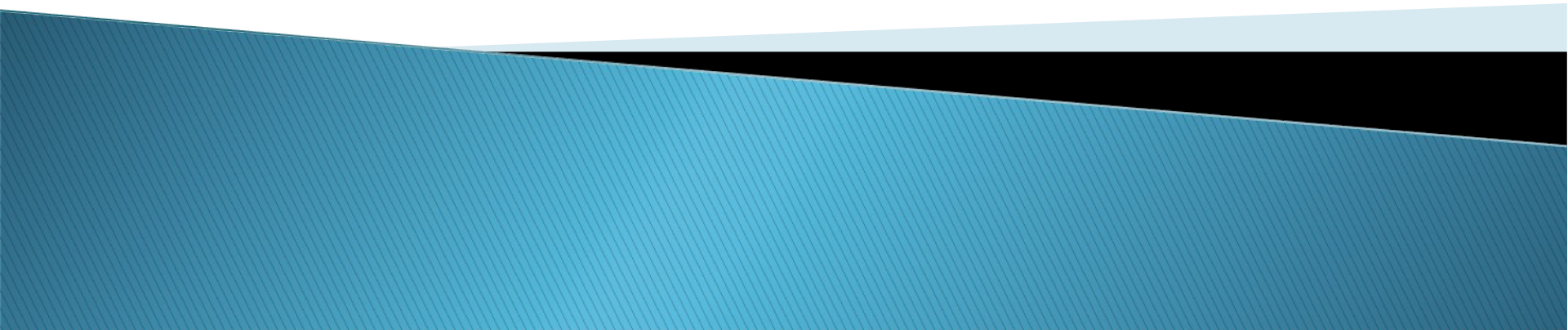
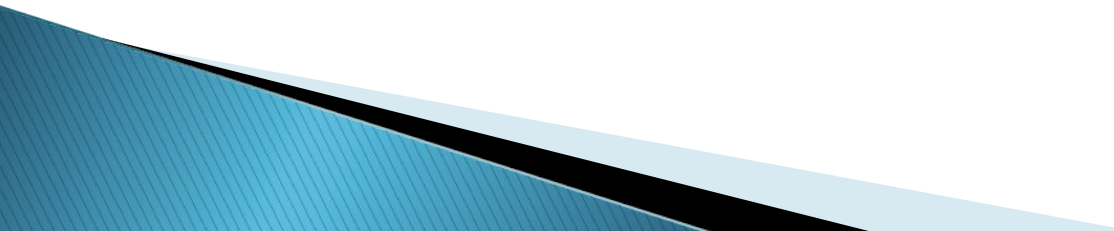


Agentes BDI (Belief Desire Intention)



Agenda

- ▶ Motivação
 - ▶ Objetivos
 - ▶ Introdução
 - ▶ Modelo e Arquitetura BDI
 - ▶ Implementação de Agentes BDI
 - ▶ Conclusão
- 

Motivação

- ▶ Modelo BDI (Belief Desire Intention)
 - Uma teoria filosófica do raciocínio prático, considerando-se as seguintes atitudes mentais: crenças, desejos e intenções.
- ▶ Plataformas para implementação de Agentes BDI

Objetivo

- ▶ Estudo para apresentar as principais plataformas de desenvolvimento de agentes BDI (Belief Desire Intention)

Introdução

- ▶ Modelo BDI (Bratman, 1987)
 - Teoria filosófica que explica o comportamento humano com três atitudes mentais:
 - Beliefs (crenças)
 - Desires (desejos)
 - Intentions (intenções)

Modelo BDI

▶ Modelo BDI

◦ Beliefs - crenças

- Características do ambiente
- Atualizadas após a percepção de cada ação
- Componente informativo do sistema
- Representam o conhecimento sobre o mundo

◦ Desires - desejos

- Informação sobre os objetivos a serem atingidos
- Representação do estado motivacional do sistema
- São relacionados eventualmente ao estado de mundos que o agente quer provocar

Modelo BDI

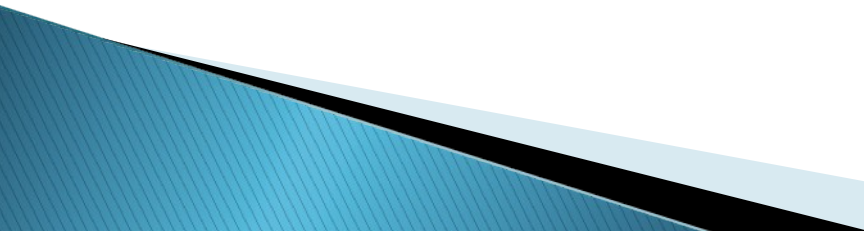
▶ Modelo BDI

- Intentions - intenções
 - Atual plano de ação escolhido
 - Componente deliberativo do sistema
 - Correspondem aos estados de mundo que o agente quer efetivamente provocar

