**Lecture 5: Modeling Enterprises**

**Last Week:**
Elicitation Techniques
- Interviews, surveys, etc
- Cognitive approaches
- Contextual approaches

**This Week:**
Modeling Enterprises
- General Modeling Issues
- Modeling Human Activity
- Decomposition, Means-Ends Analysis and task dependencies

**Next Week:**
Modeling Info and Behaviour
- Structured and OO methods
- ER and Class Hierarchies
- State machines

---

**RE involves a lot of modelling**

→ A model is more than just a description

% It has its own phenomena, and its own relationships among those phenomena.

> The model is only useful if the model’s phenomena correspond in a systematic way to the phenomena of the domain being modelled.

% Example:

```
<table>
<thead>
<tr>
<th>Designations</th>
<th>Designations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book</td>
<td>Person</td>
</tr>
<tr>
<td>P = Person</td>
<td>R = Wrote</td>
</tr>
<tr>
<td>Book: entity</td>
<td>Person: entity</td>
</tr>
<tr>
<td>author: relation</td>
<td></td>
</tr>
</tbody>
</table>
```

For every B, at least one P exists such that R(P, B)

→ A model is perfect?

% “If the map and the terrain disagree, believe the terrain”
% Perfecting the model is never a good use of your time...

---

**It's only a model**

→ There will always be:

% phenomena in the model that are not present in the application domain
% phenomena in the application domain that are not in the model

→ Modelling can guide elicitation:

% It can help you figure out what questions to ask
% It can help to surface hidden requirements

% i.e. does it help you ask the right questions?

→ Modelling can provide a measure of progress:

% Completeness of the models → completeness of the elicitation (?)
% i.e. if we’ve filled in all the pieces of the models, are we done?

→ Modelling can help to uncover problems

% Inconsistency in the models can reveal interesting things...
% e.g. conflicting or infeasible requirements
% e.g. confusion over terminology, scope, etc
% e.g. disagreements between stakeholders

→ Modelling can help us check our understanding

% Reason over the model to understand its consequences
% Does it have the properties we expect?
% Animate the model to help us visualize/validate the requirements

---

% Common Phenomena
% ghost writers...
% ...pseudonyms...
% ...anonymity...
% every book has at least one author...
% every book has a unique ISBN...
% no two people born on same date with same name...
% ISBN
% title
% name
% DOB

Phenomena not captured in the model

Phenomena not true in the world
Choice of modelling notation

- **natural language**
  - extremely expressive and flexible
  - useful for elicitation, and to annotate models for readability
  - poor at capturing key relationships
- **semi-formal notation**
  - captures structure and some semantics
  - can perform (some) reasoning, consistency checking, animation, etc.
  - e.g. diagrams, tables, structured English, etc.
  - mostly visual - for rapid communication with a variety of stakeholders
- **formal notation**
  - precise semantics, extensive reasoning possible
  - underlying mathematical model (e.g. set theory, FSMs, etc)
  - very detailed models (may be more detailed than we need)
  - RE formalisms are for conceptual modelling, hence differ from most computer science formalisms

Desiderata for Modelling Notations

- **Implementation Independence**
  - does not model data representation, internal organization, etc.
- **Abstraction**
  - extracts essential aspects
  - e.g. things not subject to frequent change
- **Formality**
  - unambiguous syntax
  - rich semantic theory
- **Constructability**
  - can construct pieces of the model to handle complexity and size
  - construction should facilitate communication
- **Ease of analysis**
  - ability to analyze for ambiguity, incompleteness, inconsistency
- **Traceability**
  - ability to cross-reference elements
  - ability to link to design, implementation, etc.
- **Executability**
  - can animate the model, to compare it to reality
- **Minimality**
  - No redundancy of concepts in the modelling scheme
  - i.e. no extraneous choices of how to represent something

Meta-Modeling

- Can compare modeling schema using meta-models:
  - What phenomena does each scheme capture?
  - What guidance is there for how to elaborate the models?
  - What analysis can be performed on the models?
- **Example meta-model:**
  - Propositions about the application domain
  - Facts
    - modify
    - record
  - Events
    - trigger
  - Activities
    - State changes in the application domain
    - Actions inducing change of facts in the application domain

