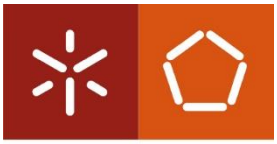


Universidade do Minho

Agentes e Sistemas Multiagente

Mestrado [Integrado] em Engenharia Informática
Agentes Inteligentes
Perfil: Sistemas Inteligentes

Autores
António Silva (a73827), a73837@alunos.uminho.pt
Marcelo Pinto (pg35396), mqspinto@gmail.pt

Vila Real, Outubro de 2017

Resumo

Os Sistemas de Partilha de Bicicletas (SPB) permitem aos utilizadores alugar bicicletas para realizar viagens curtas. O funcionamento de um SPB sem intervenção intencional resulta em desequilíbrios na gestão de ocupação da estação: enquanto algumas estações sofrem de falta de bicicletas, o que impede o aluguer nessas estações, outros sofrem de congestionamento, o que impede a devolução de bicicletas. Para garantir a alta satisfação dos utilizadores e um aumento de receitas ao operador, é necessária uma abordagem efetiva de reequilíbrio para manter um estado de sistema equilibrado.

Os agentes e Sistemas Multiagente constituem uma área que estuda, constrói e aplica sistemas em que diversas entidades computacionais (Agentes ou Sistemas Multiagente) interagem e perseguem um conjunto de objetivos e/ou realizando um conjunto de tarefas [7]. Assim, neste trabalho pretende-se a resolução do problema por meio de um sistema multiagente.

Índice

1. Introdução

Agentes e sistemas multiagente é um novo paradigma para desenvolver aplicações. Em 1992, a computação baseada em agentes foi aclamada como "o próximo avanço significativo no desenvolvimento de software" [1], e em 1994, como “a nova revolução do software” [2]. Nos anos seguintes até hoje, o conceito de agente tornou-se importante tanto na inteligência artificial (AT) como na informática convencional.

Este tipo de sistemas tem sido largamente usado, como podemos constatar na secção 3.7 deste trabalho. Quanto à investigação em Sistemas multiagente está focada no desenvolvimento de princípios e modelos computacionais para construir, descrever, implementar e analisar as formas de interação e coordenação de agentes em sociedades de reduzida ou elevada dimensão, sendo dado grande enfase na construção de standards, princípios e modelos que permitam a criação de pequenas e grandes sociedades de agentes semiautónomos, capazes de interagir convenientemente de forma a atingirem os seus objetivos.

Os Sistemas de Partilha de Bicicletas (SPB) permitem aos utilizadores alugar bicicletas para realizar viagens curtas. Os utilizadores alugam e devolvem as bicicletas em estações de bicicleta dedicadas, que normalmente encontram-se distanciadas a uma centena de metros entre cada uma das estações. Cada estação possui uma capacidade fixa, que determina o número de bicicletas que podem ser armazenadas. Apesar deste sistema ganhar recentemente grande popularidade como uma alternativa de transporte ecológico nas grandes cidades, sofre de um problema comum: o Problema do Reequilíbrio de Partilha de Bicicletas (PRPB). O funcionamento de um SPB sem intervenção intencional resulta em desequilíbrios na gestão de ocupação da estação: enquanto algumas estações sofrem de falta de bicicletas, o que impede o aluguer nessas estações, outros sofrem de congestionamento, o que impede a devolução de bicicletas. Para garantir a alta satisfação dos utilizadores e um aumento de receitas ao operador, é necessária uma
abordagem efetiva de reequilíbrio para manter um estado de sistema equilibrado.

Neste trabalho pretende-se a resolução deste problema por meio de um sistema multiagente.

2. Objetivos da Etapa do Trabalho Prático

- Estado da arte sobre os agentes e sua aplicação a domínios concretos, abordando as diferentes propriedades e vertentes;
- Conceber e modelar uma arquitetura distribuída baseada em agentes para o dado problema.

3. Estado da Arte

Os agentes e Sistemas Multiagente constituem uma área que estuda, constrói e aplica sistemas em que diversas entidades computacionais (Agentes ou Sistemas Multiagente) interagem e perseguem um conjunto de objetivos e/ou realizando um conjunto de tarefas [7].

3.1. O que é um agente inteligente?

Um agente inteligente é um subsistema, situado num ambiente, que é capaz de agir autonomamente nesse ambiente de forma a atingir os seus objetivos de *design*. A autonomia, neste caso, pode referir-se à capacidade de o agente agir sem intervenção humana ou de outros agentes e controlar as suas ações e estado interno [3]. Além da autonomia, um agente deve apresentar flexibilidade, isto é, exibição de capacidades como reação, iniciativa, aprendizagem e socialização [6].

Um agente processa informação interna, e interage com outros agentes por meio de sensores e de atuadores (figura 1). O comportamento do agente é dado abstratamente pela função do agente $[f: \mathcal{P}^* \rightarrow \mathcal{A}]$ onde $\mathcal{P}^*$ é uma sequência de perceções e $\mathcal{A}$ é uma ação. O agente age numa arquitetura física para produzir f e é constituído por uma arquitetura e um programa [4].

