Unit-3 Design

3.1- Design Process

3.2- Design Concepts

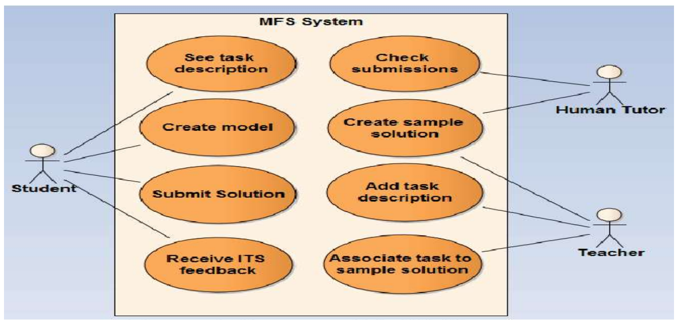
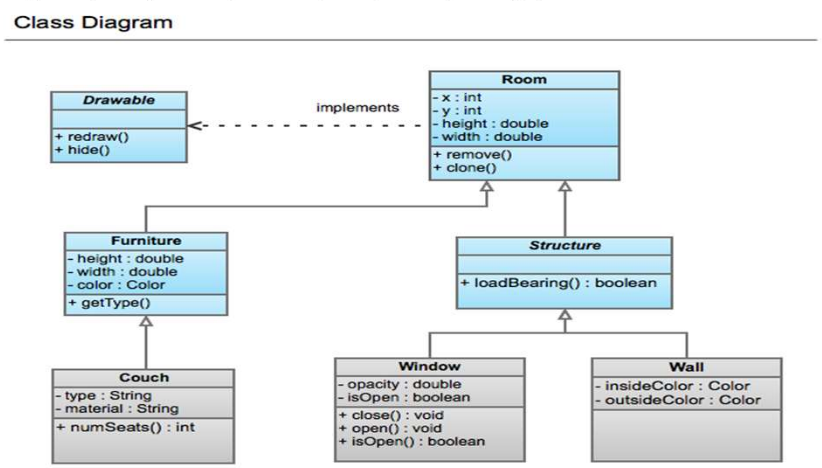
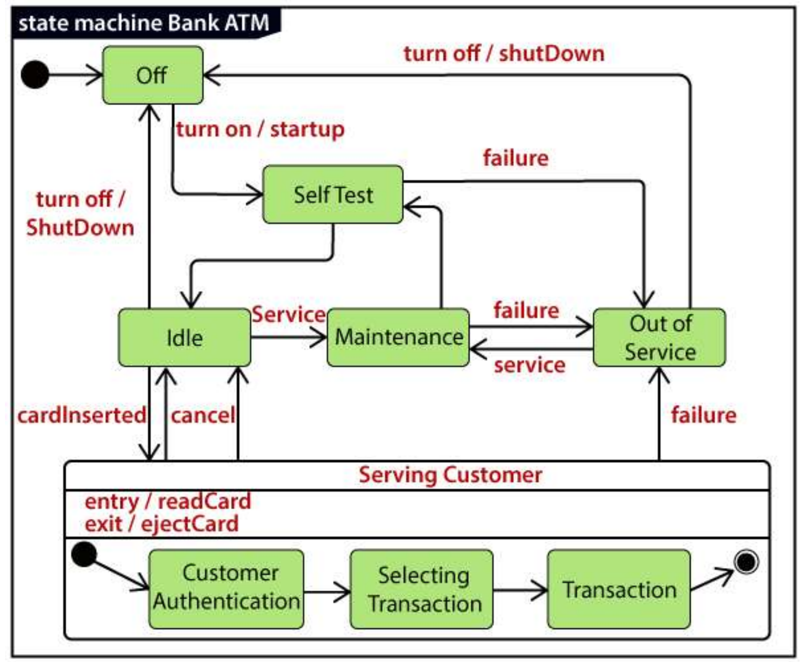


