Design Engineering Concepts

- **Software Design**
  - It is a process to transform stakeholder's requirements, business needs, and technical considerations into some suitable form, which helps the programmer to add details needed for the actual computer system implementation.

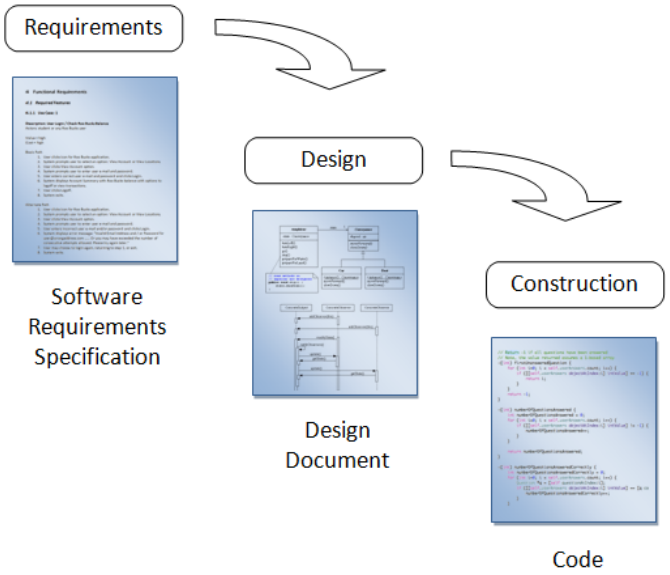

Figure 1

Software Design between Software Requirements Specification and the Construction Phase

(Retrieved from http://sce2.umkc.edu/BIT/burrise/pl/design/SoftwareDesignAdvanced.ppt)

- **Software Design Objectives/Properties**
  - Correctness
  - Verifiability
  - Completeness
  - Traceability
  - Efficiency
  - Simplicity

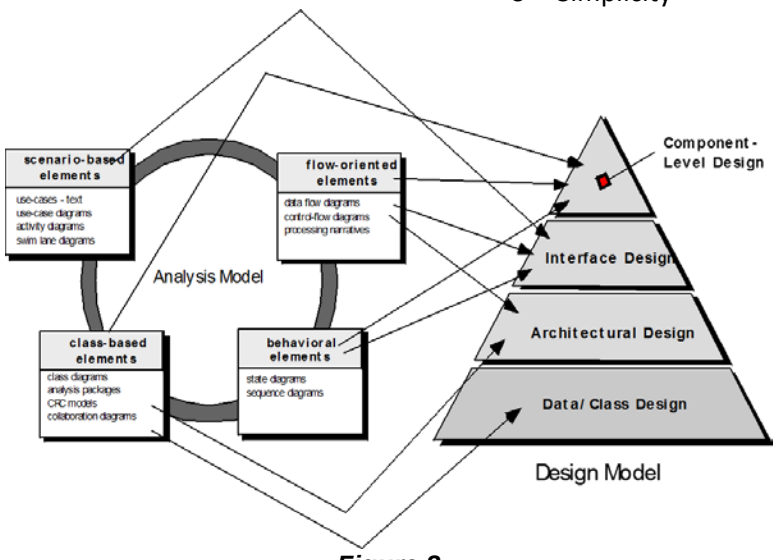

Figure 2

Translating the analysis model into a software design

(Retrieved from: http://files.pgimehsana.webnode.in/200003639-7ecf37fc8f/Unit%202%20Principles%20of%20Software%20Engineering%20AND%20UNIT%203.docx*)*

- **Architectural Design Process**
  - System Structuring
  - Control Modelling
  - Modular Decomposition

- **Key Architectural Concepts**
  - **Three (3) canonical building blocks**
    - Components
    - Connector
    - Configuration
  - **Subsystem**
  - **Module**

- **Abstraction**

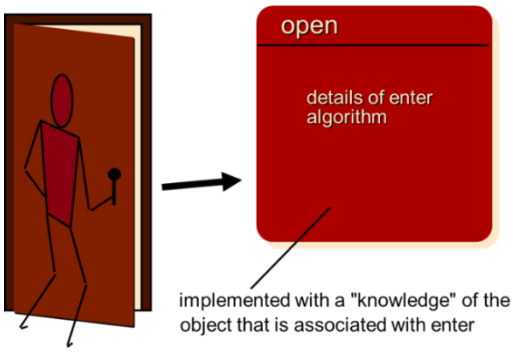

Figure 3

Procedural/functional abstraction

(Retrieved from: http://files.pgimehsana.webnode.in/200003639-7ecf37fc8f/Unit%202%20Principles%20of%20Software%20Engineering%20AND%20UNIT%203.docx)

