodologias, em domínios como os sistemas periciais, a aprendizagem máquina ou a robótica, quer mesmo ao nível dos modelos de representação de conhecimento. Uma área, porém, há que destacar das demais: a da CBA (M.Wooldridge & N.Jennings 1995*a*) (M.Wooldridge & N.Jennings 1995*b*).

O conceito de "agente" surge-nos nas mais diversas áreas, desde o desenvolvimento de interfaces adaptativos até à implementação de SP distribuídos, ou mesmo ao desenvolvimento de sistemas para a procura e filtragem de in-

formação em ambientes como a Internet. O leque de domínios ou áreas do conhecimento aplicacionais em que esta tecnologia se aplica é extremamente vasto e está em contínua expansão.

O aparecimento, em 1995, da primeira conferência internacional de SM - ICMAS´95[1], a organização da primeira WDAI[2] e a realização, no ano transacto, da primeira conferência de aplicações práticas de agentes - PAAM´96[3], acompanhados pela realização anual da *workshop* europeia MAAMAW[4], desde 1989, e por muitas outras realizações internacionais do género, são, com certeza, eventos reveladores do grande interesse da comunidade científica pelo domínio da CBA. Mais recentemente, esta tendência é também partilhada por sectores da indústria e dos serviços.

Apesar da ocorrência deste gigantesco "boom", ainda restam questões em aberto, linhas de investigação por explorar e, naturalmente, muitas aplicações práticas para realizar. Mas talvez a questão mais controversa, e ao mesmo tempo irrecusável, que tem persistido no domínio da CBA seja a definição do próprio conceito de agente [5]. Esta problemática tem sido frequentemente abordada na tentativa de se chegar a um consenso com respeito à definição de agente. As mensagens que habitualmente circulam na lista de correio electrónico "agentes@sun.lab.com" - uma das listas de discussão por excelência dos agentes de software - têm comprovado que a dilucidação do conceito de agente continua a ser, pelo menos até ao momento, "uma questão

[1]"The First International Conference on Multi-Agent Systems", São Francisco, EUA, Junho, 1995.

[2]"First Australian Workshop on Distributed Artificial Intelligence", Camberra, Australia, Novembro, 1995.

[3]"The First Conference on Practical Applications of Multi-Agent Systems", Londres, Inglaterra, Abril, 1996.

[4]"Modelling Autonomous Agents in a Multi-Agent World"

[5]Em L.Foner (1993) pode-se encontrar um trabalho que apresenta e explora, com certo detalhe, a questão da definição de agente.

em aberto". Os investigadores, e os próprios utilizadores, divergem sistematicamente na definição do conceito, muitas vezes influenciados pelo domínio de conhecimento em que estão inseridos ou pelo tipo de aplicação em causa. Complementarmente, a definição é também, não raras vezes, afectada pela dimensão e pelo grau de granularidade da entidade que se pretende caracterizar como agente. O termo é cada vez mais popular a todos os níveis: na comunidade científica, na indústria de software, nos media e em muitos outros domínios de interesse, originando diversos tipos de abordagem, bem como motivos de discussão e reflexão acerca do tema.

Entretanto, o número crescente de aplicações baseadas em agentes vem revelar a grande dinâmica que o domínio da CBA apresenta nos dias de hoje. O espectro aplicacional é bastante lato, tal como se pode constatar pela seguinte relação de aplicações:

- Ambientes distribuídos para a integração de SP cooperativos (T.Wittig 1992) e para suporte à cooperação de agentes inteligentes (E.Oliveira & F.Mouta 1993) (O.Belo & J.Neves 1996*a*).
- Arquitecturas para a integração de múltiplos processos de raciocínio (D.Riecken 1994*b*).
- Sistemas de apoio à concepção e desenvolvimento de SP (A.Ovalle 1994).
- Sistemas distribuídos para apoio a processos de tratamento médico (J.Huang, N.Jennings & J.Fox 1995*b*) e para ajuda à monitorização de tratamentos médicos intensivos (B.Hayes-Roth 1993).
- Assistentes pessoais artificiais com capacidades de aprendizagem (T.Mi-

tchell, R.Caruana, D.Freitag, J.McDermott & D.Zabowski 1994) e sistemas de suporte à aprendizagem (T.Selker 1994) em ambientes multiagente heterogéneos (P.Edwards & W.Davies 1993).

- Linguagens para o desenvolvimento e a implementação de sistemas de agentes, como a Model-K (W.Karbach & A.Vobeta 1993), a KQML (T.Finin, R.Fritzon, D.McKay & R.McEntire 1994), a April (F.McCabe & K.Clarck 1995) e a OASIS (F.Cheong 1992).

- Aplicações para ambientes de simulação (D.Gagné, J.Desbiens & G.Nault 1994) (P.Mossinger, D.Polani, R.Spalt & T.Uthmann 1995)(A.Ribeiro, V.Alves & J.Neves 1993).

- Aplicações no domínio das redes de computadores orientadas para a realização de acções de filtragem de informação (A.Reinhardt 1994) e para a gestão de redes telefónicas (G.Kiss 1991).

- Sistemas de ajuda na pesquisa de informação e na gestão de serviços disponibilizados na Internet (F.Cheong 1996).

- Aplicações de agentes móveis para suporte a actividades de comércio electrónico (C.Munday, J.Dangedej, T.Cross & D.Lukose 1995).

Esta lista de ambientes, arquitecturas, linguagens e aplicações poderia ser generosamente alargada, porém, os exemplos referidos são suficientes para demonstrar a grande diversificação dos domínios de aplicação da CBA ao longo dos últimos anos, e como base para endereçar novas áreas de estudo e de aplicação. A leitura de D.Riecken (1994*a*), A.Gadomski & J.Zytkow (1994), N.Jennings (1995*a*) e B.Hayes-Roth (1995) providenciará certamente um maior leque de informação acerca da CBA, além de possibilitar uma

melhor identificação dos problemas e do pensamento actual nesse domínio de conhecimento.

2.1 Agentes

Ao designarmos uma entidade computacional - uma "peça" de software - como agente estamos a identificar um programa de computador com um domínio de aplicação específico e bem definido, capaz de actuar como uma espécie de assistente artificial autónomo, independente e com capacidades de decisão relativamente ao meio no qual está inserido. Estas entidades, de uma forma geral, têm a capacidade de observar o meio em que estão inseridas, reagir a eventuais acontecimentos que nele ocorram, aplicar as suas estratégias de resolução de problemas de acordo com o conhecimento e a perícia que possuam e reflectir no ambiente o resultado dos seus processos internos de análise e decisão.

O desempenho de um agente está dependente, de entre um conjunto alargado de condicionantes, da sua perícia, do conhecimento que possui acerca do domínio da aplicação, das suas capacidades adaptativas - muitas vezes associadas às capacidades de aprendizagem - e do grau de autonomia que lhe foi concedido pelo seu construtor.

Não é fácil o estabelecimento de um padrão comparativo para estas entidades, pelo simples facto de que foram, não raras vezes, especialmente criadas com o objectivo de desenvolverem um "papel" particular num domínio de aplicação específico. Assim, não é de surpreender a existência de uma grande variedade de tipos de agentes, apresentando grandes diferenças em termos orgânicos e funcionais.

É possível encontrar diversos tipos de agentes espalhados por inúmeras áreas de aplicação, todavia, um dos domínios onde a tecnologia da CBA tem tido uma crescente implantação é o da Internet. "Web Robots", "Spiders", "Wanderers" and "Worms" são apenas algumas das designações de agentes que se podem encontrar no domínio da Internet [6], relacionados com a realização de um leque de serviços muito alargado (F.Cheong 1996).

2.1.1 Conceitos Gerais

No domínio da IA, os agentes têm sido encarados numa perspectiva mais ambiciosa; não são apenas entidades possuidoras de uma "carteira" de acções despoletadas a partir de um conjunto de estímulos ou instruções, mas entidades artificiais candidatas à representação, à emulação do comportamento humano em diferentes domínios do conhecimento, capazes de raciocinar e de agir da mesma forma que os humanos, isto é, actuando como assistentes inteligentes (R.Brooks 1991).

O projecto Oz [7] (J.Bates 1992*a*) (J.Bates 1992*b*), em curso na Universidade de Carnegie Mellon, é um bom exemplo desses objectivos, já que as suas principais linhas de acção se desenvolvem em torno da implementação de um ambiente para a simulação de um mundo capaz de acolher um conjunto de criaturas artificiais - os "Woggles" - semelhantes aos humanos em termos comportamentais, de animação, com capacidades para interactuarem em tempo real, e com as quais os humanos poderão eventualmente trabalhar ou simplesmente divertir-se.

[6] http://info.webcrawler.com/mak/projects/robots/robots.html
[7] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/oz/web/oz.html

No presente trabalho [8], todavia, não se considerarão os agentes como entidades artificiais possuidoras de comportamentos equivalentes ao ser humano, mas somente como sendo susceptíveis de reflectir, de alguma forma, partes desse comportamento.

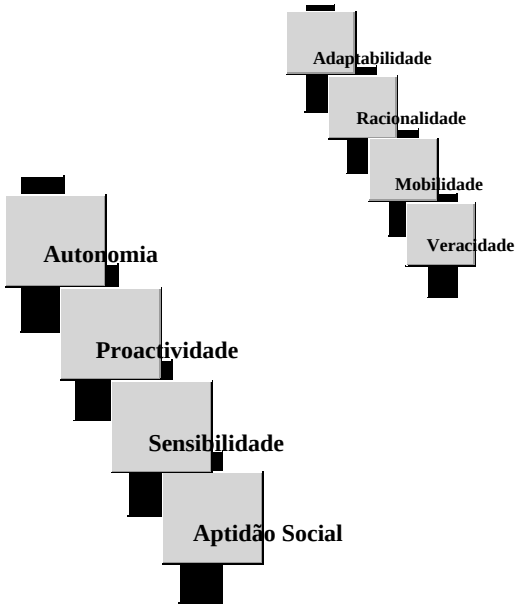


Figura 2.1: Características Genéricas de um Agente.

De uma forma simplificada, os agentes podem ser caracterizados com base nas seguintes propriedades (M.Wooldridge & N.Jennings 1995*a*) (Figura 2.1):

- **Autonomia**.

  Devem ser capazes de manipular e resolver a maioria dos seus proble-

[8] Os agentes, neste trabalho, são entidades artificiais classificadas de acordo com a sua aptência, perícia e conhecimento. Alguns poderão ser equivalentes a SBC, com capacidades de comunicação suficientes para desenvolver acções conjuntas com vista à resolução de problemas, sanar conflitos, realizar processos de verificação de consistência sobre o conhecimento que possuem ou, em alguns casos, revelar mesmo capacidades de aprendizagem sobre os próprios processos que realizam. Noutros casos, serão plataformas de interface para os utilizadores ou simples entidades de suporte operacional. A apresentação da sua classificação e características será efectuada nos Capítulos 4 e 5.

mas, sem sofrer directamente a intervenção humana ou de outros agentes, devendo ainda possuir um grau de controlo acima das suas próprias acções e do seu estado de conhecimento interno (C.Castelfranchi 1995).

- **Aptidão social**.

  Devem ser capazes de interactuar, quando julgarem apropriado, com outros agentes ou humanos, por forma a completarem os seus próprios processos de resolução de problemas ou a ajudarem-se mutuamente.

- **Sensibilidade**.

  Devem ser sensíveis à informação que circula no ambiente em que estão integrados e responder, em tempo útil, a eventuais acontecimentos que nele possam ocorrer.

- **Proactividade**.

  Não devem limitar-se actuar em resposta a eventos que ocorram no seu ambiente de trabalho, mas antes exibir um carácter oportunístico, isto é, serem capazes de tomar a iniciativa quando acharem conveniente.

Complementarmente, são desejáveis num agente as características a seguir enunciadas:

- **Adaptabilidade**.

  Capacidade para modificar o seu comportamento ao longo do tempo, a fim de responder às mutações das condições ambientais ou à alteração do seu próprio estado de conhecimento.

- **Racionalidade**.

  Suposição de que actua no sentido de alcançar os seus objectivos, agindo somente em prol de uma "boa causa".

- **Veracidade**.

  Suposição de que não comunica informação falsa.

- **Mobilidade**.

  Capacidade de mudar a sua própria localização.

Os graus de autonomia, perícia e conhecimento são factores preponderantes na classificação de um agente. Nas sociedades humanas conseguem identificar-se com uma certa facilidade diferentes tipos de acções e reacções, que permitem determinar e categorizar o comportamento dos indivíduos nela inseridos. Da mesma forma, é de esperar que um agente seja capaz de revelar diferentes tipos de comportamento, de acordo com as condições do meio em que está inserido. Espera-se, assim, que possa realizar mais do que simples tarefas de resolução de problemas, previamente estabelecidas e regulamentadas pelos padrões da comunidade. O agente deve ser capaz de manifestar outros tipos de comportamento. A capacidade de resolução de problemas deve ser complementada pela sua capacidade de comunicação, de interacção com o meio, de raciocínio, de cooperação e de aprendizagem. Não raras vezes, espera-se que o agente se adapte ao tipo de problemas em desenvolvimento no seu ambiente e, oportunamente, contribua de forma efectiva para a sua resolução. Esta aptidão revelará a sua capacidade de reacção e o seu carácter oportunístico. Complementarmente, as contribuições que faz para a comunidade constituirão uma medida do seu posicionamento no seio desta. A uma entidade com este tipo de características atribuir-se-á a designação de agente inteligente.

2.1.2 Tipos de Agentes

A definição de uma taxonomia em suporte da categorização de agentes não tem sido tarefa fácil nem muito frutuosa. Os domínios de aplicação e a definição das competências atribuídas aos agentes implicam muitas vezes que os seus construtores os classifiquem dentro de uma nova classe. Também pode suceder que, no desenvolvimento de uma aplicação baseada em agentes, ocorram processos de reunião e de importação de características normalmente associadas a determinada "classe" de agentes, dando origem ao aparecimento de agentes com características híbridas. Designações como *agentes de interface*, *agentes autónomos*, *agentes cooperativos* e *simbióticos* (R.Beale & A.Wood 1994) são vulgares na terminologia usada em CBA.

O desenvolvimento e implementação de agentes de interface para suporte aos processos de interacção entre utilizadores e máquinas constituem manifestamente uma das áreas mais activas na aplicação de agentes inteligentes. Tutores inteligentes, agentes de navegação, agentes para a procura de informação e assistentes inteligentes para o planeamento de tarefas são apenas alguns exemplos de potenciais domínios de aplicação para o desenvolvimento de agentes de interface (J.King 1995).

Os agentes de interface, muitas vezes encarados como assistentes personalizados, mudam radicalmente o estilo de interacção homem-máquina. Os utilizadores deste tipo de agentes delegarão algumas das suas tarefas nestas entidades (personalização), que evoluirão de acordo com o perfil do utilizador, através de um processo de adaptação, aprendizagem e cooperação com outros agentes (P.Maes 1994) (Y.Lashkari, M.Metral & P.Maes 1994).

O desenvolvimento de interfaces adaptativos surge, assim, como uma das

áreas de potencial aplicação para este tipo de agentes, porém, não é apenas ao nível do interface homem-máquina que podemos encontrá-lo. O desenvolvimento e utilização destas entidades como forma de assegurar um meio que garanta a interoperacionalidade de programas, assegurando a comunicação entre ambientes de software altamente heterogéneos, potencia-se como outra das suas áreas de aplicação.

2.1.3 Modelo Genérico de um Agente

Nas secções anteriores figuram algumas das funções normalmente associadas a agentes inteligentes, descrições sobre eventuais comportamentos que estes possam apresentar e uma caracterização de alguns dos seus tipos. Desse rol de características e modelos, e independentemente do tipo de agente, todos apresentam uma espécie de "esqueleto" comum, isto é, todos observam o ambiente envolvente através das suas capacidades sensoriais e definem comportamentos relativamente a eventos que nele ocorram de acordo com os seus modelos de decisão internos.

Os agentes, reagindo a estímulos provenientes de eventos que ocorrem no seu ambiente, ou mesmo actuando segundo uma qualquer função de utilidade, apresentam, de uma forma geral, um mesmo ciclo de vida, constituído por três fases elementares: percepção, decisão e acção (Figura 2.2).

Uma possível arquitectura para agentes (Figura 2.3) pode, por conseguinte, ser definida pelos seguintes módulos funcionais:

- **Sensores**.

  Dispositivos que permitem ao agente analisar o ambiente no qual está

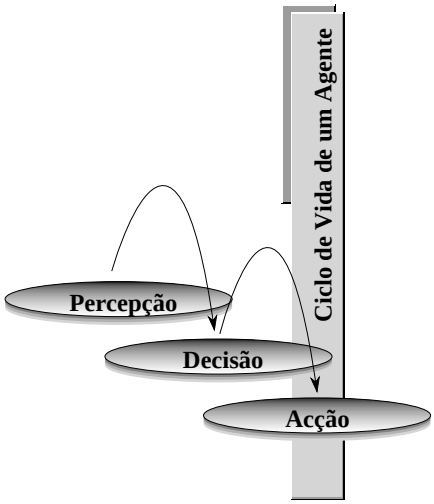


Figura 2.2: Ciclo de Vida Básico de um Agente.

integrado e colher a informação que achar relevante para os seus processos de decisão internos.

- **Mecanismos de Decisão**.
  Normalmente constituídos por mecanismos de inferência que, baseados nos modelos de comportamento e de representação do conhecimento, determinam o tipo de acção a executar pelo agente.

- **Modelos de Comportamento**.
  Estruturas de conhecimento que definem o modo de actuação do agente de acordo com o seu estado actual e as observações realizadas sobre o ambiente.

- **Modelos de Representação**.
  Estruturas que traduzem o conhecimento do agente acerca do seu universo de discurso e do ambiente no qual se integra, bem como sobre

outros agentes que partilhem do mesmo ambiente.

- **Actuadores**.

  Dispositivos orientados para reflectirem sobre o ambiente do agente o resultado de acções realizadas internamente por este.

- **Interface**.

  Módulo responsável pela interacção agente-utilizador, permitindo o desenvolvimento de "diálogos" entre estes.

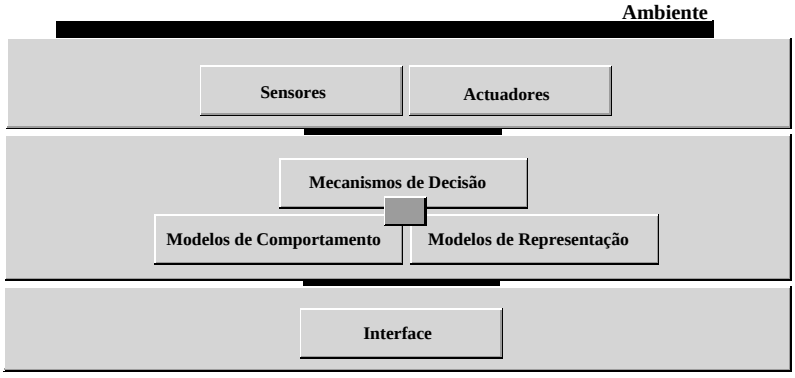


Figura 2.3: Arquitectura Funcional de um Agente.

A Figura 2.4 apresenta uma breve descrição do comportamento de um agente em reacção a um evento ocorrido no seu ambiente [9].

[9] O algoritmo descreve basicamente o comportamento de um agente genérico. Nele podem identificar-se quatro fases de acção distintas: 1) arranque do agente - fase na qual o agente prepara o ambiente local e faz a leitura das condicionantes de percepção (*Condicionantes*), isto é, as condições a que deve obedecer durante a fase de percepção; 2) percepção - fase na qual o agente analisa o ambiente na tentativa de encontrar informação que lhe possa interessar (*Percepção*) e que preencha os requisitos impostos pelas condicionantes anteriores; 3) decisão - fase na qual o agente analisa a percepção e decide qual a acção a realizar e que tipo de estratégia aplicar; 4) acção - fase na qual o agente actualiza a base local com o conhecimento adquirido (*Novo-Conhecimento*), reflectindo-o, também, sobre o ambiente.

```
procedimento Agente-Genérico
    inicialização-arranque-agente;
    leitura-condicionantes-percepção(Condicionantes)
    repetir
        repetir
            analisar-ambiente(Condicionantes,Percepção)
        até Percepção
        analisar-percepção(Percepção,Processar)
        se Processar então
            actualizar-estado-interno(Percepção,Novo-Estado,Estratégia);
            aplicar-estratégia-decisão(Novo-Estado,Estratégia,
                    Resultado,Novo-Conhecimento)
            se Resultado então
                actualizar-ambiente(Novo-Conhecimento)
            fim-se
            actualizar-estado-interno(Novo-Conhecimento)
        fim-se
    até Fim-Execução
fim-procedimento.
```

Figura 2.4: Descrição do Comportamento de um Agente.

Um agente deve possuir a capacidade de lidar com diferentes tipos de informação, relacionando conhecimento heterogéneo, incerto ou incompleto. Os cenários com que se depara poderão ser muito variados, devendo o agente ser capaz de atingir a auto-satisfação. As estratégias a utilizar deverão estar de acordo com os seus modelos internos de percepção, decisão e acção.

2.1.4 A Natureza de um Ambiente

O ambiente no qual um agente está integrado tem um papel preponderante na sua conduta, na evolução do seu estado de conhecimentoe nas acções que este pode desenvolver. As alterações do ambiente, como resultado da modificação das estruturas de conhecimento nele armazenadas, e o tráfego de informação que normalmente aí ocorre geram eventos, traduzindo-se em estímulos que, ao

serem detectados pelo agente, através de acções de sensorização, podem levar à alteração do seu estado de conhecimento. As características de um ambiente afectam, por conseguinte, as estruturas de conhecimento e os modelos de comportamento dos agentes, pelo simples facto de que a generalidade dos processos a desencadear por um agente depende das suas capacidades de observação do ambiente.

Os ambientes podem ser categorizados da seguinte forma (S.Russel & P.Norvig 1995):

- **Acessíveis/inacessíveis**.
  Conforme o sistema sensorial do agente lhe permite obter, ou não, toda a informação relacionada com o estado do ambiente.

- **Determinísticos/não determinísticos**.
  Consoante o próximo estado do ambiente possa ou não ser completamente determinado a partir do seu estado corrente e das acções desencadeadas pelos agentes.

- **Episódicos/não episódicos**.
  Num ambiente episódico, a experiência do agente está dividida em episódios. Estes não dependem das acções realizadas em episódios anteriores.

- **Dinâmicos/estáticos**.
  Conforme o ambiente sofra ou não modificações no seu estado de conhecimento enquanto um agente está a operar.

- **Discretos/contínuos**.
  Dependendo da existência ou não de um conjunto limitado de obser-

vações e de acções distintas e claramente definidas, assim o ambiente será, respectivamente, discreto ou contínuo.

A natureza de um ambiente influencia directamente a forma como o agente o observa, impondo-lhe a posse de modelos de comportamento flexíveis, dinâmicos e efectivos como condição para usufruir do conhecimento que possa conter. Por exemplo, um ambiente acessível a um agente evita-lhe a necessidade de armazenar em memória a informação acerca da evolução do mesmo, o que reduz significativamente as suas necessidades de memorização de informação. Por outro lado, se o ambiente em que o agente está integrado apresentar características dinâmicas, este, para além dos seus processos internos, tem que, concorrentemente, observar a evolução do estado do ambiente, caso contrário, arrisca-se a que a acção que irá desencadear não tenha qualquer tipo de reflexo neste, ou, no pior dos casos, tenha um impacto inesperado, ou mesmo negativo. É importante, pois, realizar uma análise detalhada e cuidada das características do ambiente antes de se tomar qualquer decisão com respeito ao processo de concepção e implementação de um agente. Diferentes tipos de ambiente requerem diferentes tipos de agentes. Só assim os problemas poderão ser abordados de forma eficiente e o ambiente manipulado de forma efectiva pelos seus agentes.

2.1.5 Tipos de Comportamento

Diferentes são os tipos de comportamento que podem ser apresentados pelos agentes durante o processo de resolução de um problema. Efectivamente, pode-se analisar o comportamento de um agente de acordo com os seguintes aspectos (Y.Demazeau & J.Muller 1990*a*):

- local onde a tarefa será executada pelo agente:
  - **localmente** - ao nível da sua estrutura interna;
  - **globalmente** - no ambiente em que está integrado;
- capacidade do agente para executar por si próprio uma tarefa específica.

De acordo com a categorização anterior, e tomando em consideração que no ambiente em que o agente está integrado coexistem outros agentes, podem-se também descrever os agentes segundo as suas capacidades de coabitação, colaboração e distribuição.

As características de coabitação de um agente estão directamente relacionadas, não só com a sua capacidade de executar uma tarefa por si próprio, mas também com a forma como a acompanha quando executada por outros agentes. A possibilidade de integração de um agente num ambiente em que outros coexistem, partilhando a sua informação ou interactuando com eles, traduz a sua maior ou menor capacidade de sobrevivência nesse ambiente.

As características cooperativas de um agente estão relacionadas com as acções que ele ou outros agentes podem desenvolver, quando só por si não conseguem resolver o problema que lhes foi apresentado. Tal circunstância pode dever-se ao facto de outros agentes terem executado essa tarefa mais rapidamente e/ou de forma mais eficiente. A capacidade de colaboração de um agente é analisada segundo aspectos que se prendem com questões relativas à distribuição de tarefas por diferentes tipos de agentes. Porém, não raras vezes, a distribuição de tarefas é dificultada pela necessidade de determinar qual o agente que melhor pode executar cada uma delas.

A intersecção dos domínios de aplicação dos agentes, com o seu reflexo na perícia e conhecimento destes, ajuda a obter sistemas que, de um ponto de vista global, se tornam mais robustos, seguros e eficazes. Em alguns casos, podem ser propostos aos agentes problemas que, sendo apresentados como peça única, são passíveis de gerar situações que aqueles não têm capacidade nem meios para resolver. Esses problemas necessitam de ser fragmentados de acordo com o conhecimento e a perícia dos agentes disponíveis. A sua divisão em partes, ou problemas mais pequenos, e a sua distribuição pelos diversos agentes permitirão uma abordagem mais flexível e efectiva do problema global.

Os agentes observam o ambiente no qual estão integrados através de mecanismos sensoriais, acção que se traduz numa das formas de comunicação indirecta entre eles, comum em situações em que há partilha de informação, através do uso de estruturas de conhecimento localizadas no ambiente de trabalho. A comunicação directa é, porém, uma das formas mais vulgares que utilizam na troca de informação para solicitarem a realização de uma tarefa em particular ou passarem o controlo de operações em curso a outro(s) agente(s).

Nos processos de troca de informação, os agentes podem adquirir informação relacionada com o conhecimento e a perícia de outros agentes. Os tipos de conhecimento que podem trocar entre si vão da simples troca de informação necessária à resolução de um problema até trocas de informação face à detecção de situações ambíguas, incertas ou mesmo contraditórias. Caso ocorram situações de conflito, os agentes nesta situação deverão comunicar entre si a fim de as resolverem. Este processo passa, frequentemente, pelo desenvolvimento de processos de negociação entre agentes, com o objectivo

de se chegar a um ponto de entendimento. Torna-se necessário que cada um explique e justifique a sua posição. A troca de soluções intermédias ou finais é desencadeada quando um agente, só por si ou em conjunção com outros, alcança um estado de satisfação parcial ou total. Digamos que o estado alcançado constitui uma base para a continuação do processo (solução parcial) ou para a resolução do problema (solução final). Pode-se, eventualmente, definir uma estratégia que, partindo das várias propostas de solução de um problema, as consolide numa única.

Na eventualidade da existência de mais do que uma solução, independentemente de esta ser parcial ou global, os agentes, quando a operar conjuntamente, devem decidir por qual das soluções propostas optar. Para isso, terão que encontrar uma plataforma de entendimento, a fim de continuarem com o processo de resolução do problema. Podem, eventualmente, ser desencadeados processos de negociação como forma de resolução de conflitos.

2.2 Sistemas Multiagente

Os SM (M.Wooldridge & N.Jennings 1995*a*) (E.Werner 1992) (I.Craig 1994*a*) podem ser encarados como um modelo de organização da sociedade, facilitando o estudo do comportamento de indivíduos quando chamados a operar em grupo. As potencialidades inerentes ao modelo dos SM permitem-nos criar as estruturas de conhecimento para o estudo e análise da relação das ciências cognitivas com os modelos de organização social. Estes modelos reflectem o modo de estar de um grupo de indivíduos que interactuam entre si, adoptando comportamentos que lhes permitem evoluir de uma forma organizada e apoiando-se mutuamente na resolução de problemas que emergem no seio

da comunidade a que pertencem.

O modelo organizacional que o homem adopta em comunidade é gregário por definição. Por outro lado, o indivíduo, ao analisar e interpretar o modelo da sociedade em que está inserido, contribui decisivamente para a evolução desse agregado. Uma sociedade pode-se auto-estruturar, mudando a sua organização de acordo com as necessidades subjacentes à sua evolução ou reflectindo novos estados, resultantes das acções realizadas pelos seus membros.

Toda esta problemática pode e deve ser transportada e adquirida no contexto dos SM. Os indivíduos de uma comunidade, agrupados de acordo com um modelo de organização social, conseguem ser emulados por um sistema computacional através de entidades artificiais adequadas - os agentes - que, independentemente da sua categorização, podem ser integrados em grupos, especializando-se.

2.2.1 Comunidades de Agentes

Os agentes, quando integrados numa comunidade, têm que obedecer aos requisitos e normas aplicáveis aos seus membros, devendo saber discernir que atitudes e comportamentos poderão ou não revelar. A sua integração pode ocorrer graças ao grau de conhecimento que possuem ou ser despoletada por ocorrências externas, provocadas por outros agentes ou por intervenientes humanos. Os agentes deverão, não só poder lidar com este tipo de eventos, de forma a não quebrarem as normas da comunidade, mas também saber como adoptar a melhor situação de compromisso, no sentido de não limitarem a sua capacidade de intervenção. É que, na verdade, os agentes são entidades limitadas; o seu conhecimento é finito, assim como a sua capacidade de

intervenção e os recursos de que dispõem.

O agrupamento de agentes em comunidades permite criar nichos de entidades especialistas em domínios de aplicação específicos. As capacidades de resolução de problemas que daí emergem poderão ser diminuídas se os agentes não forem capazes de efectivar as suas trocas de informação e partilhas de conhecimento.

Os motivos que levam os agentes a estabelecer processos de comunicação podem ser os mais variados, podendo ir desde simples trocas de informação, originadas por pedidos directamente trocados entre si, até à difusão de estruturas de conhecimento no ambiente da comunidade.

Não faz muito sentido reunir um conjunto de agentes com diferentes competências e domínios de conhecimento num único ambiente, organizando-se assim uma comunidade multifacetada, se o resultado não for uma combinação ajustada do conjunto das capacidades de resolução correspondentes a cada um dos agentes. Isto é, assume-se que o motivo que originou o agrupamento dos agentes é suportado pela necessidade de se potenciar uma maior capacidade na resolução de problemas ou na realização de tarefas, através da combinação e conjugação das perícias e conhecimento dos agentes que integram a comunidade. Na realidade, aquilo que se pretende é criar e desenvolver uma estrutura pela qual seja possível analisar e resolver problemas multidisciplinares através do desenvolvimento de acções cooperativas entre os vários agentes que constituem a comunidade.

O desenvolvimento de processos de cooperação entre agentes, tanto de características "pacíficas" como "conflituosas", exige efectivos e robustos canais de comunicação. Nestes últimos, poderiam ser desencadeados processos de

negociação entre os agentes, na tentativa de se alcançar um compromisso.

A capacidade de comunicação dos agentes é tanto maior quanto mais expressiva for a linguagem utilizada. É importante que se definam as formas de comunicação intra-agentes e, no caso de estes serem heterogéneos, que se criem os mecanismos de tradução adequados à transmissão de mensagens sem perda de informação ou integração involuntária de ruído no seu conteúdo.

Quando se permite a integração de diferentes comunidades de agentes num mesmo ambiente, o factor segurança é crucial para que seja garantida a privacidade de informação entre comunidades (quando tal for requirido). Devem ser estudados mecanismos que assegurem que agentes estranhos à comunidade não tenham acesso directo às trocas de informação, salvo quando possuam autorização explícita para o fazerem. A utilização de técnicas de encriptação é uma hipótese a considerar quando os processos e as trocas de informação em causa são declarados como privados.

2.2.2 Comunicação entre Agentes

No ambiente de um SM, o peso da comunicação no desempenho do sistema torna-se crítico por definição. Não obstante, os custos associados à implementação de canais de comunicação interagentes poderão tornar-se proibitivos, para além de poderem afectar significativamente o desempenho global do sistema. Dever-se-á, então, ponderar as vantagens inerentes a este modelo de comunicação, de acordo com as características específicas dos domínios de aplicação em causa. As estruturas de comunicação rígidas apresentam menores custos, mas também menor versatilidade.

Os agentes que usufruem directamente de estruturas de comunicação flexíveis deverão atender a critérios de "ética" na utilização deste tipo de meio: comportar-se como entidades passivas, assistindo às trocas de mensagens e observando o ambiente como simples espectadores, reagindo apenas face a estímulos. As vantagens serão menores em situações nas quais os agentes tenham necessidade de recolher "peças" de informação específicas. Neste último caso, eles terão necessariamente que analisar a informação disponível no ambiente. O melhor compromisso aparenta ser a conciliação dos dois tipos de estruturas de comunicação, de acordo com o domínio da aplicação em causa. O modo de comunicação a estabelecer será definido no momento, de acordo com uma tomada de posição por parte do agente, baseada no conhecimento que possui acerca do problema e de acordo com as prerrogativas de acesso à informação relevante para a resolução daquele.

É extremamente importante que a comunicação seja efectiva, mesmo quando numa comunidade se encontram integrados diferentes tipos de agentes, "exprimindo-se" em diferentes linguagens. Para que haja comunicação, é necessário estabelecer os canais adequados, sem perda de significado ou incorporação de ruído nas mensagens trocadas entre os agentes. Esta situação seria praticamente incomportável se os processos de comunicação fossem suportados por mensagens demasiado complexas ou de difícil tradução. Normalmente, o que se verifica nos SM é que o conjunto de mensagens definido (e utilizado) é de baixa cardinalidade, podendo, assim ser, facilmente manipulável.

Para assegurar a comunicação entre agentes, há que definir uma linguagem universal, uma espécie de dialecto do sistema, que os agentes possam reconhecer. A Figura 2.5 é um exemplo de uma arquitectura em que vários agentes

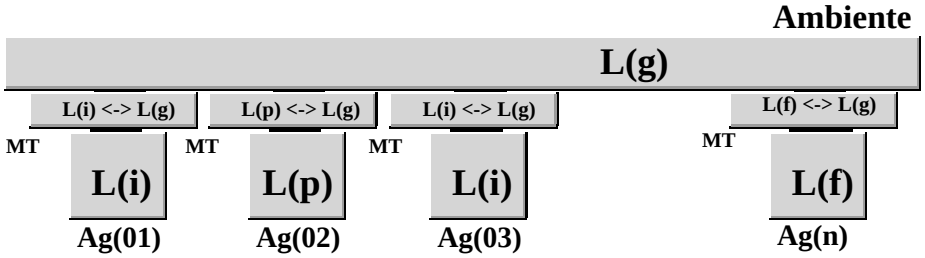


Figura 2.5: Comunicação entre Agentes com diferentes Dialectos.

coabitam, comunicando através de uma linguagem de carácter universal Lg, mas expressando-se nas suas linguagens específicas $L(i)$,$L(p)$ e $L(f)$, sendo os processos de tradução de uma linguagem para a outra (ex.: $L(i) <-> L(g)$) realizados por um módulo especial (MT) embebido na estrutura do agente ou disponibilizado pelo ambiente no qual os agentes possam estar integrados.

O estado de um agente é naturalmente privado. O conhecimento que possui só a ele diz respeito. Existem, contudo, situações em que essa informação poderá ser requerida ou voluntariamente cedida pelos agentes a terceiros. No caso da informação requerida ser recusada, os agentes devem estabelecer contacto no sentido do agente requerente tomar conhecimento do(s) motivo(s) que estiveram na origem da não satisfação do pedido endossado à outra parte. Este processo de justificação permitirá evitar eventuais situações de contenção ou "deadlock" nos processos em curso nos agentes que tenham solicitado essa informação, e que, sem mecanismos de "time-out", poderiam auto-suspender-se.

A difusão de informação por parte de um agente, através das estruturas de comunicação que o ligam aos diferentes elementos de uma comunidade, só deverá ocorrer se o problema em questão assim o justificar; isto é, se o agente em causa entender que tal acto poderá contribuir para a evolução do seu pró-

prio estado de conhecimento ou do estado global do sistema, ou se as normas e premissas de actuação da comunidade em que o agente está inserido assim o determinarem. Se num mesmo SM coexistirem diferentes comunidades, a difusão de informação deverá ser, em princípio, entre constituintes de uma mesma comunidade. A excepção, a ocorrer, deverá potenciar sinergias intercomunidades, tendo como actores apenas os agentes que desenvolvam e terminem o exercício com algum valor acrescentado (I.Craig 1994*a*).

Capítulo 3

Sistemas de Agentes Baseados em Conhecimento

Abordagem da nova geração de SBC e análise da sua correspondência com os sistemas orientados por agentes. O conhecimento e sua representação segundo a perspectiva da nova geração de SBC. Acções de coordenação, cooperação, resolução de conflitos e verificação de consistência em SM. SBQN e sua influência no modelo adoptado para o sistema.

O número de aplicações que se baseiam, directa ou indirectamente, na tecnologia de agentes tem vindo a aumentar rapidamente. O domínio da CBA pressupõe o desenho e o desenvolvimento de sofisticados componentes de software ou de hardware, de características independentes, que podem interactuar com outros componentes similares. Este tipo de abordagem é diferente da abordagem tradicional da *Engenharia de Software* (ES), que disponibiliza um modelo para a especificação e implementação de componentes de software. Através da CBA, a interacção entre os diferentes componentes de um programa deixa de obedecer a padrões de comportamento pré-estabelecidos,

desenvolvendo-se antes através de estruturas de conhecimento dinâmicas, com a satisfação de invariantes pré-definidos; isto é, a interacção entre componentes de software não pode ser desenvolvida em torno de estruturas de conhecimento rígidas, estabelecidas *a priori*, mas sim através de processos de cooperação, de negociação ou argumentação, complementados por acordos de natureza procedimental - compromissos entre os diversos intervenientes no processo. Contudo, o comportamento de um sistema na perspectiva da CBA não é estático. O sistema deverá ser capaz, através dos seus diversos componentes, de ajustar o seu próprio comportamento de acordo com o seu conhecimento e o estado corrente dos problemas. Ao sistema deverão ser dados os meios para que possa autonomamente modificar o seu comportamento (N.Jennings & M.Wooldridge 1995).

Não é simples a identificação das propriedades que um ambiente computacional baseado em agentes deve possuir. Se pensarmos em caracterizar tal ambiente como "universal", verificamos que essa dificuldade é acrescida de um factor não raras vezes proporcional à generalidade ambicionada para o próprio sistema. Parte dos sistemas disponíveis foi implementada com base em domínios de aplicação específicos, desenvolvendo-se e "crescendo" de acordo com as necessidades que os seus utilizadores apresentaram, que os seus criadores identificaram durante os processos de aquisição de conhecimento ou que foram naturalmente requeridas pelo próprio domínio de aplicação.

Os SBC podem ser encarados como *Agentes Baseados em Conhecimento* (ABC) ou *agentes cognitivos*. O tipo de estrutura funcional e as aplicações às quais normalmente estão associados dão-lhes um carácter muito peculiar. Contudo, ao analisarem-se esses dois tipos de sistema, verifica-se que o domínio de aplicação de um SBC é, em princípio, mais abrangente do que o de

um agente cognitivo. O "volume" de estruturas e modelos de representação de conhecimento, o leque de decisões normalmente em jogo, a capacidade de manipulação de dados são significativamente maiores do que os vulgarmente apresentados por um agente cognitivo. Em contrapartida, a aptidão social de um agente cognitivo é superior à de um SBC convencional. Em geral, uma abordagem orientada para o agente considera a possibilidade do estabelecimento de canais de comunicação com outros tipos de agentes. Um SBC tradicional actua normalmente sobre o seu domínio de conhecimento, no qual é "dono e senhor". Não tem, numa perspectiva tradicional, a necessidade de comunicar com outro tipo de sistema equivalente, porque deveria conter, em princípio, a perícia e o conhecimento necessários para resolver isoladamente qualquer problema que se lhe apresentasse dentro da sua área de conhecimento.

A migração deste tipo de sistemas para arquitecturas distribuídas trouxe-lhes um acrescento de potencialidades, mas também lhes apresentou novos desafios. Os SBC distribuídos exploram como um todo as capacidades de cada uma das suas partes constituintes, para que estas actuem em bloco na resolução de problemas. Do ponto de vista da CBA, o tipo de sistema apresentado anteriormente é definido como um SM.

Muitos dos princípios, características e funcionalidades incorporados no paradigma da CBA vão ao encontro da nova vaga dos SBC. Em conjunto, geram sinergias para o desenvolvimento de SBC mais poderosos, mais flexíveis, com um grau superior de autonomia e maiores capacidades de cooperação na resolução de problemas. Tal estratégia de incorporação recíproca de características e funcionalidades entre a CBA e os SBC pode impulsionar e suportar o esforço que se tem vindo a fazer no desenvolvimento da tecnologia, de

parceria com a análise, planeamento e concepção, de SBC - a principal actividade da EC (J.K.Lee, J.Liebowitz & Y.M.Chae 1996) (J.David, J.Krivine & R.Simmons 1993) (N.Jennings & J.Pople 1993).

3.1 Uma Nova Geração de Sistemas Baseados em Conhecimento

A tecnologia dos SBC tem vindo a evoluir no sentido de colmatar alguns dos pontos críticos daquela que foi considerada a 1ª geração de SP (D.Waterman 1986) (F.Hayes-Roth, D.Waterman & D.Lenat 1983). Essa evolução tem-se verificado segundo duas linhas de orientação distintas e paralelas:

- tornar os SBC capazes de manipular e combinar num mesmo meio, independentemente das suas características, diversos modelos de representação de conhecimento e raciocínio;
- definir uma metodologia orientada por conhecimento para o desenho e desenvolvimento dos SBC.

Nesta linha, os sistemas orientam-se pelo princípio da racionalidade, seleccionando as acções a realizar que consideram mais relevantes para o processo de resolução de um problema e, consequentemente, para a obtenção de uma solução.

O conhecimento é um problema central. A sua modelação é vital para a criação de sistemas mais poderosos, que se possam compreender mais facilmente e com maiores facilidades de manutenção. No sentido de se reflectir adequadamente sobre o processo de resolução de um problema real, é fre-

quentemente necessária a combinação de diferentes formas de representação de conhecimento e raciocínio.

A possibilidade de se utilizarem diferentes modelos de representação de conhecimento e raciocínio tem um impacto significativo e directo na eficiência e na competência dos futuros SBC. Na realidade, a actual geração de SBC está essencialmente preocupada com a identificação do conhecimento considerado imprescindível para a resolução de um problema e com a melhor forma de o representar, de maneira que um sistema computacional possa maximizar o seu desempenho. Estas duas vertentes são fundamentais na caracterização daquilo que actualmente é identificado como SBC de 2ª geração (J.David et al. 1993).

As fronteiras entre estas duas gerações de SBC não estão claramente estabelecidas, reflectindo mais uma evolução de ideias, técnicas ou mesmo de estilos que se têm vindo a desenvolver no domínio dos SBC. No entanto, em conjunto, estas duas vertentes contribuem para uma melhor modelação do domínio da aplicação e para o entendimento do papel a representar por cada uma das partes, em termos de formas de representação de conhecimento, cooperação ou sistemas de raciocínio. Processos relacionados com a aquisição de conhecimento e as capacidades de explicação do sistema, bem como aspectos relativos a robustez, eficiência e reutilização, são significativamente beneficiados com este tipo de abordagem efectuada pela nova geração de SBC.

A necessidade de melhorar as capacidades de aquisição de conhecimento assume um papel central nesta nova geração de SBC. Na geração anterior, o processo de aquisição restringia-se, em larga medida, à realização de processos de transferência de conhecimento entre os especialistas e o engenheiro do

conhecimento, e à sua consequente codificação. Agora, o processo de aquisição de conhecimento é visto como um processo de síntese de programas dividido em duas fases distintas: modelação e instanciação. Os SBC de 1ª geração dão uma visão deformada do mundo real que pretendem modelar. Na actual geração esse modelo do mundo tornou-se explícito. Os modelos ao nível do conhecimento são úteis para os processos de aquisição, já que ajudam a atenuar as falhas eventualmente verificadas entre o discurso do especialista do domínio e o seu reflexo no sistema implementado. A instanciação do modelo é a segunda fase do processo. Esta é conduzida pelo modelo desenvolvido na fase de modelação, e indica qual o conhecimento a ser adquirido. Complementarmente, define o papel que se espera que cada "peça" de conhecimento venha a desempenhar.

A validação de uma base de conhecimento envolve a validação do seu arquétipo e do conhecimento de diferentes instanciações deste. Ambos funcionam, em termos de conhecimento a representar, como uma forma de especificação funcional, por camadas, do próprio sistema, podendo ser objecto de análise pelos especialistas dos domínios da aplicação em causa. Complementarmente, definem os procedimentos a seguir no processo de aquisição de conhecimento.

A avaliação de um SBC deve ser realizada com base na sua aptência para a resolução de problemas, na sua eficiência e efectividade, no seu desempenho e rapidez e, naturalmente, na sua capacidade de explicação das soluções encontradas ou das decisões tomadas. Os modelos de explicação da geração anterior, baseados frequentemente na análise das cadeias de inferência protagonizadas pelo sistema, não apresentavam, geralmente, explicações "satisfatórias", devido, em grande medida, ao facto de o conhecimento necessário ao processo de explicação estar implicitamente definido na base de conheci-

mento do próprio sistema. Uma das formas de se ultrapassar esta limitação será, naturalmente, a utilização explícita de conhecimento para a geração de explicações, mas a um nível superior de abstracção, em termos do arquétipo do próprio sistema.

Os sistemas orientados para problemas de diagnóstico ou classificação, entre outros, baseando-se na procura de uma solução a partir de critérios previamente estabelecidos, não facilitam a explicação de uma solução. Em alternativa, o método dito *construtivo* assenta no princípio de que quem resolve o problema, a máquina de raciocínio, passa as soluções e as observações que recolhe durante o processo de resolução do problema a um segundo sistema - o sistema de explicação. Este último terá como função justificar e explicar as conclusões alcançadas pelo primeiro sistema, desenvolvendo os processos de raciocínio necessários à explicação, através do recurso a perícia própria, conhecimento e formas de raciocínio adequados à realização dessa tarefa.

A utilização de diferentes formas de raciocínio e de modelos para a representação de conhecimento aumenta a fiabilidade, a competência e a robustez de um SBC. A sua separação de acordo com as fontes de conhecimento permite uma abordagem mais efectiva para cada tipo de conhecimento utilizado.

A forma como o conhecimento está representado e os métodos utilizados para a sua interpretação afectam, naturalmente, o grau de eficiência do sistema. O desenvolvimento de novos modelos de SBC em que os sistemas de representação de conhecimento e de raciocínio se hibridizam, não prefigurando um comportamento estático, afigura-se-nos promissor. Por outro lado, qualquer componente de um SBC - uma base de conhecimento, uma estrutura de raciocínio, etc. - que seja replicável pode ter um contributo significativo na atenuação de eventuais estrangulamentos provocados pelos processos de

aquisição de conhecimento, bem como na redução dos custos de implementação.

A reutilização de componentes é um recurso sempre presente em cada uma das fases de desenvolvimento de um SBC da actual geração. Desde o estudo e o desenho dos modelos de representação de conhecimento até à sua implementação (código ou "shells" que implementam modelos genéricos), a reutilização de componentes é uma actividade que se pretende constante. Porém, a dificuldade de adaptação de componentes ditos de aplicação genérica a problemas específicos continua latente. Quer a programação orientada por objectos (G.Booch 1994) quer a visualizável através de agentes (N.Jennings et al. 1993) (A.Ovalle 1994) irão, com certeza, dar um contributo válido para a modulação de SBC, a sua manutenção e a sua potencial modularização.

Figura 3.1: Objectivos da Nova Geração de Sistemas Baseados em Conhecimento.

Em síntese (Figura 3.1), esta nova geração assenta no desenvolvimento de SBC:

- orientados pelo princípio da racionalidade;

- em que o conhecimento é um problema central, devendo a sua modelação ser explícita;
- com melhores capacidades de aquisição de conhecimento;
- que utilizam diversos modelos de raciocínio e de representação de conhecimento;
- com melhores e mais efectivas capacidades de explicação;
- que apresentam uma actividade constante de reutilização de componentes.

3.2 Agentes Baseados em Conhecimento

A procura de novos modelos computacionais, a aposta na distribuição, o advento de novas formas de raciocínio, de representação de conhecimento ou de verificação de consistência podem ser encontrados na tecnologia de ABC. De uma forma geral, este tipo de agentes apresenta uma estrutura orgânica em muito semelhante aos SBC. Os ABC "conhecem" o seu mundo, possuindo conhecimento relacionado com o domínio de aplicação. As suas capacidades de resolução de problemas podem evoluir ou ser melhoradas (uma forma de aumentar a sua competência) através de uma nova modelação do universo, de novos processos de aquisição de conhecimento ou por meio de processos específicos de aprendizagem inerentes ao próprio agente. As aptidões sociais de um ABC habilitam-no a realizar de parceria com outros agentes tarefas de análise e de resolução de problemas, o que lhes permite, possivelmente, alcançar melhores e mais efectivas soluções para um problema, através do desenvolvimento de acções cooperativas.

Os ABC têm incorporados na sua estrutura modelos representativos da sua área de aplicação, que utilizam como ponto de partida para os seus processos de decisão. Para a concretização dos seus objectivos, os ABC precisam, normalmente, de conhecer:

- qual o estado actual do conhecimento do ambiente em que estão integrados;
- como tratar informação incompleta;
- quais os objectivos a alcançar;
- qual a repercussão das suas acções no estado do conhecimento do ambiente e no seu próprio estado interno.

Um ABC é designado como um agente racional se realizar as operações necessárias, de forma correcta, face a alterações que porventura se desenvolvam no seu ambiente de trabalho, maximizando o seu desempenho. A medida do desempenho de um agente é definida em termos da satisfação de um conjunto de critérios que medem o grau de sucesso do agente no seu domínio de aplicação. A racionalidade de um agente, num dado momento, depende dos seguintes factores (S.Russel & P.Norvig 1995) :

- a sua medida de desempenho;
- o conjunto de percepções por ele realizadas até ao momento, isto é, a sua sequência de percepção;
- o seu conhecimento acerca do ambiente no qual está integrado;
- o conjunto de acções que é capaz de realizar.

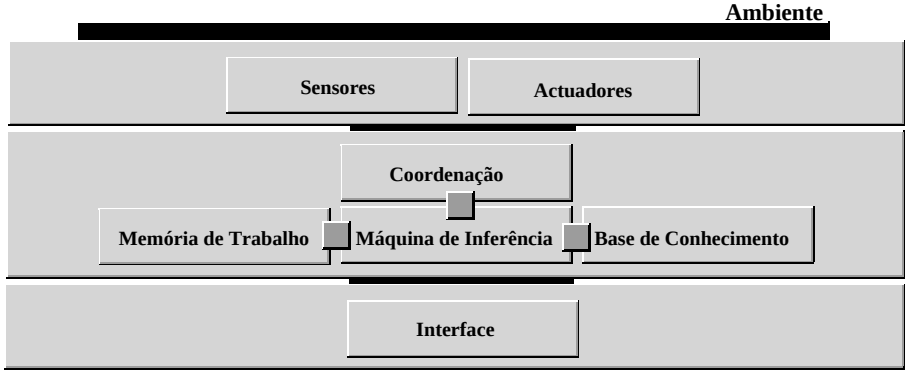


Figura 3.2: Arquitectura Funcional de um Agente Baseado em Conhecimento.

No Capítulo 2, Secção 2.1.3, foi apresentado o modelo genérico da arquitectura funcional de um agente. A complementação do modelo proposto anteriormente com as funcionalidades e características de um ABC dá corpo a uma nova arquitectura (Figura 3.2), tendo como base os seguintes módulos funcionais:

- **Sensores**.

  Implementam mecanismos cuja função é analisar a informação que circula no ambiente do sistema, procurando aquela que possa ser de utilidade para os processos internos do agente.

- **Actuadores**.

  Implementam mecanismos responsáveis pela colocação no ambiente do sistema da informação gerada pelos processos do agente ou disponível nas suas estruturas de conhecimento internas.

- **Máquina de Inferência**.

  A parte do sistema que opera sobre a base de conhecimento do agente, produzindo inferências.

- **Base de Conhecimento**.