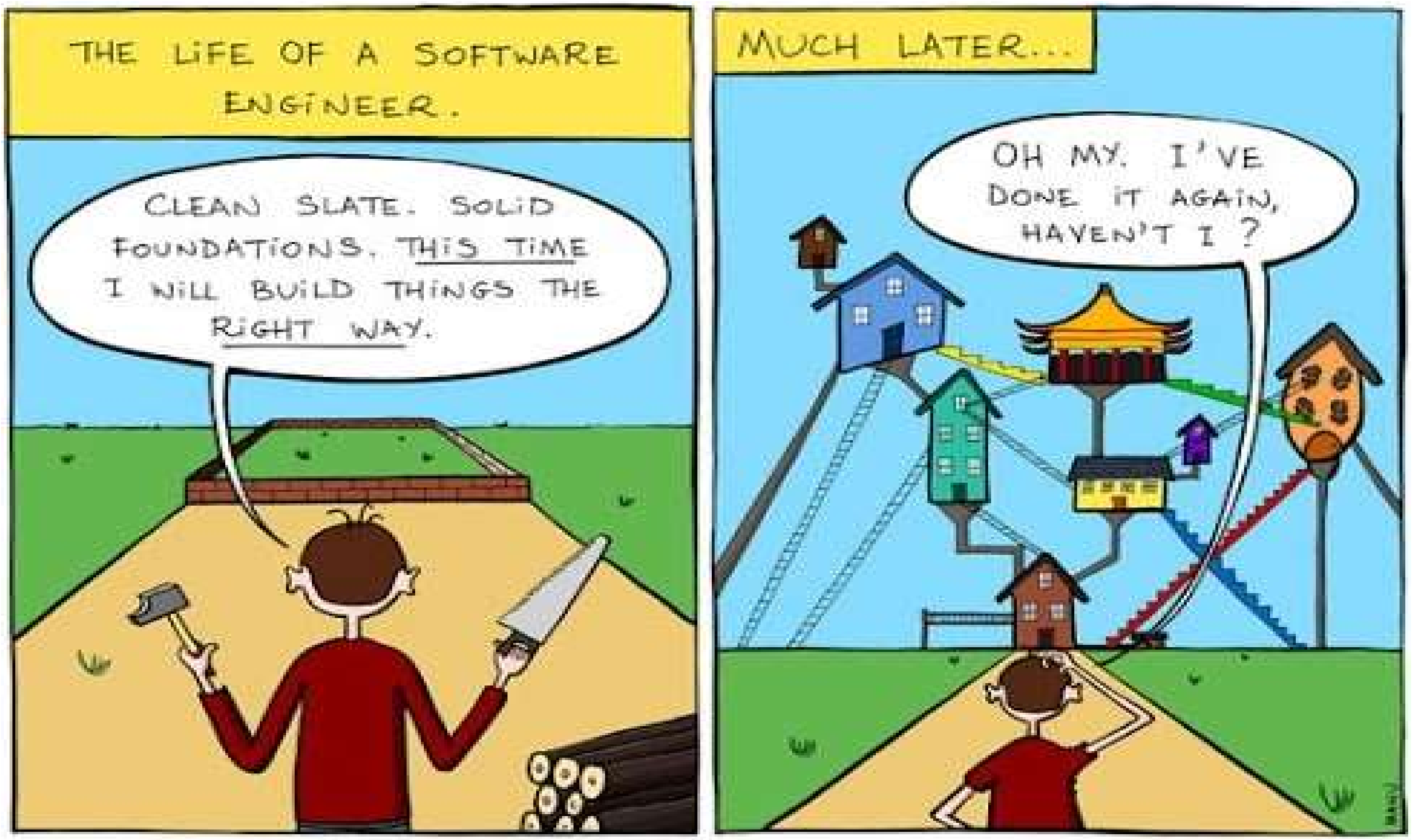
Figure 1. Use cases of the MFS system





Software Design

"NOW, REMEMBER... DON'T TELL DADDY YOU DESIGNED THE SOFTWARE HE USES AT WORK... I WANT HIM TO GET TO KNOW YOU FIRST!"


THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?

Design Process

- S/w design is an iterative process through which requirements are translated into a “blueprint” for constructing the s/w.
- As design iteration occur, subsequent refinement leads to design representation at much lower levels of abstraction

Goal of design process

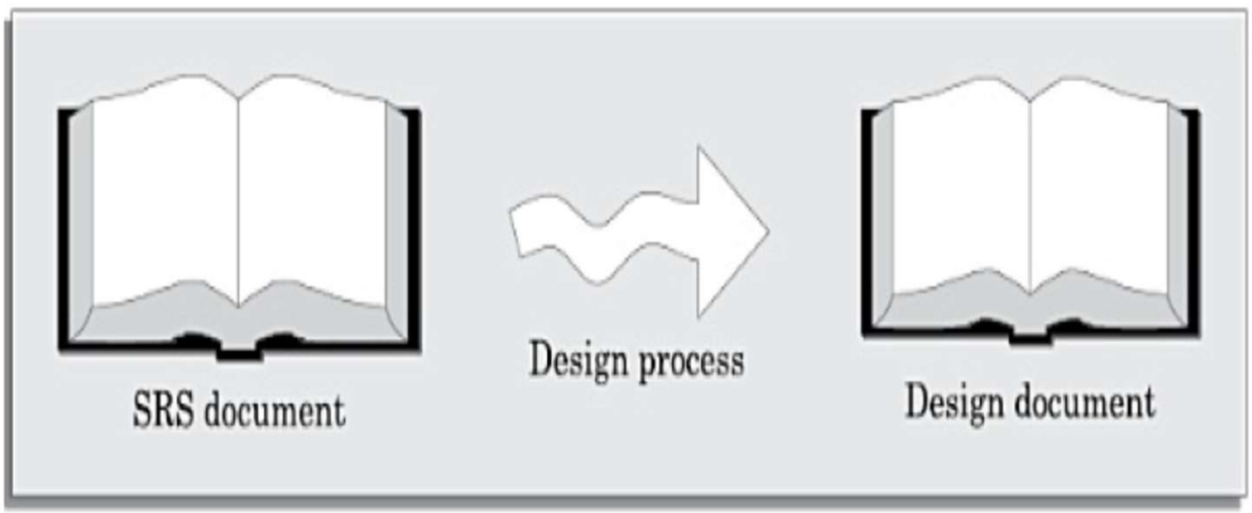
- The design must implement all of the explicit requirements contained in the analysis model, and it must accommodate all of the implicit requirements desired by the customer.

- The design must be a readable, understandable guide for those who generate code and for those who test and subsequently support the software.

- The design should provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective

SOFTWARE DESIGN

- The activities carried out during the design phase (called as design process) transform the SRS document into the design document.
- The design document produced at the end of the design phase should be implementable using a programming language in the subsequent (coding) phase.



Outcome of the Design Process

- The following items are designed and documented during the design phase.

1. Different modules required.
2. Relationships among modules.
3. Interfaces among different modules
4. Data structures of the individual modules
5. Algorithms required to implement the individual modules

Classification of Design Activities

- Depending on the order in which various design activities are performed, we can broadly classify them into two important stages.

1. Preliminary (or high-level) design, and
2. Detailed design.

- Through **high-level design**, a problem is decomposed into a set of modules. The control relationships among the modules are identified, and also the interfaces among various modules are identified.
- During **detailed design** each module is examined carefully to design its data structures and the algorithms.

