Artificial Intelligence: Methods and Applications


Juan Carlos Nieves Sánchez
December 16, 2014
Outline

• BDI Model.
• Some Multiagent Platforms
Characterization of an intelligent agent

In general, intelligent (software) agents are expected to express some kind of behavior which – to some degree – resembles the human mind's capability of problem solving.

A popular definition of the properties of an intelligent agent are:

- **Autonomy**: An agent executes actions on its own incentive, not (generally) depending on the interaction with external entities like a human user.
- **Proactivity**: An agent shall be able decide about actions which purposefully bring it closer to achieving its goals.
- **Reactivity**: An agent reacts to changes in its environment, adapting its plans appropriately.
- **Social capabilities**: An agent is capable of exchanging information with other agents and utilizes it for achieving its goals.
A Generic Multi-Agent System Architecture

Execution engine
scheduler, single
thread, multiple
threads

Agents
Individual state and
behavior

Platform design

Platform services
via application programming interface

Messaging  Management  Mobility  ...

Application programming

Platform design; functional enhancements

.execute

call
Rational Behaviour

Practical reasoning according to Believes, Desires and Intentions (BDI) model.
BDI as a model for MAS-Platforms

In order to design a platform with BDI support, we could at least:

- deliver suitable programming elements (classes, components) to represent beliefs, desires, and intentions;

- run some algorithms following the practical reasoning notion, or

- implement some of the high-level processes like “build plan” (means-end reasoning) or “pick intention” (deliberation).
Social Ability – High-Level Communication, Organisation

- An essential feature: some tasks are only possible if agents interact.

- In order to cooperate or to coordinate their actions, agents typically use a high-level form of communication based on the idea of speech-acts.

- Agents can be programmed to take part in an agent organisation all within the context of multiagent oriented programming.

- For a generic platform, we require an information exchange language.

- A successful way for setting up such a generic communication was inspired by speech act theory and led to the definition of the Knowledge Query and Manipulation Language KQML.
The Foundation for Intelligent Physical Agents (FIPA)

- **FIPA** is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies.

- FIPA approaches the challenge of achieving compatibility between different agent systems from the application point of view.

- In 2002, FIPA completed a process of standardising a sub-set of 25 specifications ([http://www.fipa.org/repository/standardsspecs.html](http://www.fipa.org/repository/standardsspecs.html)).
  - An example of these standards is the FIPA Agent Communication Language (ACL) which is strongly inspired by KQML. FIPA adds a formal semantic model and elaborates on predefined protocols and additional speech act types.

Programming Multiagent Systems
FIPA ACL

Compliance to the FIPA specifications means that agent systems must provide appropriate messaging services and process ACL messages, but are still free to decide on concrete realizations.

We can conclude that this message is sent from an agent named “MyAgent” to an agent “MonitorAgent”, requesting it to send a message to “MyAgent”, including the value for “number of agents” from its knowledge base as soon as it exceeds 50.
Programming Languages for BDI agents

- JADE
- JASON
- JADEX
- APL
JADE

- JADE is a pure Java-based platform intended to support the creation and execution of multi-agent applications.

- A middle-ware for Multi-Agent System (MAS)
  - target users: agent programmers for MAS
  - agent services
    - life-cycle, white-page, yellow-page, message transport
  - tools to support debugging phase
    - remote monitoring agent, dummy agent, sniffer agent
  - designed to support scalability
    - (from debugging to deployment)
    - from small scale to large scale

- Implements Foundation for Intelligent Physical Agents (FIPA).

- JADE does not explicitly assist in the creation of deliberative capabilities.

- Fully implemented in Java
  - distributed under GNU Lesser General Public License.
JADE Platform

Container
Agent
Agent
Agent

Container
Agent
Agent

Computer A
Computer B
JASON

- JASON implements the **operational semantics** of a variant of **AgentSpeak** (AgentSpeak is an agent-oriented programming language. It is based on logic programming and the BDI architecture)
- Has various extensions aimed at a more **practical programming** language (e.g. definition of the MAS, communication, ...)
- Highly customised to simplify **extension and experimentation**
JASON: Main Language Constructs and Runtime Structure

- **Beliefs**: represent the *information available* to an agent (e.g. about the environment or other agents)
- **Goals**: represent *states of affairs* the agent wants to bring about
- **Plans**: are recipes for actions, representing the agent’s know-how
- **Events**: happen as *consequence to changes* in the agent’s beliefs or goals
- **Intentions**: plans instantiated to achieve some goal
JASON – Reasoning Cycle
JADEX

- JADEX is a Java-based, modular, and standards compliant, agent platform that allows the development of goal-oriented agents following the BDI model.

- It allows for programming intelligent software agents in XML and Java and can be deployed on different kinds of middleware such as JADE.

- http://jadex-agents.informatik.uni-hamburg.de/
The Abstract Architecture of JADEX
2 APL

- 2APL provides programming constructs both (1) to specify a multiagent system in terms of a set of individual agents and a set of environments, as well as (2) to implement cognitive agents based on the BDI architecture.

- 2APL is a modular programming language allowing the encapsulation of cognitive components in modules. Its graphical interface, through which a user can load, execute, and debug 2APL multi-agent programs using different execution modes and several debugging/observation tools.

A screenshot of the 2APL platform
Sources of this Lecture

