Agent Oriented Software Engineering (AOSE)

PROMETHEUS METHODOLOGY
ASSIGNMENT (due October 21 at class)

1. Give an example (from your project) of concept and its description (as per slides 28, 31, 32)
2. Define an agent from your project (as per slide 30)
3. Write the description (or draw a diagram or show an image) of a use case for your project.

For 1, 2, 3 above as well as for any part of the other assignments due October 21) prepare as well a .ppt slide for each to present at class for clarifying the way forward to accomplish your project.

We will negotiate together on October 21 at class the expectations that I have from your projects.
Software Engineering

“The establishment and use of sound engineering principles (methods) in order to obtain economically software that is reliable and works on real machines” (Bauer, F. L. Software Engineering. Information Processing 71., 1972)
Software Engineering

“The computer science discipline concerned with developing large applications.”; process, notation, ...

Typical activities in developing a system:
- Requirements
- Analysis
- Design
- Implementation
- ...


Agent Oriented Software Engineering (AOSE)

- Software Engineering …
  - “We know how to do SE, how apply to agents?”

- … of Agent Oriented systems
  - “We know about agents and AI, how engineer large (agent) systems?”

- Relatively recent development (first AOSE workshop was in 2000)
Why not traditional SE?

- High level design differs for different programming paradigms, different abstractions:
  - Procedural: What does it do?
  - OO: What objects are there? (data+operations)
  - Agent: What **goals** are there? What are the **relationships** between agents?
Why not traditional SE?

- Low level design differs since agent systems face uncertainty and failing actions:
  - Need to have alternative plans - can’t assume things will work
  - Not monolithic single plans!
Software Engineering

- Some principles
  - Modularity
  - Abstraction
- Process of developing software
  - Requirement analysis, design, testing,…
- Software Architecture
Core Platform Functionality (FIPA - mK)

- a collection of services that are closely coupled
- provides an infrastructure where agents are deployed
- a FIPA-compliant AP consists of three agents: AMS, ACC, DF
FIPA-OS as Microkernel

- Application Agents
- A.C.L.
- Agent Configuration
- Agent Shell
- Task Management; Conversation Management
- Directory Facilitator
- Agent Management
- Message Transport Services; Internal Message Transport Protocols
- Agent Communication Channel
Supply Chain Scenario
- **System Architecture**
  - Agent Definitions
  - Knowledge Modeling
  - Agent Class Structure

Agency Design

- JADE Service Agents
- Agent Template (JADE)
- Platform Agents
- Application Specific Agents
- Protocol and Communication Layer
- Transport Layer
- Network
THE COOPERATION COMMUNICATION LAYER

- URL: [protocol://] [id@] host : port [/path]

- It is a place where agents execute (often a single computer)
- It acts as the representative of an area; coordinates between agents and yellow pages; and may activate a dormant agent on request
- It manages Cooperation Domains
- It allows lookup of agents by service
- It usually registers itself with the LAC; can participate in cooperation domains; and can offer or request services
- Virtual place where agent interactions take place

An Architecture for Collaborative MAS
Agents

Definition

“An agent is a system situated within and part of environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.”

Agent properties

- Autonomy—has control over its behavior
- Re-activity—continuously observes and react to changes in its environment in timely fashion
- Pro-activity—goal oriented
- Sociality—communicate in a high-level way
Agents

- Contributors to agent research
  - Artificial intelligence
  - Object systems
  - Human-computer interface design

- Multi-agent systems
  - Agent communication languages
  - KQML & FIPA
The Problem

- Agents are a powerful technology with demonstrated potential
- But …
- Agent systems still scarce - why?

⇒ Agents hard to develop
AOSE: Milestones

- **Agent-based computing** is a synthesis of both Artificial Intelligence (AI) and Computer Science.

- An **agent** is an *encapsulated computer system* that is situated in some *environment* and that is capable of *flexible, autonomous action* in that environment in order to meet its *design objectives*.

- Agents are being advocated as a next generation model for engineering *open, complex, distributed* systems.
  - **Open**: components can join or leave the dynamic operating environment and the operating conditions change in unpredictable ways.
  - **Complex**: the software has a large number of *components* that interact following complex *interaction protocols*; every agent has a *partial view* of the environment and there is **no centralized control**.
AOSE: Challenges

- In real-world applications the environment is open, complex and dynamic. As a consequence,
  - **Interaction** among components cannot be completely foreseen at compile-time
  - The system's inherent **organizational structure** must be explicitly represented
- We need the right abstractions, methodologies and instruments to correctly engineer applications of this kind
Agent-Oriented Software Engineering [Jen00]

- The case for agent orientation to software engineering
  - **Agent-oriented decomposition** is an effective way of partitioning a problem space
  - **Agent ‘mindset’** (agent, interactions, and organizational relationships) are a natural means for modeling complex systems
Agent-Oriented Software Engineering

- Problems of agent-based approaches to software engineering
  - Unpredictable patterns and outcomes of the interactions
  - Difficult (or impossible) to predict the behavior of the overall system based on its constituent components
Agent Modeling Tools

- UML – unsuitable for agent modeling
- Two major extensions to UML
  - AUML – extends UML specifically it extends UML interaction diagrams to support agent protocols
  - AML – extends UML and uses concepts from AUML, OWL, MESSAGE, FIPA-S...
Methodologies

- Methodologies based on agent theory
- Extensions of object-oriented methodologies
- Methodologies based on knowledge engineering
- Hybrid methodologies
AOSE

To make developing intelligent agents easier (without losing the power of the BDI model).