  Potencial de informação num domínio particular de conhecimento, o qual foi descrito (formalizado) de acordo com um modelo apropriado para a prática de raciocínios.

- **Memória de Trabalho**.

  Potencial de informação relacionada com o processo de resolução de problemas em curso.

- **Interface**.

  Mecanismos de inferência e estruturas de conhecimento orientadas para o suporte dos processos de comunicação entre o agente e os seus eventuais utilizadores.

- **Coordenação**.

  Unidade responsável pela coordenação de todos os processos que correm na estrutura interna do agente.

De entre os novos elementos funcionais incluídos no sistema do agente, há que realçar os do subsistema de protocolo (constituído essencialmente pelos sensores e actuadores do agente), que concedem a esta entidade a capacidade de se inter-relacionar com outros agentes. Tais características potenciam novas sinergias no sistema, quer sejam de cooperação ou de simples transmissão de informação.

O conhecimento que um agente integra na sua base de conhecimento, os seus mecanismos de inferência e as suas capacidades de aprendizagem podem ser suficientes para a resolução de um problema. No entanto, num contexto multidisciplinar, é comum recorrer-se à combinação da perícia e do conhecimento de diversos especialistas.

```
procedimento Agente-Baseado-Conhecimento
    activar-processamento-interno;
    ler-condicionantes-percepção(Condicionantes)
    repetir
        repetir
            analisar-ambiente(Condicionantes,Percepção)
        até Percepção
        analisar-percepção(Percepção,Actuação)
        se Actuação então
            actualizar-estado-interno(Percepção,_,Novo-Estado);
            repetir
                verificar-consistência(Consistência)
                se ¬Consistência então
                    restaurar-consistência(Consistência,Inferência)
                fim-se
            até Consistência
            se Inferência então
                activar-inferência(Percepção,Novo-Estado,Estratégia);
                definir-acção-para-execução(Estratégia,Acção);
                executar-acção(Acção,Resultado);
                se Resultado então
                    inserir-resultado-no-ambiente(Resultado)
                fim-se
                actualizar-estado-interno(Acção,Resultado,_)
                aplicar-mecanismos-aprendizagem(Acção,Resultado)
            senão
                Fim-Execução
            fim-se
        fim-se
    até Fim-Execução
    fim-procedimento.
```

Figura 3.3: Descrição Genérica do Comportamento de um Agente Baseado em Conhecimento.

Um ABC é uma entidade especialista num domínio particular de conhecimento. Para a resolução de um problema que envolva vários domínios do conhecimento, impõe-se a necessidade de se combinar a perícia e o conhecimento de diversos agentes. O objectivo principal de uma comunidade de agentes especialistas deve ser, sempre, a obtenção, através de todos os recur-

sos disponíveis, de uma solução para os casos que os seus interlocutores lhe apresentarem.

Os agentes têm visões parciais do mundo. A complexidade dos problemas que lhes são propostos, associada à distribuição dos recursos envolvidos e aos níveis de incerteza e contradição que a informação dos problemas transporta, impossibilitam os agentes de conhecer de forma completa todo o conhecimento respeitante ao contexto dos problemas em resolução. As aplicações de ABC dão-se em domínios de conhecimento envolvendo diversos especialistas, cujas perícias se complementam. A cooperação entre agentes torna-se, assim, essencial, face à incapacidade de um agente para resolver por si só os problemas que lhe são propostos.

O comportamento de um ABC é apresentado na Figura 3.3, com base nos pressupostos a que deve obedecer um agente genérico (Capítulo 2, Secção 2.1.3), complementados com as funcionalidades associadas a um SBC tradicional[1].

[1] O algoritmo descreve, basicamente, o comportamento de um ABC genérico. É uma extensão ao algoritmo apresentado anteriormente no Capítulo 2, Secção 2.1.3. No algoritmo identificam-se quatro fases de acção distintas: 1) arranque do agente - fase na qual o agente prepara o ambiente local e faz a leitura das condicionantes de percepção (*Condicionantes*); 2) percepção - fase na qual o agente analisa o ambiente na tentativa de encontrar informação que lhe possa interessar (*Percepção*) e que preencha os requisitos impostos pelas condicionantes de percepção; 3) análise - fase na qual o agente analisa a percepção e decide qual a acção a realizar e o tipo de estratégia que deve aplicar; 4) actuação - fase na qual o agente actualiza a sua base de conhecimento com o conhecimento adquirido (*Percepção*), verifica a consistência da informação do novo estado de conhecimento que alcançou (*Novo-Estado*) e, caso não se verifique nenhum tipo de inconsistência neste novo estado de conhecimento, activa os seus mecanismos de inferência; se for detectado um estado de inconsistência, então o agente tenta restaurar a sua base de conhecimento (*restaurar-consistência(Consistência,Inferir)*), a fim de obter de novo um estado consistente; em seguida, define qual a acção a executar, segundo a estratégia definida na fase de inferência, executa-a e coloca no ambiente o resultado que alcançou (*Resultado*); para finalizar a fase de actuação, o agente actualiza o seu estado de conhecimento interno com o resultado, aplicando de seguida os seus mecanismos de aprendizagem, no sentido de melhorar as suas estratégias de resolução de problemas. Todas as fases, à excepção da fase de arranque, são continuamente executadas, até que o agente receba a indicação de que

3.3 Conhecimento e Representação

A aptência de um ABC para a resolução de problemas depende directamente do tipo de informação que possui na sua base de conhecimento. A forma como o agente utiliza esse conhecimento varia de acordo com o modelo utilizado na sua representação. O desempenho de um agente na resolução de um problema é, portanto, função dessa representação.

Conhecimento e representação são dois pilares distintos, embora complementares, em qualquer tipo de SBC (R.Simmons & R.Davis 1993):

- o **conhecimento** é uma descrição do mundo, condicionando a competência do sistema para a resolução de problemas;
- a **representação** é a forma como o conhecimento está codificado, condicionando o desempenho do sistema na resolução de um problema.

A distinção entre estes dois conceitos faz-se tanto ao nível do desenho do sistema como na sua fase de análise e estruturação. A distinção entre conhecimento e representação assume maior relevância na nova geração de SBC. Em sistemas que integrem múltiplas fontes de conhecimento e/ou modelos de representação de conhecimento, o entendimento da distinção entre os dois conceitos é fundamental.

A informação associada a uma determinada área de conhecimento pode ser dividida por diferentes fontes de conhecimento, processo que facilitará significativamente a sua aquisição e actualização, influindo positivamente na resolução dos problemas. Tal circunstância requer que os SBC estejam habi-

deve terminar a sua execução (*Fim-Execução*).

litados a lidar simultaneamente com diferentes paradigmas de representação de conhecimento, no sentido de conseguirem incorporar todo o conhecimento necessário à representação dos domínios das aplicações.

Dois fortes motivos sustentam esta estratégia. Em primeiro lugar, a utilização de diferentes fontes de conhecimento torna o processo de desenvolvimento dos SBC mais simples e de mais fácil compreensão, já que permite a definição de módulos de conhecimento mais pequenos, logo, de mais fácil manipulação, e adicionam uma estrutura à forma como o conhecimento é manipulado. Em segundo lugar, ao utilizarem-se diversas representações, torna-se possível criar e desenvolver novas bases de conhecimento para uma mais eficiente e efectiva resolução dos problemas.

Um terceiro motivo, que emerge como consequência dos anteriores, é definido em termos de uma melhor estruturação, de uma mais eficiente análise do desempenho e manutenção do sistema e de uma detecção e correcção de erros simplificada, na sua fase de pós-desenvolvimento.

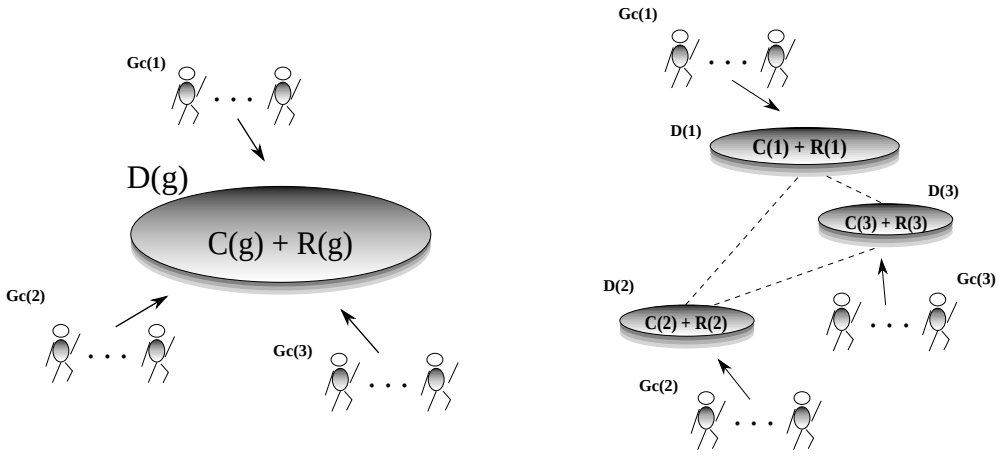


Figura 3.4: Abordagem à Resolução de Problemas na Nova Geração de Sistemas Baseados em Conhecimento.

A Figura 3.4 apresenta uma possível ilustração da forma como o processo de aquisição de conhecimento pode ser encarado na anterior e na actual geração de SBC.

Os métodos de resolução de problemas requerem, frequentemente, conhecimento acerca das tarefas a realizar, bem como dos modelos de comportamento das entidades envolvidas. Estes tipos de conhecimento são invariavelmente encontrados em qualquer SBC. Similarmente, diferentes áreas de aplicação podem ser identificadas e analisadas em separado, sendo possível adquirir o conhecimento com elas relacionado através de processos de aquisição autónomos. Tal estratégia tem como consequência imediata a rentabilização do tempo, a optimização na utilização de recursos e uma maior flexibilização no processo de aquisição de conhecimento, melhorando os seguintes aspectos:

- **Processos de aquisição** - adquire-se o conhecimento separadamente, desenvolvendo-se em seguida os processos da sua integração nas diversas fontes de conhecimento definidas.
- **Extensibilidadde do sistema** - facilmente se poderá, mais tarde, acrescentar novas funcionalidades ao sistema, assim como integrar novo conhecimento sem que outras partes do sistema sejam afectadas.
- **Reutilização de blocos funcionais ou das estruturas de conhecimento** - reutilização de componentes de software no desenvolvimento de novas fontes de conhecimento.

A dificuldade em dividir adequadamente o domínio da aplicação apresenta-se como o primeiro senão deste tipo de abordagem. Nem sempre a forma de o fazer é óbvia. Por vezes, há dificuldades em separar o conhecimento

relacionado com o domínio e o conhecimento que rege as acções que aí possam decorrer. Se bem que este tipo de abordagem permita a modularização das fontes de conhecimento, com as vantagens daí inerentes, também provoca a geração de "fronteiras" artificiais num mesmo domínio de aplicação. Complementarmente, se os domínios de conhecimento incorporados em duas fontes tiverem zonas de intersecção, será necessário desenvolver processos que assegurem a consistência desse conhecimento, o que implica a existência de estruturas e mecanismos de interface e de comunicação.

O compromisso entre a capacidade de expressão e a eficiência computacional num modelo de representação de conhecimento não é fácil de se conseguir. As características do domínio de aplicação em causa condicionam marcadamente a melhor forma de o conseguir. Podem-se equilibrar estas duas características com o intuito de garantir um compromisso através de restrições, aplicadas ora sobre uma ora sobre a outra. Por exemplo, ao limitar-se a capacidade de expressão do modelo de representação, embebendo-se nos algoritmos de inferência aspectos relacionados com o domínio do conhecimento em causa, consegue-se aumentar a eficiência computacional do sistema. Esta alternância e o seu consequente estudo e análise, realizados em etapas sucessivas, permitem verificar o grau de adaptabilidade do modelo (capacidade de expressão mais eficiência) ao domínio da aplicação em causa.

Decidir qual a representação a utilizar e como combinar os diferentes tipos de modelos são algumas das dificuldades encontradas nestas situações. A decisão sobre qual o melhor modelo para determinado problema é sempre difícil de tomar. A situação complica-se quando se torna necessário estabelecer relações entre o conhecimento do domínio da aplicação e o modelo de representação. As características dos problemas condicionam sempre o

tipo de representação a utilizar. O ideal seria ter-se um modelo de representação especificamente desenvolvido para cada domínio de aplicação em causa. A transmissão de informação entre entidades obedecendo a diferentes paradigmas de representação de conhecimento é, não raras vezes, suportada por formas de representação intermédias, com procedimentos associados, de modo a possibilitar a migração de conhecimento entre as partes envolvidas.

3.4 Sistemas Multiagente Baseados em Conhecimento

A partir da abordagem da resolução de problemas de forma distribuída e cooperativa emergiu uma nova geração de SBC, através de uma categorização da sua especificidade, com base nos seguintes pressupostos (E.Durfee, V.Lesser & D.Corkill 1989):

- **Negociação**.
  Definição de planos a seguir na resolução de um problema através da combinação de diferentes formas de diálogo e teorias de argumentação.
- **Cooperação funcionalmente precisa**.
  Resolução de inconsistências através da avaliação de soluções parciais para o problema em questão, como forma de optimizar o processo de convergência das soluções parciais para a solução final.
- **Estrutura organizacional**.
  Utilização de conhecimento que emana da topologia ou forma de socialização seguida no SM, com vista a optimizar os termos em que a resolução de um problema pode ser equacionada.

- **Planeamento multiagente**.

  Partilha de informação no sentido de se regulamentar a cooperação entre agentes.

- **Controlo local (sofisticado)**.

  Integração das sinergias de um agente com as da comunidade a que pertence, como forma privilegiada de potenciar o seu desempenho.

- **Modelos teóricos**.

  Favorecimento da via formal para a concepção e implementação de SM, em detrimento de formas empíricas e informais.

Com base nas características de um SM convencional, pode-se encarar cada um dos SBC como ABC que cooperam através da conjugação de perícias e conhecimento, partilhando recursos computacionais ou assistindo outros agentes. Desta forma, obtém-se um SM de ABC: um *Sistema Multiagente Baseado em Conhecimento* (SMBC). A resolução de problemas por SMBC é uma tarefa que envolve normalmente um grupo de ABC distribuídos que orientam as suas sinergias, a fim de maximizar e optimizar os seguintes processos:

- **Coordenação**.

  As actividades desenvolvidas ao nível de um SMBC devem ser orientadas de modo que se criem sinergias com base no agente e se eliminem eventuais tentativas de "monopolização" de recursos, assim como a potencial ocorrência de situações de "deadlock", contenção ou estrangulamento em processos interagentes.

- **Modulação da comunicação**.

  O desenvolvimento da cooperação entre agentes exige uma forma de

comunicação prática, que disponibilize as estruturas funcionais sobre as quais assenta esse processo e o meio de comunicação.

- **Resolução de conflitos**.

  A interacção entre agentes potencia o aparecimento de conflitos, pelo que é imperativa a existência de mecanismos para a formulação de uma argumentação, a sua análise em contexto e a tomada de uma decisão com eliminação do conflito.

- **Verificação de consistência**.

  A verificação da consistência da informação armazenada nas bases de conhecimento dos agentes ou da informação partilhada constitui uma forma expedita de compatibilizar o conhecimento que materializa cada agente.

3.5 Coordenação de Sistemas de Agentes

A ocorrência de conflitos, a repetição de procedimentos e a emergência de problemas de contenção ou estrangulamento são apenas alguns dos maus sintomas que um SM possuidor de um modelo de coordenação desajustado pode apresentar.

Se qualquer agente possuisse todo o conhecimento acerca do ambiente, perícia e conhecimento dos restantes agentes da comunidade, e se a disponibilidade de recursos fosse inesgotável, a coordenação desse sistema de agentes seria provavelmente fácil de alcançar. Porém, os custos para a manutenção de uma estrutura como esta seriam exagerados, já que haveria a considerar desde questões de alocação de recursos computacionais até à própria estrutura

para comunicações, a fim de se garantir, simplesmente, que a mudança de uma "peça" de conhecimento de um dado agente fosse visível por toda a comunidade. Tal situação é irrealista e impraticável.

A utilização de um agente supervisor como forma de garantir um certo grau de sociabilidade do sistema seria uma solução bastante mais prática e com menores custos que a anterior. Contudo, em aplicações reais há decisões a tomar pelos agentes que não se coadunam com a sobrecarga de processamento naturalmente associada às tarefas de supervisão de um agente controlador. O controlador, com o seu tipo de comportamento em algumas aplicações, gera sérias situações de estrangulamento nos processos.

Desenhar, desenvolver e implementar sistemas onde a ocorrência de situações de estrangulamento e contenção sejam mínimas e cujo desempenho não se vá degradando ao longo do tempo conduziu a uma outra aproximação, baseada na distribuição de controlo e de dados por todos os elementos activos no sistema, segundo as suas perícias e conhecimento.

O controlo distribuído permite aos agentes usufruirem de um maior grau de autonomia no desenvolvimento das suas acções. De qualquer forma, apesar das vantagens desta aproximação, torna-se necessário garantir que os agentes integrados numa dada comunidade não passem a maioria do seu tempo ocupados consigo próprios, não desenvolvendo formas de cooperação. A granularidade das entidades deverá ser tal que permita equilibrar a dualidade pequenas unidades modulares / custos de distribuição (E.Durfee et al. 1989).

Ao disponibilizarem-se agentes com mecanismos que lhes permitam tomar "consciência" das suas próprias acções e dos benefícios que poderão tirar das suas interacções com outros agentes, garantir-se-ão com certeza as bases

para desenvolver um modelo de coordenação (N.Jennings 1993*a*) (J.Neves & J.Machado 1997). A necessidade de implementar acções de coordenação num sistema de agentes deve-se essencialmente a três motivos (N.Jennings 1996):

- a existência de dependências entre as acções a executar pelos diferentes elementos da comunidade;
- a necessidade de se criarem sinergias;
- o facto de, normalmente, nenhum agente individual possuir competência, recursos ou informação para resolver um problema a título individual.

3.6 Cooperação e Resolução de Conflitos

Muitas vezes, num SM, os agentes não possuem a perícia, o conhecimento ou os recursos necessários para resolverem de forma independente os problemas que lhes são propostos. Essas situações vêm reforçar os motivos para a implementação de soluções que potenciem a cooperação entre agentes, e que serão justificados tendo em conta o trabalho desenvolvido por E.Durfee et al. (1989):

- incrementar o aparecimento de novas soluções, através da formação de subsoluções ou soluções intermédias;
- tornar fluidos os processos de troca de informação:
- responder a situações de ruptura do sistema ("fault tolerance");

- prevenir situações de bloqueio fatal ("deadlock"), estrangulamento e redundância;
- melhorar a utilização dos recursos de comunicação disponíveis através da imposição de métricas ao sistema de manipulação e troca de mensagens interagentes.

Duas formas de cooperação são apresentadas em R.Smith & R.Davis (1988) para a resolução de problemas de forma distribuída: a partilha de tarefas e a partilha de resultados.

A partilha de tarefas é utilizada na organização da decomposição de problemas através da definição de relações tarefas/subtarefas explícitas entre os diversos nodos de processamento. A estruturação hierárquica de tarefas torna-se útil na síntese de respostas e soluções. Esta estratégia de cooperação assume que as diversas subtarefas em que se divide a tarefa original podem ser executadas independentemente, reduzindo-se, assim, ao mínimo os processos de comunicação interagentes. A eficiência global do sistema depende, em grande medida, da eficiência de cada agente quando considerado autonomamente. O desenvolvimento de processos de negociação entre o agente encarregue da alocação de tarefas e os agentes processadores das mesmas pode ser uma forma de lidar com as dificuldades de alocação de tarefas às entidades processadoras, potenciando, assim, maior eficiência para o sistema.

A partilha de resultados é uma forma de cooperação na qual os diversos nodos individuais de processamento se ajudam mutuamente através da troca de resultados, com base nas suas próprias perspectivas de resolução relativamente ao problema global. Nos sistemas em que se utiliza esta forma de cooperação, as acções de controlo são tipicamente orientadas por dados.

Os processos realizados num nodo, em determinado instante, dependem da informação que possui localmente ou a que tem acesso por via remota em outros nodos. Não existem, explicitamente, relações de hierarquia entre tarefas e subtarefas. Este tipo de cooperação é mais vantajoso em domínios de aplicação onde os resultados alcançados por um agente influenciam os de outros agentes.

C.Zhang (1992) sugere quatro formas de cooperação, de acordo com as relações de dependência entre as entidades envolvidas no processo, que se podem aplicar ao domínio dos SMBC:

- **Horizontal**.

  Em que cada SBC pode autonomamente gerar uma solução para um problema sem que para isso necessite, directa ou indirectamente, da ajuda de um outro SBC. A cooperação neste tipo de sistemas desenvolve-se com a conjugação de estratégias de resolução, conhecimento e recursos inerentes aos mesmos.

- **Em árvore**.

  Supõe diferentes SBC integrados a distintos níveis. O sistema distribui um conjunto de subproblemas por diferentes subsistemas, que mais tarde lhe reportam as suas propostas de solução. O primeiro tratará de reunir as diferentes propostas e, com a aplicação da sua perícia e conhecimento, apresentar uma solução para o problema.

- **Recursiva**.

  Os diferentes subsistemas apresentam-se como mutuamente dependentes no processo de procura de uma solução para o problema proposto.

- **Híbrida**.

  Os diferentes subsistemas recorrem à cooperação horizontal de parceria com a cooperação em árvore ou recursiva.

As situações de conflito manifestam-se tanto ao nível da cooperação por partilha de tarefas como da cooperação por partilha de resultados, obedecendo à seguinte taxonomia (E.Oliveira, F.Mouta & A.Rocha 1993*b*):

- **Partilha de tarefas**.
  - **Positiva**.

    Sempre que há agentes capazes de executar uma tarefa solicitada por outro agente.
  - **Negativa**.

    Sempre que não existem agentes com aptidão para a execução, em tempo útil, de uma tarefa solicitada por outro agente.
- **Partilha de resultados**.
  - **Positiva**.

    Sempre que um conjunto de agentes que está a tentar executar uma mesma tarefa obtém resultados ou soluções diferentes mas complementares, ou resultados similares com diferentes factores de confiança.
  - **Negativa**.

    Quando diferentes agentes, ao executarem uma mesma tarefa, dão origem a soluções ou resultados contraditórios ou incongruentes.

A negociação é um processo fundamental na resolução de conflitos[2] (A.Rocha 1994), podendo ser baseada nos seguintes modelos:

- **Contratação**.

  Processo que assenta essencialmente na distribuição de tarefas pelos agentes presentes no sistema, por forma a maximizar o desempenho deste. Existe um agente que anuncia as tarefas que tem para distribuir, recebe as propostas das partes interessadas e, após definir quem realiza o quê, estabelece acordos explícitos, ou contratos, com os agentes seleccionados (R.Smith 1988).

- **Planeamento global/parcial**.

  Técnica através da qual o agente planeia as suas próprias acções. Cada agente pode representar e auscultar as actividades de outros agentes, assim como decidir a forma da sua participação no funcionamento global do sistema - estas representações são denominadas *planos de acção parcial/global*. Os agentes vão sucessivamente gerando novos planos (que se pretendem mais completos) com base na informação entretanto adquirida relativamente às actividades dos seus pares, por forma a manterem vivo o sistema global (E.Durfee & V.Lesser 1991) (E.Durfee 1988).

- **Modelos baseados na teoria dos jogos**.

  Modelos normalmente baseados em negociações por etapas, nos quais os agentes, com vista a alcançarem os seus objectivos, fazem ofertas e contra-ofertas, no sentido de chegarem a um acordo ou plano que estabelecerá as "regras" pelas quais se orientarão na execução das suas acções (J.Szép & F.Forgó 1985) (R.Gibbons 1992).

[2]Para uma abordagem mais detalhada do tratamento de situações de conflito em SM e sua resolução, veja-se E.Oliveira et al. (1993*b*) e A.Rocha (1994).

- **Argumentação persuasiva**.

  Este tipo de negociação implica que cada agente do sistema possua informação acerca dos objectivos dos seus pares para, de alguma forma, condicionar o seu comportamento (K.Sycara 1989).

3.7 Implementação de Estratégias de Cooperação

Não é suficiente definir um modelo de cooperação tal como os apresentados anteriormente, torna-se também necessário determinar as estratégias para a sua implementação. Estas devem estabelecer as políticas relativas à comunicação e à organização dos agentes no sistema, por forma a assegurar um modelo coerente. As estratégias de cooperação podem ser dadas por políticas organizacionais e políticas de distribuição de informação (S.Cammarata, D.McArthur & R.Steeb 1988).

As políticas organizacionais ditam a forma de decomposição de uma tarefa em subtarefas, que poderão ser alocadas a agentes individuais. Este tipo de políticas permite definir qual o papel de cada agente, quando integrado num grupo ou numa comunidade de agentes. Por outro lado, permite estabelecer os canais de comunicação entre agentes, assim como o tipo de mensagens que estes podem trocar entre si. De uma forma geral, as políticas organizacionais direccionam e condicionam fortemente o comportamento dos agentes.

Uma política de distribuição de informação determina a natureza da comunicação entre agentes. Isto é, as formas de comunicação entre agentes estão condicionadas pela escolha de um modelo social, o qual, à partida e por definição, estabelece o modo de comunicação entre agentes. Não é, porém,

suficiente definir um modelo social, já que este implica que sejam tomadas decisões acerca de quando e como a comunicação se pode processar. Devem ainda ser tomadas decisões com respeito ao tipo de comunicação que um agente poderá desenvolver, nomeadamente (S.Cammarata et al. 1988):

- **Difusão ou comunicação selectiva**.
  Dever-se-á condicionar o acesso à informação ou permitir o acesso a esta por parte de cada agente? Quais os critérios a seguir no caso de se optar por uma comunicação selectiva?
- **Comunicação solicitada ou não solicitada**.
  Numa situação em que um agente tem conhecimento do destinatário da comunicação, deverá tomar a iniciativa de comunicar a informação de que dispõe ou esperar que lhe seja solicitada?
- **Comunicação com ou sem aviso de recepção**.
  Deverá o agente remetente solicitar ao agente receptor que o informe da recepção da sua mensagem?
- **Transmissão única ou repetida**.
  Determinada peça de informação deverá ser emitida apenas uma vez ou ser repetida?

3.8 Verificação de Consistência

Os sistemas de revisão de "crenças" ("Belief Revision Systems") (B.Malheiro, N.Jennings & E.Oliveira 1994) são sistemas susceptíveis de ressumar-se face à informação que possuem de acordo com o seu universo de discurso. Numa das suas vertentes existem os sistemas de manutenção de verdade ("Truth

Maintenance Systems") (J.Martins 1990), que podem ser encarados como entidades autónomas, sendo, por isso, facilmente integráveis na estrutura de outros sistemas, tais como os SBC. O mesmo tipo de aplicação pode fazer-se no domínio dos SMBC.

A forma como um processo de verificação de consistência se desenvolve está directamente relacionada com o tipo de sistema em causa e com o seu domínio de aplicação. As quebras de consistência são originadas, na maioria dos casos, por situações de informação contraditória ou incompleta. Nessas circunstâncias, o agente poderá duvidar da veracidade da informação factual que possui, quer porque dispõe de motivos para não acreditar quer, simplesmente, porque mão detém informação suficiente para manifestar tal segurança.

Os sistemas de verificação de consistência actuam normalmente com as seguintes finalidades:

- manter a consistência da informação factual do agente;
- registar os motivos que permitem ao agente confiar na informação factual que possui;
- identificar informação contraditória;
- garantir a consistência da base de conhecimento do agente, numa situação de emergência de nova informação (por exemplo, a geração de uma solução).

O tipo de acções que os sistemas de verificação de consistência protagonizam faz deles entidades imprescindíveis para o normal funcionamento das

máquinas de inferência dos ABC. A máquina de inferência fornece à unidade de verificação de consistência uma solução (parcial ou final) complementada com uma justificação para o resultado encontrado (e.g. as cadeias de inferência). A unidade de verificação de consistência analisa estes resultados procurando eventuais quebras de consistência na informação armazenada na base de conhecimento do agente.

Poder-se-ão desenvolver diferentes tipos de sistemas implementando estratégias distintas de verificação de consistência:

- **Sistemas de manutenção de consistência baseados em suposições** ("Assumption Based Truth Maintenance Systems" (ATMS)).
  Neste tipo de sistema, o conhecimento a assimilar é validado por um corpo de suposições dadas como verdadeiras (J.deKleer 1986*a*) (J.deKleer 1986*b*).

- **Sistemas de manutenção de consistência baseados em justificações** ("Justification Based Truth Maintenance Systems" (JTMS)).
  Sistemas nos quais a informação em que um agente tem confiança é aquela que ele já teve ocasião de validar (J.Doyle 1979).

- **Sistemas de manutenção de consistência baseados em lógica** ("Logic Based Truth Maintenance Systems" (LTMS)).
  A informação em que um agente tem confiança é apresentada na forma de teorias ou programas lógicos (D.McAllester 1978).

Cada um dos sistemas referidos apresenta vantagens e desvantagens quando comparados em termos de atributos como a eficiência, o modo de operação em tempo real e o tipo de soluções desejado.

Os ambientes distribuídos abriram novas perspectivas e trouxeram novos problemas à implementação e desenvolvimento de SM. A consistência do conhecimento num sistema distribuído exige uma política de revisão de consistência ao nível do agente. Os agentes devem, por isso, estar providos de mecanismos especiais que lhes possibilitem assegurar a consistência do próprio conhecimento e, de parceria com os restantes, garantir que a informação que eventualmente partilhem se mantenha consistente durante os processos de intervenção dos agentes. Um sistema de revisão de consistência deve, por conseguinte, não apenas considerar a análise do conhecimento global, partilhado por toda a comunidade de agentes, mas também despoletar, em cada um dos agentes integrados no ambiente do sistema, as acções necessárias à verificação da consistência dos seus corpos de conhecimento.

Num SM, o problema da manutenção de consistência das bases de conhecimento dos agentes pode ser analisado segundo graus de consistência e graus de "boa fundamentação" ("well-foundedness") (M.Huhns & D.Bridgeland 1991). Em termos de consistência de informação, podem distinguir-se os seguintes graus:

- **Inconsistência**.

  Quando existe pelo menos um agente que possui internamente uma peça de informação em que acredita e para a qual não tem uma justificação válida.

- **Consistência local**.

  As peças de informação que cada agente possui e em que acredita têm uma justificação válida. Não existe, contudo, a garantia de que, globalmente, o sistema seja consistente, já que aquilo em que um agente acredita pode não ter crédito para terceiros.

- **Consistência local e em termos de partilha**.
  Cada agente é individualmente consistente, assim como os aglomerados de agentes em que participa. Todavia, estas não são condições suficientes para garantir uma situação de consistência ao nível global do sistema.

- **Consistência global**.
  Os agentes do sistema são individual e mutuamente consistentes, o que significa que as suas bases de conhecimento poderão ser fundidas numa única entidade sem que nenhuma das partes necessite de ser modificada.

Em termos de graus de "boa fundamentação", um SM pode apresentar situações em que a informação se encontre:

- **Mal fundamentada**.
  Quando um ou mais agentes acreditam em informação cujos argumentos não estão firmados.

- **Bem fundamentada localmente**.
  Quando cada um dos agentes acredita que possui informação que está bem fundamentada. No entanto, pode ter partilhado informação para a qual não encontra uma justificação.

- **Bem fundamentada localmente e em termos de partilha**.
  Quando cada um dos agentes acredita que possui informação bem fundamentada e que toda a peça de informação partilhada apresenta uma justificação válida.

- **Bem fundamentada globalmente**.
  Quando toda a peça de informação possui globalmente uma justificação.

Há, por conseguinte, que definir, de acordo com o domínio de aplicação em causa, a melhor política para aplicação dos mecanismos de verificação de consistência, de forma a evitar que surjam situações de estrangulamento no sistema originadas por estratégias de revisão de consistência demasiado exigentes em termos de recursos computacionais. Contudo, na prática, este compromisso não é fácil de conseguir, tornando-se mesmo, por vezes, inatingível.

Em B.Malheiro et al. (1994), M.Huhns & D.Bridgeland (1991), C.Mason & R.Johnson (1989) e A.Dragoni (1992) desenvolvem-se alguns sistemas de revisão de consistência que podem servir de base a uma nova geração de tais sistemas.

3.9 Sistemas Baseados em Quadros Negros

Os SBQN (P.Nii 1986) (R.Engelmore & T.Morgan 1988) (D.Corkill 1991) têm sido aplicados com sucesso na implementação de SM distribuídos. A sua aptência natural para a conjugação de diferentes tipos de contribuições, resultantes da intervenção de diferentes tipos de especialistas, torna o modelo bastante poderoso para este tipo de aplicações.

Os sistemas baseados neste tipo de paradigma suportam diferentes modelos de organização do conhecimento, disponibilizando as estruturas necessárias para o controlo e utilização dinâmica do mesmo. Estas características permitem que o processo de resolução de um problema e a consequente obtenção de uma solução possam ser efectuados de uma forma incremental e oportunística.

As arquitecturas baseadas em quadros negros têm sido um dos modelos mais inspiradores na resolução de problemas em IA, tendo as suas potencialidades sido exploradas na implementação de sistemas distribuídos, na criação de paralelismo, no desenvolvimento de sistema inteligentes em tempo real e como ferramentas de prototipação rápida (V.Jagannathan, R.Dodhiawala & L.Baum 1989).

O paradigma dos quadros negros faz a sua aparição no domínio da literatura da IA por volta de 1962, por intermédio de Newell, que investigava os problemas organizacionais de programas, muitos deles concebidos sobre modelos de procura orientados por "geração-e-teste" e cujo principal problema era a rigidez. Newell utilizou então o termo "quadro negro" na descrição do sistema que tinha idealizado, e que lhe permitia manter um conjunto de rotinas isoladas mas utilizando uma estrutura de dados comum. Tal estratégia veio aliviar o grau de inflexibilidade em termos de controlo e acessibilidade de dados restritos dos programas, que na altura utilizavam estruturas ditas tradicionais (P.Nii 1986).

Uma etapa significativa no uso do modelo é marcada pela sua aplicação no desenvolvimento do sistema HEARSAY-II (R.Engelmore, AMorgan & H.P.Nii 1988*a*). Em R.Engelmore, AMorgan & H.P.Nii (1988*b*) pode-se encontrar uma relação cronológica do desenvolvimento de ideias e sistemas desde o projecto do HEARSAY-II.

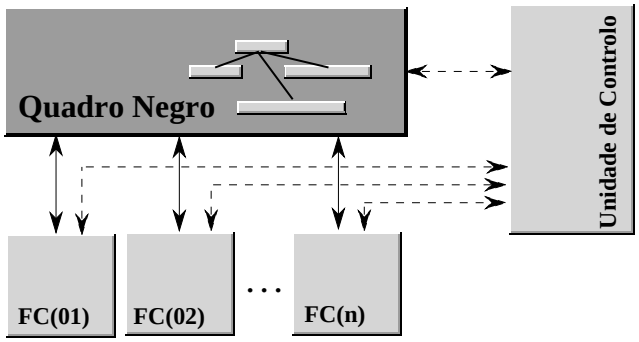


Figura 3.5: Ilustração do Modelo Genérico de uma Arquitectura baseada em Quadros Negros.

Um SBQN (Figura 3.5) apresenta genericamente, em termos organizacionais, três componentes básicos (D.Corkill 1991):

- **As Fontes de Conhecimento**.

  Componentes constituídos por módulos independentes que contêm o conhecimento necessário à resolução de problemas. Cada uma das fontes tem conhecimento das normas pelas quais deve pautar a sua acção. A sua granularidade é superior à apresentada pela base de regras de um SP tradicional. Contudo, as fontes de conhecimento são meros repositórios de informação.

- **O Quadro Negro**.

  Componente que se comporta como uma base de dados global, podendo conter dados de entrada, soluções parciais ou outro tipo de informação, conforme o estado da resolução do problema. É um componente do sistema acessível a todas as fontes de conhecimento e que se apresenta como uma estrutura de memória partilhada onde são colocados dados em "bruto", soluções intermédias e/ou finais e, naturalmente, informação

para suporte às acções de controlo do sistema. Funciona como o meio através do qual se faz a comunicação entre as fontes de conhecimento.

- **A Unidade de Controlo**.

  Este componente tem como função básica dirigir o processo de resolução de problemas, permitindo que cada uma das fontes de conhecimento actue de forma oportunística conforme as alterações de estado do quadro negro. A unidade de controlo decide, com base no estado corrente do quadro negro e no conjunto das activações de cada uma das fontes de conhecimento, qual o melhor caminho a seguir para a resolução do problema.

O processo de resolução de um problema deve ser simultaneamente incremental e oportunístico. Numa perspectiva incremental, as soluções são construídas "peça a peça" e a diferentes níveis de abstracção. A estratégia a seguir é muitas vezes referida como um processo incremental de hipóteses e de testes ou como agregação de evidências. Este processo envolve, em primeiro lugar, a geração de uma hipótese de solução (parcial) com base em dados incompletos e, de seguida, a obtenção de dados adicionais que confirmem tal hipótese. Desta forma, a fase de teste pode servir como um processo de refinação das hipóteses geradas, assim como uma forma de ultrapassar as situações de incerteza relacionadas com essas hipóteses. Numa óptica oportunística, a obtenção de uma solução pressupõe que o sistema defina as acções a executar, o que lhe permitirá escolher a melhor sequência destas (N.Carver & V.Lesser 1992).

A adopção de uma arquitectura baseada em quadros negros na implementação de um sistema distribuído permite alcançar, concretamente, o seguinte conjunto de objectivos (S.Shapiro 1990*a*):

- **Redução das combinações de um processo de procura**.
  Uma arquitectura baseada em quadros negros potencia o raciocínio a diferentes níveis de abstracção, providencia independência entre as diversas fontes de conhecimento e disponibiliza meios de escalonamento ("scheduling") oportunístico.

- **Incorporação de diversos tipos de fontes de conhecimento num único sistema de resolução de problemas**.
  Visa preservar a distinção entre as fontes de conhecimento, permitindo que estas possam acolher diferentes tipos de perícias e formas de representação de conhecimento.

- **Compensação pela não fiabilidade do conhecimento disponível**.
  Permite que as fontes de conhecimento possam operar de forma redundante sobre o mesmo subproblema. As aplicações podem combinar diferentes intervenções de várias fontes de conhecimento não fidedignas, por forma a conseguir formas de solução credíveis.

- **Compensação da incerteza verificada ao nível dos dados**.
  Permite que as fontes de conhecimento possam incorporar diferentes métodos de pesquisa e dados, podendo as aplicações do sistema explorar qualquer um desses métodos, de acordo com as condições no momento.

- **Aplicação do conhecimento disponível de forma inteligente na ausência de um algoritmo de resolução de problemas**.
  Uma arquitectura baseada em quadros negros disponibiliza meios de escalonamento oportunístico, que permitem decidir, a cada momento, qual a acção a executar.

- **Suporte para o desenvolvimento de sistemas cooperativos**.
  As fontes de conhecimento, sendo funcionalmente independentes, podem ser encaradas como módulos e ser idealizadas e desenvolvidas independentemente.

- **Suporte à evolução do sistema e suas modificações**.
  As funcionalidades das fontes de conhecimento podem ser acrescentadas, modificadas ou removidas separadamente. Existe uma distinção forte entre o domínio do conhecimento e os mecanismos de escalonamento. As modificações que as fontes de conhecimento possam eventualmente sofrer não afectam a componente de escalonamento, e vice-versa.

São muitos os domínios do conhecimento aos quais o modelo dos quadros negros poderá ser aplicado, porém, ele revela aptência natural para problemas que possam envolver (P.Nii 1989):

- grandes espaços de soluções;
- informação não garantida ou com ruído;
- uma vasta variedade de dados de entrada;
- a integração de diversos tipos de informação;
- a adopção de diversas formas de cooperação entre diferentes entidades, independentes ou semi-independentes, na procura de uma solução;
- a utilização de diversos métodos e formas de raciocínio;
- soluções que possam evoluir ao longo do tempo.

Devido à sua filosofia e características de base, o modelo dos quadros negros tem sido adoptado e explorado num leque muito alargado de arquitecturas e sistemas aplicacionais (V.Jagannathan et al. 1989) (R.Engelmore & T.Morgan 1988), tendo sofrido inúmeras alterações, que vão de simples complementos em termos da filosofia de activação das fontes de conhecimento até novas políticas de distribuição das próprias acções de controlo pelas fontes de conhecimento do sistema. Muitas destas evoluções foram provocadas, naturalmente, pelas especificidades do domínio de conhecimento dos problemas ou pelos próprios requisitos de implementação.

A adopção do modelo dos quadros negros na implementação de SM é facilmente compreendida, quando se substituem as fontes de conhecimento por agentes, uma vez que permite criar um ambiente com as estruturas e funcionalidades necessárias ao suporte das principais actividades de uma comunidade de agentes: coordenação, cooperação e negociação.

3.10 O Modelo Adoptado

O sistema BEABLE aplica muitos dos princípios apresentados pelos SBQN. As características e funcionalidades pretendidas para o sistema e o tipo de aplicações a que se destina (Capítulo 1 Secção 1.3) conduziram a um modelo com características em tudo semelhantes às disponibilizadas por um qualquer SBQN. Na verdade, ao analisar-se a arquitectura da actual versão do sistema e a forma como as suas entidades estão estruturadas, verifica-se que as características e funcionalidades de base não são mais do que "espelhos" das existentes numa arquitectura convencional de quadros negros.

Em termos gerais, a adopção de um modelo baseado em quadros negros é sus-

tentada pelas opções de conceptualização e implementação da arquitectura global do sistema e, em particular, pelos seguintes motivos:

- haver necessidade de integração de diferentes tipos de conhecimento num único ambiente;
- o facto de o modelo dos quadros negros providenciar um canal de comunicação comum entre os diversos agentes do sistema, disponibilizando as estruturas para o suporte à partilha de conhecimento e distribuição de tarefas e para o armazenamento da informação de controlo necessária à coordenação das actividades de cada agente;
- ser necessário que as acções desencadeadas pelos agentes do sistema, tanto a nível individual como colectivo, sejam objecto de coordenação e controlo através de informação mantida no quadro do sistema, por forma a evitarem-se eventuais situações de contenção ou estrangulamento, normalmente originadas por esquemas apertados de controlo centralizado;
- basearem-se os SBQN numa filosofia de descentralização de tarefas que potencia o aparecimento de formas de cooperação entre diferentes entidades, facultando-lhes o meio para uma actuação oportunística em processos de geração incremental de soluções;
- apresentarem os SBQN fortes características modulares, o que permite aumentar significativamente o grau de autonomia dos diversos agentes do sistema, incrementar a taxa de reutilização de componentes e facilitar as operações de manutenção, com a consequente redução de custos associados;

- oferecerem os SBQN um excelente meio de integração e combinação de diferentes tipos de conhecimento, perícias e estratégias de resolução de problemas.

A arquitectura tradicional de um modelo genérico de um SBQN apresenta uma entidade de controlo global centralizada, que, apesar de ser útil e vantajosa em alguns tipos de aplicações, não o é no caso particular do sistema BeAble. A opção de implantar uma entidade de controlo com tais características provocaria sérios casos de estrangulamento e contenção nas actividades regulares dos agentes do sistema, independentemente da sua classe. Desde o início do projecto que se optou pela implementação de uma estratégia de controlo descentralizada, definida à custa do desenvolvimento de processos de organização cooperativos suportados pelas estruturas locais de cada agente do sistema. Esta medida tornou possível o desenvolvimento para o sistema de um modelo computacional bastante flexível e robusto em termos operacionais.

Capítulo 4

O Modelo Computacional do Sistema

Apresentação geral do modelo computacional do sistema. Características e funcionalidades principais, arquitectura, ambiente distribuído, definição das classes de agentes que podem ser integradas no ambiente do sistema e linguagem de comunicação global. Aspectos de implementação e de instalação do sistema. Características, arquitectura e funcionalidades dos agentes de supervisão, monitorização e suporte operacional.

Muitas das aplicações baseadas em agentes, nomeadamente as relacionadas com problemas de diagnóstico, controlo ou classificação, necessitam de lidar com situações de carácter rotineiro e, muitas vezes, não previsível. Tal circunstância vem em defesa da necessidade de se possuirem agentes capazes de actuar de forma inteligente, sempre que possível apoiados pela sua própria experiência, eficientes e efectivos no emprego e reajustamento das estratégias de resolução de problemas, na procura de uma solução e, eventualmente, na execução de um plano de trabalho. Baseando-se neste tipo de conduta, os

agentes que partilhem um dado ambiente podem desenvolver entre si uma espécie de comportamento social, promovendo acções de cooperação ou, simplesmente, desenvolvendo processos de troca de conhecimento.

Com base nestes pressupostos, na observação de modelos de comportamento humanos e no estudo de alguns exemplos de organizações sociais, foi desenhado e desenvolvido um ambiente distribuído para a implementação de SM: o sistema BEABLE (O.Belo & J.Neves 1996*a*) (O.Belo & J.Neves 1995*a*).

O sistema é um tipo particular de ambiente integrativo orientado para a resolução de problemas por recurso a SM, de forma cooperativa e distribuída. Os agentes podem ser SBC, simples plataformas de interacção para os utilizadores do sistema ou entidades puramente computacionais capazes de emular algum tipo de dispositivo físico. Têm, além disso, a capacidade de entrar ou sair do sistema de forma dinâmica, não necessitando para o efeito de realizar qualquer tipo de procedimento especial. O grau de autonomia dos agentes e os mecanismos de controlo que o sistema possui permitem que a saída/entrada dos agentes no sistema seja um acto facilmente programável e controlável pelos administradores dos mesmos.

O ambiente do sistema é acessível a todas as entidades. Sendo reconhecidas pelo sistema e integradas no seu ambiente, estas podem aceder ao tráfego de conhecimento que nele se desenvolve. Para isso, apenas precisam de conhecer a chave e a forma de manipular adequadamente esse conhecimento. As acções que os agentes desenvolvem no ambiente do sistema acarretam a modificação do seu estado de conhecimento ao longo do tempo. Esta evolução, alcançada através das sucessivas mutações que as estruturas de conhecimento vão sofrendo, torna o ambiente do sistema dinâmico. Esse dinamismo é, naturalmente, proporcional ao número de agentes no sistema, à complexidade do

problema em questão e ao volume de conhecimento requerido pelo processo de resolução em curso.

As acções de controlo estão distribuídas por todos os componentes do sistema, de acordo com as suas responsabilidades e capacidades de intervenção, por forma a evitar situações de contenção ou de bloqueio fatal ("deadlock").

O sistema acolhe várias classes de agentes, definidas de acordo com o seu tipo de actividade: administração, monitorização, resolução, interface ou suporte. Em termos gerais, comporta-se como uma comunidade de agentes, no seio da qual cada um dos elementos tem um papel a desempenhar, um conjunto de funções específicas, directamente dependentes do seu nível de conhecimento, perícia e aptidão social.

Os agentes do sistema, atendendo às potenciais áreas de aplicação, são entidades benévolas, que não prejudicam as actividades dos seus pares.

4.1 Apresentação do Sistema

Um dos objectivos principais deste projecto era o desenvolvimento de um ambiente computacional tão "genérico" quanto possível que, de uma forma simples, disponibilizasse os meios e as estruturas necessárias aos seus utilizadores para que pudessem usufruir da perícia e conhecimento de um conjunto de especialistas em diferentes domínios de aplicação.

O sistema deveria também permitir a integração de novos agentes, a actualização do seu conhecimento e a optimização das suas estruturas de raciocínio e controlo de forma flexível, não condicionando a sua actividade à realização destas mesmas operações.

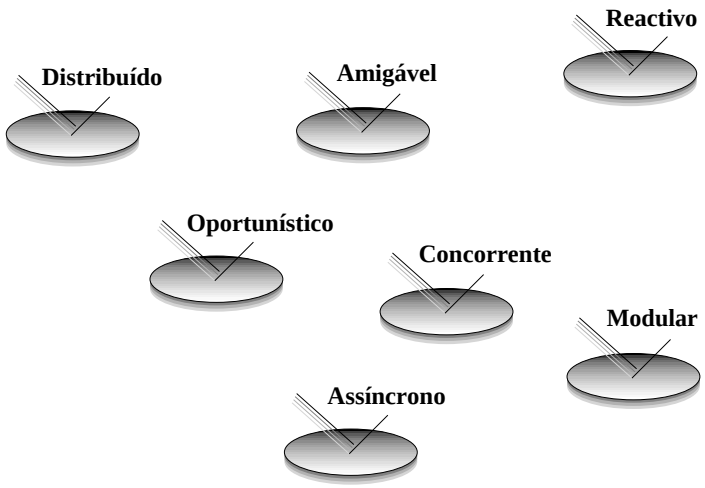


Figura 4.1: Características Globais do Sistema.

Os agentes do sistema, que actuariam de acordo com o conhecimento que possuíssem em determinadas áreas, poderiam agir isoladamente, quando tal fosse directamente solicitado, desenvolvendo processos de resolução personalizados, ou em equipas, resolvendo conjuntamente problemas de natureza multidisciplinar. Adicionalmente, cada agente especialista deveria ser capaz de gerir e atender simultaneamente mais do que um utilizador.

O sistema deveria possuir uma arquitectura robusta, capaz de suportar a actividade concorrente de vários agentes e a utilização simultânea de diversos utilizadores.

Pretendia-se, igualmente, que os componentes do sistema pudessem ser distribuídos através das estruturas de uma rede de computadores, por forma a acompanhar as necessidades de utilização e de recursos envolvidos, a providenciar um suporte não centralizado de conhecimento e de perícia e a facilitar as acções de manutenção por parte das equipas de desenvolvimento dos agentes. Desta forma, conseguir-se-ia aumentar significativamente a robustez de

cada um dos agentes em particular, garantindo-lhes um maior grau de autonomia.

Durante o processo de estudo, planeamento e desenho do sistema, todas as características referidas anteriormente (Figura 4.1) estiveram presentes de alguma forma, constituindo elementos-chave nos critérios de implementação assumidos. Facilmente se identifica, através da análise da arquitectura, das funcionalidades globais do sistema e dos seus componentes, a presença, de uma forma ou de outra, dessas mesmas características.

A distribuição do conhecimento envolvido nos processos de resolução pelos diversos locais onde os agentes se encontram originou a criação de estruturas localizadas (ou quadros negros) para suporte aos processos de cada agente e para o armazenamento do seu conhecimento: as memórias de trabalho dos agentes. O ambiente do sistema pode ser, em certos aspectos, comparável a um quadro negro global, cuja principal função é assegurar um meio de comunicação directo entre os agentes do sistema.

O conhecimento que o ambiente do sistema tem armazenado é basicamente constituído por algumas estruturas de controlo dos processos de inferência desenvolvidos pelos vários agentes do sistema, por conhecimento transmitido pelos utilizadores ou pelos próprios agentes durante esses processos, bem como por informação dispersa relativamente ao controlo dos processos desencadeados pelos agentes activos no sistema. À excepção desta última, toda a informação referida está armazenada nas memórias de trabalho dos diversos agentes, como requisito de segurança e meio de ultrapassar eventuais anomalias verificadas no processo de comunicação entre agentes. Por outro lado, esta duplicação de informação permite reduzir o número de processos de comunicação interagente, possibilitando ainda uma mais fácil recuperação da

informação eventualmente perdida ou "danificada" ao nível do agente.

A partir do momento em que o sistema é activado, os agentes poderão actuar como servidores ou como clientes de informação do sistema. Tais desempenhos são directamente influenciados pelo tipo, aptidões, funcionalidades e processos específicos dos próprios agentes. A disponibilização, a qualquer momento, do estado de um problema a toda a comunidade de agentes do sistema, através do seu ambiente, permite que estes possam verificar até que ponto, e em que momento, podem ou não intervir no processo.

Os agentes do sistema são activados pelos seus administradores, à excepção dos agentes que suportam os processos de interacção dos utilizadores com o sistema, que são activados pelos próprios utilizadores no momento em que estes solicitam um processo de consulta.

Os agentes do sistema revelam-se maioritariamente como entidades oportunísticas. Este tipo de comportamento permite-lhes eliminar a necessidade de desenvolver planos iniciais para a sua intervenção no sistema. Tal possibilidade reduz significativamente o esforço inicial deste na preparação prévia das tarefas dos seus agentes. Noutro tipo de comportamento, não oportunístico, tais planos teriam que ser elaborados de acordo com o domínio do problema em causa, por forma a estabelecer as formas e os momentos de intervenção de cada um dos agentes. Em particular, ao providenciar-se um tipo de comportamento oportunístico, é possível melhorar efectivamente a dinâmica dos agentes do sistema, com realce para o seu comportamento assíncrono. No entanto, desta forma, o desenvolvimento de acções cooperativas entre os agentes é algo mais complexo, já que é necessário criar mecanismos para a inibição, em algumas situações, da sua natureza de auto-iniciativa. Com estes mecanismos de inibição de comportamento conseguem-se evitar eventuais

intervenções menos desejáveis, e pouco produtivas, dos agentes nos processos em curso no sistema, optimizando o seu próprio desempenho.