Design Concepts

- **Abstraction**
  - When we consider a modular solution to any problem, many levels of abstraction can be posed.
  - At the highest level of abstraction, a solution is stated in broad terms
  - At lower levels of abstraction, a more detailed description of the solution is provided.
  - Problem-oriented terminology is coupled with implementation-oriented terminology in an effort to state a solution.
  - Finally, at the lowest level of abstraction, the solution is stated in a manner that can be directly implemented

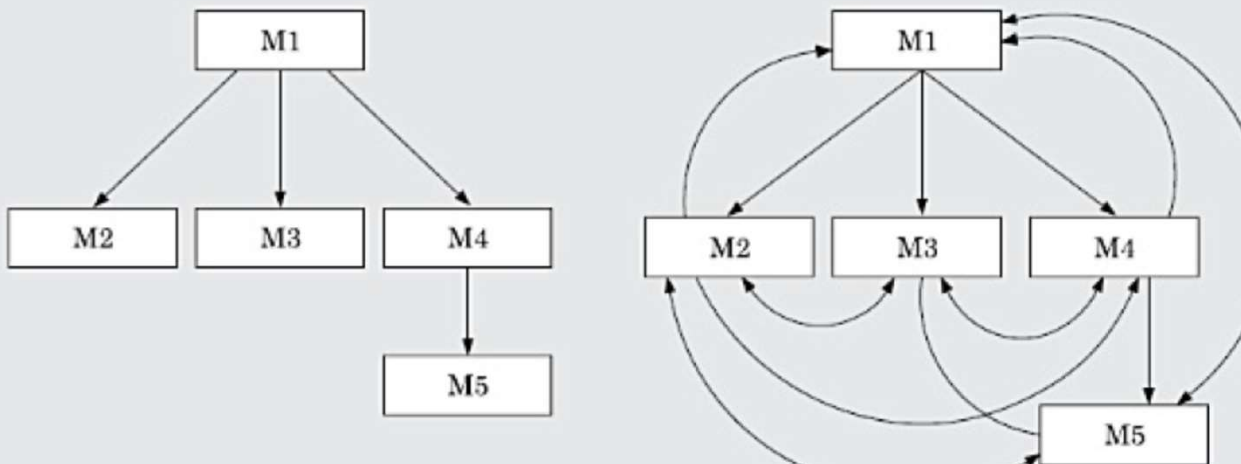
Design Concepts

- Refinement
  - refinement is a top-down design strategy
  - Refinement is actually a process of elaboration
  - Refinement causes the designer to elaborate on the original statement, providing more and more detail as each successive refinement (elaboration) occurs
  - Refinement helps the designer to expose low-level details as design progresses

Design Concepts

- **Modularity**
  - A modular design, in simple words, implies that the problem has been decomposed into a set of modules that have only limited interactions with each other.
  - Decomposition of a problem into modules facilitates taking advantage of the divide and conquer principle.
  - A software design with high cohesion and low coupling among modules is the effective problem decomposition. Such a design would lead to increased productivity during program development by bringing down the perceived problem complexity.

