Universidade do Porto

FEUP Faculdade de Engenharia

Mestrado Integrado em Engenharia Informática e Computação
Agentes e Inteligência Artificial Distribuída
4º Ano

Determinação de percursos usando agentes BDI

Relatório Intercalar

Autores:
Bruno Filipe Neves Ferreira - 080509088 - ei08088@fe.up.pt
Carlos Tiago da Rocha Babo - 080509118 - ei08118@fe.up.pt
Hélder Alexandre dos Santos Moreira - 080509170 - ei08170@fe.up.pt

9 de Dezembro de 2011

Conteúdo

Capítulo 1

Enunciado

1.1 Descrição do cenário

Numa viagem de automóvel o condutor tem que tomar várias decisões até chegar ao seu destino. Tendo em mente que o agente condutor pode ter várias intenções sobre o percurso a realizar, a escolha do caminho tem como base essas intenções.

Independentemente da intenção do condutor (chegar rapidamente a um local ou realizar um percurso turístico), o agente não conhece antecipadamente as condições do percurso escolhido. Apenas toma conhecimento dessas condições quando estas se encontram no seu campo de visão, ou então quando o rádio informa da ocorrência de um acidente.

1.2 Objetivos do trabalho

No âmbito da Unidade Curricular de Agentes e Inteligência Artificial Distribuída, o objetivo deste trabalho é criar um sistema multi-agente que consiga recolher informações sobre o espaço e indicar ao condutor qual o percurso que este deve tomar, tendo em conta os vários estímulos possíveis: acidentes, pontos de interesse, estado do tempo, rádio, entre outros. O sistema deve avaliar os estímulos (condições do percurso), bem como as condições pré-definidas de modo a que o condutor chegue ao destino do modo pretendido.

1.3 Resultados esperados e forma de avaliação

No final do trabalho o agente responderá a várias simulações de forma correta e eficiente de acordo com as diferentes configurações definidas, por

exemplo, diferentes mapas de estradas.

A avaliação será feita modificando parâmetros e analisando o comportamento do agente, por exemplo, ao definirmos que queremos um percurso direto entre o ponto de partida e o ponto de chegada, esperamos ver o agente a deslocar-se diretamente para o destino.

Com o objetivo de ter uma avaliação detalhada, iremos abranger todas as combinações. Partindo do exemplo anterior, podemos ter o mesmo percurso mas desta vez adicionando um acidente, obrigando assim a que seja calculado um desvio. Adicionalmente a forma de transmitir essa informação ao agente condutor também pode ser modificada, ou seja, pode ser transmitida por rádio ou então só quando estiver no campo de visão do agente é que é calculado o desvio.

Não esquecendo que essas mesmas combinações também serão feitas para percursos com pontos de interesse escolhidos, e variando o local onde existirá um acidente.

Relativamente aos acidentes, importa ainda referir que estes podem pré-definidos ou criados dinamicamente. Qualquer alteração ao estado do tempo pode também levar a visitar ou não um determinado ponto.

Capítulo 2

Plataforma/Ferramenta

2.1 Para que serve

O *Jadex* BDI 2.0 é um motor que permite especificar agentes inteligentes baseado no modelo *Belief-Desire-Intention* com recurso a *XML* e Java. Uma das principais vantagens da ferramenta é não introduzir nenhuma linguagem de programação nova, permitindo usar *IDEs* focados no desenvolvimento orientado a objetos, como o Eclipse.

2.2 Características Principais

Os agentes racionais têm uma representação explícita no ambiente e dos objetivos que estão a tentar atingir. Estes são racionais, pois escolhem sempre a ação mais promissora (de acordo com a base de conhecimento do mundo) para atingir os seus objetivos. Dado que normalmente não têm conhecimento dos efeitos de cada ação previamente, os agentes deliberam sempre sobre as opções disponíveis.

