What is a model?

“A model is an abstraction of something for the purpose of understanding it before building it.” [BR05]

“A description of static and/or dynamic characteristics of a subject area, portrayed through a number of views (usually diagrammatic or textual).” [Lar05]
What are we modelling?

- **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**: is a description of the **behaviours** that the program must have in order to meet the requirements.
  - Can only be written in terms of shared phenomena!

**Application Domain**
- D - domain properties
- R - requirements

**Machine Domain**
- C - computers
- P - programs

---

Systems to Model

**Subject System**
- Needs information about
- Maintains information about
- Uses
- Information system
- Usage System
- builds
- Development System

---

Modelling Example

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

- **Book**: entity
  - title
  - author
  - ISBN

- **Person**: entity
  - name

- **Wrote**: relation

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

---

Modelling

- **Modelling can guide elicitation**: It can help you figure out what questions to ask.
- **Modelling can provide a measure of progress**: Completeness of the models \( \iff \) completeness of the elicitation (?).
Modelling [2]

• 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

Remember: “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

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, security, performance, interoperability,...

[Jac95]
Unified Modeling 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)
    - Rational Corp. is owned by IBM

UML Diagrams

- Activity diagrams
- Class diagrams
- State diagrams
- Use case diagrams
- Sequence charts
- Component diagrams
- Deployment diagrams

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

Desiderata for Modelling Notations [2]

- **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

[LK95]
**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
  - Partitioning (Decomposition)
  - Projection (Viewpoints)
  - Modularization
  - Patterns (e.g., problem frames)
    - Structure of a model that is known to occur in many different applications

- **Partitioning**
  - Partition a problem into independent pieces, to study separately
    - 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

- **Abstraction**
  - strip away detail to concentrate on the important things
  - 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: no response from device; incorrect response; self-test failure; etc...

- **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

[Ref: LK95]
Example: UML Class Diagram

Modelling Notations in CSC340

- **UML**
  - Activity Diagrams
  - Class Diagrams
  - Statechart Diagrams
  - Use Case Diagrams
  - Sequence Diagrams

- **Non-UML**
  - Goal models
  - ER Diagrams
  - SCR
  - Fault tree models

You will learn:
- what information each model captures (what does it tell you?)
- how the models relate to each other
- syntax
- how to develop the models

References