Modelling principles

- Facilitate Modification and Reuse
  - Experienced analysts reuse their past experience
  - they reuse components (of the models they have built in the past)
  - they reuse structure (of the models they have built in the past)
  - Smart analysts plan for the future
  - they create components in their models that might be reusable
  - they structure their models to make them easy to modify
- **Helpful ideas:**
  - Abstraction
    - strip away detail to concentrate on the important things
  - Decomposition (Partitioning)
    - Partition a problem into independent pieces, to study separately
  - Viewpoints (Projection)
    - Separate different concerns (views) and describe them separately
  - Modularization
    - Choose structures that are stable over time, to localize change
  - Patterns
    - Structure of a model that is known to occur in many different applications
### Modelling Principle 1: Partitioning

**Partitioning**
- Captures aggregation/part-of relationship

**Example:**
- Goal is to develop a spacecraft
- Partition the problem into parts:
  - Guidance and navigation;
  - Data handling;
  - Command and control;
  - Environmental control;
  - Instrumentation;
  - Etc.

**Note:** this is not a design, it is a problem decomposition
- Actual design might have any number of components, with no relation to these sub-problems
- However, the choice of problem decomposition will probably be reflected in the design.

### Modelling Principle 2: Abstraction

**Abstraction**
- A way of finding similarities between concepts by ignoring some details
- Focuses on the general/specific relationship between phenomena
  - Classification groups entities with a similar role as members of a single class
  - Generalization expresses similarities between different classes in an 'is-a' association

**Example:**
- Requirement is to handle faults on the spacecraft
- Might group different faults into fault classes based on location:
  - Instrumentation fault,
  - Communication fault,
  - Processor fault,
  - Etc...

- Based on symptoms:
  - Instrumentation fault;
  - No response from device;
  - Incorrect response;
  - Self-test failure;
  - Etc...

### Modelling Principle 3: Projection

**Projection:**
- Separates aspects of the model into multiple viewpoints
- Similar to projections used by architects for buildings

**Example:**
- Need to model the requirements for a spacecraft
- Model separately:
  - Safety
  - Commandability
  - Fault tolerance
  - Timing and sequencing
  - Etc.

**Note:** Projection and Partitioning are similar:
- Partitioning defines a part of relationship
- Projection defines a view of relationship
- Partitioning assumes the parts are relatively independent

### Survey of Modelling Techniques

**Modelling Enterprises**
- Goals & Objectives
- Organizational structure
- Tasks & dependencies
- Agents, roles, intentionality

**Modelling Information & Behaviour**
- Information Structure
- Behavioral views
  - Scenarios and use cases
  - State machine models
  - Information flow
- Timing/Sequencing requirements

**Modelling System Qualities (NFRs)**
- All the 'ilities':
  - Usability, reliability, evolvability, safety, etc...
  - Security, performance, interoperability

**Source:** Adapted from Davis, 1990, and Loucopoulos & Karakostas, 1995, p78.
What is this a model of?

Summary

→ Modelling plays a central role in RE
  % Allows us to study a problem systematically
  % Allows us to test our understanding

→ Many choices for modelling notation
  % In this course, we’ll use (and adapt) various UML notations

→ All models are inaccurate (to some extent)
  % Use successive approximation
  % ...but know when to stop perfecting the model
  % Every model is created for a purpose
  % The purpose is not usually expressed in the model
  % ...so every model needs an explanation

Motivation for enterprise modeling...

Imagine we have interviewed some stakeholders...

<table>
<thead>
<tr>
<th>Chief Executive</th>
<th>Catering Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>When flight is full, VIPs are first to be upgraded.</td>
<td>The food loaded is dictated by the number of passengers travelling in a particular class.</td>
</tr>
<tr>
<td>Discounted tickets should be offered to politicians, since they make important decisions affecting the airline.</td>
<td>A predicted number of passengers on a flight, must be available 24 hours prior to departure.</td>
</tr>
<tr>
<td>Info about frequent fliers should not be made available to outside contractors.</td>
<td>Passengers requiring special meals must indicate their request 24 hours prior to departure.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chief Security Officer</th>
<th>Airline Sales manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of bags in the aircraft’s hold should tally against the list of passengers on board.</td>
<td>A ticket may only be issued when a fare is paid.</td>
</tr>
<tr>
<td>Passenger lists should not be made available to the public.</td>
<td>Passengers requiring special meals must indicate their request 24 hours prior to departure.</td>
</tr>
<tr>
<td>Passengers should check-in only once.</td>
<td>When a discounted ticket is booked, the normal book-ahead requirements do not apply.</td>
</tr>
<tr>
<td>All tickets must carry appropriate endorsements relating to the terms and conditions of issue of reservations.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Travel Agent</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>An agent is responsible for holding and cancelling reservations.</td>
<td></td>
</tr>
<tr>
<td>Tickets offered by an agency have different fares, negotiated with the airline sales department.</td>
<td></td>
</tr>
</tbody>
</table>

How do we get from here to an agreed specification?

