Roadmap – Where we go next!

Background

- Agent Orientation
  - Agent Concept(s)
  - Multiagent Concept(s)
  - Illustrations
Agent Concept(s)

“What are the properties an object must possess in order to be an agent?”

No commonly accepted precise definition of “agent”

- applied differently by different people and in different contexts
- often based on intuitive understanding and used as in everyday life
Emerging “standard definition”:

An agent is a (computational) entity that is situated in some environment and that is capable of flexible, autonomous activity – action and interaction – in order to meet its design objectives.

Remarks on this “standard” – key characteristics:

- situatedness
- flexibility (reactivity + proactivity) → problem solving, planning, learning (“intelligence”)
- autonomy (internal state + external behavior)
- objectives (purpose, goal)
Agent Concept(s)

Other characteristics of agency sometimes claimed to be essential:

- rationality
- mobility
- adaptivity
- introspection
- benevolence

Often mental attitudes are attached to agency, e.g.

- belief, knowledge, ... (information)
- intention, plan, commitment, ... (control)
- desire, preference, ... (motivation)
Agent Concept(s)

Agents and Objects

- Both encapsulate identity ("who"), state ("what"), and passive behavior ("how, if invoked").
- Agents additionally encapsulate active behavior ("when", "why", "with whom", "whether at all")
- The agent and object concepts
  - allow for qualitatively different system perspectives
  - are concerned with different levels of abstraction
  - thus are complementary rather than mutually exclusive
- Think of a gradual transition from agents to objects, rather than a sharp borderline (active object concept, constructs such as preconditioning in Eiffel)
Agents and the evolution of programming concepts (Odell, Jennings):

<table>
<thead>
<tr>
<th></th>
<th>MONOLITHIC</th>
<th>MODULAR</th>
<th>OO</th>
<th>AO</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT BEHAVIOR</td>
<td>nonmodular</td>
<td>modular</td>
<td>modular</td>
<td>modular</td>
</tr>
<tr>
<td>UNIT STATE</td>
<td>external</td>
<td>external</td>
<td>internal</td>
<td>internal</td>
</tr>
<tr>
<td>UNIT INVOCATION</td>
<td>external</td>
<td>external</td>
<td>external</td>
<td>internal</td>
</tr>
</tbody>
</table>

- move from machine- to problem-oriented abstractions
- units show increasing localization and encapsulation
- thus: agents can be viewed as a natural next step
- evolution, not revolution
A basic view of the agent concept:
“Definition”:

A multiagent system is a system composed of multiple agents that act and interact to fulfill individual and/or joint design objectives.

The key feature is distribution of activities and processes:

- distributed sensing and acting
- distributed planning, distributed learning, ...
- negotiation, conflict handling, ...
- communication, ontologies, ...
Multiagent Concept(s)

A basic view of the multiagent concept:
Multiagent Concept(s)

Forms of interaction:

- Coordination
- Interaction
- Competition
- Cooperation
- Collaboration
- Communication
Multiagent Concept(s)

- Coordination vs. collaboration (Grosz 1996):
Multiagent Concept(s)

- Cooperation vs. sum of individual activities (Grosz 1996):
Interaction vs. collaboration (Grosz 1996):

- Driving in a convoy: a collaboration.
- Driving in Boston: highly interactive, but not a collaboration.
Automated production:

- Parts: M_11, M_12, M_13
- Production steps: A, B, C
- Machines: M_31, M_32, M_33, M_34
- Products: A, B, C
Docking station:
Traffic fbw regulation:
Transportation:

X = { x1, ..., x5 }

Y = { y1, ..., y10 }

Z = { z1, z2 }
Illustrations

Robotic Orientation:

Background • Agent Orientation
Supply chain management:
Personal software assistants:

Meeting

When?
Agenda?

......
Illustrations

Games – “Blocks World”:

agents

A B C D
Games – “Predator-Prey”:
Further examples of application domains:

- logistics
- telecommunication
- robotics, autonomous vehicles
- interactive games (avatars)
These examples indicate several distinguishing features of multiagent systems:

- environment: diversity, dynamics, predictability, ...
- agents: number, homogeneity, goals, ...
- interaction: frequency, levels, patterns, ...
Roadmap – Where we go next!

► Background
  ■ Agent Orientation
  ■ Modeling
    • General Characterization
    • Modeling Foci
    • Integration of Modeling Foci
    • Techniques for Finding Model-Relevant Facts
General Characterization

Informal definitions of “model”:
- “simplified representation of a complex reality”
- “useful representation of some subject”

Characteristics of a model (Ross 1977):
- purpose (goal)
- range (scope, “universe of discourse”)
- viewpoint (focus, coverage)
- precision (granularity)
The role of modeling in software engineering – consider the simple waterfall model:

- Main cause of failure: poor modeling in requirements engineering phases
Data modeling (object/information modeling) → “standard”

- What: data structures (objects, classes, attributes, relationships)
- How: ERM, UML, Jackson diagrams, organization charts, pictograms, formal (Chomsky) grammars and languages, ...