Os agentes podem cooperar de várias maneiras: através dos resultados (intermédios e/ou finais) que cada um deles produz e que são comunicados posteriormente ao ambiente do sistema no qual estão inseridos, através de uma solicitação directa ou através da passagem de controlo dos processos em curso de agente para agente.

A comunicação de conhecimento entre agentes é efectuada através do ambiente do sistema. Os agentes do sistema também comunicam, embora neste caso indirectamente, através da partilha de estruturas de conhecimento contidas no ambiente do sistema, sobre as quais efectuam operações de inserção/remoção de conhecimento. Este tipo de operações altera o estado global de conhecimento do sistema, podendo influenciar os processos em curso. Isto é, o estado e a dinâmica do ambiente do sistema (caracterizada em termos gerais pela circulação e mutação das suas estruturas de conhecimento), conjugados com o estado, o conhecimento e a perícia dos agentes, permitem o desenvolvimento e a evolução dos processos em curso no sistema.

A concorrência revela-se no sistema de duas formas distintas. Em primeiro lugar, os componentes do sistema podem observar em simultâneo as estruturas de conhecimento (de controlo e de dados) que o ambiente do sistema possui em cada momento e reagir aos estímulos colocados pelos outros componentes. A segunda forma diz respeito à própria utilização de diferentes componentes do sistema que, quando activados, não interferem directamente com os processos em curso em outros componentes. Isto é, um qualquer componente do sistema pode dar curso a diferentes processos, eventualmente até relacionados com um problema que envolva vários tipos de agentes, sem

causar nenhum tipo de interferência directa nos processos dos outros agentes.

A arquitectura do sistema potencia um modelo híbrido com características de um sistema de memória partilhada conjugadas com as de um modelo orientado por troca de mensagens.

Os motivos que suportam tal conjugação, no caso de uma arquitectura de memória partilhada, são os seguintes:

- o desejo de uma aproximação ao modelo computacional suportado por uma arquitectura baseada em quadros negros, pelas razões expostas no Capítulo 3, Secção 3.10;
- as facilidades que apresenta para a exploração de processos paralelos e concorrentes, ao nível dos agentes do sistema;
- a disponibilidade de um modelo onde a implementação de uma base de dados ou de conhecimento centralizada para o armazenamento do conhecimento envolvido nos processos cooperativos dos agentes está naturalmente simplificada;
- o facto de assegurar um meio de comunicação directo e expedito entre os diversos agentes do sistema.

No caso de uma arquitectura orientada por troca de mensagens, estão em causa razões como as que seguem:

- existência de um suporte facilitado para a definição dos processos de comunicação entre os agentes durante o desenvolvimento de processos de cooperação ou de simples troca de informação;

- disponibilização de um bom processo para a modelação da comunicação com os utilizadores do sistema, com a comunicação interagentes, com ferramentas computacionais ou sistemas de interface homem-máquina.

4.2 Os Componentes do Sistema

A organização dos diversos tipos de componentes do sistema, através de uma divisão por classes, é apresentada na Figura 4.2. As classes foram definidas de acordo com as funcionalidades associadas a cada um dos componentes do sistema, quando integrados na sua estrutura funcional.

O sistema pode incorporar seis diferentes classes de agentes:

1. **Agentes de Administração**.
   Classe constituída apenas por um único agente, responsável pelas operações de arranque e manutenção do sistema e pela observação das actividades dos restantes agentes activos no sistema.
   Agentes: *Supervisor (Su).*

2. **Agentes Cognitivos**.
   Integram os especialistas disponíveis no sistema, que mais não são que SBC cuja perícia e conhecimento estão definidos num domínio de aplicação específico.
   Agentes: *AgC(1) ... AgC(m)*, $(m \geq 0)$.

3. **Agentes de Interface**.
   Possuem os meios e disponibilizam os serviços para uma interacção homem-máquina, permitindo que os utilizadores do sistema possam

usufruir do conhecimento e perícia dos agentes cognitivos.

Agentes: *AgI(1) ... AgI(n)*, ($n \geq 0$).

4. **Agentes Encaminhadores**[1].

   Permitem estabelecer elos de ligação lógicos entre dois ambientes distintos BEABLE (potenciando a ligação entre diferentes comunidades de agentes), definindo as formas e as normas de comunicação entre os agentes cognitivos das diferentes comunidades. Nesta classe apenas há a considerar um agente : o encaminhador.

   Agentes: *Encaminhador (En)*.

5. **Agentes Guardiões**.

   Possibilitam o acesso ao sistema a partir do exterior, através de um "browser" da Internet. Têm como função estabelecer e garantir a satisfação das normas de acesso ao sistema.

   Agentes: *AgG(1) ... AgG(l)*, ($l \geq 0$).

6. **Agentes de Suporte**.

   São responsáveis pela disponibilidade e manutenção do ambiente do sistema e pela monitorização das acções protagonizadas pelos agentes activos do mesmo.

   Agentes: *Monitor* (Mo) e *"Motor" do Sistema* (MS).

De notar que o número de agentes referidos para cada uma das classes corresponde apenas a uma instanciação de uma comunidade de agentes do sistema.

No Capítulo 6 será apresentada e analisada a problemática da implementação de modelos baseados em múltiplas comunidades de agentes através da replicação do sistema BEABLE.

[1] "Routers", na terminologia inglesa.

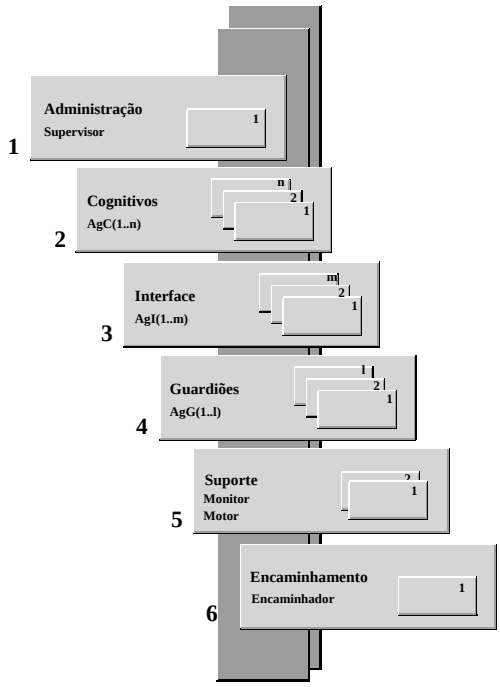


Figura 4.2: Classes de Agentes do Sistema.

4.3 O Ambiente do Sistema

O modelo que suporta a actual arquitectura do sistema (Figura 4.3) desenvolve-se em torno de uma estrutura de dados global: o ambiente do sistema. Este comporta-se não só como o meio através do qual há comunicação entre os diversos componentes do sistema, mas também como dispositivo de armazenamento de informação de controlo, assim como de parte das estruturas de conhecimento necessárias aos processos em curso.

O conhecimento associado aos processos de consulta em curso, relacionado com os utilizadores, não se encontra localizado nessa estrutura, mas nas plataformas computacionais em que os agentes envolvidos estão localizados.

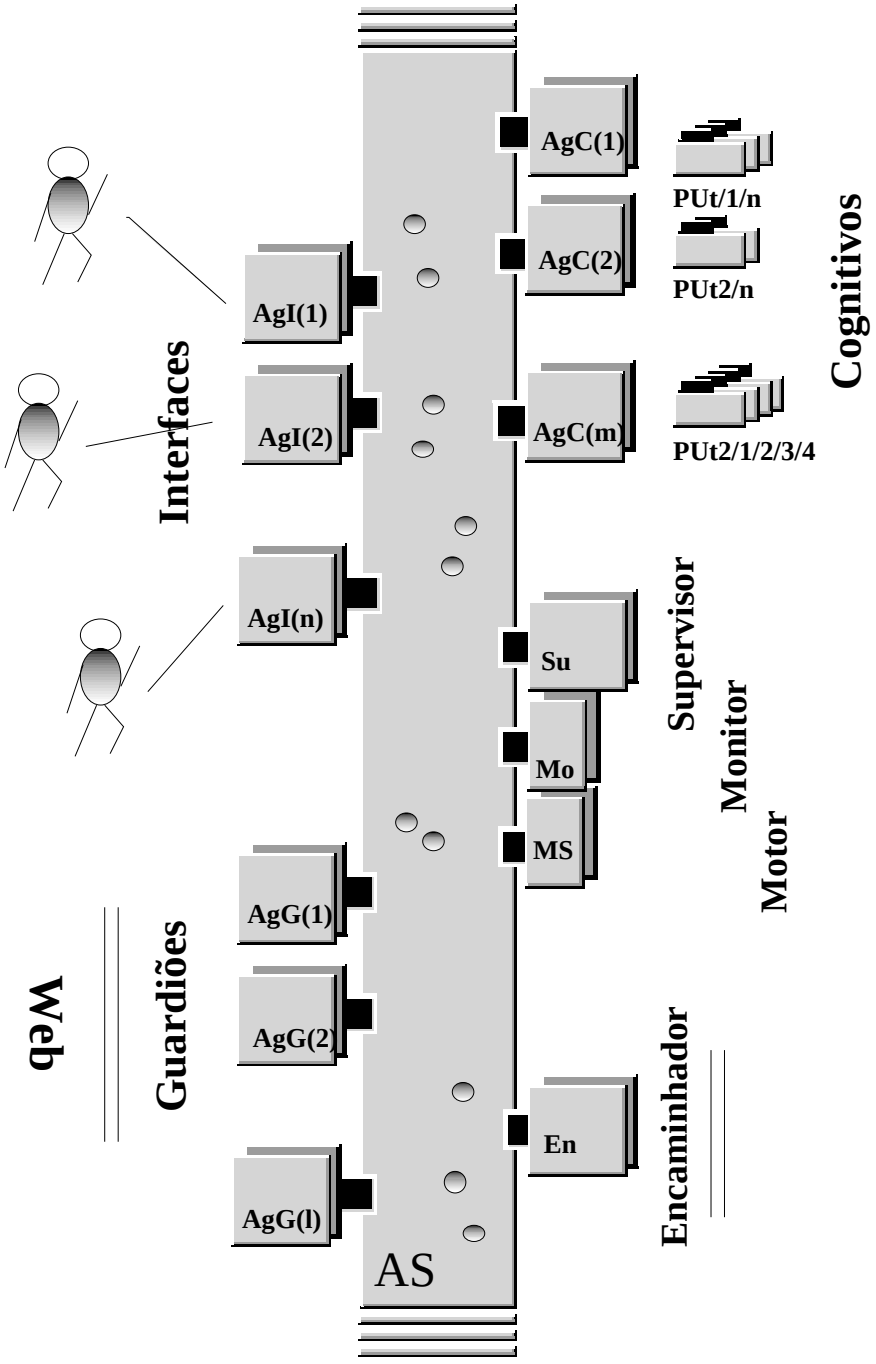


Figura 4.3: Modelo da Arquitectura Global do Sistema.

Esta característica, além de aliviar a estrutura de comunicação, em termos de necessidades de memória, permite uma maior eficiência, rapidez e fiabilidade nos processos e na troca de informação entre agentes, além de facilitar a recuperação dos agentes em eventuais situações de falha do sistema.

A definição inicial do ambiente do sistema não é afectada pela forma como os agentes, e outros componentes do sistema, estão distribuídos pela rede de computadores. Todo o tráfego de informação (estruturas de controlo e de gestão e conhecimento relacionado com os processos) entre os componentes do sistema circula pelo ambiente do mesmo. Com a activação do ambiente estão criadas as condições para que os agentes do sistema possam comunicar entre si e, consequentemente, desenvolver formas de cooperação.

A utilização de uma estrutura dedicada à comunicação entre os agentes é deveras vantajosa. Nela é armazenado o conhecimento necessário aos processos de gestão e operação do sistema, ao desenvolvimento de acções ccooperativas entre os agentes, o que a torna um meio privilegiado para o tipo de filosofia de organização e funcionamento a implementar pelo sistema.

4.4 A Linguagem de Comunicação

A comunicação em ambientes distribuídos assume um papel central no estabelecimento e na manutenção dos processos de interacção entre as entidades que aí estão integradas. Nos SM, a comunicação é um processo vital para o correcto entendimento entre os diversos tipos de agentes e, naturalmente, para o desenvolvimento de acções cooperativas entre os mesmos. Assim, é desejável que se assegurem canais de comunicação robustos e fiáveis entre os agentes, bem como uma "forma" de entendimento comum, definida em torno

de uma linguagem de comunicação "universal".

A linguagem de comunicação utilizada pelos agentes do sistema foi desenvolvida no sentido de assegurar um conjunto de protocolos de comunicação de alto nível e de fácil manutenção. Essa linguagem disponibiliza também as estruturas necessárias para que o sistema possa suportar dois tipos de comunicação entre os seus agentes:

- **Directa**.

  Que se materializa em termos de troca directa de mensagens entre agentes, relacionadas com a passagem de controlo ao nível das cadeias de inferência dos agentes cognitivos.

- **Indirecta**.

  Que se desenvolve através de processos de partilha de informação entre os agentes implicados em operações de aquisição de conhecimento - inserção, modificação ou remoção - sobre estruturas de conhecimento especiais, localizadas no ambiente do sistema e destinadas a suportar os processos cooperativos. Estas acções são visíveis para os restantes agentes cognitivos do sistema, levando-os, porventura, a intervir.

Os processos de comunicação directa, por sua vez, podem desenvolver-se de duas formas distintas:

- **Personalizada**.

  O agente remetente envia uma mensagem a um dado conjunto de agentes, os destinatários.

- **Difusa**.

  O agente remetente envia uma mensagem para o ambiente do sistema,

sendo, neste caso, potenciais destinatários todos os agentes activos no sistema.

As formas de comunicação referidas em epígrafe são globais, no entanto, esses tipos subdividem-se de acordo com os processos de comunicação que cada uma das classes de agentes pode desenvolver. Assim, é possível identificar seis tipos de comunicação interagentes:

- **Supervisor/Agentes do Sistema**.
  Forma de comunicação normalmente associada às acções de gestão e administração do sistema.

- **Agentes de Interface/Agentes Cognitivos**.
  Forma de comunicação despoletada pelos agentes cognitivos, no seu relacionamento com os agentes de interface.

- **Agentes Guardiões/Agentes Cognitivos**.
  Forma de comunicação muito semelhante às anteriores, mas diferindo em termos de suporte aos utilizadores, pois em vez de recorrer a um agente de interface do sistema, utiliza um "browser" da Internet.

- **Agentes Cognitivos/Agentes Cognitivos**.
  Forma de comunicação que ocorre quando há cooperação entre os agentes, materializando-se numa troca directa de mensagens ou através das estruturas de partilha de informação que suportam os processos de comunicação indirecta.

- **Agentes Cognitivos/Encaminhador**.
  Forma de comunicação que ocorre nas situações em que um agente cognitivo não encontra no ambiente do sistema de que faz parte o conheci-

mento ou a perícia de terceiros que procura, solicitando, por isso, ajuda a um ambiente externo, através de mensagens especiais endereçadas ao agente encaminhador do sistema.

- **Agentes do Sistema/Monitor**.
  Forma de comunicação que ocorre quando os agentes do sistema (cognitivos, de interface, guardiões ou encaminhador) detectam que o agente monitor do sistema está activo. Nessa circunstância, devem comunicar-lhe, através de mensagens, todas as acções em que participam.

Os agentes remetentes e os destinatários possuem mecanismos de protocolo próprios para a interpretação de mensagens, ficando assim assegurado o entendimento mútuo.

Na Figura 4.1 apresentam-se alguns dos tipos de mensagem mais utilizados pelos agentes do sistema.

Não é colocada de parte a possibilidade de os agentes cognitivos do sistema possuirem a sua própria linguagem. Nesse caso, para que seja possível assegurar a interpretação das mensagens e a compreensão das estruturas de conhecimento partilhadas, os agentes deverão possuir mecanismos que lhes permitam realizar as pontes entre a sua própria linguagem e a linguagem do sistema. Estes mecanismos, a serem implementados, implicarão a acoplagem de software específico às "shells" que suportam os agentes cognitivos do sistema.

Mensagem/Aridade	Descrição da Mensagem
agi_stymulus/2	Lista de estímulos enviada por uma agente de interface para o início de um processo cooperativo.
agc_question/4	Questão gerada por um agente cognitivo por forma a obter algum tipo de informação relacionado com a tarefa em curso.
agc_why/3	Pedido de um agente de interface a um agente cognitivo: justificação para uma dada questão.
agc_why/4	Justificação de um agente cognitivo a um agente de interface.
agc_solution/5	Apresentação de uma solução por um agente cognitivo a um agente de interface.
agi_explain/3	Pedido de explicação de um agente de interface a um agente cognitivo sobre o modo de interpretar uma solução.
agc_explain/4	Explicação de uma solução por parte de um agente cognitivo a um agente de interface.
agc_methods/4	Pedido de um agente de interface a um agente cognitivo sobre a forma de tratar uma solução.
agc_generics/2	Pedido de um agente de interface a um agente cognitivo acerca da descrição de funcionalidades, conhecimento e perícia.
ags_agr_kill/0	Indicação do agente supervisor ao agente encaminhador para dar por terminada a sua intervenção.
(...)	(...)

Tabela 4.1: Exemplos-tipo de Mensagens.

4.5 Análise e Classificação das Características do Sistema

O modelo para a classificação de sistemas apresentado em T.Wittig (1992) é uma base adequada para a categorização do sistema, tendo em consideração o seu espectro aplicacional e a análise das suas características segundo os seguintes factores:

- **Modelo** - *Nível de Distribuição.*

  Possíveis instanciações das diferentes classes de agentes constituem-se

em comunidades de agentes.

- **Granularidade** - *Nível da Decomposição do Problema.*

  O sistema é constituído por entidades de grande granularidade, como é o caso dos agentes cognitivos que são SBC.

- **Escala** - *Nível de Paralelismo* (larga, média e pequena).

  - O número de processos computacionais não ultrapassou, durante as diversas etapas de teste e validação do sistema, as duas dezenas (supervisor, agentes de interface, agentes cognitivos, etc.), o que nos revela um sistema de pequena escala.

- **Agentes** - *Nível de Autonomia.*

  A maioria das classes de agentes do sistema é constituída por agentes autónomos, independentes em termos de perícias, mas em parte dependentes do conhecimento de outras fontes ou agentes.

- **Controlo**.

  A maioria das acções de controlo do sistema está distribuída pelos seus diversos agentes.

- **Independência**.

  Característica a ser analisada segundo duas perspectivas:

  - **Racionalidade e Responsabilidade**.

    Um agente pode alcançar uma solução ou tomar decisões individualmente.

  - **Domínio do Problema**.

    Os agentes do sistema, em particular os cognitivos, são independentes do domínio do problema, já que são tipicamentes "shells" de aplicação genérica.

- **Construção** (decomposição ou síntese).

  O sistema actua como um elo de ligação entre os diversos agentes que o materializam, podendo-se, assim, encarar esta sucessiva reunião de agentes como um processo de síntese, cujo resultado é um SM heterogéneo.

- **Uniformidade** (homogénea ou heterogénea).

  O sistema é heterogéneo, já que permite a integração no seu ambiente de diferentes tipos de agentes, com diferentes perícias e conhecimentos, além de possibilitar a integração de outros tipos de entidades, através do desenvolvimento de plataformas específicas.

- **Recursos** - *Nível de Restrições.*

  Os recursos envolvidos na resolução de problemas estão distribuídos pelos diversos agentes, segundo uma estratégia de distribuição de responsabilidades, à excepção das estruturas de conhecimento partilhadas pelos processos de cooperação e das utilizadas em acções de controlo indirecto.

- **Interacção** - *Níveis de Complexidade.*

  Os agentes interagem através de mensagens; porém, durante os processos de cooperação e eventual negociação, podem partilhar estruturas de conhecimento que se encontram acessíveis através do ambiente do sistema.

4.6 Implementação do Sistema

O protótipo do sistema foi desenvolvido em SICStus Prolog (M.Carlsson & J.Widen 1993) em ambiente X Windows e implementado numa rede de

workstations com suporte TCP/IP.

Os mecanismos de suporte à comunicação entre agentes foram desenvolvidos com base nas primitivas do Linda, disponíveis no SICStus Prolog (J.Almgran & S.Andersson 1993).

O modelo do Linda (S.Ahuja, N.Carriero & D.Gelertner 1986) (N.Carriero & D.Gelernter 1989*a*) (N.Carriero & D.Gelernter 1989*b*) foi desenvolvido por David Gelernter na Universidade de Yale e assenta, tipicamente, num modelo de memória partilhada por um conjunto de processos distribuídos por várias plataformas computacionais.

Primitiva	Descrição
"out"	Operação que permite a escrita de um tuplo na zona de memória do Linda.
"in"	Operação utilizada para retirar tuplos do "Tuple Space"; actua como um processo de leitura com remoção.
"rd"	Operação similar à anterior, mas que, em vez de remover o tuplo da zona de memória, faz uma cópia do mesmo.
"eval"	Operação que permite criar tuplos com o objectivo de realizar determinado conjunto de acções.

Tabela 4.2: Primitivas de Manipulação de Dados do Linda.

A zona de memória do Linda, designada por "Tuple Space", além de suportar comunicações entre os processos, permite-lhes também a partilha de conhecimento. O tipo de conhecimento que flui neste espaço de memória partilhada apresenta-se sob a forma de tuplos. O Linda interpreta os tuplos como colecções de dados ordenados. A sua manipulação é feita através de um conjunto de primitivas básicas do Linda (Tabela 4.2). Os tuplos que são colocados na zona de memória do Linda por um processo em particular ficam disponíveis a qualquer outro processo que tenha acesso "Tuple Space"

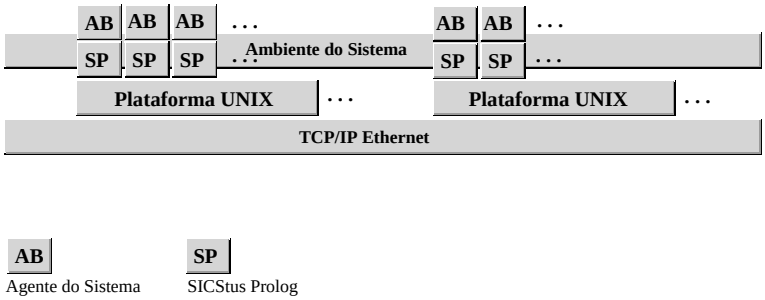


Figura 4.4: Níveis de Software do Sistema.

do Linda.

O sistema utiliza o Linda para a criação e suporte do seu ambiente distribuído, usando as suas primitivas de base para o desenvolvimento dos mecanismos de comunicação e de protocolo entre e intra-agentes. A forma como interactuam (leitura e escrita de tuplos da zona de memória do Linda) é assegurada pelas primitivas do próprio Linda.

Para o desenvolvimento e teste do sistema foram utilizadas, essencialmente, workstations SOLBOURN s4000, SUN compatível, com a seguinte configuração: processador Sparc, 24 MB de memória RAM, 200MB de disco rígido, 25.5 MIPS, 1.7 DP MFLOPS, 13 SPECmark e sistema operativo OS/MP UNIX com X Windows. A Figura 4.4 apresenta os diversos níveis de software utilizados na instalação e utilização do sistema.

4.7 Instalação

A Figura 4.5 apresenta uma possível instalação do sistema. Pode-se observar a distribuição dos diversos tipos de módulos de software, correspondentes às diferentes classes de agentes do sistema.

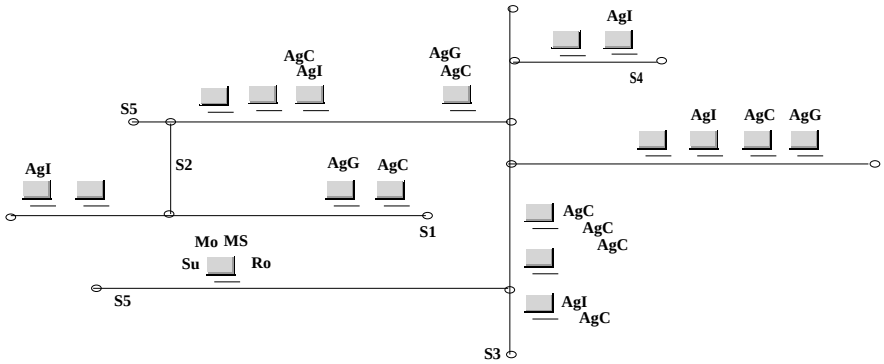


Figura 4.5: Ilustração da Instalação Física do Sistema

O módulo de supervisão (agente supervisor), o módulo de monitorização (agente monitor), o módulo de suporte do ambiente do sistema (agente motor) e o módulo de encaminhamento (agente encaminhador) são módulos únicos, de acordo com a filosofia do próprio sistema. Os restantes módulos (agentes de interface, guardiões e cognitivos) podem replicar-se, dependendo o número de réplicas apenas da capacidade computacional do nodo da rede onde forem instalados.

A localização dos agentes de interface, guardiões e cognitivos é transparente para os utilizadores e administradores dos agentes do sistema.

O sistema é flexível, devido, em grande parte, à não obrigatoriedade de uma localização rígida dos seus diferentes componentes, bem como à não hierarquização da sua forma de utilização.

4.8 Actividades de Administração e Controlo

A filosofia base de gestão do sistema assenta fundamentalmente numa estrutura funcional semelhante à de um ambiente de gestão de bases de dados de quarta geração, onde existe a figura de um administrador geral, a fim de regulamentar e garantir o bom funcionamento e utilização do sistema gestor de bases de dados e da própria base de dados. As suas atribuições vão desde o controlo e administração de acessos até à criação de novas formas de manipulação de dados, bem como à definição de normas de segurança e privacidade de informação, entre outras.

Este modelo de gestão, quando aplicado ao ambiente de um SM, é algo rígido, nomeadamente ao nível do acesso dos agentes à informação dentro do próprio sistema.

As acções de administração, neste tipo de sistemas, têm como objectivo evitar eventuais situações de contenção e de "deadlock", assegurando uma eficiente gestão dos recursos disponíveis, daí que tenha sido criada a figura do administrador de sistema, cujas funções estão estritamente associadas às situações descritas. As suas directivas de gestão e de controlo são traduzidas para o sistema através do agente de supervisão. A operacionalidade do sistema depende directamente da disponibilidade deste agente, que, por sua vez, depende directamente do administrador do sistema. Os agentes de suporte operacional do sistema e o agente de encaminhamento estão também sob a sua supervisão. Os outros tipos de agentes (cognitivos, guardiões e de interface) são da responsabilidade de administradores locais e dos utilizadores do sistema, não dependendo directamente da actuação do administrador do sistema.

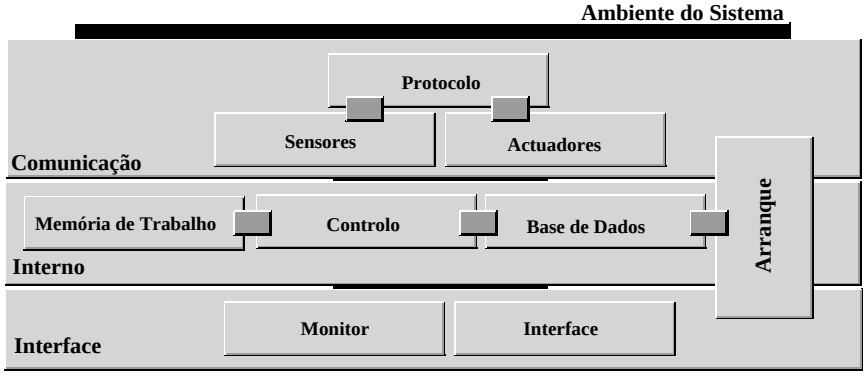


Figura 4.6: Arquitectura Funcional do Agente de Supervisão.

4.8.1 Administração do Sistema

A administração do sistema está alocada ao seu agente de supervisão: o supervisor. As actividades realizadas por este agente, relativas ao controlo e à gestão do sistema, visam directamente o arranque e a consequente disponibilização do ambiente do sistema a todas as classes de agentes. Para suporte às acções de administração, o agente de supervisão possui vários mecanismos que possibilitam ao administrador do sistema, sempre que este o desejar, a recolha de informação sobre os agentes activos no sistema e as actividades que no momento estão a desenvolver. Complementarmente, permite fazer o controlo da actividade do agente de monitorização, activando-o ou desactivando-o de acordo com as instruções do administrador, o mesmo acontecendo relativamente ao agente de serviços de encaminhamento.

Arquitectura e Funcionalidades

A arquitectura do agente de supervisão (Figura 4.6) integra os seguintes módulos:

- **Arranque**.

  Permite a activação e consequente disponibilização do ambiente do sistema às diferentes classes de agentes. É invocado apenas no início da operação do sistema, activando um agente de suporte autónomo cuja função é criar o "Tuple Space" do Linda numa plataforma computacional específica. Este agente está permanentemente activo, por forma a manter a estrutura do ambiente do sistema sempre disponível aos agentes do sistema. O processo inverso - desactivação do ambiente do sistema - é igualmente executado pelo módulo de arranque do agente de supervisão.

- **Comunicação**.

  É responsável pelo controlo e gestão das comunicações entre o agente de supervisão e o ambiente do sistema, sendo constituído pelos seguintes blocos funcionais:

  - **Protocolo**.

    Conjunto de mecanismos que permite a invocação das primitivas do Linda, que estabelecem a ligação entre o agente de supervisão e o ambiente do sistema.

  - **Sensores**.

    Juntamente com os mecanismos de protocolo, detectam e analisam eventuais formas de conhecimento endereçadas ao agente de supervisão.

  - **Actuadores**.

    Permitem colocar no ambiente do sistema, com a ajuda dos mecanismos de protocolo, a informação gerada internamente pelo agente de supervisão.

- **Interno**.

  Implementa os mecanismos necessários à administração e controlo do sistema, sendo constituído pelos seguintes módulos:

  - **Controlo**.

    Controla as acções realizadas localmente pelo agente de supervisão (ex.: activação dos agentes monitor e encaminhador) e analisa a informação que este recolhe do ambiente do sistema (ex.: agentes cognitivos activos, resolução de problemas em curso e utilizadores ligados ao sistema).

  - **Memória de Trabalho**.

    Armazena informação intermédia relacionada com as acções do agente de supervisão.

  - **Base de Dados**.

    Faz a gestão local da informação relacionada com a activação do agente de supervisão e daquela que vai sendo armazenada na memória de trabalho durante a realização das suas acções locais.

- **Interface**.

  Módulo constituído pelos mecanismos e estruturas de conhecimento que garantem a comunicação do agente de supervisão com o administrador do sistema, sendo constituído por dois blocos funcionais:

  - **Monitor Local**.

    Cuja função é relatar ao administrador do sistema as acções que estão a ser realizadas localmente no ambiente do agente de supervisão.

  - **Interface**.

    Que permite ao administrador interactuar directamente com o

agente de supervisão.

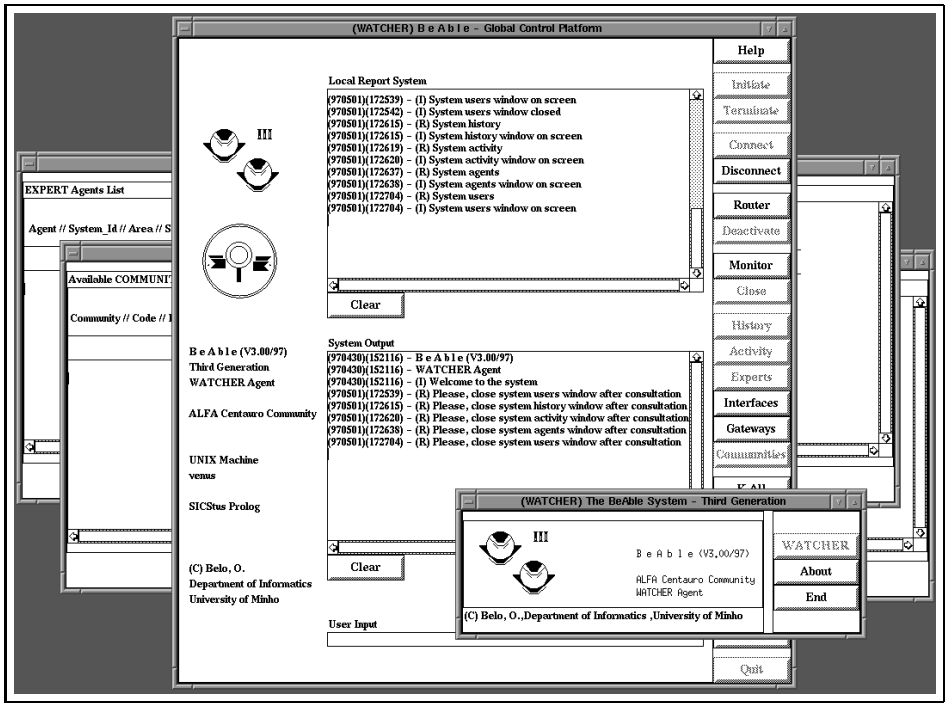


Figura 4.7: Ambiente do Agente de Supervisão.

O Agente de Supervisão

O agente de supervisão (Figura 4.7) disponibiliza ao administrador do sistema um meio de controlo e observação das actividades que se desenrolam no seio deste. Através da figura é possível ter-se uma visão das diferentes funcionalidades do sistema[2]:

[2]As designações dos componentes do agente de supervisão em língua inglesa, visam facilitar a divulgação do presente trabalho na comunidade científica. O mesmo critério será utilizado durante as apresentações dos ambientes dos outros agentes do sistema, nos capítulos seguintes.

- **Monitor** - *"Local Report System"*.
  Onde o agente de supervisão apresenta as acções que vai realizando localmente.

- **Saída de Informação** - *"System Output"*.
  Zona utilizada pelo agente de supervisão para comunicar ao administrador do sistema as suas necessidades em termos de recursos ou as acções a realizar para o bom andamento dos processos em curso.

- **Entrada de Informação** - *"User Input"*.
  Zona utilizada pelo administrador para apresentar ao agente de supervisão a informação solicitada ou qualquer outro tipo de dados relacionados com as acções de administração.

- **Acções de Comando**[3].
  Traduzem para o sistema acções de gestão indicadas pelo administrador do mesmo (Tabela 4.3).

A Tabela 4.3 apresenta as principais funcionalidades disponibilizadas pelo agente de supervisão. Este agente não toma qualquer atitute autonomamente. Todas as suas acções são directamente controladas pelo administrador do sistema, que as selecciona e activa. As acções "Monitor" e "Close" permitem ao administrador do sistema controlar directamente o agente de monitorização global, activando-o e desactivando-o a partir do agente de supervisão. As acções de encaminhamento (Capítulo 6), que permitem o acesso dos agentes cognitivos do sistema a outras comunidades de agentes, são igualmente autorizadas a partir do agente de supervisão, embora asseguradas por um

[3]As acções de comando são executadas através dos "botões" que o agente de supervisão tem disponíveis no lado esquerdo do seu ambiente (Figura 4.7).

Acção	Descrição da Acção
Help	Apresenta o texto de ajuda relativo ao funcionamento do agente de supervisão.
Initiate	Possibilita o arranque ("booting") do sistema.
Terminate	Concretiza a desactivação do sistema.
Connect	Liga o agente de supervisão ao ambiente distribuído do sistema.
Disconnect	Desliga o agente de supervisão do ambiente distribuído do sistema.
Router	Activa o agente de encaminhamento do sistema.
Deactivate	Desactiva o agente de encaminhamento do sistema.
Monitor	Activa o agente de monitorização global.
Close	Desactiva o agente de monitorização global.
History	Apresenta a história dos processos de resolução realizados no sistema.
Activity	Visualiza os processos em curso no sistema.
Agents	Visualiza os nomes dos agentes cognitivos que se encontram activados.
Users	Visualiza os utilizadores activos no sistema.
K.All	Termina a intervenção de todos os agentes do sistema.
K.Agent	Termina a intervenção de um agente cognitivo.
K.User	Termina a intervenção de um agente de interface.
Generics	Solicita a consulta do ficheiro de genéricos (Capítulo 5, Secção 5.4.1) de um agente cognitivo.
Consult	Invoca o ambiente de um agente de interface.
Quit	Abandona o ambiente do agente de supervisão.

Tabela 4.3: Acções de Controlo do Agente de Supervisão.

agente de uma classe especial: o encaminhador. Para a activação e desactivação do agente encaminhador, o supervisor utiliza, respectivamente, as acções "Router" e "Deactivate" do agente de supervisão. Através de uma acção "Consult", o administrador pode invocar o ambiente de um agente de interface, tornando-se, assim, um simples utilizador do sistema (Capítulo 5, Secção 5.2).

```
procedimento Agente-Supervisão;
    inicialização-arranque-agente
    repetir
        ler-comando-utilizador(Comando)
        caso Comando seja
            'initiate'   :   activar-ambiente-sistema(Endereço-Ambiente);
                             activar-funções(['Terminate','Connect', ... ])
            'connect'    :   obter-endereço-ambiente-sistema(Endereço-Ambiente);
                             ligar-agente-supervisão-ambiente(Endereço-Ambiente);
                             activar-funções(['Disconnect','Router','Monitor', ... ])
            'router'     :   activar-router;
                             difundir-activação-router
            'monitor'    :   activar-monitor;
                             difundir-activação-monitor
            'activity'   :   identificar-agentes-processos-activos(Lista-Agentes);
                             recolher-informação-processos(Lista-Agentes,
                                     Informação-Processos,Estado);
                             activar-processo-visualização(Estado);
                             visualizar-processos(Informação-Processos)
            'agents'     :   identificar-agentes-cognitivos-activos(Lista-AgC);
                             recolher-informação-agentes(Lista-AgC,
                                     Informação-Agentes,Estado);
                             activar-processo-visualização(Estado);
                             visualizar-lista(Informação-Agentes)
            'generics'   :   ler-identificação-agente(Agente);
                             requerer-ficheiro-genéricos-agente(Agente) ;
                             aguardar-resposta-agente(Agente,Genéricos,Estado);
                             activar-processo-visualização(Estado);
                             visualizar-genéricos(Agente,Genéricos)
            'consult'    :   activar-agente-interface(administrador)
            (...)        :   (...)

            'quit'       :   activar-garbage-collection;
                             terminar-processos-locais(Fim-Execução)
            outro-caso   :   trata-erro(Comando)
        fim-caso
        informar-administrador-via-monitor;
        actualizar-logfile-agente
    até Fim-Execução
fim-procedimento
```

Figura 4.8: Descrição Genérica do Comportamento do Agente de Supervisão.

```
% \\\
% \\\ Ligação do agente de supervisão ao ambiente do sistema através do
% \\\ predicado 'comando(conectar)'.
comando(conectar) :-
        % \\\ Invocação dos predicados de comunicação com os monitores
        % \\\ global e local.
        ver_mensagem(suco01,Me1),
        monitoring(Me1),
        % \\\ Identificação do endereço lógico do ambiente do sistema
        % \\\ para a comunidade de agentes Co.
        wm(comunidade_co(Co)),
        name(Co,SCo),
        rsh_server(Co),
        rec_server(SCo,Servidor),
        % \\\ Ligação do agente de supervisão ao ambiente do sistema.
        lig_cliente(Servidor),
        % \\\ Activação das funcionalidades do agente de supervisão.
        obj_visual(bea,tec_conectar,Tec_conectar),
        Tec_conectar    => disable,
        obj_visual(bea,tec_lista,Tec_lista),
        Tec_lista       => enable,
        obj_visual(bea,tec_desconectar,Tec_desconectar),
        Tec_desconectar => enable,
        obj_visual(bea,tec_historia,Tec_historia),
        Tec_historia    => enable,
        (...)
        % \\\ Activação das funcionalidades dos agentes encaminhador
        % \\\ e monitor.
        bot_monitor(Tec_monitor,Tec_nomonitor),
        bot_router(Tec_router,Tec_norouter), ...
(...)
% \\\ Obtenção, por via remota, do endereço do ambiente do sistema
% \\\ para a comunidade Co.
rsh_server(Co)
        :-
        name(Co,SCo),
        append("rcp venus:/home/obelo/BeAble/bea_bbcom.end.",SCo,SR1),
        append(SR1," . ",SCommand),
        name(Command,SCommand),
        unix(shell(Command)).
rsh_server(_)
        :-
        fail.
(...)
```

Figura 4.9: Ligação do Agente de Supervisão ao Ambiente do Sistema.

A Figura 4.8 apresenta uma breve descrição de parte do comportamento do agente de supervisão[4]. Nela se podem observar algumas das acções que o agente de supervisão realiza durante as operações de controlo sobre o ambiente do sistema, cuja explicação se pode encontrar na Tabela 4.3.

A Figura 4.9 apresenta uma peça de código Prolog que implementa os mecanismos correspondentes à ligação do agente de supervisão à estrutura que suporta o ambiente do sistema. Com base num endereço lógico, gerado pelo agente "motor" do ambiente do sistema (ver subsecção 4.8.3 deste capítulo) aquando da criação do "Tuple Space" do Linda, a ligação é estabelecida, através de primitivas especiais do Linda, e as funcionalidades gerais do agente de supervisão activadas.

4.8.2 Monitorização do Sistema

As Acções de Monitorização

A necessidade de se efectuar uma boa gestão e controlo do sistema conduz necessariamente a uma análise sistematizada das actividades realizadas pelos agentes integrados no ambiente do mesmo. Nesse sentido, foram implementados em todos os agentes do sistema as estruturas de conhecimento e os procedimentos necessários para que fosse possível, a cada momento, conhecer as actividades por eles desenvolvidas. Assim, sempre que os agentes executam uma dada acção, esta é complementada com uma acção de monitorização,

[4]O algoritmo descreve duas das principais fases de operação do agente de supervisão: 1) arranque e inicialização (*inicialização-arranque-agente/0*) - na qual o agente prepara o ambiente local (criação das janelas de interface e disponibilização das acções de controlo) para o administrador do sistema; 2) gestão do sistema - representada essencialmente através da estrutura algorítmica *repetir ... até*, na qual o agente de supervisão recebe as instruções do administrador relativamente às acções de controlo que este deseje executar.

cujo objectivo é relatar ao supervisor do sistema a tarefa realizada. Estas acções de monitorização podem-se realizar a dois níveis:

- **Globalmente** - sendo consideradas apenas as acções de monitorização efectuadas pelo agente de monitorização;
- **Localmente** - contemplando as acções de monitorização efectuadas ao nível dos monitores locais de cada agente do sistema.

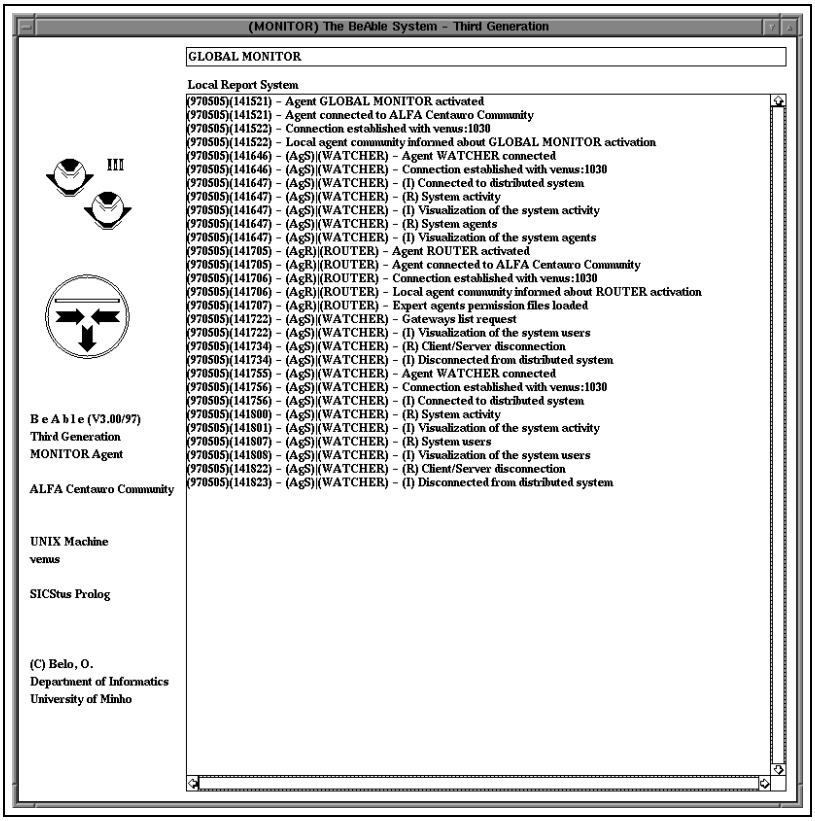


Figura 4.10: Agente de Monitorização do Sistema.

As acções de monitorização são uma ajuda preciosa para as actividades de gestão do supervisor do sistema. O agente de monitorização foi especialmente

desenvolvido para ajudar o administrador do sistema nessas actividades. É um agente autónomo, cujas acções se podem realizar em simultâneo com as do agente de supervisão, apesar de este último ser o responsável pela sua activação e desactivação. A função do agente de monitorização é reportar ao administrador do sistema as actividades realizadas pelos agentes activos no sistema, incluindo as do próprio agente de supervisão.

O Agente de Monitorização do Sistema

Após ter sido activado, o agente de monitorização corre em simultâneo com os outros componentes do sistema, actuando como uma espécie de "demónio" ao analisar o sistema em busca de eventuais mensagens que contenham informação de monitorização a ele endereçada. Tudo o que consegue recolher é de imediato apresentado ao supervisor do sistema (Figura 4.10).

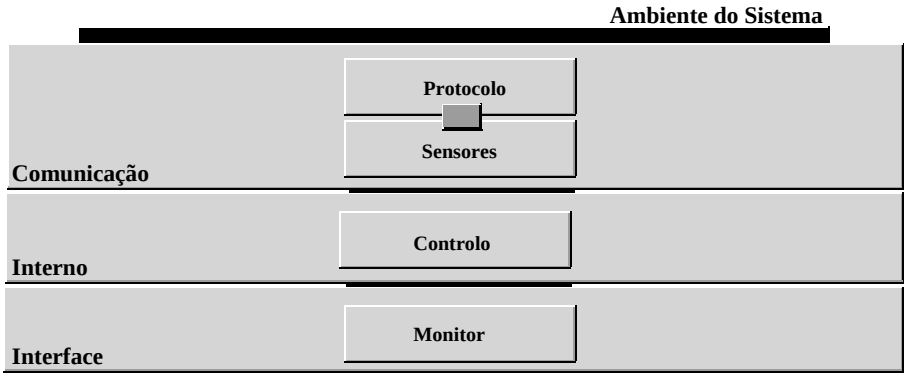


Figura 4.11: Arquitectura Funcional do Agente de Monitorização Global.

O comportamento do agente de monitorização não interfere directamente com o desempenho dos outros agentes do sistema, nem estes interferem directamente com o comportamento daquele. A única excepção a esta regra

verifica-se relativamente ao agente de supervisão, que, tal como já foi referido anteriormente, é o responsável pelo seu controlo directo.

A arquitectura funcional do agente de supervisão integra os seguintes módulos (Figura 4.11):

- **Comunicação**.
  Módulo responsável pelo controlo e gestão dos processos de comunicação entre o agente de monitorização e o ambiente do sistema, sendo constituído por dois blocos funcionais:
  - **Protocolo**.
    Conjunto de mecanismos que invocam primitivas do Linda responsáveis pelo estabelecimento dos canais de comunicação entre o agente de monitorização e o ambiente do sistema.
  - **Sensores**.
    Conjunto de mecanismos que, juntamente com os de protocolo, analisam o conhecimento presente no sistema, com o objectivo de detectarem eventuais estruturas de conhecimento endereçadas ao agente de monitorização.
- **Interno**.
  Módulo que implementa os mecanismos necessários ao controlo do agente de monitorização.
- **Interface**.
  Módulo constituído unicamente pela estrutura de monitorização, que suporta a comunicação das actividades realizadas pelos agentes do sistema ao administrador do mesmo.

```
procedimento Agente-Monitorização;
    inicialização-arranque-agente(Parâmetros-Operacionais,Condições-Percepção);
    difusão-parâmetros(Parâmetros-Operacionais)
    repetir
        repetir
            analisar-ambiente(Condições-Percepção,Percepção)
        até Percepção
        analisar-percepção(Percepção,Reportar)
        se Reportar então
            visualizar-percepção-monitor-local(Percepção)
        fim-se
    até Fim-Execução
fim-procedimento
```

Figura 4.12: Descrição Genérica do Comportamento do Agente de Monitorização.

Na Figura 4.12 apresenta-se uma descrição parcial do comportamento do agente de monitorização[5]. A Figura 4.13 apresenta uma peça de código Prolog que implementa parte dos mecanismos de comunicação do agente de monitorização, nomeadamente os relacionados com a recolha de mensagens contendo informação de monitorização, ou os respeitantes à troca de informação de controlo entre o agente de supervisão e o de monitorização.

[5]O algoritmo descreve o comportamento genérico do agente de monitorização. Nele podemos identificar três fases: 1) arranque do agente (*inicialização-arranque-agente/2*) - destinada a preparar o ambiente local do agente de monitorização para a sua execução normal, com base nos parâmetros recolhidos (*Parâmetros-Operacionais*), e a identificar as condições (*Condições-Percepção*) sob as quais deverá actuar sobre o ambiente do sistema; 2) análise de mensagens - realizada sobre o ambiente, onde o agente de monitorização procura (*analisar-percepção/2*) continuamente algum tipo de mensagem a ele endereçada, obedecendo às condições de percepção definidas; 3) acção - caso encontre uma mensagem nessas condições, o agente recolhe-a do ambiente do sistema, visualizando-a de seguida no monitor global (*visualizar-percepção-monitor-local/1*), para informar do sucedido o administrador do sistema.

```
% \\\
% \\\ Recolha de uma mensagem do ambiente do sistema enviada por um
% \\\ dos seus agentes e sua visualização no monitor local.
mo01    :::     mensagem        ::
                [existe_obj_bb(bb(t_mo_message(Me)))]
        -->
                [remove_obj_bb(bb(t_mo_message(Me))),
                mon_local(Me)].

% \\\ Recolha de uma mensagem do agente de supervisão indicando ao de
% \\\ monitorização que deve terminar a sua execução.
mo02    :::     fim     ::
                [existe_obj_bb(bb(sup,t_su_mo_kill))]
        -->
                [remove_obj_bb(bb(sup,t_su_mo_kill)),
                stop].
(...)

% \\\ Visualização da mensagem recolhida pelo agente de
% \\\ monitorização
mon_local(Me)
        :-
        ind_data(Dt,Ho),
        user:obj_visual(mon,tex_monitor,Objecto),
        Objecto => insert(format("(~s)(~s) - ~s ~n",[Dt,Ho,Me])).
mon_local(_).
(...)
```

Figura 4.13: Peça de Código em Prolog do Sistema de Comunicação do Agente de Monitorização.

4.8.3 O Agente "Motor" do Ambiente do Sistema

O agente "motor" do ambiente do sistema é o segundo da classe de agentes de suporte operacional. Assume um papel vital na operacionalidade do sistema, já que é o responsável pela geração e manutenção da estrutura distribuída que o suporta. Após a activação do agente de supervisão é necessário que o ambiente do sistema seja disponibilizado às outras classes de agentes através da activação do agente "motor" pelo administrador do sistema, a partir do

agente de supervisão.

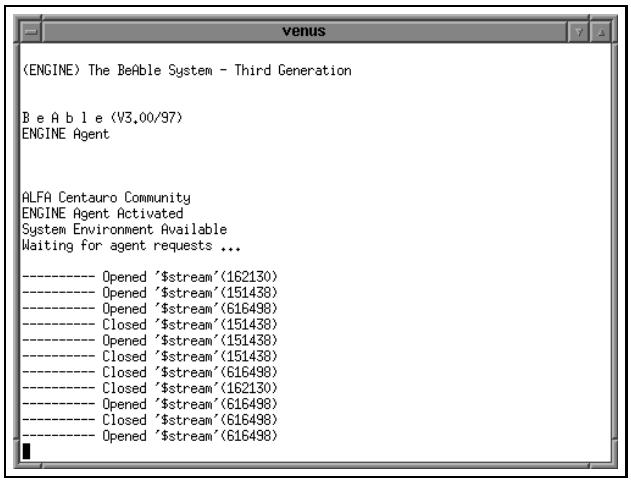


Figura 4.14: O Agente "Motor" do Sistema.

As acções "Initiate" e "Terminate" (Tabela 4.3), despoletadas a partir do agente de supervisão, activam e desactivam, respectivamente, o agente "motor" (Figura 4.14) e, consequentemente, criam ou eliminam uma réplica do sistema. Após a fase de arranque - activação dos agentes de supervisão e "motor" - o sistema fica pronto para utilização.