Approaches to Enterprise Modeling

→ 1970’s
  % Soft Systems Approaches:
  > involve the entire organisation
  > Be sensitive to political and social context for organisational change
  % Examples: SSM, ISAC

→ 1980’s
  % Knowledge-based Approaches:
  > Use knowledge representation schemes to build executable domain models
  > Capture static and dynamic aspects of the domain
  % Examples: RML, Requirements Apprentice, Nature

→ 1990’s
  % Teleological Approaches:
  > Requirements are really just goals, so model goal hierarchies
  > Focus on the ‘why’ question, rather than ‘what’/‘how’
  > And use scenarios as concrete examples of how goals are (can be) satisfied
  % Examples: KAOS, i*, CREWS...

→ 2000’s...?
**ISAC**

Information Systems Work & Analysis of Changes (ISAC)
- Developed in the 1970's in Sweden
- Emphasizes cooperation between users, developers and sponsors
- Developers' role is to facilitate the process
- Good for information systems; not applicable to control systems.

**ISAC Process**
1. Change Analysis
   - What does the organization want?
   - How flexible is the organization with respect to changes?
2. Activity Study
   - Which activities should we regroup into information systems?
   - Which priorities do the information systems have?
3. Information Analysis
   - Which inputs and outputs do each information system have?
   - What are the quantitative requirements on each information system?
4. Implementation
   - Which technology (info carriers; h/w; s/w) do we use for the information systems?
   - Which activities of each information system are manual, which automatic?

**SSM Approach**
1. Existing situation (unstructured problem)
2. Analyze the problem situation
   - Draw a rich picture
   - Look for problem themes (describe them in natural language)
3. Define relevant systems and root definitions (CATWOE)
   - A root definition is a concise description of a human activity system
4. Build a conceptual model
   - Of the activity system needed to achieve the transformation
   - Process oriented model, with activities & flow of resources
5. Compare conceptual model with step (2)
   - Ordered questioning - questions based on the model
   - Event reconstruction - take past events and compare them to the model
   - General comparison - look for features of the model that are different from current situation
   - Model overlay - paint by point comparison of the two models
6. Debate feasible and desirable changes
   - Three types of change: structural, procedural, attitudinal
7. Implement changes
SSM modeling

Root definition:

“A hospital-owned system, which provides records of spending on drugs so that central action by administrators and doctors to meet defined budgets can be taken jointly.”

Customers: Administrators, Doctors
Actors: Not stated
Transformation: Need to know spending on drugs
Weltanschauung: Monitoring spending on drugs is possible and is an adequate basis for joint control action
Owner: Hospital
Environment: Hospital mechanisms, roles of administrators and doctors, defined budgets

1. appreciate mechanisms by which spending on drugs occurs
2. obtain info on budgets
3. appreciate administrator and doctor roles in controlling spending on drugs
4. decide how to collect info on spending on drugs
5. collect info on spending on drugs
6. make records available to administrators and doctors
7. record info on spending on drugs
8. decide how to record info, so that control against budget by administrators and doctors is possible

Monitor 1–8
Take control action

Appreciate hospital aspirations for the system
Define criteria for effectiveness, efficacy and efficiency of the system

1. appreciate hospital aspirations for the system
2. obtain info on budgets
3. appreciate administrator and doctor roles in controlling spending on drugs
4. decide how to collect info on spending on drugs
5. collect info on spending on drugs
6. make records available to administrators and doctors
7. record info on spending on drugs
8. decide how to record info, so that control against budget by administrators and doctors is possible

Approach

- SD model shows dependencies between actors:
  - goal/softgoal dependency - an actor depends on another actor to attain a goal
  - resource dependency - an actor needs a resource from another actor
  - task dependency - an actor needs another actor to carry out a task
- SR model shows interactions between goals within each actor:
  - Shows task decompositions
  - Shows means-ends links between tasks and goals

