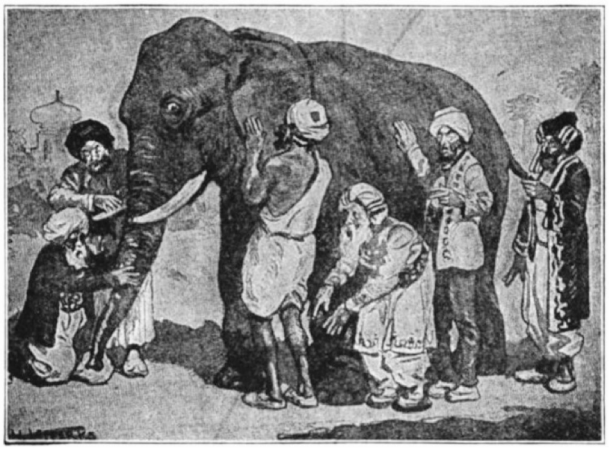
DA-IICT

ASE: Modeling Complexities

Dr Saurabh Tiwari

The Six Blind Men of Indostan

The Blind Men and the Elephant American poet John Godfrey Saxe (1816-1887) based the following poem on an Indian folktale.



The Six Blind Men of Indostan

It was six men of Indostan, To learning much inclined, Who went to see the elephant, (Though all of them were blind), That each by observation might satisfy his mind.

The first approached
the elephant,
And happening to fall
Against his broad and
sturdy side,
At once began to bawl:
"God bless me! But the
elephant
Is very like a wall!"

The second, feeling of the tusk,
Cried: "Ho! What have we here,
So very round and smooth and
sharp?
To me 'tis very clear,
This wonder of an elephant
Is very like a spear!“

3

The Six Blind Men of Indostan

The third approached the
animal,
And happening to take
The squirming trunk within his
hands,
Thus boldly up and spake:
"I see," quoth he, "the elephant
Is very like a snake!"

The fourth reached out an eager
hand,
And felt about the knee.
"What most this wondrous beast is
like is might plain," quoth he;
"'Tis clear enough the elephant
Is very like a tree.“

The fifth, who chanced to touch
the ear,
Said: "E'en the blindest man
Can tell what this resembles most:
Deny the fact who can,
This marvel of an elephant
Is very like a fan."

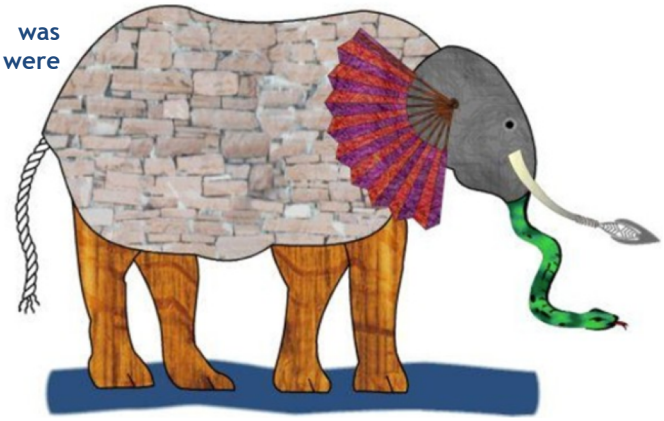
The sixth no sooner had begun
About the beast to grope,
Than seizing on the swinging
tail
That fell within his scope,
"I see," quoth he, "the elephant
Is very like a rope.“

4

The Six Blind Men of Indostan

And so these men of Indostan Disputed loud and long, Each in his own opinion Exceeding stiff and strong.

Though each was partly right, All were in the wrong.



5

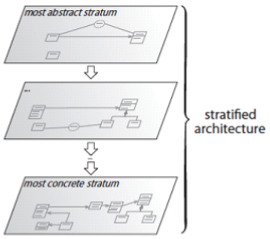
Handling Complexity

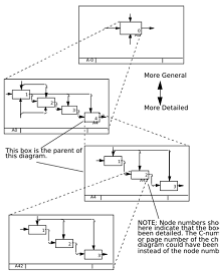
1. Abstraction
2. Decomposition
3. Modularity
4. Hierarchy
5. Set of Principles

6

Abstraction

- Define levels of abstractions
- Define models at these levels
- Define transformation of models from higher to lower levels
- Implement





