Queen Mary
University of London

ECS505U
SOFTWARE ENGINEERING

MUSTAFA BOZKURT

LECTURER IN SOFTWARE ENGINEERING

Queen Mary
University of London

Week 2

Systems Concepts and Modelling

Queen Mary
University of London

LESSON OBJECTIVES

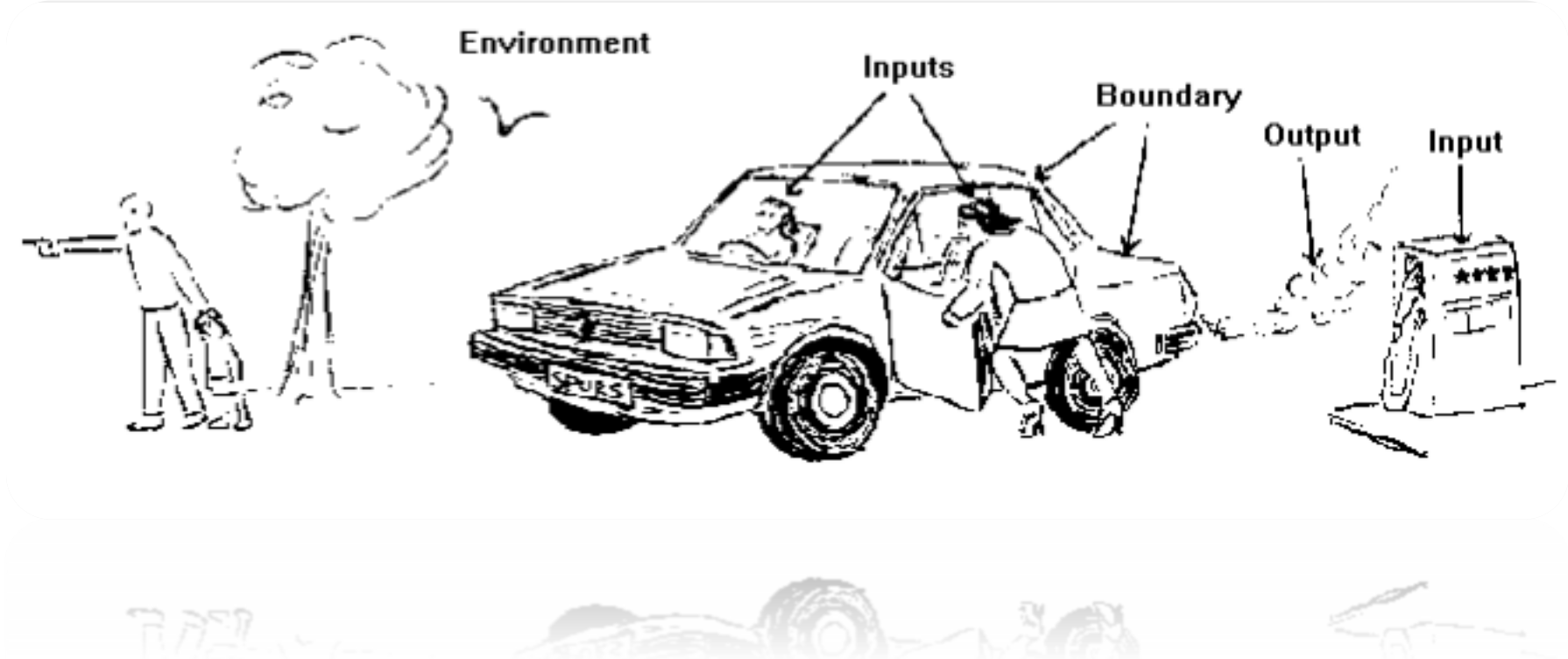
- **Understand basic systems concepts**
- **Understand the basic concepts of modeling**
- **Understand the basic concepts of design**
  - Modularity, abstraction and encapsulation
  - Trade-off between coupling and cohesion

WHAT IS A SYSTEM?

“A system is an organised or complex whole: an assemblage or combination of things or parts forming a complex or unitary whole.” (Kast & Rosenzweig)

“A system is a set of interrelated elements” (Ackoff)

A SYSTEM CAR



A SYSTEM CAR

A SYSTEM CAR

Queen Mary
University of London

A SYSTEM: COMPANY PAYROLL

Key Inputs: employee information

Key outputs: payslips, cheques, cash

Physical components: people, paper, computers

Conceptual components: pension fund

A SYSTEM: LONDON UNDERGROUND

Queen Mary University of London

Key inputs: passengers, MONEY

Key outputs: weary and frustrated passengers, fumes

Physical components: trains, drivers, track, signals

Conceptual components: routes, timetables

Purpose from different perspectives?