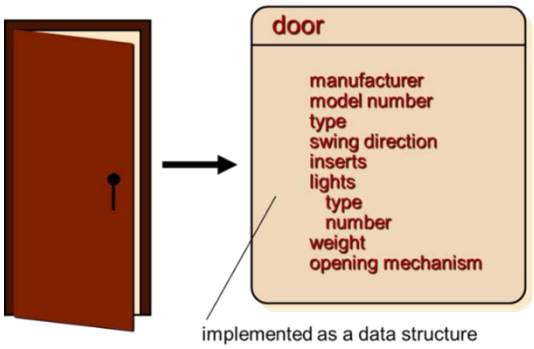

Figure 4

Data abstraction

(Retrieved from: http://files.pgimehsana.webnode.in/200003639-7ecf37fc8f/Unit%202%20Principles%20of%20Software%20Engineering%20AND%20UNIT%203.docx)

- **Modularization**

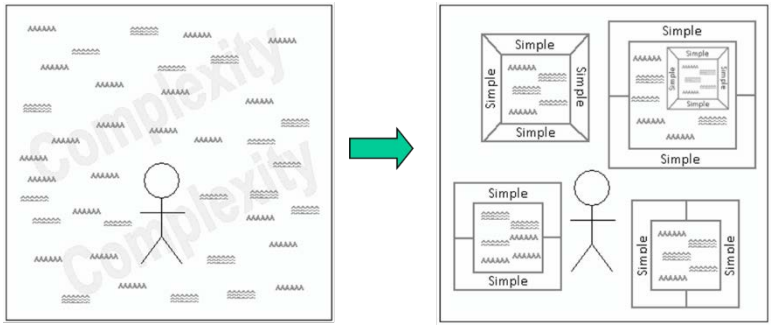

Figure 5

(Retrieved from http://sce2.umkc.edu/BIT/burrise/pl/design/SoftwareDesignAdvanced.ppt)

- **Advantages of Modularization**
  - Modular systems are easier to understand and explain
  - Modular systems are easier to document
  - Programming individual modules is easier
  - Testing and debugging individual modules is easier
  - Bugs are easier to isolate and understand
  - Well-composed modules are more reusable; also, a good module should be easy to extract from one program and insert into another

- **Information Hiding**

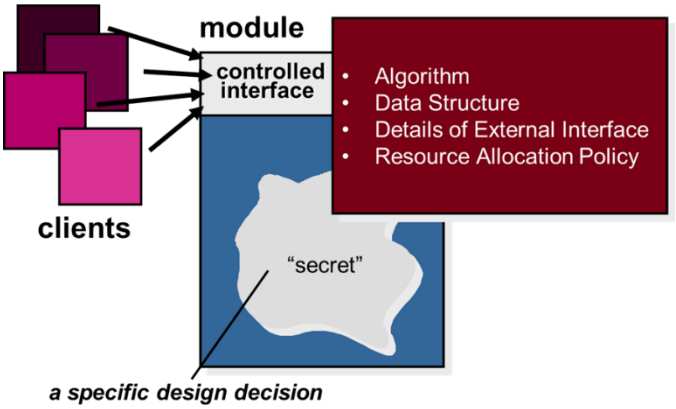

Figure 6

(Retrieved from: *http://files.pgimehsana.webnode.in/200003639-7ecf37fc8f/Unit%202%20Principles%20of%20Software%20Engineering%20AND%20UNIT%203.docx*)

- **Functional Independence**
  - **Reasons why functional independence is a key to any good design**
    - Error isolation
    - Scope of reuse
    - Understandability
  - **Two Important Criteria**

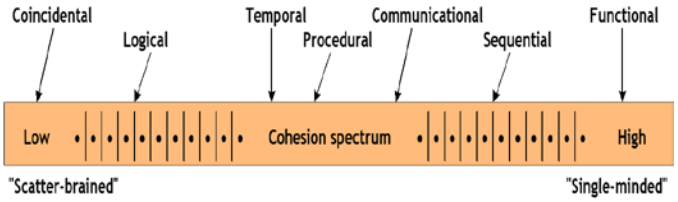

Figure 7
Cohesion and Its Classification

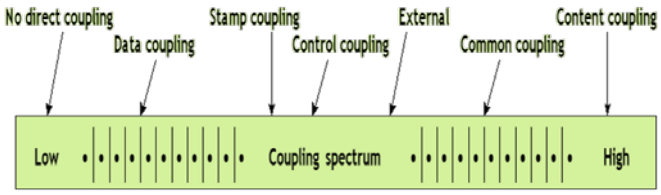

Figure 8
Coupling and Its Classification

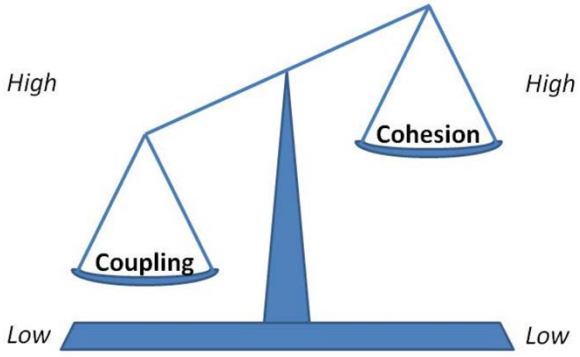

Figure 9
Relationship between Coupling and Cohesion
(Retrieved from http://sce2.umkc.edu/BIT/burrise/pl/design/SoftwareDesignAdvanced.ppt)