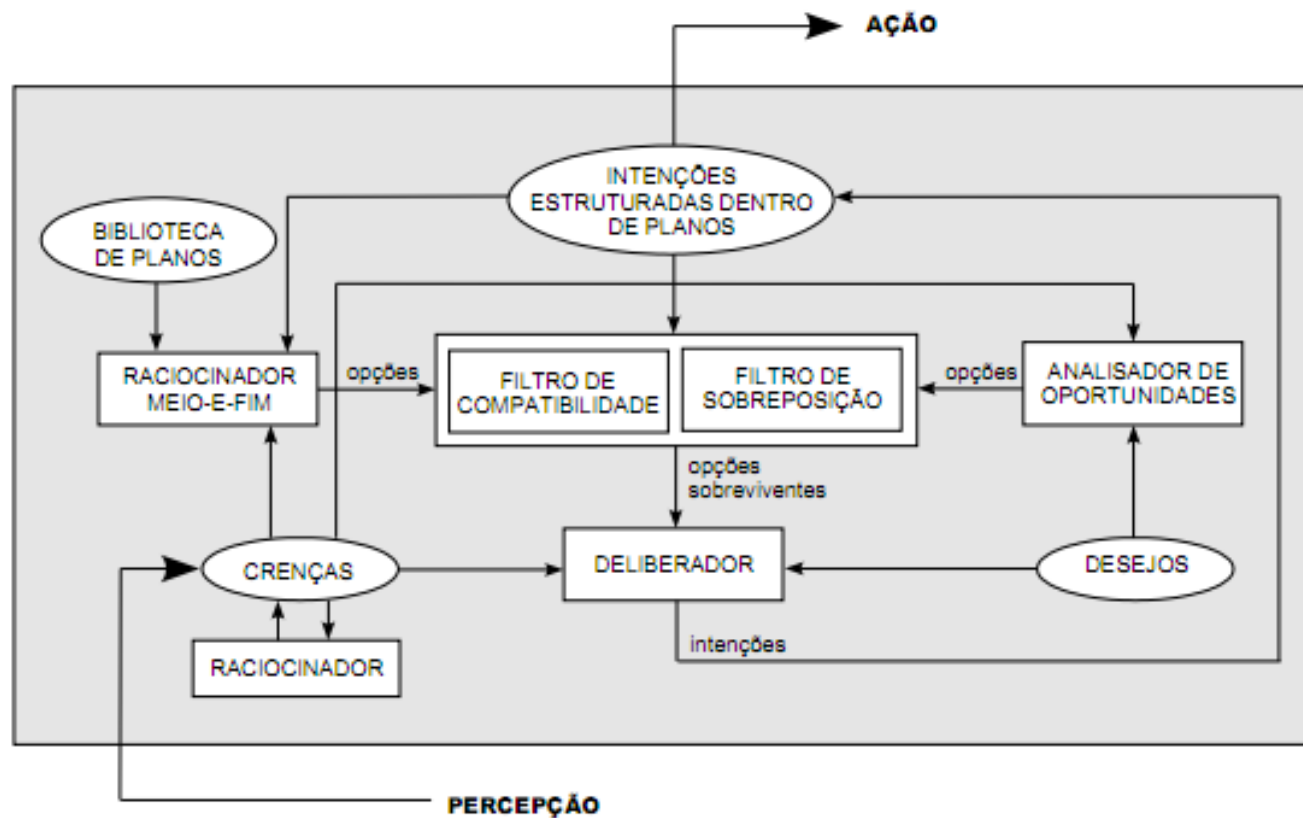
BDI

- ▶ As ideias básicas da abordagem BDI consistem em descrever o processamento interno do estado de um agente utilizando um conjunto de categorias mentais (crenças, desejos e intenções) e definir uma arquitetura de controle através da qual o agente seleciona racionalmente o curso de suas ações

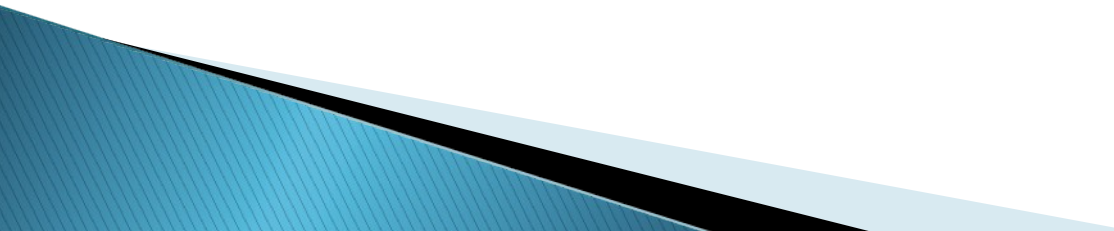
Arquitetura Intelligent Resource- bounded Machine Architecture (IRMA)

