Systems Modelling (EE5525)

Prof Peter R Hobson

What is a system?

- Integrated whole comprised of identifiable sub-units and processes
- The whole has properties *beyond* those of the parts that comprise it
  - New behaviours
  - New causality
  - Concept of order

Concepts 1

Systems can be seen as a *network of relationships*

Relationships are key to the organisation of the system

Not all properties of a system contribute to the behaviour of the system

A variety of different views of the same system are useful (see later – this is why in UML there are a wide variety of different model views, and associated graphical symbols, available)

Concepts 2

- Systems are special
- Systems arise from *organisation*
- Getting the organisation correct is a key, and major, aspect of creating a system
- Systems exist in *environments*
  - No real system is isolated
  - Isolation is a tool to understanding complex behaviour
  - Beware of side-effects
  - Defining the boundary is important as is understanding which external actors interact with the system across it.

What is a model?

- “A model is an object that for its user stands in for some other thing, the original, and with which the user can interact to answer questions about the original” – Dr M Elstob
- “A model is a representation in a certain medium of something in the same or another medium” – Rumbach, Jacobson & Booch

What are models for?

- Capturing and stating requirements and knowledge so that *stakeholders** may agree and understand them
- To aid thinking about the design of a system
- To capture design decisions
- To generate a product
- To organise information about large complex systems
- To explore multiple solutions
- To master complexity (reduction by abstraction)

* Note: Stakeholders may or may not also be actors

Types of model

Models *abstract* from reality, they generalise, leave out details and may be applicable beyond their original problem.

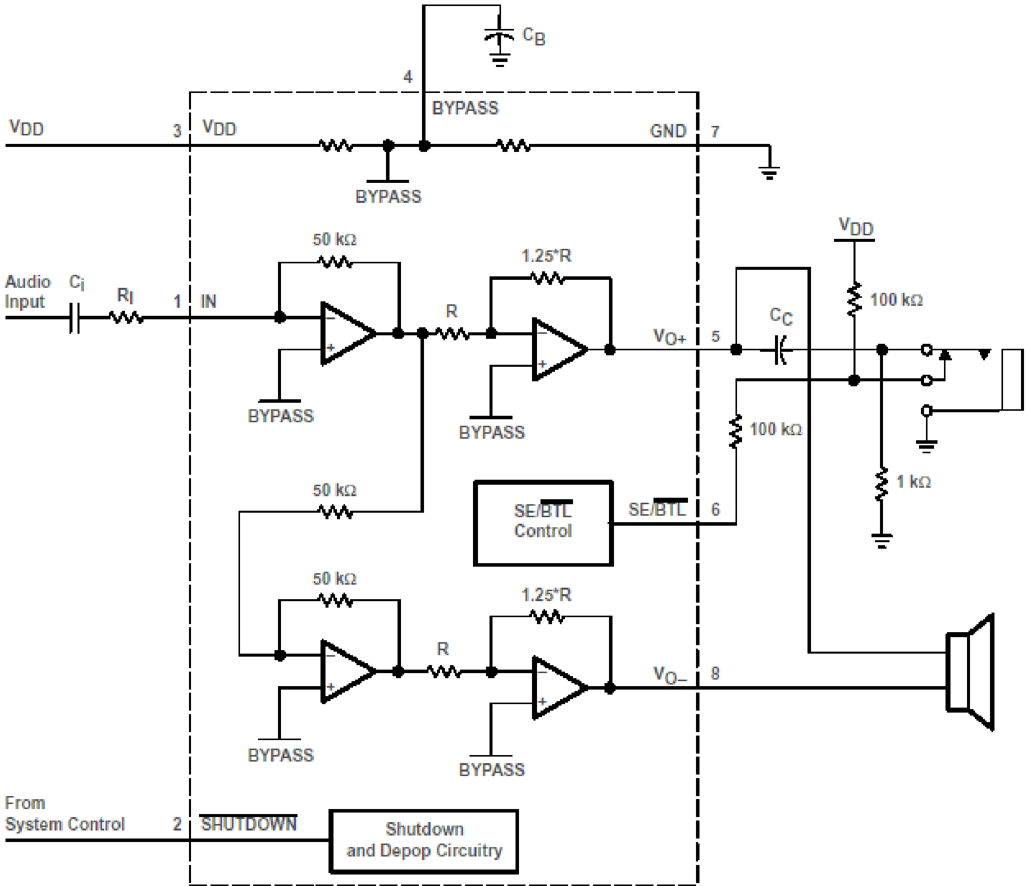
- **Physical** – scale model, working model, mock-up
- **Iconic** (picture like) – photographs, drawings, sculpture
- **Diagrammatic** – plans, electronic wiring, charts, block diagram, flow chart
- **Linguistic** – memos, reports, lists, pseudo-code
- **Mathematical** – equations in physics & engineering, financial models, computer simulations