A Figura 4.15 apresenta uma peça de código Prolog que implementa os mecanismos correspondentes à activação do "Tuple Space" do Linda - a estrutura que suporta o ambiente do sistema -, bem como ao armazenamento do seu endereço lógico num ficheiro de dados próprio. Através deste endereço, gerado no momento da criação do ambiente, os agentes conseguem identificá-lo e interligar-se a partir da plataforma computacional em que se encontram instalados.

```
% \\\
% \\\ Definição do endereço lógico do ambiente do sistema, activação
% \\\ do processo do servidor do ambiente (``Tuple Space'') do sistema.
beable
        :-
        % \\\ Recolha do identificador da comunidade de agentes
        % \\\ a criar via sistema operativo.
        par_unix(Argv),
        name(Argv,Co),
        % \\\ Carregamento dos parâmetros relativos à comunidade (Co)
        % \\\ indicada.
        load_community_par(Co),
        ap_nome(Ap_Nome),
        ap_versao(Ap_Versao),
        ap_modulo(Ap_Modulo),
        format("~2n~s ~n",[Ap_Nome]),
        format("~2n~s ~n",[Ap_Versao]),
        format("~s ~2n",[Ap_Modulo]),
        comunidade_id(NCo),
        format("~2n~s~n",[NCo]),
        format("ENGINE Agent Activated~n",[]),
        format("System Environment Available~n",[]),
        format("Waiting for agent requests ...~2n",[]),
        % \\\ Activação do ambiente da comunidade com a invocação de
        % \\\ um processo servidor do Linda.
        linda(E-(user:ind_server(E,Co))).
(...)
% \\\ Armazenamento em ficheiro do endereço lógico do ambiente do
% \\\ sistema para a comunidade de agentes Co.
ind_server(E,Co)
        :-
        comunidade_ts(_,Na,TDi),
        append(TDi,".",TDi2), append(TDi2,Co,TDif),
        atom_chars(F,TDif), atom_chars(C,Co),
        open(F,write,S),
        format(S,'bea_bbts(~q,~q).~n',[C,E]),
        close(S),
        append("rcp ",TDif,E1),
        append(E1," ",E2), append(E2,Na,E3),
        append(E3,":",E4), append(E4,TDif,Ef),
        atom_chars(Ea,Ef),
        plsys(shell(Ea)).

(...)
```

Figura 4.15: Activação do Ambiente do Sistema e Armazenamento do seu Endereço Lógico.

Capítulo 5

A Resolução de Problemas

Apresentação, descrição e caracterização das classes de agentes directamente envolvidas nos processos de resolução de problemas: agentes de interface, agentes guardiões e agentes cognitivos. Modelos de coordenação, cooperação e aprendizagem dos agentes cognitivos.

Diferentes formas de raciocínio podem ser observadas nos seres humanos quando estes se encontram perante um problema. A definição e exploração de cenários e a procura de possíveis pistas que os possam ajudar na sua resolução constituem estratégia muito comum. Esse comportamento pode fornecer os elementos básicos para o desenvolvimento efectivo de um plano que defina a forma mais eficaz de atacar o problema e, consequentemente, de o resolver.

A reacção a estímulos gerados por modificações verificadas nos ambientes onde estão integrados é outra das formas de atitude típicas dos seres humanos. Tal comportamento permite-lhes escolher, para um dado acontecimento ou situação, a melhor forma de reacção e aplicar as mais efectivas estratégias

para a resolução de problemas.

A dificuldade em estabelecer um modelo comparativo a fim de avaliar apropriadamente estas duas formas de abordagem é significativa. Uma eventual decisão por uma ou pela outra poderá ser induzida pelo enquadramento em que se desenvolve o problema, o que vem confirmar que a opção por dada estratégia é, muitas vezes, definida de acordo com as especificidades do problema em questão. É através de tipos de comportamento como este que os humanos actuam, consciente ou inconscientemente, quando confrontados com o imperativo da resolução de problemas.

Em certa medida, essas duas formas do comportamento humano podem ser emuladas por entidades artificiais: os agentes cognitivos. Estas entidades têm a capacidade de planear as suas acções, desenvolver estratégias de resolução, comunicar ou mesmo definir formas de partilha dos seus próprios recursos, enquanto num sistema reactivo têm a possibilidade de seguir os acontecimentos à medida que eles vão sucedendo (B.Hayes-Roth 1992).

Por forma a constituirem uma "força de trabalho" efectiva, os agentes devem ser capazes de desenvolver entre si formas de cooperação. A partilha de resultados e tarefas são duas das formas mais comuns de cooperação entre agentes (Capítulo 3, Secção 3.6).

O comportamento oportunístico dos agentes cognitivos do sistema permite que os seus processos de cooperação sejam essencialmente desenvolvidos em torno da partilha de resultados e da eventual troca de informação entre si. Complementarmente, esse carácter oportunístico confere-lhes um maior dinamismo e uma maior flexibilidade quando chamados a intervir, permitindo-lhes ainda preservar a sua autonomia e independência. São tipicamente SBC

cuja perícia e conhecimento estão definidos num domínio de aplicação específico, bastante especializado. O conjunto de todos os agentes cognitivos do sistema constitui a comunidade de agentes especialistas.

A estrutura distribuída que apresenta o ambiente do sistema BEABLE possibilita a distribuição da sua comunidade de agentes por diferentes localizações, porventura geograficamente distintas. Contudo, as suas actividades estarão sempre condicionadas por estímulos que estes possam eventualmente detectar no seio do sistema (fluxos de informação ou alterações do estado de conhecimento).

O desenho e o desenvolvimento do sistema giraram em torno de uma ideia base: a criação de um ambiente que disponibilizasse aos seus utilizadores as estruturas de conhecimento e os mecanismos de raciocínio "encarnados" numa comunidade de agentes necessária para a resolução de problemas. Os utilizadores terão necessidade de permissão para aceder a essa comunidade e apresentar-lhe os seus problemas. Essa apresentação é a etapa inicial de um processo de consulta que irá gerar o primeiro conjunto de estímulos, o qual será depositado no ambiente do sistema e analisado pelos agentes cognitivos, podendo ou não despoletar a sua atenção. Estas plataformas de utilização do sistema deram origem a uma classe específica de agentes: os agentes de interface.

5.1 Intervenientes num Processo de Resolução

Após o arranque do sistema, a interligação dos agentes dependerá dos administradores locais e das plataformas computacionais onde estes estão situados. A activação de um agente é realizada a nível local, independentemente

do agente de supervisão ou do administrador do sistema. Os agentes não poderão ser activados se os processos de supervisão e controlo ainda não tiverem sido desencadeados, ou se o ambiente distribuído do sistema não tiver ainda sido disponibilizado a toda a comunidade de agentes. A forma como os agentes estão ligados ao sistema, bem como a sua localização física, são completamente transparentes para os utilizadores. Os agentes só são disponibilizados para a resolução de um problema a partir do momento em que ficarem ligados ao ambiente do sistema e tiverem acesso às estruturas de conhecimento que nele permanecem ou circulam.

Enquanto activos no sistema, os agentes estão em constante comunicação com o seu ambiente, por forma a detectar as estruturas de conhecimento que, de acordo com o conhecimento que possuem, forem consideradas relevantes para o funcionamento dos seus processos internos. Os agentes podem "entrar e sair" do sistema a qualquer momento (entenda-se, quando o respectivo administrador ou utilizador o desejar) sem que a operacionalidade global do sistema seja afectada. Isto significa que, no caso de um agente abandonar a estrutura do sistema, os outros processos em curso, a cargo de outros agentes, não serão afectados.

No processo de resolução de problemas (Figura 5.1), duas classes de agentes do sistema assumem particular relevância: os agentes de interface e os agentes cognitivos. Os primeiros facultam aos utilizadores um meio privilegiado para comunicarem com a comunidade de agentes cognitivos, a fim de lhe colocarem os seus problemas. Alternativamente, os agentes de interface podem ser substituídos por agentes da classe dos guardiões, quando a consulta ao sistema é desencadeada a partir de um sistema de "browsing" da Internet. De qualquer forma, quer se trate de um agente da classe dos guardiões quer

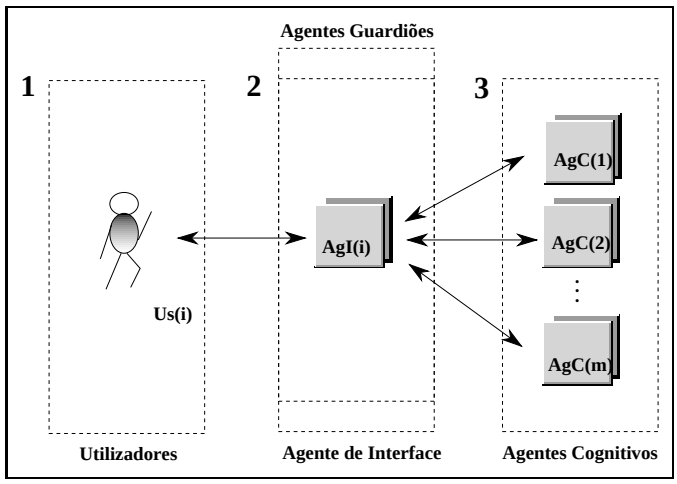


Figura 5.1: Intervenientes num Processo de Resolução de Problemas.

de um agente de interface, os procedimentos a seguir são idênticos.

5.2 Os Agentes de Interface

Os processos de análise e resolução de problemas são desencadeados a partir da plataforma computacional dos agentes de interface[1]. Quando necessitam de consultar um ou mais especialistas, os utilizadores usam essa plataforma. Esta classe de agentes permite aos utilizadores interactuar com o sistema global, disponibilizando os meios necessários para a consulta de um agente cognitivo em particular ou mesmo para a intervenção da comunidade de agentes cognitivos como um todo, para a análise de um problema.

A plataforma computacional de um agente de interface pode ser invocada a

[1]Assim acontece em problemas típicos de diagnóstico e classificação. No entanto, em algumas aplicações relacionadas com problemas de controlo e monitorização, não havendo a necessidade de os utilizadores intervirem directamente nos processos de resolução, estes são desencadeados e controlados pelos próprios agentes cognitivos do sistema (Capítulo 7, Secções 7.2.2 e 7.2.3).

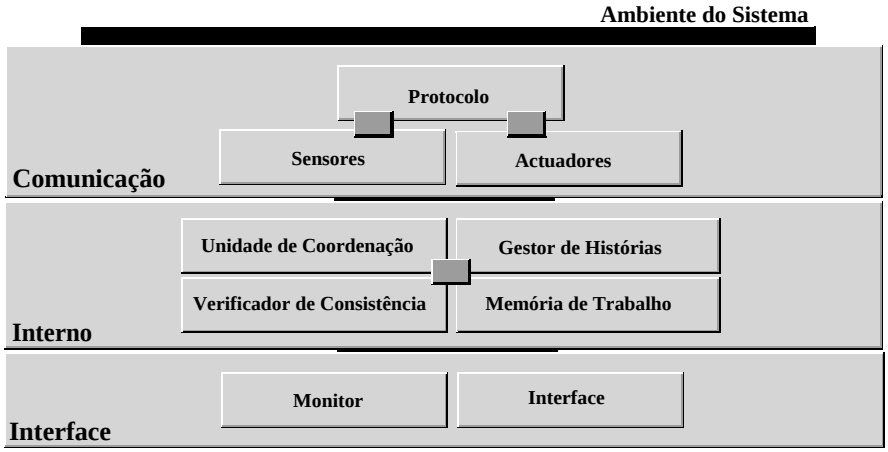


Figura 5.2: Arquitectura Funcional dos Agentes de Interface.

partir de qualquer nodo da rede na qual o sistema está instalado. O processo de invocação deste agente é completamente independente de outros processos que possam estar em execução no sistema.

5.2.1 Arquitectura Funcional

A arquitectura funcional de um agente de interface (Figura 5.2) define-se pelos seguintes módulos:

- **Comunicação**.
  Módulo que controla e faz a gestão dos processos de comunicação entre o agente de interface e o ambiente do sistema, subdividindo-se nos seguintes blocos funcionais:
  - **Protocolo**.
    Permite estabelecer a ligação entre o agente e o ambiente do sistema, através de um conjunto de mecanismos desenvolvidos sobre

os comandos do Linda.

- **Sensores**.

  Conjunto de mecanismos que analisa o conhecimento que circula no sistema, com a ajuda dos mecanismos de protocolo, a fim de detectar possíveis estruturas de conhecimento ou de dados relevantes para os processos internos do agente.

- **Actuadores**.

  Conjunto de mecanismos que transfere para o ambiente do sistema, em parceria com os mecanismos de protocolo, a informação disponibilizada pelo agente de interface.

- **Interno**.

  Módulo responsável pelas acções de controlo e coordenação do agente, sendo constituído pelos seguintes blocos funcionais:

  - **Unidade de Coordenação**.

    Permite ao agente de interface analisar a informação que recolheu do ambiente do sistema e validá-la de acordo com o conhecimento que possui acerca do problema, seleccionando aquela que considerar relevante para os processos internos do agente.

  - **Memória de Trabalho**.

    Armazena a informação relacionada com o processo de resolução de problemas em curso (conhecimento e dados intermédios e/ou finais).

  - **Gestor de Histórias**.

    Faz a gestão local da informação que vai sendo armazenada na memória de trabalho durante os processos de consulta dos utilizadores.

  - **Verificador de Consistência**.

    Permite ao agente de interface uma actuação em conformidade com as indicações enviadas pelos sistemas de verificação de consistência dos agentes cognitivos, possibilitando-lhe manter a consistência da informação da sua memória de trabalho.

- **Interface**.

  Módulo responsável pelos processos de comunicação entre o agente de interface e o utilizador, sendo constituído pelos seguintes blocos funcionais:

  - **Monitor**.

    Permite relatar ao utilizador do agente de interface as acções que estão a ser realizadas por este.

  - **Interface**.

    Permite ao utilizador apresentar as suas contribuições para os processos de resolução de problemas em curso ou responder a eventuais solicitações de informação por parte do agente de interface.

Nas Figura 5.3 é apresentado um excerto de uma peça de código em Prolog do sistema de comunicação de um agente de interface, visando o processamento de questões enviadas pelos agentes cognitivos e a recolha de uma solução proposta por um agente cognitivo.

5.2.2 Utilização e Dinâmica dos Agentes de Interface

Os processos de consulta dos agentes cognitivos pelos utilizadores são realizados através da plataforma computacional dos agentes de interface (Figura

```
% \\\
% \\\ Recolha das mensagens enviadas pelos agentes cognitivos com pedidos
% \\\ de informação relativos ao problema exposto pelo utilizador.
us02    :::     processing_requests ::
                [
                queue_opened, wm(infu(Us)),
                % /// Recolha dos pedidos de informação (questões)
                % /// enviados pelos agentes cognitivos.
                pro_questoes(Us,Qu,[ok]
                ]
        -->
                [
                monitoring(d,Us,usqu01),
                Qu = t_ag_question(Ag,Us,P,NP,LQF,LQR,Ci,Da,Ho),
                % /// Análise e tratamento dos pedidos recolhidos.
                inf_resume_q(Ag,Us,P,NP,LQF,LQR,Ci,Da,Ho),
                ins_question(Ag,Us,P,NP,LQF,LQR,Ci,Da,Ho),
                ger_questao(Ag,Us,P,NP,LQF,LQR,Ci,Da,Ho),
                % /// Recolha e envio das respostas dos utilizadores
                % /// aos agentes cognitivos.
                ace_resposta(Ag,Us,P,NP,LQR)
                ].
% \\\ Recolha das mensagens enviadas pelos agentes cognitivos com
% \\\ propostas de solução para o problema exposto pelo utilizador.
us03    :::     solution        ::
                [
                wm(infu(Us)),
                % /// Identificação da mensagem com uma solução enviada
                % /// por um agente cognitivo.
                existe_obj_bb(bb(Ag,Us,t_ag_solution(Ag,Us,S,VS,NS)))
                ]
        -->
                [
                monitoring(d,Ag,usso01),
                remove_obj_bb(bb(Ag,Us,t_ag_solution(Ag,Us,S,VS,NS))),
                % /// Apresentação da solução ao utilizador.
                apr_solucao(Ag,Us,S,VS,NS),
                sol_accao(Ag,Us,S,NS,VS)
                ].
```

Figura 5.3: Recepção de Pedidos de Informação e Soluções enviadas pelos Agentes Cognitivos.

5.4). É a partir desta plataforma que os utilizadores do sistema estabelecem "diálogos" com os agentes cognitivos, podendo optar por dois tipos de consulta:

- **Individual** - em que participa apenas um agente cognitivo, tendo qualquer problema proposto como universo de resolução a base de conhecimento do agente.
- **Colectiva** - susceptível de envolver vários agentes cognitivos, que poderão cooperar para a resolução do problema proposto.

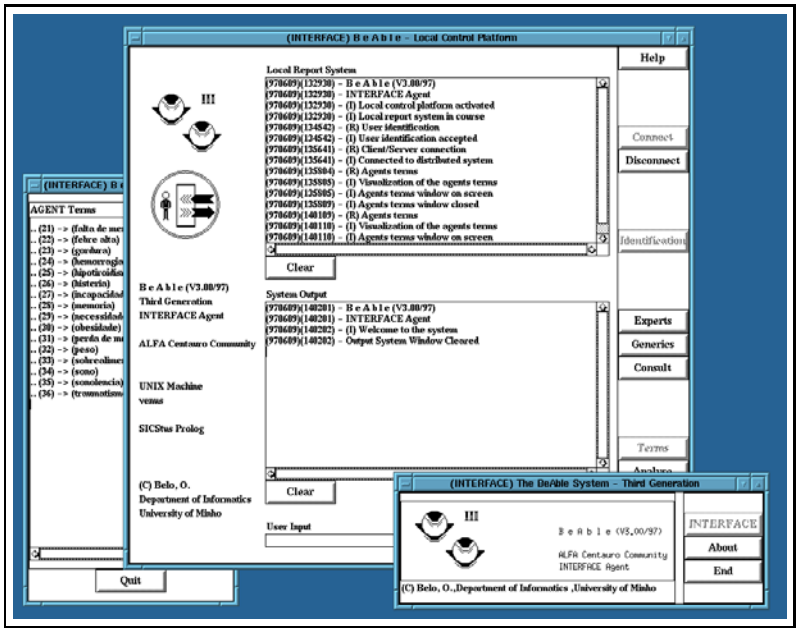


Figura 5.4: Ambiente Principal de um Agente de Interface.

O ambiente de trabalho - ou plataforma computacional - de um agente de interface pode ser invocado a partir de qualquer nodo da rede na qual o sistema esteja instalado e onde exista localmente um agente de interface. O

processo que suporta a execução de um agente de interface é independente dos outros processos que possam estar a ser executados no sistema, sendo a sua execução feita em paralelo.

As primitivas de controlo de um agente de interface (Figura 5.4) encontram-se dispersas por quatro zonas funcionais:

- **Monitor** - *"Report System"*.
  Permite ao agente de interface reportar as actividades que executa localmente.
- **Saída de Informação** - *"System Output"*.
  É utilizada pelo agente de interface para a comunicação com o utilizador.
- **Entrada de Informação** - *"User Input"*.
  Permite ao utilizador do agente de interface apresentar a informação que deseja comunicar ao sistema.
- **Acções de Comando**.
  Disponibiliza ao utilizador os meios para comunicar ao agente quais as acções a realizar (Tabela 5.1).

Qualquer utilizador que deseje consultar o sistema necessita de se identificar perante o agente de interface. Só após essa identificação terá acesso às funcionalidades do ambiente de controlo do agente de interface, caso contrário, este veda-lhe o acesso ao sistema. A identificação do utilizador servirá para personalizar a utilização do sistema por parte daquele.

Acção	Descrição da Acção
Help	Fornece o texto de ajuda do sistema para a utilização do ambiente de controlo do agente de interface.
Connect	Liga a plataforma local do agente de interface ao ambiente do sistema.
Disconnect	Desliga a plataforma local do agente de interface do ambiente do sistema.
User	Recolhe a identificação do utilizador para posterior personalização de processos.
Agents	Apresenta uma lista dos agentes cognitivos activos no sistema.
Generics	Solicita a consulta do ficheiro de genéricos de um agente cognitivo (Capítulo 5, Secção 5.4.1).
Consult	Solicita um processo de consulta a um agente cognitivo.
Terms	Consulta os termos que os agentes cognitivos reconhecem e interpretam como estímulos.
Analyse	Submete uma lista de estímulos para ser analisada pelos agentes cognitivos do sistema. Processo de consulta colectivo.
Quit	Permite a saída do ambiente de controlo do agente de interface.

Tabela 5.1: Acções de Comando do Ambiente de Controlo de um Agente de Interface.

Para a realização de um processo de consulta, o utilizador deve indicar o tipo de consulta pretendida - "Consult" para uma consulta individual ou "Analyse" para uma consulta colectiva. Qualquer uma delas "transportará" o utilizador para o ambiente de consulta do agente de interface.

Este ambiente (Figura 5.5) encontra-se dividido em duas zonas funcionais:

- **Saída de Informação** - *"Inference System Output"*.

  É utilizada pelo agente de interface para formular questões ao utilizador, dar respostas, apresentar soluções ou disponibilizar qualquer outra informação relacionada com o processo de resolução de um problema.

- **Acções de Comando**.

  Zona que permite ao utilizador indicar ao agente quais as acções que

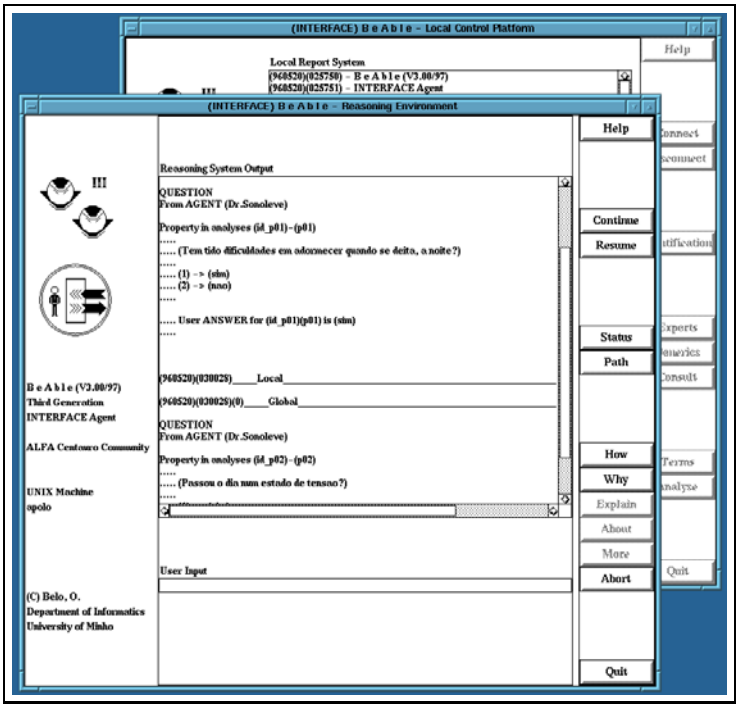


Figura 5.5: Ambiente de Consulta de um Agente de Interface.

deseja que este efectue (Tabela 5.2).

Os actuais processos de consulta realizam-se através de "diálogos" entre o utilizador e o agente de interface, terminando de uma das seguintes formas:

- o agente de interface comunica uma solução para o problema;
- o utilizador decide abandonar o processo;
- o agente de interface decide pôr-lhe termo.

Os agentes de interface actuam sempre como elos de ligação entre os utilizadores e os agentes cognitivos, independentemente da modalidade de consulta seleccionada.

```
procedimento Processamento-Questões;
    recolhe-enqueue-questões-agentes-cognitivos-ambiente(Lista-Questões);
    para cada (questão(Agente-Cognitivo,Mensagem)) ∈ queue-questões
    até Lista-Questões = [ ] fazer
        dequeue-questões(questão(Agente-Cognitivo,Mensagem));
        verificar-questão-respondida(questão(Agente-Cognitivo,Mensagem),
                Respondida)
        se Respondida então
            remover-questões(questão(Agente-Cognitivo,Mensagem))
        senão
            apresentar-questão-utilizador(Agente-Cognitivo,Mensagem)
            aguardar-validar-resposta-utilizador(Resposta,Parâmetros)
            caso Resposta seja
                'continue'      :
                        actualizar-lista-LAA-ambiente(Resposta,Parâmetros);
                        actualizar-lista-LAA-memória-trabalho(Resposta,Parâmetros)
                'why'           :
                        enviar-mensagem(bb(Utilizador,Agente-Cognitivo,
                                t_iag_why(Utilizador,Agente-Cognitivo,Mensagem)));
                        esperar-mensagem-agente-cognitivo(Utilizador,
                                Agente-Cognitivo,Mensagem-Retorno)
                'how'           :
                        enviar-mensagem(bb(Utilizador,Agente-Cognitivo,
                                t_iag_how(Utilizador,Agente-Cognitivo,Mensagem)));
                        esperar-mensagem-agente-cognitivo(Utilizador,
                                Agente-Cognitivo,Mensagem-Retorno)
                            (...)
                'quit'          :
                        recolher-lista-agentes-cognitivos-envolvidos(Lista-Agentes)
                        para cada Agente-Cognitivo ∈ Lista-Agentes
                            até Lista-Agentes = [ ] fazer
                                enviar-mensagem(bb(Utilizador,Agente-Cognitivo,
                                        t_iag_end(Utilizador,Agente-Cognitivo)))
                        fim-para
                        terminar-processo-inferência(Utilizador)
            fim-caso
        fim-se
    fim-para
fim-procedimento
```

Figura 5.6: Descrição Genérica do Comportamento de um Agente de Interface.

Acção	Descrição da Acção
Help	Fornece o texto de ajuda do sistema para a utilização do ambiente de consulta do agente de interface.
Continue	Indica ao agente de interface que pode continuar o processo.
Resume	Repete a última questão apresentada pelo agente de interface.
Status	Visualiza o estado actual do conhecimento do problema.
How	Solicita ao agente de interface que apresente a cadeia de inferência seguida até ao momento na resolução do problema.
Why	Pede ao sistema uma justificação acerca da informação que o agente de interface está a solicitar.
Explain	Pede ao agente de interface que explique a solução apresentada.
About	Solicita ao agente de interface informação relativa aos passos a seguir após a apresentação de uma solução.
More	Pede ao agente de interface que procure uma nova solução.
Abort	Interrompe o processo em curso, indicando ao agente de interface que recomece o processo de análise do problema.
Quit	Termina o processo de consulta do utilizador.

Tabela 5.2: Acções do Ambiente de Consulta de um Agente de Interface.

A Figura 5.6 apresenta uma descrição de algumas das acções passíveis de serem executadas por um agente de interface durante o processo de resolução de um problema[2].

[2]O algoritmo descreve as principais acções desenvolvidas por um agente de interface durante o processo de recepção, análise e tratamento de questões enviadas pelos agentes cognitivos. Este processo divide-se em duas fases: 1) recolha de questões - na qual o agente de interface retira do ambiente do sistema (*recolhe-enqueue-questões-agentes-cognitivos-ambiente/1*) as questões (*Lista-Questões*) enviadas pelos agentes cognitivos, colocando-as na *queue* de tratamento local, ordenadas de acordo com a sua recepção; 2) tratamento das questões - na qual o agente procede à análise e processamento de cada questão recolhida; o agente de interface vai verificar (*verificar-questão-respondida/2*) se cada uma das questões armazenadas na *queue* já foi ou não respondida em estados anteriores do problema; se isso tiver acontecido, abandona a questão (*remover-questões/1*), passando a atender a próxima da *queue*; no caso inverso, apresenta-a ao utilizador (*apresentar-questão-utilizador/2*) e aguarda que este lhe indique (*aguardar-validar-resposta-utilizador/2*) qual a acção a realizar ('value','why','how', etc.). O processo repete-se até ao momento em que o agente de interface não tenha mais questões na *queue*, receba uma mensagem com a indicação de fim de execução ou ocorra uma situação de "time-out". As respostas dos utilizadores são comunicadas aos agentes cognitivos através de uma estrutura de dados (*Lista de Activação dos Agentes* - LAA) mantida no ambiente do sistema durante os processos de consulta dos utilizadores.

5.3 Os Agentes da Classe dos Guardiões

5.3.1 Integração de Tecnologia da Web

A definição de plataformas de interface está normalmente associada ao desenvolvimento de meios de interligação entre um sistema em particular e os seus utilizadores. Este tipo de programas é normalmente característico do próprio sistema onde está integrado, diferindo as suas estruturas de representação de conhecimento e formas de interacção de sistema para sistema. Tal especificidade é normalmente induzida pelos requisitos e domínios de aplicação do sistema. Os agentes de interface referidos no Capítulo 5, Secção 5.2 enquadram-se nesta categoria.

A aplicação conjunta das tecnologias dos SM e da Internet permite disponibilizar os meios necessários para o desenvolvimento de interfaces amigáveis com a capacidade de suportar processos de interacção entre os convencionais "browsers", característicos dos ambientes disponibilizados pela Web ("World Wide Web"), e os SM. Este tipo de tecnologia viabiliza a utilização remota dos SM, através de uma redução nos custos de desenvolvimento e manutenção de interfaces para tais aplicações.

A tecnologia actual da Web (T.Berners-Lee, R.Cailliau, J.Groff & B.Pollerman 1992) disponibiliza ferramentas poderosas e flexíveis, de utilização universal. O desenvolvimento de interfaces de alto nível, a possibilidade de utilizar os recursos da Web e de "navegar" numa enorme rede de computadores acedendo aos seus recursos, são apenas algumas das vantagens de que se pode usufruir através de uma simples ligação à Internet.

A Internet (H.Hahn & R.Stout 1994) é um real fenómeno de popularidade.

Descoberta a um ritmo vertiginoso por diferentes indivíduos e organizações, providencia um variadíssimo leque de serviços e ferramentas, acusando um crescimento impressionante em termos de acessos a cada momento que passa. As últimas versões da linguagem HTML ("HiperText Markup Language"), suportadas pelos mais populares "browsers" da Web, têm revelado grande aplicabilidade no desenvolvimento de plataformas de interface de alto nível, bastante poderosas e amigáveis, para acesso a recursos da Internet (A.Costa, E.Rodrigues, F.Pinto, J.Macedo & M.Nicolau 1995).

A utilização de um interface baseado no ambiente de um "browser" disponibiliza um meio de interface quase universal (pelo menos ao nível actual da tecnologia da Web), bastante simples de utilizar, com custos de desenvolvimento reduzidos (ao nível das páginas em HTML) e concebido de origem com o objectivo de possibilitar o acesso a recursos computacionais remotos. Assim, por que não utilizar tais capacidades para desenvolver interfaces de alto nível para acesso a SM remotos?

Algumas experiências realizadas no âmbito do projecto do sistema XPertWeb[3] (O.Belo & A.Ribeiro 1995) demonstraram a viabilidade da aplicação da tecnologia da Web na implementação de interfaces de alto nível para acesso a SBC remotos. Muitas das ideias exploradas e implementadas no projecto XPertWeb foram adoptadas no actual modelo dos agentes da classe dos guardiões e expandidas no sentido de permitirem uma interligação, não com diferentes SBC distribuídos geograficamente, mas com um ambiente que permite a integração de vários desses sistemas e lhes disponibiliza os meios e as estruturas para que possam cooperar entre si.

[3] O sistema XPertWeb permite criar um meio através da Web para aceder a SP distribuídos pela Internet.

5.3.2 Características dos Agentes Guardiões

Com base no exposto, foi objecto de estudo um outro tipo de agente de interface do sistema orientado para suportar os processos de interacção entre utilizadores da Web e o sistema BEABLE. Esta nova classe de agentes do sistema, a que se atribui a designação de agentes guardiões (O.Belo & A.Ribeiro 1996), estabelece um novo tipo de interface entre o sistema e os seus utilizadores, através de um "browser" convencional da Internet. Desta forma, os utilizadores da Web têm a possibilidade de aceder a uma comunidade de agentes cooperativos baseados em conhecimento: os agentes cognitivos do sistema BEABLE (Capítulo 5, Secção 5.4).

Contudo, a abertura do ambiente do sistema a entidades externas, a largos grupos de utilizadores, requer a implementação de medidas de contenção, de mecanismos que possam garantir a privacidade e a segurança da informação do sistema.

Na estrutura dos agentes da classe dos guardiões foram integrados mecanismos de autenticação e verificação de permissões que restringem o acesso ao ambiente do sistema sempre que tal necessidade seja detectada. Além destas capacidades relativas a segurança e tipos de acesso, os agentes da classe dos guardiões disponibilizam um lote de serviços semelhante ao de um normal agente de interface do sistema (Capítulo 5, Secção 5.2).

5.3.3 Arquitectura Funcional

Os agentes da classe dos guardiões são constituídos basicamente por dois módulos funcionais distintos:

- **Unidade Web**.

  Plataforma virtual que suporta e apoia a interacção dos utilizadores com o sistema, disponibilizando os meios e as estruturas para a comunicação com a unidade protocolar do agente guardião. É constituída por um conjunto de páginas escritas em HTML e um "browser" da Internet[4] utilizado pelo utilizador.

- **Unidade Protocolar**.

  Conjunto de mecanismos que permite estabelecer o contacto entre a unidade Web e o ambiente do sistema, aplicando os critérios de autenticação e verificação de permissões de acesso previamente definidos pelos administradores do sistema e dos agentes cognitivos que nele estão integrados.

A organização e a constituição dos agentes guardiões foram definidas de forma a satisfazer os requisitos básicos para o estabelecimento deste tipo de interface: o desenvolvimento de um módulo que garantisse a interacção com os utilizadores da Web e de outro que estabelecesse a ligação entre esse e o ambiente do sistema BEABLE.

Um "browser" da Web permitirá aos utilizadores aceder aos serviços e recursos da unidade Web. A interacção do utilizador com o sistema é assegurada através de um conjunto de páginas escritas em HTML, complementadas com programas de suporte para o estabelecimento da "ponte" virtual com a outra unidade.

[4] O sistema de "browsing" utilizado no desenvolvimento e implementação dos agentes guardiões foi o NetScape da NetScape Corporation.

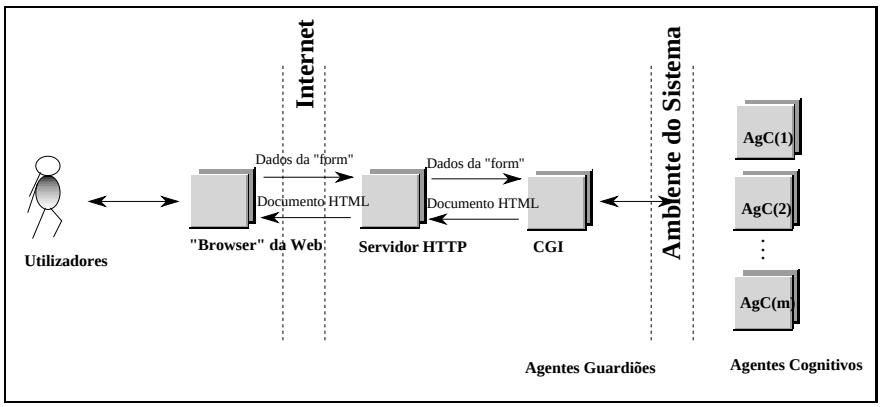


Figura 5.7: Interligação do Sistema BeAble com a Web.

Os utilizadores efectuam os seus pedidos utilizando objectos gráficos - formulários ("forms") - apresentados nas páginas da unidade Web. Após o preenchimento dos atributos desses formulários, que variam de pedido para pedido conforme o contexto da resolução do problema, os dados transmitidos pelo utilizador são recebidos por um guião "Common Gateway Interface" (CGI), que os analisa em termos de correcção e consistência, de acordo com os dados que possui no momento. Se o guião CGI não detectar qualquer irregularidade, estabelece a ligação com o ambiente do sistema e envia a mensagem correspondente ao pedido do utilizador, à atenção dos agentes cognitivos. Após o envio da mensagem, a unidade protocolar fica à espera de uma eventual resposta dos agentes cognitivos.

A unidade protocolar é constituída por um conjunto de guiões CGI escritos em Prolog[5]. Os guiões são invocados a partir do sistema de "browsing", utilizando-se referências (nome do guião e local onde está posicionado) contidas nos formulários das páginas em HTML. A forma como é estabelecida

[5]Os guiões CGI foram implementados com base numa biblioteca de predicados para o SICStus Prolog desenvolvida no Departamento de Ciências da Computação da Universidade Técnica de Madrid (http://www.clip.dia.fi.upm.es/miscdocs/html_pl/html_pl.html).

a comunicação entre o "browser", o guião CGI e o sistema é apresentada na Figura 5.7. Em D.Cabeza & M.Hermenegildo (1996) podem-se encontrar elementos para uma explicação mais detalhada acerca da forma como o processo de comunicação é estabelecido e a informação transferida entre um formulário e um guião CGI escrito em Prolog.

Na unidade protocolar é mantido um registo das interacções entre os utilizadores da Web e os agentes do sistema. Este processo tem como objectivo manter actualizada a história dos relacionamentos do sistema com os utilizadores da Web. Complementarmente, esse registo de ocorrências é também utilizado pelo guião CGI por forma a garantir-se que a resposta de um utilizador respeita os parâmetros da mensagem enviada pela unidade protocolar, que também aí foram armazenados.

O comportamento da unidade protocolar de um agente guardião é muito semelhante ao de um agente de interface. Estas unidades estão localizadas em nodos específicos da Internet, com endereços fixos e actuam como pontes virtuais entre as unidades Web e o sistema BEABLE. Sempre que o módulo de comunicação da unidade protocolar detecta uma mensagem proveniente da unidade Web correspondente, a mensagem é autenticada e as permissões de acesso verificadas, de acordo com as directivas actuais do agente guardião. Passado este "crivo", a mensagem é reescrita segundo um formato próprio do agente guardião e enviada para o sistema.

A arquitectura funcional da unidade protocolar dos agentes da classe dos guardiões (Figura 5.8) é caracterizada pelos seguintes módulos funcionais:

- **Comunicação**.

  Módulo que assegura as comunicações com o ambiente do sistema, sen-

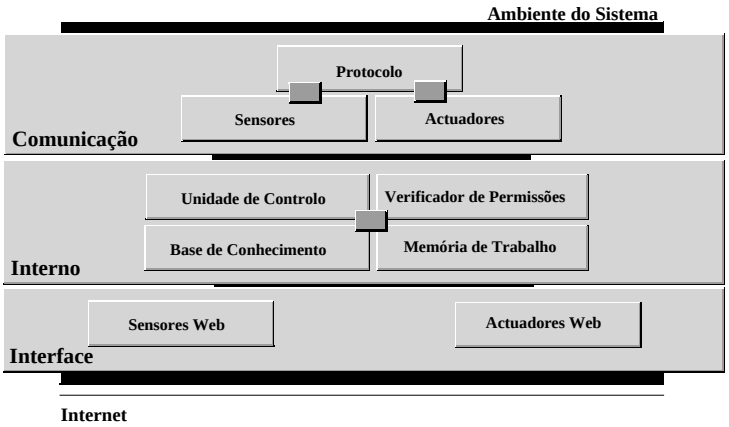


Figura 5.8: Arquitectura Funcional da Unidade Protocolar.

do constituído por três blocos funcionais:

- **Protocolo**.

  Bloco que estabelece e garante os processos de comunicação entre a unidade protocolar do agente guardião e o ambiente do sistema.

- **Sensores**.

  Conjunto de mecanismos que analisa o tráfego de estruturas de conhecimento no ambiente do sistema, procurando identificar as mensagens endereçadas ao utilizador que o agente guardião está a suportar.

- **Actuadores**.

  Conjunto de mecanismos que coloca no ambiente do sistema os comandos enviados pelo utilizador via unidade Web, após autenticação pelo módulo de controlo do bloco interno da unidade protocolar.

- **Interno**.

  Módulo que assegura as comunicações com a unidade Web. Nele se

desenvolvem os processos de decisão do agente guardião, sendo constituído por quatro blocos funcionais:

- **Unidade de Controlo**.

  Bloco que interpreta as mensagens enviadas pela unidade Web.

- **Verificador de Permissões**.

  Conjunto de mecanismos que verifica as permissões dos utilizadores quando acedem à comunidade local de agentes cognitivos.

- **Memória de Trabalho**.

  Área onde se armazena a informação relacionada com os processos de consulta dos utilizadores à comunidade local de agentes cognitivos.

- **Base de Conhecimento**.

  Repositório de informação respeitante às normas de acesso, formatos de mensagens e acções de controlo a desenvolver pelo agente guardião.

- **Interface**.

  Bloco que assegura o diálogo entre a unidade protocolar e a unidade Web do agente guardião, sendo constituído por quatro blocos funcionais:

  - **Sensores Web**.

    Conjunto de mecanismos que recebe os pedidos do utilizador enviados através da unidade Web, convertendo-os para o formato interno do sistema de controlo do agente guardião.

  - **Actuadores Web**.

    Conjunto de mecanismos que recebe as instruções da unidade de

controlo relativas aos pedidos dos utilizadores entretanto já concretizados, converte-as para HTML e prepara a mensagem a ser enviada à unidade Web.

A Figura 5.9 apresenta um excerto de uma peça de código em Prolog de parte do sistema de comunicação de um guião CGI de uma unidade protocolar de um agente guardião, respeitante às acções relacionadas com o pedido da lista dos agentes cognitivos activos no sistema por parte de um utilizador.

A unidade protocolar possui mecanismos de "time-out" que lhe permitem verificar periodicamente se as mensagens que colocou à consideração dos agentes cognitivos foram objecto de tratamento. Se o tempo de permanência da mensagem no ambiente do sistema exceder o concedido pela unidade protocolar, esta devolve-a à unidade Web. Existem apenas duas situações em que a unidade protocolar efectua essa devolução:

- quando a mensagem não passa no processo de análise de entrada no sistema;
- quando o tempo de permanência da mensagem no sistema excedeu o concedido pela unidade protocolar.

Em qualquer dos casos, a unidade protocolar devolve a mensagem com uma justificação dos motivos que originaram a sua devolução. As respostas aos utilizadores das unidades Web são identificadas e recolhidas no ambiente do sistema pelo módulo de percepção do agente guardião. Segue-se o processo de análise e a geração do código em HTML correspondente.

```
%
% /// Activação do guião CGI
beable
        :-
        % /// Recepção dos valores dos atributos da ‘‘form’’ relativos
        % /// ao pedido do utilizador.
        html_get_form_input(In),
        html_get_value(In,community,Le),
        html_get_value(In,user,Us),
        html_get_value(In,code,Co),
        html_get_value(In,typedoc,Td),
        % /// Ligação do guião ao ambiente do sistema BeAble.
        connect_environment(Le,Us,Co,Td,Re),
        % /// Geração do código HTML correspondente ao cabeçalho e ao
        % /// rodapé da nova página da unidade Web.
        get_top(Top),
        get_bottom(Bot),
        % /// Geração do código HTML correspondente à nova página da
        % /// unidade Web contendo a resposta ao pedido do utilizador.
        output_html([Top,Re,Bot]).

connect_environment(Le,Us,Co,Td,Re)
        :-
        % /// Verificação do perfil do utilizador.
        get_authorization(Le,Us,Co),
        % /// Obtenção do endereço actual do ambiente do sistema
        % /// correspondente à comunidade dos agentes cognitivos
        % /// solicitada pelo utilizador.
        get_environment_address(Le,Us),
        % /// Análise do pedido do utilizador.
        what_to_do(Td,Re),
        % /// Quebra da ligação entre a unidade protocolar e o am-
        % /// biente do sistema.
        disconnect_environment(Us).

% /// Tratamento do pedido do utilizador relativo à lista dos agentes
% /// cognitivos activos no ambiente do sistema.
what_to_do('exp',Re)
        :-
        % /// Recolha da informação relativa aos agentes cognitivos
        % /// activos no sistema.
        get_agent_list(LAg),
        % /// Geração do código HTML correspondente à informação
        % /// recolhida sobre os agentes.
        agent_list_html(LAg,Re).
```

Figura 5.9: Peça de Código em Prolog do Sistema de Comunicação de uma Unidade Protocolar de um Agente Guardião.

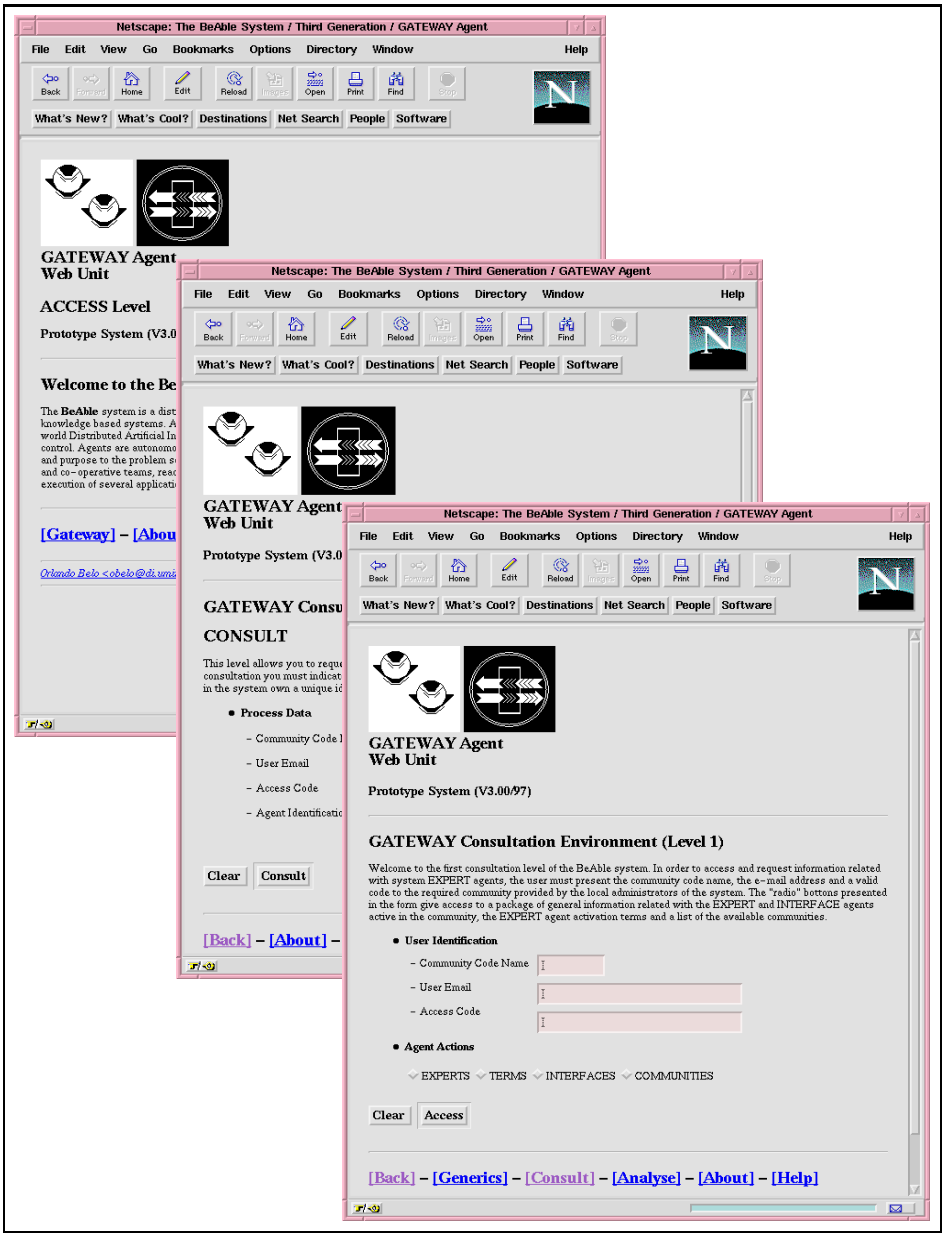


Figura 5.10: Exemplos de algumas Páginas da Unidade Web.

A Figura 5.10 apresenta algumas das páginas da unidade Web visualizadas num sistema de "browsing". As páginas geradas pela unidade protocolar são então enviadas para a unidade Web, que nesta situação é o próprio "browser". O processo recomeça após a resposta do utilizador à nova página que recebeu, ou é simplesmente interrompido através de uma ordem directa do utilizador.

```
procedimento Unidade-Protocolar
   recolha-pedido-utilizador(Utilizador,Pedido);
   autenticação-pedido-utilizador(Utilizador,Pedido,Autenticação)
   se Autenticação então
      conversão-pedido(Utilizador,Pedido,Pedido-Interno,Conversão)
      se Conversão então
         análise-parâmetros(Pedido-Interno,Análise-Correcta)
            se Análise-Correcta então
               coloca-pedido-ambiente-sistema(Utilizador,Pedido-Interno)
               repetir
                  recolher-respostas-agentes-cognitivos(Time-Out,Respostas)
               até Time-Out ∨ Respostas
               se Time-Out então
                  informar-utilizador(Utilizador,Time-Out);
                  abandonar-pedido-utilizador(Utilizador,Pedido-Interno)
               senão
                  conversão-respostas(Utilizador,Respostas,Texto-HTML);
                  enviar-mensagem(Utilizador,Pedido,Texto-HTML)
               fim-se
            fim-se
      fim-se
   fim-se
   actualizar-logfile-unidade-protocolar(Utilizador,Pedido,Autenticação,
         Conversão,Análise-Correcta,Time-Out,Respostas)
fim-procedimento
```

Figura 5.11: Descrição Genérica do Comportamento de uma Unidade Protocolar de um Agente Guardião.

Para cada utilizador que solicite o acesso ao sistema BEABLE via Web é necessária a activação de um agente guardião. O processo de interacção com o sistema é idêntico ao desencadeado quando se utiliza o ambiente de um

agente de interface normal. A vantagem é que o utilizador não necessita de estar localizado numa plataforma computacional onde exista uma réplica de um agente de interface.

Na Figura 5.11 é apresentada uma breve descrição do comportamento de uma unidade protocolar de um agente da classe dos guardiões[6].

5.4 Os Agentes Cognitivos

Os especialistas humanos são emulados no sistema através dos agentes cognitivos (O.Belo & J.Neves 1995*c*). De uma forma geral, a estrutura de um agente cognitivo é semelhante à de uma "shell" para a implementação de SP, enriquecida com os mecanismos necessários à comunicação entre agentes. A "shell" desenvolvida disponibiliza um meio único para a implementação dos agentes cognitivos. A estrutura modular que apresenta simplifica, em grande medida, os processos de manutenção e reutilização de software do agente.

Os agentes cognitivos são entidades independentes e autónomas, cujas acções são orientadas pelo seu estado de conhecimento interno e pela sua capacidade de reacção a novos eventos. Quando activos e interligados com o ambiente

[6] O algoritmo descreve as principais acções desenvolvidas por uma unidade protocolar de um agente guardião durante o processo de recepção, análise e tratamento dos pedidos de consulta enviados pelos utilizadores aos agentes cognitivos através de um "browser". Este processo está organizado em quatro fases: 1) recepção - na qual a unidade protocolar recolhe (*recolha-pedido-utilizador/2*) o pedido (*Pedido*) do utilizador (*Utilizador*) enviado através da unidade Web; 2) autenticação - verificação do perfil do utilizador e do tipo de acesso que possui; 3) processamento - análise do pedido (*análise-parâmetros/2*) do utilizador e sua colocação no ambiente do sistema (*coloca-pedido-ambiente-sistema/2*) após conversão (*conversão-pedido/4*) para o formato reconhecido pelos agentes cognitivos; 4) envio - a unidade protocolar aguarda uma resposta dos agentes cognitivos e, após a sua recepção (*recolher-respostas-agentes-cognitivos/2*), converte-a para HTML (*Texto-HTML*) e envia-a (*enviar-mensagem/3*) para a unidade Web, que a comunicará ao utilizador através do "browser" em utilização.

do sistema, agem como entidades assíncronas em contínua execução.

A activação dos agentes cognitivos pode ser desencadeada de três[7] formas distintas, conforme a via utilizada:

- **agentes de interface**, que colocam no ambiente do sistema requisitos operacionais ou listas de estímulos, a fim de provocarem a reacção dos agentes cognitivos;
- **agentes da classe dos guardiões**, que recebem dos utilizadores via Internet solicitações equivalentes às colocadas pelos agentes de interface;
- **outros agentes cognitivos**, que ao realizarem operações de manipulação das estruturas de conhecimento partilhadas pela comunidade de agentes chamam a atenção de outros agentes cognitivos.

Quando activados pelos seus mentores[8], os agentes cognitivos comunicam ao agente supervisor do sistema a sua identificação e o seu domínio de aplicação, bem como informação relacionada com a forma de activação. Cada agente cognitivo pode suportar em simultâneo mais do que um processo de consulta, podendo assim atender vários utilizadores ao mesmo tempo.

[7] Existe uma quarta forma de activação dos agentes cognitivos, desencadeada com base nos agentes da classe de encaminhamento. Esta possibilidade será apresentada no Capítulo 6, dedicado à implementação do modelo de multicomunidades, no qual os agentes da classe de encaminhamento assumem um papel preponderante.