A SYSTEM: GENERAL PROPERTIES

Queen Mary University of London

- Made up of components, both physical and conceptual.
- Receives inputs and transforms these into outputs.
- Exists within an environment.
- Boundary divides things inside the system from things outside.
- Exhibits behaviour.
- Fulfils some specific purpose which varies according to particular viewpoints.

Queen Mary
University of London

MODELLING SYSTEMS

Software Design is about modelling software systems.

A model is an abstraction of some aspect of an existing or planned system.

MODELLING SYSTEMS

Systems modelling (or system modelling) is the interdisciplinary study of the use of models to conceptualize and construct systems in business and IT development.

Wikipedia

MODELLING SYSTEMS

The software modeling community is primarily concerned with reducing the gap between problem and software implementation through the use of models that describe complex systems at multiple levels of abstraction…

Atlee et al. “Modeling in Software Engineering” International Conference on Software Engineering 2007

Queen Mary
University of London

MODELLING SYSTEMS

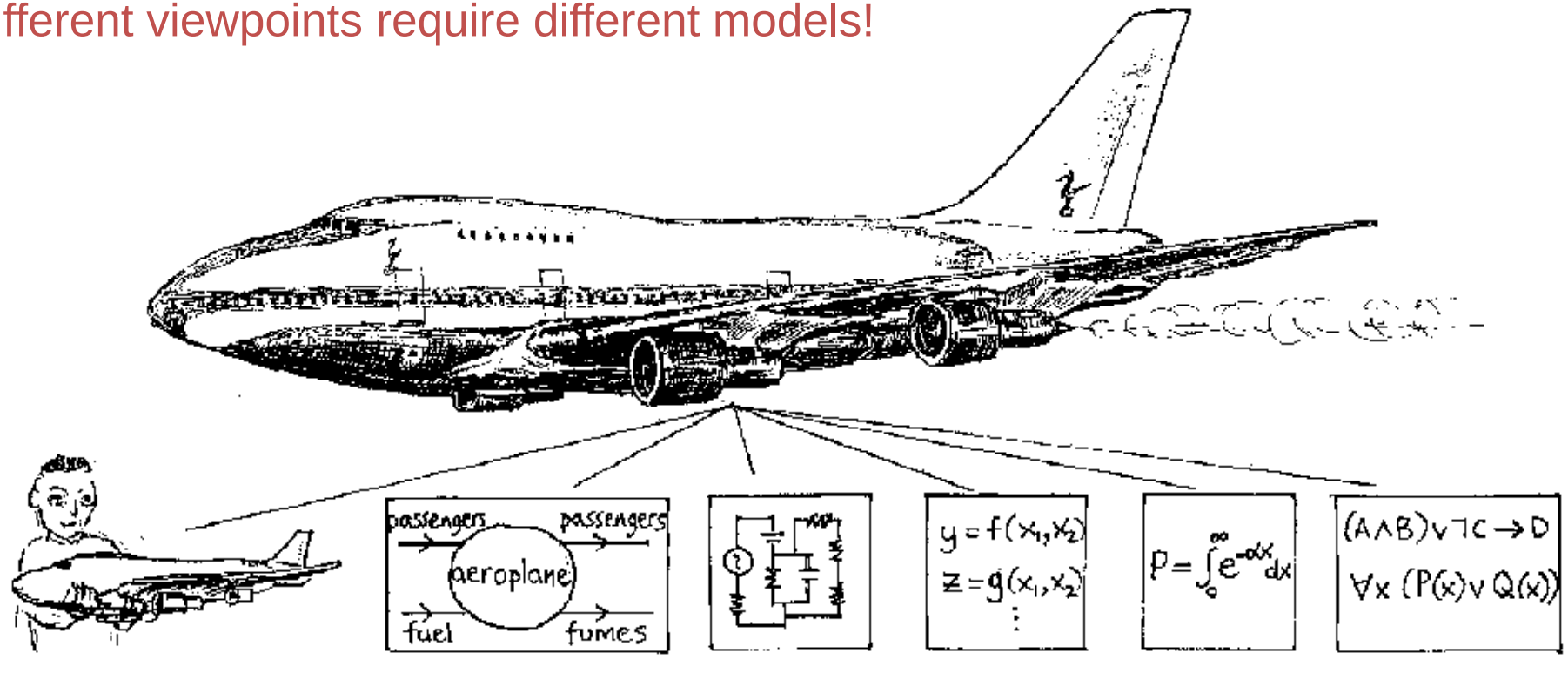
In order to communicate system requirements clearly and accurately to both business and IT stakeholders, it is vital for analysts and designers to be able to construct models from a variety of perspectives.

In most cases these models will provide the basis for more detailed design.

BCS Systems Modelling Certificate

MODELLING SYSTEMS

Different viewpoints require different models!



