4. Multiagent Systems Design
Part 1:
Agent-Oriented Software Engineering Methodologies. The GAIA methodology.

Javier Vázquez-Salceda
SMA-UPC

Introduction (to Agent Methodologies)

- Software Engineering
- Agent-Oriented Software Engineering
- Software Methodologies
- Agent-Oriented Methodologies
Software Engineering
Status of Software Engineering in the New Millennium

- Current tendency to make software functionalities and business cases coincide - stimulated by the Internet era and reinforced by the DOTCOM economy
  - Leads to linking software construction and business dynamics more closely than ever
- In industry there is a need for swiftly-developed, complex software projects that are both research-like and mission-critical
  - Software development must no longer be thought of as oriented toward a product BUT it is an ongoing process which continually delivers value (continuous evolution)
- Software crisis
  - Hardware costs were decreasing while software costs were increasing.

Software Engineering
Abstractions

- Software deals with “abstract” entities, having a real-world counterpart
  - Numbers, dates, names, persons, documents, ...
- In what term shall we model them in software?
  - Data, functions, objects, agents, ...
  - I.e., what are the abstractions that we have to use to model software?
- May depend on available technologies
"Objects are far from perfect, but are the only game in town"  
-- Grady Booch

- Maybe the agent community would like to reply...

- A lot of research work has been done to define what an agent and a MAS are, how they compare to object-oriented concepts and which their distinguishing features are

- AO paradigm subsumes the concepts supported by the previous programming paradigms, and in particular by the object-oriented programming
  - Tries to raise the abstraction level
  - Software agents are undoubtedly more than a promising approach to complex software development
Agent-Oriented Software Engineering

Abstractions

- The development of a multiagent system should fruitfully exploit higher level abstractions
  - Agents, autonomous entities, independent loci of control, situated in an environment, interacting with each others
  - Environment, the world of entities and resources agents perceive, control, consume or exploit.
  - Roles and interactions: identify functionalities, activities, responsibilities and interaction patterns.
  - Organizational Rules, which can be constraints on roles and interactions, or relations between roles, between protocols, and between roles and protocols (open/close systems)
  - Organizational Structures and Patterns: Identify the topology of interaction patterns and the control regime of activities (efficiency, robustness, degree of openness)

Agent-Oriented Software Engineering

Characterisation of a MAS
Agent-Oriented Software Engineering
Agent-Oriented Computing

● There has been some debate
  ■ On what an agent is, and what could be appropriately called an agent

● Two main viewpoints in agent development
  ■ The (strong) artificial intelligence viewpoint
    ▪ A multi-agent system is a society of individual (AI software agents) that interact by exchanging knowledge and by negotiating with each other to achieve either their own interest or some global goal
  ■ The (weak) software engineering viewpoint
    ▪ A multi-agent system is a software systems made up of multiple independent and encapsulated loci of control (i.e., the agents) interacting with each other in the context of a specific application

Agent-Oriented Software Engineering
Software Engineering Viewpoint on AO Computing

● The Second is useful because
  ■ It focuses on the characteristics of agents that have impact on software development
    ▪ Concurrency, interaction, multiple loci of control
    ▪ Intelligence can be seen as a peculiar form of control independence; conversations as a peculiar form of interaction
  ■ It is more general:
    ▪ Several software systems, even if never conceived as agents-based one, can be indeed characterized in terms of weak multi-agent systems
Agent-Oriented Software Engineering

Key Characteristics of Agents

- Basic characteristics (SE Viewpoint)
  - Autonomy & Proactivity (delegation of responsibility)
  - Situatedness
  - Interactivity (communication, collaborative or competitive interactions)

- Additional characteristics (SE Viewpoint)
  - Openness (need of standards; need of proper infrastructures supporting the interoperations)
  - Learning & Adaptative Capabilities (Improving the effectiveness of its actions; adapting their behaviour to changing situations)

Agent-Oriented Software Engineering

There is more to Agent-Oriented Software Engineering

- AOSE is not only for “agent systems.”
  - Most of today’s software systems have characteristics that are very similar to those of agent and multiagent systems
- AOSE is suitable for a wide class of scenarios and applications

Agent-based computing, and the abstractions it uses, represent a new and general-purpose software engineering paradigm
Software Methodologies