Abstraction

- Abstraction allows us to ignore unessential details
- Two definitions for abstraction:
  - Abstraction is a *thought process* where ideas are distanced from objects
    - Abstraction as activity
  - Abstraction is the *resulting idea* of a thought process where an idea has been distanced from an object
    - Abstraction as entity
- Ideas can be expressed by models

Technique to deal with Complexity: Decomposition

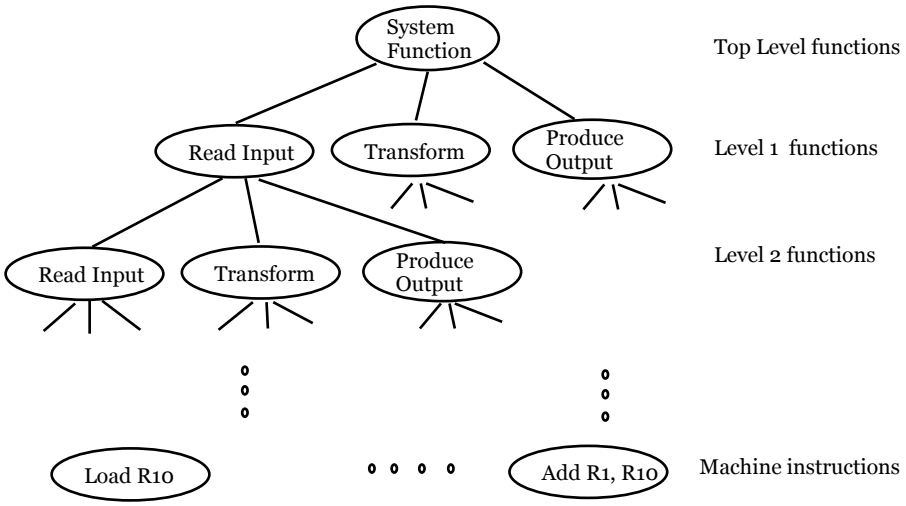
- A technique used to master complexity ("divide and conquer")
- Two major types of decomposition
  - Functional decomposition
  - Object-oriented decomposition
- Functional decomposition
  - The system is decomposed into modules
  - Each module is a major function in the application domain
  - Modules can be decomposed into smaller modules.

Decomposition (cont'd)

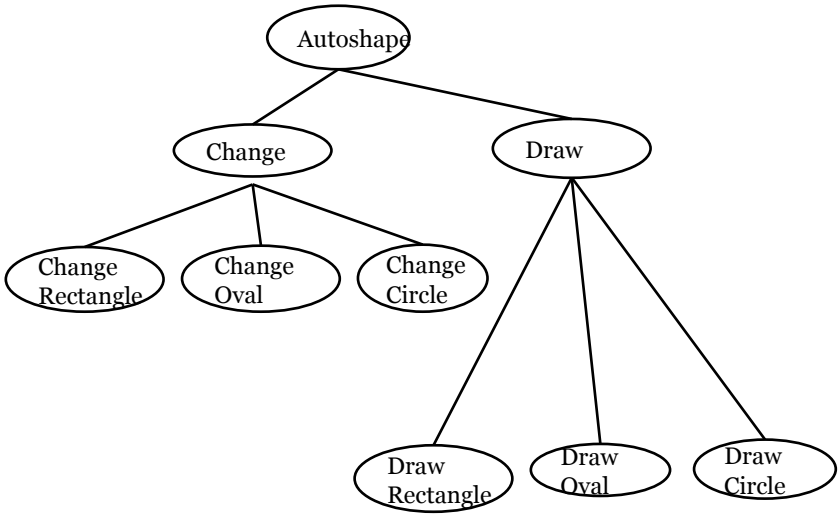
- Object-oriented decomposition
  - The system is decomposed into classes ("objects")
  - Each class is a major entity in the application domain
  - Classes can be decomposed into smaller classes
- Object-oriented vs. functional decomposition

Which decomposition is the right one?