MODELLING SYSTEMS

Ian Sommerville's book

Process models: Process models show the overall process and the processes that are supported by the system

Behavioural models: Behavioural models are used to describe the overall behaviour of a system

Data-processing models: Data flow diagrams (DFDs) may be used to model the system's data processing.

A DFD is a graphical representation of the "flow" of data such as customer names and transaction details through an information system

SOFTWARE MODELLING METHODS

Queen Mary University of London

1970s: Flowcharts, data flow diagrams, Jackson structured design

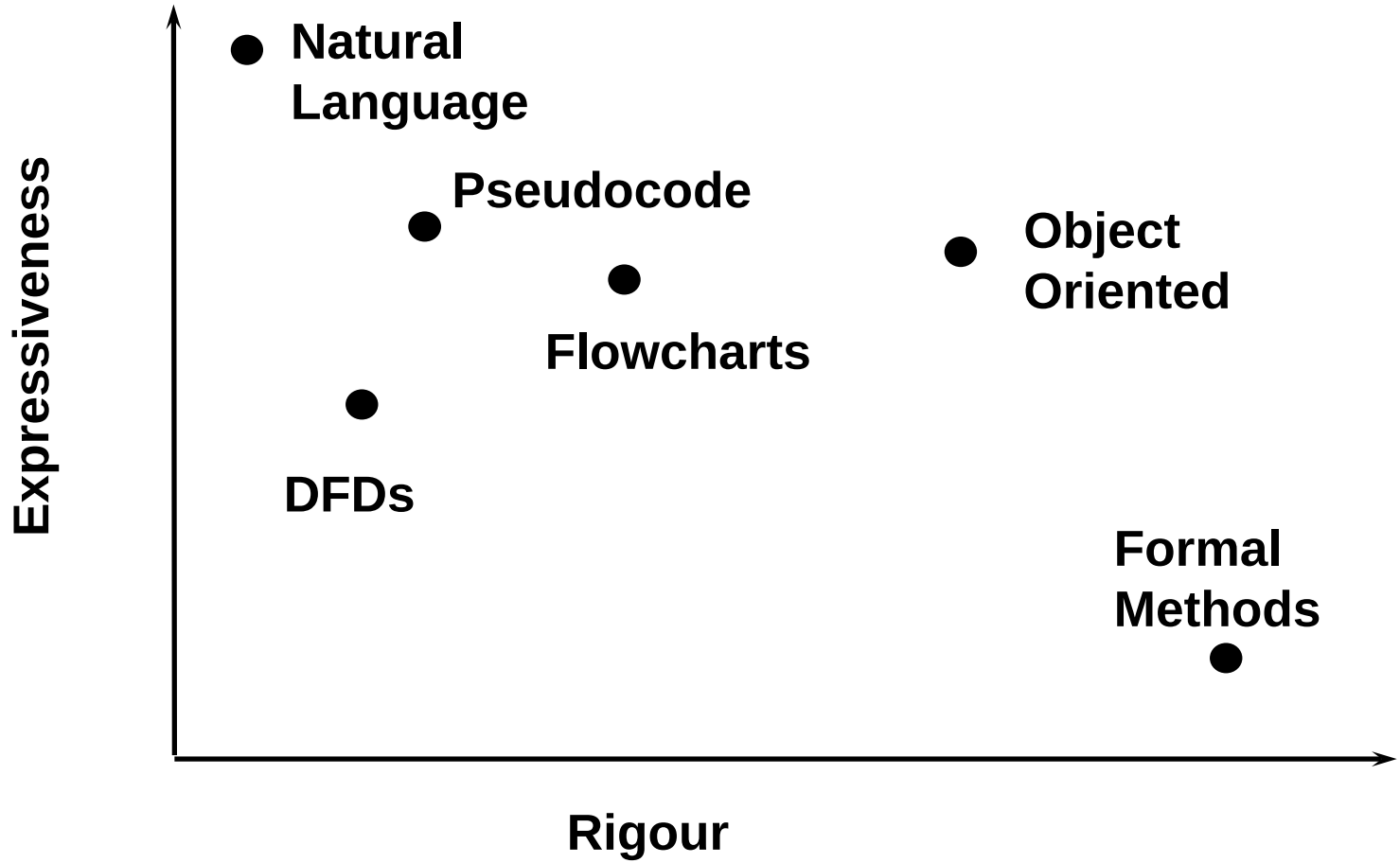
1980s: Formal methods

1990s: Object-oriented methods

2000s: OO methods consolidated in UML

RIGOUR AND EXPRESSIVENESS OF METHODS

Queen Mary University of London



DESIGNING SYSTEMS

System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements.

Queen Mary University of London

DESIGN PRINCIPLES

Modularity

- Breaking of a complex system into manageable parts

Abstraction

- Providing simplified representation of something complex

Encapsulation

- Keeping implementation separate from ‘contractual’ interface and hidden from the clients

Hierarchy

- Later!