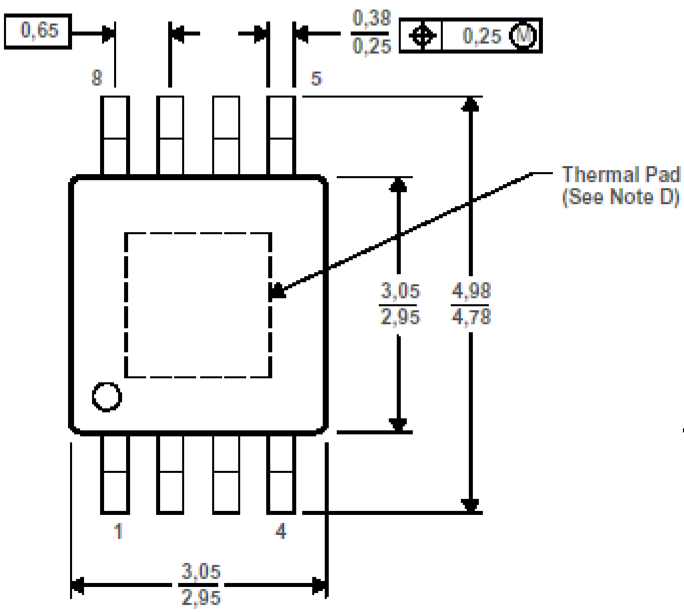
Spectrum and purpose of models

- **Physical** – strongly resemble the **original**. Connections are usually obvious (however see water-flow model of British economy in the London Science Museum)
- **Iconic –** usually **visually** similar to the original (or similar to some aspect of the original, e.g. electronic circuit diagrams)
- **Symbolic –** (for example linguistic and mathematical models) have no resemblance to the original

Models have different forms for different purposes, they often have widely differing levels of abstraction

Iconic models





Levels of detail in models

- High level
  - These are built early in a project to focus the initial thought process of the *stakeholders*. Options are highlighted and requirements are captured
- Abstract specifications
  - Focusing on key concepts and mechanisms
  - Some correspondence with the final system, but many details missing
- Full specifications of a final system
  - Contain enough information to build a system
- Examples of *possible systems* (what could be as well as what will be)

What is in a model?

- Semantics, presentation and rules
  - Semantics = meaning
  - Presentation = notation (e.g. UML) = syntax
  - Rules = pragmatics
- Semantics
  - Capture the meaning in a series of logical constructs. The semantic model can be used to construct code (for example).
  - The semantic model has *structure*, *rules*, *dynamics*
  - Some correspondence with the final system, but many details missing
- Context

½ way summary

- Systems
  - Relationships, complexity, organisation
  - More than the sum of the component parts
  - Context (environment)
- Models
  - A way of handling complexity
  - Different types of model
  - Different levels of detail
  - *Semantics* and *notation*

Events

- The concept of an event is central to modelling functional requirements.
- Occurs at a specific time & place
- Events trigger all the processing done by a system, thus listing and analyzing them is useful when defining the requirements.

What follows is based on Chapter 5 of Satzinger JW, Jackson RB & Bird SD "Systems Analysis & Design in a Changing World"

Events

- Treat system as a black box
- Focus on external users (actors)
- Three catagories of event
  - External (in the environment)
  - Temporal
  - State

External Events

- Actors initiate these from outside the system boundary.

  e.g. *Customer Places Order*
- Try to identify all the actors/stakeholders who might want something from the system.
- Some possible external events
  - Actor wants something resulting in a transaction
  - Actor wants information
  - Actor wants data updated

Temporal Events

- A point in time is reached

  e.g. Show next frame of animation
- Not initiated by an external actor
- Some possible external events
  - Exception or summary reports
  - Operational reports
  - Update graph or picture

State Events

- Internal trigger
- Not directly initiated by an external actor
- Can have similarities to temporal events but they particular point in time cannot be defined
- Some possible state events
  - Re-order stock
  - Flush cache

Identifying Events

- Distinguish between the event the condition and the response.
- E.g. “Buy book from Foyles (a large London bookstore)”
  - Student discovers she needs a book to understand the EE5551 course
  - Looks in Brunel library but it is not available
  - Goes to Foyles bookshop
  - Takes book from shelf and asks to purchase it
  - Hands over credit card
  - Completes transaction and returns to room to read it

Looking at Events

- For each event
  - What occurred (the *trigger*)
  - Locate the *source*
  - What happens as a result (what the system does is known as a *Use Case*)
  - What is the output (*response*) and which *actor* receives it

Things

- Modelling *things* is a key activity
  - In OO *things* will become identified with objects
  - Types of thing:
    - Tangible Book, Car, Document,
    - Role play Customer, Accountant
    - Organisational Units Department, Faculty, Task Force
    - Devices Sensor, Keyboard, Mouse, Menu, Button
    - Incidents Logon, Order, Flight, Payment
    - Locations Factory, Desktop, Shop

OO System development

- Need to have a systematic approach (except for *very trivial* systems)
- Several different methods have been proposed and used (see later lectures)
- In general there are distinct operations
  - Requirements specification
  - Analysis
  - Design
  - Implementation
  - Maintenance

OO analysis

- Functional or data-driven analysis focus on behaviour and/or data *separately*
- OO combines these and looks for objects
- We must
  - Find objects
  - Organise them: determine the class hierarchy
  - Describe their interactions: interfaces
  - Define the operations (methods): limit complexity
  - Implement the methods internally
- All of these steps are interdependent.

OO construction

- Implementation of the analysis model
- Tensions between structure and efficiency
- Reuse of components (e.g. source code)
- Try to have objects identified in the analysis phase mapped on to objects in the design phase – traceability
- Programming note: OO programming is a technique which is *not restricted* to languages that support inheritance, polymorphism, encapsulation

OO testing

- Test from low level units and progress to integration
- Because objects communicate with each other the low level phase typically deals with larger units than in non-OO system testing
- Integration testing starts quite early on in the overall process
- Some new problems arise due to inheritance

How to find out more

- Systems thinking
  - There is an interesting set of pages at
    - http://www.mapnp.org/library/systems/systems.htm
- Book
  - A book which compares "traditional" (i.e. structured) analysis and design with OO is:
    - Satzinger JW, Jackson RB & Bird SD "Systems Analysis & Design in a Changing World" 4th ed., Course Technology, 2006