- A methodology for software development...
  - is intended to discipline the development
  - defines the abstractions to use to model software
    - Data-oriented, flow-oriented, object-oriented, ...
    - Defines the mindset of the methodology
  - disciplines the software process
    - What to produce and when
    - Which artefacts to produce
- Def: a software methodology is the set of guidelines for covering the whole lifecycle of system development both technically and managerially
  - full lifecycle process
  - comprehensive set of concepts and models
  - full set of techniques (rules, guidelines, heuristics)
  - fully delineated set of deliverables
  - modelling language
  - set of metrics
  - quality assurance
  - coding (and other) standards
  - reuse advice
  - guidelines for project management

Software Methodologies

The Classical “Cascade” Process

- The phases of software development:
  - Independent of programming paradigm;
  - Methodologies are typically organized around this classical process
    - Inputs, outputs, internal activities of “phases”
Software Methodologies

Tools

- **Notation tools**
  - To represent the outcome of the software development phases
    - Diagrams, equations, figures, ...

- **Formal models**
  - To prove properties of software prior to development
    - Lambda calculus, Petri-nets, Z, ....

- **CASE tools**
  - To facilitate activities: rapid prototyping, code generators, ...

Agent-Oriented Methodologies

- There is need for **agent-oriented methodologies**
  - Centred around specific **agent-oriented abstractions**
  - The adoption of OO methodologies would produce mismatches
    - Classes, objects, client-servers: little to do with agents

- Each methodology may introduce further abstractions
  - Around which to model software and to organize the software process
    - E.g., roles, organizations, responsibilities, belief, desire and intentions, ...
  - Not directly translating into concrete entities of the software system
    - E.g. the concept of role is an aspect of an agent, not an agent
Agent-Oriented Methodologies
Agent-Based Analysis

- **Analysis** aims to understand, at least
  - What are the main actors interacting with the system
  - How the system interacts with these actors
  - What the system is supposed to do
- The system is a closed entity and we do not look into it to avoid anticipating design issues and decisions
- **In AO, we associate agents with the entities of the scenarios we are analyzing**
- Then, we associate accordingly
  - **Roles**, responsibilities and capabilities
  - **Interaction patterns** between agents
- This provides a neutral view of the problem.
- Methodologies such as Tropos and GAIA, do not use the word agent to identify analysis-phase entities

---

Agent-Oriented Methodologies
Agent-Based Design

- **Design** aims to engineer, at least
  - What are the main components interacting within the system
  - What are the responsibilities and the capabilities of each component in the system
  - How the components interact to implement the system, i.e., the architecture of the system
- **In AO, we associate agents with the components we use to build the system**
- Then, we associate accordingly
  - **Roles**, responsibilities and capabilities
  - **Interaction patterns** between agents
- Differently from analysis: we need to choose on which agents to use and how they interact
Agent-Oriented Methodologies

Several methodologies and approaches for designing MASs exist in literature. In general they tackle different aspects of the MAS and in some cases they are quite complementary:

- **GAIA**
  - Encourages a developer to think of building agent-based systems as a process of organisational design.
- **TROPOS**
  - It is founded on the concepts of goal-based requirements adopted from the i* and GRL (Goal-oriented Requirements Language). Its distinguishing feature is the emphasis on requirements analysis.
- **Prometeus**
  - It focuses mainly on the internal agent architecture; it is basically a methodology for designing BDI agent systems.
- **ADELFE**
  - It is a methodology for the development of adaptive multiagent systems.
- **MESSAGE**
  - It covers most of the fundamental aspects of the MAS development, focusing mainly on analysis and high-level design. The main objective was to combine the best features of the pre-existing approaches, but … the result was a complex and farraginous methodology.
- **PASSI**
  - It is a step-by-step requirement-to-code methodology. Integrates design models and concepts from both object oriented software engineering and artificial intelligence approaches.

The GAIA Methodology

- GAIA v.1
- GAIA v.2
GAIA Methodology

- Gaia is appropriate for the development of systems with the following main characteristics:
  - Gaia is not intended for systems that admit the possibility of true conflict.
  - Gaia makes no assumptions about the delivery platform;
  - The organisation structure of the system is static, in that inter-agent relationships do not change at run-time.
  - The abilities of agents and the services they provide are static, in that they do not change at run-time.
  - The overall system contains a comparatively small number of different agent types (less than 100).