[8] Cada agente cognitivo tem um supervisor local, cuja função principal é garantir a sua operacionalidade. As suas tarefas estão relacionadas, essencialmente, com a activação ou desactivação do agente cognitivo e com a monitorização das suas actividades. O papel do supervisor foi definido com vista a garantir que um agente cognitivo só será ligado ao sistema após haver passado por todo um processo de validação.

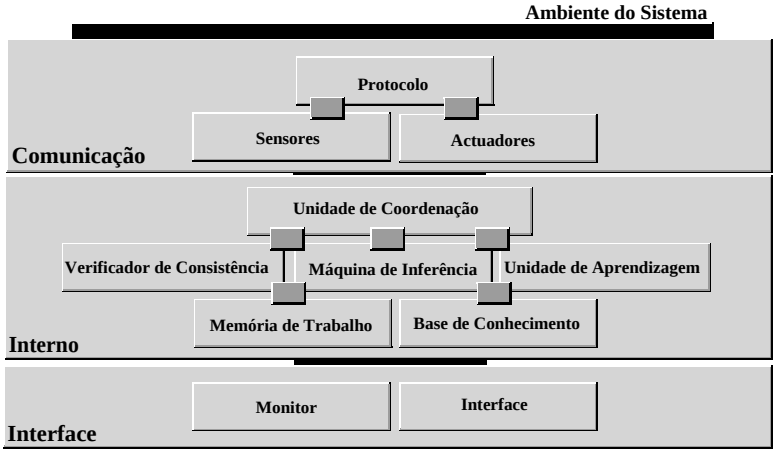


Figura 5.12: Arquitectura Funcional de um Agente Cognitivo.

5.4.1 Arquitectura Funcional

A arquitectura funcional de um agente cognitivo (Figura 5.12) segue em alguns aspectos a de um agente de interface, estando organizada em torno dos seguintes módulos funcionais:

- **Comunicação**.

  Controla e faz a gestão dos processos de comunicação entre o agente cognitivo e o ambiente do sistema, sendo constituído por três blocos funcionais:

  - **Protocolo**.

    Apoiado por primitivas do Linda, este bloco estabelece os canais de ligação do agente cognitivo ao ambiente do sistema.

  - **Sensores**.

    Conjunto de mecanismos que, associado ao bloco de protocolo, analisa o conhecimento que circula no sistema, a fim de detec-

tar estruturas de conhecimento com utilidade para os processos internos do agente cognitivo.

- **Actuadores**.

  Conjunto de mecanismos que, associado ao bloco de protocolo, comunica para o sistema a informação resultante da aplicação dos processos de raciocínio do agente ou proveniente das suas bases locais.

- **Interno**.

  Módulo responsável pelas acções de coordenação, raciocínio e armazenamento de informação relacionada com os processos de consulta desencadeados pelos utilizadores, entre outras. É constituído pelos seguintes blocos funcionais:

  - **Unidade de Coordenação**.

    Bloco que permite ao agente cognitivo:

    * analisar localmente a informação que recebe, validá-la de acordo com o conhecimento que possui acerca do problema e fazer "juízos" sobre a sua aplicabilidade;
    * activar os mecanismos de prova;
    * fornecer aos agentes de interface informação relativa aos processos que tem em curso (ex.: propriedades verificadas, soluções alcançadas, cadeias de inferência);
    * aplicar o modelo de coordenação definido pelo seu supervisor, por forma a não criar "desfasamentos" operacionais com os outros agentes;

  - **Unidade de Aprendizagem**.

    Analisa as condições em que decorre a resolução de problemas, por

forma a optimizar o seu desempenho com base em experiências passadas e a assimilar novo conhecimento.

- **Verificador de Consistência**.

  Identifica, analisa e resolve eventuais quebras de consistência no conhecimento que o agente cognitivo possui acerca dos problemas em resolução.

- **Máquina de Inferência**.

  Opera na base de conhecimento do agente aplicando estratégias de resolução e formas de raciocínio, com vista a obter interpretações para as questões formuladas.

- **Base de Conhecimento**.

  Repositório do conhecimento do agente relativo ao seu domínio de aplicação, ao ambiente em que se integra e aos outros agentes que partilhem do mesmo ambiente.

- **Memória de Trabalho**.

  Área que armazena temporariamente informação respeitante aos processos em curso.

- **Interface**.

  Módulo que suporta a comunicação entre o agente cognitivo e o respectivo supervisor, sendo constituído pelos seguintes blocos funcionais:

  - **Monitor**.

    Bloco cuja função é relatar ao supervisor do agente cognitivo todas as acções que este realizou localmente.

  - **Interface**.

    Permite ao supervisor controlar a operação do agente cognitivo.

A Figura 5.13 apresenta um excerto de uma peça de código em Prolog de parte do sistema de comunicação de um agente cognitivo, relacionada com as seguintes acções:

- activação do agente cognitivo num processo de resolução;
- análise das listas de activação dos agentes cognitivos relacionadas com os processos dos utilizadores activos no sistema.

Durante a fase de activação, os agentes cognitivos carregam para memória as bases de conhecimento que estão contidas em ficheiros especiais. Cada agente adoptará o comportamento que estiver descrito no respectivo ficheiro (para um ficheiro distinto um comportamento distinto). Este ficheiro é complementado por outro que contém uma descrição textual do domínio de conhecimento do agente, da sua perícia e do seu conhecimento relativamente ao problema em curso, funcionando como um descritivo das capacidades do mesmo.

A base de conhecimento de um agente cognitivo está organizada da seguinte forma:

- **Genéricos**.
  Ficheiro de texto (*<id_base>.gen*) que contém uma descrição, em português corrente, do domínio de aplicação, da perícia e do conhecimento do agente cognitivo.
- **Base de Conhecimento**.
  Ficheiro de texto (*<id_base>.rul*) contendo o conhecimento que define o comportamento do agente e suporta as suas acções. É constituída pelas seguintes partes:

```
%
% /// Activação do agente cognitivo num processo de resolução.
ag01    :::     activation ::
                [
                identification(Ag),
                % /// Verificação das condições de activação.
                existe_obj_bb(bb(Ag,t_ag_activ(Ag,Co,Ar,Es))),
                existe_obj_bb(bb(Us,Ag,t_us_inic(Us,Ag)))
                ]
        -->
                [
                % /// Acções locais de monitorização.
                monitoring(d,Us,agac01),
                insere_obj_bb(bb(Ag,Us,t_ag_process(Us,Ag,Co,Ar,Es))),
                remove_obj_bb(bb(Us,Ag,t_us_inic(Us,Ag))),
                % /// Activação dos mecanismos de inferência.
                inferencia(Us,run)
                ].
(...)

% /// Análise das listas de activação dos agentes cognitivos
% /// relacionadas com os processos dos utilizadores activos
% /// no sistema.
ag03    :::     aal     ::
        if
                [
                % /// Recolha das listas de activação.
                obtem_lista_bb(bb(_,t_us_aar(_,_)),La),
                La \== []
                ]
        -->
                [
                monitoring(d,"",agal01), inf_activ_us(Lu),
                % /// Análise das listas e eventual activação dos
                % /// mecanismos de inferência.
                ver_lista_aar(La,Lu)
                ].

(...)
```

Figura 5.13: Activação de um Agente Cognitivo e Análise das Listas de Activação.

- **Identificação do agente**.

  Informação relativa à identificação do agente cognitivo: nome, domínio de aplicação, etc.

- **Termos e chaves de activação**.

  Informação necessária para caracterizar a forma como o agente pode ser activado.

- **Informação relativa a outros agentes**.

  Informação relacionada com os domínios de aplicação, perícia e conhecimento de outros agentes no sistema.

- **Informação de suporte**.

  Descrições relativas aos identificadores dos termos lógicos utilizados nas regras, denotando especificidades do agente, bem como informação relacionada com as acções de interacção entre os agentes cognitivos e os utilizadores do sistema.

- **Regras**.

  Regras de produção na forma *Condição* –> *Acção*, dando corpo ao sistema de inferência do agente.

Na base de conhecimento encontram-se igualmente definidos os limites para a intervenção de cada um dos agentes cognitivos. São regras que delimitam, não a sua perícia, mas o seu comportamento oportunístico, definindo as condições em que os agentes cognitivos podem intervir. Trata-se de uma medida essencial para a coordenação geral da actividade do sistema, definida em termos do comportamento de cada um dos agentes cognitivos.

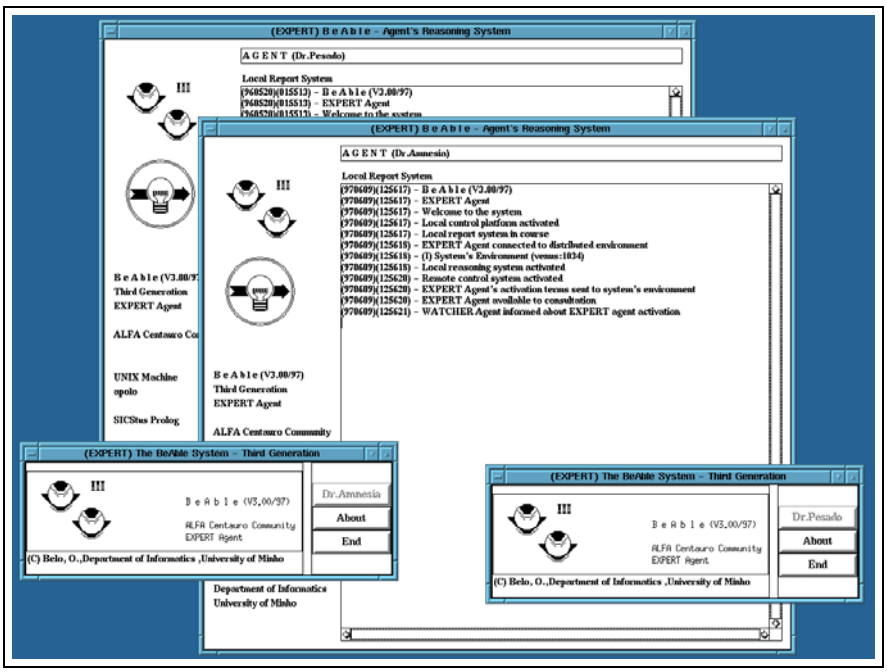


Figura 5.14: Ambiente Principal de um Agente Cognitivo

5.4.2 Activação dos Agentes Cognitivos

Os agentes cognitivos são activados localmente pelos administradores, que decidem quando está o agente em condições de se tornar numa entidade activa no sistema. Os agentes cognitivos tornam-se operacionais com a invocação da "shell" que suporta a sua execução. Na fase de arranque, a "shell" verifica se o ambiente do sistema está ou não activo. Caso não esteja disponível, a "shell" não permite que a fase de arranque se concretize, alertando o administrador local para a situação. Em circunstâncias normais de funcionamento, estabelece a ligação do agente cognitivo ao ambiente do sistema, procedendo de seguida ao carregamento da base de conhecimento do agente. Após estas operações, o agente estará preparado para intervir nos processos de resolução de problemas (Figura 5.14).

Cada agente cognitivo está em constante observação do ambiente do sistema, a fim de detectar qualquer "pacote" de conhecimento que lhe seja dirigido ou lhe possa ser útil para os processos de resolução nos quais está envolvido. Os agentes cognitivos podem intervir em simultâneo em mais do que um processo, contribuindo para a resolução de mais do que um problema.

Os agentes cognitivos podem entrar ou sair do sistema (entenda-se ligar-se ou desligar-se) por acção do supervisor, através do seu ambiente de controlo, não afectando com isso a operacionalidade global do sistema.

```
procedimento Agente-Cognitivo
    activação-agente(Código,Agente,Domínio,Especialização);
    informa-supervisor(Código,Agente,Domínio,Especialização)
    repetir
        obter-listas-estímulos-ambiente(Utilizadores,Listas-Estímulos)
        se Listas-Estímulos então
            processar-listas-estímulos(Utilizadores,Lista-Estímulos,
                    Utilizadores-Activados,Resultados)
            se Resultados então
                enviar-resultados-utilizadores(Utilizadores-Activados,Resultados);
                criar-listas-activação-utilizadores(Utilizadores-Activados,Resultados);
                marcar-listas-estímulos(Utilizadores-Activados,Resultados)
            fim-se
        fim-se
        obter-listas-activação-ambiente(Utilizadores-Activos,Listas-Activação)
        se Lista-Activação então
            analisar-listas-activação(Utilizadores-Activos,Listas-Activação)
        fim-se
        acções-monitorização(Código,Agente);
        verificar-instruções-supervisor(Fim)
    até Fim
    desactivar-agente(Código,Agente,Domínio,Especialização)
fim-procedimento
```

Figura 5.15: Comportamento Genérico de um Agente Cognitivo.

O estado interno de cada agente cognitivo não pode ser investigado directamente por outros agentes cognitivos operacionais no sistema. Não há pos-

sibilidade de um agente inspeccionar e manipular a informação contida nos estados internos de outro agente. Esta forma de contenção e restrição ao acesso à informação localizada no estado interno de um agente cognitivo permite assegurar minimamente a privacidade da informação que o agente contém. Contudo, os agentes podem possuir informação acerca de outros em termos da sua perícia, conhecimento ou mesmo formas de actuação. Esta informação está contida na sua base de conhecimento e é-lhes facultada por forma a permitir-lhes desenvolver processos de cooperação com outros agentes cognitivos.

A Figura 5.15 apresenta uma descrição do comportamento genérico de um agente cognitivo[9].

5.4.3 Acções de Coordenação e de Cooperação

A resolução de um problema num ambiente de um SM envolve, na maioria dos casos, a intervenção de mais do que uma entidade. A própria natureza de um SM e os motivos da sua implantação, apesar de serem directamente

[9]O algoritmo descreve as principais acções desenvolvidas por um agente cognitivo durante o arranque e inicialização do seu ambiente, na análise das listas de estímulos e na activação dos processos de consulta dos utilizadores. Este processo está dividido nas seguintes fases: 1) arranque - na qual o agente cognitivo é activado (*activação-agente/4*) e o supervisor do sistema informado (*informa-supervisor/4*) acerca da sua operacionalidade; 2) tratamento das listas de estímulos - na qual o agente cognitivo recolhe (*obter-listas-estímulos-ambiente/2*) do ambiente do sistema as listas de estímulos (*Listas-Estímulos*) geradas pelos agentes de interface com respeito aos processos dos utilizadores (*Utilizadores*), analisando-as (*processar-listas-estímulos/4*) e, caso verifique que contêm informação suficiente para a criação de processos de consulta, informando os utilizadores (*enviar-resultados-utilizadores/2*) dos resultados alcançados e gerando (*criar-listas-activação-utilizadores/2*) as listas de activação correspondentes; 3) tratamento das listas de activação - na qual o agente cognitivo recolhe (*obter-listas-activação-ambiente/2*) e analisa (*analisar-listas-activação/2*) as listas de activação correspondentes aos processos de resolução de problemas em curso no sistema; na Figura 5.16 apresenta-se uma descrição mais detalhada desta fase; 4) finalização - na qual o agente termina a sua execução e abandona o sistema (*desactivar-agente/4*).

dependentes do domínio de aplicação e da perícia e conhecimento apresentados pelos seus agentes, suportam e implicam diferentes intervenções. Assim, a necessidade de se estabelecer um modelo de coordenação das actividades dos agentes é vital para o bom desempenho do SM (N.Jennings 1993*a*).

Um bom modelo de coordenação permitirá assegurar que os agentes intervêm no sentido de contribuirem colectivamente para a resolução do problema, justificando porventura as suas intervenções, directas ou indirectas, a partir das realizadas por outros agentes. A eficácia do modelo poderá, por conseguinte, ser avaliada em termos da capacidade evidenciada pelos agentes para lidarem com diferentes situações, tais como:

- a degradação gradual do desempenho dos agentes ao longo dos processos de resolução de problemas em que vão intervindo;
- a atenção do sistema perante eventuais ocorrências de situações de contenção ou de "deadlock" e sua eliminação;
- a eliminação de possíveis conflitos resultantes de processos de partilha de recursos entre os agentes;
- a prevenção da realização de acções por parte dos agentes que possam resultar em tentativas infrutíferas de resolução do problema ou que forneçam soluções incorrectas ou redundantes para o mesmo, com as consequentes perdas de tempo e gastos de recursos.

A utilização de um modelo de coordenação com base numa entidade de controlo centralizada, apesar de ser uma opção "cómoda" em termos de gestão de processos, apresenta grandes desvantagens em ambientes cujos agentes

revelem uma postura dinâmica. A necessidade de comunicarem sistematicamente entre si provocaria um enorme tráfego de mensagens, que envolveria a unidade de coordenação em processos de análise e validação de informação monopolizadores dos seus recursos e dos do sistema. Tal circunstância conduziria a situações de contenção e de "deadlock", afectando o desempenho dos agentes envolvidos e, consequentemente, de todo o sistema.

No caso do sistema BEABLE, a descentralização das acções de coordenação, através da distribuição de competências pelos agentes, revela-se mais vantajosa e menos restritiva em termos de desempenho. O grau de autonomia apresentado pelos agentes do sistema, e em particular pelos cognitivos, exige que as políticas de coordenação e controlo do sistema tenham reflexo ao nível das estruturas de conhecimento do próprio agente. Isto significa que o agente deverá possuir na sua base de conhecimento elementos suficientes para assegurar que as suas intervenções em processos de resolução partilhada ocorram pela positiva. Este cenário é assegurado através de um conjunto mínimo de propriedades relacionadas com o problema em causa, que deve estar disponível para o agente, de modo que os seus procedimentos de activação sejam satisfeitos. Este modelo permite reduzir as intervenções potencialmente infrutíferas dos agentes e assegurar um primeiro nível de coordenação e controlo descentralizado de todo o sistema. O desenvolvimento de processos de cooperação entre os agentes tenderá a minimizar o número de tais intervenções. Através da simples partilha de informação, os agentes poderão defnir quais as estratégias mais aconselháveis para a resolução de um dado problema.

Nos processos de consulta individual, que envolvem apenas um agente de interface e um agente cognitivo, as acções de coordenação do processo são

desenvolvidas com base num diálogo entre os agentes envolvidos. Qualquer situação anómala que se verifique é resolvida à custa de troca de mensagens entre o agente cognitivo e o agente de interface, com suporte dos seus sistemas de comunicação.

Nos processos colectivos, a situação altera-se significativamente. As acções de coordenação são suportadas pelas estruturas partilhadas pelos agentes ao nível do ambiente do sistema e pelos estados dos problemas armazenados nas memórias de trabalho de cada um dos agentes cognitivos envolvidos. A informação que os agentes cognitivos têm na sua memória de trabalho resultou, na maioria dos casos, da aplicação das suas regras, da troca directa de informação entre esses agentes ou da aquisição de conhecimento através das estruturas de partilha de informação relacionadas com cada um dos processos dos utilizadores.

As estruturas envolvidas na partilha global de informação podem ser de dois tipos:

- **Listas de estímulos**.

  Contêm propriedades indicadas pelos utilizadores para caracterizar o estado incial dos seus problemas e são utilizadas para "despertar" a atenção dos agentes cognitivos. Existe uma lista de estímulos por cada utilizador que requereu um processo colectivo de consulta.

- **Listas de activação**.

  Contêm conjuntos de registos na forma <*agente, propriedade, valor, data, hota*>, que caracterizam as diversas propriedades reveladas pelos agentes cognitivos intervenientes nos processos de resolução de problemas dos utilizadores. Estas listas compilam o conhecimento dos diver-

sos agentes cognitivos com respeito aos processos colectivos de consulta desencadeados pelos utilizadores. Existe uma lista de activação por cada utilizador que requereu um processo colectivo de consulta.

As listas de estímulos e de activação são criadas para cada processo colectivo de consulta e guardadas no ambiente do sistema enquanto esses processos se mantiverem activos. Os agentes cognitivos utilizam-nas para partilharem o conhecimento que possuem relativamente à resolução de um problema de um utilizador. Esta forma de divulgação de informação permite reduzir significativamente a comunicação entre os agentes, bem como o tempo de resolução dos problemas.

A Figura 5.16 apresenta uma descrição do comportamento de um agente cognitivo na análise das listas de activação[10].

[10]O algoritmo descreve as principais acções desenvolvidas por um agente cognitivo na análise das listas de activação correspondentes aos processos de resolução de problemas dos utilizadores em curso no sistema. Este processo está dividido nas seguintes fases: 1) recolha - na qual o agente cognitivo verifica se no ambiente do sistema existem *Listas de Activação de Agentes* (LAA) (*Lista-LAA*); se existirem, ele copia-as para a sua memória de trabalho (*obter-listas-activação-ambiente/1*) e identifica a que utilizadores correspondem (*identificar-utilizadores-activos/2*); 2) análise e processamento - para cada utilizador (*Utilizador*) referido nas listas de activação recolhidas, o agente cognitivo verifica (*identificar-novos-dados-utilizador/4*) a existência ou não de novos dados (*Novos-Dados*), com base no útimo estado (*Estado*) que possui relativo ao problema desse utilizador; se forem detectados novos dados, o agente identifica (*seleccionar–items-relevantes-novos-dados/3*) as novas propriedades e respectivos valores (*Items*) e verifica a sua consistência (*verificação-consistência/3*) relativamente a dados anteriores; se não se resgitar qualquer situação de quebra de consistência, os novos dados são armazenados na memória de trabalho do agente (*integrar-novos-items-memória-trabalho/2*), os mecanismos de inferência são activados (*aplicar-mecanismos-inferência/2*) e os resultados (*Resultados*) que daí resultarem armazenados (*integrar-resultados-memória-trabalho/2*); no caso dos resultados obtidos constituirem uma proposta de solução para o problema, o agente actualiza (*actualizar-lista-LAA-utilizador-amb-sistema/2*) a lista de activação do utilizador correspondente no ambiente do sistema com a solução; caso contrário, informa o utilizador (*enviar-mensagem-utilizador/2*) dos resultados, requerendo, normalmente, mais informação acerca do problema; na hipótese de se verificar uma quebra de consistência nos dados do problema do utilizador, o agente tenta resolver os conflitos (*resolução-conflitos/2*) que originaram essa situação e, no caso de não ter sucesso, finaliza o processo (*terminar-processo-utilizador/1*) desse utilizador e divulga (*divulgar-situação/1*) o sucedido junto da comunidade de agentes cognitivos.

```
procedimento Analisar-Listas-Activação
    obter-listas-activação-ambiente(Lista-LAA);
    identificar-utilizadores-activos(Lista-LAA,Utilizadores)
    para  cada Utilizador ∈ Utilizadores até Lista-LAA = [ ]
        fazer
            seleccionar-lista-propriedades(Utilizador,Lista-LAA,
                    Propriedades)
            recolher-dados-último-estado(Utilizador,Estado)
            identificar-novos-dados-utilizador(Utilizador,Estado,
                    Propriedades,Novos-Dados);
            se  Novos-Dados então
                seleccionar–items-relevantes-novos-dados(Utilizador,
                        Novos-Dados,Items);
                verificação-consistência(Utilizador,Items,Consistência)
                repetir
                    se Consistência então
                        integrar-novos-items-memória-trabalho(Utilizador,Items);
                        aplicar-mecanismos-inferência(Utilizador,Resultados);
                        integrar-resultados-memória-trabalho(Utilizador,
                            Resultados);
                        caso Resultados.Tipo seja
                            solução     : actualizar-lista-LAA-utilizador-sistema
                                          (Utilizador,Resultados)
                            outro       : enviar-mensagem-utilizador
                                          (Utilizador,Resultados)
                        fim-caso
                        Fim ← verdadeiro
                    senão
                        resolução-conflitos(Utilizador,Consistência)
                        se ¬ Consistência então
                            terminar-processo-utilizador(Utilizador);
                            divulgar-situação(Utilizador)
                        senão
                            Fim ← falso
                        fim-se
                    fim-se
                até Fim
            fim-se
    fim-para
fim-procedimento
```

Figura 5.16: Análise das Listas de Activação dos Agentes Cognitivos.

5.4.4 Mecanismos de Aprendizagem

Por vezes, ao consultar-se um agente cognitivo, é-se confrontado com uma certa rigidez na forma como este aborda, analisa ou propõe uma solução para um dado caso. Esta situação, em muitos dos casos, revela-se consoante o tipo de aplicação em causa. Ora nem sempre é desejável ou conveniente redefinir com frequência a forma como os agentes abordam os problemas. É desejável uma certa continuidade e razoabilidade neste processo.

Para uma melhor compreensão desta situação, considere-se o caso do diagnóstico médico em que um agente, nos últimos 20 casos analisados diagnosticou 8 casos de gripe. Será de esperar que este agente, num próximo caso, analise com outro tipo de estratégia o mesmo tipo de enfermidade? Um diagnóstico em tempo de Inverno será executado da mesma forma que um diagnóstico em tempo de Verão?

Nestas situações, a aplicação por parte dos agentes cognitivos de um mesmo tipo de estratégia, de problema para problema, pode não ser a forma mais correcta para abordar e analisar o problema ou encontrar uma solução para o mesmo. Por conseguinte, o provimento dos agentes cognitivos com estruturas e funcionalidades necessárias à análise de procedimentos ou estratégias é uma hipótese a considerar, numa óptica de optimização do desempenho dos agentes cognitivos. Basicamente o problema resume-se à maneira de dotar os agentes cognitivos com os meios para que estes se auto-reestruturem. Por outras palavras, não é mais do que dar-lhes a "aptidão" para reescreverem a sua própria base de conhecimento.

As estratégias a seguir para a resolução de problemas por parte de um agente cognitivo são parte integrante da base de conhecimento do agente, compreen-

dendo estruturas de controlo, informação relacionada com os problemas em análise e conhecimento acerca de outros agentes. A alteração de estratégias de resolução implica que se redefinam os procedimentos do agente cognitivo que operam sobre a sua base de conhecimento, produzindo inferências. Estes procedimentos são dados por regras em tudo similares às que se podem encontrar em sistemas de produção tradicionais, com a particularidade de, neste caso, terem associadas medidas da sua motivação ou pesos.

As regras de um agente cognitivo obedecem ao seguinte formato:

rule(Id-Regra →(Tipo,Nr-Antecedentes,Peso)
:: if Antecedente-1 ... Antecedente-N then Consequente).

- **Id-Regra** - sequência de caracteres usados para identificar cada componente do sistema de inferência ou regra.
- **Tipo** - tipo de dados definido em termos das operações que se podem realizar sobre as regras que constituem o sistema de inferência, cujo domínio é dado pelo conjunto fix, dyn:
  - **fix** - a regra mantém-se inalterável durante o tempo de vida do agente - *regra estática*;
  - **dyn** - a regra pode ser alterada, modificando-se por conseguinte o seu posicionamento relativo no conjunto de regras que constituem o sistema de inferência do agente - *regra dinâmica*.
- **Nr-Antecedentes** - conjunto de termos que dão corpo às condições a serem verificadas para que a regra possa ser aplicada.

- **Peso** - valor que, associado a cada regra, permite a reorganização do sistema de inferência do agente, sendo recalculado sempre que um dos antecedentes da regra se verificar.
- **Antecedente-1 .. Antecedente-N** - conjunto de termos que materializam o corpo de antecedentes da regra.
- **Consequente** - conjunto de termos que materializam o corpo de consequentes da regra.

Os agentes cognitivos reformulam os seus sistemas de inferência de acordo com o tipo de intervenção que tiveram no passado, definindo, assim, novas formas de comportamento.

O sistema de aprendizagem não é mais do que um sistema de reescrita que, a partir das medidas de desempenho das regras, altera o sistema de inferência do agente consoante essas medidas. Isto é, a aplicação do sistema de reescrita à base de conhecimento do agente gera uma nova ordenação das regras de tipo dinâmico, alterando, consequentemente, o comportamento do agente.

O sistema de inferência de um agente opera através da repetição das operações: quaisquer regras cujas condições sejam satisfeitas são pré-seleccionadas, uma é então eleita para execução (dependendo do seu grau de motivação) e as acções que corporiza são desencadeadas. Complementarmente, é actualizada ao nível da base de conhecimento do agente uma estrutura de dados temporária na forma *infa(Regra,Nr- Antecedentes-Validados)*, o conjunto das quais corporiza a extensão de um predicado, o *infa*. Desta forma pode-se conhecer, para cada regra despoletada, o número de antecedentes validados, dados pelo atributo (*Nr-Antecedentes-Validados*). É então possível

decidir acerca da activação ou não do sistema de aprendizagem, por forma a reescreverem-se as regras que fazem o sistema de inferência do agente, com base em critérios predefinidos e constantes da base de conhecimento do agente cognitivo. Estes critérios são definidos *a priori*, de acordo quer com o tipo das potenciais aplicações em que o agente poderá ser chamado a intervir quer com a sua sazonalidade. Esta restrição à aplicação dos mecanismos de reescrita justifica-se, quanto mais não seja, por uma questão de bom senso, uma vez que, para um certo corpo de conhecimento, e tendo os agentes uma postura racional, é de esperar um comportamento relativamente uniforme das regras que compõem o respectivo sistema de inferência.

Em caso de aplicação do sistema de reescrita, os novos pesos das regras são calculados através da seguinte expressão:

$$NovoPeso = Peso + Nr\text{-}Ant\text{-}Validados \;/\; Nr\text{-}Antecedentes.$$

Este processo pode então ser redefinido através das seguintes fases:

- **Análise**.
  Análise das cadeias de inferência criadas no decurso da resolução de um problema, por forma a potenciar o cálculo do novo peso das regras que constituem o sistema de inferência do agente.
- **Selecção RD** *versus* **Selecção RF**.
  Compatibilização das regras de natureza estática com as regras de natureza dinâmica, no sentido de se estabelecerem diferentes estratégias para a resolução de um problema.
- **Reestruturação**.

Reestruturação da base de conhecimento do agente cognitivo de acordo com as diferentes estratégias estabelecidas na fase anterior.

```
procedimento Aprendizagem-Agentes-Cognitivos
    obter-parâmetros-aprendizagem(Período,Condições-BC-Aprendizagem);
    verificar-estado-agente(Ciclo,Condições-MT-Aprendizagem)
    se (Ciclo ∈ Período) ∧
        (Condições-MT-Aprendizagem ≡ Condições-BC-Aprendizagem) então
            construir-lista-informação-controlo(Lista-Regras-Seleccionadas)
            para cada (Regra,Nr-Antecendentes-Validados)
                        ∈ Lista-Regras-Seleccionadas
                até Lista-Regras-Seleccionadas = [ ]
                    fazer
                        calcular-novo-peso(Regra,Nr-Antecendentes-Validados,
                            Novo-Peso);
                        reescrever-regra(Regra,Novo-Peso);
            fim-para
            executar-selecção-RD(Lista-Regras-Tipo-dyn);
            executar-selecção-RF(Lista-Regras-Tipo-fix);
            ordenar-estratégia-parcial(Lista-Regras-Tipo-dyn,
                    Nova-Lista-Regras-Tipo-dyn);
            definir-nova-estratégia(Lista-Regras-Tipo-fix,
                    Nova-Lista-Regras-Tipo-dyn);
            reestruturar-base-conhecimento(Nova-Lista-Regras-Tipo-dyn,
                    Lista-Regras-Tipo-fix);
            reportar-acções-realizadas;
    fim-se
fim-procedimento.
```

Figura 5.17: Aprendizagem dos Agentes Cognitivos.

A Figura 5.17 apresenta uma breve descrição do processo de aprendizagem dos agentes cognitivos [11].

[11] O algoritmo descreve as principais acções desenvolvidas pelos agentes cognitivos durante a realização de um processo de aprendizagem. Este está dividido nas seguintes fases: 1) inicialização - na qual o agente cognitivo recolhe na sua base de conhecimento os parâmetros (*Condições-BC-Aprendizagem*) que determinam a activação ou não dos mecanismos de aprendizagem (*obter-parâmetros-aprendizagem/2*) e consulta na sua memória de trabalho o estado corrente dos parâmetros de inferência (*Condições-MT-Aprendizagem*) do agente cognitivo (*verificar-estado-agente/2*); 2) preparação - na qual o agente deter-

mina se as condições de activação (*(Ciclo ∈ Período) ∧ (Condições-MT-Aprendizagem ≡ Condições-BC-Aprendizagem)*) dos seus mecanismos de aprendizagem estão satisfeitas; se isso se verificar, o agente constrói uma lista com informação relativa às regras que foram seleccionadas no último ciclo de inferência (*construir-lista-informação-controlo/1*) e passa à fase seguinte; caso contrário, abandona o processo de aprendizagem; 3) cálculo - para cada uma das regras que foram seleccionadas no último ciclo de inferência, o agente calcula um novo peso (*calcular-novo-peso/3*), com base no número de antecedentes validados (*Nr-Antecendentes-Validados*), reescrevendo a regra na base de conhecimento (*reescrever-regra/2*); 4) reestruturação - após o cálculo dos novos pesos das regras seleccionadas, o agente procede à reestruturação da base de conhecimento: selecciona as regras do tipo *fix* (*executar-selecção-RD/1*) e *dyn* (*executar-selecção-RF/1*), ordena as regras do tipo *dyn* por ordem decrescente dos seus pesos (*ordenar-estratégia-parcial/2*) e reescreve a parte da base de conhecimento do agente relativa às regras, colocando as do tipo *fix* em primeiro lugar, seguidas das do tipo *dyn* (*reestruturar-base-conhecimento/2*); por fim, o agente apresenta no seu monitor local todas as acções que realizou (*reportar-acções-realizadas/0*).

Capítulo 6

Multicomunidades de Agentes

Extensão do modelo do sistema para um ambiente multi-comunidade de agentes. Caracterização e descrição das funcionalidades dos agentes da classe dos encaminhadores. Análise dos modelos de comunicação e de cooperação intercomunidades.

Quando se analisa a vida em comunidade, é possível detectar entre os seus elementos padrões de comportamento que, de uma forma directa ou indirecta, condicionam as suas acções. Uma universidade ou uma empresa são exemplos de comunidades cujos membros se relacionam tendo em vista atingir objectivos comuns.

A forma como os indivíduos de uma comunidade actuam e reagem face a acontecimentos que podem ocorrer no seu universo é indicadora do seu desenvolvimento tecnológico e cultural. As comunidades regem-se por normas, modelos de comportamento e laços de afinidade entre os seus membros - embora nem sempre explicitamente definidos - que nos revelam uma cultura própria, desenvolvida e aperfeiçoada ao longo do tempo. Assim, é de esperar

que, no seio de uma comunidade, as acções desenvolvidas pelos seus membros, com as respectivas trocas de informação e distribuição de tarefas, sejam reconhecidas como específicas dessa mesma comunidade. Este enquadramento não é com certeza suficiente para caraterizar adequadamente o comportamento de seres racionais, quando agregados numa comunidade. Porém, alguns ensinamentos podem ser importados e explorados nos domínios da IAD e, em particular, nos SM.

Ao estabelecer-se uma relação entre o modelo anterior e o de uma eventual comunidade de agentes cognitivos, constata-se que muitos dos princípios expostos podem ser aplicados a uma comunidade artificial de agentes. Nos SM, o conceito de comunidade pode ser utilizado para representar um grupo de agentes cujo comportamento e objectivos têm a ver com o tipo de tarefas que são chamados a executar. Assim, os agentes não deverão ser agrupados com base num eventual padrão de comportamento, o que seria extremamente difícil de realizar, mas de acordo com a sua área de actuação, o que dará origem à criação de múltiplos ambientes autónomos, onde os agentes se integrarão de acordo com o seu domínio de actuação.

6.1 Ambientes Comunitários

O ambiente no qual os indivíduos de uma comunidade estão integrados, se movimentam e desenvolvem as suas actividades é designado como espaço de uma comunidade. É um espaço para discussão, troca de ideias e resolução de problemas. Pode ser aberto ou fechado, isto é, acessível ou não a agentes externos à comunidade.

A acessibilidade do ambiente de uma comunidade pode ser analisada segundo

duas perspectivas. Na primeira há partilha das estruturas de conhecimento que circulam no ambiente da comunidade. Neste caso, as acções desenvolvidas internamente pelos agentes e o conhecimento partilhado entre estes podem ser "capturados" por qualquer agente externo, desde que para isso lhe seja concedida permissão. Na segunda, um pouco menos crítica que a anterior, há consulta directa aos agentes da comunidade por outros pertencentes a comunidades externas. Um ambiente fechado caracteriza uma comunidade onde não entra nem sai qualquer tipo de informação, não sendo permitida qualquer espécie de consulta aos seus agentes por parte de qualquer entidade externa. Toda a informação que aí circula é considerada confidencial e restrita aos membros da comunidade.

Os casos extremos, isto é, um ambiente completamente aberto ou completamente fechado, não são recomendáveis para aplicação de um modelo multicomunidade, já que se integram, naturalmente, num modelo de um único ambiente.

Da mesma forma que um agente pode possuir conhecimento relacionado com os agentes internos da sua comunidade, pode também possuir conhecimento acerca de agentes externos. A diferença reside em que, na primeira situação, o pedido de ajuda a uma agente interno é directo, enquanto na segunda são necessários meios suplementares para aceder ao ambiente onde o agente externo se encontra, nomeadamente protocolos especiais e permissões de acesso ao exterior e ao ambiente externo.

Não restringindo o espaço de actuação de uma comunidade de agentes é possível conceber uma situação, envolvendo um problema multidisciplinar, na qual um agente sabe de quem, e de que forma, pode receber ajuda de fora da sua comunidade. Devem, assim, ser criados mecanismos de suporte

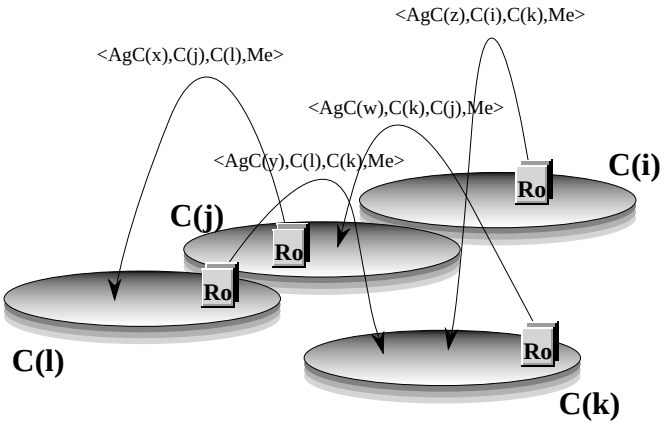


Figura 6.1: Modelo de Comunicação Intercomunidades.

à comunicação intercomunidades que, disponibilizando recursos, assegurem os processos de comunicação e cooperação entre agentes. A Figura 6.1 ilustra esse processo a partir de quatro comunidades diferentes. As estruturas da forma $< AgC(x), C(y), C(z), Me >$, que figuram nas ligações intercomunidades, materializam esse processo, desencadeado por um agente especial (Ro) com funções de encaminhamento, requerido por um agente cognitivo $AgC(x)$, entre duas comunidades $C(y)$ e $C(z)$, com o objectivo de enviar a mensagem Me.

6.2 Multicomunidades de Agentes

A implementação de um modelo computacional capaz de emular o comportamento de agentes pertencentes a diferentes comunidades passa, naturalmente, pelo estudo e análise de várias questões, nomeadamente:

- que tipo de estruturas de dados e mecanismos computacionais é necessário para suportar estes processos de comunicação;
- quais as políticas de segurança a adoptar e a que nível deverão ser aplicadas;
- como direccionar correcta e eficazmente as mensagens dos agentes internos para os agentes externos.

Tal como foi definida anteriormente, uma comunidade é constituída por um conjunto de agentes autónomos que possuem interesses e objectivos idênticos e partilham um mesmo ambiente. Estes agentes podem apresentar diferentes tipos de perícia e de conhecimento. Como entidades autónomas e "inteligentes" que são, podem actuar autonomamente, porém, quando os problemas a resolver são de natureza multidisciplinar, a existência de grupos de trabalho a cooperarem entre si pode ser a melhor forma de organização. A resolução de problemas é conseguida através da troca de informação, da partilha de resultados ou do diálogo desenvolvido entre os diferentes grupos de trabalho. Desta forma serão criadas as bases para a geração de uma comunidade global de comunidades de agentes.

Os termos *microcooperação* e *macrocooperação* [1] serão utilizados por forma a representarem, respectivamente, os processos de cooperação desenvolvidos pelos agentes cognitivos dentro da sua própria comunidade e os desenvolvidos entre os agentes cognitivos pertencentes a diferentes comunidades.

[1] A aplicação destas expressões pode ser entendida da mesma forma que os termos *micro* e *macro* utilizados nas Ciências Económicas.

6.2.1 O Modelo Monocomunidade

Cada comunidade tem associado um ambiente de trabalho específico, utilizado pelos seus agentes como meio de comunicação e de partilha de informação. O sistema BEABLE disponibiliza, no seu modelo base (Capítulo 4), os meios e estruturas necessárias para assegurar a integração dos diversos agentes cognitivos de uma mesma comunidade, assim como os meios para que os utilizadores possam consultar esses especialistas. Complementarmente, os agentes cognitivos que estão integrados no sistema podem desenvolver entre si processos de cooperação, sempre que o problema em questão for de natureza multidisciplinar, ou quando a perícia e o conhecimento de um agente for insuficiente para a resolução do problema.

A criação de um SM ou, se preferirmos, de uma comunidade de agentes cooperativos tem sempre associado um domínio particular do conhecimento. Os agentes que integram o sistema são motivados para desenvolverem o seu próprio espírito de corpo, por forma a resolverem eficientemente os problemas que porventura lhes forem colocados. Todavia, não raras vezes, ocorrem situações que ultrapassam a área[2] de conhecimento do sistema multiagente.

Suponhamos que o sistema em causa lida com problemas relacionados com o diagnóstico médico e terapias associadas. Pode suceder que, durante a resolução de um problema nessa área, surja a necessidade de consultar um ou vários especialistas no domínio das Ciências Jurídicas, a fim de verificar se a aplicação de determinado procedimento médico é susceptível de vir a infringir as leis em vigor. Um procedimento perfeitamente natural seria o de consultar uma empresa de juristas com vista a obter esse tipo de acon-

[2] A área de conhecimento de um sistema multiagente é a "reunião" dos diversos domínios de conhecimento dos agentes cognitivos integrados na comunidade.

selhamento. Podemos encarar essa empresa como um segundo SM, no qual os agentes cognitivos seriam os juristas. Temos, assim, duas comunidades de agentes: uma especializada no diagnóstico de um conjunto de doenças e outra especializada na aplicação de procedimentos legais.

A possibilidade de os diversos agentes de uma comunidade cooperarem entre si (microcooperação) parece naturalmente necessária e vital para um bom julgamento e resolução do problema. Por outro lado, é natural que uma comunidade não tenha a capacidade de resolver problemas que não se enquadrem na área de conhecimento, requerendo, então, ajuda a uma comunidade externa (macrocooperação).

Sucede, por vezes, que durante a resolução de um problema os especialistas são confrontados com situações com que nunca lidaram, ou que não foram, até ao momento, exploradas ou analisadas convenientemente. Nessas situações, podem consultar outros especialistas da mesma área e da mesma comunidade. Pode suceder que as áreas abrangidas nessas situações não sejam do domínio dos especialistas ou dos seus pares, saindo, segundo o modelo monocomunidade apresentado, da esfera de actuação da comunidade. Qual deverá ser a atitute do especialista perante uma situação como esta?

O modelo multicomunidade, ao agrupar os especialistas por áreas de conhecimento, gerando comunidades altamente especializadas, torna possível a criação do necessário enquadramento para que se possa desenvolver a cooperação entre especialistas de diferentes comunidades, surgindo como uma possível resposta à questão formulada em epígrafe.

6.2.2 O Modelo Multicomunidade

O modelo multicomunidade do sistema BEABLE (O.Belo & J.Neves 1997*a*) permite que vários SM (vistos como comunidades autónomas de agentes) possam comunicar entre si e desenvolver trocas de informação e formas de cooperação. A cada uma destas comunidades será alocado um ambiente próprio, obtido a partir de uma réplica do agente de supervisão do sistema BEABLE.

Cada comunidade tem um identificador. O agente de supervisão do sistema, com base nesse identificador, cria para essa comunidade um ambiente específico, ao qual associa um endereço lógico único. O funcionamento de cada uma destas réplicas do agente de supervisão (uma por comunidade) é garantido e controlado por um administrador, que define as regras de interacção e os parâmetros de funcionamento de cada uma das comunidades envolvidas. O processo de instalação e arranque de cada comunidade segue em tudo os princípios relativos à administração e controlo do sistema apresentado anteriormente (Capítulo 4.8).

Os endereços lógicos gerados para cada comunidade, de acordo com processos Linda, encontram-se armazenados num único ficheiro - o ficheiro de endereços das comunidades, com endereço próprio, em máquina própria. Sempre que uma comunidade é activada ou desactivada, o ficheiro de endereços é actualizado (Tabela 6.1).

Por forma a possibilitar a comunicação e troca de informação entre ambientes, foi criada uma nova classe de agentes para ser integrada no ambiente do sistema BEABLE: os agentes encaminhadores. A sua principal tarefa consiste em recolher, no ambiente da comunidade onde estão inseridos, os pedidos dos

Designação da Comunidade	Endereço Lógico
alfa	venus:1291
beta	diana:1292
omega	andromeda:1293
(...)	(...)

Tabela 6.1: Exemplos de Endereços de Comunidades.

agentes cognitivos endereçados a agentes exteriores. Os agentes encaminhadores actuam tipicamente como "funcionários aduaneiros", supervisionando o tráfego da informação entre comunidades e mantendo operacionais os meios e estruturas de comunicação; são uma espécie de "ponte virtual" entre comunidades com controlo de "fronteira".

A Figura 6.2 apresenta um exemplo de um ambiente multicomunidade. As designações *Alfa, Beta, Gama*, etc., designam os ambientes das comunidades. As linhas a tracejado correspondem a eventuais "pontes" de comunicação, estabelecidas pelos agentes encaminhadores (Ro) das comunidades cujos ambientes se encontram interligados. Cada uma dessas ligações significa que o agente encaminhador da comunidade estabeleceu comunicação com uma segunda comunidade, por forma a poder executar uma operação. Estas ligações apenas se mantêm activas durante o tempo em que o agente encaminhador de uma comunidade estiver a enviar mensagens para o ambiente de outra comunidade.

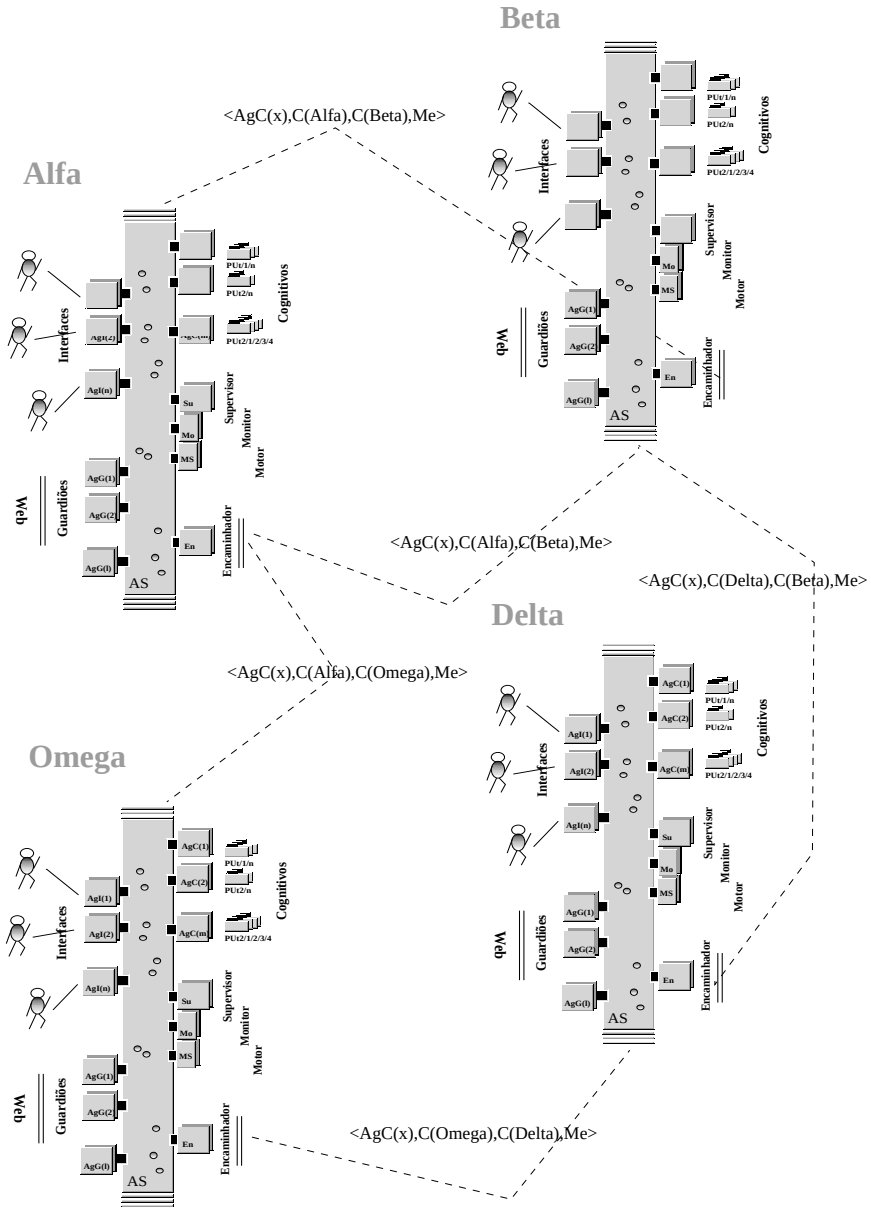


Figura 6.2: Modelo do Sistema Multicomunidade.

Uma comunidade de agentes é organizada de acordo com uma área de aplicação, a que se associam potenciais utilizadores. Os agentes são seleccionados tendo em atenção a sua perícia e conhecimento, e integrados no ambiente da

comunidade sempre que se verifique serem benéficos para esta.

O acesso dos agentes de uma comunidade a um ambiente externo é condicionado pelo seu perfil, pelo supervisor de cada agente ou pelo administrador da comunidade. Os agentes encaminhadores, sempre que há um pedido para consulta externa, deverão respeitar essas normas.

Agente	Área de Aplicação	Accesso	Ajuda
AgM-A01	Classificação de Leveduras	A	F
AgM-B04	Controlo de Ar Condicionado	A	F
AgM-C06	Febre	F	F
(...)	(...)	(...)	(...)

Tabela 6.2: Tipos de Permissão de Acesso.

As restrições de acesso de um agente a uma comunidade podem ser definidas ao nível do agente ou da comunidade. No primeiro caso indica-se qual o agente que pode aceder à comunidade e o tipo de acesso ou ajuda - (A) aberto e (F) fechado - (tabela 6.2). No segundo caso indica-se qual a comunidade externa que tem acesso. A eventualidade de não haver referência a um agente ou comunidade em particular significa que a estes está negado qualquer tipo de acesso. Uma comunidade fechada é, pois, aquela cujo ficheiro de permissões é o ficheiro vazio.

A cooperação entre os agentes especialistas de diferentes comunidades - macrocooperação - pode ocorrer de três formas distintas:

- solicitando-se a um agente externo a execução de uma tarefa em particular;
- solicitando-se a um agente externo informação ou dados que se julgam

pertinentes para a resolução de um problema;

- enviando-se a descrição de um problema a uma comunidade, a fim de que esta defina propostas para a sua resolução, através dos seus agentes cognitivos.

Este tipo de cooperação intercomunidades levanta uma série de problemas relacionados com questões de segurança e privacidade da informação. Em particular, há que actuar em termos de autenticação de mensagens e de análise de permissões. Nos pedidos de ajuda a agentes especialistas exteriores, há que mencionar a fonte, o tipo de solicitação externa, os parâmetros de cada tipo de serviço solicitado e qual o ambiente externo (o destino das mensagens). Desta forma, e através da análise do perfil do agente que solicitou o serviço, é possível conhecer o tipo de operações que podem realizar. Para tornar possível o encaminhamento da comunicação intercomunidades, deverá ser mantido pelos agentes locais um registo das comunicações realizadas. Tal estratégia é de grande utilidade em caso de eventuais falhas do sistema, dada a possibilidade de recuperação deste, a partir de estados de conhecimento anteriores.

A estratégia de distribuição e a dinâmica do ambiente para a implementação de multicomunidades de agentes podem ser comparadas a uma rede de sistemas computacionais em que a filosofia ponto-a-ponto[3] foi implementada[4]. Basicamente, os nodos definidos num ambiente multicomunidade agem como sistemas servidores convencionais, interligados através de uma rede de computadores, onde todos podem ser clientes/servidores em relação aos restantes,

[3]"Peer to peer" na terminologia inglesa.

