Agent-Oriented Software Engineering

Lin Zuoquan

Information Science Department

Peking University

lz@is.pku.edu.cn

http://www.is.pku.edu.cn/~lz/teaching/stm/saswws.html

Outline

- Introduction
- AOSE
- Agent-oriented analysis and design
- Formal method for AOSE
- Pitfalls of agent-based solutions

Introduction

Methodologies

- Object-oriented Programming
- Agent-oriented Programming

Class Discussion: AO vs. OO

- Similarities
- Differences

AOP vs OOP

	OOP	AOP
Structural Elements		
	abstract class	generic role
	class	domain specific role
	class variables	knowledge, belief
	methods	capabilities
Relations		
	collaboration (`uses`)	negotiation
	composition (`has`)	holonic agents
	inheritance (`is`)	role multiplicity
	instantiation	domain-specific role + individual knowledge
	polymorphism	service matchmaking

AOSE

Agent-based approaches to SE

- Techniques for tackling complexity in software
  - Decompositio
  - Abstraction
  - Organization

Agent-oriented decomposition

Decompose the problem in terms of autonomous agents that can engage in flexible, high-level interactions.

- Agents *autonomy reduces control complexity* since the system's control is localised inside each individual problem solving component.
- *Action selection* can be based on the local situation
- Agents make *decisions* about the nature and scope of the interactions **at** *run-time* – this makes the engineering of complex systems easier for two reasons:

  - It is difficult/impossible to know *a priori* all potential interactions, at what time, for what reason - agents have the ability *to initiate and respond to interactions in a flexible manner, to deal with unanticipated requests*

  - Management of control relationships between the software components is significantly reduced - any *coordination* that is required is handled bottom-up *through inter-agent interaction.*

Agent-oriented abstraction

Clear and strong degree of correspondence between the notions of subsystems and agents.

- Subsystem components are represented as agents
- The interactions between the subsystems is viewed in terms of *high-level social interactions*.
- Booch: “at any given level of abstraction, we find meaningful collection of entities that collaborate to achieve some higher level view”

  Agents are *cooperating* to achieve common objectives, *coordinating* their actions or *negotiating* to resolve conflicts

Agent-oriented organizational structures

The agent-oriented approach provides an explicit representation of organisational relationships and structures.

- Represent and manage organizational structures; may vary during problem solving
- The entire subsystem can be viewed as a *primitive component* or as a team or collection of agents recursive agent structures can be defined to lower the complexity.
- Individual or organisational groupings can be developed in relative isolation and then added into the system in an incremental manner.

Agent-oriented analysis and design

Methdologies

Two groups of agent-oriented methodologies

- Extend or adapt OO methodologies to the purpose of AOSE
- Adapt knowledge engineering methodologies

Extending OO Methodologies

- AAII
- GIGA
- AUML

AAII

AAII methodology (Kinny et al., 1996)

- based on the BDI model
- A set of models which, when fully elaborated, define an agent specification

External model

- Presents a system level view: agents and relationships
- Inheritance relationships between agent classes + instances of these classes at run-time
- *Agent model: agent class model + agent instance model*
- Each agent has at least 3 attributes: beliefs, desires, intentions + how they are overridden during inheritance

Internal model - implements the agents

AAII(contd.)

- Identify the relevant **roles** in the application domain and develop an **agent class hierarchy**
- Identify the **responsibilities** associated with each role, the **services** required by and provided by the role, the **goals** associated to each service
- For each role, determine **plans** that may be used to achieve it and the context conditions under which each plan is appropriate
- Determine the **belief structure** of the system - the information required for each plan and goal

GAIA

GAIA methodology (Wooldrige et al., 1999)

- Go from a statement of requirements to a detailed design
- Moves from abstract concepts to increasingly concrete concepts
- Basic idea: think of building agent-based systems as a process of organizational design

Abstract concepts	Concrete concepts
Roles	**Agent types**
Responsibilities	**Services**
Liveness properties	**Acquaintances**
Safety properties	
Permissions	
Activities	
Protocols	

GIGA(contd.)

- **Organization** = a collection of roles that stand in certain relationships to one another and take part in systematic patterns of interaction with other roles
- **Roles** = may be instantiated to agents; not necessarily fixed
- **Responsibilities** = functionality
  - Liveness property = state of affair the agent has to bring about given environment conditions
  - Satfety properties = invariants
- **Permissions** = rights associated to a role; identify the resources available to the role to realize responsibilities
- **Activities** = computations associated with a role that may be carried out by the agent without interacting with other agents
- **Protocols** = the way a role can interact with other roles

AgentUML

AUML methodology (Odell et al.,2000)

Based on previous research of extending UML

Extensions to UML:

- Support for expressing concurrent threads of interaction, and thus agent protocols
- A notion of role that extends that provided in UML and allows an agent to play many roles

From the **AUML** Web site: http://www.auml.org/

- To recommend technology for adoption of common semantics, meta-model, and abstract syntax for agent-based analysis and design methodologies.
- To recommend technology for adoption that enable interoperability across the lifecycle of AUML tools designs/work products
- To leverage existing FIPA and OMG specifications
- Currently, AUML is a goal—not an existing modeling language

Adapting KE Methodologies

Several solutions have been proposed for MAS modelling extending CommonKADS (European standard for knowledge modelling)

The MAS-CommonKADS methodology

- **Agent Model**: describes the main characteristics of the agents, including reasoning capabilities, skills (sensors/effectors), services, goals, etc.
- **Task Model**: describes the tasks carried out by agents, and task decomposition, using textual templates and diagrams.
- **Expertise Model**: describes the knowledge needed by the agents to carry out the tasks.)
- **Coordination Model**: describes the conversations between agents, that is, their interactions, protocols and required capabilities.
- **Organization Model**: describes the organisation in which the MAS is going to be introduced and the organisation of the agent society.
- **Communication Model**: details the human-software agent interactions, and the human factors for developing user interfaces.
- **Design model**: collects the previous models and is subdivided into three submodels:
  - **application design**: composition or decomposition of the agents and selection of the most suitable agent architecture for each agent;
  - **architecture design**: designing of the relevant aspects of the agent network: required network, low level communication facilities
  - **platform design**: selection of the agent development platform for each agent architecture

Formal Method for AOSE

Formal Methods

- Specification
- Refinement
- Verification

Pitfalls of agent-based solutions

Pitfalls

- Oversell agent solutions
- Get religious or dogmatic about agents
- You don't know why you want agents
- You believe that agents are a silver bullet
- You have too many agents or too few agents
- Your agents use too much AI
- You decide you want your own agent architecture
- You spend all your time implementing infrastructure

Readings

- M. Wooldridge and P. Ciancarini, Agent-Oriented Software Engineering: The State of the Art, In: P. Ciancarini and M. Wooldridge (eds.), Agent-Oriented Software Engineering, Springer, LNAI 1957, 2001
- N. Jennings, On Agent-based Software Engineering, Artificial Intelligence, 117(2), 277-296, 2000
- M. Wood and S. DeLoach, An Overview of the Multiagent Systems Engineering Methodology, The First International Workshop on Agent-Oriented Software Engineering (AOSE-2000), 2000 (optional)