MODULARITY

https://www.st.cs.uni-saarland.de/edu/se/2010/lecture-slides/07-SoftwareDesign_Handout1.pdf

THE CRUCIAL NOTION OF MODULARITY

Queen Mary University of London

To control complexity

To enable appropriate ‘independence’

- Separation of concerns
- Enable independent development
- Easier to test and maintain
- Damage control when error occurs
- Enables reuse

But how best to design modularity?

SOFTWARE SYSTEMS AND 'COMPONENTS'

Queen Mary University of London

- Systems
- Subsystems
- Packages
- Classes
- Methods
- …

COHESION: DEFINITION

The cohesion of a component is the extent to which the *component has a single clear purpose or function*

Queen Mary University of London

COHESION: TYPES

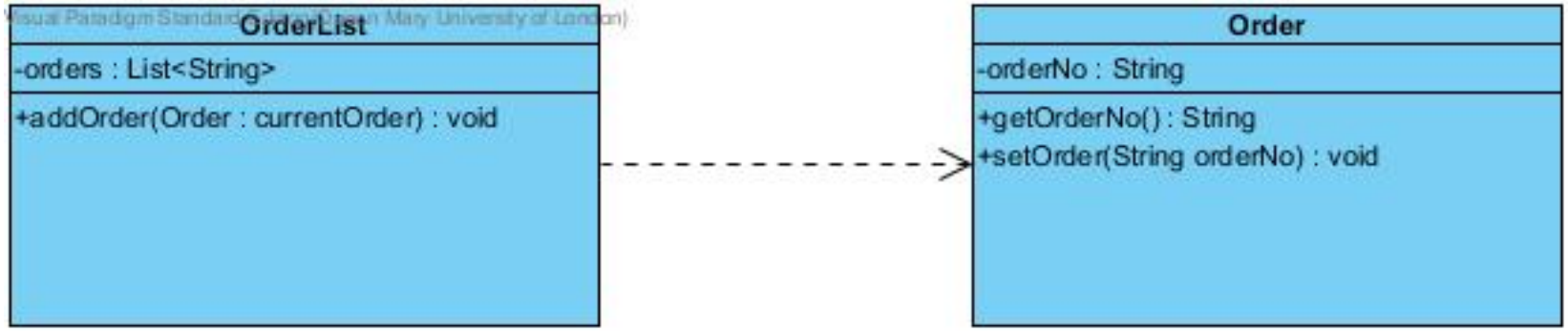
Cohesion Type	Comments
Functional	Facilities are kept together that perform only one computation with no side effects. Everything else is kept out.
Layer	Related services are kept together, everything else is kept out, and there is a strict hierarchy in which higher-level services can access only lower-level services. Accessing a service may result in side effects.
Communicational	Facilities for operating on the same data are kept together, and everything else is kept out. Good classes exhibit communicational cohesion
Sequential	A set of procedures, which work in sequence to perform some computation, is kept together. Output from one is input to the next. Everything else is kept out
Procedural	A set of procedures, which are called one after another, is kept together. Everything else is kept out
Temporal	Procedures used in the same general phase of execution, such as initialization or termination, are kept together. Everything else is kept out
Utility (coincidental)	Related utilities are kept together, when there is no way to group them using a stronger form of cohesion

Queen Mary
University of London

COHESION: BENEFITS

1. Easier to understand, less complex modules
2. Ease of maintenance
3. Increased reusability

COMPONENT DEPENDENCY

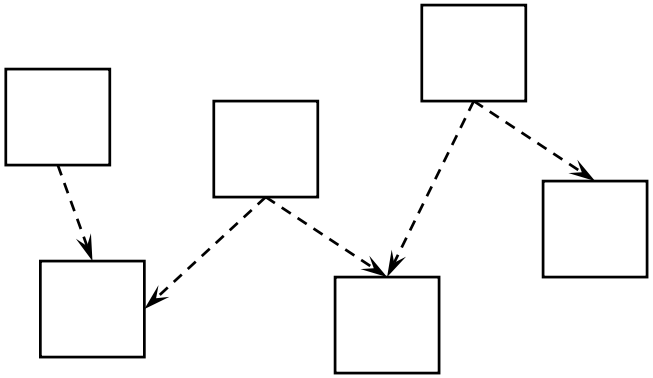


Component OrderList depends on Order

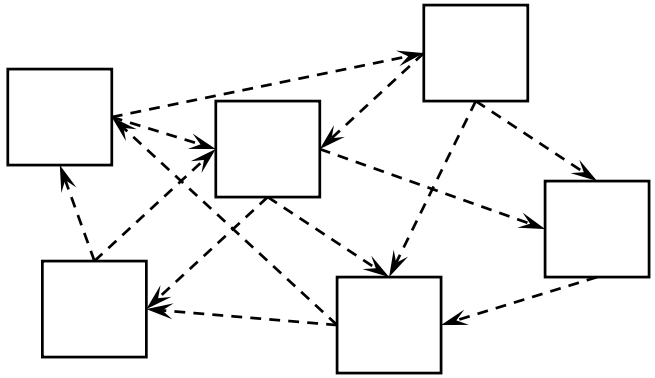
A change to Order may require a change to OrderList

COUPLING: DEFINITION

The level of interaction between components in a system.