GAIA Methodology
Case Study

**Auction agent**

1. The *configurator*: a GUI component, enables the user to control and monitor the agent's activity
2. The *parser*: translates retrieved information into an internal structure
3. The *bidder*: submits bids according to buying strategy. Implements two stages, bid and confirmation
4. The *manager*: controls the agent's activity, monitors the auction site, activates the parser, determines the next bid, activates the bidder and terminates the agent's purchasing activity
GAIA Methodology
Disciplines

- **Requirements capture phase** are considered *independent of the paradigm used* for analysis and design
  - For this reason *Gaia does not deal with the requirements* capture phase
- The **analysis phase** consists of the following models:
  - *Role definition* (permissions, responsibilities and protocols)
  - *Interaction model* (used for protocol description)
- The **design phase** consists of the following models:
  - *Agent model*
  - *Service model* (*input, output, pre and post condition*)
  - *Acquaintance model*
GAIA Methodology
Disciplines (Process Description)

GAIA Methodology
Work Products from all phases
GAIA Methodology
Analysis Phase

4. Multiagent Systems Design

- System Analyst
  - Identify the Roles ()
  - Identify the associated Protocols ()

- Interactions Model
- Roles Model

Prototypical
Roles Model

- Template for role schemata

- Protocols, state the interactions of the role with other roles. In addition state the internal activities of the role

- Permissions, state what resources may be used to carry out the role and what resource constraints the role’s executor is subject to

- Responsibilities, determine the functionality of the role. This functionality is expressed in terms of safety and liveness properties

jvazquez@lsi.upc.edu 27

jvazquez@lsi.upc.edu 28
GAIA Methodology
Analysis Phase: Role Model

Role Schema: Manager (MA)

Description:
Controls the auction agent activities

Protocol and Activities:
CheckAuctionSite, ActivateParser, CheckForBid, Bid

Permission:
- reads supplied ItemNumber // the item number in the auction site
- AuctionDetails // the auction information

Responsibilities:
Liveness:
Manager = (CheckAuctionSite, ActivateParser, CheckForBid)[Bid]
Safety:
true

- The Manager role scheme

jvazquez@lsi.upc.edu

GAIA Methodology
Analysis Phase: Interaction Model

AuctionAgent
CheckAuctionSite
Manager AuctionSite
supplied ItemNumber
AuctionDetails
Supplied ItemNumber
AOM
Protocol name
Description
input
Sender
Receiver
output

- The Interaction Model of the CheckAuctionSite protocol

jvazquez@lsi.upc.edu
GAIA Methodology
Analysis Phase (Process Description)

- Requirements Statement (From Req. Cap. phase)
- Identify the roles in the system
- Identify and document the associated protocols
- Interactions Model
- Elaborate the roles model

GAIA Methodology
Design Phase

Agent Designer
- Aggregate Roles into an Agent Type()
- Document the instances of each Agent Type()
- Identify Services ()
- Identify Acquaintance Relationship ()
GAIA Methodology
Design Phase: Models

- The Agent Model

<table>
<thead>
<tr>
<th>Service</th>
<th>Input</th>
<th>Output</th>
<th>Pre-condition</th>
<th>Post-condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get auction details</td>
<td>ItemNumber</td>
<td>AuctionDetails</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>Validate user</td>
<td>User</td>
<td>Exists</td>
<td>true</td>
<td>(exists=true) ∨ (exists=false)</td>
</tr>
<tr>
<td>Bid</td>
<td>User, ItemNumber, Price</td>
<td>Success</td>
<td>user exists</td>
<td>(success=true) ∨ (success=false)</td>
</tr>
</tbody>
</table>

- The Service Model

- The Acquaintance Model
GAIA v.2

- First version of GAIA
  - Designed to handle *small-scale, closed* agent-based systems
  - Modelled agents, roles, interactions
  - Missed in modelling explicitly the social aspects of a MAS
- GAIA v.2: Official extension of GAIA
  - Thought for *open* agent systems
  - Significantly extends the range of applications to which Gaia can be applied
  - Focused on the *social organization* of the system
- Two further abstractions
  - Organizational rules
  - Organizational structures

GAIA v.2
Work Products from all phases

**Analysis**
- Environmental Model
- Prototypical Roles Model
- Prototypical Interactions Model
- Organizational Rules
- Organizational Structure