Em 1987, Bratman introduziu uma arquitetura para descrever os agentes racionais. Esta consiste em explorar as atitudes mentais que geram as ações do homem - crenças, desejos e intenções. Assim, as crenças captam as atitudes de informação, os desejos as atitudes de motivação, e as intenções as atitudes de deliberação dos agentes. Mais tarde, Rao e Geroff adotaram este modelo e transformaram-no numa teoria mais formal e num modelo de execução para software de agentes, baseado na noção de crenças, objetivos e planos (*beliefs*, *goals* e *plans*).

Deste modo, o *Jadex* reaproveitou o modelo de Rao e Geroff e portou-o para o contexto da programação orientada a objetos. Assim, os *beliefs*, *goals* e *plans* representam objetos de classes que são criadas e manipuladas no contexto de cada agente. Os agentes comportam *beliefs*, representados

por qualquer tipo de objeto Java e que pertencem à sua *beliefbase*. Os *goals* representam motivações concretas (estados a atingir) e que influenciam o comportamento do agente. Para atingir os seus objetivos (*goals*), o agente executa planos (*plans*), que são descritos em Java.

Este modelo comporta, desta forma, duas componentes essenciais. Por um lado, o agente reage a mensagens, eventos internos e *goals* selecionando e executando *plans*. Por outro lado, o agente delibera continuamente sobre os seus *goals* atuais, adaptando-se às condições de cada momento.

2.3 Funcionalidades Relevantes para o Trabalho

2.3.1 *Agents* (agentes)

Para especificar um agente é necessário criar dois tipos de ficheiros: um ficheiro *XML* com as definições do agente e um ficheiro JAVA para cada plano especificado no *XML*.

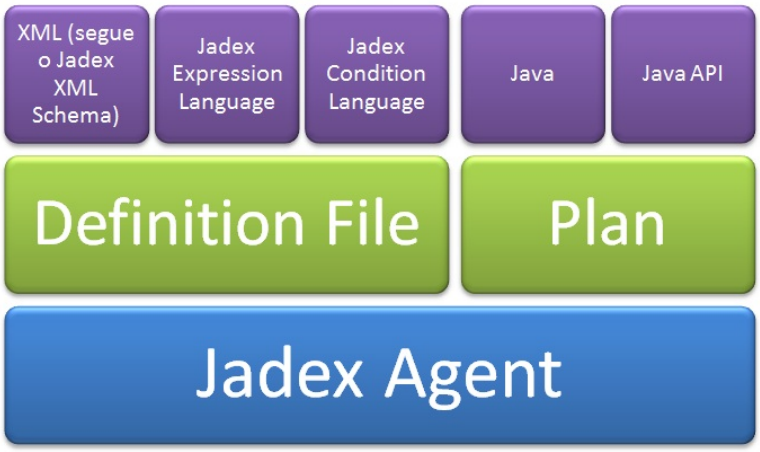

Figura 2.1: Esquema representativo da plataforma Jadex.

- *Beliefs*: representam o conhecimento do agente sobre o ambiente e sobre si mesmo. São representados por qualquer tipo de objeto Java. Podem ser referenciados em expressões, acedidos e modificados pelos planos, usando a interface da *beliefbase*, ou até mesmo herdados de uma *capabilitie* utilizando o sufixo *ref*.

  Exemplo em que um agente tem como *belief* a sua posição no espaço

(que herda de uma *capabilite*) e informação sobre o seu modo de operação (turístico ou direto):

```
<agent>
  ...
 <beliefs>
  <beliefref name="myself">
   <concrete ref="move.myself"/>
  </beliefref>
  ...
  <belief name="turistica" class="boolean">
   <fact>true</fact>
  </belief>
 </beliefs>
  ...
</agent>
```

- *Goals*: representam as motivações do agente e que despoletam as ações. Podem ser de quatro tipos, mas apenas dois se adequam no contexto do trabalho:

  - *Perform goal*: algo que precisa de ser feito, mas sem ter necessariamente um objetivo.