E.g. Strategic Dependency Model

E.g. Strategic Rationale Model
KAOS

→ Background
  - Developed in the early 90’s
    - first major teleological requirements modeling language
    - full tool support available
    - has been applied to a number of industrial case studies
  - Two parts:
    - Informal goal structuring model
    - Formal definitions for each entity in temporal logic

→ Approach
  - Method focuses on goal elaboration:
    - define initial set of high level goals & objects they refer to
    - define initial set of agents and actions they are capable of
  - Then iteratively:
    - refine goals using AND/OR decomposition
    - identify obstacles to goals, and goal conflicts
    - operationalize goals into constraints that can be assigned to individual agents
    - refine & formalize definitions of objects & actions

Business Processes

→ Business Process Automation
  - Leave existing business processes as they are
  - Can make an organisation more efficient; has least impact on the business

→ Business Process Improvement
  - Make moderate changes to the way the organisation operates
  - E.g. improve efficiency and/or effectiveness of existing process
    - Techniques: Duration analysis; activity-based costing; benchmarking

→ Business Process Reengineering
  - Fundamental change to the way the organisation operates
  - Techniques:
    - Outcome analysis - focus on the real outcome from the customer’s perspective
    - Technology analysis - look for opportunities to exploit new technology
    - Activity elimination - consider each activity in turn as a candidate for elimination

Modelling Business Processes

→ Business processes involve:
  - Multiple actors (people, business units, ...)
  - Concurrent activities
  - Explicit synchronization points
    - E.g. same task cannot start until several other concurrent tasks are complete
  - End-to-end flow of activities

→ Choice of modelling language:
  - UML Activity diagrams
    - based on flowcharts and petri nets
    - Not really object oriented (poor fit with the rest of UML)
  - Business Process Modelling Notation (BPMN)
    - New (emerging) standard, loosely based on pi calculus
Using UML for enterprise modelling

→ Use Cases
  % Already assume the basic functions of the machine have been decided
  % Hence, it's premature to look for Use Cases yet...

→ Collaboration/Activity Diagrams
  % Show how classes (actors?) collaborate to perform tasks
  ➢ Represent “message” flows between objects
  ➢ Offer a simple way of diagramming scenarios
  % But do not show:
  ➢ Intentionality, task dependency, task decomposition

→ Class diagrams
  % Show the actors/roles and entities in the domain
  ➢ Concentrate on static structure
  ➢ Can implicitly capture business rules through multiplicity constraints
  % Must remember to model domain entities rather than implementation classes

→ Conclusion
  % UML offers only very crude tools for enterprise modeling

Example Activity Diagram

Collaboration Diagrams

→ Example - “select courses to teach”

Example Activity Diagram with Swinlanes

Activity Diagram with Swinlanes
Capturing Business Rules

→ Why do we care about business rules?

- They help us to understand the business context
- They could be important constraints for the design of the new system
  - E.g. constraints on when and how operations can happen
  - E.g. constraints on the state space of objects
- They help us write "operation specifications" for class operations

→ How do we specify business rules?

- Natural Language
  - such descriptions can be highly ambiguous
- Structured English
  - use a subset of a natural language (limited syntax and vocabulary)
  - can be hard to write, hard to verify, and too close to program code
- Decision Tables
  - use a table representation of alternative outcomes (similar to truth tables)
  - Decision Trees
  - use a tree representation of alternative outcomes
- Object Constraint Language
  - UML notation for adding extra constraints to models
  - Can also be used for specifying pre- and post-conditions on operations

Decision Tables

→ Inputs as columns, actions (outputs) as rows

- If there are n parameters (conditions) to a decision, each with k₁, k₂,..., kₙ values, then table has:
  - k₁ x k₂ x ... x kₙ columns
  - as many rows as there are possible actions

- For example:
  - "If the plane is more than half full and the flight costs more than $350 per seat, serve free cocktails, unless it is a domestic flight. Charge for cocktails in all domestic flights where cocktails are served, i.e., those that are more than half full"

<table>
<thead>
<tr>
<th>conditions</th>
<th>Domestic?</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>≥ half full?</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>≥ $350/seat</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Free cocktails?</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Decision Trees

→ Represent the decision logic as a tree:

- Nodes of the tree represent input parameters (questions)
- Leaves of the tree represent outputs (actions)

→ Example: