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!

Systems to Model

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

What are we modelling?

<table>
<thead>
<tr>
<th>Application Domain</th>
<th>Machine Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>D - domain properties</td>
<td>C - computers</td>
</tr>
<tr>
<td>R - requirements</td>
<td>P - programs</td>
</tr>
</tbody>
</table>

Lecture 5, Part 2: Modelling Requirements

Jennifer Campbell
CSC340 - Winter 2007
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.

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

The application domain

Designations for the application domain

B = Book
P = Person
W = Wrote

Common Properties

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

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…

[Jac95]

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?

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]

[LK95]
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

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
    - Choose structures that are stable over time, to localize change
    - Patterns (e.g., problem frames)
    - Structure of a model that is known to occur in many different applications
  - 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**
  - 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 symptoms:**
  - Instrumentation fault, etc.
  - No response from device, etc.
  - Communication fault, etc.
  - Processor fault, etc.
  - 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

Example: UML Class Diagram

- **Generalization** (an abstraction hierarchy)
  - Patient
    - Name
    - Date of Birth
    - Physician history
  - Out-patient
    - Room
    - Bed
    - Treatments
    - Food preferences
  - In-patient
    - Last visit
    - Next visit
  - Prescription
    - Natural/artif.
    - Vision
    - Colour
- **Aggregation** (a partitioning hierarchy)
  - Patient
    - Name
    - Date of Birth
    - Physician history
  - Mental Health
    - Mental health
  - Nutrition
    - Nutrition
  - Vision
    - Vision
  - Colour
    - Colour

Modelling Notations in CSC430

- **UML**
  - Activity Diagrams
  - Class Diagrams
  - Statechart Diagrams
  - Use Case Diagrams
  - Sequence Diagrams
- **Non-UML**
  - Goal models
  - ER Diagrams
  - SCR
  - Fault tree models

References