[4]De referir que a cada ponto do ambiente multicomunidades corresponde uma réplica da versão monocomunidade do sistema BEABLE, que por si só é, também, um ambiente distribuído.

sendo as regras de comunicação definidas pelo administrador do sistema.

6.3 Os Agentes Encaminhadores

A comunicação e a troca de informação entre diferentes comunidades são suportadas pelo agente encaminhador de cada comunidade[5](O.Belo & J.Neves 1997*b*). Este agente possui os meios e a capacidade para interligar o ambiente da sua comunidade aos ambientes de outras comunidades, física e logicamente distintos. O agente encaminhador é essencial para assegurar e suportar a cooperação entre comunidades. Cada comunidade possui um único endereço lógico. Os encaminhadores apenas dirigem para a respectiva comunidade o tráfego de mensagens relacionado com o exterior. As comunicações externas que supervisionam e suportam não afectam as comunicações ou as trocas de informação que ocorram dentro dessa comunidade.

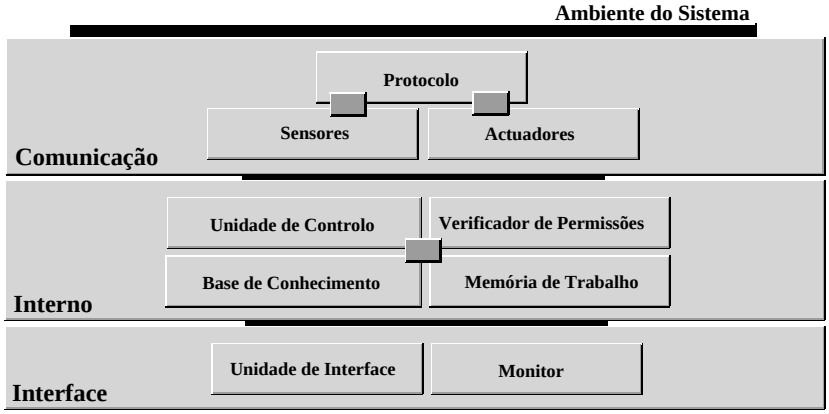


Figura 6.3: Arquitectura Funcional dos Agentes Encaminhadores.

[5]Cada comunidade apenas possui um agente encaminhador. Todavia, poder-se-ão instalar mais destes agentes, através de um processo de replicação, no caso de surgirem situações de contenção ou de estrangulamento originadas por um volume muito grande de pedidos ao exterior.

6.3.1 Arquitectura Funcional

A arquitectura funcional dos agentes encaminhadores (Figura 6.3) é constituída pelos seguintes módulos funcionais:

- **Comunicação**.

  Módulo que assegura as comunicações com o ambiente da comunidade. É constituído pelos seguintes blocos funcionais:

  - **Protocolo**.

    Mecanismos que estabelecem e garantem a comunicação entre o agente encaminhador e o ambiente do sistema.

  - **Sensores**.

    Mecanismos que analisam o tráfego de estruturas de conhecimento no ambiente do sistema, procurando identificar eventuais mensagens endereçadas ao agente encaminhador.

  - **Actuadores**.

    Mecanismos que colocam no ambiente do sistema da comunidade em causa o reflexo das acções do agente encaminhador resultantes do tratamento das mensagens dos agentes da comunidade ou das mensagens enviadas por outros agentes encaminhadores, externos à comunidade.

- **Interno**.

  Módulo do agente encaminhador que controla as comunicações com o exterior. É o módulo responsável pelas tomadas de decisão deste agente, incluindo quatro blocos funcionais:

  - **Unidade de Controlo**.

    Mecanismos que fazem a interpretação das mensagens enviadas pelos agentes internos e pelos agentes encaminhadores exteriores. Aplicam os critérios de decisão por forma a assegurar o tratamento adequado dos pedidos dos agentes especialistas.

  - **Verificador de Permissões**.

    Mecanismos que verificam as permissões dos agentes cognitivos que pediram apoio externo, ou dos agentes encaminhadores que tentam estabelecer diálogo com a comunidade local.

  - **Memória de Trabalho**.

    Área da memória que armazena os dados intermédios e finais relativos aos processos de encaminhamento.

  - **Base de Conhecimento**.

    Repositório da informação relacionada com as normas de acesso, formatos de mensagens e informação necessária às acções de controlo desenvolvidas pelo agente encaminhador.

- **Interface**.

  Módulo do agente encaminhador que permite a ligação ao seu administrador, integrando os seguintes blocos funcionais:

  - **Unidade de Interface**.

    Mecanismos que possibilitam o diálogo entre o agente encaminhador e o seu administrador.

  - **Monitor**.

    Mecanismos que reportam as acções ocorridas no ambiente do agente encaminhador.

Na Figura 6.4 apresenta-se um excerto de uma peça de código em Prolog do sistema de comunicação de um agente encaminhador, relativa ao arranque e preparação do ambiente local deste.

6.3.2 Dinâmica e Funcionalidades

Um agente cognitivo que esteja activo no sistema tem, em princípio, sob o seu controlo um ou mais pedidos de consulta por parte de utilizadores do sistema. Ao procurar satisfazer estes pedidos, pode verificar que não possui a perícia nem o conhecimento necessários, defrontando-se assim com uma situação que requer aconselhamento externo (já que não encontra na comunidade local a ajuda necessária) ou, em última análise, o abandono do processo em curso. Contudo, essa não será a melhor forma de ultrapassar este tipo de situação, ainda que seja a mais fácil. Se o agente possuir na sua base de conhecimento informação relativa a agentes cognitivos (internos e externos) com aptidão para lidar com essas situações, pode optar por requerer a sua ajuda através de solicitação directa.

Este tipo de procedimento é comum dentro da mesma comunidade. No entanto, se o problema não puder ser resolvido internamente, haverá que solicitar ajuda a agentes de comunidades externas. O procedimento é equivalente ao de um pedido interno, devendo, contudo, ser analisado, avaliado e autorizado pelo agente encaminhador da comunidade à qual o agente pertence.

Quando o agente encaminhador recebe um pedido da parte de um agente cognitivo para uma consulta externa, deverá tratar esse pedido em termos de:

```
% \\\
% \\\ Arranque e preparação do ambiente do agente de encaminhamento.

router
        :-
        % \\\ Recolha dos parâmetros de identificação da comunidade do
        % \\\ agente de encaminhamento.
        comunidade_id(Id),
        wm(comunidade_co(Co)),
        format("~2nBeAble's Environment ~n",[]),
        format("Agent ROUTER activated~2n",[]),
        format("Community ~s ~2n",[Id]),
        % \\\ Activação do monitor local do agente de encaminhamento.
        monitor,
        % \\\ Reconhecimento e recolha do endereço da comunidade do
        % \\\ agente de encaminhamento.
        rsh_server(ACo),
        name(ACo,Co),
        rec_server(Co,Se),
        lig_cliente(Se),
        (...)
        % \\\ Comunicação à comunidade da activação do agente.
        insere_obj_bb(bb(rou,t_ro_activ)),
        (...)
        % \\\ Importação das permissões dos agentes cognitivos da
        % \\\ comunidade para acesso ao exterior.
        load_permission_files,
        % \\\ Actualização da base de conhecimento do agente de
        % \\\ encaminhamento com a informação do ficheiro de
        % \\\ endereços globais das comunidades externas.
        retrieve_addresses([],LCn),
        update_environment(LCn),
        (...)
        % \\\ Activação dos mecanismos de inferência do agente.
        pd_inference_engine,
        (...)
```

Figura 6.4: Arranque do Agente de Encaminhamento.

- **Autenticação**.

  Conjunto de mecanismos que permitem a identificação da fonte que solicitou o pedido do serviço, a área de especialização do agente, o tipo de serviço solicitado, o ambiente externo da comunidade à qual se destina e a entidade que o analisará e tratará no destino.

- **Verificação de Permissões**.

  Conjunto de mecanismos que permitem a consulta do ficheiro de permissões do agente encaminhador, por forma a verificar se o agente especialista que solicitou o pedido de consulta externa está ou não autorizado a fazer tal solicitação.

- **Localização da Comunidade Externa**.

  Conjunto de mecanismos que possibilitam ao agente encaminhador, através da consulta do ficheiro de localizações das comunidades, verificar se o ambiente da comunidade externa está activo e tentar estabelecer uma ligação para comunicação com esse ambiente.

- **Envio do Pedido**.

  Conjunto de mecanismos que permitem ao agente encaminhador colocar o pedido de serviço do agente cognitivo no ambiente da comunidade externa.

Sempre que um agente cognitivo envia uma mensagem a um agente encaminhador solicitando um serviço externo, este desencadeia o processo de autenticação da mensagem, verifica o nível de permissão desse agente e analisa o seu perfil.

```
procedimento Despacho-de-Pedidos.
   recolher-pedidos-ambiente-agentes-cognitivos(Lista-Pedidos);
   guardar-pedidos-memória-trabalho(Lista-Pedidos)
   com cada mensagem(Agente-Cognitivo,Comunidade-Interna,Comunidade-Externa,
      Agente-Destino,Tipo,Operação) ∈ Lista-Pedidos
      fazer
         Mensagem ← mensagem(Agente-Cognitivo,Comunidade-Interna,
            Comunidade-Externa,Agente-Destino,Tipo,Operação)
         autenticar-avaliar-pedido-externo(Mensagem,Autenticação)
         se Autenticação então
            verificar-permissão-externa(Mensagem,Acesso-Externo)
            se Acesso-Externo então
               verificar-ligação-comunidade-externa(Comunidade-Externa,
                     Conectividade,Endereço)
            fim-se
         fim-se
         se ¬Autenticação ∨ ¬ Acesso-Externo ∨ ¬ Conectividade então
            abandonar-pedido(Mensagem);
            informar-agente-cognitivo(Agente-Cognitivo,Mensagem)
         senão
            estabelecer-ligação-comunidade-externa(Comunidade-Externa,Endereço);
            definir-parâmetros-pedido(Mensagem,Código);
            enviar-pedido-comunidade-externa(Código,Mensagem,Endereço);
            armazenar-localmente-réplica-pedido(Código,Mensagem,Endereço);
            fechar-ligação-comunidade-externa(Comunidade-Externa,Endereço)
         fim-se
         actualizar-logfile(Código,Mensagem,Autenticação,Acesso-Externo,
               Conectividade)
   fim-com
fim-procedimento
```

Figura 6.5: Tratamento e Despacho de Pedidos para o Exterior.

Se a decisão do agente de encaminhamento for a de aceitar o pedido, então estabelece um canal de comunicação com a comunidade externa e despacha o pedido para o ambiente da comunidade referenciada através dos seus actuadores. Caso o pedido seja recusado, por falta de permissão ou por impossibilidade de estabelecer um canal de comunicação, o agente que o formulou é informado do sucedido, sendo a mensagem devolvida.

O agente encaminhador local, ao detectar a resposta ao pedido de serviço externo no ambiente da comunidade local, desenvolve as acções necessárias para a enviar ao agente cognitivo local que a solicitou.

Na Figura 6.5 apresenta-se uma descrição do comportamento do agente encaminhador no tratamento e despacho das mensagens que lhe foram endereçadas pelos agentes cognitivos requerendo serviço externo[6].

Em caso de pedido externo, formulado por um agente cognitivo de uma comunidade exterior, o agente encaminhador local recolhe a mensagem, procedendo à respectiva autenticação, permissões e avaliação, e remete-a ao agente cognitivo referido ou devolve-a ao agente encaminhador externo, no caso de não aceitar o seu pedido de serviço.

Na Figura 6.6 apresenta-se uma descrição das acções realizadas por um agente

[6]O algoritmo descreve sumariamente o comportamento do agente encaminhador no tratamento dos pedidos de consulta externa dos agentes cognitivos locais. Nele podemos identificar as seguintes fases: 1) recolha e armazenamento dos pedidos - o agente encaminhador recolhe do ambiente do sistema as mensagens enviadas pelos agentes cognitivos relativas a pedidos de serviço externo (*recolher-pedidos-ambiente-agentes-cognitivos/1*) e armazena-as localmente na sua memória de trabalho para posterior tratamento (*guardar-pedidos-memória-trabalho/1*); 2) análise dos pedidos - para cada mensagem recolhida, o agente encaminhador autentica o pedido de serviço externo (*autenticar-avaliar-pedido-externo/2*), a fim de identificar e estabelecer o perfil do agente cognitivo requerente, e verifica (*verificar-permissão-externa/2*) as suas permissões de acesso (*Acesso-Externo*) ao exterior; 3) despacho dos pedidos - se o pedido do agente cognitivo passar as etapas de verificação anteriores, o agente encaminhador verifica se a comunidade externa solicitada está activa e acessível (*verificar-ligação-comunidade-externa/3*) através do seu endereço lógico (*Endereço*); no caso de conseguir estabelecer a ligação com o ambiente da comunidade externa (*estabelecer-ligação-comunidade-externa/2*), o agente encaminhador define os parâmetros da mensagem (*definir-parâmetros-pedido/2*), colocando-a num formato legível para os agentes da comunidade externa, envia-a para o ambiente dessa comunidade (*enviar-pedido-comunidade-externa/3*), armazena localmente uma cópia da mensagem (*armazenar-localmente-réplica-pedido/3*) para posterior validação da resposta e fecha a ligação estabelecida (*fechar-ligação-comunidade-externa/2*); se não conseguir comunicar com a comunidade externa, o agente encaminhador abandona o tratamento do pedido (*abandonar-pedido/1*), informando o agente cognitivo requerente do sucedido (*informar-agente-cognitivo/2*); as actividades realizadas pelo agente encaminhador são registadas numa "logfile" (*actualizar-logfile/5*).

encaminhador durante a recepção e o tratamento de mensagens provenientes de outras comunidades[7].

Os agentes da classe dos encaminhadores possuem mecanismos de "time-out" para uma gestão equilibrada dos tempos de resposta, a fim de controlar eventuais pedidos de consulta externa muito demorados. Nessas circunstâncias, o serviço externo é anulado e os agentes cognitivos envolvidos são informados da ocorrência. Pretende-se, desta forma, tornar os processos de encaminhamento o mais expeditos possível. Na realidade, se o agente que pediu um serviço externo é informado de que o seu pedido está muito demorado, porque não explorar formas de acção alternativas? Por outro lado, os agentes da classe dos encaminhadores podem ser encarados também como "gateways" inteligentes. Podem traduzir as mensagens, caso reconheçam o seu formato original, convertendo-as para o seu formato de representação interno e, endereçando-as às comunidades de destino, optimizar os processos de comunicação e actuar como "portas" de acesso aos ambientes dessas comunidades,

[7] O algoritmo descreve o comportamento do agente encaminhador após a recolha de mensagens provenientes de agentes cognitivos pertencentes a outras comunidades endereçadas a agentes locais. Nele podemos identificar essencialmente duas fases: 1) recolha e armazenamento dos pedidos - em que o agente encaminhador recolhe (*recolher-pedidos-ambiente-agentes-cognitivos/1*) do ambiente do sistema e armazena na sua memória de trabalho (*guardar-mensagens-memória-trabalho/1*) as mensagens (*Lista-Mensagens*) enviadas por agentes cognitivos de outras comunidades através de um agente de encaminhamento, e relativas a pedidos de serviço à comunidade local de agentes cognitivos ou a respostas a pedidos externos dos agentes cognitivos locais; 2) análise dos pedidos - em que o agente de encaminhamento procede à análise de cada uma das mensagens que recolheu na fase anterior com o objectivo de fazer a sua autenticação (*autenticar-avaliar-pedido-externo/2*) e a verificação das permissões (*verificar-permissões-acesso/2*) dos agentes cognitivos exteriores que enviaram as mensagens; caso não haja qualquer impedimento para o prosseguimento do tratamento da mensagem recebida, isto é, se o agente externo for reconhecido e tiver permissão para acesso à comunidade local, o agente encaminhador verifica se a mensagem é um pedido de serviço, uma resposta ou uma recusa de um pedido de um agente cognitivo, dando-lhe o seguimento adequado; se o agente encaminhador verificar que a mensagem não deve ser processada, estabelece ligação com a comunidade (*estabelecer-ligação-comunidade-externa/2*) do agente cognitivo que enviou a mensagem e devolve-lha (*devolver-mensagem-comunidade-externa/3*); no final destas operações actualiza a sua história com os dados relativos às acções que realizou (*actualizar-logfile/4*).

assegurando o estado do seu conhecimento e a sua privacidade, de acordo com as normas definidas pelos agentes da comunidade, através dos seus supervisores.

```
procedimento Recepção-Mensagens-Externas.
  recolher-mensagens-ambiente-para-agentes-cognitivos(Lista-Mensagens);
  guardar-mensagens-memória-trabalho(Lista-Mensagens)
  com cada (Código,mensagem(Agente-Externo,Comunidade-Externa,Comunidade-Interna,
    Agente-Cognitivo,Tipo,Operação)) ∈ Lista-Mensagens
    fazer
      Mensagem ← mensagem(Agente-Externo,Comunidade-Externa,
          Agente-Cognitivo,Tipo,Operação);
      autenticar-avaliar-pedido-externo(Mensagem,Autenticação)
      se Autenticação então
        verificar-permissões-acesso(Mensagem,Ajuda-Exterior)
      fim-se
      se ¬ Autenticação ∨ ¬ Ajuda-Exterior então
        estabelecer-ligação-comunidade-externa(Comunidade-Externa,Endereço);
        devolver-mensagem-comunidade-externa(Código,Mensagem,Endereço);
        abandonar-mensagem-externa(Código,Mensagem);
        fechar-ligação-comunidade-externa(Comunidade-Externa,Endereço)
      senão
        processar-código-mensagem(Código,Mensagem);
        identificar-agente-cognitivo-destino(Agente-Cognitivo,Identificação)
        caso Tipo seja
          pedido   : enviar-pedido-agente(Agente- Cognitivo,Mensagem);
          resposta : enviar-resposta-agente(Agente- Cognitivo,Mensagem);
          recusa   : enviar-recusa-agente(Agente- Cognitivo,Mensagem)
        fim-caso
      fim-se
    actualizar-logfile(Código,Mensagem,Autenticação,Ajuda-Exterior)
  fim-com
fim-procedimento
```

Figura 6.6: Recepção e Tratamento de Mensagens

A Figura 6.7 apresenta uma janela com algumas das acções executadas pelo agente encaminhador de uma comunidade.

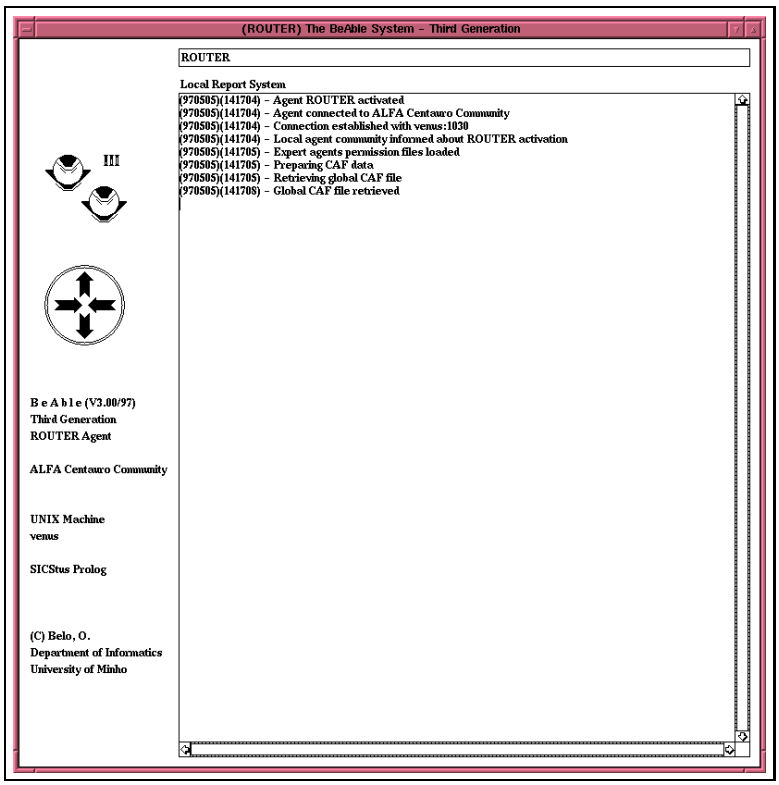


Figura 6.7: Ambiente de Trabalho de um Agente Encaminhador.

Capítulo 7

Aplicações

Análise do comportamento do sistema em domínios do conhecimento relacionados com problemas de classificação, controlo e diagnóstico.

No desenvolvimento do sistema BEABLE esteve sempre presente a ideia de criar um ambiente onde pudessem ser integrados diversos tipos de entidades com diferentes comportamentos, perícias e conhecimento. Essas entidades poderiam ser simples dispositivos de controlo, sistemas de interface ou mesmo sistemas "inteligentes" - vulgo SBC - que, quando agrupados, potenciariam sinergias por forma a alcançar objectivos comuns. Cada um desses grupos poderá ser encarado como dando corpo a aplicações derivadas do sistema.

Este capítulo fornece uma panorâmica da utilização de um ambiente computacional como o do sistema BEABLE, através da construção de quatro plataformas de teste[1] orientadas para a resolução de problemas nas áreas

[1] A designação *plataforma de teste* corresponde ao estudo, planeamento e desenvolvimento de uma aplicação real, com o objectivo de estudar e analisar o comportamento e desempenho do sistema BEABLE.

do diagnóstico, do controlo ou dos sistemas de classificação (Figura 7.1). Cada um dos cenários seleccionado deu origem a um SM, permitindo, em particular, testar diferentes formas de cooperação, negociação e resolução de conflitos.

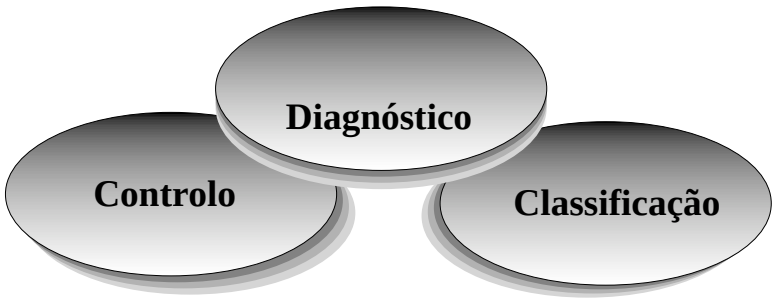


Figura 7.1: Áreas de Aplicação em Estudo.

7.1 Aplicações do Sistema

A utilização do sistema numa área em particular apenas dependerá da disponibilidade de agentes com a perícia e o conhecimento necessários à resolução dos problemas que aí se possam colocar. A actual versão do sistema encontra-se ainda em fase de protótipo, se bem que apresente características e desempenhos aceitáveis para um sistema com aplicabilidade num domínio real.

A decisão de aplicação do sistema a um domínio de conhecimento deve, em particular, ter em consideração os seguintes aspectos:

- há interesse em não vincular os agentes ao ambiente do sistema, dando-lhes a possibilidade de entrar ou sair do sistema quando julgarem necessário ou conveniente;

- alguns dos agentes que integram o sistema comportam-se como entidades inteligentes, enquanto outros dão um mero apoio funcional;
- os agentes necessitam de uma área de trabalho a partir da qual possam desenvolver formas de cooperação com terceiros ou desencadear simples trocas de informação;
- pode haver necessidade de se criarem "nichos" de entidades especializadas - as comunidades do sistema BEABLE no modelo multicomunidade;
- o ambiente deve facilitar o desencadeamento de procedimentos com vista à supervisão, monitorização e controlo das entidades activas no sistema.

A concepção, o desenho e o desenvolvimento do sistema BEABLE foran orientados para áreas de conhecimento que envolvessem processos de classificação, diagnóstico e controlo. O eventual desenvolvimento de um sistema tendo em vista a resolução de um número (restrito) de problemas adstritos a uma área em particular encerra, naturalmente, uma certa dose de risco. Os critérios de desenho e desenvolvimento do sistema acompanharam de perto as questões e características normalmente associadas aos domínios referidos, sendo de salientar, entre outros, aspectos como a definição da linguagem de comunicação entre agentes no sistema e o modelo de representação do conhecimento utilizado nas bases de conhecimento dos agentes cognitivos.

7.2 Plataformas de Teste

As plataformas de teste são aplicações baseadas em SM desenvolvidas com base em casos práticos reais com características muito específicas, sendo usadas

para teste da robustez do sistema, sua adequação à resolução de problemas nas áreas de diagnóstico, classificação e controlo, bem como para creditação das formas de raciocínio, representação de conhecimento, comunicação e cooperação.

Foram desenvolvidas quatro plataformas de teste: uma para um problema de classificação, duas para problemas de controlo e uma última para um problema de diagnóstico e aconselhamento médico:

- **Sistema para a Classificação de Leveduras**.
  Partindo de um plano predefinido de análises e testes, o SM desenvolve os seus próprios processos de análise e classificação de leveduras.
- **Sistema de Controlo e Distribuição de Ar Condicionado**.
  O problema diz respeito à forma adequada de controlar uma comunidade de consumidores de ar condicionado de diferentes tipos, ligados a uma rede de distribuição alimentada por uma única central produtora.
- **Sistema de Controlo e Gestão de uma Linha Especializada de Produtos**.
  Simulação de uma linha de produção para o fabrico de peças e ferramentas em metal duro.
- **Sistema de Diagnóstico e Aconselhamento Médico**.
  SM constituído essencialmente por agentes orientados para o diagnóstico médico em clínica geral, que actuam como uma espécie de junta médica.

Ao pensar-se em estabelecer uma relação de ordem entre as plataformas referidas, como forma de avaliar quer os recursos computacionais envolvidos

quer a eficácia de implementação, o critério de ordenação deverá ter em conta os seguintes factores:

- número de agentes envolvidos;
- modelos de simulação, coordenação e cooperação utilizados;
- esforço de concepção, desenvolvimento, implementação e validação da aplicação.

7.2.1 O Sistema de Classificação de Leveduras

O leque de casos de classificação candidato a estudo, análise e desenvolvimento de SBC é muito rico, vasto e diversificado. Desde a área da Botânica, onde encontramos inúmeras situações para aplicação desta tecnologia, à área da Biologia, o número de casos é, sem sombra de dúvida, considerável.

O sistema para a classificação de leveduras é extremamente motivador e atraente sob o ponto de vista do desenho e desenvolvimento de uma plataforma de teste para o sistema BeAble, já que releva uma área com potencial para o desenvolvimento de um SP de classificação.

O Processo de Classificação de Leveduras

A actual classificação de leveduras (J.Barnett, R.Payne & D.Yarrow 1990) é baseada em estirpes ("strains"). Uma estirpe é constituída por descendentes de um único bloco isolado numa cultura pura, mais concretamente de uma única colónia. Muitas vezes é assumido que, sem evidência, essa colónia deriva de uma única célula. Além disso, é impraticável cultivar certos tipos

de estirpes sem que se formem ascósporos, especialmente quando são isoladas em primeiro lugar. As estirpes, na prática, não dão origem a clones, o que seria ideal, pois estes poderiam ser derivados vegetativamente de uma única célula, podendo ser geneticamente homogéneos. As estirpes são agrupadas em espécies. As espécies com características em comum são organizadas em géneros que, por sua vez, formam famílias. As famílias são agrupadas em ordens, as ordens em classes e as classes em divisões. As características principais tidas em conta nos processos de classificação de leveduras são as seguintes:

- aparência microscópica das células;
- modo de reprodução;
- algumas actividades fisiológicas - especialmente nutricionais;
- algumas características bioquímicas.

Os procedimentos para a identificação das leveduras podem diferir, em princípio, daqueles que são usados para a sua classificação. Enquanto as descrições taxonómicas deverão ser tão completas quanto possível, a natureza e o número de critérios utilizados para a identificação devem depender de atributos apresentados pelas leveduras, do número de identificações a realizar, do propósito para o qual são determinados e das possíveis espécies no nicho em estudo[2].

[2]Para uma melhor compreensão do processo de classificação de leveduras ver J.Barnett et al. (1990) e H.Phaff, M.Miller & E.Mrak (1978).

O Caso em Estudo

A classificação de leveduras obedece a uma "sequência" de testes laboratoriais; trata-se de um processo construtivo, o que o torna, naturalmente, elegível para aplicação da metodologia de desenvolvimento de SP. Podemos justificar tal eleição pelos seguintes motivos:

- O sistema actua basicamente sobre uma matriz de decisão de dimensão variável, aproximadamente com 469×83 células (469 espécies de leveduras e 83 testes), que relacionam os possíveis valores dos testes com as espécies de leveduras conhecidas. Cada célula é susceptível de receber um de vários valores, nomeadamente: positivo, negativo, indiferente, variável, fraco ou atrasado.
- O processo de classificação pode ser reduzido a um processo de procura e redução.
- Os valores que as células da matriz podem conter nem sempre são conhecidos, apresentando estas, não raras vezes, mais do que um valor possível.
- Durante um processo de classificação surge frequentemente a necessidade de dados sobre as leveduras, as cadeias de inferência, possíveis soluções, etc.
- É possível, em certas circunstâncias, face a tendências nos processos de classificação, evitar a morosa tarefa de realizar obrigatoriamente os testes de uma forma exaustiva.
- É possível, com base em testes realizados, comparar os resultados e, a partir destes, conhecer as tendências da classificação.

Uma primeira abordagem do problema pode encontrar-se em P.Camacho (1996): um SP protótipo, centralizado, disponibiliza um conjunto de "ferramentas" para o registo dos testes laboratoriais (J.Barnett et al. 1990). Basicamente, este sistema recolhe os elementos resultantes de cada teste realizado, armazena-os, cataloga-os e, com base nos resultados obtidos, procura redefinir, para cada teste efectuado, o espaço de procura da solução; isto é, tenta reduzir o conjunto das leveduras candidatas à selecção.

A actual plataforma de teste permite uma nova abordagem do processo de classificação de leveduras, já que apresenta as seguintes características:

- possibilidade de tratamento em simultâneo de vários processos de classificação;
- optimização dos mecanismos de raciocínio e reescrita do modelo de representação de conhecimento do anterior SP;
- fornecimento de formas alternativas de armazenamento da informação adstrita aos processos de classificação;
- disponibilização de um conjunto de novas funcionalidades ao nível do acompanhamento do processo de classificação, bem como da análise das cadeias de inferência;
- possibilidade de optimizar os processos de classificação a partir de novos mecanismos de procura baseados em chaves lógicas de redução[3].

[3]Uma *chave lógica de redução* foi a expressão escolhida para caracterizar um conjunto de propriedades que actuarão como filtros num processo de classificação, diminuindo, se possível, com base em tendências e características comuns das leveduras, o número de testes a efectuar, simplificando e, consequentemente, reduzindo o tempo de um processo de classificação.

A realização desta plataforma de teste tinha como objectivos principais os seguintes:

- analisar de uma forma geral a dinâmica de todo o sistema;
- analisar os processos de cooperação entre agentes cognitivos de diferentes tipos - classificação e monitorização;
- verificar a robustez do sistema numa aplicação real;
- demonstrar a viabilidade do sistema em áreas de classificação.

Configuração e Dinâmica do Sistema

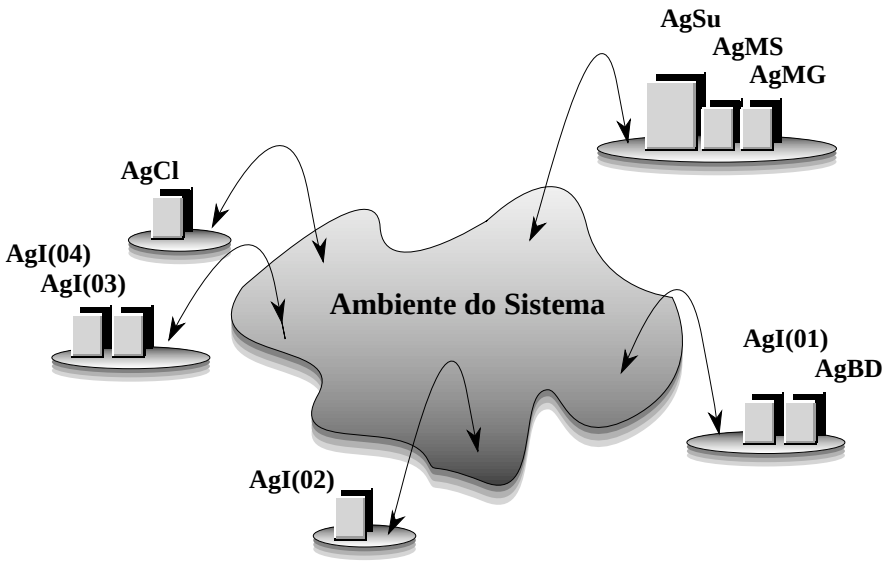


Figura 7.2: Modelo do Sistema de Classificação de Leveduras.

O sistema de classificação de leveduras é a plataforma de teste do sistema mais simples, de menor dimensão e com menor número de agentes, de uma

forma geral. A comunidade de agentes tem como protagonistas os agentes do sistema BeAble, nomeadamente os agentes de supervisão, monitorização e manutenção do ambiente, bem como um número não definido à partida de agentes de interface e dois agentes específicos:

- **Classificador de espécies de leveduras** (AgCL).
  É o agente cognitivo responsável pela análise e acompanhamento dos testes de classificação de leveduras. Complementarmente:
  - possui a capacidade de lidar com mais do que um processo de classificação de cada vez, pertencendo a diferentes utilizadores;
  - actua como um "ajudante" de laboratório, fornecendo informação sobre o estado dos processos de classificação em curso e as características dos testes laboratoriais que pode realizar.
- **Sistema gestor da base de dados do sistema de classificação** (AgBD).
  É o agente responsável pela manutenção da informação (características, valores de testes e imagens) relativa às leveduras que o AgCL pode identificar. Complementarmente:
  - auxilia o agente de classificação nos seus processos, através do fornecimento de dados que ele não possui na sua base de conhecimento local;
  - através de interrogações convencionais, é passível de ser directamente questionado pelos utilizadores do sistema sobre a informação que possui acerca das leveduras.

Os processos de classificação são desencadeados a partir dos agentes de interface, por iniciativa dos utilizadores. Ter-se-ão tantos agentes de interface quantos os utilizadores presentes no sistema. Estes agentes asseguram, via ambiente do sistema, a ligação entre os utilizadores e os agentes de classificação de espécies de leveduras e de gestão da base de dados do sistema de classificação. As arquitecturas funcionais dos agentes que integram esta plataforma de teste acompanham, de uma forma geral, as dos agentes de interface e cognitivos apresentadas anteriormente no Capítulo 5.

7.2.2 O Sistema de Controlo de Ar Condicionado

Na área da simulação, é frenética a procura de novas ferramentas computacionais como forma de dar resposta à crescente complexidade dos problemas que aí se colocam. A combinação de técnicas da IAD e dos SM dá corpo a um projecto orientado para o desenho e desenvolvimento desse tipo de sistemas. A conciliação de múltiplos paradigmas computacionais e a integração de diferentes perícias e fontes de conhecimento são alguns dos caminhos a seguir para se atingir esse objectivo.

O Caso em Estudo

O caso em estudo diz respeito à simulação do funcionamento de uma rede de distribuição de ar condicionado (O.Belo & J.Neves 1995*d*). O problema consiste no controlo de uma comunidade de consumidores ligados a uma rede de distribuição de ar condicionado abastecida por uma única central - a unidade de produção de ar condicionado. Os consumidores são caracterizados pelos seguintes elementos:

- um local, no qual o consumidor se encontra;
- um conjunto de sensores de temperatura;
- um valor médio da temperatura desejável no local;
- um factor de consumo, que caracteriza o tipo de consumidor;
- um tempo, que indica o momento em que o agente de controlo do consumidor deve activar ou desactivar o seu sistema de controlo local.

De acordo com o estado global do sistema e com o estado de cada consumidor, estes regulam os seus pedidos de ar condicionado à unidade de produção central. Esta é condicionada pela sua própria capacidade de produção. Até que a sua capacidade máxima seja atingida, todos os consumidores recebem o fluxo de ar condicionado que solicitam. No caso de a unidade central não possuir a capacidade necessária para satisfazer as solicitações dos consumidores, um processo de negociação é despoletado, a fim de tentar estabelecer um plano de distribuição de ar condicionado que respeite, tanto quanto possível, os pedidos pontuais de cada consumidor. Nesse processo de negociação utilizou-se um modelo baseado em redes de contrato ("contract nets") (R.Smith 1988). Neste processo, um gestor (a unidade de produção central) anuncia aquilo que entende como necessário para optimizar a distribuição. Após uma avaliação do anúncio da unidade central, os consumidores apresentam ao gestor da rede a sua contraproposta. A unidade de produção avalia então as propostas dos consumidores e estabelece um novo plano de distribuição de ar condicionado.

A Figura 7.3 apresenta uma breve descrição do comportamento do agente que controla a unidade central de ar condicionado perante uma situação de

excesso de pedidos de fornecimento[4].

Para a realização desta plataforma de teste foram definidos como objectivos principais os seguintes:

- analisar de uma forma geral a dinâmica do sistema BEABLE;
- testar formas de cooperação entre agentes baseadas na partilha de informação;
- verificar o desempenho do sistema numa aplicação real;
- verificar a viabilidade de aplicação do sistema ao controlo.

Configuração e Dinâmica do Sistema

Um produtor e um grupo de consumidores são os agentes centrais na implementação da rede de distribuição de ar condicionado. De notar que neste

[4] O algoritmo descreve o processo de satisfação de pedidos de ar condicionado à central de produção e, com particular ênfase, a análise da situação em que a central apresenta insuficiente capacidade de produção de ar condicionado relativamente ao número de pedidos dos agentes consumidores. Podem-se identificar três etapas no processo de tratamento de pedidos de ar condicionado: 1) recolha - na qual se procede à recolha no ambiente do sistema dos pedidos (*Pedidos*) enviados à central pelos consumidores (*Agentes*); 2) análise e avaliação - na qual se faz a avaliação da capacidade da central para satisfazer os pedidos dos consumidores; se a central não tiver capacidade para satisfazer os pedidos, ocorre uma situação de sobrecarga (*Sobrecarga*), sendo a central obrigada a estabelecer um novo plano de fornecimento de ar condicionado (*definir-novo-plano-fornecimento/3*) de acordo com o valor de compensação (*Compensação*) alcançado (*compensação-para-balanceamento-da-rede/3*), para que seja possível equilibrar a distribuição de ar condicionado na rede; após a definição do novo plano, os agentes que originaram a sobrecarga são informados (*informar-agentes-sobre-novo-plano/2*), no sentido de reverem as suas necessidades de ar condicionado de acordo com o novo plano estipulado pela central; depois da revisão dos seus pedidos, os consumidores enviam as suas (novas) necessidades (*Propostas*) à central, que reanalisa o plano de distribuição (*ajustar-plano-distribuição/3*) e determina as quantidades finais (*Fornecimentos-Finais*) de ar condicionado a fornecer a cada um dos consumidores; 3) satisfação - os fornecimentos de ar condicionado são distribuídos pelos consumidores (*fornecer-ar-condicionado/1*) de acordo com o plano de distribuição em vigor, e os registos internos da central relativos ao fornecimento de ar condicionado são actualizados.

```
procedimento Análise-da-Produção
    repetir
        recolher-pedidos-de-fornecimento(Pedidos);
        avaliar-pedidos(Pedidos,Sobrecarga)
        se Sobrecarga então
            compensação-para-equilíbrio-da-rede(Pedidos,
                    Sobrecarga,Compensação);
            definir-novo-plano-fornecimento(Pedidos,Compensação,
                    Novo-Plano-Distribuição);
            identificar-agentes-envolvidos(Pedidos,Agentes);
            calcular-fornecimento-por-agente(Pedidos,Agentes,
                    Novo-Plano-Distribuição,Fornecimentos-Possíveis);
            informar-agentes-sobre-novo-plano(Agentes,Fornecimentos-Possíveis);
            recolher-propostas-dos-agentes-para-novo-plano(Propostas);
            validar-propostas(Propostas,Novo-Plano-Distribuição,Ajustes);
            ajustar-plano-distribuição(Novo-Plano-Distribuição,
                    Ajustes,Fornecimentos-Finais);
        senão
            satisfazer-pedidos(Pedidos,Fornecimentos-Finais)
        fim-se
        fornecer-ar-condicionado(Fornecimentos-Finais);
        actualizar-registo-fornecimentos(Fornecimentos-Finais)
    até Fim-Execução
fim-procedimento.
```

Figura 7.3: Descrição do Comportamento do Agente de Controlo da Unidade Central.

grupo não se incluem os agentes de supervisão e controlo do sistema BEABLE. Os consumidores encontram-se divididos em cinco categorias, que correspondem a diferentes níveis de consumo de ar condicionado. Os agentes do sistema de ar condicionado são integrados no ambiente do sistema através das "shells" dos agentes cognitivos.

Cada réplica da "shell", eventualmente localizada num local diferente e numa plataforma computacional distinta, emula um membro específico da comunidade de consumidores ligada à rede de distribuição de ar condicionado, que reage de acordo com o conhecimento e as estruturas de controlo contidas no

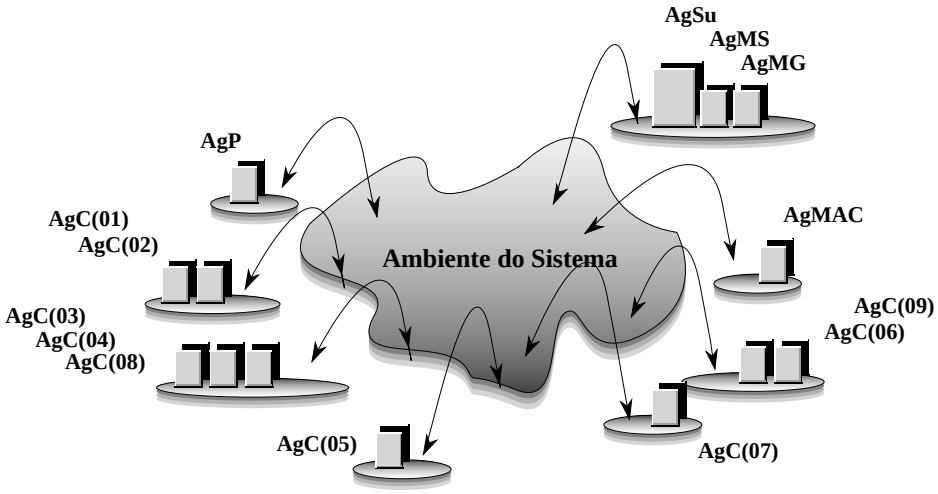


Figura 7.4: Modelo do Sistema de Distribuição de Ar Condicionado.

perfil do agente. Este perfil é criado e mantido localmente pelo administrador delegado do agente de controlo de ar condicionado.

Cada agente de controlo - produtor ou consumidor - acompanha de muito perto a arquitectura de um agente cognitivo, porém, está organizado de forma mais simples, já que o conhecimento e as estruturas de controlo que detém são elementares. Uma "shell" é constituída por três módulos funcionais:

- **Protocolo** - que assegura a comunicação entre o agente e o ambiente do sistema (global).
- **Controlo** - que implementa as acções de supervisão e controlo dos agentes locais e realiza os cálculos das necessidades de ar condicionado, passando à unidade de produção essa informação; simultaneamente, comunica ao supervisor e ao monitor da rede as acções que desencadeou.
- **Monitor Local** - que dá ao administrador do agente a possibilidade

de acompanhar localmente as suas acções.

Foi ainda desenvolvido um agente monitor do sistema de distribuição de ar condicionado, que tem a seu cargo a monitorização da unidade de produção e dos consumidores (Figura 7.5).

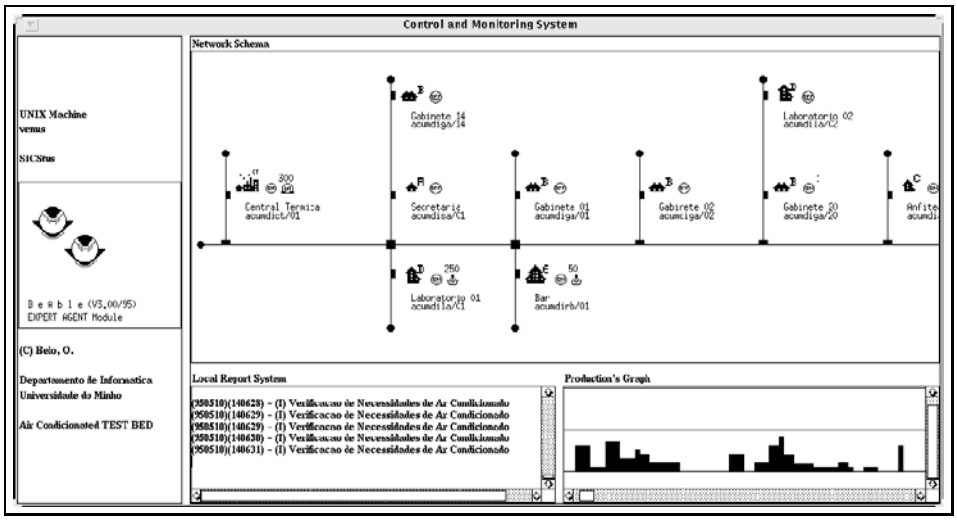


Figura 7.5: Ambiente do Monitor do Agente de Monitorização da Rede de Distribuição de Ar Condicionado.

Nesta plataforma de teste não foi utilizado qualquer tipo de agente de interface. Na verdade, não é necessário contemplar a intervenção directa dos utilizadores do sistema, no sentido de, eventualmente, controlarem a operação do sistema de distribuição de ar condicionado. Este sistema é apenas controlado por agentes cognitivos (produtor + consumidores). O sistema de controlo da rede é autónomo, modular e flexível, uma vez que permite a integração ou desactivação de consumidores de per si.

7.2.3 Um Sistema de Produção Automatizado

É necessário, de algum modo, testar o sistema com aplicações dirigidas à produção de bens e serviços, já que estas interferem com o bem-estar das populações. A qualidade e, consequentemente, a viabilidade comercial dos produtos de uma empresa são largamente influenciadas pela disponibilidade, adequação, qualificação e motivação das equipas de trabalho envolvidas nos processos produtivos. A diversidade das linhas de produtos de uma empresa, assim como a sua inserção no mercado, funcionam como uma forma de certificação da própria empresa. A consecução destes objectivos pode ser facilitada pela introdução de técnicas de IA, e em particular pela utilização de SP (T.Katz 1991).

O significativo incremento da complexidade dos processos produtivos, a exigência de produtos de maior qualidade, a existência de ciclos de produção e de venda cada vez mais curtos são apenas alguns dos factores que apontam para a inadequação de algumas técnicas tradicionais de produção às actuais exigências do mercado. A IA tem vindo a ser aplicada a situações que têm resistido a aproximações convencionais, nomeadamente através do uso de sistemas baseados em redes neuronais, sistemas de classificação e lógicas "fuzzy" (T.Yurtsever 1994), entre outras.

O uso de técnicas desenvolvidas no contexto dos SM facilita, frequentemente, a análise do comportamento dos sistemas de produção. A abordagem modular que um SM providencia, associada à possibilidade de os agentes emularem a perícia e o conhecimento de especialistas humanos, potenciam o desenvolvimento de sistemas de controlo de linhas de produção bastante flexíveis. Por outro lado, as operações de manutenção e reutilização de componentes de

software são igualmente simplicadas.

Os principais benefícios da aplicação de técnicas de IAD a problemas de controlo em aplicações industriais vêm sumariados em T.Wittig, N.Jennings & E.Mamdani (1994) da seguinte forma:

- **modularidade** - induz uma diminuição do grau de complexidade nos componentes de software, o que facilita a sua construção e implementação;
- **desempenho** - a divisão de um sistema em subsistemas potencia o aparecimento de situações de concorrência e paralelismo;
- **fiabilidade** - se um dos subsistemas entrar em quebra, isso não significa que todo o sistema o siga;
- **aquisição de conhecimento** - o processo de aquisição de conhecimento a partir de especialistas do domínio do problema torna-se mais produtivo, pelo simples motivo de se estar a lidar com áreas de conhecimento restritas, em que é de esperar um melhor desempenho por parte dos especialistas no domínio;
- **reutilização** - pequenos sistemas são mais fáceis de replicar e de reutilizar.

O Caso em Estudo

A simulação constitui o meio por excelência para aferir o desempenho de um sistema produtivo, já que permite testar, verificar, analisar, optimizar ou criar novos cenários antes de se partir para a aplicação real (X.Yan 1995)

(M.Akhun & M.Durmusoglu 1994) (S.Goksenand, T.Dayioglu, E.Dorman, S.Imer & A.Kaylan 1994).

O caso em estudo diz respeito à concepção de um SP distribuído para gestão e controlo de um processo de produção de peças e ferramentas em metal duro[5] (O.Belo 1996) (O.Belo 1997). Tendo em atenção que parte da informação a tratar é de natureza confidencial [6], não foram consideradas todas as etapas do processo, subtarefas, matérias-primas e subsidiárias utilizadas, bem como os critérios de decisão directamente relacionados com o tipo de produtos ou técnicas de produção que, directa ou indirectamente, poderiam respeitar aos processos produtivos em causa. O problema prende-se com a simulação do controlo de uma linha automatizada, controlada e gerida por um SM.

O sistema será responsável por gerir e controlar a produção de uma gama de produtos em metal duro, denominada, neste cenário específico de simulação, como "alfa". Após a recepção das encomendas dos clientes, com as especificações das características das peças "alfa" a produzir, pessoal técnico especializado elabora uma ordem de produção, da qual constam as matérias-primas a utilizar, as operações a executar por cada uma das máquinas intervenientes no processo e os padrões de qualidade a ter em atenção, além de eventuais acções a desenvolver caso haja necessidade de recuperar o produto em linha, antes de ser classificado como "não recuperável".

Após a elaboração da ordem de produção, esta é comunicada ao agente encarregado de a lançar em linha, que trata de desenvolver as acções necessárias

[5] Os metais duros, geralmente carbonetos ou carbonitrites sinterizados (ou cimentados), são uma gama de ligas muito duras, refractárias e resistentes ao desgaste, produzidas por técnicas metalúrgicas de "pó" ("powder metallurgical techniques") (K.Brookes 1987).

[6] O modelo de produção idealizado foi inspirado num modelo real de produção de peças e ferramentas em metal duro.

para o processo produtivo arrancar. O processo de produção considerado para o caso em estudo e para o desenvolvimento da plataforma de teste (Figura 7.6) desenvolve-se em nove ou seis fases. Nove fases para as peças que necessitem de um melhor processo de acabamento, seis para aquelas que prescidam desse processo.

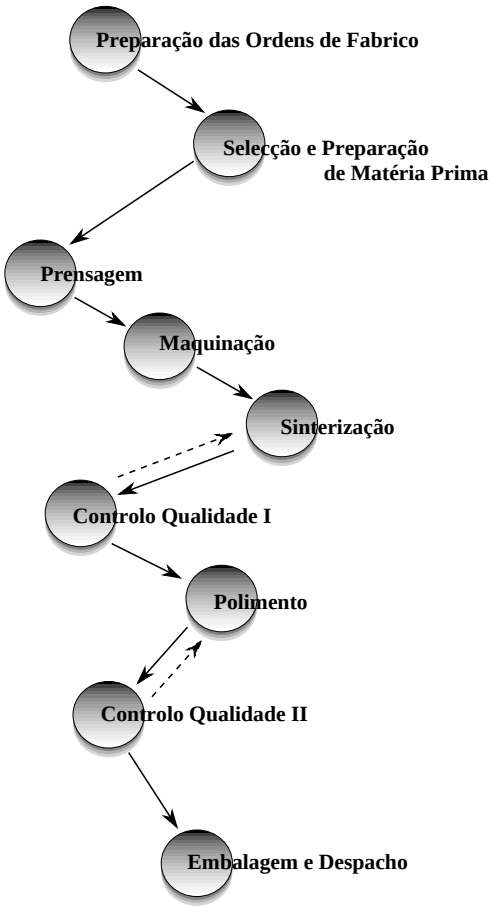


Figura 7.6: Fases Envolvidas na Simulação do Processo Produtivo.

De acordo com o plano de produção apresentado para o fabrico de uma peça de metal duro "alfa", há necessidade de executar diferentes operações em pontos distintos da planta produtiva:

1. Preparar as ordens de fabrico, nas quais se definem as condições de produção das peças.

2. Seleccionar e misturar as matérias-primas necessárias para a fase que se segue, a prensagem. Complementarmente, verificar se a mistura das matérias-primas apresenta as características adequadas para o tipo de peça que vai entrar em produção.

3. Com a mistura de matérias-primas obtida na fase anterior, efectuar a prensagem das peças e, em seguida, verificar o estado das mesmas.

4. Após a recepção das peças prensadas, proceder à sua maquinação, de acordo com as especificações dadas. Eventuais peças deterioradas durante esta fase voltam à fase anterior para serem recicladas.

5. Todas as peças que passaram pela fase de maquinação são sinterizadas, por forma a obterem as características de um metal duro.

