Plano de Aula

- Introdução
- Conceitos Básicos
- KQML
- FIPA ACL
- Exemplo – Agente Comprando Livros

1

Motivação : Agentes

- Reflexão
  - Cenário dinâmico
  - Integração de diversas mídias e dispositivos
- Como implementar?
  - Sugestão: usar *Agentes Inteligentes*
- Agentes
  - Ideal para aplicações *autônomas*, *móveis* e inseridas em uma *sociedade*
  - Naturalmente capazes de:
    - **Reagir** de forma autônoma e reativa
    - **Raciocinar** de forma pró-ativa
    - **Adaptar-se** ao ambiente onde estão inseridos
    - **Interagir** com outras entidades

2

Recordar é Viver...

- IAD lembra...
  - Comportamento Social
  - Organização
  - Interação
    - Coordenação
    - Cooperação
    - Ação

3

Recordar é Viver... II

- RDP
  - Agentes projetados para o problema.
  - Benevolência
  - Organização em Tempo de Projeto
  - Controle Global
  - Pouca Flexibilidade
- SMA
  - Agentes preexistentes
  - Raciocínio sobre ações, cooperação
  - Maior Flexibilidade
  - Organização Variável
  - Interações Genéricas

4

Assim...

- SMA...
  - Realiza a decomposição das tarefas
  - Tem conflitos...ou junta esforços
  - É variável...

Como isso acontece??

Interação é a chave...

5

Motivação : Agentes

- Interação
  - Solução compartilhada de problemas
  - Requisito básico
    - Mecanismo de Comunicação
      - Linguagem comum
      - Infra-estrutura
- Mas como implementar?

6

Conceitos Básicos

7

Comunicação...

Relação dinâmica mediada por sinais, que quando interpretados, afetam os agentes envolvidos.

Possibilita... Cooperação e Coordenação

- Em outra palavras... Comunicação é:
  - Uma forma de ação
  - Utilizada pelo agente para tornar realidade um estado do mundo

8

Tipos de comunicação

- Telepatia
  - Acesso direto à uma *Knowledge Base (KB)*
    - Lendo a mente...
    - Cada um com sua KB
    - KB compartilhada
  - Problemas
    - Podem usar o mesmo símbolo para denotar coisas diferentes
    - Podem usar símbolos diferentes para denotar a mesma coisa
- Linguagem de Comunicação de Agentes (ACL)
  - Comum

9

Atos de Fala

- Categorização das expressões humanas qto a
  - **Intenção** do **locutor**
  - **Impacto** do ato comunicativo no **receptor**
- Existem aproximadamente 4600 atos de fala!
  - Não é esse o objetivo de uma ACL
- Ajuda a decidir as primitivas das ACLs
  - Toda primitiva de uma ACL é um ato de fala

10

Para dar certo...

- Segundo Austin
  - É preciso saber o significado da performativa
  - O procedimento deve ser executado corretamente
  - Deve haver sinceridade.

11

Para dar certo... Searle

- Condições normais de Entrada/Saída
- Condições Preparatórias
  - Receptor capaz de realizar ação
  - Emissor acredita nisto
  - Receptor não deve fazer isto de qq maneira

12

Atos da Fala

- Linguagem Humana
  - Atos comunicativos são interpretados a partir da mensagem e do contexto
  - Nem sempre esta interpretação é óbvia
- Dificuldades
  - "Cale-se !" (Comando)
  - "Por favor, você pode se calar ?" (Pedido)
  - "Você vai se calar ou não ?" (Pergunta)
  - "Eu gostaria que você se calasse." (Informação)

13

Atos da Fala : Categorias

- Representativos
  - Expressam uma Proposta
  - Ex: ***"Vamos jogar boliche amanhã?"***
- Diretivos
  - Expressam um Pedido ou Comando
  - Ex: ***"Cale-se!"***
- Comissivos
  - Expressam Promessa ou Ameaça
  - Ex: ***"Eu prometo doar R$1.000,00."***
- Expressivos
  - Expressam Desculpas ou Agradecimentos
  - Ex: ***"Desculpe pelo atraso"***
- Declarativos
  - Mudam o Estado do Mundo
  - Ex: ***"Eu os declaro marido e mulher!"***
- Vereditos
  - Expressam um Julgamento
  - Ex: ***"West é criminoso."***

14

Várias Dimensões dos Atos da Fala

- Ato Locucionário
  - Emissão da mensagem
    - **"Você poderia fazer um café?"**
- Efeito Ilocucionário
  - O que o locutor entende por algo
    - **"Ele me pediu para fazer café"**
- Efeito Perlocucionário
  - O real efeito do ato comunicativo no receptor
    - **"Ele me fez fazer café"**
- Ideal
  - Efeito *perlocucionário previsível*
  - Para um dado Ato Locucionário,
    - Efeito Ilocucionário = Efeito Perlocucionário.
  - Ou seja, prever a reação do receptor!

15

Linguagens de Comunicação

- Baseadas na ***Teoria dos Atos da Fala*** *(Searle, Austin)*
- Objetiva a troca de informações entre agentes
  - Transporte de mensagens na rede através de protocolos de baixo nível
    - SMTP, TCP/IP, IIOP ou HTTP
  - Esse transporte é transparente para o usuário
- Para o KSE (ARPA Knowledge Sharing Effort), uma ACL é dividida em:
  - Ontologia
  - Linguagem de Conteúdo (lógica)
    - Codificação do conteúdo
  - Linguagem externa
    - Ato Locucionário (Performativa)
    - Encapsulamento de infomações para o roteamento

16

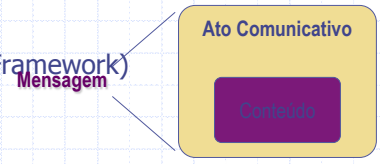
Ontologias

- Definição
  - Vocabulário cujos termos são relacionados a um certo domínio e precisamente definidos.
  - Os relacionamentos entre esses termos também são especificadas usando técnicas de modelagem formal
- Uma vez especificada...
  - Pode-se construir uma KB relacionada ao domínio coberto pela ontologia
  - Esta KB pode ser compartilhada com outros agentes interessados em conhecimentos sobre este domínio

17

Linguagem de Conteúdo (LC)

- *O que é?*
  - Linguagem em que deve ser codificado o conhecimento que se deseja compartilhar com o destinatário da mensagem
- Exemplos
  - KIF (*Knowledge Interchange Format*)
    - Versão prefixada ("*LISP like*") da lógica clássica de primeira ordem
    - Não está totalmente implementada para uso comercial
    - Produzida pelo KSE
  - Linguagem natural
  - Prolog, SQL, LISP
  - RDF (Resource Description Framework)
  - *XML*



18

Exemplo de uma Mensagem KIF

- (=(temperatura m1) (scalar 40 Celsius))
- (defrelation solteiro (?x) :=
  (and (man ?x)
  (not(casado ?x))))

19

Linguagem Externa (LE)

- O que é ?
  - Linguagem que encapsula:
    - O ato comunicativo = ato locucionário = performativa
    - Informações para o roteamento
      - Agente receptor
      - Ontologia
      - Linguagem de Conteúdo
      - Conteúdo
      - ...
- Exemplos
  - KQML
  - FIPA ACL

20

KQML

Knowledge Query and Manipulation Language

21

KQML

- Objetivo
  - Interoperabilidade entre agentes de software em aplicações distribuídas e/ou heterogêneas
- Pioneira (*KSE* – 1994)
- Aplicações
  - eCommerce (negociação)
  - Jogos
  - BDs distribuídos e/ou heterogêneos
  - Integração de tecnologias
  - ...

22

KQML : Características

- Principais características da linguagem:
  - Independente de *Linguagem de Conteúdo*
  - Independente de *Ontologia*
  - Independente de *Protocolos de Transporte de Rede*
  - Comunicação baseada nos *atos da fala*
    - Performativas
  - Sugere o uso de Agentes Facilitadores
  - Camadas:
    - *Comunicação, Conteúdo e Mensagem*

23

KQML : Sintaxe

```
( performativa
      :sender        <word>
      :receiver      <word>
      :reply-with    <word>
      :in-reply-to   <word>
      :language      <word>
      :ontology      <word>
      :content       <expression>
 )
```

Camada de Comunicação

Camada de Mensagem

Camada de Conteúdo

24

KQML : Performativas

- Performativas Típicas
  - ***tell* :** informa que o conteúdo da mensagem está na KB do *locutor*
  - ***ask-if* :** o *locutor* quer saber se o conteúdo de sua mensagem é verdadeiro para o *receptor*
  - ***advertise*** : o locutor quer que o receptor saiba que ele pode processar mensagens no modelo do seu conteúdo
  - ***insert* :** o locutor "pede" ao receptor que adicione o conteúdo da mensagem à sua KB
- Outras Performativas
  - achieve, ask-about, ask-all, ask-one, break, broadcast, broker-all, broker-one, deny, delete, delete-all, delete-one, discard, eos, error, evaluate, forward, generator, monitor, ...

25

Exemplo

- *A1* envia a seguinte mensagem para *A2* ...

```
( advertise
      :sender       A1
      :receiver     A2
      :reply-with   id1
      :language     KQML
      :ontology     kqml-ontology
      :content      ( ask-if
                          :sender          A2
                          :receiver        A1
                          :in-reply-to  id1
                          :language        Prolog
                          :ontology        CIn
                          :content     "Professor(X,Y)"))
```

26

Exemplo

- *A2* pergunta então a *A1* ...

```
( ask-if
     :sender          A2
     :receiver        A1
     :in-reply-to     id1
     :reply-with      id2
     :language        Prolog
     :ontology        CIn
     :content         "Professor(X, 'Agentes')"
)
```

27

Exemplo

- *A1* responde ao agente *A2* ...

```
( tell
     :sender          A1
     :receiver        A2
     :in-reply-to     id2
     :reply-with      id3
     :language        Prolog
     :ontology        CIn
     :content         "[Lesser, Sycara,
                      Jennings,...]"
)
```

28

KQML : Problemas

- Ambigüidade e Termos Vagos
  - O significado de performativas é pouco claro.
- Performativas com nomes inadequados
  - Algumas performativas têm nomes que *não* correspondem diretamente ao ato comunicativo a ela associado
  - Ex: "*tell*"
- Ausência de performativas
  - Alguns atos comunicativos não estão representados entre as performativas disponíveis - Comissivas

29

FIPA

Foundations for Intelligent Physical Agents

www.fipa.org

30

FIPA : Histórico

- FIPA é uma associação sem fins lucrativos fundada em 1996 e localizada em Genebra na Suíça
- Objetiva promover o sucesso das aplicações, serviços e equipamentos baseados em agentes
- Provê especificações para maximizar a interoperabilidade entre aplicações, serviços e equipamentos baseados em agentes

31

FIPA

- FIPA x KQML
  - Ao contrário do KQML, as especificações FIPA têm uma grande preocupação com a *modelagem semântica*
- Linguagem de Comunicação de Agentes
  - *FIPA ACL*
- Linguagens de Conteúdo
  - *FIPA Content Language Library (FIPA-CLL)*
  - *FIPA-SL, FIPA-RDF, FIPA-CCL, FIPA-KIF, ...*
  - *Mas também é independente de LC*

32

FIPA ACL

- Sintaxe
  - Praticamente igual ao KQML
  - Conjunto de performativas menor
    - *accept-proposal, agree, cancel, cfp, confirm, disconfirm, failure, inform, inform-if, inform-ref, not-understood, propose, query-if, query-ref, refuse, reject-proposal, request, request-when, request-whenever, subscribe*
- Semântica
  - Definida precisamente
  - *Mensagens são mapeadas na FIPA-SL*
  - *<i, inform(j, φ) pre: $B_i \varphi \wedge \neg B_i (Bif_j \varphi \vee Uif_j \varphi)$*
    - *Efeito: $B_j \varphi$*

33

Definindo Performativas...

- Inform – ato representativo, uma ação
  - Está sol lá fora.
  - Estou com sono.
- Visa..
  - Modificar as crenças de agentes
  - Não pode ser realizado por acidente

34

Request

- Ato Diretivo
- Tenta modificar as intenções do destinatário
- Três tipos diferentes
  - Request
  - Request-when
  - Request-whenever

35

FIPA ACL x KQML

- Semelhanças
  - Independência de Linguagem de Conteúdo e de ontologias
  - Sintaxe idêntica
- Diferenças
  - Visível principalmente na semântica
  - Performativas são diferentes
  - Os agentes de FIPA ACL são proibidos de manipular diretamente a KB de outros agentes

36

FIPA ACL x KQML

- Neutralizando Diferenças (situação fictícia)
  - Em KQML
    - Mensagem de A para B: *achieve goal X*
    - Usa todo o vocabulário da ACL (performativa "achieve")
  - Em FIPA ACL
    - Mensagem de A para B: inform *(achieve goal X)*
    - A *ACL* não entende! A *LC* sim!
    - Elimina a necessidade da performativa *achieve* na ACL

Conteúdo

37

Exemplo

- *A1* envia a seguinte mensagem para *A2* ...

```
( inform
      :sender       A1
      :receiver     A2
      :reply-with   id1
      :language     ...          :ontology    ...
      :content      ((= available-service A1)
                    (set(provide Professor(X,Y))))"
```

38

Exemplo

- *A2* pergunta então a *A1* ...

```
( query-ref
    :sender          A2
    :receiver        A1
    :in-reply-to     id1
    :reply-with      id2
    :language        Prolog
    :ontology        CIn
    :content         "Professor(X, 'Agentes')"
)
```

39

Exemplo

- *A1* responde ao agente *A2* ...

```
( inform
    :sender          A1
    :receiver        A2
    :in-reply-to     id2
    :reply-with      id3
    :language        Prolog
    :ontology        CIn
    :content         "X = Geber, X = Flávia,
                     X = Jacques..."
)
```

40

Exemplo

A simple one...