“Easy things should be easy, hard things should be possible”

Target audience: professional developers or senior undergrads

BDI = belief, desire, intention

Tools

Concepts

Notation

Methodology

Semantics
The Prometheus Methodology  
(Padgham & Winikoff)

- Supports the development of *intelligent* agents
- “start-to-end” support
- Detailed process and products
- Evolved out of practical industrial and pedagogic experience
- Has been used
- Hierarchical structuring: scales to large designs
- Cross checking

Prometheus was the wisest Titan. His name means “forethought” and he was able to foretell the future. Prometheus is known as the protector and benefactor of man. He gave mankind a number of gifts including fire.  
(http://www.greekmythology.com/)
Methodology

- Concepts (what are we talking about? E.g. object, class, inheritance for OO)
- Models (and a notation to express them)
  - Static - system structure
  - Dynamic - system behaviour
- Process (for generating/deriving models)

- Other: tools, software engineering (coupling, cohesion, abstraction …)
AOSE regarded as a software methodology for the real-world

- **Agent**: autonomous, situated in an environment
- **Real world**: dynamic, uncertain, non-deterministic … things will go wrong
- **An intelligent agent blends** pro-active & reactive behaviour:
  - Stimulus response rules
    - reactive only
    - proactive only
    - Fully scripted behaviour

- Also flexible, robust, rational, social, …
Agent Concepts

- Situated so actions, percepts, time

- Fire engine:
  - See nearby fires & road conditions, hear messages from other agents, hear civilian calls for help.
  - Move, squirt, tell (broadcast), say, plan route (internal)
Agent Oriented Concepts

Agent Structure

Software Agents

Multi Agent Systems

- Embedded features: mobility, conversation, learning, scheduling
- Associated with the type of agent
- User interface (GUI)
- CORBA, Voyager, RMI, etc.

AGENT (Configuration)
Application specific code (Task)
Data (application specific)
Interface specific code
Communication interface

Messages
Agency Design

- System Architecture
- Agent Definitions
- Knowledge Modeling
- Agent Class Structure

The agents are defined and their responsibilities are presented.

Example:

<table>
<thead>
<tr>
<th>Role Schema: Manufacturer Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Manufacturers represent providers of components to the Assembly plant consumer of resources (parts) provided by suppliers</td>
</tr>
<tr>
<td><strong>Protocols:</strong> FIPA REQUEST and FIPA ITERATED CONTRACT NET</td>
</tr>
<tr>
<td><strong>Responsibilities:</strong> To respond to requests to supply the required components, to initiate request for needed parts, to transfer the required components, to receive and consume resources, to be aware of suppliers of parts and their capabilities, to participate in negotiations about the terms of acquiring and/or supplying a specified resource</td>
</tr>
</tbody>
</table>
Agent Concepts

- **Reactive so events**
  (significant occurrence)
  - New fire, fire extinguished, fire urgent, help requested

- **Proactive so goals**
  - Put out fire, discover fire, assist, coordinate
Agent Concepts

- Implementation uses **plans** and **beliefs**
  - Cache for means, and world information respectively
  - *Beliefs:* Map (incl. fires, buildings), fire assignment and priority
  - *Plans:* Put out fire, roam, …
SYSTEM SPECIFICATION:

- In this stage "percepts" and "actions" that characterize the agent's interaction with the environment are defined.

- **Functional descriptors** that contain a **name**, **description**, **actions**, **percepts**, **data used**, and **produced** and a description of interactions are defined.

- **Use cases**: contain an **identification number**, a natural language **overview**, an optional field **context**, the **scenario**, a summary of all the **information used**, and a **list of small variants**.
AOSE: Prometheus (3)

ARCHITECTURAL DESIGN:

- Identification of which agents should belong to the MAS
- Identification of groups of agents which share the same functionalities
- Identification of the agent acquaintance diagram which defines the links among interacting agents
AOSE: Prometheus (4)

- definition of the **agent descriptors**, characterized by name, description, cardinality, functionalities, reads data, writes data, interacts with
- definition of the **events**, messages and shared data objects

- identification of the **system overview diagram** which ties together agents, events and shared data objects
- definition of the **interaction diagrams** and **interaction protocols** using AUML

**SYSTEM OVERVIEW DIAGRAM**

S.A. = shop assistant
W.M. = warehouse manager
C.R. = customer relations
Ca = cashier
AOSE: Prometheus (5)

● DETAILED DESIGN:
   ● Focuses on developing the internal structure of each agent and how it will achieve its task within the system.
   ● The developer must define capabilities, internal events, plans and detailed data structures.
   ● **Capability descriptor:** contains inputs and produced events, a description of functionality, a name, interactions with other capabilities, inclusions and a reference to read and write data.
   ● **Agent overview diagram:** shows capabilities within an agent.
   ● **Capability overview diagram:** takes a single capability and describes its features.
   ● The final design artifacts are the individual plan, even and data descriptors.
   ● **The Plan descriptor** provides an identifier, the triggering event type, the plan steps, a context and a list of data read and written.
Agency Design

- System Architecture
- Agent Definitions
- Knowledge Modeling
  - Agent Class Structure

Customer Agent

<table>
<thead>
<tr>
<th>Interface: Customer_GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content: ContentManager</td>
</tr>
<tr>
<td>Content_Language: Codec</td>
</tr>
<tr>
<td>Scm_Ontology: Ontology</td>
</tr>
<tr>
<td>setup(addBehaviour())</td>
</tr>
</tbody>
</table>

Request_Behavior

| Concepts: Concept |
| Predicates: Predicate |
| Actions: AgentAction |
| action()           |

JADE Agent

- ContentManager
- Codec
- Ontology
- Concept
- Predicate
- AgentAction
Static Structure

S.A. = shop assistant
W.M. = warehouse manager
C.R. = customer relations
Ca = cashier
Dynamic behavior (Agent-UML)
Tool support for Methodology