‘loosely’ coupled system



‘tightly’ coupled system

Queen Mary University of London

COUPLING: TYPES

Coupling type	Comments
Content	A component surreptitiously modifying internal data of another component. Always avoid this.
Common	The use of global variables. Severely restrict this
Control	One procedure directly controlling another using a flag. Reduce this using polymorphism.
Stamp	One of the argument types of a method is one of your application classes. If it simplifies the system, replace each such argument with a simpler argument (an interface, a superclass or a few simple data items)
Data	The use of method arguments that are simple data. If possible, reduce the number of arguments.
Routine call	A routine calling another. Reduce the total number of separate calls by encapsulating repeated sequences.
Type use	The use of a globally defined data type. Use simpler types where possible (superclasses or interfaces).
Inclusion/import	Including a file or importing a package. Eliminate when not necessary
External	A dependency exists to elements outside the scope of the system, such as the operating system, shared libraries or the hardware. Reduce the total number of places that have dependencies on such external elements.

EXAMPLE: GLOBAL VARIABLES


```
1 public class Global {
2
3       public static int globalVariable;
4
5       public void setGlobalVariable(int globalVariable) {
6             this.globalVariable = globalVariable;
7       }
8
9 }
```

Queen Mary
University of London

COUPLING: BENEFITS

1. Easier to understand
2. Increased maintainability
3. Increased reliability (relatively)

COUPLING AND COHESION: GETTING THE RIGHT BALANCE

Queen Mary University of London

As far as possible systems should be decomposed at each relevant level of abstraction into components which individually have high cohesion and such that there is a low level of overall coupling.

COUPLING AND COHESION: GETTING THE RIGHT BALANCE

Queen Mary University of London

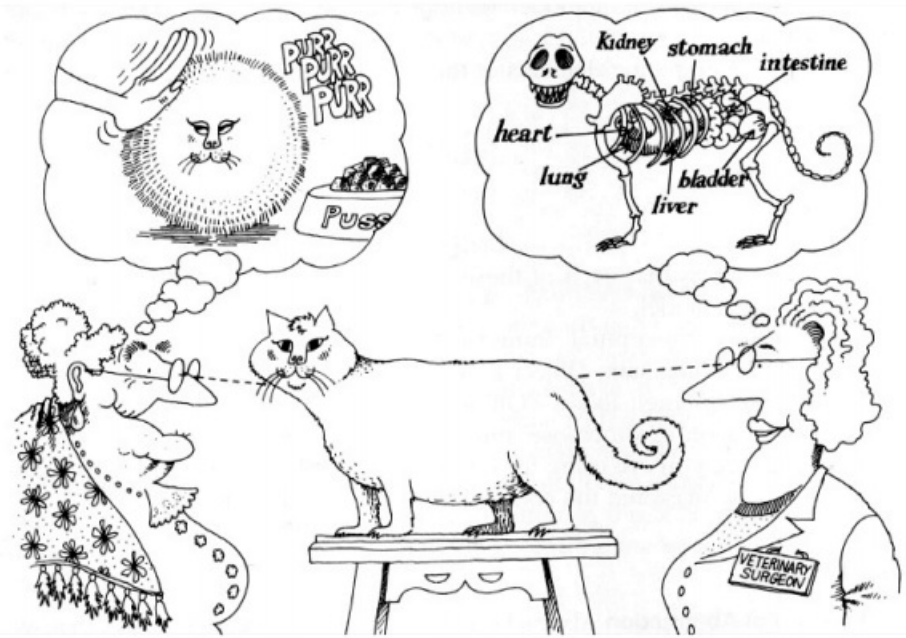
According to the literature:

Low coupling often correlates with high cohesion, and vice versa.

In a sample of 450 routines, the routines with the highest coupling-to-cohesion ratios had 7 times more errors than those with the lowest ratios.

Richard W. Selby, and Victor R. Basili, *Analyzing Error-Prone System Structure*, IEEE Transactions on Software Engineering, 1991

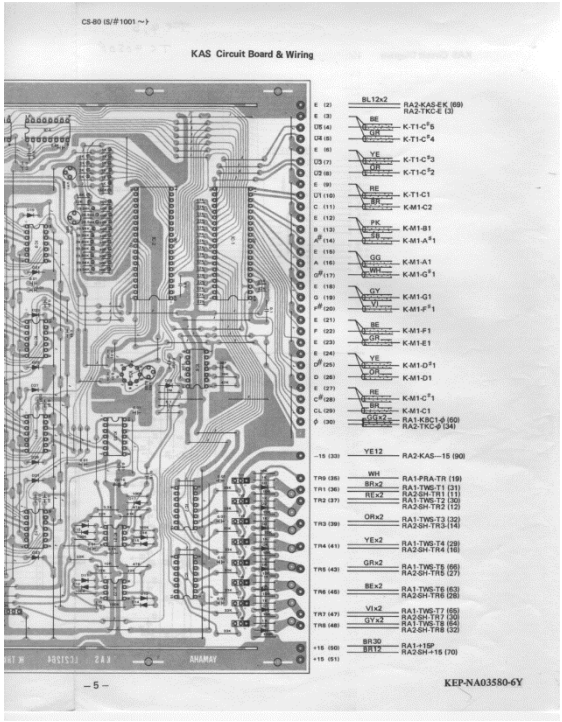
ABSTRACTION

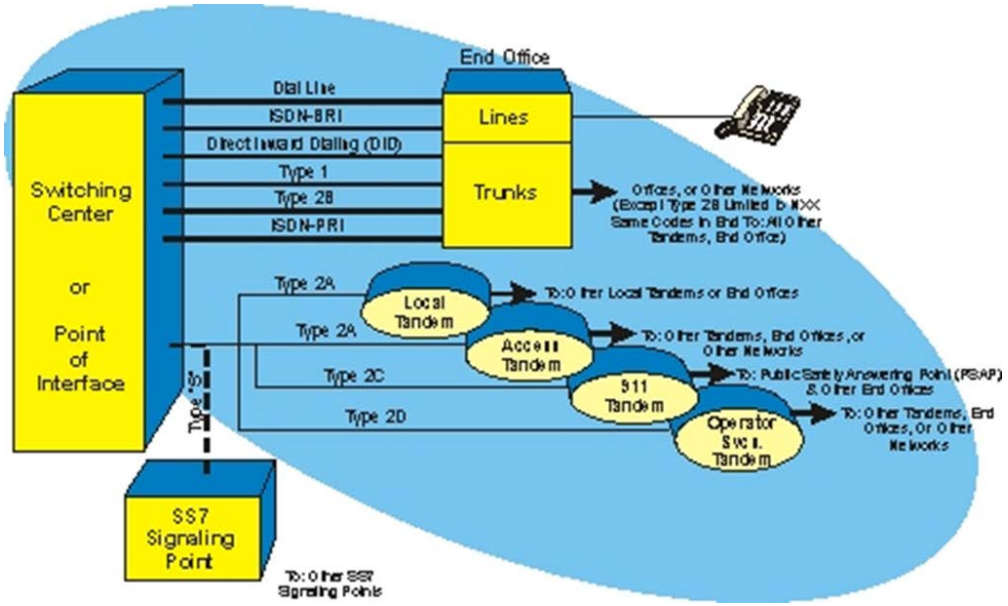


https://www.st.cs.uni-saarland.de/edu/se/2010/lecture-slides/07-SoftwareDesign_Handout1.pdf