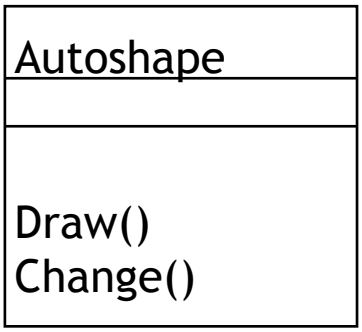
Functional Decomposition



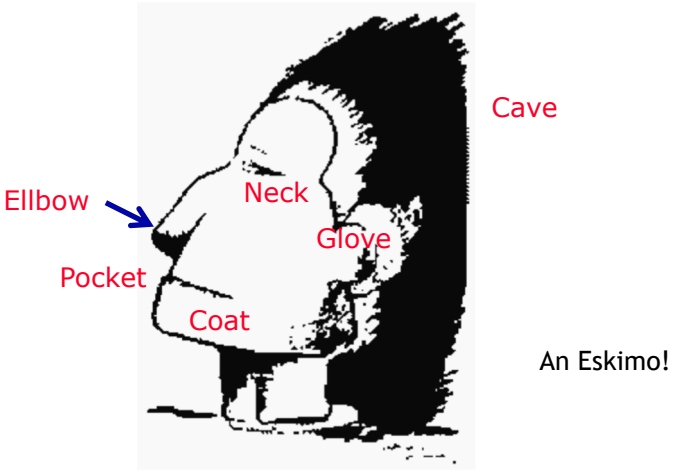
Functional Decomposition: Autoshape

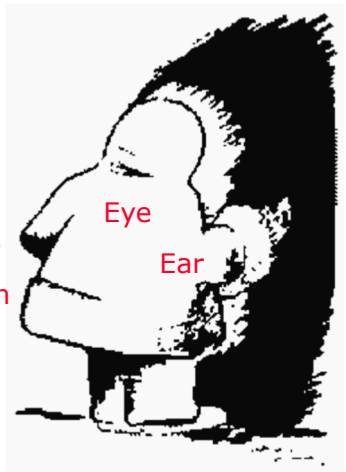


Object-Oriented View

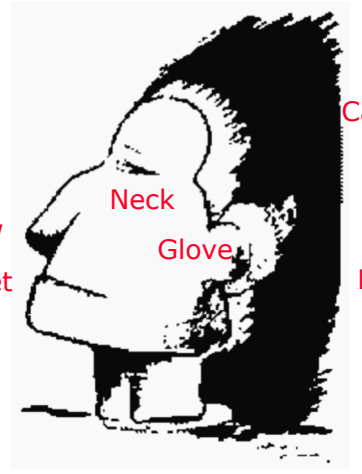


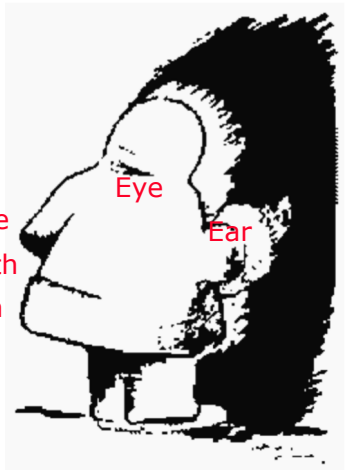
What is This?





A Face!





Depending on the purpose of the system different objects might be found: A nose is suddenly an elbow, hair is a cave, an ear turns out to be a glove.

Modularity

Modularization is the process of dividing a software system into multiple independent modules where each module works independently.

Advantages of Modularization:

- Easy to understand the system.
- System maintenance is easy.
- A module can be used many times as their requirements. No need to write it again and again.

Highly cohesive structure

- Loosely coupled
- Interactions through clearly defined

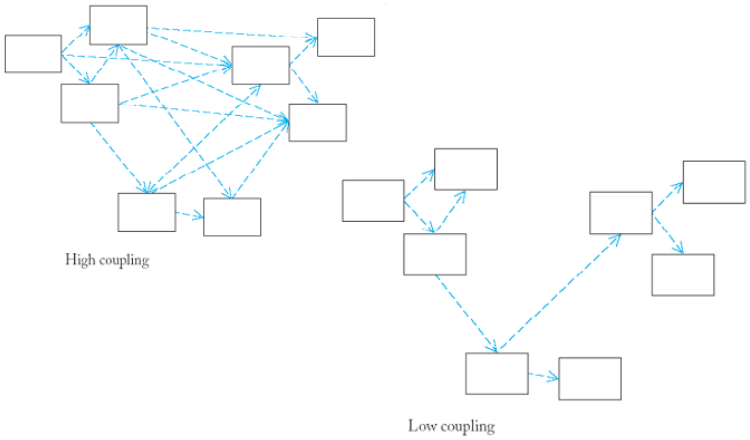
Modularity

Coupling:
Coupling is the measure of the degree of interdependence between the modules.

