Multi-agent Toolkits

Paolo Giorgini

DISA
University of Trento
pgiorgini@cs.unitm.it

... WOW ...

- AgentBuilder
- Agent Building Shell
- Agent Development Environment
- AgentX
- Agles Workbench
- Bee-gent
- Bond
- Cybele
- DECAF
- dMARS
- Grasshopper
- Gyp
- IMPACT/CATS
- InfoSpider
- Infosphere Infrastructure
- Intelligent Agent Library/Infactory
- Jack
- Jam
- Java Agent Template/JavaScript
- Jafinas
- Java-To-Go
- Kafka
- Kaos
- Sodabot
- Knowbot
- KYMA-Atlantis
- LALO
- LiveAgent Pro
- Mole
- Odyssey
- Open Agent Architecture
- OSCAR
- RETSINA
- WAVE
- Zeus
How to choose?

(1)

Agent Capabilities:

- **autonomy**: agents operate without direct intervention of humans and/or of other agents;
- **social ability**: agents interact with each other; usually achieved through the provision of an agent communication language;
- **reactivity**: agents perceive their environment; usually achieved through an explicit model of the environment with an active or passive interface;
- **pro-activeness/rationality**: agents exhibit goal-directed behavior and act as to achieve their goal; usually achieved through explicit representation of goal and the provision of a planner;
- **mentalist notions**: agents have explicit representations for e.g. knowledge, belief, intention;
- **mobility**: agents are mobile, i.e. the code and data can move from one machine to another;

How to choose?

(2)

Practical Criteria:

- **implementation language**: the programming language in which the agents will be implemented;
- **hardware**: the hardware on which these agents can exist;
- **availability**: the price and obtainability of the platform;
- **standards**: the standards supported by this platform;
- **domains**: the domains this platform has been used in;
- **success stories**: evidence for successful use of the platform;
<table>
<thead>
<tr>
<th>Platform</th>
<th>A. access</th>
<th>Search</th>
<th>Scalability</th>
<th>Pre-active</th>
<th>Security</th>
<th>User-friendly</th>
<th>S. effectiveness</th>
<th>Reliability</th>
<th>Robustness</th>
<th>S. security</th>
<th>Naming</th>
<th>Unknown</th>
<th>Emulation</th>
<th>Unknown</th>
<th>Executable</th>
<th>S. usage issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentRat the Pro</td>
<td>nice</td>
<td>yes</td>
<td>yes</td>
<td>nice</td>
<td>yes</td>
<td>nice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agent Heat Dog Hunt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agent Development Environment</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agents</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AgentWorkshops</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inspector</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cylane</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAVEA</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dBASES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrossTalk</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CyberCat</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMPACTCATS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intralinks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internetphone Interconnection</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPtoolkit</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANAS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetAgent</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Satellite</td>
<td>nice</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eloyal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X: TALK</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L:WUN</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USENet and Free</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenAgent</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenAgent At the Source</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONSCAR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R:STRAIN</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENUM</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coda</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Jafmas**
University of Cincinnati
www.ececs.uc.edu/~abaker/JAFMAS

- Framework to guide the development of multi-agent systems
- Set of Java Classes for agent development
- It supports speech-act performatives, but does not use KQML standard
- Scalable and supports peer to peer communication
- Difficult to set up
- Free

---

**JATLite**
Center for Design Research - Stanford University
java.stanford.edu/java_agent/html/

- Set of Java packages + a Java runtime router
- Based on the client/server mechanism
- KQML based + FTP and SMPT extensions
- Robust communications platform
- FIPA compliant
- Slow KQML parser (?? for real time systems)
- Difficult to set-up
- Free
Agent Builder
Reticular Systems, Inc., San Diego, CA
www.agentbuilder.com

- Integrated tool for constructing intelligent software agents: Toolkit + run-time System
- Developed in Java and the agents are Java programs
- KQML based
- FIPA compliant
- Scalability problems
- AgentBuilder Lite ($495) and Pro ($4995)

JACK Intelligent Agents™
Agent Oriented Software Pty Ltd
Australia
http://www.agent-software.com
Brief History

- **Procedural Reasoning System (PRS)**
  - developed by SRI International in the mid '80s
  - Lisp
- **dMARS**
  - built in the mid '90s by Australia Artificial Intelligence Institute
  - C++
- **Jack**
  - developed by Agent Oriented Software (AOS), 1998
  - Java
  - 1.3 released May 1999
  - 2.0 November 1999

Entirely written in Java

- **Portability**:
  - capable of running on any system on which Java is available
  - laptops to high-end multi-CPU Enterprise servers
- **Access to all Java capabilities, including**:
  - multiple threads possibly running on multiple CPUs
  - platform independent GUIs
  - third party libraries such as JDBC
- **Easy integration with external packages using standard infrastructure, e.g., CORBA**
The BDI Agent Model