- ▶ Arquitetura que incorpora os aspectos inerentes ao modelo BDI
 - ▶ Proposta por Braman, Israel e Pollack
 - ▶ Objetivo: maior a descrição dos processos de um raciocínio prático em agentes com recursos limitados
- 

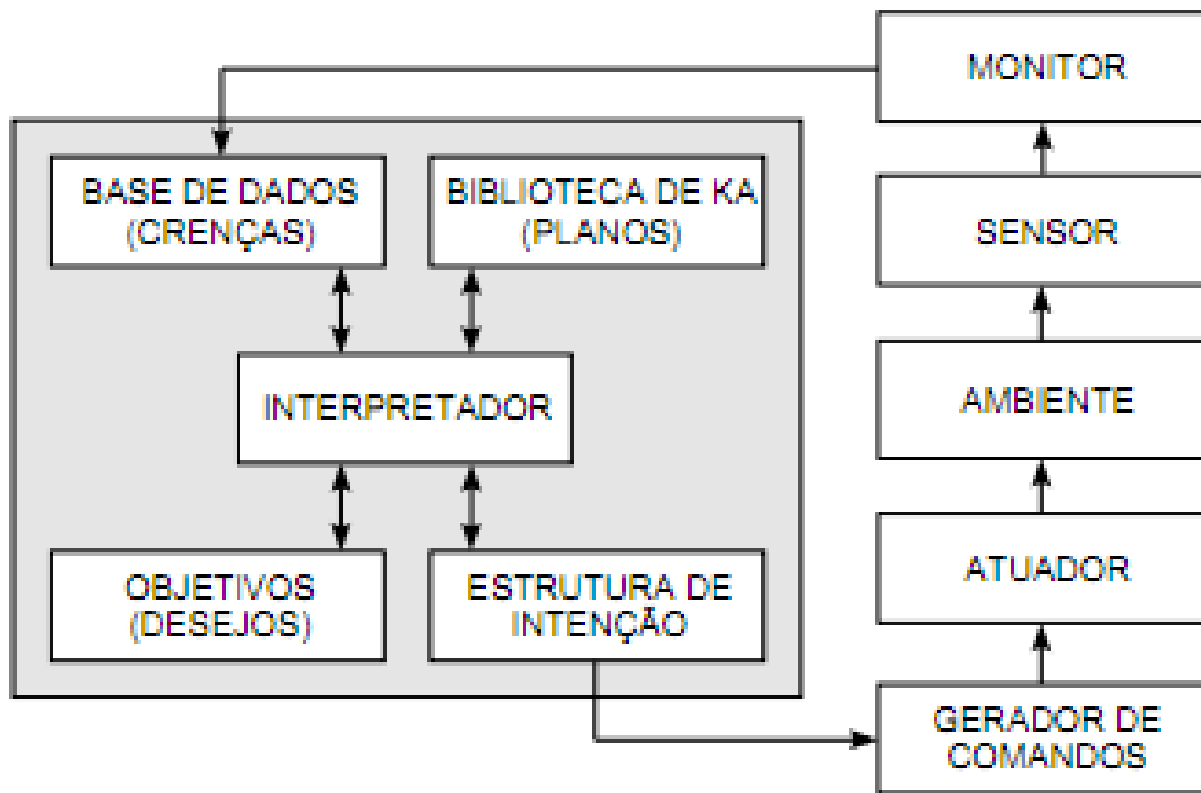
Arquitetura abstrata Intelligent Resource-bounded Machine Architecture (IRMA)



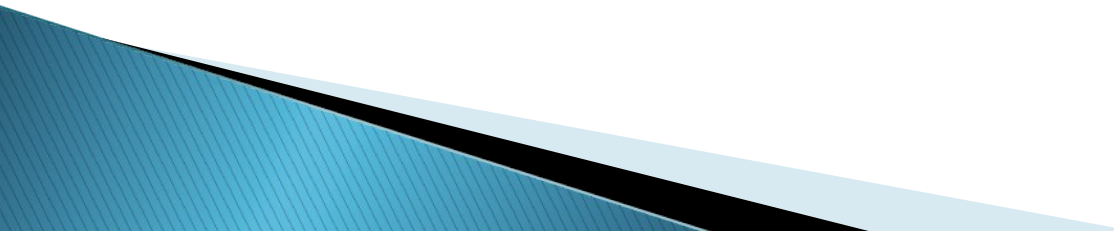
Arquitetura Procedural Reasoning System (PRS)

- ▶ Arquitetura híbrida que incorpora o modelo BDI
 - ▶ Proposta por Georgeff e Lansky
- 

Arquitetura Procedural Reasoning System (PRS)