- **Benefits of High Cohesion and Low Coupling**
  - Modules are easier to read and understand
  - Modules are easier to modify
  - There is an increased potential for reuse
  - Modules are easier to develop and test

- **Stepwise Refinement**

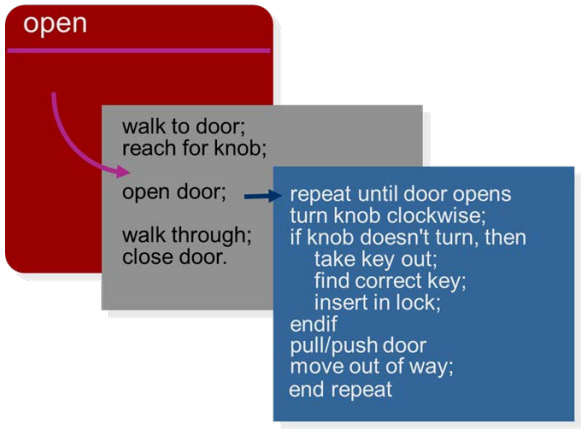

Figure 10

(Retrieved from: http://files.pgimehsana.webnode.in/200003639-7ecf37fc8f/Unit%202%20Principles%20of%20Software%20Engineering%20AND%20UNIT%203.docx)

Software Architecture

- **Software Architecture**
  - It is a description of how the subsystems and components of a software system is organized and the relationships between them

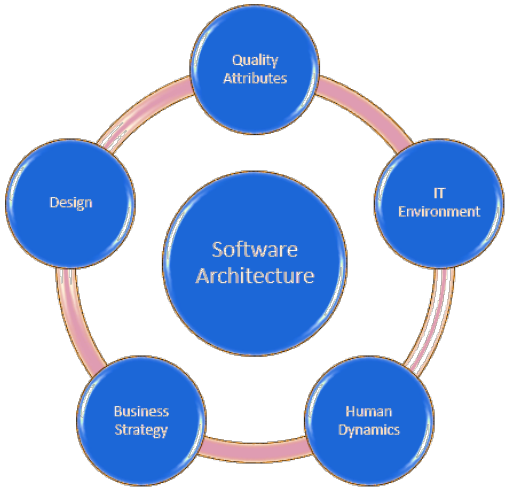

Figure 11
Software Architecture

- **Advantages**
  - Stakeholder Communication
  - System Analysis
  - Large Scale Reuse

- **Aspect of Architectural Design Representation**
  - Structural properties
  - Extra-functional properties
  - Families of related systems

- **Aspect of Architectural Design Representation**
  - Structural Model
  - Framework Model
  - Dynamic Model
  - Process Model
  - Functional Model

References:

Agarwal, B. B, S. P Tayal, and M Gupta (2010). *Software Engineering & Testing*. 1st ed. Sudbury, Mass.: Jones and Bartlett.

Architectural Design (2000) Retrieved from: http://sunset.usc.edu/~neno/cs477_2003/February11.ppt

Coupling (computer programming) (n. d.) Retrieved from: https://en.wikipedia.org/wiki/Coupling_(computer_programming)

Pfleeger, S. L., & Atlee, J. M. (2009). *Software engineering: Theory and practice* (4th ed.). Boston, MA, United States: Pearson Prentice Hall.

Pressman, RS. (2010). *Software engineering: a practitioner's approach (7th ed.)*. McGraw-Hill/Higher Education

Software Architecture & Design Introduction (n. d.) Retrieved from: https://www.tutorialspoint.com/software_architecture_design/introduction.htm

Software Design (n. d.) Retrieved from: http://sce2.umkc.edu/BIT/burrise/pl/design/SoftwareDesignAdvanced.ppt

Software Design (n. d.) Retrieved from: https://www.slideshare.net/syedmuhammadhammad/software-design-13430869

Sommerville, I. (2010). *Software engineering* (9th ed.). Boston: Addison-Wesley Educational Publishers.

Unit 2 Principles of Software Engineering; and Requirements Modeling. (n. d.) Retrieved from: http://files.pgimehsana.webnode.in/200003639-7ecf37fc8f/Unit%202%20Principles%20of%20Software%20Engineering%20AND%20UNIT%203.docx