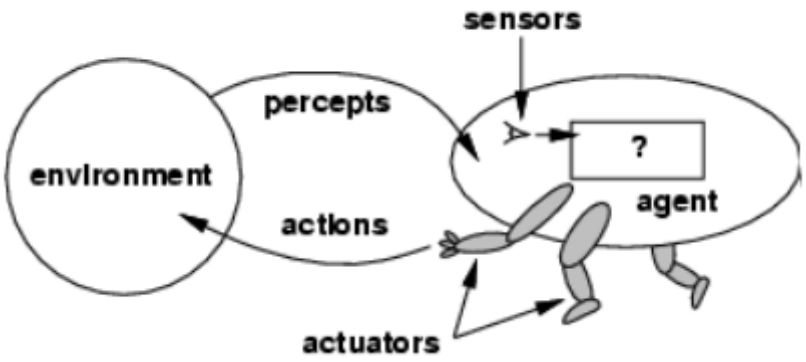


Figura 1: Interação dos sensores e atuadores no ambiente.

3.2. Sistemas Multiagente

Os Sistemas Multiagente são sistemas compostos por múltiplos agentes, que exibem um comportamento autónomo ao mesmo tempo interagem com os outros agentes presentes no sistema. Estes agentes exibem duas características fundamentais:

- são capazes de agir de forma autónoma tomando decisões levando à satisfação dos seus objetivos;
- são capazes de interagir com outros agentes utilizando protocolos de interação social inspirados nos humanos e incluindo funcionalidades como: coordenação, cooperação, competição e negociação.

Os Sistemas Multi-Agente incluem diversos agentes que interagem ou trabalham em conjunto, podendo compreender agentes homogéneos ou heterogéneos. Cada agente é basicamente um elemento capaz de resolução autónoma de problemas e opera assincronamente, com respeito aos outros agentes.

3.3. Propriedades de Agentes

Seguindo a noção fraca de agente, um agente deve conter no mínimo características como [6]:

- Autonomia: capacidade de o agente agir sem intervenção humana ou de outros agentes e controlar as suas ações e estado interno;
- Reatividade: capacidade de percecionar os eventos que ocorrem no seu universo de discurso e resposta adequada e atempadamente a mudanças ocorridas nesse ambiente;
- Proatividade: capacidade de tomar iniciativa, conduzindo as suas próprias ações mediante m comportamento dirigido por objetivos;
- Sociabilidade: capacidade de relacionamento com outros agentes, comunicando, competindo ou cooperando na resolução de problemas que lhes sejam colocados.

Um agente é considerado forte quando as entidades com que se depara são eminentemente cognitivas, passiveis de desenvolver a sua própria consciência, de se apresentar como tendo um conjunto de mais valias como a percetibilidade, a sentimentalidade e a emoção [6].

Outras características interessantes num agente são a intencionalidade, aprendizagem, competência, veracidade, racionalidade, benevolência (adoção de objetivos de terceiros, desde que não prejudiquem os seus próprios objetivos), emotividade e mobilidade (capacidade de movimentação através da rede formada pelos seus pares, executando as suas tarefas).

3.4. Arquiteturas

As arquiteturas de sistemas multiagentes diferenciam-se pela forma como a tomada de decisão ocorre internamente até à realização de uma ação. As arquiteturas de agentes são o conjunto das especificações teóricas e a obtenção de resultados práticos, na medida em que oferecem a implementação de sistemas segundo essas especificações. Uma arquitetura permite a visualização do mapa interno dos agentes: estruturação dos seus dados, operações e controle de fluxo.

Uma arquitetura deve apresentar as propriedades do agente e especifica como o agente pode ser decomposto na construção de um conjunto de módulos, e como esses módulos podem interagir.

As arquiteturas podem ser divididas em três áreas:

- A arquitetura deliberativa;
- A arquitetura reativa;
- Arquiteturas híbridas.

3.4.1. Arquitetura Deliberativa

Este tipo de arquitetura possui uma representação simbólica do mundo, sendo que suas deliberações (também chamadas decisões) são feitas por meio de um processo baseado em raciocínio lógico. Este raciocínio trabalha sobre um conjunto de símbolos que, sendo fisicamente concebíveis, podem ser combinadas formando-se estruturas sobre o qual se pode operar.

Algumas caraterísticas deste tipo de arquitetura são:

- Possui um modelo simbólico do ambiente;
- As decisões são tomadas via raciocínio lógico;
- Possui um conjunto de metas e intenções;
- Elaboram planos de ações para alcançar um objetivo comum;
  Ex: Arquitetura BDI [9].