Implementação de Agentes BDI

- ▶ Variedade de Linguagens e Plataformas para Implementação de Agentes BDI
 - ▶ Plataformas:
 - JACK™ Intelligent Agents
 - Jadex
 - JAM
 - Jason
 - CogniTA0
- 

JACK™ Intelligent Agents

- ▶ Desenvolvido pela AOS Agent Oriented Software Pty. Ltd.
 - Melbourne, Austrália
 - <http://www.agent-software.com/>
- ▶ Linguagem
 - JACK Agent Language

JACK

▶ Características

- Leve, requer poucos recursos de sistema
- Comunicação transparente entre agentes
- Ferramentas de Desenvolvimento
- Aplicações Comerciais
 - Suporte à Decisão
 - Gerenciamento de produção de óleo
 - Sistemas avançados de simulação militar
 - Sistemas financeiros
 - Assistentes pessoais inteligentes

JACK

- ▶ JACK Agent Language
 - Linguagem de programação orientada a agentes
 - Extensões à linguagem Java
 - Novas classes, interfaces e métodos
 - Extensões à sintaxe de Java
 - Compilador: JACK AL para Java
 - Extensões semânticas
 - Suporte ao modelo de execução requerido por um sistema orientado a agente

JACK Development Environment

The image shows a screenshot of the JACK Development Environment. On the left is a project tree with the following structure:

- ▶ Project Name: **NewProject**
 - ▶ Design Views
 - ▼ Agent Model
 - ▶ Capability Types
 - ▶ Plan Types
 - ▶ Event Types
 - ▶ Team Types
 - ▶ TeamPlan Types
 - ▶ Role Types
 - ▶ Named Roles
 - ▶ Named Data
 - ▶ Data Model
 - ▶ Other Files

The main area on the right is a splash screen with a grey background. It features the text:

JACK
Development Environment
by Agent Oriented Software

Below the text is a graphic of a jester character standing on a stack of puzzle pieces. The puzzle pieces are labeled with the letters J, A, C, and K. At the bottom of the splash screen, the URL <http://www.agent-software.com> is displayed.

