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

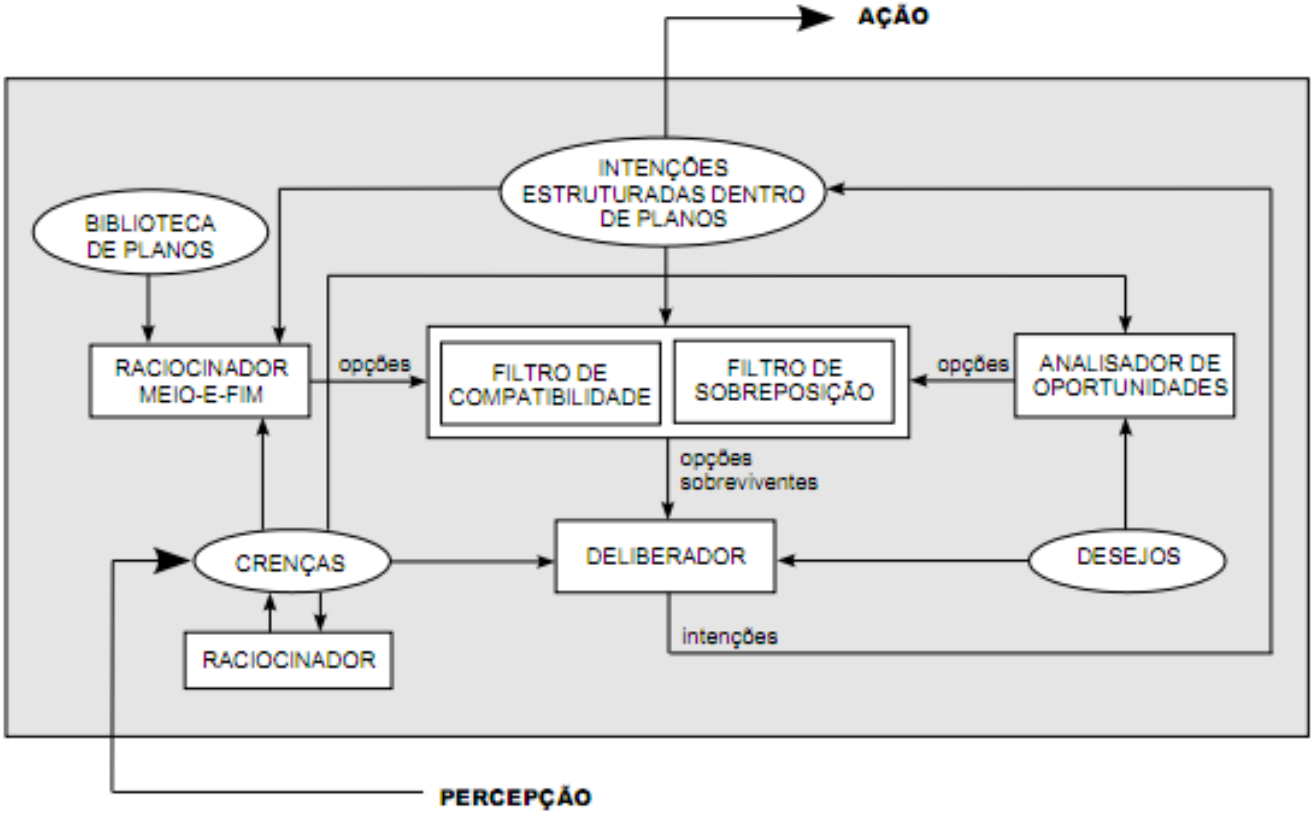
BDI

- 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

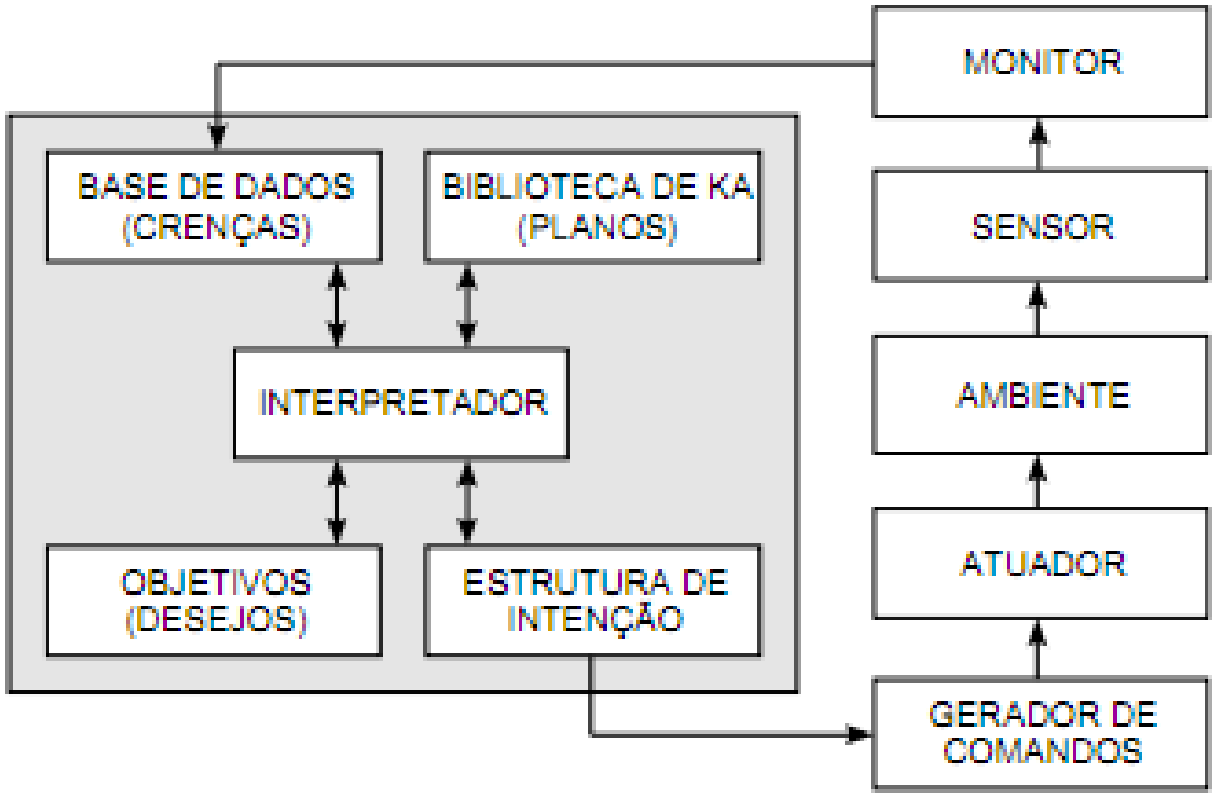
Arquitetura abstrata Intelligent Resource-bounded Machine Architecture (IRMA)