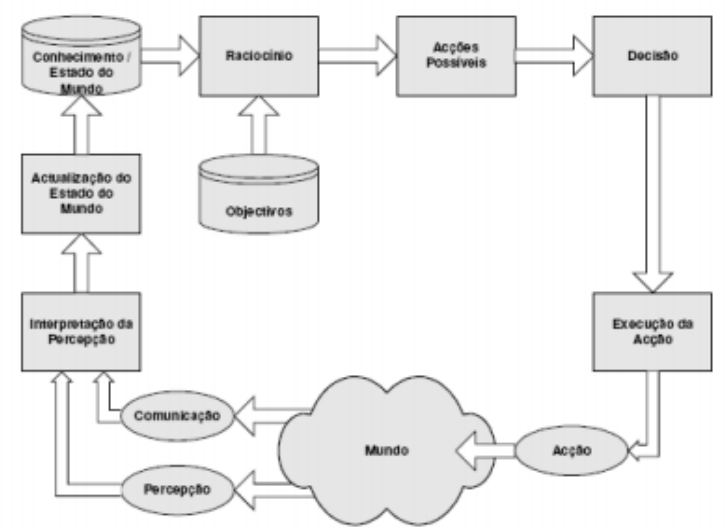


Figura 2: Arquitetura Deliberativa.

3.4.2. Arquitetura Reativa

Este tipo de arquitetura define que, em uma arquitetura reativa, o processo de tomada de decisão de um agente ocorre em tempo real, em resposta a estímulos do ambiente, captados pelos seus sensores, ou a mensagens enviadas por outro agente. Neste tipo de agente, o mecanismo de controlo é, geralmente, implementado por um conjunto de regras evento-ação.

Algumas caraterísticas deste tipo de arquitetura são:

- O agente desta arquitetura não possui representações simbólicas do seu ambiente;
- Não usam mecanismos de raciocínio simbólico;
- As decisões tomadas são implementadas em alguma forma de mapeamento direto da situação para a ação, usando regras de condição/ação (estímuloresposta);
- O processo de tomada de decisão de um agente ocorre em tempo real, em resposta a estímulos do ambiente, captados por seus sensores

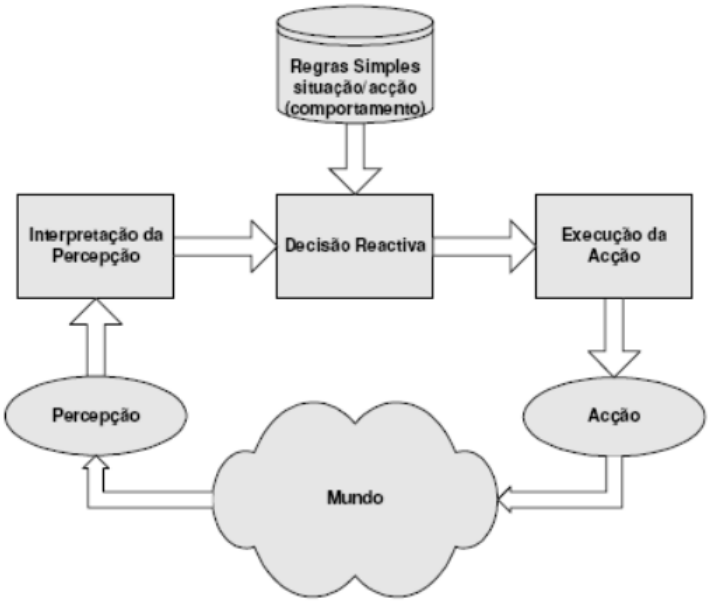


Figura 3: Arquitetura Reativa.

3.4.2. Arquitetura Híbridas

As arquiteturas híbridas são provenientes das deficiências encontradas nas arquiteturas deliberativas e reativas, reunindo propriedades de ambas. As arquiteturas reativas têm dificuldades para modificar seus planos de ação a partir do momento em que a situação passa a divergir de seus objetivos iniciais. Este tipo de arquiteturas pode ter dificuldades em lidar com situações imprevistas que exigem decisões rápidas. Estas devem definir agentes dotados de capacidades reativas, de raciocínio e planejamento, resolvendo as limitações provenientes das abordagens mais "puras".

Algumas caraterísticas deste tipo de arquitetura são:

- Mistura componentes das arquiteturas deliberativa e reativa;
- As decisões são tomadas via várias camadas de software;
- Nasceu devido às lacunas não preenchidas pelas arquiteturas deliberativa e reativa, segundo argumentavam muitos pesquisadores.

