Intelligent Agent Framework

Chapter 7
Dr Ahmed Rafea
Chapter Outline

- Requirements
- Design Goals
- Functional Specifications
- Intelligent Agent Architecture
- The CIAgent Framework
- CIAgent Life Cycle
- Boolean RuleBase Enhancement
Requirements

• The Intelligent Agent Framework
  – has the ability to add intelligence
  – must be applicable to solve real world problems
  – provides stimulating learning experience
  – has a flexible and simple architecture
  – Has a decent user interface
Design Approaches

- View IA as adding value to an application. In this case, the approach is application-centric and less complex. By providing IA functions as an OO framework, we can easily add intelligent behavior to any Java application.

- The second approach is to view IA’s as a freestanding community of agents that interact with each other and other applications. This approach is more agent-centric. There must be an agent manager that enables applications to communicate and in this case it may be necessary to change the code of each application to be “agent-aware”
Design Goals

• The agent framework
  – must be easily understood
  – must illustrate how IA’s can be used to enhance applications and easily added
  – Will not be able to handle all unexpected input data and error conditions
  – must support inter-agent communication
  – must be flexible to support new applications
Functional Specifications

- It must be easy to add an IA to an existing Java application
- A graphical construction tool must be available to compose agents out of other Java components and other agents
- The agents must support a relatively sophisticated event-processing capability
- We must be able to add domain knowledge to our agent
- The agent must be able to learn
- Multi-agent applications must be supported using KQML-like message protocol
- The agent must be persistent
Intelligent Agent Architecture-1

• It must be easy to add an IA to an existing Java application
  – The application instantiate and configure the agent and then call the agent’s method. The application is always and we can hardly call this an intelligent agent.
  – The application instantiate and configure the agent and then start it up as in a separate thread. This gives the agent some autonomy but still running in the process space of the application
  – The IA run in a separate thread but use event rather than direct method invocation to communicate with the application. Because this gives both autonomy and flexibility, this is the design we will pursue
Intelligent Agent Architecture-2

• A graphical construction tool must be available to compose agents out of other Java components and other agents
  – There are graphical development tools such as Borland’s JBuilder, WebGain’s Visual Café,… for Java that allow you to construct applications using “construction from parts metaphor”. Java also provides the java.beans package that allow software parts to be treated as parts
The agents must support a relatively sophisticated event-processing capability

- The JDK 1.1 release featured a powerful even-processing model called the Delegation Event Model. The agent can be an EventListener and whenever the agent is notified that an event occurred, it can process the event. We can describe our own event type by subclassing the EventObject class.
Intelligent Agent Architecture-4

• We must be able to add domain knowledge to our agent
  – The BooleanRuleBase, Rule, and RuleVariable classes can be used in our agent to provide forward and backward reasoning. We will have to extend the functionality to support sensors and effectors.

• The agent must be able to learn
  – We can use the DecisionTree, BackPropNet, and KMapNet classes to provide these functions to our agents
Intelligent Agent Architecture-5

• Multi-agent applications must be supported using KQML-like message protocol
  – We can use the Java event model to provide the communication mechanism between agents and the facilitator and define our own event objects to hold the message content.

• The agent must be persistent
  – The java.io package in the Java run time supports serialization of Java objects. This can be used to make saving and loading the objects very easy to do
CIAgent Framework

- CIAgent is the base class that defines a common programming interface and behavior for all the agents in our framework.
- CIAgent uses several helper classes including CIAgentState, CIAgentTimer, CIAgentListener and CIAgentEventQueue:
  - CIAgentState contains a single data member, the `int state`.
  - CIAgentTimer class provides the autonomous behavior and the ability to process events in asynchronous manner.
  - CIAgentListener interface is implemented by CIAgent class to communicate with other CIAgents.
  - CIAgentEventQueue class implements a queue of CIAgentEvent class which is derived from Java EventObject class.
CIAgent Life Cycle

The methods provided by the CIAgent base class are intended to be used in a specific sequence:

1. Construct the CIAgent object instance. The state is set to UNINITIATED.
2. Set the JavaBean properties.
3. Call the initialize() method. The state is set to INITIATED.
4. Call the StartAgentProcessing() method to start the eventQueue thread, and any asynchronous even and time processing. The state is set to ACTIVE.
5. Use the agent in an application by calling the process() method directly, or by sending action events to be processed asynchronously.
6. Use the suspendedAgentProcessing() and resumeAgentProcessing() methods to temporarily halt the autonomous behavior and the invocation of the processTimerProp() method.
7. Call the stop CIAgentProcessing() method to stop the eventTimer thread. The state is set to UNKNOWN.