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
  - CogniTAO

JACK™ Intelligent Agents

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

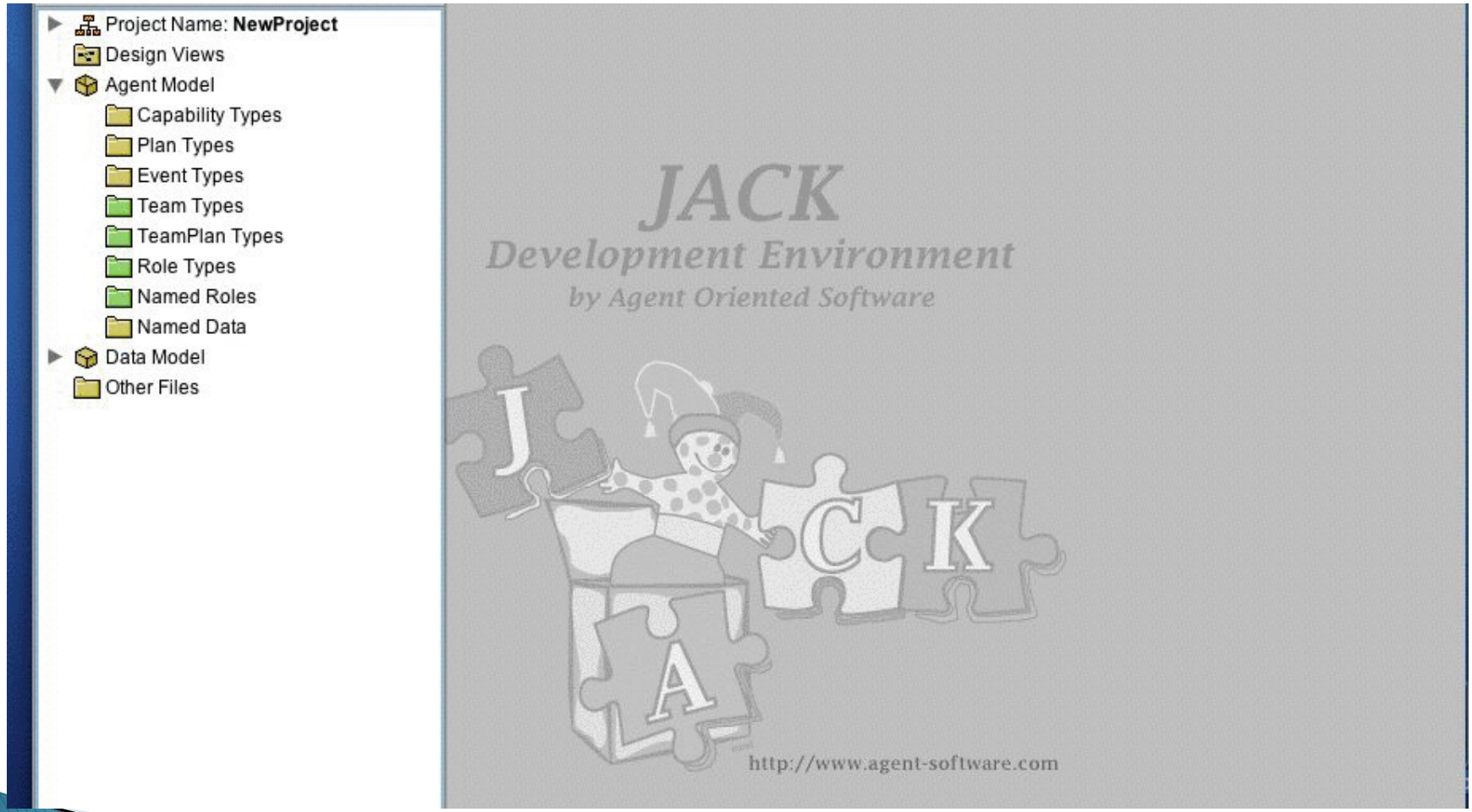
JACK

- 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



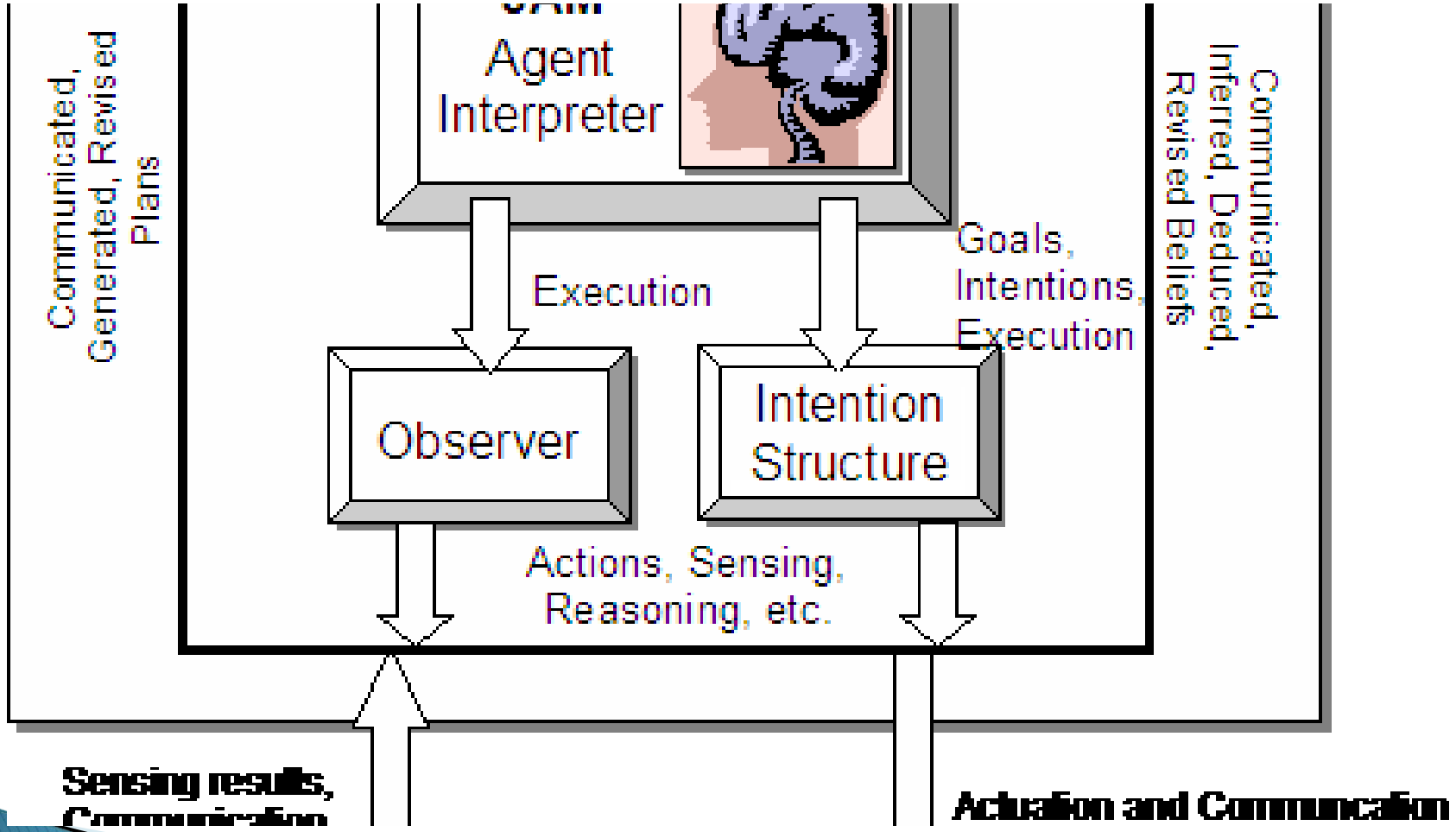
JAM

- 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


Agent
Interpreter
Communicated,
Generated, Revised
Plans
Execution
Goals,
Intentions,
Execution
Communicated,
Inferred, Deduced,
Revised Beliefs
Observer
Intention
Structure
Actions, Sensing,
Reasoning, etc.
Sensing results,
Actuation and Communication

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)