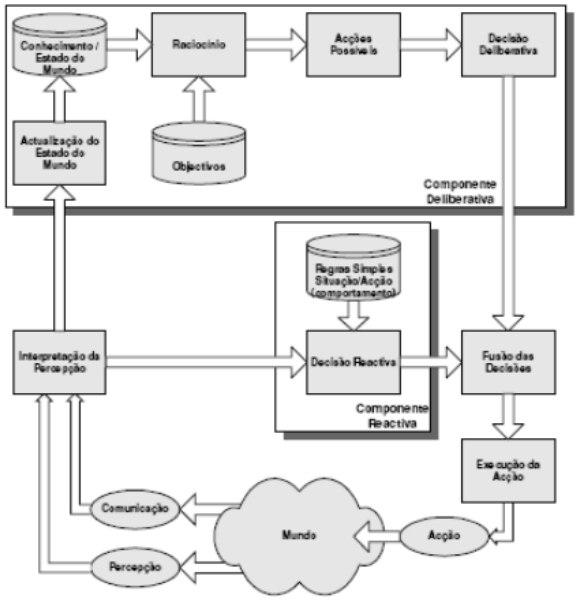


Figura 4: Arquitetura Híbrida.

3.6. Protocolos e Níveis de Comunicação

Os protocolos de comunicação são usualmente definidos a vários níveis. Os níveis inferiores definem o método de interligação dos agentes e os níveis intermédios definem o formato (sintaxe) da informação transmitida. Nos níveis superiores encontram-se as especificações do sentido (semântica) da informação. Genericamente podemos definir que um protocolo contém a seguinte estrutura de dados [8]:

- Emissor;
- Recetor(es);
- Linguagem utilizada;
- Funções de codificação e descodificação da linguagem;
- Ações que o recetor deve executar.

No início dos anos 90, foi fundado nos Estados Unidos da América, o Knowledge Sharing Effort (KSE) financiado pelo DARPA 14, com o objetivo de desenvolver protocolos para a troca e representação de informação entre sistemas de informação autónomos. O KSE gerou dois produtos finais principais (Finin et al., 1993):

- A Linguagem KQML (Knowledge and Query Manipulation Language). KQML: uma linguagem externa para comunicações entre agentes. Define um invólucro para formatar mensagens que determina o significado locutório da mensagem. O KQML não está preocupado com o conteúdo da

mensagem mas sim com a caracterização da informação necessária à compreensão desse conteúdo;

- O Formato KIF (Knowledge Interchange Format). A KIF é uma linguagem que se destina explicitamente a representar o conhecimento sobre um domínio de discurso específico. Foi desenvolvido primariamente como forma de definir o conteúdo de mensagens expressas em KQML. Para além do KQML e KIF, existem várias linguagens definidas no âmbito da comunicação em Sistemas Multiagente. De entre as mais utilizadas destaca-se o FIPA ACL (Agent Communication Language).

Em 1995, a FIPA – Foundation for Intelligent Physical Agents iniciou o desenvolvimento de standards para Sistemas Multiagente. A parte fulcral desta iniciativa situava-se ao nível do desenvolvimento de uma linguagem de comunicação para agentes (FIPA, 1999). A ACL resultante é semelhante ao KQML, sendo primariamente uma linguagem de comunicação externa e não obriga à utilização de qualquer linguagem específica para o conteúdo.

Para além da preocupação com a definição de uma linguagem de comunicação, os agentes que interagirem num sistema terão que falar a mesma linguagem e atribuir significados idênticos aos conceitos em discussão, só assim serão capazes de entender e serem entendidos pelos outros agentes. Torna-se assim necessário a existência de uma ontologia que especifique o significado dos objetos e conceitos em discussão. No entanto, as especificações das linguagens e das plataformas de comunicação são totalmente independentes.

3.7. Aplicações

As aplicações de sistemas multiagente são múltiplas, e por isso destacamos algumas áreas tais como:

- gestão de tráfego e serviços de transporte;
- controle de redes de comunicação e de computadores;
- sistemas de potência;
- gerenciamento de transações de bancos, comércio;
- monitoração, supervisão e controlo de processos industriais;
- sistemas de manufaturação;
- atendimento e serviços médicos;

Relativamente a aplicações singulares, cita-se:
Artigo: Multi-agent distributed data mining approach for classifying meteorology data: case study on Iran's synoptic weather stations

Data: Maio de 2017
Autores: A. Niazalizadeh Moghadam, R. Ravanmehr.
Descrição: Nova abordagem de mineração de dados distribuídos chamada mineração de dados hierárquicos multiagente para classificar dados de meteorologia, que foi coletada de diferentes sites amplamente distribuídos em todo o país (Irão). O método utiliza uma versão modificada do algoritmo REPTree, que foi otimizado para funcionar em sistema multiagente.