Design Concepts



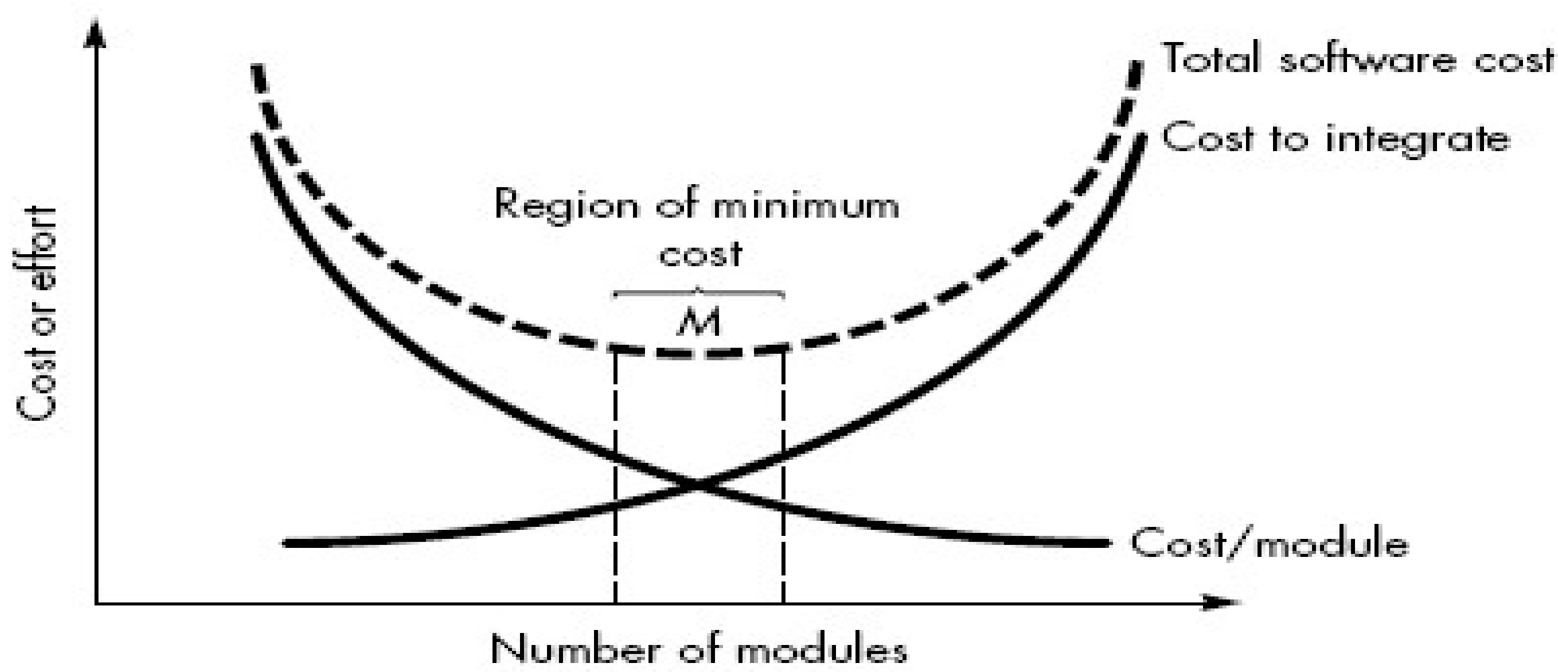
(a) A modular and hierarchical design

(b) A design solution exhibiting poor modularity and hierarchy

Design Concepts

- Architecture and design pattern embody modularity.
- Software is divided into separately named and addressable components, sometimes called modules, which are integrated to satisfy problem requirement.
- modularity is the single attribute of software that allows a program to be intellectually manageable
- It leads to a "divide and conquer" strategy. – it is easier to solve a complex problem when you break into a manageable pieces.
- Refer fig. that state that effort (cost) to develop an individual software module does decrease if total number of modules increase.
- However as the no. of modules grows, the effort (cost) associated with integrating the modules also grows.

Modularity and software cost



Design Concepts

- Undermodularity and overmodularity should be avoided. But how do we know the vicinity of M?
- We modularize a design so that development can be more easily planned.
- Software increments can be defined and delivered.
- Changes can be more easily accommodated.
- Testing and debugging can be conducted more efficiently and long-term maintained can be conducted without serious side effects.

Design Concepts

- Refactoring
  - refactoring is a reorganization technique that simplifies the design of a component without changing its function or behavior
  - When software is refactored, the existing design is examined for
    - redundancy, unused design elements, inefficient or unnecessary algorithms, poorly constructed or inappropriate data structures, or any other design failure that can be corrected

Design Concepts

- **Information Hiding**
  - The principle of information hiding suggests that modules be "characterized by design decisions that (each) hides from all others modules.“
  - In other words, modules should be specified and designed so that information (algorithm and data) contained within a module is inaccessible to other modules that have no need for such information.
  - The intent of information hiding is to hide the details of data structure and procedural processing behind a module interface.

Design Concepts

- Control Hierarchy
  - system is a hierarchy of components
  - To design such a hierarchy
    - Top-down and bottom up
  - A top-down design approach starts by identifying the major components of the system, decomposing them into their lower-level components
  - A bottom-up design approach starts with designing the most basic or primitive components and proceeds to higher-level components that use these lower-level components