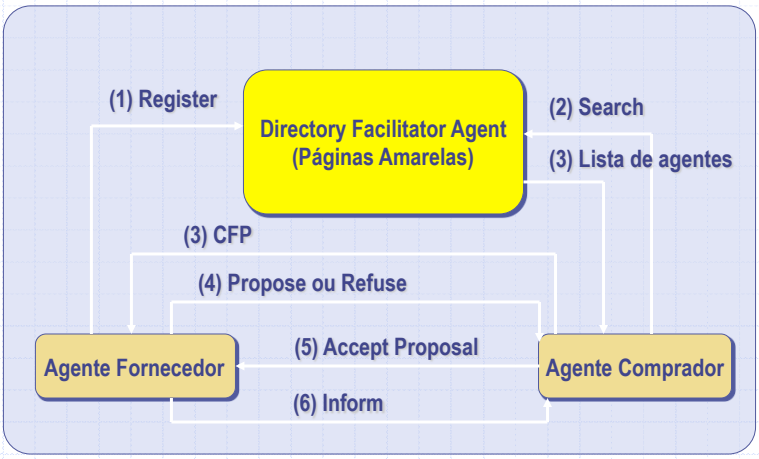
41

Compra de Livros

- Linguagens
  - Java
  - FIPA-ACL
- Ferramentas
  - JADE (Java Agent DEvelopment Framework)

42

Compra de Livros



43

Comportamento Agente Fornecedor

```
private class OfferRequestsServer extends CyclicBehaviour {
    public void action() {
            ACLMessage msg = myAgent.receive();
            if (msg != null) {  // Message received. Process it
               String title = msg.getContent();
               ACLMessage reply = msg.createReply();
               Integer price = (Integer) catalogue.get(title);
               if (price != null) {   // The requested book is available for sale. Reply with the price
                  reply.setPerformative(ACLMessage.PROPOSE);
                  reply.setContent(String.valueOf(price.intValue()));
               } else {   // The requested book is NOT available for sale.
                  reply.setPerformative(ACLMessage.REFUSE);
                  reply.setContent("not-available");
               }
               myAgent.send(reply);
            }
    }
}
```

44

Comportamento do Agente Comprador

```
private class RequestPerformer extends Behaviour {
    private AID bestSeller; // The agent who provides the best offer
    private int bestPrice; // The best offered price
    private int repliesCnt = 0; // The counter of replies from seller agents
    private MessageTemplate mt; // The template to receive replies
    private int step = 0;

    public void action() {
        switch (step) {
            (...)
        }
    }
```

45

Comportamento do Agente Comprador (cont.)

```
case 0:
        // Send the cfp to all sellers
        ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
        for (int i = 0; i < sellerAgents.length; ++i) {
            cfp.addReceiver(sellerAgents[i]);
        }
        cfp.setContent(targetBookTitle);
        cfp.setConversationId("book-trade");
        cfp.setReplyWith("cfp"+System.currentTimeMillis()); // Unique value
        myAgent.send(cfp);
        // Prepare the template to get proposals
        mt = MessageTemplate.and(MessageTemplate.MatchConversationId("book-
trade"),
        MessageTemplate.MatchInReplyTo(cfp.getReplyWith()));
        step = 1;
        break;
```

46

Comportamento do Agente Comprador (cont.)

```
case 1:
    // Receive all proposals/refusals from seller agents
    ACLMessage reply = myAgent.receive(mt);
    if (reply != null) { // Reply received
        if (reply.getPerformative() == ACLMessage.PROPOSE) {  // This is an offer
            int price = Integer.parseInt(reply.getContent());
            if (bestSeller == null || price < bestPrice) { // This is the best offer at present
                bestPrice = price;
                bestSeller = reply.getSender();
            }
        }
        repliesCnt++;
        if (repliesCnt >= sellerAgents.length) // We received all replies
            step = 2;
    }
    else block();
    break;
```


Comportamento do Agente Comprador (cont.)

```
case 2:
    // Send the purchase order to the seller that provided the best offer
    ACLMessage order = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
    order.addReceiver(bestSeller);
    order.setContent(targetBookTitle);
    order.setConversationId("book-trade");
    order.setReplyWith("order"+System.currentTimeMillis());
    myAgent.send(order);
    // Prepare the template to get the purchase order reply
    mt = MessageTemplate.and(MessageTemplate.MatchConversationId("book-
    trade"),
    MessageTemplate.MatchInReplyTo(order.getReplyWith()));
    step = 3;
    break;
```

Comportamento do Agente Comprador (cont.)

```
case 3:
    // Receive the purchase order reply
    reply = myAgent.receive(mt);
    if (reply != null) {
        // Purchase order reply received
        if (reply.getPerformative() == ACLMessage.INFORM) {
            // Purchase successful. We can terminate
            System.out.println(targetBookTitle+" successfully purchased.");
            System.out.println("Price = "+bestPrice);
            myAgent.doDelete();
        }
        step = 4;
    }
    else block();
    break;
```