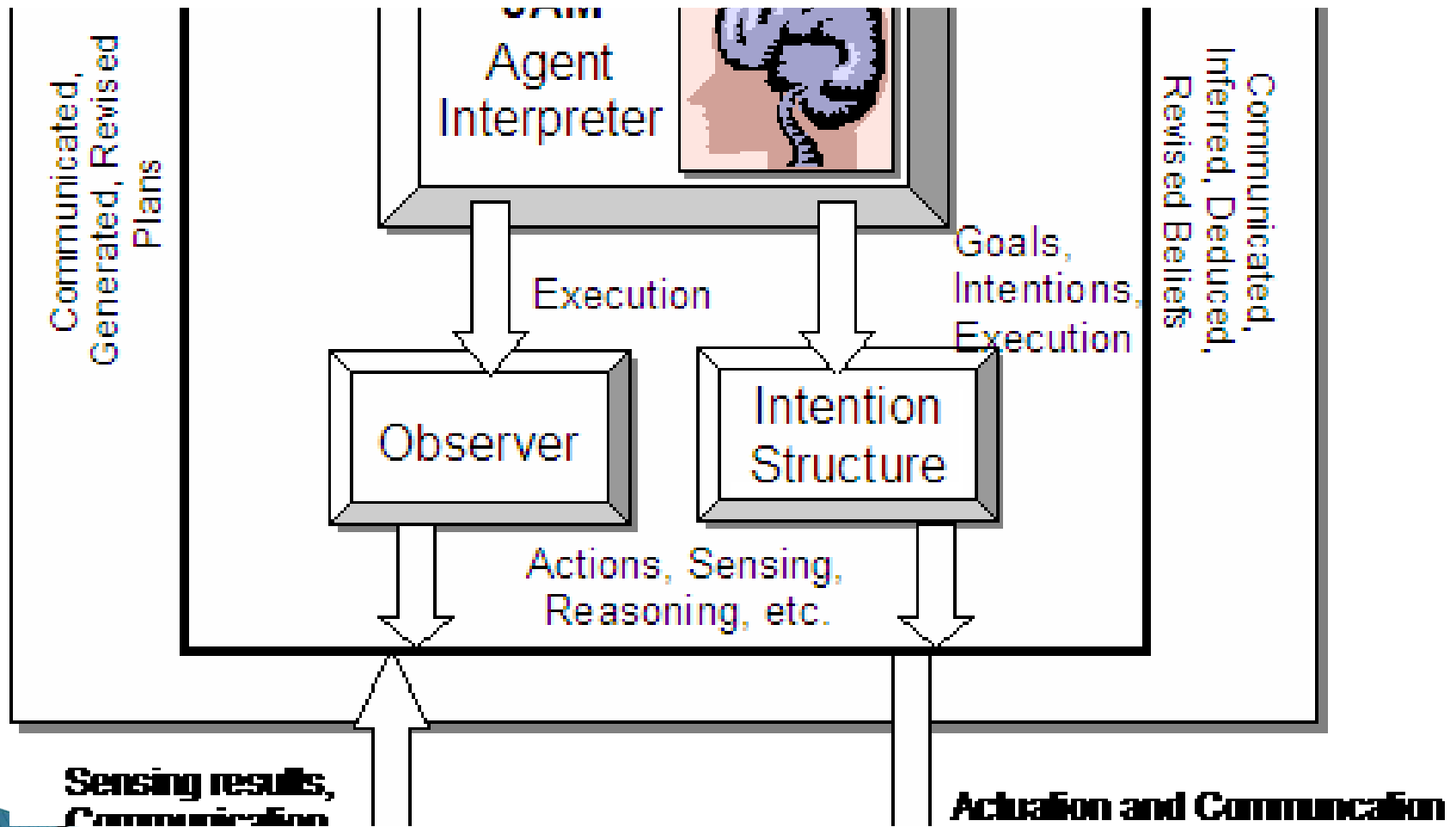
JAM

- ▶ Arquitetura de agentes inteligentes
- ▶ Desenvolvido pela Intelligent Reasoning Systems (I.R.S.)
 - Oceanside, California
- ▶ Linguagem
 - JAM

JAM

- ▶ Características
 - Limitações/Bugs reportados
 - Nenhuma aplicação comercial conhecida
 - Nenhuma ferramenta de desenvolvimento

JAM



Jason

- ▶ Interpretador para uma versão estendida da linguagem AgentSpeak(L), Open Source distribuído sob a licença GNU LGPL
- ▶ Desenvolvido por
 - Jomi F. Hübner (Blumenau, BR)
 - Rafael H. Bordini (Porto Alegre, RS)
- ▶ Linguagem
 - Agent Speak (L)



Jason

▶ Características

- Agent Speak (L) possui semântica formal
 - Possibilita verificação formal
- Ferramentas de Desenvolvimento
- Disponível plug-in para Eclipse

Jason

- ▶ Agent Speak (L)
 - Linguagem de programação orientada a agentes
 - Baseada na lógica de primeira ordem
 - Inspirada na
 - Arquitetura BDI
 - Lógica BDI

Jason

The screenshot shows the Jason IDE interface. The main editor displays the following code:

```
1 /* Initial beliefs and rules */
2
3 available(beer,fridge). // initially, I believe that there are some beer in t
4 limit(beer,10). // my owner should not consume more than 10 beers a d
5
6 too_much(B) :-
7   .date(YY,MM,DD) & .count(consumed(YY,MM,DD,-,-,,-, B),Qt dB) &
8   limit(B,Limit) & Qt dB > Limit.
9
10 /* Plans */
11
12 +!has(owner,beer) : available(beer,fridge) & not too_much(beer)
13   <- lat(robot,fridge);
14     open(fridge);
15     get(beer);
16     close(fridge);
17     lat(robot,owner);
18     hand_in(beer);
19     ?has(owner,beer);
20 // remember that one beer is consumed
```

The left sidebar shows a Project Viewer with a tree structure for 'robot.asl':

- +!has(owner,beer)
- +!has(owner,beer)
- +!has(owner,beer)
- !has(...)
- +lat(robot,P)
- +lat(robot,P)
- +delivered(beer,Qt)
- +stock(beer,0)
- +stock(beer,N)
- +?time(T)

The bottom panel contains a 'Jason console' with the following output:

```
Launching DomesticRobot.mas2j
Parsing project file... parsed successfully!
Parsing AgentSpeak file 'supermarket.asl'... parsed successfully!
```

The bottom right corner shows 'Project agents' with a list: robot, owner, supermarket.

At the bottom of the window, the status bar displays: 13,20 Top (asl,none,ISO-8859-1) - --- U 8/14 Mb

Jason

The screenshot displays the Jason Mind Inspector interface, titled "Jason Mind Inspector :: cycle 22 ::". The "Agents" list on the left shows "r2" and "r1", with "r1" selected. The main area, "Agent Inspection", shows the "Inspection of agent r1 (cycle #12)".

- Beliefs

- pos(back,3,0)_[source(self)]:
- pos(r1,3,0)_[source(percept)]:
- pos(r2,3,3)_[source(percept)]:
- garbage(r1)_[source(percept)]:

- Events

Sel	Trigger	Intention
X	+!ensure_pick(garb)	4

+ Options

- Intentions

Sel	Id	Pen	Intended Means	Stack (show details)
X	4		+!ensure_pick(S)	{ S = garb }
			+!take(S,L)	{ S = garb, L = r2 }
			+!carry_to(R)	{ R = r2, Y = 0, X = 3 }
			+garbage(r1) _[source(percept)]	

Actions

Pend	Feed	Sel	Term	Result	Intention
X		X	pick(garb)	false	4

Agent History

A timeline slider shows the history from Cycle 0 to Cycle 22, with a blue marker at Cycle 12.