**Architectural Design**
- Sub-Organiizations Description
- Organizational Services Model
- Roles Model
- Environmental Model
- Prototypical Interactions Model
- Organizational Rules
- Organizational Structure

**Detailed Design**
- Agent Model
- Services Model
GAIA v.2 Analysis
Sub-Organizations Description

- Identify sub-organizations based on
  - the requirements or their presence in the application structure
  - subgoals that need to be achieved
  - limited interaction with other parts
  - required skills that are not needed in other parts

GAIA v.2 Analysis
Environmental Model

- Identify resources
  - a list of abstract computational resources, e.g. variables, tuples
  - the nature of the environment can be distributed
  - Relations between resources
  - The dynamics of the environment
GAIA v.2 Analysis
Preliminary Role Model

- Identify roles
  - Identify *Basic skills* (partial roles)
  - Basic skills can be turned to complete roles if all other roles are known
  - The complete set of roles are known when the organization structure is known.
- Basic skills
  - **Permission**: resource access and the amount of access (when mismatch redefine environment or add new roles)
  - **Responsibility**: expected behaviours (liveness and safety properties)

GAIA v.2 Analysis
Preliminary Interaction Model

- Identify interactions
  - Relations and dependencies between roles
  - Interactions are described as abstract protocols
    - **Protocol Name**, e.g. assign task
    - **Initiator**, the role starting the interaction
    - **Partner**, the role to interact with
    - **Input**, information used by initiator
    - **Outputs**, information provided by partner
    - **Description**, the purpose of the protocol and its activities

jvazquez@lsi.upc.edu
GAIA v.2 Analysis
Organizational Rules

- Identify **Organizational Rules**
  - Organizational rules are defined as
    - constraints on roles and protocols,
    - constraint and relations between roles,
    - constraint and relations between protocols,
    - constraint and relations between roles and protocols
  - Organizational rules are considered as *responsibilities* of the organization as a whole

**Requirements Statement**

**Sub-O rganizations**

**Systems Design**

4. Multiagent Systems Design

jalvarez@lsi.upc.edu

GAIA v.2 Architectural design
Organizational Rules

- Two kinds of Organizational rules
  - **Liveness rules**, e.g. a role can be played by an entity after it has played another role
  - **Safety rules**, e.g. two roles can never be played by the same entity
- Due to their similar nature, organizational rules can be expressed by making use of the *same formalism* adopted for specifying liveness and safety rules for roles
  - Eg:
    - In the manufacturing pipeline, the correct management of the pipeline requires each of the stage roles to be played only once. This can be expressed by the **safety rule**:
      \[
      R = (\text{STAGE}[1], \text{STAGE}[2], \ldots, \text{STAGE}[N])
      \]
GAIA v.2 Architectural design
Organizational Structure

- In GAIA v.1 the role model may define the organizational structure in an implicit way. The structure of a MAS is more appropriately derived from the explicit choice of an appropriate organizational structure
  - Organizational structures viewed as first-class abstractions

- Manufacturing pipeline: collective of peers organization
- Manufacturing pipeline: hierarchical organization

**Organizational structure**
- Control Regime: Work Partitioning, Work Specialization, Market-based Models

**Decision Parameters for Organizational Structure**
- Computation and coordination complexity
- (influence of) Organizational Rules
- Structure of Real-World Organization
- Simplicity

**Organizational Patterns**
- Catalogue of organizational structures
GAIA v.2 Architectural design
Role Model & Interaction Model

- Complete role models and interaction models
  - Based on decided organizational topology
    - Define all activities in which a role is involved (incl. Liveness and Safety)
    - Define organizational roles (not from analysis phase)
  - Based on decided control regime
    - Complete the definition of the protocols (e.g. which roles are involved)
    - Define organizational protocols (adoption of organizational structure)

GAIA v.2 Detailed design
Agents Model

- Define agents model
  - An agent is a computational entity that can play a set of roles
    - Which agent classes should be defined to play specific roles?
    - How many instances of each agent class have to be instantiated?
  - Trade-off
    - Coherence of agent classes
    - Efficiency of agent classes
    - Similarity to real-world organization
GAIA v.2 Detailed design
Services Model

- Define services model
  - Identify the services associated with each agent class
  - Services are derived from protocols, activities, responsibilities, permissions
  - Properties of services
    - Input/output (derived from protocols)
    - Pre- and post-conditions (safety and organizational rules)

References (I)

References (II)

