Towards developing multi-agent systems in Ada

Javi Palanca
<jpalanca@dsic.upv.es>
GTI-IA
Summary

- Agents and multi-agent systems
- SPADE
- ADA application interface for SPADE
- Conclusions
- Future Work
Where do agents come from?

- Software Objects
- Artificial Intelligence
- Distributed Systems
- Psychology, society
- Intentional systems
Where do agents come from?

- Software Objects
  - Inter-Object Messages
  - Inheritance
  - Classes
  - Independence
- Artificial Intelligence
  - Knowledge
  - Reasoning
  - Learning
  - Focus: sensors+deliberation+actuation
- Distributed Systems
  - Distributed data
  - Distributed processes
  - Networks
  - Interoperability
- Psychology, society
  - Cognitivism
  - Behaviorism
- Intentional systems
  - Autonomous
What is an Agent?

- Permanent process
- Independent
- Autonomous
- Intelligent
- Flexible
- Reactive, Proactive and Social
Multi-Agent Systems

- Interaction between some similar or heterogeneous agents
- No global control
- Decentralized data
- Individual points-of-view
Agent Applications

- E-Commerce
- Traffic control
- Intelligent manufacture
- Information Agents
- Co-operation networks
- Software Engineering

- Huge and distributed problems
- Open and dynamic environments
- Flexible, inter-operable, efficient, robust, trust...
FIPA

- **Foundation for Intelligent Physical Agents**
- IEEE Computer Society standards organization
- Promotes agent-based technology and the interoperability of its standards with other technologies

www.fipa.org
SPADE

Smart Python multi-Agent Development Environment

- Developed using Python
- Covers the FIPA standard
- Allows different OS and platforms
- Based on the JABBER protocol

- Allows different programming languages using the Jabber protocol
SPADE and Jabber

• SPADE is based on the JABBER protocol

• Jabber is an Instant Messaging protocol to communicate people

• Jabber enables to exchange messages, presence and other structured information in close to real-time

• Jabber is a set of XML protocols and technologies

• SPADE uses Jabber to communicate agents!
SPADE: Platform Model

- Based on the Jabber server (XML Router)
- Links all the platform components (ACC, agents, AMS...) one with each other.

- Every SPADE component is an agent
- Agents use the ACC (Agent Communication Channel) to route messages inside the platform
Jabber gives SPADE some extra features:

- Presence Notification
- Multi-user Conference
The Agent model is composed by:

- A connection mechanism to the platform (a TCP/IP connection to the Jabber server)
- A message dispatcher
- A set of different behaviors.

SPADE agents do reach their goals by running deliberative and reactive behaviors.
ADA API

• The interface that connects agents with SPADE is **XML**

• This allows to develop agents in any programming language (that could work with sockets and XML)

• SPADE provides *ONLY* a python API to develop agents. (and its ok)

• And there is no other supported programming language...
• The interface that connects agents with SPADE is **XML**

• This allows to develop agents in any programming language (that could work with sockets and XML)

• SPADE provides *ONLY* a python API to develop agents. (and its ok)

• And there is no other supported programming language... except for **ADA**
• Our aim is to develop agents using ADA which will be able to connect to a python agent platform (SPADE)
• Package structure
package Spade.Aids is

  type Aid is private;

  function Get_Name ( From: Aid ) return Aid_Name;
  function Get_Addresses ( From: Aid ) return List_Addresses;
  function Get_Resolvers ( From: Aid ) return List_Resolvers;

  procedure Set_Name ( To: in out Aid; Name: in Aid_Name );
  procedure Add_Address ( To: in out Aid; Address: in Address );
  procedure AddResolver ( To: in out Aid; Resolver: in Resolver );

  ...

end Spade.Aids;
• Types Hierarchy
package Spade.Agents is

  type Basic_Agent is new Message Receiver with private;

  function Get_Aid (From: Basic_Agent Class) return Aid;

  procedure Start (What: in out Basic_Agent'Class);

  procedure Take_Down (What: in out Basic_Agent'Class);

  procedure Setup (What: in out Basic_Agent'Class);

  procedure Kill (What: in out Basic_Agent'Class);

  procedure Add_Behavior (To: in out Basic_Agent'Class;
                          Behav: in Behavior'Class;
                          Template: in Acl_Template);

  function Search_Agent (From: Basic_Agent'Class;
                        Template: Ams_Agent_Description)
                          return List_Ams_Agent_Description;

  procedure Register_Service(From: in Basic_Agent'Class;
                             Service: in Df_Agent_Description);

  procedure Send_Message (From: in out Basic_Agent'Class;
                           Env: in Envelope;
                           Message: in Acl_Message);

  type Agent is new Basic_Agent with private;

  type Platform_Agent is new Basic_Agent with private;

  end Spade.Agents;
Example

An_Agent: Agent;
Behavior_One: Periodic_Behavior;
Behavior_Two: One_Shoot_Behavior;
A_Template: Acl_Template;

Set_Default_Behavior (To => An_Agent, Behav => Behavior_One);

Add_Template (To => Behavior_Two, Template => A_Template);
Add_Behavior (To => An_Agent, Behav => Behavior_Two);
Start (What => An_Agent);
Conclusions and Future Work

• A middleware that allows the development of intelligent agents using Ada has been developed.

• This middleware focuses on creating Ada agents that are compatible with the SPADE agent platform.

• It allows bringing the advantages of Ada to the agent realm and vice-versa.

• As Future Work we will test both implementations for performance and scalability.
Towards developing multi-agent systems in Ada

Javi Palanca
<jpalanca@dsic.upv.es>
GTI-IA