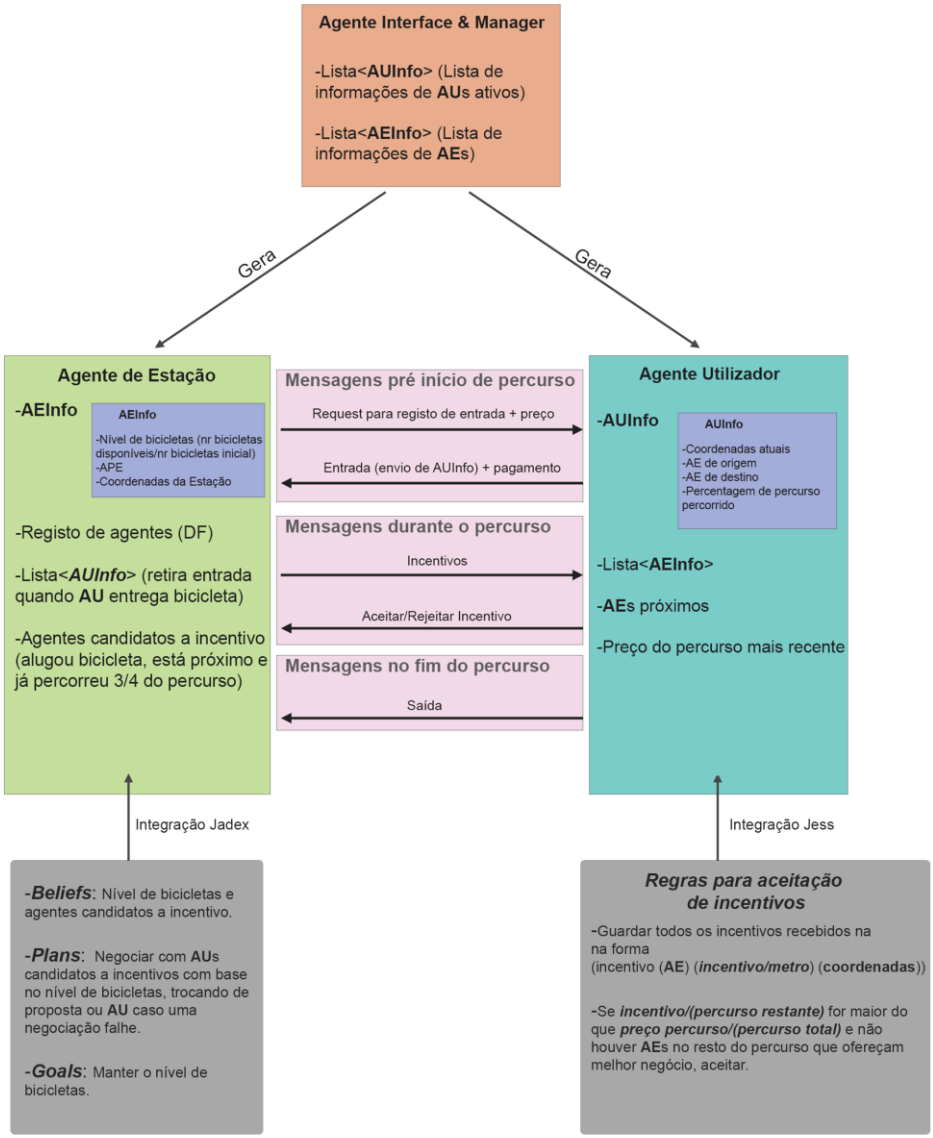
Artigo: An investigation of timed transfer coordination using event-based multi agent simulation
Data: Agosto de 2017
Autores: Le Minh Kieu, Ashish Bhashar, Mario Cools, Edward Chung
Descrição: Transferências mal coordenadas aumentam significativamente o tempo de espera dos passageiros, especialmente no caso de conexões perdidas. O artigo propõe uma abordagem de simulação para investigar as possibilidades de diferentes estratégias de transferência cronometrada tanto no planeamento de cronograma como no controlo operacional. Em particular, propõe-se um modelo de simulação multiagente baseada em eventos (EMAS), que captura as interações entre os veículos de trânsito, os passageiros e o ambiente considerando os veículos de trânsito e os passageiros como classes separadas de agentes que interagem em um sistema dinâmico. O modelo é validado utilizando os dados da Reserva Automática do Veículo Observada e da Reserva Automática de Tarifas de duas rotas com transferências no Sudeste das Queensland, Austrália. O EMAS é então utilizado para avaliar diferentes estratégias de transferência temporizadas para planejamento de programação e controle operacional.

Artigo: The adaptive distributed observer approach to the cooperative output regulation of linear multi-agent systems
Data: Janeiro de 2017
Autores: He Cai, Frank L. Lewis, Guiquang Hu, Jie Huang
Descrição: O problema de regulamentação de produção cooperativa de sistemas multiagentes lineares usando abordagem de observação distribuída foi resolvido sob o pressuposto de que cada seguidor conhece a matriz do sistema do sistema líder. Para remover esta suposição, neste trabalho, foi proposto um esquema de controlo distribuído utilizando o observador adaptativo distribuído, que consiste em três etapas. Primeiro, um observador distribuidor adaptativo é projetado para estimar tanto a matriz do sistema quanto o estado do sistema líder. Em segundo lugar, um algoritmo adaptativo é desenvolvido para calcular as soluções das equações do regulador online. Em terceiro lugar, o feedback de estado adaptativo

e os controladores de retorno de saída de medição adaptativa são sintetizados para resolver o problema da regulação de saída cooperativa sem a hipótese de que cada seguidor conheça a matriz do sistema do líder.

