Lecture 11: Requirements Modelling

- A little refresher:
  - What are we modelling?
  - Requirements: Systems, Systems Thinking

- Role of Modelling in RE
  - Why modelling is important
  - Limitations of modelling

- Brief overview of modelling languages

- Modelling principles
  - Abstraction
  - Decomposition
  - Projection
  - Modularity

Refresher: Definitions

- Some distinctions:
  - Domain Properties: things in the application domain that are true whether or not we ever build the proposed system
  - Requirements: things in the application domain that we wish to be made true by delivering the proposed system
  - A specification: a description of the behaviours the program must have in order to meet the requirements

- Two correctness (verification) criteria:
  - The Program running on a particular Computer satisfies the Specification
  - The Specification, in the context of the given domain properties, satisfies the requirements

- Two completeness (validation) criteria:
  - We discovered all the important requirements
  - We discovered all the relevant domain properties

Refresher: Systems to model

Refresher: Systems Thinking
Modelling...

→ 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

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>Book</th>
<th>ISBN</th>
<th>title</th>
<th>name</th>
<th>DOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>1.0</td>
<td>B</td>
<td>Bob</td>
<td>1950</td>
</tr>
<tr>
<td>B2</td>
<td>2.0</td>
<td>B</td>
<td>Bob</td>
<td>1950</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Person</th>
<th>author</th>
<th>relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>John</td>
<td>House</td>
</tr>
<tr>
<td>P2</td>
<td>John</td>
<td>Office</td>
</tr>
</tbody>
</table>

"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

→ A model is never perfect
  % "If the map and the terrain disagree, believe the terrain"
  % Perfecting the model is not always a good use of your time...

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 organisation, 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

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)
  - Redundancy: Usability, reliability, evolvability, safety, security, performance, interoperability.
- Meta-Modelling
  - Can compare modelling 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 record
    - Activities
      - trigger
      - State changes in the application domain
      - Actions induce change of facts in the application domain

the Unified Modelling Language (UML)

- Third generation OO method
  - Booch, Rumbaugh & Jacobson are principal authors
    - Still evolving
    - Attempt to standardize the proliferation of OO variants
  - Is purely a notation
    - No modelling method associated with it
    - Was intended as a design notation (some features unsuitable for RE)
  - Has become an industry standard
    - But is primarily owned by Rational Corp. (who sell lots of UML tools and services)
- Has a standardized meta-model
  - Use case diagrams
  - Class diagrams
  - Message sequence charts
  - Activity diagrams
  - State diagrams
  - Module diagrams
  - Platform diagrams
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

Note:
- Projection: relates concepts from different models
- Partitioning: relates components within a model
- Abstraction: relates different aspects of a single model

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: or ☰ based on symptoms:
      ☰ instrumentation fault; or ☰ no response from device;
      ☰ communication fault; or ☰ incorrect response;
      ☰ processor fault; or ☰ 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

**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

Source: Adapted from Davis, 1990, p67-68