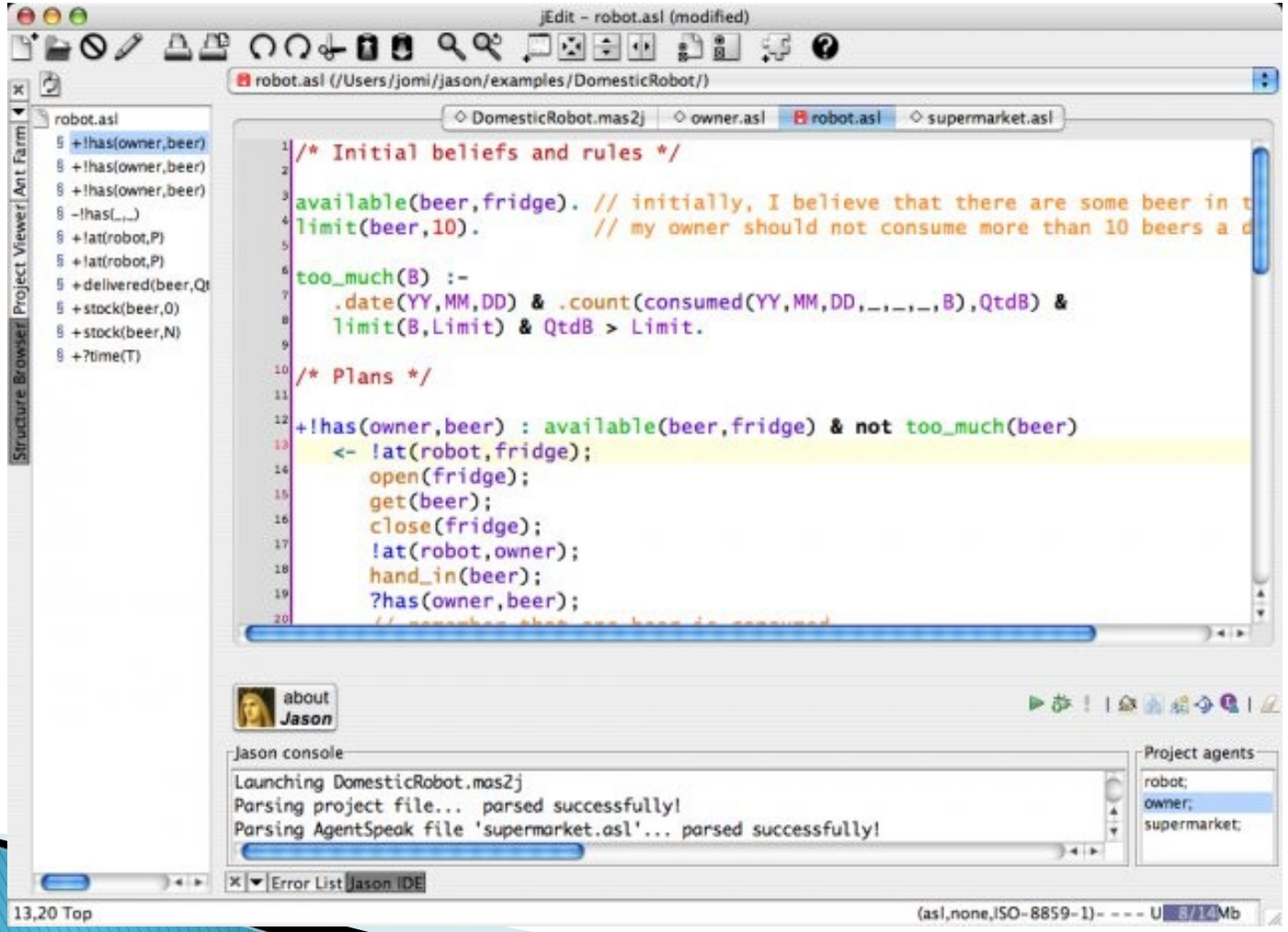
Jason

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

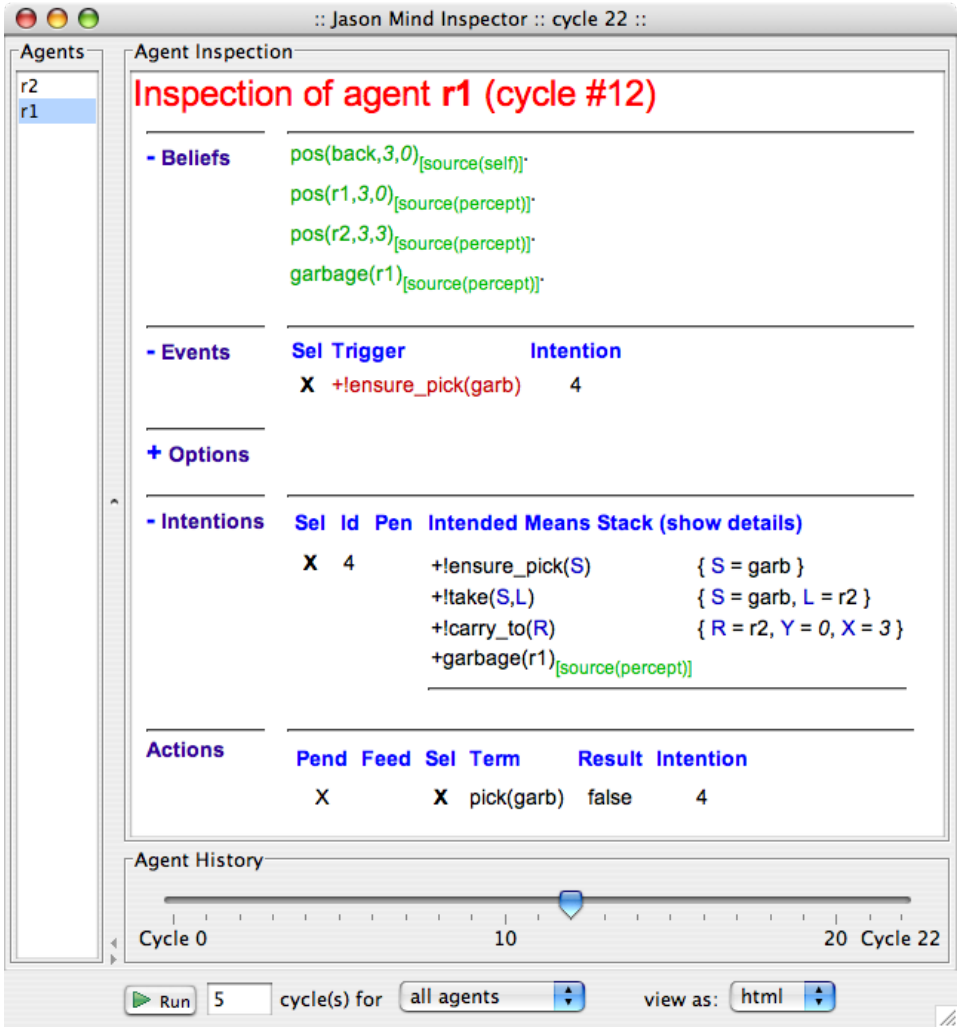
Jason

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

Jason


jEdit - robot.asl (modified)
robot.asl (/Users/jomi/jason/examples/DomesticRobot/)
DomesticRobot.mas2j
owner.asl
robot.asl
supermarket.asl
/* Initial beliefs and rules */
available(beer,fridge). // initially, I believe that there are some beer in t
limit(beer,10). // my owner should not consume more than 10 beers a d
too_much(B) :-
.date(YY,MM,DD) & .count(consumed(YY,MM,DD,_,_,_,B),QtdB) &
limit(B,Limit) & QtdB > Limit.
/* Plans */
+!has(owner,beer) : available(beer,fridge) & not too_much(beer)
<- !at(robot,fridge);
open(fridge);
get(beer);
close(fridge);
!at(robot,owner);
hand_in(beer);
?has(owner,beer);
robot.asl
+!has(owner,beer)