A good software will have low coupling.

Cohesion:
Cohesion is a measure of the degree to which the elements of the module are functionally related.
OR
degree to which all elements directed towards performing a single task are contained in the component.

A good software design will have high cohesion.



Hierarchies

We got abstractions and decomposition

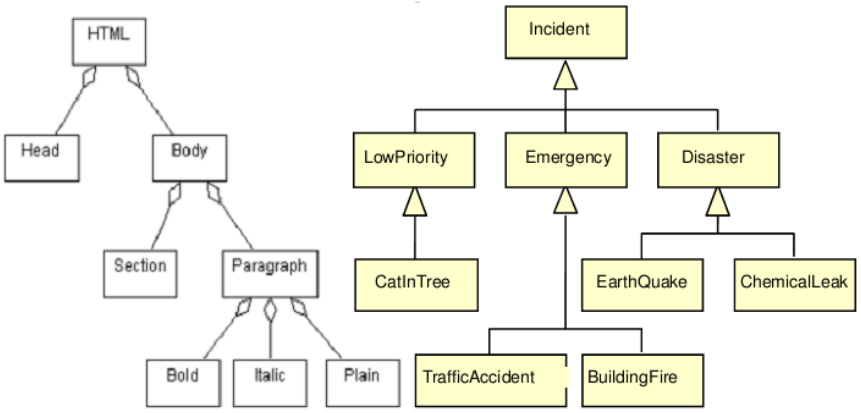
- This leads us to chunks (classes, objects) which we view with object model
- This should be modular

• Another way to deal with complexity is to provide simple relationships between the chunks

• One of the most important relationships is hierarchy

Two important hierarchies
- "Part of" hierarchy
- "Is-kind-of" hierarchy

20



Part-Of Kind-Of

21

Principles of S/W Development

Core Principles [David Hooker 1996]

- Ensure Usability
  - Produce to be consume
- Communication
- Simplicity
- Open to change
- Employ Reuse
- Feedback
- Think

22

Principles of S/W Development

Communication Principles

- Listen
- Prepare before you communicate
- Employ Facilitator
- Face-to-Face Communication
- Take notes and document decisions
- Strive for collaboration
- Move-on
- Strive for "win-win" situation

23

Principles of S/W Development

Planning Principles

- Understand and define the scope of the project
- Involve the customer
- Expect "change"
- Estimate rationally
- Have a RISK handling plan
- Be practical and methodological (W^5HH Principles by Boehm)
  - Why is the system being developed?
  - What will be done?
  - When will it be accomplished?
  - Who is responsible for what?
  - Where are they organizationally located
  - How will the job be done technically and managerially
  - How much of each resource is needed?

24

Principles of S/W Development

Analysis Modeling Principles

- Information domain of a problem must be understood and represented
- Define functional and non functional requirements clearly
- Define a complete and consistent set of requirements
- Analysis should help development

25

Principles of S/W Development

Design Principles

- Ensure traceability between analysis and design models
- Architecture comes first
- Data design is as good as functional design
- Clearly describe interfaces
- UID is critically important
- Cohesion and Coupling
- Open-Close Principle
- Dependency Inversion Principle
- The Liskov Substitution Principle
- Code-friendly designs
- Ensure efficient change management

26

Principles of S/W Development

Coding Principles

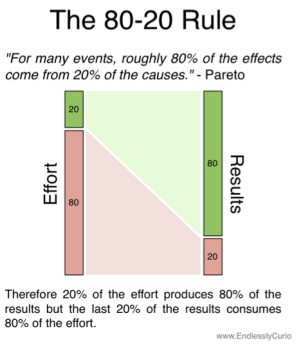
- Code as little as possible
- Code should assist efficient and effective testing
- Employ code refactoring
- Perform unit testing
- Keep program logic as simple as possible
- Use standard coding style
- Write code that is self documenting
- Employ peer reviews if needed

27

Principles of S/W Development

Testing Principles

- Tests to be traceable to the requirements
- Plan early for testing
- Exhaustive testing is not possible
- Testing from "in the small" to "in the large"
- Pareto principle applies in software testing



28

What is Modeling?