Human \(\rightarrow\) Belief, Desire, Intentions Agent

**Execution Engine**

Beliefs - database of perceived world knowledge

Goals or desires

Pre-compiled plans

JACK BDI Execution

beliefs

desires

intentions

messages

databases

events

plans

assert retract

observe change

select plan (plan choice)

handle

post

step intention
The Framework

Jack consists of three main extensions to Java

(1) **Syntactical**
- keywords for the identification of the main components of an agent (such as agent, plan, and event)
- sentences for the declaration of attributes and other characteristics of the components (for instance, the information contained in beliefs or carried by events)
- sentences for the definition of static relationship (for instance, which plans can be adopted to react to a certain event)
- sentences for the manipulation of the agent’s state (for instance, additions of new goals or sub-goals to be achieved, changes of beliefs, interaction with other agents)

(2) A **compiler** that converts the syntactic additions into pure Java classes and statement that can be loaded with, and be called by, other Java code.

(3) A set of classes (called **kernel**) provides the required run-time support to the generated code:
- the automatic management of concurrency among tasks being pursued in parallel (*intentions* in the BDI terminology);
- the default behavior of the agent in react to events, failure of actions and tasks, and so on; and
- communicative infrastructure for multi-agent applications.
Agents

- A type:
  - `agent AgentType extends Agent {...}

- Encapsulates knowledge and behaviour (databases, events and plans)
- Reacts to events -- performs tasks
- Receives messages -- performs services
- Interfaces other system components
  - GUI, back-end, other processes

Agent Type Definition

```java
gagent <AgentType> extends Agent {
    events handled by this agent;
    belief structures of this agent;
    plans of this agent;
    capabilities of this agent;
    ... (construction, etc.) ...
}
```
Capability Concept

- A type:
  - \texttt{capability CapabilityType} extends \texttt{Capability} {...}
- Allows agent elements to be structured
- Provides functionalities
- Provides interactions between functionalities

Capability Type Definition

\begin{verbatim}
capability <CapabilityType> extends Capability {
  \textit{external} events for this capability;
  \textit{exported} belief structures;
  \textit{imported} belief structures;
  \textit{internal} events for this capability;
  \textit{private} belief structures;
  plans of this capability;
}
\end{verbatim}
Events

- A type:
  - `event EventType extends Event {...}

- Provides the connections between agents and plans
  - both agents and plans must declare the events they handle as well as the events they post or send

- `Event`, `MessageEvent`, `BDIGoalEvent`, ...

---

Event definition in Jack

```jack
event <EventType> extend BDIGoalEvent {
    data members;
    posting methods;
}
```
Belief Structures

- JACK Databases
  - relational data modelling
- A type:
  - `database DatabaseType extends ClosedWorld {...}`
- Defines knowledge capability (relational modelling)
  - closed world / open world semantics

Databases definition in Jack

database <DBType> extends ClosedWorld {
  key and value field declarations;
  query declarations;
  modification call-back definitions;
}

Plans

- A type:
  - `plan PlanType extends Plan {...}"`
- Defines context dependent responses to event occurrences
- Plan processing *succeeds or fails*
- Reasoning methods
  - requires each step to succeed
  - each step processed atomically
- Meta-level plans to handle plan choice

Plans definition in Jack

```java
Plan <PlanType> extends Plan {
    relevance: event type handled;
    context for applicability;
    knowledge base access;
    plan body - reasoning method;
    ...(other reasoning methods encapsulating plan steps)...
}
```
An example

Agents “ping” each other, that is, exchange empty messages

- The message being exchanged is represented as an event that originates with one agent and is received by another:

```java
event PingEvent extends MessageEvent {
    int value;
    #posted as
    ping(int value)
    {
        this.value=value;
    }
}
```

An example

```java
plan BouncingPlan extends Plan {
    #handle event PingEvent ev;
    #sends event PingEvent pев;
    body()
    {
        @send ( ev.from, pев.ping ( ev.value +1 ));
        // Replay to the sender of the event
    }
}
```
An example

agent PingAgent extends Agent {
    # handles event PingEvent;
    # uses plan PingPlan;
    # posts event PingEvent pev;
    void ping (String other)
    {
        send ( other, pev.ping(1));
    }
}

Applications

- **AOS:**
  - Business procedures
  - workflow
  - simulation of human reasoning
- **Trento:**
  - Culture project
  - Knowledge management
Reference

- Jack intelligent agents. Agent Oriented Software Pty Ltd. http://www.agentsoftware.com