+!has(owner,beer)
+!has(owner,beer)
-!has(_,_)
+!at(robot,P)
+!at(robot,P)
+delivered(beer,Qt
+stock(beer,0)
+stock(beer,N)
+?time(T)
Structure Browser
Project Viewer
Ant Farm
about Jason
Jason console
Launching DomesticRobot.mas2j
Parsing project file... parsed successfully!
Parsing AgentSpeak file 'supermarket.asl'... parsed successfully!
Project agents
robot;
owner;
supermarket;
Error List
Jason IDE
13,20 Top
(asl,none,ISO-8859-1)

Jason


:: Jason Mind Inspector :: cycle 22 ::
Agents
r2
r1
Agent Inspection
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
Cycle 0
10
20
Cycle 22
Run
5
cycle(s) for
all agents
view as:
html

Jadex

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

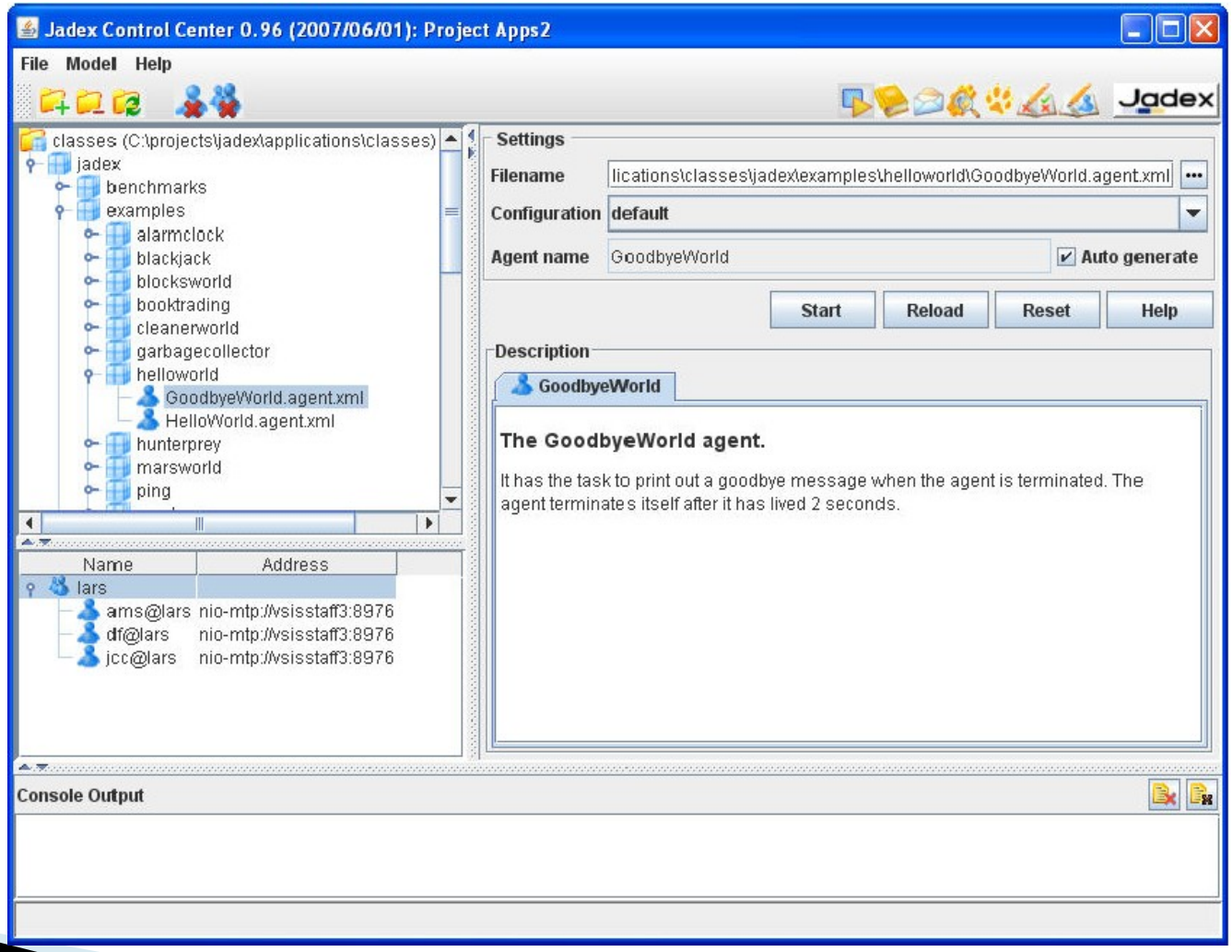
Jadex – Código Exemplo

```
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();
    }
}
```