    Exemplo: criar um *goal* para deslocar um agente até ao destino final. De acordo com a sua posição, e os pontos que já visitou, o *plan* respetivo terá de analisar e encontrar os diferentes objetivos intermédios até ao destino final. Este *goal* é descartado quando surge um acidente no caminho (*drop condition*):

    ```
    <performgoal name ="go_destiny" retry="false" >
     <dropcondition language="jcl">
       ISpaceObject $target &amp;&amp;
       $beliefbase.acidente == $target &amp;&amp;
       $target.state=="notavoid"
     </dropcondition>
    </performgoal>
    ```

  - *Achieve goal*: representa um estado desejado, mas não especifica como lá chegar.

    Exemplo em que surge um acidente e pretende-se que o condutor se desloque até um estado em que evitou o acidente. Este *goal* é criado automaticamente quando surge um acidente no campo de visão do condutor:

```
<achievegoal name="avoid_accident" retry="false">
 <parameter name="accident" class="ISpaceObject">
  <value>
   $accident
  </value>
 </parameter>
 <creationcondition language="jcl">
  ISpaceObject $target &amp;amp;&amp;amp;
  $beliefbase.acidente == $target &amp;amp;&amp;amp;
  $target.state=="notavoid"
 </creationcondition>
 ...
</achievegoal>
```

- *Plans*: descrevem como é que as ações do agente se desenrolam. Os *plans* são selecionados de acordo com a ocorrência de *events* e *goals*. A seleção de *plans* é feito de forma automática pelo sistema e representa um dos aspetos principais da infraestrutura BDI. No *Jadex* os *plans* dividem-se em duas partes: um *head* declarado em XML e um *body* declarado em Java.

  Exemplo de um *plan* que é acionado pelo *goal go_destiny*:

  Head:

```
<plan name="go_destiny">
 ...
 <body class="GoDestiny" />
  <trigger>
   <goal ref="go_destiny"/>
  </trigger>
</plan>
```

  Body:

```
public class GoDestiny extends Plan
{
 public void body()
 {
  ...
 }
}
```

2.3.2 *Capabilities* (capacidades)

No *Jadex*, uma *capability* representa um módulo de agente encapsulado composto por *beliefs*, *goals* e *plans*. Este conceito permite criar um pacote

com todas as características de um agente e que pode ser reutilizado em qualquer outro agente. Normalmente, as *capabilities* não têm representação explícita no ambiente, representando uma extensão de um ou vários agentes. Para aceder às *beliefs* ou *goals* de uma *capability*, o agente tem de incluir uma referência na sua declaração, não sendo obrigatório herdar todas as características desta, introduzindo a noção de hierarquia.

Definição da capability:

```
<capability xmlns="http://jadex.sourceforge.net/jadex-bdi"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://jadex.sourceforge.net/jadex-bdi
 http://jadex.sourceforge.net/jadex-bdi-2.0.xsd"
 name="MyCapability" package="mypackage">

 <beliefs> ... </beliefs>
 <goals> ... </goals>
 <plans> ... </plans>
 ...
</capability>
```

No agente que herda a *capability*:

```
<agent ...>
 <capabilities>
  <capability name="MyCapability"
   file="mypackage/MyCapability.capability.xml"/>
  ...
 </capabilities>
 ...
</agent
```

2.3.3 *Events* (eventos)

Uma característica importante do agente é a capacidade de reagir a eventos internos, ou mensagens externas. Os eventos internos podem ser usados para denotar uma ocorrência dentro de um agente, enquanto que as mensagens externas representam a comunicação entre dois agente. Os *events* podem despoletar *plans*.

2.3.4 *Expressions* (expressões)

As *expressions* permitem especificar *beliefs* e valores dos parâmetros numa sintaxe que se assemelha a consultas em *SQL*. Esta funcionalidade facilita a seleção de objetos.