At the bottom, there is a "Run" button, a "5" cycle(s) for field, a dropdown menu set to "all agents", and a "view as:" dropdown menu set to "html".

Jadex

Jadex



BDI Agent System

Jadex

- ▶ Mecanismo de raciocínio BDI para agentes inteligentes
- ▶ Projeto conduzido pelo Distributed Systems and Information Systems Group
 - University of Hamburg, Alemanha
- ▶ Linguagem
 - Java e XML

Jadex

▶ Características

- Não introduz nova linguagem
- FIPA Compliant
 - Uso do JADE como plataforma SMA
- Integração com ferramenta de projeto de Ontologias
 - Protégé
- Ferramentas de Desenvolvimento
- Aplicações Comerciais
 - MedPAge
 - Dynatech
 - Bookstore

Jadex

▶ Principais Componentes

◦ Belief

- Conhecimento do agente sobre ambiente e si mesmo
- Podem ser qualquer objeto Java
- Armazenadas em uma base de crenças
- Permite consulta através de OQL-like query language

◦ Goal

- Orientam ações do agente
- Desejos concretos e momentâneos do agente
- Agente executa ações apropriadas até que o objetivo seja considerado
 - Atingido
 - Inatingível
 - Não mais desejado

Jadex

▶ Principais Componentes

◦ Plan

- Forma como o agente atuará em seu ambiente
- Dependendo da situação corrente
 - Planos selecionados como resposta à ocorrência de eventos ou de objetivos
- Seleção de planos feita automaticamente pelo sistema

◦ Capability

- Crenças, planos e objetivos podem ser colocados em um módulo de agente
- Podem conter subcapacidades formando uma hierarquia
- Possibilidade de reuso

Jadex

▶ Principais Componentes

◦ Event

- Importante propriedade dos agentes
 - Capacidade de reagir a diferentes tipos de eventos
- Jadex suporta dois tipos de eventos a nível de aplicação
 - Eventos internos
 - Usados para denotar uma ocorrência dentro de um agente
 - Eventos mensagem
 - Comunicação entre dois agente ou mais
- Normalmente tratados por planos

Jadex – Código Exemplo

```
package jadex.examples.helloworld;
```

```
import jadex.runtime.Plan;
```

```
/**
 * The hello world plan prints out a short welcome message.
 */
public class HelloWorldPlan extends Plan
{
    //----- methods -----

    /**
     * Handle the ping request.
     */
    public void body()
    {
        System.out.println("\nHello world!");
        waitFor(2000);
        System.out.println("\n"+getBeliefbase().getBelief("msg").getFact());
        waitFor(2000);
        System.out.println("\nSee you. Bye! ");

        killAgent();
    }
}
```


Jadex

```
name="HelloWorld"  
package="jadex.examples.helloworld">  
  
<beliefs>  
  <belief name="msg" class="String" exported="true">  
    <fact>"Welcome to Jadex."</fact>  
  </belief>  
</beliefs>  
  
<plans>  
  <!-- Plan which prints out a message and kills the agent. -->  
  <plan name="hello">  
    <body class="HelloWorldPlan"/>  
    <!-- <body>new HelloWorldPlan()</body> -->  
  </plan>  
</plans>  
  
</configuration>
```

Jadex – Control Center

Jadex Control Center 0.96 (2007/06/01): Project Apps2

File Model Help

classes (C:\projects\jadex\applications\classes)

- jadex
 - benchmarks
 - examples
 - alarmclock
 - blackjack
 - blocksworld
 - booktrading
 - cleanerworld
 - garbagecollector
 - helloworld
 - GoodbyeWorld.agent.xml
 - HelloWorld.agent.xml
 - hunterprey
 - marsworld
 - ping

Settings

Filename: ...

Configuration: default

Agent name: GoodbyeWorld Auto generate

Start Reload Reset Help

Description

GoodbyeWorld

The GoodbyeWorld agent.

It has the task to print out a goodbye message when the agent is terminated. The agent terminates itself after it has lived 2 seconds.