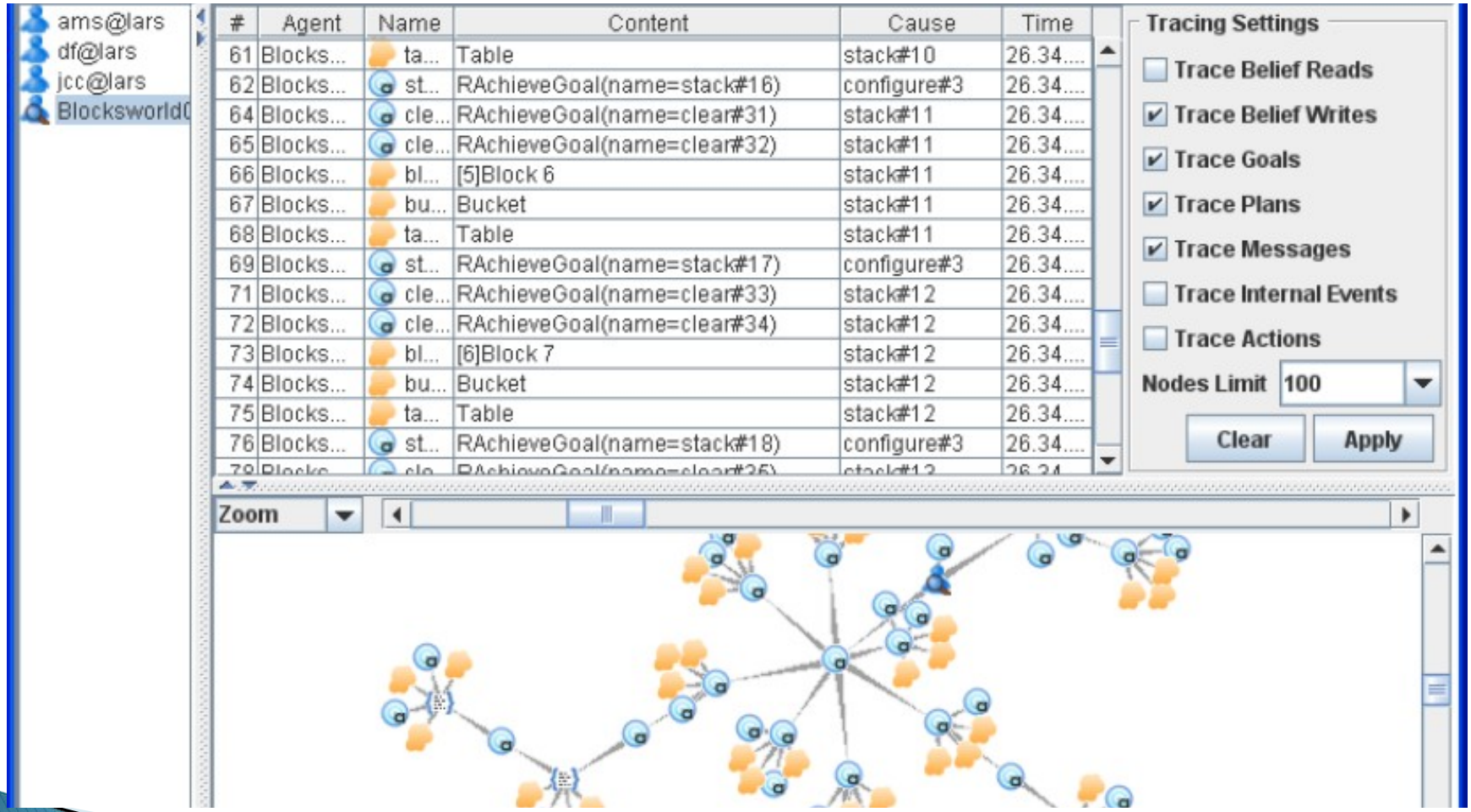
Jadex

```
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>
```