6. Depois do processo de sinterização, as peças são controladas laboratorialmente, para detecção de possíveis anomalias. Em resultado desse processo de análise, algumas peças poderão ser ressinterizadas, por forma a recuperar algumas das características desejáveis para as peças após sinterização. As peças que não necessitarem de ser submetidas a outros processos de acabamento seguem para a fase de embalagem, na situação de peças em bruto.

7. A fase de acabamento incide sobre as peças que passaram pelo controlo de qualidade anterior. Nesta fase são executadas, tipicamente, operações de rectificação e de polimento.

8. Depois de polidas e rectificadas, de acordo com os requisitos expressos

na ordem de fabrico, as peças passam por uma segunda fase de controlo de qualidade. Aquelas que se encontrem dentro dos padrões de qualidade estabelecidos para esta nova fase de controlo passam imediatamente para a fase de embalagem, as outras são objecto de processos orientados para a sua recuperação.

9. Segue-se a fase de embalagem das peças e o seu envio aos clientes.

Esta é, em síntese, uma descrição do processo produtivo que foi simulado através do sistema BEABLE. No sistema, as trocas de informação susceptíveis de ocorrer entre os agentes podem representar ordens de fabrico, planos de produção ou as próprias peças em produção. Todas as trocas de informação são feitas através de mensagens específicas enviadas pelo ambiente do sistema e endereçadas a todos os agentes ou a um agente em particular, conforme a situação em causa. O processo produtivo é passível de ser acompanhado através dos monitores dos agentes.

A realização desta plataforma de teste teve como objectivos principais os seguintes:

- analisar globalmente a dinâmica do sistema BEABLE;
- testar formas de coordenação de um sistema de agentes orientados por tarefas;
- analisar o desempenho dos mecanismos de controlo do sistema em situações de "deadlock";
- testar e analisar o comportamento do sistema em situações de falta de recursos;

- verificar o desempenho do sistema numa aplicação real;
- demonstrar a viabilidade de aplicação do sistema ao controlo e, em particular, ao caso do problema em questão.

Configuração e Dinâmica do Sistema

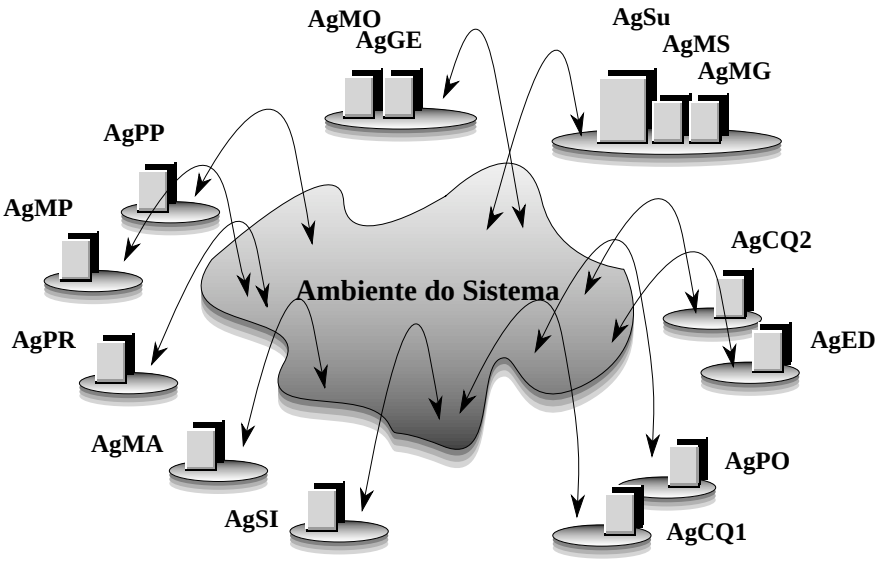


Figura 7.7: Modelo do Sistema de Controlo de Produção.

Além dos agentes de base, para controlo e manutenção do sistema BEABLE e respectivo ambiente, o sistema de produção compreende ainda outro conjunto de agentes cognitivos, que emulam cada uma das fases de produção enunciadas na secção anterior, nomeadamente:

- **AgPP** - *Preparação do Processo.*

  Recebe a ordem de fabrico de um operador e, em sintonia com os outros agentes envolvidos no processo de produção, estabelece um plano

de acção. Complementarmente, verifica se todos os agentes intervenientes nesse processo de fabrico estão operacionais. Após a realização deste processo de verificação, e na situação de se registarem todas as condições necessárias ao processo de fabrico, este agente distribui a cada um dos outros agentes envolvidos o seu plano de trabalho para a ordem de fabrico em causa.

- **AgMP** - *Selecção e Preparação de Matéria-Prima.*
  Após a recepção da ordem de fabrico proveniente do agente AgPP, o agente AgMP analisa-a e prepara as quantidades de matérias-primas necessárias, procedendo à sua mistura e preparação final. O agente AgMP envia-a de seguida ao agente de prensagem (AgPR).

- **AgPR** - *Prensagem.*
  De acordo com a ordem de fabrico enviada pelo agente AgPP, o agente AgPR selecciona o molde e inicia a prensagem das peças. Eventuais peças danificadas durante as operações de manuseamento são colocadas de lado para posterior recuperação.

- **AgMA** - *Maquinação.*
  Nas ordens de fabrico consta o desenho da peça a produzir. Por vezes, a peça não é obtida directamente na fase de prensagem, sendo necessário maquiná-la. O AgMA interpreta as instruções de maquinação enviadas previamente pelo AgPP e controla a máquina durante a operação de maquinação.

- **AgSI** - *Sinterização.*
  Este agente controla o processo de sinterização, ajustando, quando necessário, os elementos que condicionam o processo.

- **AgCQ1** - *Controlo de Qualidade I.*

  As peças, ao alcançarem este ponto do processo de produção, passam por um conjunto de testes, definido previamente para cada modelo de peças "alfa". Se não tiverem fase de acabamento, o agente AgCQ1 remete-as directamente para o agente responsável pela embalagem e envio aos clientes. Caso contrário, passa-as para o agente responsável pelo polimento.

- **AgPO** - *Polimento.*

  Existem peças que não seguem para os clientes em bruto, precisando, por isso, de acabamento. Isto é, passam por um processo de polimento ou de rectificação que garantam as especificações da ordem de fabrico e os padrões de qualidade requeridos. As acções de polimento são uma das várias acções de acabamento que se podem executar sobre as peças. O agente AgPO controla a acção da máquina durante a fase de polimento das peças.

- **AgCQ2** - *Controlo de Qualidade II.*

  O tipo de acções que este agente executa é muito semelhante às realizadas pelo agente AgCQ1, envolvendo, porém, outro género de testes.

- **AgED** - *Embalagem e Despacho.*

  Com base nas instruções remetidas pelo agente AgPP, de acordo com as peças enviadas pelo agente de controlo de qualidade II e segundo o plano de produção, o agente de despacho faz um relatório da situação ao monitor. Se as condições de despacho para o programa de produção em causa estiverem satisfeitas, embala as peças e despacha-as para o cliente. No caso de não verificar as condições de despacho adequadas, este agente faz um relatório no qual apresenta os motivos da situação

ocorrida e aguarda ordens superiores.

- **AgMO** - *Monitor de Processos.*

  Disponibiliza ao gestor da linha de produção informação acerca das actividades dos agentes dessa mesma linha.

- **AgGE** - *Gestor de Processos.*

  Permite interrogar cada um dos agentes envolvidos no processo.

Agente	Tarefa Principal	Tipo
AgPP	Preparação das Ordens de Fabrico	Interface
AgMP	Selecção e Preparação de Matéria-Prima	Controlo
AgPR	Prensagem	Controlo
AgMA	Maquinação	Controlo
AgSI	Sinterização	Controlo
AgCQ1	Controlo de Qualidade I	Controlo
AgPO	Polimento	Controlo
AgCQ2	Controlo de Qualidade II	Controlo
AgED	Embalagem e Despacho	Controlo
AgMO	Monitorização de Processos	Monitor
AgGE	Gestão de Processos	Interface

Tabela 7.1: Os Agentes do Sistema de Produção.

O comportamento dos agentes do sistema de produção é tipicamente oportunístico, o que lhes confere a capacidade de actuar sobre uma ordem de fabrico sem que esta lhes esteja previamente destinada. Assim, podem decidir autonomamente que ferramenta seleccionar ou qual a melhor forma de realizar a tarefa requerida pela ordem de fabrico.

A Figura 7.8 apresenta uma breve descrição do comportamento genérico de um agente de produção durante o período em que está activo no sistema[7].

[7] O algoritmo descreve as principais acções realizadas por um agente de produção. Nele podem-se identificar três etapas: 1) arranque - fase na qual o agente prepara a sua

```
procedimento agente-de-produção (Identificação-Agente)
    activação-agente(Identificação-Agente,Configuração);
    carregar-base-conhecimento(Identificação-Agente,Configuração);
    obter-endereço-sistema(Endereço);
    comunicação-identificação-agente(Identificação-Agente,Endereço);
    activação-mecanismos-inferência;
    obter-condições-de-operação(Endereço,Operação)
    se Operação então
        repetir
            recolher-ordens-fabrico-do-ambiente(Endereço,Ordens-Fabricação);
            selecção-validação-ordens-fabrico(Ordens-Fabricação,Ordem-Fabrico)
            se Ordem-Fabrico então
                executar-tarefas-ordem-fabrico(Ordem-Fabrico)
            fim-se
            elaboração-relatório(Identificação-Agente,Operação)
            recolher-instruções-supervisor(Fim-Execução)
        até Fim-Execução
    fim-se
    desactivação-mecanismos-inferência;
    limpeza-estruturas-residuais;
    finalização-tarefas-locais(Identificação-Agente,Endereço)
fim-procedimento
```

Figura 7.8: Comportamento Genérico de um Agente de Produção.

plataforma de trabalho local (*activação-agente/2*), carregando para memória a sua base de conhecimento (*carregar-base-conhecimento/2*) recolhendo informação relacionada com a localização (*obter-endereço-sistema/1*) do ambiente do sistema (*Endereço*), informa o supervisor do sistema (*comunicação-identificação-agente/2*) de que está activo e activa os seus mecanismos de inferência, de acordo com a informação contida na sua base de conhecimento (*Configuração*) e com as condições (*Operação*) gerais de operação na linha de produção que recolhe (*obter-condições-de-operação/2*) no ambiente do sistema; 2) tratamento de ordens de fabrico - fase em que o agente recolhe do ambiente do sistema as ordens de fabrico em curso (*recolher-ordens-fabrico-do-ambiente/2*), analisando-as (*selecção-validação-ordens-fabrico/2*), a fim de verificar se pode realizar trabalho sobre alguma delas, e, se tal suceder, executando sobre as peças as operações que a ordem de fabrico (*Ordem-Fabrico*) contiver; após a realização dessas operações regista na sua base de conhecimento o trabalho realizado (*elaboração-relatório/2*) e verifica, de seguida, se existe alguma instrução do supervisor do sistema (*recolher-instruções-supervisor/1*); 3) saída do sistema - o agente desactiva os seus mecanismos de inferência, elimina eventuais estruturas de conhecimento temporárias e residuais que foi criando ao longo das suas actividades - "garbage collection" - e termina a sua execução informando o supervisor do sistema de que já não se encontra activo.

Todos os agentes comunicam entre si através de mensagens, enviadas a partir do ambiente do sistema BeAble. A forma como os diversos agentes se integram no sistema é idêntica à exposta anteriormente para os agentes cognitivos do sistema BeAble (Capítulo 5).

Na Figura 7.9 pode-se observar, através da janela do monitor da linha de produção, a geometria da linha de produção e dos diversos agentes envolvidos no sistema produtivo, assim como alguma da informação de controlo subjacente às acções produtivas. O modelo do sistema contempla as relações interagentes apresentadas na Figura 7.6, que são asseguradas através da passagem de informação entre os agentes (o que corresponde à passagem das peças entre duas máquinas).

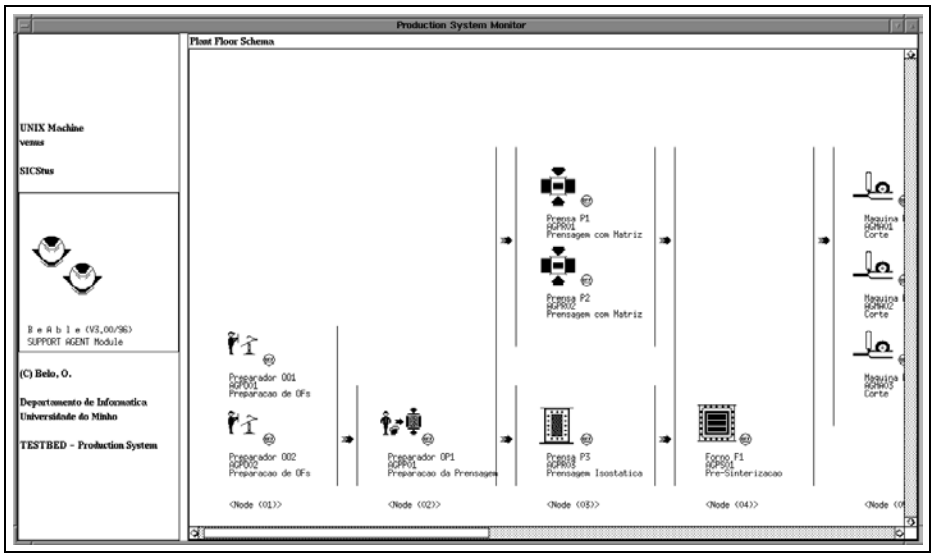


Figura 7.9: Ambiente do Monitor do Agente de Monitorização da Linha de Produção.

7.2.4 O Sistema de Diagnóstico e Aconselhamento Médico

A Medicina sempre constituiu uma área onde a Inteligência Artificial e, em particular, os SBC encontram inúmeras formas de aplicação. Sistemas de suporte à tomada de decisões (C.Kahn 1994) e sistemas para assistência operacional em unidades intensivas de tratamento (M.Dojat, L.Brochard, F.Lemaire & A.Harf 1992) são apenas alguns dos exemplos que se podem apresentar relativamente às potencialidades reais de aplicação dos SBC à área da Medicina. Paralelamente, tem-se estudado a utilização de sistemas médicos computadorizados a fim de determinar o seu impacto no relacionamento médico/paciente (E.Shortliffe 1992) e avaliar as vantagens da aplicação desse tipo de sistemas à Medicina (E.Shortliffe 1993).

Os casos clínicos que exijam pluridisciplinaridade, que requeiram a intervenção de diverso pessoal médico e auxiliar e que impliquem o diálogo e a cooperação entre os intervenientes no processo de resolução de problemas constituem potenciais cenários para aplicação da tecnologia baseada em agentes. Se a estas características se adicionar a necessidade de efectuar uma gestão eficaz da distribuição e partilha dos recursos - que podem encontrar-se em locais físicos distintos - estar-se-á perante uma situação típica para aplicação de um SM.

Em domínios relacionados com o tratamento médico, uma aplicação baseada em agentes deve atender às seguintes características destes últimos (J.Huang, N.Jennings & J.Fox 1995*a*):

- requerem procedimentos explícitos para a gestão dos processos de comunicação;

- exigem mecanismos e estruturas adequadas para assegurar que as tarefas são delegadas nos agentes apropriados;
- implicam mecanismos para a tomada de decisões que lhes permitam raciocinar com informação temporal, incompleta ou contraditória;
- necessitam de especificar e adoptar um conjunto explícito de procedimentos, por forma a serem capazes de monitorizar os seus próprios objectivos e planos.

Tais requisitos podem servir como base para perspectivar a aplicação do sistema BEABLE a domínios de aplicação médica, em particular a áreas de diagnóstico e aconselhamento médico. O desenvolvimento de uma plataforma de teste para o sistema nesta área ajudará a comprovar tal aplicabilidade e a determinar a que nível esta poderá processar-se.

O Caso em Estudo

Com este caso de estudo pretende-se emular o comportamento de um grupo de especialistas [8] através de um conjunto de agentes cognitivos que cooperam para a resolução de problemas, colocados como casos clínicos. Relativamente aos casos já apresentados neste capítulo, este é, sem dúvida, o de maior dimensão, tanto no que respeita ao número de agentes implicados como no concernente à complexidade do modelo de coordenação do sistema de agentes. É o único, dos casos em estudo, que aborda a problemática da aprendizagem.

[8] O conhecimento integrado nas bases dos agentes cognitivos não é suficiente e está longe de permitir a emulação de um especialista. Contudo, o conhecimento integrado, embora insuficiente, permitiu desenvolver as formas de conhecimento necessárias para testar o sistema BEABLE.

A realização desta plataforma de teste tinha como objectivos principais os seguintes:

- analisar de uma forma geral a dinâmica do sistema, e em particular os processos que se desenvolvem entre os agentes de interface e os agentes cognitivos;
- testar os modelos de cooperação baseados na partilha de informação e na passagem do controlo do processo de resolução do problema;
- analisar o comportamento dos agentes cognitivos com a definição de inibidores de activação na sua base de conhecimento;
- verificar a efectividade do controlo do sistema nas entradas e saídas do sistema dos agentes de interface e especialistas durante os processos de resolução de problemas;
- verificar o desempenho do sistema numa aplicação real;
- confirmar a aplicabilidade do sistema à área do diagnóstico;
- atender à reutilização de alguns componentes dos agentes do sistema.

Configuração e Dinâmica do Sistema

O sistema de diagnóstico e aconselhamento médico é constituído por um conjunto de doze agentes cognitivos, representando outros tantos especialistas, distribuídos por diferentes domínios (Tabela 7.2), para além dos agentes de controlo e supervisão do sistema. Cada um dos agentes especialistas emula parte da perícia de um médico de clínica geral.

Agente	Domínio Médico de Aplicação
AgM01	Sensações de Mal-Estar
AgM02	Ansiedade
AgM03	Fadiga
AgM04	Perda de Peso
AgM05	Febre
AgM06	Problemas de Sono
AgM07	Estados Depressivos
AgM08	Garganta Inflamada
AgM09	Problemas de Memória
AgM10	Tonturas
AgM11	Rouquidão
AgM12	Excesso de Peso

Tabela 7.2: Relação dos Agentes Cognitivos do Sistema de Diagnóstico Médico.

Os processos de diagnóstico são desencadeados a partir dos agentes de interface, por iniciativa dos utilizadores. Serão necessários tantos agentes de interface quantos os utilizadores presentes no sistema. Estes agentes asseguram, via ambiente do sistema, a ligação entre os utilizadores e os agentes cognitivos.

Os utilizadores podem activar dois tipos de diagnóstico:

- **Individual** - consultando apenas um agente cognitivo, através da invocação directa do agente;
- **Colectivo** - consultando um conjunto de agentes cognitivos, através da apresentação de uma lista de palavras-chave, ou sintomas.

Na primeira situação, o diagnóstico desenvolve-se a partir de um diálogo entre o utilizador e o agente cognitivo. Em casos de insucesso, o controlo do processo de diagnóstico pode ser passado a outros agentes cognitivos,

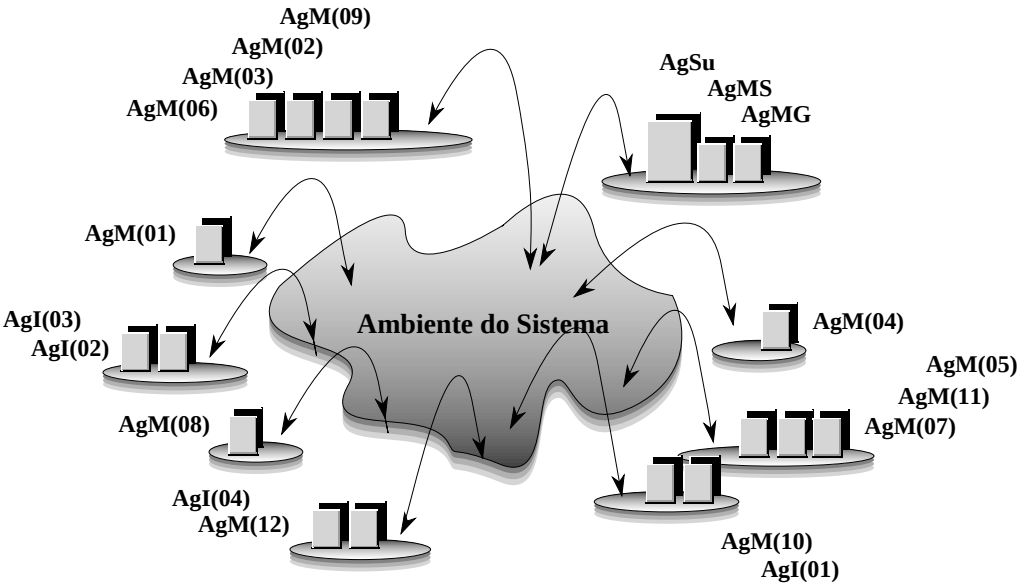


Figura 7.10: Modelo do Sistema de Diagnóstico e Aconselhamento Médico.

que, em diálogo com os utilizadores, procurarão continuá-lo e concluí-lo. A Figura 7.11 apresenta os monitores de um agente de interface e de um agente cognitivo, fazendo referência a algumas das acções por eles realizadas durante um processo de diagnóstico.

Num processo colectivo, a lista apresentada pelo utilizador é enviada para o sistema através de um agente de interface, em formato próprio, agindo selectivamente sobre a comunidade de agentes cognitivos presente no ambiente do sistema. Isto é, apenas são activados os agentes com potencial para a resolução do problema. Cada agente actua por si, embora partilhe a informação que vai coligindo com os restantes agentes do sistema.

Durante um processo de diagnóstico, seja ele individual ou colectivo, o utilizador tem a possibilidade de questionar o agente, indagar sobre o estado da resolução do problema e obter uma explicação ou mesmo aconselhamen-

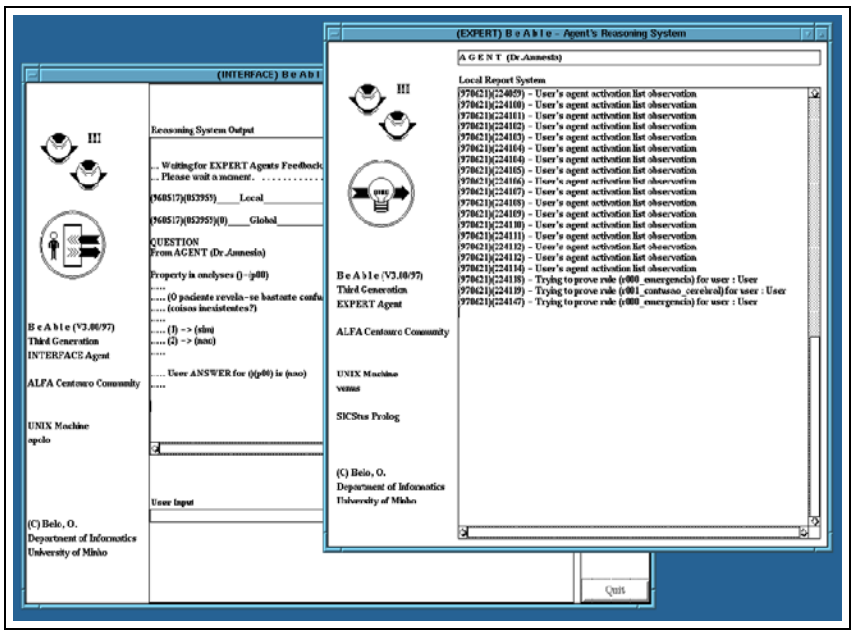


Figura 7.11: Ambientes de um Agente de Interface e de um Agente Cognitivo num Processo de Diagnóstico.

to sobre medidas. Os agentes cognitivos têm a faculdade de dar atenção simultânea a vários processos de diagnóstico, provenientes de diferentes utilizadores.

7.2.5 Síntese das Características das Plataformas de Teste

O desenvolvimento e a implementação das plataformas de teste permitiu analisar a aplicabilidade e o desempenho do sistema quando utilizado em problemas de classificação, controlo e diagnóstico. Os SM utilizados nas plataformas de teste necessitaram de pequenas adaptações, em particular os agentes cognitivos, que em alguns casos precisaram de ser reestruturados. A estrutura global do sistema não sofreu, todavia, qualquer tipo de

reestruturação. O desenvolvimento de novos agentes de monitorização em duas das plataformas (Sistema de Ar Condicionado e Sistema de Controlo da Produção) ficou a dever-se, apenas, à natureza dos problemas em estudo - simulações.

Plataformas	Agentes por Processo	Cooperação	Interacção
C. de Leveduras	$n \times A_gI + 2 \times A_gC$	Tarefas/Horizontal	$nA_gI \rightarrow 1A_gC$ $1A_gC \rightarrow 1A_gC$
Ar Condicionado	$n \times A_gC$	Resultados/Árvore	$nA_gC \rightarrow 1A_gC$ $1A_gC \rightarrow nA_gC$
S.C. da Produção	$2 \times A_gI + k \times A_gC$	Tarefas/Horizontal	$1A_gC \rightarrow 1A_gC$ $nA_gC \rightarrow 1A_gC$
S.D. Médico	$n \times A_gI + k \times A_gC$	Resultados/Horizontal Resultados/Árvore	$nA_gI \rightarrow 1A_gC$ $1A_gC \rightarrow 1A_gC$ $1A_gC \rightarrow nA_gC$

Tabela 7.3: Agentes Envolvidos nas Plataformas de Teste do Sistema.

A Tabela 7.3 apresenta, para cada uma das plataformas, o número de agentes envolvidos num processo, o tipo de cooperação desenvolvida e a forma de interacção entre os agentes. A título de exemplo, e tomando como base o sistema de classificação de leveduras, verifica-se que esta plataforma de teste atendeu aos seguintes aspectos:

- envolvimento de n agentes de interface ($n \times A_gI$) e de dois agentes cognitivos ($2 \times A_gC$) (o classificador de espécies de leveduras e o sistema gestor de bases de dados do sistema de classificação);
- cooperação entre os agentes através da realização de tarefas e de forma horizontal;
- envolvimento dos seguintes agentes, de acordo com o tipo de interacção que o sistema suporta :

– n agentes de interface e um só agente cognitivo ($nA_gI \rightarrow 1A_gC$);

– dois agentes cognitivos ($1A_gC \rightarrow 1A_gC$).

Capítulo 8

Conclusões e Trabalho Futuro

Teorias, linguagens, arquitecturas e aplicações no domínio da CBA; algumas referências à interligação com o trabalho realizado. Análise dos pontos mais relevantes do trabalho desenvolvido no âmbito do projecto BeAble. *Apresentação do seu estado actual em termos de características funcionais, arquitectura global e aplicações. Contribuições da tese. Considerações sobre o trabalho realizado e apresentação de algumas linhas de investigação futura.*

8.1 Teorias, Linguagens, Arquitecturas e Aplicações

A concepção e o desenvolvimento de arquitecturas para agentes inteligentes constituiu desde sempre uma das áreas privilegiadas pela comunidade da IA. A partir dos anos 80, em particular, esse interesse materializa-se naquilo que hoje se designa por CBA (Capítulo 2). A Simulação, a Modelação e o Controlo de Sistemas, a Engenharia do Conhecimento, a Vida Artificial e os Sistemas de Informação com aplicação à Engenharia são algumas das áreas

nas quais a CBA já "marcou alguns pontos" [1].

A ICMAS, a PAAM, a MAAMAW e a ATAL são exemplos de eventos onde o número de submissões e a qualidade dos trabalhos têm vindo a aumentar paulativamente. Paralelamente, a influência da CBA também se tem feito sentir em outras áreas do conhecimento, sendo de referir a área da Simulação, com três grandes acontecimentos - o ESS ("European Simulation Symposium"), a Eurosim ("European Simulation Congress") e a ESM ("European Simulation and Management") - e uma conferência de SP, o WCES ("World Congress on Expert Systems").

A partir da análise dos tópicos tratados em conferências, da forma como as revistas e jornais da área estão organizados e da catalogação de endereços da Internet, é possível identificar quatro linhas de acção para o estudo e desenvolvimento da CBA (M.Wooldridge & N.Jennings 1995*a*):

- **Teorias**.

  Em que se considera a definição e o desenvolvimento de formalismos para a representação das propriedades dos agentes.

- **Linguagens**.

  Em que se visa a construção e o desenvolvimento de linguagens de programação, por forma a dar corpo aos princípios expostos pelos teóricos da CBA.

- **Arquitecturas**.

  Que permitem estabelecer a "ponte" entre a especificação de agentes e

[1] Nos endereços da Internet *http://www.cs.umbc.edu/agents/technology/* e *http://www.cl.cam.ac.uk/users/rwab1/ag-pages.html* podem-se encontrar referências e apontadores para endereços WWW que endossam a temática da CBA.

a sua implementação, disponibilizando os mecanismos e as estruturas necessárias para a implementação de SM.

- **Aplicações**.
  Em que se procede ao desenvolvimento de software integrando técnicas da CBA.

O desenvolvimento do sistema BEABLE processou-se no domínio das arquitecturas. Ao providenciar um ambiente computacional distribuído de aplicação genérica, suportado por classes de agentes orientadas fundamentalmente para tarefas de supervisão, interface e resolução de problemas, o sistema disponibiliza os meios[2] essenciais para implementação de SM em diferentes domínios de conhecimento, em particular nas áreas do diagnóstico, classificação e controlo.

Apresentar-se-á em seguida o estado do sistema, fazendo-se menção a outros sistemas de características similares, que constituem provas de que esta linha de acção continua em crescendo, na procura de ambientes computacionais dirigidos ao agente cada vez mais poderosos e versáteis. É, contudo, de salientar que muitas das arquitecturas referidas usualmente na literatura da CBA surgiram a partir do desenvolvimento de aplicações em domínios de conhecimento específicos. Tal circunstância indicia uma parceria, ou mesmo cumplicidade, entre arquitecturas e aplicações. Convém, todavia, ter em atenção a forte influência das restantes linhas na modelação, validação, normalização e regulamentação dos sistemas desenvolvidos, com especial realce

[2]Um ambiente distribuído para partilha de conhecimento no seio de uma comunidade de agentes, plataformas de suporte à interacção com os utilizadores e à implementação de sistemas baseados em conhecimento, sistemas de interface para interligação do sistema à Web e a outras comunidades de agentes, cuja referência e apresentação pode ser encontrada no Capítulo 4.

para as teorias formais sobre agentes.

8.2 Estado e Características do Sistema

Os desenvolvimentos dos últimos anos nos meios científicos e técnicos, em termos das tecnologias de SM, influenciaram sistematicamente o desenho, o planeamento e a implementação do sistema BEABLE, conduzindo-o à actual versão.

Em síntese, e com base na informação contida nos capítulos anteriores, a análise [3] das características do sistema BEABLE é desenvolvida a partir de dois pontos principais: o ambiente computacional do sistema e o tipo de agentes que este pode acolher.

O ambiente computacional apresenta as seguintes características:

- Revela um nível de paralelismo de pequena escala, justificado pelo facto de o número de processos computacionais activados por aplicação não ter ultrapassado, durante as fases de teste e validação do sistema, as duas dezenas.
- Actua como um elo de ligação entre os diversos agentes que o materializam, através de uma estrutura dedicada à partilha e à comunicação de informação entre os agentes, bem como ao armazenamento da informação de controlo necessária à sua coordenação.
- Suporta um sistema de características heterogéneas, que permite a in-

[3] A análise aqui efectuada apenas considera algumas das características gerais do sistema, podendo encontrar-se nos Capítulos 4, 5 e 6 uma explicação mais detalhada do mesmo.

tegração de diversos tipos de agentes no seu ambiente, com diferentes perícias e conhecimentos, possibilitando igualmente a integração de outros tipos de entidades, através do desenvolvimento de plataformas específicas.

- Permite integrar agentes oriundos de domínios de conhecimento diferentes.

- Disponibiliza um esquema flexível de coordenação, definido em torno dos modelos de cada agente e de acordo com os elementos de controlo mantidos no ambiente do sistema, evitando desta forma eventuais situações de contenção ou estrangulamento, normalmente originadas pela aplicação de estratégias apertadas de controlo centralizado.

- Na resolução de problemas, o conhecimento encontra-se organizado a dois níveis:

  - **Globalmente**.

    O conhecimento é armazenado no ambiente (distribuído) do sistema, estando organizado em três camadas distintas:

    * *Coordenação e controlo.*

      Nível no qual está localizado o conhecimento que suporta as acções de supervisão e de gestão do sistema, assim como informação de carácter geral relativa aos agentes e utilizadores presentes no sistema.

    * *Cooperação.*

      Nível no qual está armazenado o conhecimento relativo à informação partilhada pelos agentes durante a resolução dos problemas.

    * *Encaminhamento.*

      Nível no qual os agentes encaminhadores armazenam o conhecimento relativo às transacções de informação intercomunidades.

  - **Localmente**.

    O conhecimento é armazenado pelos agentes:

    * nas suas memórias de trabalho, quando diz respeito aos problemas sobre os quais estão a laborar;
    * nas suas bases de conhecimento, quando está relacionado com a perícia e a informação acerca do seu domínio de aplicação.

- Encontra-se estruturado para problemas em que há cooperação entre agentes geograficamente distribuídos, permitindo-lhes intervir oportunisticamente na procura de soluções.

- Permite o desenvolvimento de processos de comunicação e cooperação entre diferentes comunidades de agentes (com ambientes computacionais próprios), suportados pelos agentes de encaminhamento, os quais estabelecem as "pontes" para a comunicação intercomunidades.

- Possibilita aos utilizadores interactuarem com o sistema através de uma ferramenta de "browsing" da Internet, de forma idêntica à utilização de um agente de interface. Esta capacidade permite alargar os horizontes de resolução de problemas dos agentes cognitivos, para além de facultar aos utilizadores o acesso ao sistema através de qualquer nodo da Internet.

Em relação ao tipo de agentes que o sistema acolhe, verifica-se o seguinte:

- Foram desenvolvidas seis classes de agentes, estando cada uma delas associada à realização de um conjunto de funções bem definidas. Existem agentes para a realização de serviços de administração e suporte operacional do sistema (Supervisor, Monitor e Motor do Sistema); de serviços de suporte à interacção homem-máquina (agentes de interface e guardiões); de serviços de análise e resolução de problemas (agentes cognitivos) e de serviços de encaminhamento entre comunidades (Encaminhador).

- As classes de agentes são constituídas por entidades autónomas, independentes em termos de perícias, mas parcialmente dependentes do conhecimento de outras fontes ou agentes.

- Os agentes são, de uma forma geral, à excepção dos agentes Monitor e Motor, entidades de grande granularidade, apresentando uma estrutura em tudo semelhante a um SBC.

- Os agentes interactuam através da troca de mensagens e partilham estruturas de conhecimento através do ambiente do sistema.

- Possíveis instanciações das diferentes classes de agentes constituem-se em comunidades de agentes.

- A resolução de um problema envolve, pelo menos, um agente de interface (ou um "navegador" da Internet mais um agente guardião) e um ou vários agentes cognitivos, a que se juntam os agentes que têm a seu cargo os serviços de supervisão e suporte operacional do sistema.

No Apêndice A apresentam-se alguns sistemas normalmente referidos em trabalhos realizados no domínio da CBA. A sua análise permitirá uma apreciação mais concreta das características do sistema BeAble.

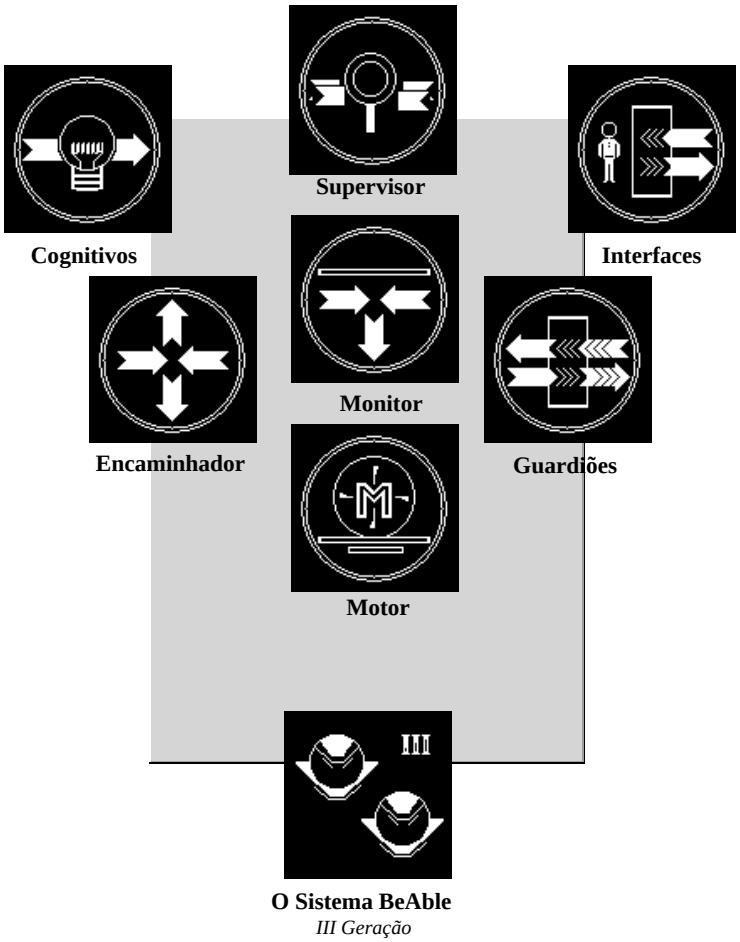


Figura 8.1: Os Agentes do Sistema BeAble.

A selecção dos sistemas foi feita com base nas características que apresentam relativamente ao tipo de agentes utilizados, aos modelos de coordenação, cooperação e distribuição que suportam e ao tipo de ambiente que disponibilizam. Os sistemas seleccionados foram o ARCHON (N.Jennings & T.Wittig 1992) (T.Wittig 1992) (N.Jennings 1994) (T.Wittig et al. 1994), o COSY (B.Burmeister & K.Sundermeyer 1992), o HECODES (C.Zhang 1990) (C.Zhang 1992) (C.Zhang 1993), o INTERRAP (J.Müller & M.Pischel 1994) (J.Müller, M.Pischel & M.Thiel 1994), o MAPS (O.Baujard, S.Pesty &

C.Garbay 1992)(O.Baujard, S.Pesty & C.Garbay 1993) e a UPShell (E.Oliveira & F.Mouta 1993) (E.Oliveira, F.Mouta & A.Rocha 1993*a*) (F.Mouta 1996).

Em A.Bond & L.Gasser (1988) e M.Wooldridge & N.Jennings (1995*b*) podem-se encontrar referências relativas a outros sistemas existentes.

8.3 Contribuições da Tese

A disponibilização de um ambiente computacional distribuído, de aplicação genérica, para a implementação de SM constitui a principal contribuição deste trabalho. A investigação desenvolvida conduziu, assim, à implementação do sistema BeAble (O.Belo & J.Neves 1996*a*), cujas características e funcionalidades são mencionadas a seguir:

- Trata-se de um sistema funcionalmente distribuído, com acções de controlo descentralizadas, suportadas por modelos de reacção, coordenação e cooperação definidos nas bases de conhecimento dos agentes, dando corpo a um sistema com:
  - grande flexibilidade em termos funcionais;
  - considerável possibilidade de reutilização dos seus componentes;
  - amplas facilidades na integração de novos componentes;
  - possibilidade de equacionar o equilíbrio de cargas nos processos dos agentes;
  - maior robustez e resistência a situações de falha ou ruptura de recursos computacionais;

  - possibilidade de paralelizar os processos computacionais dos agentes, com o consequente aumento do desempenho do sistema.

- O sistema apresenta um modelo computacional dinâmico, de geometria variável, onde os agentes podem ligar-se e desligar-se a qualquer momento, sem afectar directamente a operacionalidade daquele. O efeito provocado reflecte-se, apenas, na disponibilidade ou indisponibilidade de um agente para contribuir para a resolução dos problemas em curso (O.Belo & J.Neves 1995*a*).

- É possível definir comunidades heterogéneas de agentes, através da integração de diferentes tipos de agentes num único ambiente, distribuídos de acordo com as suas crenças, perícias e conhecimento.

- Concepção e desenvolvimento de agentes com capacidades cognitivas (O.Belo & J.Neves 1996*b*), possuindo as seguintes características e funcionalidades:

  - revelam comportamentos oportunísticos e posturas benévolas;
  - realizam as tarefas em tempo real e de forma paralela, autónoma e independente;
  - apresentam mecanismos de suporte à cooperação na horizontal, embora complementada, em alguns casos, com formas de cooperação em árvore;
  - apresentam mecanismos de verificação de consistência locais e globais;
  - possuem mecanismos de aprendizagem, por forma a potenciar as capacidades de adaptação dos agentes a novos problemas.

- Alargamento do horizonte de utilização do sistema, através da disponibilização de "portas de acesso" especificamente desenvolvidas para os utilizadores acederem ao sistema a partir de qualquer nodo da rede Internet (O.Belo & A.Ribeiro 1996).

- Um ambiente computacional para a implementação de sistemas baseados em multicomunidades de agentes, abrindo aos agentes de uma comunidade a possibilidade de cooperarem com agentes de outras comunidades. Formas de macro e microcooperação foram integradas nesta nova filosofia de interacção entre SM (O.Belo & J.Neves 1997*a*)(O.Belo & J.Neves 1997*b*).

8.4 Conclusões e Linhas de Orientação Futuras

O trabalho desenvolvido deu origem a um sistema cujas metas inicialmente definidas (Capítulo 1) foram cumpridas. O protótipo desenvolvido pode ser utilizado como uma ferramenta para a exploração de modelos de coordenação, cooperação e resolução de problemas numa comunidade de agentes (ou intercomunidades), para a reformulação de cenários de simulação (O.Belo & J.Neves 1995*d*) (O.Belo 1996) (O.Belo 1997) ou para a implementação de aplicações reais envolvendo problemas de diagnóstico, classificação, monitorização ou controlo, entre outras.

A distribuição das acções de controlo pelas diferentes entidades que compõem o sistema contribui de forma significativa para a redução de situações de contenção ou bloqueio fatal ("deadlock").

O projecto inicial tinha como objectivo desenvolver um sistema para a implementação de um comunidade de SBC cooperantes, tendo posteriormente sido ampliado e reformulado com vista ao tratamento de multicomunidades de agentes e acesso via Internet.

A comunicação e o desenvolvimento da cooperação entre comunidades são assegurados por uma classe de agentes especiais: os Encaminhadores (Capítulo 6). A segunda extensão, o acesso via Internet, permitiu abrir o sistema a um novo (e potencialmente bastante extenso) grupo de utilizadores. Foram criadas novas funcionalidades, integradas na classe de agentes dos guardiões, que suportam o interface entre o BEABLE e um sistema de "browsing" da Internet. Desta forma, os utilizadores da Web podem ter acesso ao sistema de uma forma simples, caso lhes tenham atribuído permissões para tal.

Relativamente às aplicações desenvolvidas (Capítulo 7), estas permitiram estudar e avaliar as funcionalidades gerais do sistema - fiabilidade, mecanismos de comunicação, coordenação e cooperação, distribuição e paralelização dos processos dos agentes, e desempenho -, bem como verificar a aplicabilidade do sistema a problemas de classificação, controlo e diagnóstico.

Algumas das experiências levadas a cabo durante a fase de desenvolvimento, testes e validação das aplicações desenvolvidas apontaram para a necessidade de se reestruturar alguns dos sistemas de comunicação ao nível do agente, em prol de uma maior fiabilidade, flexibilidade e eficiência.

Em termos de trabalho futuro, serão de considerar as seguintes linhas de orientação:

- Estabelecimento de meios de comunicação entre os agentes cognitivos e

o exterior via meios electrónicos ou computacionais - correio electrónico, sistemas de conversação directa ("talk") ou de conversação em grupo ("chat").

- Melhoramento das capacidades de aprendizagem e de resolução de conflitos entre agentes, juntamente com a definição de estratégias para a credibilização e garantia da qualidade das soluções. Desta forma, será possível desenvolver os meios para que:
  - em situações onde é proposta mais do que uma solução para um dado problema, os utilizadores possam avaliar qual a solução a considerar, com o auxílio dos agentes de interface;
  - os agentes cognitivos possam, não somente desencadear a intervenção de outros agentes, através de um processo de passagem de controlo, mas também avaliar qual o agente com mais possibilidades de prosseguir na resolução do problema.
- Aumento da expressividade da linguagem de representação de conhecimento dos agentes cognitivos, no sentido de alcançar um modelo de representação mais efectivo e abrangente.
- Desenvolvimento de um sistema de recuperação de falhas, para que seja possível a reposição do estado dos problemas em resolução no momento em que ocorreu uma falha.
- Implementação de mecanismos de revisão de consistência mais efectivos e abrangentes capazes de englobarem no seus processos a informação de todos os agentes do sistema envolvidos com um dado problema.
- Integração de modelos macro e microeconómicos como extensão dos actuais modelos de resolução de problemas e partilha de recursos dos

agentes cognitivos (A.Cunha & O.Belo 1997*a*) (A.Cunha & O.Belo 1997*b*).

- Definição formal do sistema[4], por forma a assegurar a sua evolução natural, facilitar a sua manutenção e definir sem ambiguidade os vários conceitos que o sistema incorpora.

- Extensão do sistema em termos estruturais e funcionais, de modo a que este possa acolher sistemas de vida artificial.

[4]Alguns exemplos de métodos formais para a especificação de SM e sua aplicação podem ser encontrados em L.Cavedon & G.Tidhar (1995), R.Fagin, J.Halpern, Y.Moses & M.Vardi (1995), B.Linder (1996) e J.Neves, J.Machado, A.Abelha & P.Novais (1996).

Bibliografia

A.Bond & L.Gasser, eds (1988), *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, Inc.

A.Costa, E.Rodrigues, F.Pinto, J.Macedo & M.Nicolau (1995), *Internet Guia Prático do Cibernauta*, Campo das Letras, Editores, S.A.

A.Cunha, C.Biscaia, M.Torres, L.Sobral & O.Belo (1996), Simulating the Use of Autonomous Intelligent Agents on Cellular Manufacturing Plant Floors, *in* 'Proceedings of the 8th European Simulation Symposium (ESS'96)', Genoa, Italy.

A.Cunha, C.Biscaia, M.Torres, L.Sobral & O.Belo (1997), Parallel Neural Network Recognition - a Multi-Agent System Approach, *in* 'Proceedings of the High - Performance Computers 97', Santiago de Compostela, Spain.

A.Cunha, L.P.Santos & O.Belo (1997), Simulador de Estratégias de Distribuição de Carga, *in* 'Actas do Terceiro Encontro Nacional do Colégio de Engenharia Electrotécnica da Ordem dos Engenheiros', Porto, Portugal.

A.Cunha & O.Belo (1997*a*), A Multi-Agent Based Approach for Load Distribution in Multi-Enterprise Environments, *in* 'Proceedings of the 15th

IASTED International Conference on Applied Informatics', Innsbruck, Austria.

A.Cunha & O.Belo (1997*b*), An Electronic Commerce Framework for Resource Allocation among Multi-Agent Enterprises, *in* 'Proceedings of the 10th International FLAIRS Conference', Florida, USA.

A.Dragoni (1992), A Model for Belief Revision in a Multi-Agent Environment, *in* E.Werner & Y.Demazeau, eds, 'Decentralized A.I.-3', Elsevier Science Publishers B.V., pp. 103–112.

A.Gadomski & J.Zytkow (1994), Abstract Intelligent Agents: Paradigms, Foundations and Conceptualization Problems, *in* 'Proceedings of the Second International Round Table on Abstract Intelligent Agent (AIA94)'.

A.Gillies (1994), Assuring Quality in Expert Systems, *in* 'Conference of The Second World Congress on Expert Systems', Lisbon, Portugal, pp. 499–503.

A.Gisolfi & V.Loia (1994), 'Designing Complex Systems Within Distributed Architectures: An Inteligent Tutoring Systems Perspective', *Applied Artificial Intelligence* **8**(3), 393–411.

A.Ovalle (1994), Using Heterogeneous Multi-agent Technology for Expert Systems Desig and Development, *in* J.Liebowitz, ed., 'Moving Toward Expert Systems in the 21st Century, Proceedings o the Second World Congress on Expert System', New York: Cognizant Communication Corporation, Lisbon, Portugal, pp. 8–15.

A.Reinhardt (1994), 'The Network with Smarts', *Byte* **19**(10).

A.Requicha (1996), 'Geometric Reasoning for Intelligent Manufacturing', *Communications of the ACM* **21**(2), 71–76.

A.Ribeiro, V.Alves & J.Neves (1993), Parallel Distributed Processing in Discrete Event Simulation - A Process Oriented Approach, *in* A.Verbraeck & E.Kerckhoffs, eds, 'Proceedings of the European Simulation Symposium (ESS 93)', Delft, The Netherlands, pp. 135–140.

A.Ribeiro, V.Alves & J.Neves (1994), Distributed Problem Solving - A Case Study, *in* N.Jackson & N.Cruz, eds, 'Proceedings of the International Training Equipment Conference and Exhibition', The Hague, The Netherlands, pp. 229–235.

A.Rocha (1994), Cooperação entre Agentes Cognitivos: Negociação e Resolução de Conflitos, Master's thesis, Faculdade de Engenharia, Universidade do Porto.

A.Slade (1994), CONSENSUS: A Methodology for Distributed Expert Systems, *in* 'Conference of The Second World Congress on Expert Systems', Lisbon, Portugal, pp. 342–346.

A.Wood (1994), Agent-Based Interaction, PhD Progress Report PR-94-4, University of Birmingham, School of Computer Science.

B.Burmeister & K.Sundermeyer (1992), Cooperative problem solving guided by intentions and perception, *in* 'Proceedings of the Third European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-91)', Elsevier Science Publishers B.V., Amsterdam, The Netherlands, pp. 77–92.

B.Codenotti & M.Leoncini (1992), *Introduction to Parallel Processing*, Addison-Wesley.

B.Gaines & J.Boose, eds (1990), *Machine Learning and Uncertain Reasoning*, Academic Press.

B.Hayes-Roth (1985), 'A Blackboard Architecture for Control', *Artificial Intelligence* **25**(3), 251–321.

B.Hayes-Roth (1992), 'Opportunistic Control of Action in Intelligent Agents', *IEEE Transactions on Systems, Man, Cybernetics: Special Issue on Planning, Scheduling, and Control* **23**(6), 1575–1587.

B.Hayes-Roth (1993), Architectural Foundations for Real-Time Performance in Intelligent Agents, *in* 'Second Generation Expert Systems', Springer-Verlag, pp. 643–672.

B.Hayes-Roth (1995), Agents on Stage: Advancing the State of the Art of AI, Technical report, Knowledge Systems Laboratory, KSL-95-50.

B.Linder (1996), Modal Logics for Rational Agents, PhD thesis, Department of Computer Science, Faculty of Mathematics and Computer Science, University of Utrecht.

B.Malheiro and N.Jennings and E.Oliveira (1994), Distributed Belief Revision, *in* '13th International Distributed Artificial Intelligence Workshop (presented as a poster)', Seattle, USA.

B.Malheiro, N.Jennings & E.Oliveira (1994), Belief Revision in Multi-Agent Systems, *in* 'Proceedings of the 11th European Conference on Artificial Intelligence (ECAI94)', Jonh Wiley & Sons, Ltd, Amsterdam.

B.Webber & N.Nilsson, eds (1981), *Readings in Artificial Intelligence*, Tioga Publishing Company.

C.Castelfranchi (1995), *Guarantees for autonomy in cognitive agent architecture*, Springer-Verlag.

C.Holsapple & V.Raj (1994), 'An Exploratory Study of Two KA Methods', *Expert Systems* **11**(2), 77–87.

C.Kahn (1994), Clinical Trial and Evaluation of a Prototype Case-Based System for Planning Medical Imaging Work-up Strategies, *in* 'Proceedings of AAAI '94 Conference, Workshop on CBR', Seattle, Washington.

C.Mason & R.Johnson (1989), DATMS: A Framework for Distributed Assumption Based Reasoning, *in* L.Gasser & M.Huhns, eds, 'Distributed Artificial Intelligence', Pitman and Morgan Kaufmann, Inc, chapter 13, pp. 293–317.

C.Munday, J.Dangedej, T.Cross & D.Lukose (1995), Motivation and Perception Mechanisms in Mobile Agents for Electronic Commerce, *in* 'Proceedings of the First Australian Workshop on Distributed Artificial Intelligence', Canberra, Australia, pp. 144–158.

C.Zhang (1990), HECODES: a framework for heterogeneous cooperative distributed expert systems, PhD thesis, University of Queensland, Australia.

C.Zhang (1992), 'Cooperation under uncertainty in distributed expert systems', *Artificial Intelligence* **56**(1), 21–69.

C.Zhang (1993), 'Hecodes: a framework for heterogeneous cooperative distributed expert systems', *SIGART Bulletin* **4**, 50–51.