Artigo: A Multiagent Approach to Database Migration for Big Data Systems
Data: Julho de 2017
Autores: Mihai Horia Zaharia
Descrição: É apresentada uma possível solução para acelerar a integração de vários dados no grande mainstream de dados. O enriquecimento de dados e a convergência de todas as fontes possíveis ainda estão no início. Como resultado, as técnicas existentes devem ser reescritas para aumentar a integração de bancos de dados já existentes ou dos específicos da Internet de Coisas, a fim de usar as vantagens dos grandes dados para cumprir o objetivo final da criação de dados na Web. Neste artigo, soluções semânticas específicas da web são usadas para projetar um sistema baseado em agentes inteligentes. Ele tenta resolver alguns problemas específicos da automação do sistema de migração de banco de dados com o objetivo final de criar uma ontologia comum em vários repositórios de dados ou produtores para integrá-los em sistemas baseados em grandes arquiteturas de dados.

4. Arquitetura



Na arquitectura que propomos, o Agente Interface cria e inicia os restantes agentes (**AE**s e **AU**s), mantendo um objecto com informações para cada um dos agentes (**AEInfo** e **AEInfo**) numa lista, de modo a que seja possível mais tarde visualizar o estado do sistema.

O Agente de Estação é o típico agente BDI, pelo que possui o *Goal* de manter o número de bicicletas o mais próximo possível do valor inicial, chegando ao mesmo executando *Plans* dependendo da informação que possui sobre o nível de bicicletas e o exterior (agentes candidatos), ou seja, os seus *Beliefs*.

O Agente Utilizador, por outro lado, usa regras para decidir se deve aceitar ou não incentivos provindos de **AE**s. Estas regras estão presentes num ficheiro *.clp* partilhado por todos os **AU**s, sendo que cada um adiciona ou actualiza registos de incentivos oferecidos pelos **AE** aquando de ter recebido uma proposta.

Desta forma, os **AU**s têm sempre acesso ao melhor incentivo conhecido para o percurso que estão a efectuar, pelo que não se corre o risco de os mesmos aceitarem sempre a primeira proposta viável que recebem. Cabe, portanto, aos **AE**s aumentarem os incentivos caso estejam a sofrer de falta de bicicletas, de modo a atraírem mais utilizadores.

4.1. Alterações feitas à arquitetura

Apesar de não terem sido efetuadas alterações significativas à forma como os agentes estão estruturados e interagem uns com os outros, devido a restrições de tempo e ao número de elementos do grupo ser reduzido, escolhemos criar o sistema apenas em Jade.

5. Implementação

5.1. *Container*

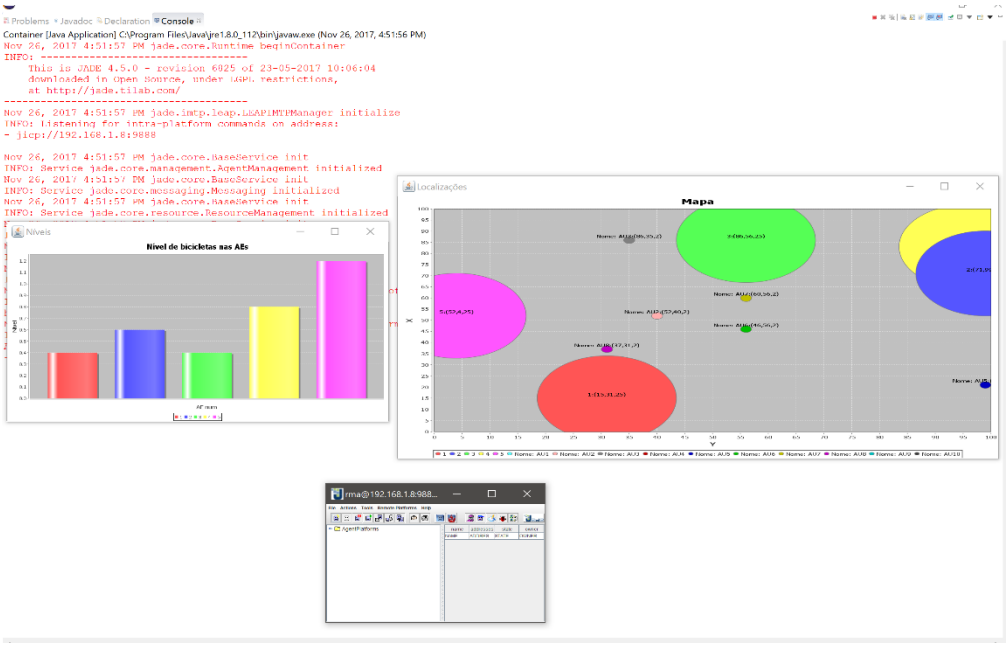
Possui a *main* do projeto e trata de iniciar todos os agentes. Cria objetos de informações (*AUInfo* e *AEInfo*) e guarda-os em listas, de forma a que o *Agente Interface* possua os dados necessários para mostrar o estado do sistema.

