Distributed Artificial Intelligence
– Agent-Oriented Engineering –

Part 3

Gerhard Weiß

Institut für Informatik
Technische Universität München

Winter 2004/5
Outline

Motivation
Agent Architectures

Coordination
  Overview
  Basic Models and Mechanisms
  Advanced Models and Mechanisms
Outline

Motivation
Agent Architectures

Coordination
Overview
Basic Models and Mechanisms
Advanced Models and Mechanisms
What is Coordination?

▶ “Coordination is managing dependencies between activities.” (Malone & Crowston 1994)

▶ “Coordination is a special case of interaction in which agents are aware how they depend on other agents and attempt to adjust their actions appropriately.” (Malone & Crowston 1991)

▶ “Any decision by an agent that uses information concerning the existence, decisions, or decision-making strategies of other agents is a coordinated decision.” (Stirling 1994)
Characterizing Cooperative Systems

- **Key features** in which cooperative systems differ from one another, and which are suited for characterizing such systems are:
  - **environment**: diversity, dynamics, predictability, ...
  - **cooperating entities**: number, homogeneity, goals, ...
  - **cooperation**: frequency, levels, patterns, ...

- **Space-Time Taxonomy** (Weber 1998):
  - participants are on the same vs different locations
  - participants interact at (nearly) the same vs different time
## Coordination Theory (Malone & Crowston 1990)

- **Components of coordination**

<table>
<thead>
<tr>
<th>Components of coordination</th>
<th>Associated coordination processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goals</td>
<td>Identifying goals</td>
</tr>
<tr>
<td>Activities</td>
<td>Mapping goals to activities (e.g., goal decomposition)</td>
</tr>
<tr>
<td>Actors</td>
<td>Selecting actors, Assigning activities to actors</td>
</tr>
<tr>
<td>Interdependencies</td>
<td>“Managing” interdependencies</td>
</tr>
</tbody>
</table>
Coordination Theory (Cont’d)

Kinds of interdependence

<table>
<thead>
<tr>
<th>Kinds of interdependence</th>
<th>Common object</th>
<th>Example of interdependence in manufacturing</th>
<th>Examples of coordination processes for managing interdependence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prerequisite</strong></td>
<td>Output of one activity is required by the next activity</td>
<td>Parts must be delivered in time to be used</td>
<td>Ordering activities, moving information from one activity to the next</td>
</tr>
<tr>
<td><strong>Shared resource</strong></td>
<td>Resource required by multiple activities</td>
<td>Two parts installed with a common tool</td>
<td>Allocating resources</td>
</tr>
<tr>
<td><strong>Simultaneity</strong></td>
<td>Time at which more than one activity must occur</td>
<td>Installing two matched parts at the same time</td>
<td>Synchronizing activities</td>
</tr>
</tbody>
</table>
Coordination Theory (Cont’d)

- Processes underlying coordination

<table>
<thead>
<tr>
<th>Process Level</th>
<th>Components</th>
<th>Examples of Generic Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordination</td>
<td>goals, activities, actors, resources,</td>
<td>identifying goals, ordering activities, assigning activities to</td>
</tr>
<tr>
<td></td>
<td>interdependencies</td>
<td>actors, allocating resources, synchronizing activities</td>
</tr>
<tr>
<td>Group decision-</td>
<td>goals, actors, alternatives, evaluations,</td>
<td>proposing alternatives, evaluating alternatives, making choices</td>
</tr>
<tr>
<td>making</td>
<td>choices</td>
<td>(e.g., by authority, consensus, or voting)</td>
</tr>
<tr>
<td>Communication</td>
<td>senders, receivers, messages, languages</td>
<td>establishing common languages, selecting receiver (routing),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>transporting message (delivering)</td>
</tr>
<tr>
<td>Perception of common</td>
<td>actors, objects</td>
<td>seeing same physical objects, accessing shared databases</td>
</tr>
<tr>
<td>objects</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Why to Coordinate?

- **Principle of Bounded Rationality** (Simon 1957): the human mind’s processing capacity is limited.
  - The amount of information that can be processed by an individual is limited
  - The detail of control an individual may wield is limited

- Increasing complexity of computer applications (distributed, open, dynamic, etc.)

- **DAI Perspective**: intelligence is not a property of isolated entities (humans, computers), but of “social” entities ⇒ to understand intelligence requires to deal with systems being able to interact appropriately
Outline

Motivation
Agent Architectures
Coordination
  Overview
  Basic Models and Mechanisms
  Advanced Models and Mechanisms
Overview

- Standard Client-Server Model
- Task and Result Sharing
- Blackboard Model
- Contract Net Model
- FA/C Principle
Standard Client-Server Model

- **Client** = process (program, application) which sends requests for operations to another process
- **Server** = process (program, application) which receives requests for operations from another process
- **Service** = the operations carried out by a server in response to request by some client
- “client” and “server” are roles which are dynamically played by processes
  - a process may act both as a client and as a server
  - a server may be a client of other servers (e.g., web server are often clients of local file servers managing the files in which the web pages are stored)
Standard Client-Server Model (Cont’d)