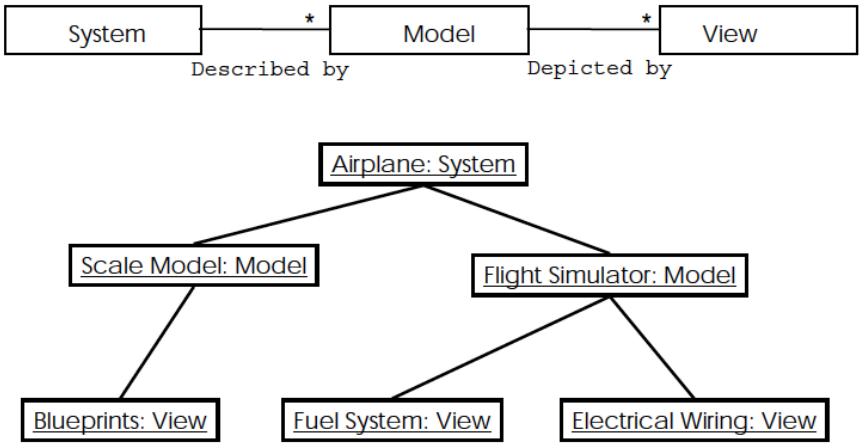
29

System, Models and Views

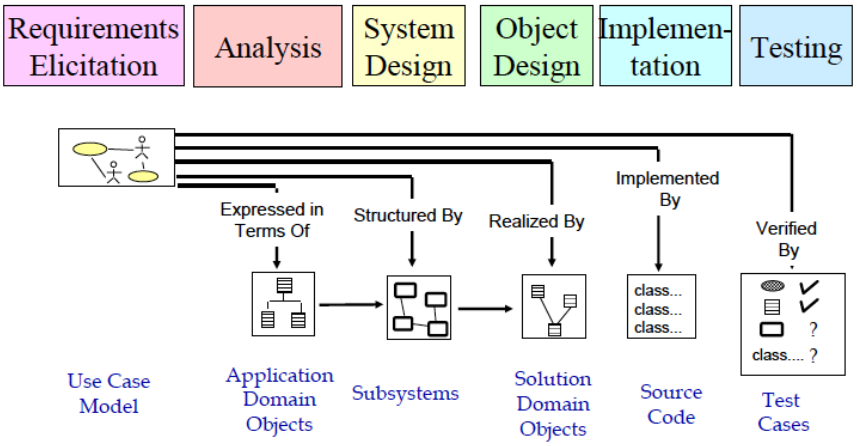
- A ***model*** is an abstraction describing a subset of a system/component
- A ***view*** depicts selected aspects of a system/component
- A ***notation*** is a set of graphical or textual rules for depicting a model
- Views and models of a single system typically overlap each other

Examples:
- System: Aircraft
- Models: All blueprints, electrical wiring, fuel system, Flight Controller

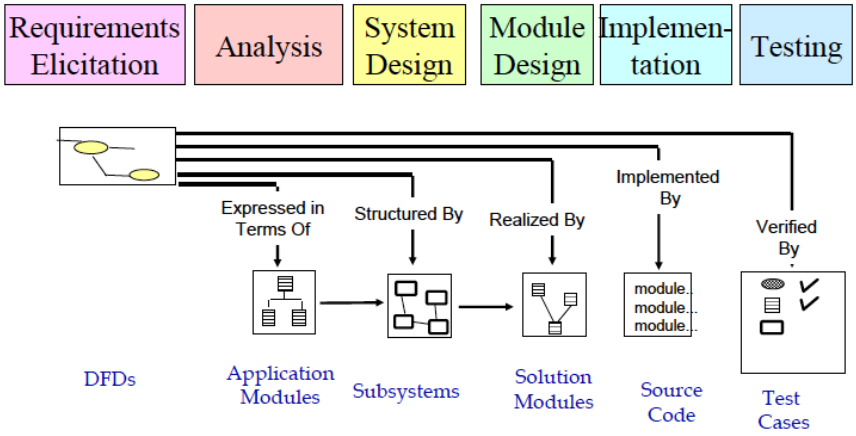
System, Models and Views



Software Life Cycle Activities (in OO Systems)



Software Life Cycle Activities (in Functional Systems)



What constitutes a good model?

- A model should
  - use a standard notation
  - be understandable by clients and users
  - lead software engineers to have insights about the system
  - provide abstraction
- Models are used:
  - to help create designs
  - to permit analysis and review of those designs.
  - as the core documentation describing the system.

Questions??