Gera também as coordenadas dos *Agentes Estação,* garantido que ficam a pelo menos 5 unidades de distância uns dos outros.

Os *Agentes Utilizador* são criados sequencialmente segundo um intervalo gerado aleatoriamente de entre 2 e 8 segundos.

5.2.Agente Interface

Cria duas *charts* de *JFreeChart*, uma medindo o nível de bicicletas (número de bicicletas atual dividido pelo número de bicicletas inicial) através de barras e outra apresentando o mapa do sistema, ou seja, a posição dos agentes:



As *charts* são atualizadas a cada segundo.

5.3.Agente Estação

```
@Override
public void action() {
    double incentivo;
    if (mapIncentivos.containsKey(au)){
        if (mapIncentivos.get(au)+stride<=aei.getThreshold()){

            incentivo=mapIncentivos.get(au)+stride;
            mapIncentivos.put(au,incentivo);

            ACLMessage entry=new ACLMessage(ACLMessage.INFORM);
            entry.addReceiver(au);
            entry.setContent(String.valueOf(incentivo));
            myAgent.send(entry);
        }
    }
}
```

Possui um *Behaviour* principal (*MsgHandler*) que recebe e trata mensagens de *Agentes Utilizador* que, dependendo do nível de bicicletas no sistema, poderá iniciar um outro *Behaviour* (*Negotiate*). Este último, como o nome indica, irá negociar com os *Agentes Utilizador* que enviaram uma mensagem de proximidade na *APE*. A negociação começa com uma oferta inicial baixa e vai aumentando até um limite (*Threshold*)

```
public double getThreshold(){
    if (nivel==0.0){
        threshold=10000.0; // aumentar indefinidamente
    }else if (nivel>0.0&&nivel<1.0/2.0){
        threshold=precoMedio;
    }else if (nivel>=1.0/2.0 && nivel<1.0){
        threshold=precoMedio/3.0;
    }
    return threshold;
}
```

que é determinado por o quão baixo é o nível de bicicletas:

- Se o nível de bicicletas for zero, a situação é crítica para a estação, visto que não pode fornecer bicicletas aos utilizadores. Quando isto acontece, a estação irá fazer tudo o que pode para cativar utilizadores candidatos a entregarem a sua bicicleta na estação. Os utilizadores em espera são adicionados a uma *queue* e serão atendidos quando a estação receber a próxima bicicleta.
- Se o nível de bicicletas estiver entre zero e metade, a estação vai aumentar o incentivo até à média de preços das viagens que partiram da estação.
- Se o nível de bicicletas se encontrar entre metade e 1, a situação não representa riscos imediatos de escassez de bicicletas, logo aumenta apenas até um terço do preço médio das viagens efetuadas.

5.4.Agente Utilizador:

Possui uma fase inicial onde gera aleatoriamente as suas próprias coordenadas e determina qual o *Agente Estação* mais próximo das mesmas, que será a origem. Gera também coordenadas para o destino (garantindo que não seja a própria origem).

A posição do agente passa a ser a da estação origem e é enviado um pedido à mesma no formato "*D-número da AE destino"*. O *Agente Estação*, caso disponha de bicicletas, irá responder com o preço da viagem, que será guardado para mais tarde decidir se deve aceitar incentivos.

Quando esta fase inicial estiver terminada, são iniciados três *subBehaviours* dentro de um *ParallelBehaviour*:

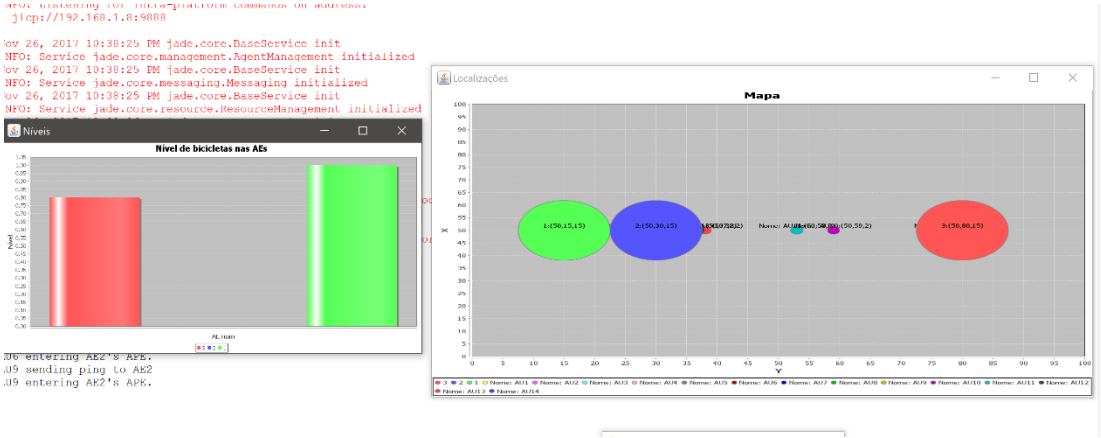
- *PercorrerPercurso,* que atualiza as coordenadas do agente até chegar à estação destino.
- *AEPing* que, caso o agente tenha percorrido 3/4 do percurso, testa se se encontra dentro de uma *APE* e, se for o caso, envia um "*ping*" à mesma. Se já tiver enviado este "*ping",* mas se encontrar fora da área, envia outra mensagem, desta vez informando que já saiu.