Exemplo de um caso em que se pretende obter um ponto de interesse chamado "Praia":

Ficheiro *XML* do agente:

```
<agent ...>
  ...
 <expressions>
  <expression name="find_poi" class="POI">
   select one POI $poi
   from $poi in $beliefbase.pois
   where $poi.getName().equals($name)
  </expression>
  ...
 </expressions>
  ...
</agent>
```

Classe que implementa o *plan*:

```
public void body
{
 IExpression query = getExpression("find_poi");
 ...
 POI poi = (POI)query.execute("$name", "Praia");
 ...
}
```

2.3.5 *Application* (aplicação)

A *application* é criada com recurso a um ficheiro *XML* e é nela onde estão declarados todos os objetos do ambiente, as suas características, posições no espaço, entre outras propriedades. É também nela onde se pode declarar os vários cenários de simulação, criando para isso diferentes conjuntos de configurações. Cada cenário pode conter objetos em diferentes posições, assim como propriedades distintas. Por exemplo, é possível ter uma configuração em que o objetivo do condutor é deslocar-se simplesmente para o seu destino e outro onde é desejado que passe nos vários pontos de interesse do mapa. Esta possibilidade surgiu na versão 2.0 do *Jadex*, não sendo necessário recorrer a ferramentas de terceiros (*Swing*) para desenhar a interface.

Capítulo 3

Especificação

3.1 Identificação e caraterização dos agentes

A aplicação contempla um agente, Driver, e duas *capabilities*, GPS e Rádio.

3.1.1 GPS

O GPS contém como *beliefs* o ambiente em que o condutor se encontra, tendo assim a informação sobre vários componentes do espaço, isto é, pontos de interesse, ponto inicial e final da viagem, o mapa de estradas (caminhos possíveis), entre outros. O GPS tem como *goals* mover o condutor, Driver, até um destino a determinar quando recebe um pedido do mesmo e também recalcular a nova rota quando toma conhecimento de um acidente no percurso estabelecido.

O GPS procura assim nas pontos de interesse nas imediações do agente e encaminha-o para lá. Em caso de acidente, é recalculada a rota para um próximo ponto de interesse e o que iria ser visitado fica guardado para ser visitado novamente. Quando todos os pontos estiverem visitados, ou não houver caminho possível para nenhum outro, o GPS encaminha o agente para o ponto final da viagem.

3.1.2 Driver

O agente Driver inclui a *capability* GPS para interagir com a mesma e possui como *beliefs* o próximo ponto turístico para onde se está a dirigir e eventualmente algum acidente que esteja a evitar. O condutor não tem desta forma acesso aos restantes componentes do espaço, indicando apenas que tipo de viagem pretende realizar (turística, rápida, etc) e deixando as

decisões a cabo do GPS, tal como acontece num ambiente real. O agente tem assim como *goals* analisar o seu espaço circundante (radar) e avisar o GPS caso encontre alguma acidente no seu percurso.

3.1.3 Rádio

A informação relativa aos acidentes é guardada ao nível do Rádio, não sendo assim conhecida previamente nem pelo condutor, nem pelo GPS, e será transmitida ao GPS através da troca de eventos por mensagem com o agente quando o rádio se encontrar ativo ou detetados pelo campo de visão do condutor.

3.1.4 Aplicação

A aplicação recorre às funcionalidades disponibilizadas pelo *Jadex* 2.0 e tem como elemento principal um espaço 2D da classe *ContinuousSpace2D* que aloja vários elementos:

- **Mapa de estradas**
  Representa os caminhos onde o condutor pode movimentar-se e será representado pelo meio de arestas. Esta representação tem como objetivo permitir o mapeamento do mapa num grafo, facilitando a determinação de percursos.

- **Pontos de interesse**
  São os pontos de interesse que o condutor deve visitar caso faça uma visita turística. Podem conter propriedades específicas, tais como o facto de só permitir visitas num determinado estado atmosférico ou também uma pré-condição para a visita (a visita de outro ponto de interesse por exemplo).