Name	Address
lars	
ams@lars	nio-mtp://vsiisstaff3:8976
df@lars	nio-mtp://vsiisstaff3:8976
jcc@lars	nio-mtp://vsiisstaff3:8976

Console Output

Jadex – BDI Tracer

The screenshot displays the Jadex BDI Tracer interface. On the left, a list of agents is shown: ams@lars, df@lars, jcc@lars, and Blocksworld. The main area contains a table with the following columns: #, Agent, Name, Content, Cause, and Time. The table lists various actions performed by the Blocksworld agent, including creating and clearing goals, and manipulating blocks and buckets. To the right of the table is a 'Tracing Settings' panel with several checkboxes: Trace Belief Reads (unchecked), Trace Belief Writes (checked), Trace Goals (checked), Trace Plans (checked), Trace Messages (checked), Trace Internal Events (unchecked), and Trace Actions (unchecked). Below these settings is a 'Nodes Limit' dropdown set to 100, and 'Clear' and 'Apply' buttons. At the bottom of the interface is a network diagram showing a central node connected to several other nodes, which are further connected to a larger network of nodes.

#	Agent	Name	Content	Cause	Time
61	Blocks...	ta...	Table	stack#10	26.34....
62	Blocks...	st...	RAchieveGoal(name=stack#16)	configure#3	26.34....
64	Blocks...	cle...	RAchieveGoal(name=clear#31)	stack#11	26.34....
65	Blocks...	cle...	RAchieveGoal(name=clear#32)	stack#11	26.34....
66	Blocks...	bl...	[5]Block 6	stack#11	26.34....
67	Blocks...	bu...	Bucket	stack#11	26.34....
68	Blocks...	ta...	Table	stack#11	26.34....
69	Blocks...	st...	RAchieveGoal(name=stack#17)	configure#3	26.34....
71	Blocks...	cle...	RAchieveGoal(name=clear#33)	stack#12	26.34....
72	Blocks...	cle...	RAchieveGoal(name=clear#34)	stack#12	26.34....
73	Blocks...	bl...	[6]Block 7	stack#12	26.34....
74	Blocks...	bu...	Bucket	stack#12	26.34....
75	Blocks...	ta...	Table	stack#12	26.34....
76	Blocks...	st...	RAchieveGoal(name=stack#18)	configure#3	26.34....
79	Blocks...	cle...	RAchieveGoal(name=clear#35)	stack#12	26.34....

Jadex – Test Center

Jadex Control Center 0.96 (2007/06/01): Project Apps2

File Model Help

classes (C:\projects\jadex\appl...)

- jadex
 - benchmarks
 - examples
 - testcases
 - beliefs
 - DABidder.agent.xml
 - DAInitiator.agent.xml
 - InitialEvent.agent.xml
 - MessageForwarder.agent.xml
 - MessageMatcher.agent.xml
 - ProtocolInitiator.agent.xml
 - ProtocolReceiver.agent.xml
 - SendAndWait.agent.xml
 - SendAndWaitMol.agent.xml
 - capabilities
 - events
 - goals
 - AbstractQueryGoal.agent.xml
 - AchieveGoal.agent.xml
 - Candy.agent.xml
 - DropGoal.agent.xml
 - GoalBindings.agent.xml
 - GoalConditions.agent.xml
 - GoalFinishedTrigger.agent.xml
 - GoalInhibition.agent.xml
 - GoalRecur.agent.xml
 - GoalReferencePoint.agent.xml
 - GoalRegression.agent.xml
 - InnerAbstractGoal.agent.xml
 - MaintainGoal.agent.xml

Test suite settings

Test cases [86]

- C:\projects\jadex\applications\classes\jadex\testcases\beliefs\Array.agent.xml
- C:\projects\jadex\applications\classes\jadex\testcases\beliefs\BeanChanges.agent.xml
- C:\projects\jadex\applications\classes\jadex\testcases\beliefs\BeliefChanges.agent.xml
- C:\projects\jadex\applications\classes\jadex\testcases\beliefs\BeliefSetChanges.agent.xml
- C:\projects\jadex\applications\classes\jadex\testcases\beliefs\BeliefSetContains.agent.xml
- C:\projects\jadex\applications\classes\jadex\testcases\beliefs\DynamicBelief.agent.xml
- C:\projects\jadex\applications\classes\jadex\testcases\beliefs\Evaluationmodes.agent.xml
- C:\projects\jadex\applications\classes\jadex\testcases\beliefs\InitialBelief.agent.xml