- *ReceberIncentivos* que recebe os incentivos das estações e testa se deve aceitar ou não. Tal como foi estabelecido na arquitetura do sistema, o agente irá aceitar um incentivo caso (*incentivo/percurso restante)* seja maior do que *(preço do percurso/percurso total)* e não houver estações no resto do percurso que ofereçam incentivos maiores. Todos os incentivos são guardados e partilhados entre utilizadores.

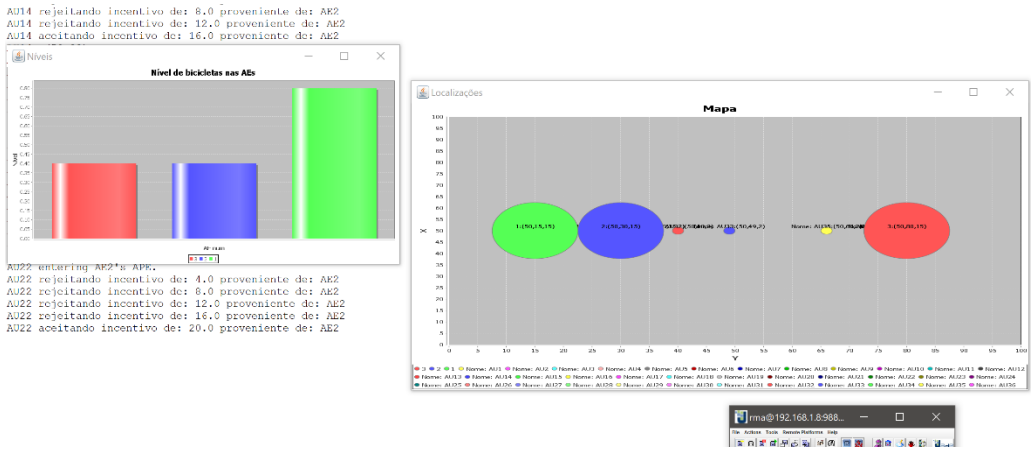
Quando o agente chega à estação destino, o *onEnd()* envia a mensagem de entrega e a estação incrementa o seu número de bicicleta.

6. Resultados

Conduzindo um teste controlado com 3 *AEs* em linha, com clientes a surgirem aleatoriamente de todas as *AE*s, mas com apenas destino às dos extremos (verde e vermelha), sem negociação, a a *AE* do meio (azul) acaba naturalmente por ficar sem bicicletas:



Conduzindo o mesmo teste com negociação, verificamos que a *AE* azul consegue atrair clientes, evitando a escassez de bicicletas:



No geral, verifica-se também uma melhoria na distribuição de bicicletas em testes não controlados ainda que, devido à aleatoriedade do posicionamento das *AE*s e *AU*s, algumas situações de escassez são inevitáveis como, por exemplo, quando existem *AE*s completamente isoladas cuja posição não é intersetada pelos trajetos dos *AU*s a não ser que esta seja o próprio destino deles.

7. Conclusão

O desenvolvimento deste trabalho envolveu o estado da arte, a conceção e modelação da arquitetura de agentes e a implementação dos mesmos. Os conceitos de agentes e sistemas multiagente aprendidos nas aulas e recordados durante a conceção do estado da arte foram de grande utilidade para o desenvolvimento da arquitetura. Assim, achamos que atingimos os objetivos propostos.

8. Bibliografia

[1] Sargent, P.: Back to school for a brand new ABC. In: The Guardian, 12 March, p. 28, 1992.
[2] Ovum R.: Intelligent agents: the new revolution in software, 1994.
[3] Jennings R., Wooldridge M.: Applications of Intelligent Agents. University of London.
[4] Russel, Norvig.: Agentes Inteligentes, Universidade Federal do ABC.
[5] Adorni G. and Poggi A.: An object-oriented language for distributed artificial intelligence, International Journal of Man-Machine Studies 38 435–453, 1993.
[6] Novais P, Analide C.: PDF's Teóricos da Unidade Curricular de Agentes Inteligentes.
[7] Weiss, G. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, 1999.
[8] Huhns M., Stephens L.: Multiagent Systems and Societies of Agents, 1999.
[9] Georgeff M.: BDI Agents: From Theory to Practice, 1995.