- **Acidentes**
  Representam um obstáculo incontornável na estrada e que obrigará o condutor a procurar outro caminho. Tal como os pontos de interesse, podem conter propriedades específicas de acordo com o estado atmosférico (só se encontra ativo em tempo de chuva por exemplo).

- **Ponto inicial da viagem**
  Representa o ponto inicial da viagem a partir do qual o condutor partirá.

- **Ponto final da viagem**
  Representa o ponto final da viagem para o qual o condutor tem como objetivo dirigir-se.

Figura 3.1: Esquema representativo da estrutura da aplicação.

3.2 Protocolos de interação

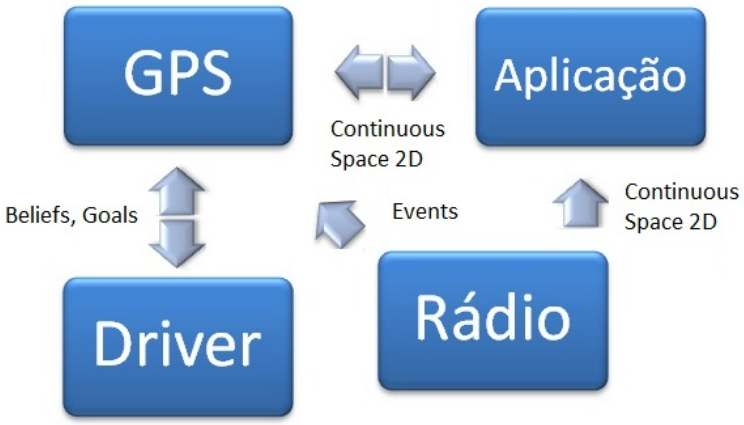

Figura 3.2: Esquema representativo das interações entre os vários componentes.

Hierarquicamente, o GPS encontra-se numa camada superior ao agente Driver e é esta que, com conhecimento do ambiente e com um pedido do agente, o move até à localização desejada. De acordo com a pesquisa que fizemos sobre a tecnologia *Jadex* e sobre as possíveis abordagens ao problema esta pareceu-nos a mais correta e viável.

Assim o processo de interação entre os vários componentes começa pelo agente que comunicará ao GPS a sua intenção de realizar uma viagem e visitar os pontos turísticos conhecidos. O GPS calcula o caminho que o agente deve seguir e move-o de acordo. Ao mover-se o condutor comunicará ao GPS

caso encontre algum acidente no seu caminho e este recalculará a nova rota.

Caso o modo rádio esteja ativado, este comunica através de eventos por mensagem com o GPS, indicando-lhe os acidentes que estão presentes ao longo do mapa e este terá a responsabilidade de avaliar se os acidentes encontrados interferem no percurso definido para o condutor e caso isso aconteça, recalcular a rota.

3.3 Faseamento do projeto

É possível identificar 3 fases distintas no desenvolvimento do projeto. A primeira, fase onde é desenvolvido o esqueleto da aplicação e implementadas as funcionalidades base que servirão para suportar todo o trabalho, a segunda, onde é verificada a viabilidade de novas funcionalidades, indo assim para um âmbito para além do que é especificado no enunciado e por fim a terceira, fase onde será dado foco à interface gráfica para permitir uma melhor demonstração e uso da aplicação.

Neste momento encontramo-nos a finalizar a primeira fase, tendo já implementado a estrutura da aplicação e a interação entre os vários componentes. Os objetivos futuros passam por terminar esta fase, implementando o mapa de estradas e de seguida abordar novas funcionalidades, como o estado atmosférico e as pré-condições de visita, antes de ser abordada a terceira fase.

Capítulo 4

Recursos

4.1 Bibliografia

- http://jadex-agents.informatik.uni-hamburg.de
- http://paginas.fe.up.pt/ eol/AIAD/aiad1112.html
- http://sourceforge.net/projects/jadex/

4.2 Software

- Eclipse - IDE utilizado para desenvolvimento.
- Jadex 2.0