Allow including the same test more than once

Testcase timeout [ms]: 20000

Testcase concurrency: 5

Add Load Save Clear

Test suite execution

State: Running

Performed: 41/86 in 27,8s Failed: 3/86

Abort Save Clear

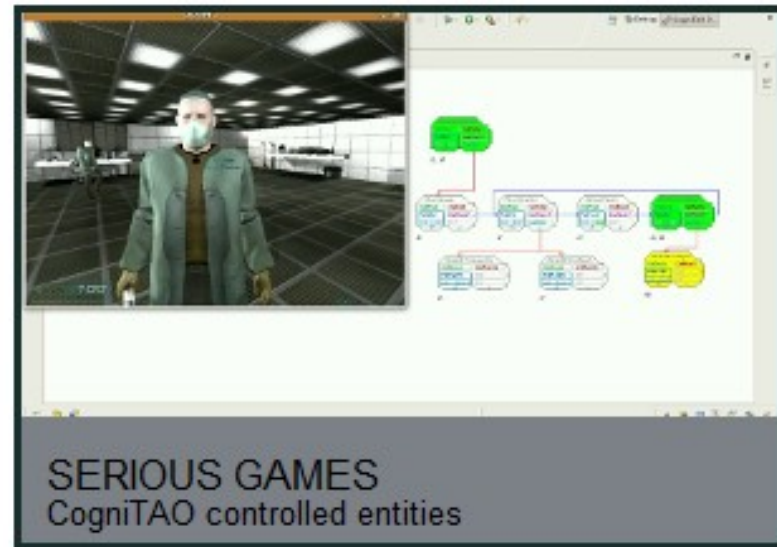
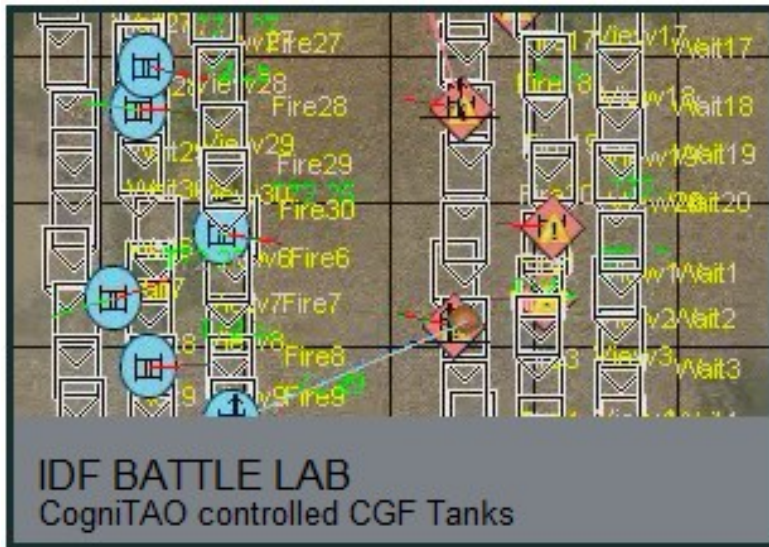
Details

Performed 40 of 86 Test Cases

- C:\projects\jadex\applications\classes\jadex\testcases\beliefs\Array.agent.xml

Performing test C:\projects\jadex\applications\classes\jadex\testcases\misc\DynamicTrigger.agent.xml

CogniTAO



CogniTAO

- ▶ Vem sendo utilizado para simulação de forças militares, como para médicos virtuais em treinamentos baseados em games.
- ▶ Compatível com diversas engines de games (como Doom e Unreal Engines)
- ▶ Desenvolvida por Cogniteam LTD.
- ▶ Linguagem
 - C++

CogniTAO

- ▶ Diversas implementações comerciais.
- ▶ Alguns clientes: Israel Aeospace Industries, Bar Ilan University, Ministério da Defesa de Israel.

CogniTAO – IDE

The screenshot displays the CogniTAO IDE interface. The top menu bar includes File, Edit, Refactor, Source, Run, Navigate, Search, Project, CogniTAO Menu, Window, and Help. The main workspace is divided into several panes:

- Navigator:** Shows a tree view of the project structure, including folders like .settings, conditions, decisions, deploy_folders, naoqi-1.8.16, plans, and variables.
- CogniTAO View:** Displays a task plan diagram for PinTestMotion.xml. The plan starts with a PinTestMotion task, followed by a sequence of tasks: TskStand, TskLookLeft, TskLookRight, and TskSquat. Each task is represented by a box with 'Condition' and 'EndMethod' fields.
- Code Editor:** Shows the implementation of the TskSquat task in TskSquat.cpp. The code defines a void TskSquat::action() method that sets task_done to true and updates the signal.
- CogniTAO World Model View:** A table showing the current state of the world model.

Plan	Name	Type	Value	Shared
<input type="checkbox"/>	PinTestMotion	task_done	false	

The Cogniteam logo is visible in the bottom left corner, and the bottom right corner shows the text 'writable Smart insert 24:28'.

Comparação

	Linguagem
JACK	JACK (extensão de Java)
Jadex	Java e XML
JAM	JAM (extensão de Java)
Jason	Agent Speak (L)
CogniTAO	C++

Comparação

	Ferramentas de Desenvolvimento
JACK	IDE e Debug
Jadex	Ferramentas para execução, debug e documentação
JAM	-
Jason	IDE e Mind Inspector
CogniTAO	Plug in para IDE, Ferramentas para execução, debug e documentação