Functional modeling → “standard”

- What: functions to be performed, flow of data
- How: data flow diagrams, functional decomposition diagrams, structure charts, ...
Dynamic modeling (behavioral modeling) → “standard”

- What: behavior (global + local), transition between states, control and processes
- How: state transition diagrams (in form of Petri nets, finite state machines, statecharts, ...), decision trees and tables, CSP (Communicating Sequential Processes), ...
- Remark: sometimes “process modeling” as a generic term for functional and dynamic modeling
Modeling Foci

- Knowledge modeling
  - elicitation of knowledge
  - expert systems, configuration systems, ...

- Viewpoint modeling
  - viewpoint = external entity interacting with the system being analyzed/developed

- Goal-based modeling
  - motivation behind system development
  - *softgoals* for representing non-functional requirements such as responsibilities, reliability, flexibility, integrity, and adaptability (i.e. goals having no clear-cut definition)
Modeling Foci

► Quality-Function modeling (QF Deployment)
  - transformation of qualitative consumer requirements into quantitative technical requirements

► Prototypes and mock-ups
  - prototype = model capturing certain aspects of the system to be analyzed/developed
  - mock-up = non-implemented prototypes
  - cheap, understandable, help to avoid misunderstandings, basis for initial testing and training
“4-Worlds Modeling” (Jarke 1992)
- subject: the world about which information is
- system: the system itself (different level of detail)
- usage: the use of the system
- development: the developmental environment

“Sub-models” (Bubenko 1995)
- concepts - objectives - activities - actors -
- non/functional requirements - information systems
Integration of Modeling Foci

“Task-oriented Modeling” (Paech)
- software itself
- application context
- user context

“Ontology modeling” (Mylopoulos 1998)
- static ontology (things, attributes, relationships)
- dynamic ontology (states, state transitions, processes)
- intentional ontology (mental states of agents)
- social ontology (social structures and dependencies)
Fact finding = asking questions (Shelly et al. 2001)

- **Who?** – Who performs what procedure? Why? Could other people perform the tasks more efficiently?
- **What?** – What is being done? Why necessary?
- **Where?** – Where are operations being performed? Why? Could they be performed more efficiently elsewhere?
- **When?** – When is an operation performed? Why at this time? Is it the best time?
- **How?** How is a procedure performed? Why in that manner? How to perform better, less expensively?
Techniques for Finding Model-Relevant Facts

Techniques:

- Sampling of existing material (forms, files, ...)
- Interviews
- Questionnaires
- Brainstorming
- Observation
- Discovery prototyping
- Workshops
- Research
Techniques for Finding Model-Relevant Facts

- and always: validation and verification
  - validation: external correctness ("Is it the right model?")
  - verification: internal consistency ("Is the model right?")

Choosing and applying technique(s) is not trivial – e.g., consider and compare interviews and questionnaires

Interviews:
  - usual distinction:
    - structured interviews: specific set of questions
    - unstructured interviews: general goal
Techniques for Finding Model-Relevant Facts

Interviws (Cont’d):

- Steps (according to Shelly 2001):
  1. Determine the people to interview
  2. Establish interview objectives
  3. Prepare for the interview
  4. Conduct the interview
  5. Document the interview
  6. Evaluate the interview
Interviews (Cont’d)

Advantages and disadvantages (Whitten et al. 2001):

+ analyst can motivate interviewee
+ analyst can probe for more feedback
+ adaption and rewording of questions
- very time-consuming, thus costly
- highly dependent on analyst’s relation skills
- may be impractical due to location of interviewees
Questionnaires

- Steps analogous to interviews

- Advantages and disadvantages (Whitten et al. 2001):
  + Can be designed so that they can be answered quickly. Completion and return at convenience.
  + Relatively inexpensive.
  + Maintain anonymity, encourage to tell the truth.
  - Often poor response.
  - Inflexible.
  - Clarification often impossible.
Roadmap – Where we go next!

Background

- Agent Orientation
- Modeling
- Rationales for and Against Agent-Oriented Modeling
  - Why Agent-Oriented Modeling is Appealing
  - Characterizing Potential Applications
  - Pitfalls of Agent-Oriented Modeling
Why Agent-Oriented Modeling is Appealing

- parallelism, robustness, scalability
- distribution of data, control, expertise, resources
- broad range of potential applications, well suited for open domains
- natural next step in programming
- available technology
- offer techniques (identified by Booch) for tackling increasing software complexity: decomposition – abstraction – hierarchy/organisation
- interactivity and intelligence are closely related
An application is particularly suited for agent-oriented modeling if it involves multiple components which

- are not all known a priori
- can not all be assumed to be fully controllable
- must interact on a sophisticated level of communication and coordination to fulfill their individual or joint design objectives.
There is no “silver bullet” in software engineering – don’t ...

- ... oversell agents.
- ... see agents everywhere.
- ... get religious about agents.
- ... confuse buzzwords with concepts.
- ... forget to exploit related technology.
- ... forget you are developing software.
Roadmap – Where we go next!

► Background

■ Agent Orientation
■ Modeling
■ Rationales for and Against Agent-Oriented Modeling
■ Levels of Agent-Oriented Modeling
“within – between – above”

Three levels:

- intra-agent: “What is within a single agent?”
- inter-agent: “What happens between agents?”
- supra-agent “What is above agents and interactions?”

The levels are related.