Agents

Structural Functional Requirements

Agent-Oriented Approach to Enterprise Modelling

What are Agents?

ERD

From ERD to ARD

JSD

Who is Agent-Oriented?

☞ System/Software Architecture
   ☞ from object to agents
   ☞ smart agents in extended CORBA & other middleware
   ☞ agent-oriented components: duplicate, move around, evolve, die

☞ Computer Games
   ☞ monolithic process -> concurrent process -> agents

☞ Interactive Cinema

☞ Robotics

☞ Information retrieval & filtering (mining)

☞ User Interface Design

☞ Industrial Process Control

☞ RE
Recall: Goal-oriented approach

*But, whose goals are they?*

*Why should the system exist in the first place?*

- Everybody’s?
- Ford’s?
- Part-supplier’s?
- Factory supervisor’s?
- Car dealer’s?
- Alice’s?
- Truck driver’s?
- Developer’s?
- Loan officer’s?

The system exists to help agents achieve their goals?

Cooperation and collaboration:

- Each with her/his own capabilities
  - E.g., can sell 200 cars/month

- Each with her/his own responsibilities
  - E.g., sell 150 cars/month

- Each with her/his own commitments
  - E.g., sell 100 cars/month (due to insufficient incentive)

Possible conflict and synergy

Lack of understanding -> a system not used

E.g., if >150 cars are sold, generate check for 0.1% bonus

View an Enterprise as a social phenomenon

System performance depends on agents & their interactions
What are "Agents"?

**Jacobson:**

* Actors are something "external" to the system.  
* An actor is a way of modelling "users" of the system (human or non-human), not the system itself. Actors are used to find and describe use cases primarily, and not for making system models.

**Recent RE community (some):**

* Agents can be a part of a system, or the whole system itself.  
* The distinction between system external and internal is a matter of choice. Agents are active entities/objects in the sense that they communicate with other agents possibly within the system.

* e.g., KAOS: explicit representation of agents, roles, goals, tasks and resources

---

**What are "Agents"?**

**Distributed AI:**

([Woolridge & Jennings, Agent Theories, Architecture and Languages: A Survey]

An agent is a software-based computer system with:

- **autonomy:**
  
  * agents have some kind of control over their actions & internal state  
  
  (e.g., negotiate thru a bidding process in electronic car purchase)

- **social ability:**
  
  * agents interact with other agents via some kind of agent-communication language  
  
  (e.g., make an offer, counter-offer, confirm, etc.)

- **reactivity:**
  
  * agents perceive their environment and respond in a timely fashion to changes that occur in it  
  
  (e.g., many buyers -> make a high initial bid, repeated attempts, etc.)

- **pro-activeness:**
  
  * agents exhibit goal-directed behavior by taking the initiative  
  
  (e.g., set up goals; consider alternative cars & sellers; trade-off analysis...)

An agent is an entity with reasoning capacities;  
A multi-agent system consists of concurrent objects, with a certain degree of control over their own execution.
**Why does it take so long to have a claim settled after an automobile accident?**

**Why does the company hire appraisers to assess damages?**

**What should a software agent do and why?**

---

**From ERD to ARD**

[Yu & Mylopoulos, ICSE, '94]

- explicit representation of agents, roles, goals, tasks & resources

---

**Client/Server Groupware**

**Workflow**

A *home mortgage application*

- Loan Request
- Credit Analysis
- Appraisal
- Approval
- Answer Customer

- Form Processing
- DB query
- e-mail to accountg
- conferencing
- fax
- spreadsheet

- Many office workers spend a high percentage of their time processing documents
- The "workflow river" carries the flow of work from port to port, value being added along the way
- The workflow definition must take into consideration:
  - *Routes* along which the object moves
  - *Rules* about what information is routed and when
    E.g., "If the loan is over $100,000, send it to the supervisor within the next hour or else send it to the next hop"
  - *Roles* define job functions independently of the people who do it
    E.g., the "supervisor" role can be handled by users "Adam" and "Eve"
**Functional Requirements**

*Structural vs. Behavioral viewpoints:*

**Structural viewpoint:**

*emphasis in the modelling of "information Structures" found in the application together with "rules" (constraints & derivations) which determine allowable states of these structures*

ERD, IDEF1X, IEF/Composer  
Object-Oriented Analysis  
JSD, DFD, PSL/PSA

**Behavioral viewpoint:**

*emphasis in the specification of "activities" operating on the information structures and "events" that trigger these activities*

STD, Structured English, Decision Tables/Trees  
FSMs, StateCharts, (Augmented) Petri-nets  
Function-oriented Models

**Entity Relationship Model (ERM)**

*Historical Background:*

Originally proposed [Chen76] as a semantic data model (SDM) to be used during the design of a database, prior to defining a logical schema.

*A database consists of a collection of:*

* symbol structure types,  
  whose instances are used to represent an application  

* operations,  
  which can be applied to any valid symbol structure  

* integrity rules,  
  which define the set of consistent symbol structure states, or changes of states

*A database typically has two layers:*

* a physical layer (schema):  
  efficient symbol structures & operations (algorithms)  
  e.g., B-trees, indices  

* a logical layer (schema):  
  address representational issue  
  e.g., tables (record sets), tuples (horizontal rows), attributes (vertical columns)
Relational Database

Students

<table>
<thead>
<tr>
<th>ID#</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Alice</td>
<td>Wonderland</td>
</tr>
<tr>
<td>2345</td>
<td>Robinhood</td>
<td>Nottingham</td>
</tr>
<tr>
<td>3456</td>
<td>Pinocchio</td>
<td>Anonymous St., Italy</td>
</tr>
</tbody>
</table>

Courses

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS6361</td>
<td>Requirements Engineering</td>
</tr>
<tr>
<td>CS6362</td>
<td>Software Architecture &amp; Design</td>
</tr>
<tr>
<td>CS6367</td>
<td>Sw Testing, Validation/Verification</td>
</tr>
</tbody>
</table>

❖ Operations: add/delete/modify tuple
❖ Integrity rule: NO 2 tuples in the same table can have the same key.

BUT, tables are "flat", can't describe complex assemblies
❖ SO, use a SDM like ERD

Lawrence Chung

ERD

Symbols used in ERDs:
- entity set
- attribute
- 1 or M
- relationship

Example: ERD for a university database

- STUDENTS
- PROFS
- COURSES
- SEMESTERS
- TAKES
- TEACHES
- NAME
- OFFICE
- YEAR
- SEASON
- ID#
- ADDRESS
- NAME
- SIN
- CHAIR-OF
- MEMBER-OF
- DEPTS

Note: multiple relationships between same entity sets (e.g., member-of, chair-of)
- Appropriate for modelling static parts of an application, but not dynamic parts
- Generalization is not supported
First step in the development process is the construction of a model of the (relevant) real world, consisting of entities & actions.

Entities have roles characterized by responses to sets of actions. They are defined in terms of their life states and the actions that cause these to change.

Actions are discrete events, which take place in the real world external to the system.

E.g., The life of a book entity

- **entity**: book
- **life states**: Purchase, Use *, Terminate
- **actions**: Order, Arrive, Check, Return, Sell, Lose, Destroy

* iteration
○ selection

For subsequent development phases, see: M. Jackson, System Development, Prentice Hall, 1983