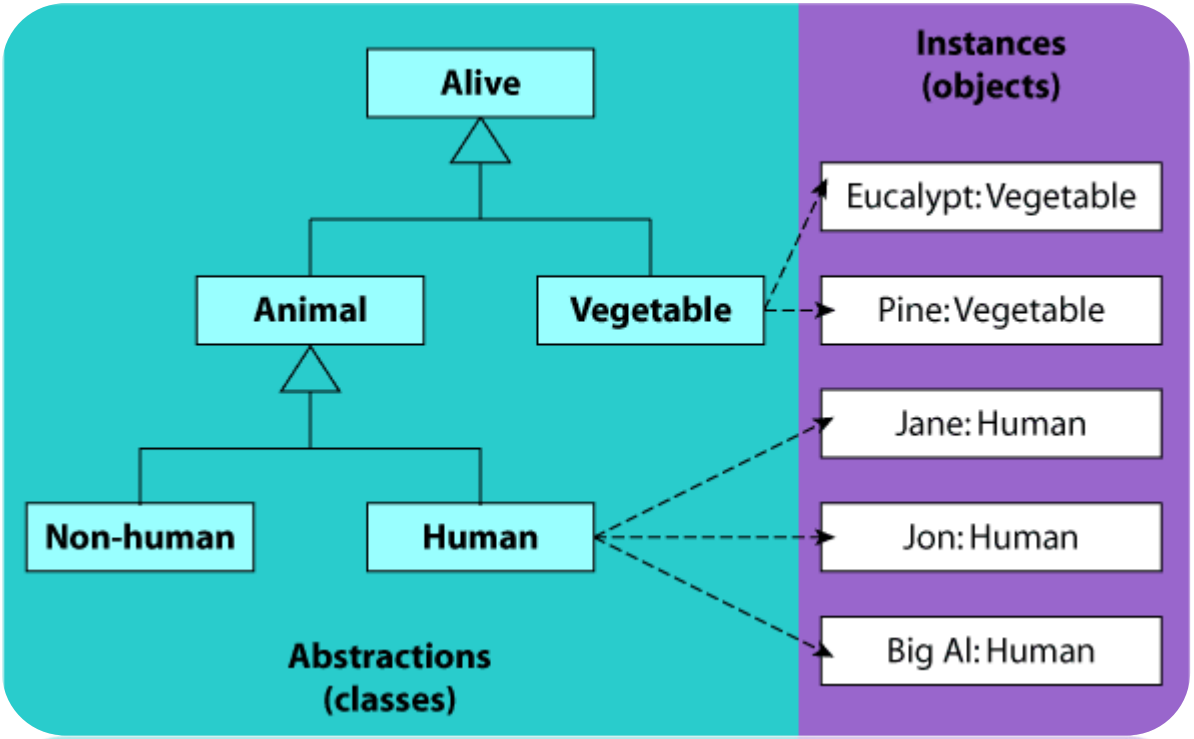
ABSTRACTION

ABSTRACTION



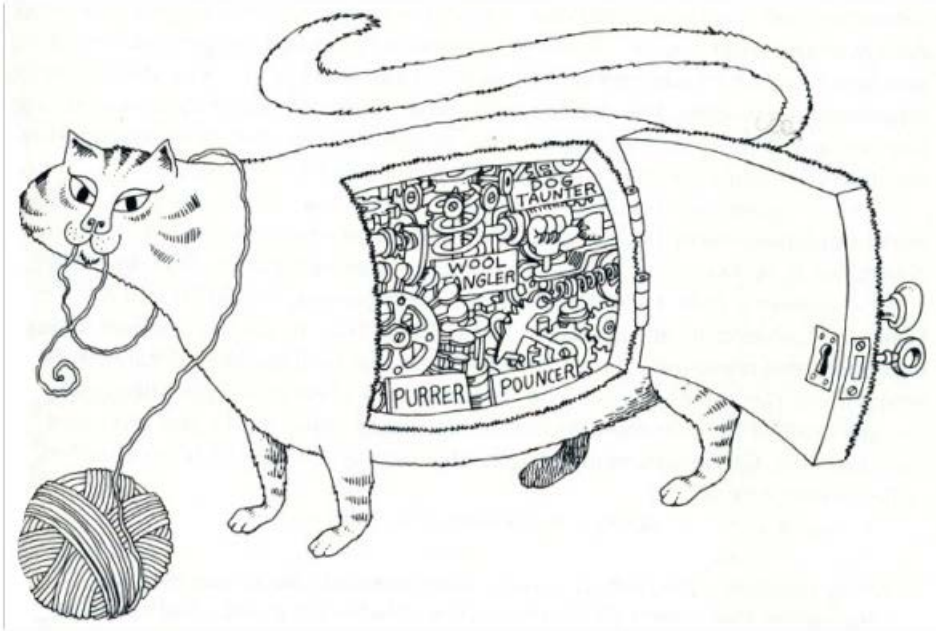


ABSTRACTION



http://www.csse.monash.edu.au/~cema/courses/CSE5910/lectureFiles/images/lect1b/abshierarchy.GIF

ENCAPSULATION



https://www.st.cs.uni-saarland.de/edu/se/2010/lecture-slides/07-SoftwareDesign_Handout1.pdf