- Main idea:

(figure from (Langendörfer & Schnorr 94))
Standard Client-Server Model (Cont’d)

- Multiple user-client interactions:

(figures from (Langendörfer & Schnorr 94))
Standard Client-Server Model (Cont’d)

- clients and servers operate at the same level, above some (efficient) operating system kernel:

![Diagram of client-server model]

(figure from (Langendörfer & Schnorr 94))
Standard Client-Server Model (Cont’d)

- Pros and Cons:
  - simple control structure
  - simple synchronization
  - server may turn out as bottleneck (centralization)
  - poor failure tolerance (centralization)
  - information used by server(s) may be outdated (e.g. due to communication delay)

⇒ server replication (but: requires coordination and synchronization on its own), hierarchical control structures
Task and Result Sharing

- Underlying **generic cooperation model**:

- **TASK** decomposition
- **SUBTASKS**
- **SOLUTION SYNTHESIS**
- **SOLUTION** synthesis
- **RESULT PARTICIPATION**
- **SUBSOLUTIONS**
Task and Result Sharing (Cont’d)

- How (sub-)tasks and (sub-)problems can be related to each other:

  - Disjoint
  
  - Overlapping

- Subsuming

- Identical
Task and Result Sharing (Cont’d)

- Task and result sharing offer **potential advantages**:
  - each subproblem can be solved with less knowledge
  - each subproblem requires less resources
  - parallelism and robustness
  - use of multiple sources of knowledge and skills
  - mutual support through exchange of pre-results

⇒ each of these potential advantages requires design efforts on its own!
Task and Result Sharing (Cont’d)

- **Key challenge** raised by task and result sharing: “Which agent is responsible for which part of the overall cooperation process?” (connection problem)
  
  - What needs an involved entity to know?
  - Efforts and costs of interaction and its control?
  - Level of task decomposition?
  - Strategies for synthesis?

⇒ answers depend on relationships between (sub-)tasks/problems
Blackboard Model

- Underlying idea:

- General requirements:
  - common memory
  - read/write control
Blackboard Model (Cont’d)

- **Characteristics:**
  - every participant reads from and writes on common memory area
  - participants may W/R independently or in a coordinated way
  - address of sender needs not to be known
  - participants themselves decide on information announcement (whether, when, …)
  - participants themselves decide on information search and evaluation
  - suited for open applications
  - supports *variability in expertise*
Blackboard Model (Cont’d)

- **Characteristics (Cont’d):**
  - flexible w.r.t. representation (data) structure, but FIXED
  - regionalization/structuring is possible, if required for reasons of efficiency and effectiveness
  - suited for event- and data-driven applications
  - supports incremental generation of solutions
  - typically used as a component in knowledge-based systems
  - first realization: HEARSAY-II (1980)
Contract Net Model

- **General characterization:**
  - Network of nodes (cooperating units) acting as managers and contractors
  - A **manager** announces tasks to be done
  - A **contractor** bids for right to carry out task
  - The contractor responding with the best bid is selected from the announcing manager

⇒ flexible and distributed control, dynamic roles (agent can act as manager and contractor)
Contract Net Model (Cont’d)

- **Node architecture:**

![Node architecture diagram]

- Task processor
- Local database
- Communication processor
- Contract processor
- NODE

NETWORK
Contract Net Model (Cont’d)

- **Negotiation steps**
  1. *Task announcement*
     - eligibility specification (minimal requirements on potential contractor)
     - task abstraction (short description)
     - bid specification (its structure and contents)
     - expiration date
  2. *Bidding*
     - response in accordance with bid specification
  3. *Contracting*
     - selection of best bid according to some criteria
Final remarks:

- Key questions a designer (or contractors/managers) need to answer:
  - What tasks should be announced? (Reasons why an entity should do a task on his/her own?)
  - Who should receive a specific announcement?
  - Why should a potential contractor bid?
  - Selection criteria for managers in case of multiple bids?
  - Selection criteria for contractors in case of multiple announcements?

- Conceptually the contract net is located between master-slave and blackboard models (predominance of manager resp. contractor)
Functionally accurate cooperation (FA/C) as a general design guideline for cooperation when the individuals’ local knowledge is incomplete, uncertain and inconsistent.

“If available information is not perfect, do not longer aim at building a system in which only completely accurate information is exchanged among cooperating entities. Instead, in response to this lack of perfection make sure that the involved parties exchange functionally correct information (tentative partial results) and that they cooperate in refining these information.”

- *functionally correct* = acceptable and reasonable from a party’s local point of view
- *cooperation* = iterative refinement, transformation of local into global correctness
FA/C Principle (Cont’d)

- FA/C requires that an involved party is able to
  - measure and evaluate functional correctness
  - detect inconsistencies (etc) between its tentative partial results and those received from others
  - integrate into its local database those portions of partial results which are consistent with its own results
  - revise and extend its tentative partial results on the basis of the newly integrated data
Note that this slide package (Part 3) is still incomplete. The complete package will be provided later!