Jadex – Control Center

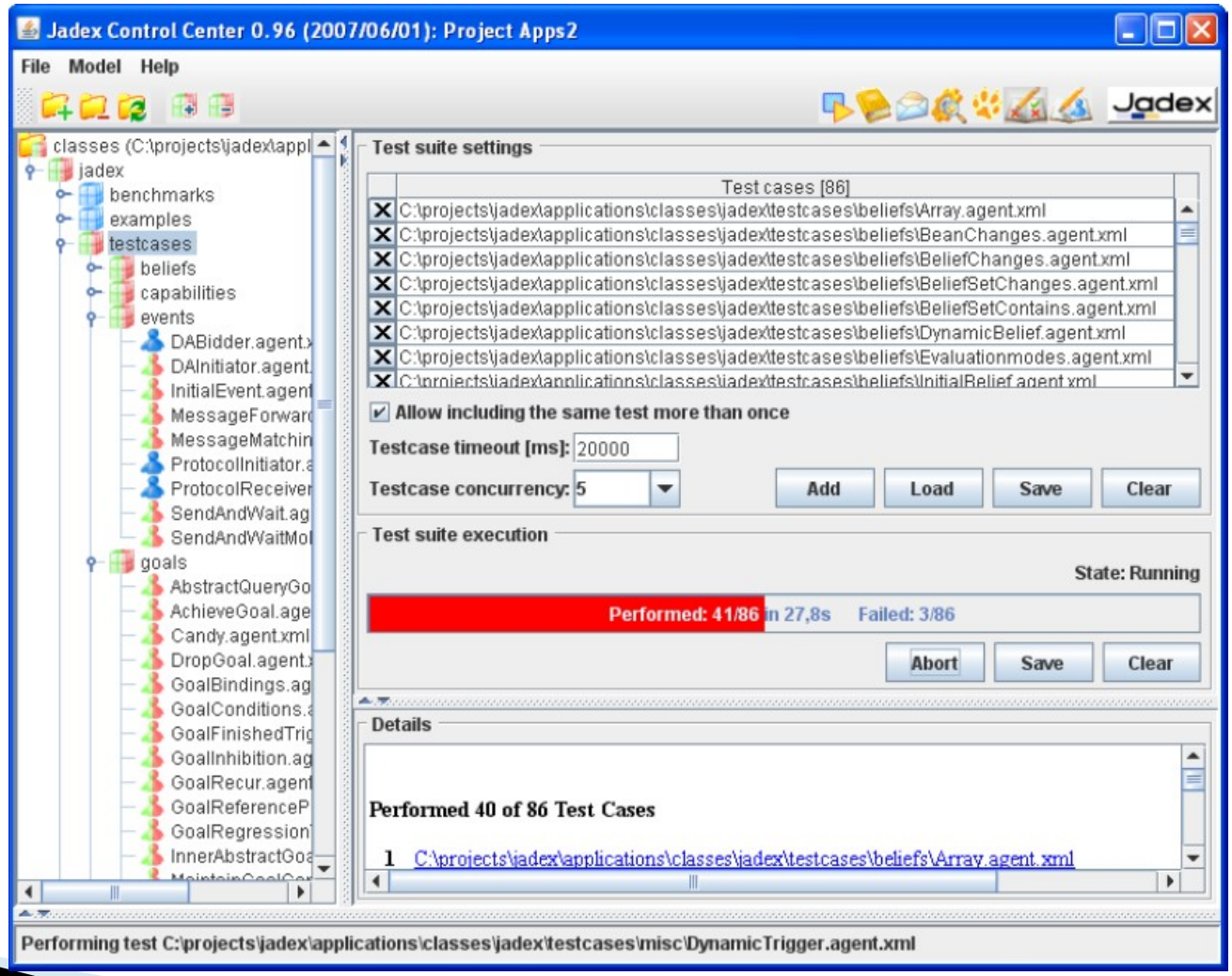


Jadex – BDI Tracer



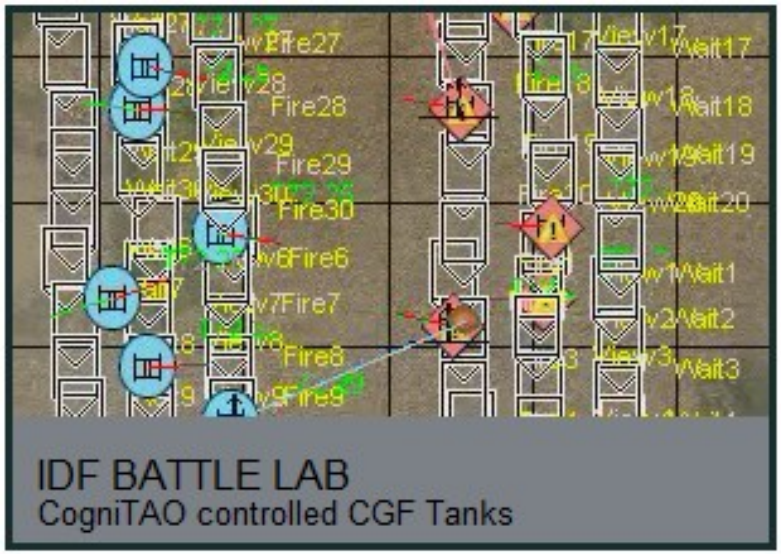
ingridnunes@gmail.com

Jadex – Test Center

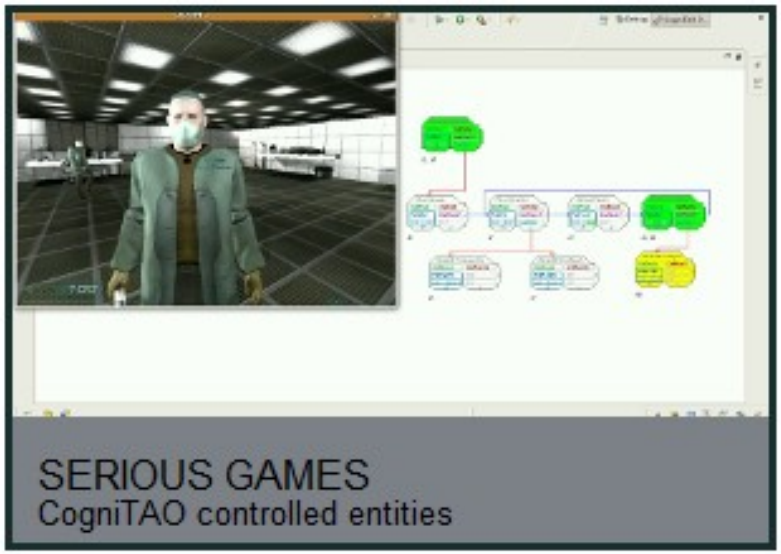


ingridnunes@gmail.com

CogniTAO



IDF BATTLE LAB
CogniTAO controlled CGF Tanks



SERIOUS GAMES
CogniTAO controlled entities