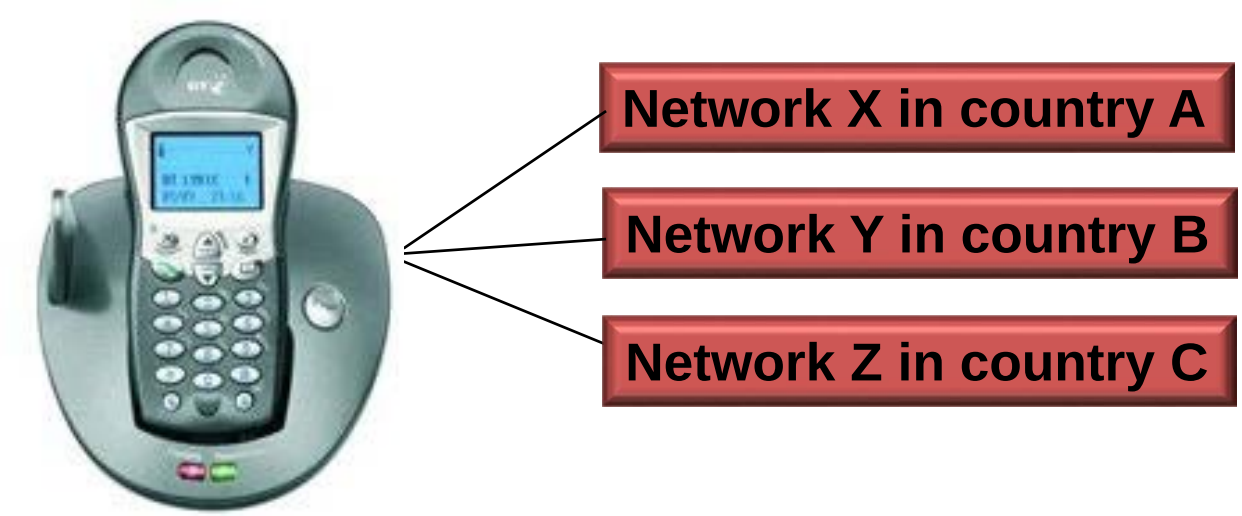
ENCAPSULATION

Analog signal

Digital signal

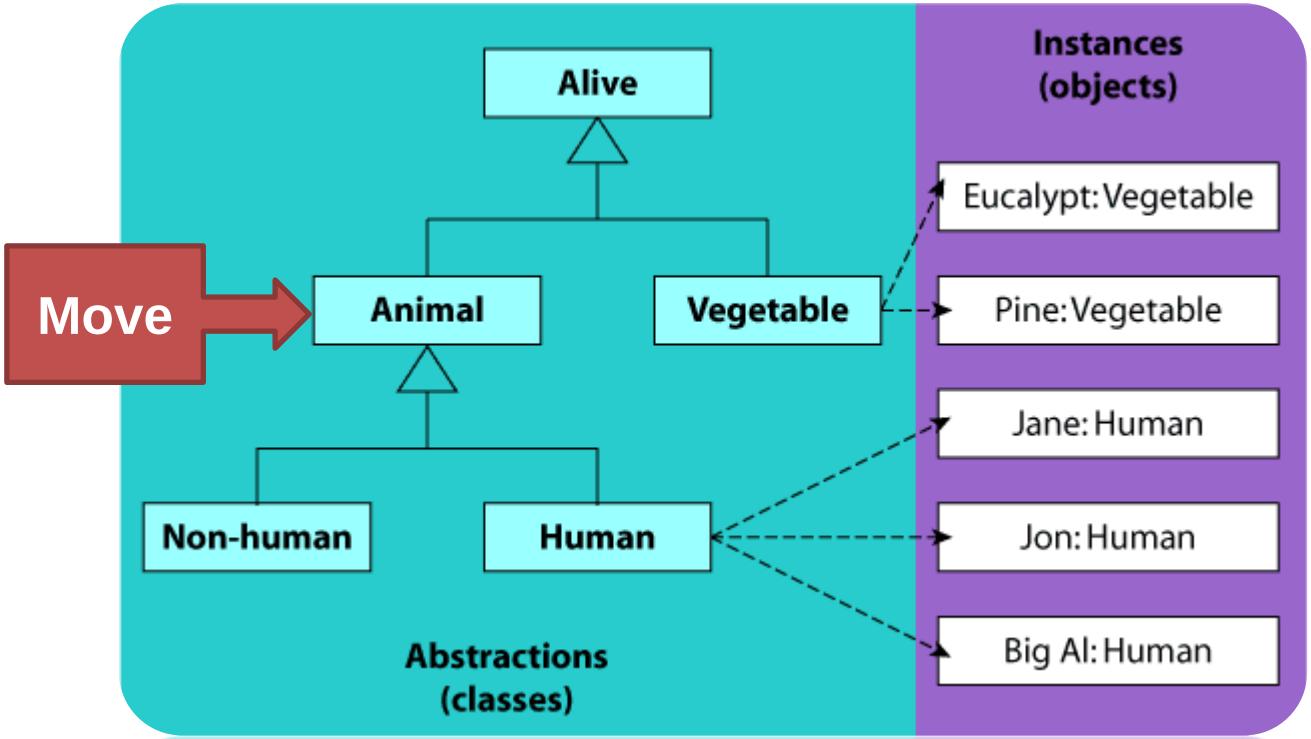
Digital with Encryption

POLYMORPHISM (OO ENCAPSULATION)



The ability to hide many different implementations behind a single interface

POLYMORPHISM



http://www.csse.monash.edu.au/~cema/courses/CSE5910/lectureFiles/images/lect1b/abshierarchy.GIF

MODULARITY, ABSTRACTION & ENCAPSULATION

Queen Mary University of London

Abstraction, modularity and encapsulation each help provide information hiding.

Queen Mary
University of London

LESSON SUMMARY

- Good design is all about sensible system decomposition
- Crucial to get the right balance between low coupling and high cohesion
- Crucial to model at the right level of abstraction