C.Zhang & D.Bell (1991), ' HECODES: a framework for heterogeneous cooperative distributed expert systems', *International Journal on Data & Knowledge Engineering* **6**(3), 251–273.

D.Bell & J.Grimson (1992), *Distributed Database Systems*, Addison-Wesley Publishing Company, Inc.

D.Cabeza & M.Hermenegildo (1996), *html.pl: A Simple HTML Package for Prolog and CLP Systems, Description and User's Manual (version 96.1.1)* , Computer Science Department, Technical University of Madrid (UPM).

D.Chu (1993), I.C. Prolog II: a Language for Implementing Multi-Agent Systems, Technical report, Department of Computing, Imperial College of Science, Technology and Medicine.

D.Chu & Y.Cosmadopoulos (1992), I.C. Prolog II: A Language for Implementing Dustributed Systems, Technical report, Department of Computing, Imperial College of Science, Technology and Medicine.

D.Cockburn & N.Jennings (1995), ARCHON: A Distributed Artificial Intelligence System for Industrial Applications, *in* G.O´Hare & N.Jennings, eds, 'Foundations of Distributed Artificial Intelligence', Wiley & Sons.

D.Corkill (1989), Design Alternatives for Parallel and Distributed Blackboard Systems, *in* V. Jagannathan, R. Dodhiawala & L. Baum, eds, 'Blackboard Architectures and Applications', Academic Press, Inc., chapter 6, pp. 99–136.

D.Corkill (1991), 'Blackboard Systems', *AI Expert* pp. 41–47.

D.Corkill, K.Gallagher & P.Johnson (1987), Achieving Flexibility, Efficiency, and Generality in Blackboard Architectures, *in* 'Sixth National Conference on Artificial Intelligence (AAAI'87)', Vol. 1, Seattle, Washington, U.S.A., pp. 18–23.

D.Gagné, J.Desbiens & G.Nault (1994), A Multi-Agent System Simulating Crew Interaction in a Military Aircraft, *in* 'Conference of The Second World Congress on Expert Systems', Lisbon, Portugal, pp. 83–86.

D.McAllester (1978), A Three Valued Truth Maintenance System, AI Memo 473, Technical report, Cambridge, MA: Massachusetts Institute of Technology, AI Lab.

D.Miranker (1987), TREAT: A Better Match Algorithm for AI Production Systems, *in* 'Sixth National Conference on Artificial Intelligence (AAAI'87)', Vol. 1, Seattle, Washington, U.S.A., pp. 42–47.

D.Riecken (1994*a*), 'A Conversation with Marvin Minsky About Agents', *Communications of the ACM* **37**(7), 22–29.

D.Riecken (1994*b*), 'M: An Architecture of Integrated Agents', *Communications of the ACM* **37**(7).

D.Steiner (1996), IMAGINE: An Integrated Environment for Constructing Distributed Artificial Intelligence Systems, *in* 'Foundations of Distributed Artificial Intelligence', John Wiley & Sons, Inc, pp. 345–365.

D.Waterman (1986), *A Guide to Expert Systems*, Addison-Wesley Publishing Company, Inc.

D.Waterman & F.Hayes-Roth (1978), *Pattern-Directed Inference Systems*, Academic Press, Inc.

E.Charniak & D.McDermott (1985), *Introduction to Artificial Intelligence*, Addison-Wesley Publishing Company, Inc.

E.Demirci & T.Arraz (1994), An Expert System Application for Production Simulation, *in* A.Kaylan, A.Lehmann & T.Oren, eds, 'Proceedings of the European Simulation Symposium', Vol. I, Istanbul, Turkey, pp. 302–306.

E.Dubois & M.Petit (1994), The Formal Requirements Engineering of Manufacturing Systems, *in* S.M.Deen, ed., 'Second International Working Conference on Cooperative Knowledge Based Systems (CKBS'94)', Keele, UK, pp. 67–82.

E.Durfee (1988), *Coordination of Distributed Problem Solvers*, Kluwer Academic Publishers.

E.Durfee (1992), 'What Your Computer Really Needs to Know, You Learned in Kindergarten', *Proceedings of the 10th National Conference on Artificial Intelligence* pp. 858–864.

E.Durfee & J.Rosenschein (1994), 'Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples', *Proceedings of the International Workshop on Distributed Artificial Intelligence* .

E.Durfee & V.Lesser (1991), 'Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation', *IEEE Transactions on Systems, Man and Cybernetics* **21**(5).

E.Durfee, V.Lesser & D.Corkill (1988), Coherent Cooperation Among Communicating Problem Solvers, *in* 'Readings in Distributed Artificial Intelligence', Morgan Kaufmann Publishers, Inc, pp. 268–284.

E.Durfee, V.Lesser & D.Corkill (1989), 'Trends in Cooperative Distributed Problem Solving', *IEEE Transactions on Knowledge and Data Engineering* **1**(1), 63–83.

E.Oliveira & F.Mouta (1993), A Distributed AI Architecture Enabling Multi-Agent Cooperation, *in* 'Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems', Edinburgh, Scotland.

E.Oliveira, F.Mouta & A.Rocha (1993*a*), Cooperation in a Multi-Agent Community, *in* 'Proceedings of the 13th International Conference on Artificial Intelligence, Expert Systems and Natural Language', Avignon, France.

E.Oliveira, F.Mouta & A.Rocha (1993*b*), Negotiation and Conflict Resolution within a Community of Cooperative Agents, *in* 'Proceedings of the International Symposium on Autonomous Decentralized Systems (ISADS93)', Kawasaki, Japan.

E.Oliveira, J.Fonseca & A.Steiger-Garcao (1997), Resource Selection and Cost Estimation on a MAS for Civil Construction Companies, *in* 'Proceedings of the International Workshop on Distributed Artificial Intelligence and Multiagent Systems (DAIMAS'97)', St. Petersburg, Russia, pp. 162–173.

E.Oliveira, N.Morujao, T. & J.Neves (1994), A Decision Support System for Blood Gas Analysis, *in* 'Conference of The Second World Congress on Expert Systems', Lisbon, Portugal.

E.Oliveira & R.Camacho (1991), 'A Shell for Cooperating Expert Systems', *Expert Systems Int. Journal of Knowledge Engineering* .

E.Shortliffe (1992), Doctors, Patients, and Computers: Will Information Technology Dehumanize Health-Care Delivery?, *in* 'Proceedings of the American Philosophical Society', pp. 390–398.

E.Shortliffe (1993), Health Care Professional Workstations: Where Are We Now?...Where Should We Be Tomorrow, *in* 'Workshop on Health Care Professional Workstations', Washington, D.C., U.S.A.

E.Werner (1992), The Design of Multi-Agent Systems, *in* E.Werner & Y.Demazeau, eds, 'Decentralized A.I. 3, Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World', North-Holland, pp. 3–28.

E.Werner & Y.Demazeau, eds (1992), *Decentralized A.I. 3, Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, North-Holland.

F.Brazier, B.Dunin-Keplicz, N.Jennings & J.Treur (1996), Modelling Distributed Industrial Processes in a Multi-Agent Framework, *in* S.Kirn & G.O'Hare, eds, 'Cooperative Knowledge Processing', Springer Verlag.

F.Brazier, B.Keplicz, N.Jennings & J.Treur (1993), Formal Specification of Multi-Agent Systems: a Real-World Case, *in* 'First International Conference on Multi-Agent Systems (ICMAS'95)', San Francisco, CA, U.S.A.

F.Cheong (1992), OASIS: An Agent-Oriented Programming Language for Heterogeneous Distributed Environment, PhD thesis, The University of Michigan, U.S.A.

F.Cheong (1996), *Internet Agents - Spiders, Wanderers, Brokers, and Bots*, New Riders Publishing.

F.Gouveia (1992), Contribution a l´Etude des Mecanismes de Coordination dans les Systemes d´Agents Autonomes, PhD thesis, Universite de Technologie de Compiegne, France.

F.Hayes-Roth, D.Waterman & D.Lenat, eds (1983), *Bulding Expert Systems*, Addison-Wesley Publishing Company, Inc.

F.Londono (1990), A Blackboard Framework to Support Concurrent Engineering, PhD thesis, Concurrent Engineering Research Center, West Virginia University.

F.Londono, K.Cleetus & Y.Reddy (1989), A Blackboard Scheme for Cooperative Problem Solving by Humans Experts, Technical report, Concurrent Engineering Research Center, West Virginia University.

F.McCabe & K.Clarck (1995), April - Agent PRocess Interaction Language, *in* 'Lecture Notes in Artificial Intelligence (890)', Springer-Verlag, pp. 324–340.

F.Mouta (1996), Ambiente de Desenvolvimento de Sistemas Multiagente para Controlo e Supervisão de Processos em Aplicaçcões Distribuídas, PhD thesis, Faculdade de Engenharia, Universidade do Porto.

F.Polat, S.Shekhar & H.Guvenir (1993), 'Distributed Conflict Resolution Among Cooperating Expert Systems', *Expert Systems* **10**(4), 227–236.

F.Santos & O.Belo (1993), D-Gene, Um Laboratório de Sistemas de Classificação baseado no Arquitectura de Quadros Negros, Technical report, Departamento de Informática, Universidade do Minho.

G.Agha (1986), *Actors: A Model of Concurrent Computation in Distributed Systems*, MIT Press.

G.Booch (1994), *Object-Oriented Analysis and Design with Applications*, 2 edn, The Benjamin/Cummings Publishing Company, Inc.

G.Kiss (1991), Autonomous Agents, AI and Chaos Theory, *in* J.Meyer & S.Wilson, eds, 'Proceedings of the First International Conference on Simulation of Adaptative Behavior', Cambridge, Massachusetts.

G.Luger & W.Stubblefield (1989), *Artificial Intelligence and the design of Expert Systems*, The Benjamin/Cummings Publishing Company, Inc.

G.O'Hare & N.Jennings, eds (1988), *Foundations of Distributed Artificial Intelligence*, Jonh Wiley & Sons.

G.Winstanley, ed. (1991), *Artificial Intelligence in Engineering*, Jonh Wiley & Sons.

H.Coelho (1995*a*), *Inteligência Artificial em 25 Lições*, Fundação Calouste Gulbenkian.

H.Coelho (1995*b*), *Sonho e Razão ao Lado do Artificial*, Círculo de Leitores.

H.Hahn & R.Stout (1994), *The Internet Complete Reference*, Osborne McGraw Hill.

H.Phaff, M.Miller & E.Mrak (1978), *The Life of Yeasts*, 2 edn, Harvard University Press.

H.Voss (1996), Dimensions of KBS Applications, *in* '2nd Knowledge Engineering Forum (Topic: Distributed Expertise)', University of Karlsruhe, Germany.

I.Bratko (1990), *Prolog Programming for Artificial Intelligence*, 2 edn, Addison-Wesley Publishing Company, Inc.

I.Brogi, F.Turini & M.Gaspari (1991), Inheritance Hierarchies in Blackboard Architectures, *in* M.Lenzerini, D.Nardi & M.Simi, eds, 'Inheritance Hierarchies in Knowledge Representation and Programming Languages', Wiley, pp. 257–274.

I.Craig (1987*a*), An Overview of Cassandra-II, Research Report CS-RR-92, Department of Computer Science, University of Warwick, Coventry, UK.

I.Craig (1987*b*), Cassandra-II: A Distributed Blackboard System, Research Report CS-RR-90, Department of Computer Science, University of Warwick, Coventry, UK.

I.Craig (1991*a*), Extending Cassandra, Research Report CS-RR-183, Department of Computer Science, University of Warwick, Coventry, UK.

I.Craig (1991*b*), Making Cassandra Parallel and Distributed, Research Report CS-RR-180, Department of Computer Science, University of Warwick, Coventry, UK.

I.Craig (1992), The New Implementation of Cassandra, Research Report CS-RR-233, Department of Computer Science, University of Warwick, Coventry, UK.

I.Craig (1993), A New Interpretation of the Blackboard Architecture, Research Report CS-RR-254, Department of Computer Science, University of Warwick, Coventry, UK.

I.Craig (1994*a*), A Perspective on Multi-Agent Systems, Research Report CS-RR-273, Department of Computer Science, University of Warwick, Coventry, UK.

I.Craig (1994*b*), Agents That Model Themselves, Research Report CS-RR-266, Department of Computer Science, University of Warwick, Coventry, UK.

J.Almgran & S.Andersson (1993), *SICStus Prolog Library User's Manual 2.1 #8*, Swedish Institute of Computer Science.

J.Almgren, S.Andersson, M.Carlsson, L.Flood, S.Haridi, C.Frisk, H.Nilsson & J.Sundberg (1993), *SICStus Prolog Library Manual*, Swedish Institute of Computer Science.

J.Barnett, R.Payne & D.Yarrow, eds (1990), *Yeasts, Characteristics and Identification*, Cambridge University Press.

J.Bates (1992*a*), The Nature of Character in Interactive Worlds and The Oz Project, Technical Report CMU-CS-92-200, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

J.Bates (1992*b*), 'Virtual Reality, Art, and Entertainment', *Presence: The Journal of Teleoperators and Virtual Environments* **1**(1), 133–138.

J.Bates (1994), The Role of Emotion in Believable Agents, Technical Report CMU-CS-94-136, School of Computer Science, Carnegie Mellon University, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

J.David, J.Krivine & R.Simmons (1993), Second Generation Expert Systems: A Step Forward in Knowledge Engineering, *in* 'Second Generation Expert Systems', Springer-Verlag, pp. 3–23.

J.deKleer (1986*a*), 'An Assumption-based TMS', *Artificial Intelligence* **28**, 127–162.

J.deKleer (1986*b*), 'Problem Solving with the ATMS', *Artificial Intelligence* **28**, 197–224.

J.Doyle (1979), 'A Truth Maintenance System', *Artificial Intelligence* **12**, 231–272.

J.Fonseca and E.Oliveira and A.Garcão (1996), MACIV- A DAI Based Resource Management System, *in* 'Proceedings of PAAM96, The Practical Application of Intelligent Agents and Multi-Agent Technology', London, UK.

J.Giarratano & G.Riley (1994), *Expert Systems - Principles and Programming*, PWS Publishing Company.

J.Gilmore, S.Roth & S.Tynor (1989), A Blackboard System for Distributed Problem Solving, *in* V.Jagannathan, R.Dodhiawala & L.Baum, eds, 'Blackboard Architectures and Applications', Academic Press, Inc., chapter 17, pp. 371–393.

J.Huang, N.Jennings & J.Fox (1995*a*), An Agent Architecture for Distributed Medical Care, *in* M.Wooldridge & N.Jennings, eds, 'Intelligent Agents', Lecture Notes in Artificial Intelligence, Springer Verlag, pp. 219–232.

J.Huang, N.Jennings & J.Fox (1995*b*), 'An Agent-based Approach to Health Care Management', *Applied Artificial Intelligence: An International Journal* **9**(4), 401–420.

J.King (1995), 'Intelligent Agents: Bringing Good Things to Life', *AI Expert* **10**(2), 17–19.

J.K.Lee, J.Liebowitz & Y.M.Chae, eds (1996), Vol. I, II, Seoul, Korea.

J.Kolodner (1991), 'Improving Human Decision Making through Case-Based Reasoning Aiding', *AI Magazine* **12**(2), 52–68.

J.Martins (1990), 'The Truth, the Whole Truth, and Nothing But the Truth', *AI Magazine* pp. 7–25.

J.Müller & M.Pischel (1994), Modelling interacting agents in dynamic environments, *in* 'Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-94)', Amsterdam, The Netherlands, pp. 709–713.

J.Müller, M.Pischel & M.Thiel (1994), Modeling Reactive Behaviour in Vertically Layered Agent Architectures, *in* M.Wooldridge & N.Jennings, eds, 'Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages (ATAL'94)', Amsterdam, The Netherlands.

J.Neves & A.Ribeiro (1994), Reactive Systems and the Simulation of Industrial Processes - A Dynamic Logic View, *in* A.Kaylan, A.Lehmann & F.Oren, eds, 'Proceedings of the European Simulation Symposium (ESS 94)', Vol. II, Istanbul, Turkey, pp. 80–84.

J.Neves & J.Machado (1997), Formalizing Context in Knowledge Management Systems, *in* 'Proceedings of the International Workshop on Distributed Artificial Intelligence and Multiagent Systems (DAIMAS'97)', St. Petersburg, Russia, pp. 154–161.

J.Neves, J.Machado, A.Abelha & P.Novais (1996), A Multi-Agent Framework for System Simulation, *in* 'Proceedings of the 8th European Simulation Symposium (ESS'96)', Genoa, Italy.

J.Rocha, C.Ramos & Z.Vale (1995), Sequencing Operations for Process Planning, *in* 'Proceedings of the Conference of The Practical Application of Prolog (PAP'95)', Paris, France, pp. 501–510.

J.Schlimmer & L.Hermens (1993), 'Software Agents: Completing Patterns and Constructing User Interfaces', *Journal of Artificial Intelligence Research* **1**, 61–89.

J.Szép & F.Forgó (1985), *Introduction to the Theory of Games*, D.Reidel Publishing Company.

J.Vargaas & C.Kee (1994), 'Improving the Scope of Intelligent Tutoring by Adapting a Case-Based Methodology through a Distributed Architecture', *Applied Artificial Inteligence* **8**(3), 413–424.

K.Brookes (1987), *World Directory and Handbook of Hardmetals*, 4 edn, International Carbide Data.

K.Sycara (1989), Multiagent Compromise via Negotiation, *in* L.Gasser & M.Huhns, eds, 'Distributed Artificial Intelligence', Pitman and Morgan Kaufmann, Inc, chapter 6, pp. 119–137.

L.Baum, R.Dodhiawala & V.Jagannathan (1989), A Blackboard System for Distributed Problem Solving, *in* V.Jagannathan, R.Dodhiawala & L.Baum, eds, 'The Erasmus System', Academic Press, Inc., chapter 16, pp. 347–370.

L.Cavedon & G.Tidhar (1995), A Logical Framework for Multi-Agent Systems and Joint Attitudes, *in* 'Proceedings of the First Australian Workshop on Distributed Artificial Intelligence', Canberra, Australia, pp. 16–30.

L.Foner (1993), What's An Agent, Anyway? - A Sociological Case Study, Technical report, Agents Group, MIT Media LAb.

L.Gasser & M.Huhns, eds (1989), *Distributed Artificial Intelligence*, Vol. II, Pitman, Morgan Kaufmann Publishers, Inc.

L.Gould (1995), 'If AI Ran the Zoo', *BYTE, The Magazine of Technology Integration* **20**(12), 79–83.

Linda-Like Systems and Their Implementation (1991), Technical report, Edinburgh Parallel Computing Centre.

L.Raedt & M.Bruynooghe (1992), 'Belief updating from integrity constraints and queries', *Artificial Intelligence* **53**(2-3), 291–307.

M.Akhun & M.Durmusoglu (1994), Worker Allocation Problem in Cellular Manufacturing: A Case Study, *in* A.Kaylan, A.Lehmann & T.Oren, eds, 'Proceedings of the European Simulation Symposium', Vol. II, Istanbul, Turkey, pp. 105–109.

M.Carlsson & J.Widen (1993), *SICStus Prolog User's Manual*, Swedish Institute of Computer Science.

M.Coen (1994), SodaBot: A Software Agent Environment and Construction System, Technical Report A.I.Technical Report 1493, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

M.Dojat, L.Brochard, F.Lemaire & A.Harf (1992), 'A knowledge-based system for assisted ventilation of pations in intensive care units', *International Journal of Clinical Monitoring and Computing* **9**, 239–250.

M.Fayad, W.Tsai & M.Fulghum (1996), 'Transition to Object-Oriented Software Development', *Communications of the ACM* **21**(2), 108–121.

M.Ginsberg (1993), *Essentials of Artificial Intelligence*, Morgan Kaufmann Publishers.

M.Hardwick, D.Spooner, T.Rando & K.Morris (1996), 'Sharing Manufacturing Information in Virtual Enterprises', *Communications of the ACM* **21**(2), 46–54.

M.Huhns & D.Bridgeland (1991), 'Multiagent Truth Maintenance', *IEEE Transactions on Systems, Man, and Cybernetics* **21**(6), 1437–1445.

M.JonhsonJr. & B.Hayes-Roth (1987), Integrating Diverse Reasoning Methods in the BB1 Blackboard Control Architecture, *in* 'Sixth National Conference on Artificial Intelligence (AAAI'87)', Vol. 1, Seattle, Washington, U.S.A., pp. 30–35.

M.Luck & M.d'Inverno (1995), Agency and Autonomy: A Formal Framework, Research Report CS-RR-276, Department of Computer Science, University of Warwick, Coventry, UK.

M.Meyer & J.Booker, eds (1991), *Eliciting and Analysing Expert Judgement - A Practical Guide*, Academic Press.

M.Minsky (1985), A Framework for Representing Knowledge, *in* R.Brachman & H.Levesque, eds, 'Readings in Knowledge Representation', Morgan Kaufmann Publishers, Inc, pp. 245–262.

M.Minsky (1986), *The Society of Mind*, Simon and Schustert, New York.

M.Pires, J.Dias & O.Belo (1997), Using Multi-Agent Systems Technology on Cooperative Traffic Control Simulation - A Case Study, *in* 'Proceedings of the 16th IASTED International Conference on Modelling, Identification and Control', Innsbruck, Austria.

M.Wooldridge, J.Muller & M.Tambe, eds (1996), *Intelligent Agents II, Proceedings of the IJCAI'95, Workshop on Agent Theories, Architectures, and Languages*, Lecture Notes in Artificial Intelligence (1037), Springer-Verlag.

M.Wooldridge & N.Jennings (1995*a*), 'Intelligent agents: theory and practice', *Knowledge Engineering Review 10* .

M.Wooldridge & N.Jennings, eds (1995*b*), *Intelligent Agents, Proceedings of the ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, Lecture Notes in Artificial Intelligence (890), Springer-Verlag.

M.Zhang & C.Zhang (1994*a*), ' A comprehensive synthesis strategy for conflict resolutions in distributed expert systems', *Australian Journal of Intelligent Information Processing Systems* **1**(2), 21–28.

M.Zhang & C.Zhang (1994*b*), Synthesis of Solutions in Distributed Expert Systems, *in* C.Zhang, J.Debenham & D. Lukose, eds, 'Artificial Intelligence Sowing the Seeds for the Future', World Scientific, pp. 362–369.

N.Carriero & D.Gelernter (1989*a*), 'How to Write Parallel Programs: A Guide to the Perplexed', *ACM Computing Surveys* **21**(3).

N.Carriero & D.Gelernter (1989*b*), 'Linda in Context', *Communications of ACM* **32**(4), 444–458.

N.Carver & V.Lesser (1992), The Evolution of Blackboard Control Architectures, Technical Report 92-71, CMPSCI.

N.Flann, T.Dietterich & D.Corpron (1987), Forward Chaining Logic Programming with the ATMS, *in* 'Sixth National Conference on Artificial Intelligence (AAAI'87)', Vol. 1, Seattle, Washington, U.S.A., pp. 24–29.

N.Jennings (1992*a*), Towards a Cooperation Knowledge Level For Collaborative Problem Solving, *in* B.Neumann, ed., '10th European Conference on Artificial Intelligence (ECAI'92)', Jonh Wiley & Sons, Ltd, pp. 224–228.

N.Jennings (1992*b*), Using GRATE to Build Cooperating Agents for Industrial Control, *in* 'Proceedings of IFAC/IFIP/IMACS Int. Sym. on Artificial Intelligence in Real Time Control', Delft, The Netherlands, pp. 691–696.

N.Jennings (1993*a*), 'Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems', *The Knowledge Engineering Revue* **8**(3), 223–250.

N.Jennings (1993*b*), 'Specification and Implementation of a Belief-Desire-Joint-Intention Architecture for Collaborative Problem Solving', *Int. Journal of Intelligent and Cooperative Information Systems* **2**(3), 289–318.

N.Jennings (1994), The ARCHON System and its Applications, *in* 'Second International Working Conference on Cooperating Knowledge Based Systems (CKBS-94)', pp. 13–29.

N.Jennings (1995*a*), Agent Software, *in* 'Proc. UNICOM Seminar on Agent Software', London, UK, pp. 12–27.

N.Jennings (1995*b*), 'Controlling cooperative problem solving in industrial multi-agent systems using joint intentions', *Artificial Intelligence* **74**(2).

N.Jennings (1996), Coordination Techniques for Distributed Artificial Intelligence, *in* 'Foundations of Distributed Artificial Intelligence', John Wiley & Sons, Inc, pp. 187–210.

N.Jennings, E.Mamdani, I.Laresgoiti, J.Perez & J.Corera (1992), 'GRATE: A General Framework for Cooperative Problem Solving', *IEE-BCS Journal of Intelligent Systems Engineering* **1**(2), 102–114.

N.Jennings & J.Pople (1993), Design and Implementation of ARCHON´s Coordination Module, *in* 'Workshop on Cooperative Knowledge Based System', Keel, UK.

N.Jennings, L.Varga, R.Aarnts, J.Fuchs & P.Skarek (1993), 'Transforming Standalone Expert Systems into a Comunity of Cooperatig Agents', *Int. Journal of Engineering Applications of Artificial Intelligence* **6**(4), 317–331.

N.Jennings & M.Wooldridge (1995), 'Applying Agent Technology', *Applied Artificial Intelligence: An International Journal* **9**(4), 351–361.

N.Jennings & M.Wooldridge (1996), 'Software Agents', *IEE Review* pp. 17–20.

N.Jennings, P.Faratin, M.Johnson, P.O'Brien & M.Wiegand (1996), Using Intelligent Agents to Manage Business Processes, *in* 'Proceedings of First International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96)', London, UK.

N.Jennings & T.Wittig (1992), ARCHON: Theory and Practice, *in* N. Avouris & L. Gasser, eds, 'Distributed Artificial Intelligence: Theory and Praxis', Kluwer Academic Press, pp. 179–195.

O.Baujard, S.Pesty & C.Garbay (1992), A Programming Environment for Distributed Applications Design in Artificial Intelligence, *in* 'Applications of Artificial Intelligence X: Knowledge-based Systems', SPIE The

International Society for Optical Engineering, Orlando, USA, pp. 110–116.

O.Baujard, S.Pesty & C.Garbay (1993), 'MAPS: a Language for Multi-Agent System Design', *Expert Systems* **11**(2), 89–97.

O.Belo (1991*a*), Planeamento e Concepção de Sistemas Periciais, PAPCC, Vol(1), PC, Technical report, Departamento de Informática, Universidade do Minho.

O.Belo (1991*b*), Sistemas Periciais, PAPCC, Vol(2), PP, Technical report, Departamento de Informática, Universidade do Minho.

O.Belo (1993), Expertys, Uma Shell para a Implementação de Sistemas Periciais, Technical report, Departamento de Informática, Universidade do Minho.

O.Belo (1994*a*), BeAble: Um Ambiente Distribuído para a Implementação de Sistema Multi-Agente, Technical report, Departamento de Informática, Universidade do Minho.

O.Belo (1994*b*), D-Expertys, Uma Plataforma Distribuída para a actividade de Agentes, Technical report, Departamento de Informática, Universidade do Minho.

O.Belo (1994*c*), Uma Estratégia para a Optimização do Desempenho de Agentes Orientados por Regras, Technical report, Departamento de Informática, Universidade do Minho.

O.Belo (1996), A Hardmetal Tools and Wear Parts Production System Simulation, *in* 'Proceedings of the 8th European Simulation Symposium

(ESS'96)', Genoa, Italy. This article was considered as one of the winners of the "Best Paper Award"of the ESS'96.

O.Belo (1997), Integração de Técnicas de Computação Baseada em Agentes no Desenvolvimento de Aplicações Industriais, *in* 'Actas do Terceiro Encontro Nacional do Colégio de Engenharia Electrotécnica da Ordem dos Engenheiros', Porto, Portugal.

O.Belo & A.Ribeiro (1995), A Web-Based Framework for Distributed Expert Systems, *in* '1st Portuguese WWW Conference (CNW3)', Universidade do Minho, Braga, Portugal.

O.Belo & A.Ribeiro (1996), A Distributed Multi-Agent System Environment, A Web Approach, *in* 'IASTED International Conference on Artificial Intelligence, Expert Systems and Neural Networks', Honolulu, Hawaii, USA.

O.Belo & J.Neves (1995*a*), A Prolog Implementation of a Distributed Computing Environment for Multi- Agent Systems Based Applications, *in* 'Proceedings of the Conference of The Practical Application of Prolog (PAP'95)', Paris, France, pp. 31–41.

O.Belo & J.Neves (1995*b*), Consistency Revision in a Distributed Multi-Agent System Environment, *in* 'Proceedings of the First International Workshop on Decentralized Intelligent and Multi-Agent Systems', Kracow, Poland, pp. II/41–II/49.

O.Belo & J.Neves (1995*c*), Cooperation Among Opportunistic Agents on a Distributed Environment, *in* 'Proceedings of the First Australian Workshop on Distributed Artificial Intelligence', Canberra, Australia, pp. 205–216.

O.Belo & J.Neves (1995*d*), Multi-Agent Systems Based Distributed Intelligent Simulation - A Case Study, *in* 'Proceedings of the Eurosim Congress '95', Vienna, Austria, pp. 1235–1240.

O.Belo & J.Neves (1996*a*), A Distributed Problem Solving Environment for Multi-Agent Systems, *in* 'Conference of The Third World Congress on Expert Systems', Seoul, Korea, pp. 815–822.

O.Belo & J.Neves (1996*b*), Cooperation Among Opportunistic Agents on a Distributed Environment, *in* C. Zhang & D. Lukose, eds, 'Lecture Notes in Artificial Intelligence (1087). Distributed Artificial Intelligence Architecture and Modelling, First Australian Workshop on DAI, Canberra, ACT, Australia, November 13, 1995. Proceedings', Springer-Verlag.

O.Belo & J.Neves (1997*a*), An Architecture for Multi-Agent Systems Communities, *in* 'Proceedings of the 10th International FLAIRS Conference', Florida, USA.

O.Belo & J.Neves (1997*b*), Multi-Agent Systems Interaction Through Intelligent Routing Services, *in* 'Proceedings of the IEEE First International Conference on Intelligent Processing Systems', Beijing, China.

O.Etzioni & D.Weld (1994), 'A Softbot-Based Interface to the Internet', *Communications of the ACM* **37**(7), 72–76.

P.Camacho (1996), Um Sistema Pericial para a Classificação de Leveduras, Technical report, Departamento de Informática, Universidade do Minho.

P.Edwards & W.Davies (1993), A Heterogeneous Multi-Agent Learning System, Technical report, Research Report, Department of Computing Science, University of Aberdeen.

P.Gmytrasiewicz, E.Durfee & D.Wehe (1991), The Utility of Communication in Coordinating Intelligent Agents, *in* 'Proceedings of the 9th National Conference on Artificial Intelligence', pp. 166–172.

P.Grogono, A.Preece, R.Shinghal & C.Suen (1994), Building Expert Systems: From Specification to Evaluation, *in* 'Conference of The Second World Congress on Expert Systems', Lisbon, Portugal, pp. 287–292.

P.Maes (1994), 'Agents that Reduce Work and Information Overload', *Communications of the ACM* **37**(7).

P.Maes, ed. (1990), *Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back*, MIT / Elsevier.

P.Mossinger, D.Polani, R.Spalt & T.Uthmann (1995), XRaptor: A Synthetic Multi-Agent Environment for Evaluation of Adaptive Control Mechanisms, *in* 'In International Conference on Building and Sharing of Very Large- Scale Knowledge Bases', North-Holland, pp. 1229–1234.

P.Nii (1986), 'Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures', *AI Magazine* **7**(2), 38–53.

P.Nii (1989), Introduction, *in* V. Jagannathan, R. Dodhiawala & L. Baum, eds, 'Blackboard Architectures and Applications', Academic Press, Inc., pp. xix–xxix.

R.Beale & A.Wood (1994), Agent-Based Interaction, *in* 'People and Computers IX: Proceedings of HCI'94', British Computer Society HCI Group, Cambridge University Press, pp. 239–245.

R.Brooks (1991), 'Intelligence without representation', *Artificial Intelligence* **47**, 139–159.

R.Davis (1992), 'From local to global consistency', *Artificial Intelligence* **55**(1), 87–107.

R.Davis, B.Buchanan & E.Shortliffe (1985), Production Rules as a Representation for a Knowledge-Based Consultation Program, *in* R.Brachman & H.Levesque, eds, 'Readings in Knowledge Representation', Morgan Kaufmann Publishers, Inc, pp. 371–387.

R.Dechter (1992), 'From local to global consistency', *Artificial Intelligence* **55**(1), 87–107.

R.Engelmore, AMorgan & H.P.Nii (1988*a*), HEARSAY-II, *in* R.Engelmore & T.Morgan, eds, 'Blackboard Systems', Addison-Wesley Publishing Company, Inc., chapter 2.

R.Engelmore, AMorgan & H.P.Nii (1988*b*), Introduction, *in* R.Engelmore & T.Morgan, eds, 'Blackboard Systems', Addison-Wesley Publishing Company, Inc., chapter 1.

R.Engelmore & T.Morgan (1988), *Blackboard Systems*, Addison-Wesley Publishing Company, Inc.

R.Fagin, J.Halpern, Y.Moses & M.Vardi (1995), *Reasoning About Knowledge*, Massachusetts Institute of Technology.

R.Gibbons (1992), *A Primer in Game Theory*, Harvester Wheatsheaf.

R.Goodwin (1993), Formalizing Properties of Agents, Technical report, School of Computer Science, Carnegie Mellon University.

R.Simmons & R.Davis (1993), The Roles of Knowledge and Representation in Problem Solving, *in* 'Second Generation Expert Systems', Springer-Verlag, pp. 27–45.

R.Smith (1988), The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, *in* A. Bond & L. Gasser, eds, 'Readings in Distributed Artificial Intelligence', Morgan Kaufmann Publishers, Inc, pp. 357–366.

R.Smith & R.Davis (1988), Frameworks for Cooperation in Distributed Problem Solving, *in* 'Readings in Distributed Artificial Intelligence', Morgan Kaufmann Publishers, Inc, pp. 61–70.

R.Steeb, D.McArthur, S.Cammarata, S.Narain & W.Giarla (1986), Distributed Problem Solving for Air Fleet Control: Framework and Implementation, *in* P.Klahr & D.Waterman, eds, 'Expert Systems Techniques, Tools and Applications', Addison-Wesley Publishing Company, pp. 391–429.

R.Unland, S.Kirn, U.Wanka, G.O´Hare & S.Abbas (1995), 'AEGIS: Agent Oriented Organisations', *Accounting, Management and Information Technologies* **5**(2), 139–162.

S.Ahuja, N.Carriero & D.Gelertner (1986), 'Linda and Friends', *Computer* **19**(8), 26–34.

S.Biondo (1990), *Fundamentals of Expert Systems Technology, Principles and Cocepts*, Ablex Publishing Corporation.

S.Cammarata, D.McArthur & R.Steeb (1988), Strategies of Cooperation in Distributed Problem Solving, *in* 'Readings in Distributed Artificial Intelligence', Morgan Kaufmann Publishers, Inc, pp. 102–105.

S.Chakravarthy, K.Karlapalem, S.Navathe & A.Tanaka (1992), Database Supported Cooperative Problem Solving, Technical report, Computer and Information Sciences, University of Florida.

S.Garfinkel & G.Spafford (1991), *Practical Unix Security*, O' Reilly & Associates, Inc.

S.Goksenand, T.Dayioglu, E.Dorman, S.Imer & A.Kaylan (1994), Productivity Improvement Via Simulation at a Garment Factory, *in* A.Kaylan, A.Lehmann & T.Oren, eds, 'Proceedings of the European Simulation Symposium', Vol. II, Istanbul, Turkey, pp. 110–114.

S.Goodman, L.Press, S.Ruth & A.Rutkowski (1994), 'The Global Diffusion of the Internet: Patterns and Problems', *Communications of the ACM* **37**(8), 27–31.

S.Kirani, I.Zualkernan & W.Tsai (1994), 'A Softbot-Based Interface to the Internet', *Communications of the ACM* **37**(11), 71–81.

S.Mullender, ed. (1990), *Distributed Systems*, 2 edn, Addison-Wesley Publishing Company.

S.Russel & P.Norvig (1995), *Artificial Intelligence - A Modern Approach*, Prentice Hall.

S.Shapiro, ed. (1990*a*), *Encyclopedia of Artificial Intelligence*, Vol. I, Jonh Wiley & Sons.

S.Shapiro, ed. (1990*b*), *Encyclopedia of Artificial Intelligence*, Vol. II, Jonh Wiley & Sons.

S.Slade (1991), 'Case-Based Reasoning: A Research Paradigm', *AI Magazine* **12**(1), 42–55.

SunNPG (1990), *Network Programming Guide*.

T.Berners-Lee, R.Cailliau, A.Luotonen, H.Nielsen & A.Secret (1994), 'The World Wide Web', *Communications of the ACM* **37**(8), 27–31.

T.Berners-Lee, R.Cailliau, J.Groff & B.Pollerman (1992), 'World-Wide Web: The Information Universe', *Electronic Networking: Research, Applications and Policy* **2**(1), 52–58.

T.Finin, D.McKay, R.Fritzon & R.McEntire (1993), KQML: An Information and Knowledge Exchange Protocol, *in* 'In International Conference on Building and Sharing of Very Large- Scale Knowledge Bases'.

T.Finin, R.Fritzon, D.McKay & R.McEntire (1994), KQML - A Language and Protocol for Knowledge and Information Exchange, Technical Report CS-94-02, University of Maryland, Computer Science Department.

T.Katz (1991), Engineering Design, Manufacture and Test, *in* G.Winstanley, ed., 'Artificial Intelligence in Engineering', Jonh Wiley & Sons, chapter 5, pp. 151–193.

T.Mitchell, R.Caruana, D.Freitag, J.McDermott & D.Zabowski (1994), 'Experience with a Learning Personal Assistant', *Communications of the ACM* **37**(7), 80–91.

T.Selker (1994), 'Coach: A Teaching Agent that Learns', *Communications of the ACM* **37**(7), 92–99.

T.Wittig, ed. (1992), *ARCHON an architecture for multi-agent systems*, Ellis Horwood Limited.

T.Wittig, N.Jennings & E.Mamdani (1994), 'ARCHON - A Framework for Intelligent Cooperation', *IEE-BCS Journal of Intelligent Systems Engineering - Special Issue on Real-time Intelligent Systems in ESPRIT* **3**(3), 168–179.

T.Yurtsever (1994), Factory Floor Simulation Controlled by Fuzzy Expert System, *in* A.Kaylan, A.Lehmann & T.Oren, eds, 'Proceedings of the European Simulation Symposium', Vol. I, Istanbul, Turkey, pp. 277–281.

V.Alves, A.Ribeiro & J.Neves (1993), Distributed Problem Solving - A Universal Computer Architecture, *in* 'Proceedings of the 5th Workshop on Logic Programming Environments', Vancouver, Canada, pp. 88–93.

V.Jagannathan, R.Dodhiawala & L.Baum, eds (1989), *Blackboard Architectures and Applications*, Academic Press, Inc.

V.Lesser (1990), 'An Overview of DAI: Viewing Distributed AI as Distributed Search', *Journal of Japonese Society for Artificial Intelligence-Special Issue on Distributed Artificial Intelligence* **5**(4), 392–400.

V.Lesser (1995), 'Multiagent Systems: An Emerging Subdiscipline of AI', *Computing Surveys AI Symposium* .

W.Davies & P.Edwards (1994), Agent-K: An Integration of AOP and KQML, Technical report, Research Report, Department of Computing Science, University of Aberdeen.

W.Karbach & A.Vobeta (1993), MODEL-K for prototyping and strategic reasoning at the knowledge level, *in* 'Second Generation Expert Systems', Springer-Verlag, pp. 721–745.

W.Leler (1990), 'Linda Meets Unix', *Computer* **23**(2), 43–54.

X.Yan (1995), An Advanced Simulation Environment for Modular Manufactering Systems, *in* 'Proceedings of the Eurosim Congress '95', Vienna, Austria, pp. 1017–1028.

Y.Demazeau & J.Muller (1990*a*), Decentralized Artificial Intelligence, *in* 'Decentralized A.I.', North-Holland, pp. 3–13.

Y.Demazeau & J.Muller, eds (1990*b*), *Decentralized A.I.*, North-Holland.

Y.Demazeau & J.Muller, eds (1991), *Decentralized A.I. 2*, North-Holland.

Y.Lashkari, M.Metral & P.Maes (1994), Collaborative Interface Agents, *in* 'Proceedings of AAAI '94 Conference', Seattle, Washington.

Apêndice A

Sistemas de Referência

A.1 ARCHON

O sistema ARCHON[1] (*"ARchitecture for Cooperative Heterogeneous ON-line systems"*) teve a sua génese num projecto ESPRIT na área da IAD, cujo objectivo era criar uma arquitectura de aplicação genérica, um enquadramento e uma metodologia para o desenvolvimento de sistemas de IAD com aplicação a situações industriais reais.

Duas das aplicações desenvolvidas com base no ARCHON - a gestão de transporte de energia eléctrica e o controlo de um acelerador de partículas - foram executadas com sucesso nas organizações para as quais haviam sido concebidas, a Iberdrola, uma empresa de electricidade no norte de Espanha, e o CERN, perto de Genebra. Outras aplicações foram ainda desenvolvidas para a Amber, para o controlo de uma cimenteira na Grécia, e em Portugal, na Faculdade de Engenharia da Universidade do Porto, para o controlo de uma célula robótica de manipulação de objectos.

[1] http://www.elec.qmw.ac.uk/dai/research.projects/archon/test_1.html

As entidades de resolução do ARCHON são designadas por agentes. Estes têm a capacidade de controlar as suas próprias acções e de interagir com outros membros da comunidade. As interacções envolvem tipicamente agentes a cooperar e a comunicar entre si, no sentido de melhorarem as suas próprias capacidades de resolução de problemas. Cada agente é constituído por uma camada de cooperação ARCHON (AL) e por um programa aplicacional denominado como Sistema Inteligente (IS). Uma distinção clara entre o "know-how" social do agente e o seu nível de conhecimento acerca do domínio da aplicação torna a solução ARCHON flexível e aberta, impondo relativamente poucas restrições ao construtor da aplicação (N.Jennings & T.Wittig 1992) (T.Wittig 1992) (N.Jennings 1994) (T.Wittig et al. 1994).

A.2 COSY

O sistema COSY (*"COoperative SYstems"*) apresenta uma arquitectura híbrida, incluindo elementos provenientes dos sistemas PRS e IRMA. O COSY foi especialmente desenvolvido para um SM de teste, denominado por DASEDIS.

A arquitectura funcional dos agentes do sistema desenvolve-se basicamente em torno de cinco módulos: sensores, actuadores, unidade de comunicação, unidade de cognição e unidade de intenções. Os sensores recebem dados não comunicáveis, os actuadores permitem ao agente executar acções não comunicáveis e a unidade de comunicação permite que o agente envie mensagens. A unidade de intenções contém objectivos, atitudes, responsabilidades e os elementos de controlo que tomam parte nos processos de raciocínio e tomada de decisão da unidade de cognição. Esta é a responsável pela mediação entre

as intenções e as crenças do agente acerca do mundo, e pela selecção das acções a executar. Na unidade de cognição está contida a base de conhecimento, que integra as crenças do agente e três componentes de execução: 1) um componente para execução de guiões ("scripts"); 2) um componente para a execução de protocolos; 3) um componente para a execução de raciocínios, decisões e reacções.

Um guião é um recipiente ou um plano estereotipado para a representação de conhecimento, orientado para a realização de um objectivo. Os protocolos são diálogos estereotipados representando enquadramentos para cooperação do tipo da encontrada nas redes de contrato (R.Smith 1988). O terceiro componente, o componente chave do sistema, é constituído por uma série de subsistemas. O sistema mantém uma agenda que contém um número de guiões activos. Estes podem ser invocados de duas formas: por objectivos ("goal-driven"), no sentido de satisfazer uma das intenções do agente, ou por dados ("data-driven"), como resposta à situação corrente do agente (B.Burmeister & K.Sundermeyer 1992).

A.3 HECODES

O sistema HECODES (*"HEterogeneous COoperative Distributed Expert Systems"*) é um enquadramento que suporta a cooperação entre SP heterógeneos distribuídos. A arquitectura do sistema define-se em torno de um sistema de controlo centralizado e de um conjunto de processadores frontais ("front-end processors"). Cada subsistema em HECODES é constituído por três módulos funcionais: o módulo de gestão, o módulo de comunicação e o módulo de interface homem-máquina. O subsistema de controlo é o "coração" do HE-

CODES, sendo responsável pela temporização e escalonamento do sistema e pela gestão da cooperação e comunicação entre os SP.

A cooperação entre os SP desenvolve-se em torno de três elementos: 1) os cooperadores; 2) o gestor da cooperação e 3) um elemento de comunicação. Os cooperadores, como o próprio nome indica, são os agentes que cooperam. São SP que, conjuntamente, contribuem para a solução de um problema. A gestão da cooperação envolve duas partes: 1) um processador frontal, que é responsável pela gestão de um SP, e 2) um temporizador, no nodo de controlo, que faz não só a gestão dos SP com processadores frontais, mas também a gestão global do sistema. Os módulos de comunicação são responsáveis pela gestão de comunicação entre os nodos de controlo e os SP.

Os SP podem interactuar de diferentes formas, existindo mecanismos que permitem tratar vários tipos de formatos de entrada e saída de dados. As transformações são executadas no momento em que um SP solicita ajuda para a realização de uma tarefa, sendo esta invocada pelo temporizador. Os utilizadores podem definir as suas próprias funções de transformação para as subtarefas e colocar os nomes das funções na base de metaconhecimento do temporizador (C.Zhang 1990, C.Zhang 1992, C.Zhang 1993).

A.4 INTERRAP

Os agentes do sistema INTERRAP[2] (*"Integration of Reactive Behaviour and Rational Planning"*) apresentam uma arquitectura organizada por camadas ou níveis. Cada uma dessas camadas encontra-se subdividida em duas camadas verticais: uma contendo bases de conhecimento e outra os vários tipos de

[2]http://www.dfki.uni-sb.de/ pischel/interrap.htm

sistemas de controlo que regulam a interacção entre as bases de conhecimento da camada a que pertencem.

Na camada inferior está situado o sistema de controlo da interface com o mundo e a correspondente base de conhecimento, com o modelo do mundo. O sistema de controlo da interface com o mundo faz a gestão do interface entre o agente e o seu ambiente, lidando com problemas de actuação, comunicação e percepção.

Acima do componente de interface com o mundo está o componente baseado em comportamento. O propósito deste componente é implementar e controlar a capacidade básica de reacção do agente, a sua reactividade. Este componente manipula um conjunto de padrões de comportamento representados pela sigla PoB. Um PoB é uma estrutura que contém:

- uma pré-condição, que define quando é activado o PoB;
- várias condições, que definem as circunstâncias perante as quais o PoB é considerado como tendo falhado ou sucedido;
- uma pós-condição, que se mantém após o PoB ter sido executado com sucesso;
- um "corpo" executável, que define qual a acção a desencadear quando o PoB for executado.

Acima do componente baseado em comportamentos está o componente baseado em planos. Este componente contém um planificador que tem a capacidade de gerar planos para os agentes, em resposta aos pedidos do componente baseado em comportamentos.

A camada superior no sistema INTERRAP é constituída pelo componente de cooperação. Este componente tem a capacidade de gerar planos, seleccionados a partir de uma biblioteca, que satisfaçam os objectivos de um conjunto de agentes. Estes são gerados como resposta aos pedidos do componente baseado em planos.

O controlo no INTERRAP é orientado tanto por dados como por objectivos. Os dados de entrada são geridos pelo componente de interface com o mundo, resultando da modificação do modelo do mundo. Como resultado destas modificações, vários padrões de comportamento podem ser activados, interrompidos ou executados. Como resultado da execução do PoB, aos componentes de planos e de cooperação podem ser solicitados, respectivamente, planos e planos conjuntos, no sentido de se alcançarem os objectivos do agente. Isto resulta na geração de acções e mensagens pelo componente de interface com o mundo (J.Müller & M.Pischel 1994) (J.Müller et al. 1994).

A.5 MAPS

O sistema MAPS[3] (*"Multi-Agent Problem Solver"*) foi desenvolvido como um ambiente de programação para suportar o desenho de SM em domínios aplicacionais complexos, como a visão de computador, o diagnóstico biomédico ou o processamento da língua natural. O MAPS foi concebido como uma linguagem que permite a integração de diferentes paradigmas de programação e de modelos de representação de conhecimento.

Os agentes do MAPS são definidos como SBC, entidades autónomas capa-

[3]http://expasy.hcuge.ch/sgaico/html/olb/Sources/Maps/Maps.html

zes de reagir a acontecimentos externos, que comunicam com outros agentes através de processos de troca de mensagens. A representação de conhecimento do MAPS assenta num modelo misto, baseado em regras de produção e programação por objectos.

Duas classes de agentes foram desenvolvidas: os servidores de conhecimento e os processadores de conhecimento. Os primeiros são usados na modelação e na transmisão de conhecimento figurativo (conhecimento sobre as propriedades e relações presentes nos enunciados dos problemas), enquanto os segundos são utilizados na modelação e manipulação de conhecimento operativo, procedimentos que têm em linha de conta os elementos figurativos.

Os pedidos dos agentes são processados de acordo com os modos de comunicação, síncrono ou assíncrono, dependendo do tipo de agente envolvido. O modo assíncrono define um estilo de comunicação que envolve um sistema típico de "caixas de correio" e está adaptado ao processamento de pedidos de transmissão de informação. Enquanto que o modo síncrono define uma comunicação do tipo cliente/servidor, entre dois agentes, e está adaptado ao processamento de informação relativa a pedidos de acção conjunta ("gathering"). Vários tipos de controlo podem ser considerados: controlo do comportamento do agente, controlo da comunicação do agente e controlo dos recursos do agente (O.Baujard et al. 1992, O.Baujard et al. 1993).

A.6 UPShell

A UPShell[4] (*"University of Porto Shell"*) é uma ferramenta para a geração de SM orientada para o desenvolvimento de aplicações industriais. Pode

[4] http://garfield.fe.up.pt:8001/ eol/UPShell.html

ser usada para gerar sistemas inteligentes específicos, mas o seu principal objectivo é a transformação de um conjunto de sistemas inteligentes numa comunidade de agentes cooperativos. A ferramenta guia o utilizador através de um sistema de menus, permitindo-lhe instalar os agentes e as máquinas nas quais serão executados.

No momento do lançamento dos agentes, a UPShell constrói no topo de cada SBC um segundo módulo, a camada de cooperação, que controla a actividade do SBC e a cooperação com os restantes elementos da comunidade. O SBC, conjuntamente com o módulo de cooperação, constitui um agente da comunidade. Todos os agentes da comunidade correm em simultâneo, envolvendo dois processos Unix por agente, um para o SBC e outro para o módulo de cooperação. Cada módulo de cooperação pode comunicar com outro da mesma espécie e com o SBC que lhe está associado, através de um mecanismo de passagem de mensagens implementado sobre "sockets" UDP em Unix.

A UPShell permite desenvolver SBC ou incorporar sistemas que já existissem anteriormente. Neste segundo caso, é necessário desenvolver os módulos de interacção com os agentes da UPShell. Quando a ferramenta lança os agentes, baseando-se nos modelos contidos nas bases de conhecimento dos SBC, cria para cada agente um modelo do conhecimento que ele deve possuir relativamente aos agentes com quem poderá cooperar. O módulo de cooperação de cada agente utiliza um quadro negro interno para o acesso à informação de que precisa a nível interno e como suporte às suas actividades externas de cooperação e de comunicação (E.Oliveira & F.Mouta 1993, E.Oliveira et al. 1993*a*) (F.Mouta 1996).

Apêndice B

Siglas

ABC	*Agentes Baseados em Conhecimento*
ATMS	*"Assumption Based Truth Maintenance Systems"*
CBA	*Computação Baseada em Agentes*
CGI	*"Common Gateway Interface"*
EC	*Engenharia do Conhecimento*
ES	*Engenharia de Software*
HTML	*"HiperText Markup Language"*
IA	*Inteligência Artificial*
IAD	*Inteligência Artificial Distribuída*
RPC	*"Remote Procedure Call"*
SBC	*Sistemas Baseados em Conhecimento*
SBQN	*Sistemas Baseados em Quadros Negros*
SM	*Sistemas Multiagente*
SMBC	*Sistemas Multiagente Baseados em Conhecimento*
SP	*Sistemas Periciais*
Web	*"World Wide Web"*

Apêndice C

Créditos

- *SICStus Prolog* é uma marca registada do *Swedish Institute of Computer Science*, Suécia.
- *Solbourne* e *OS/MP Unix* são marcas registadas da *Solbourne Computer, Inc.*
- *Sun* é uma marca registada da *Sun Microsystems, Inc.*
- *Unix* é uma marca registada da *AT&T*.
- *X Windows* é uma marca registada do *MIT*.
- *Netscape* é uma marca registada da *Netscape Communications Corporation.*