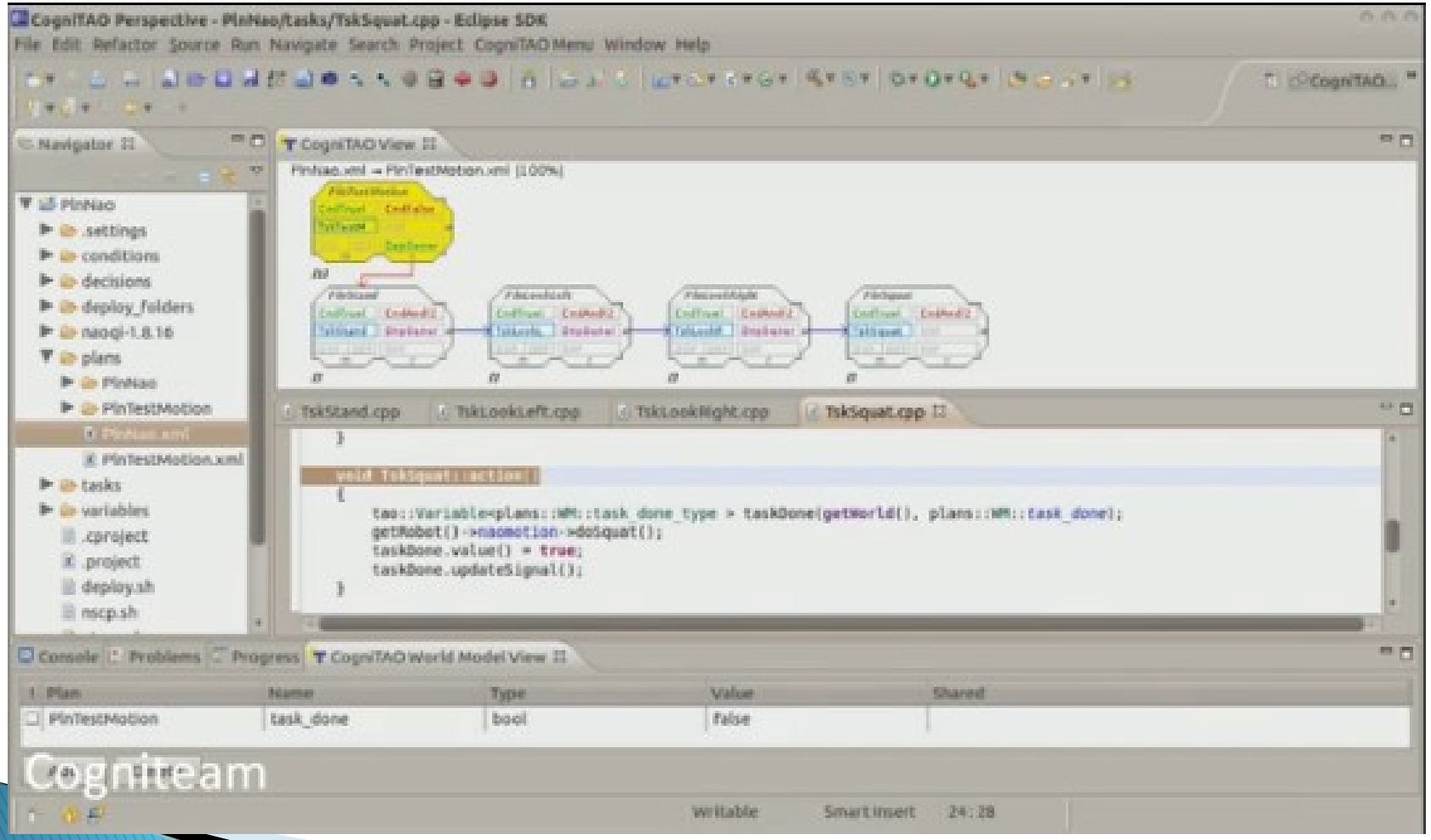
CogniTAO

- Vem sendo utilizando 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 comercias.
- Alguns clientes: Israel Aeorspace Industries, Bar llan University, Ministério da Defesa de Israel.

CogniTAO – IDE



Cogniteam

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