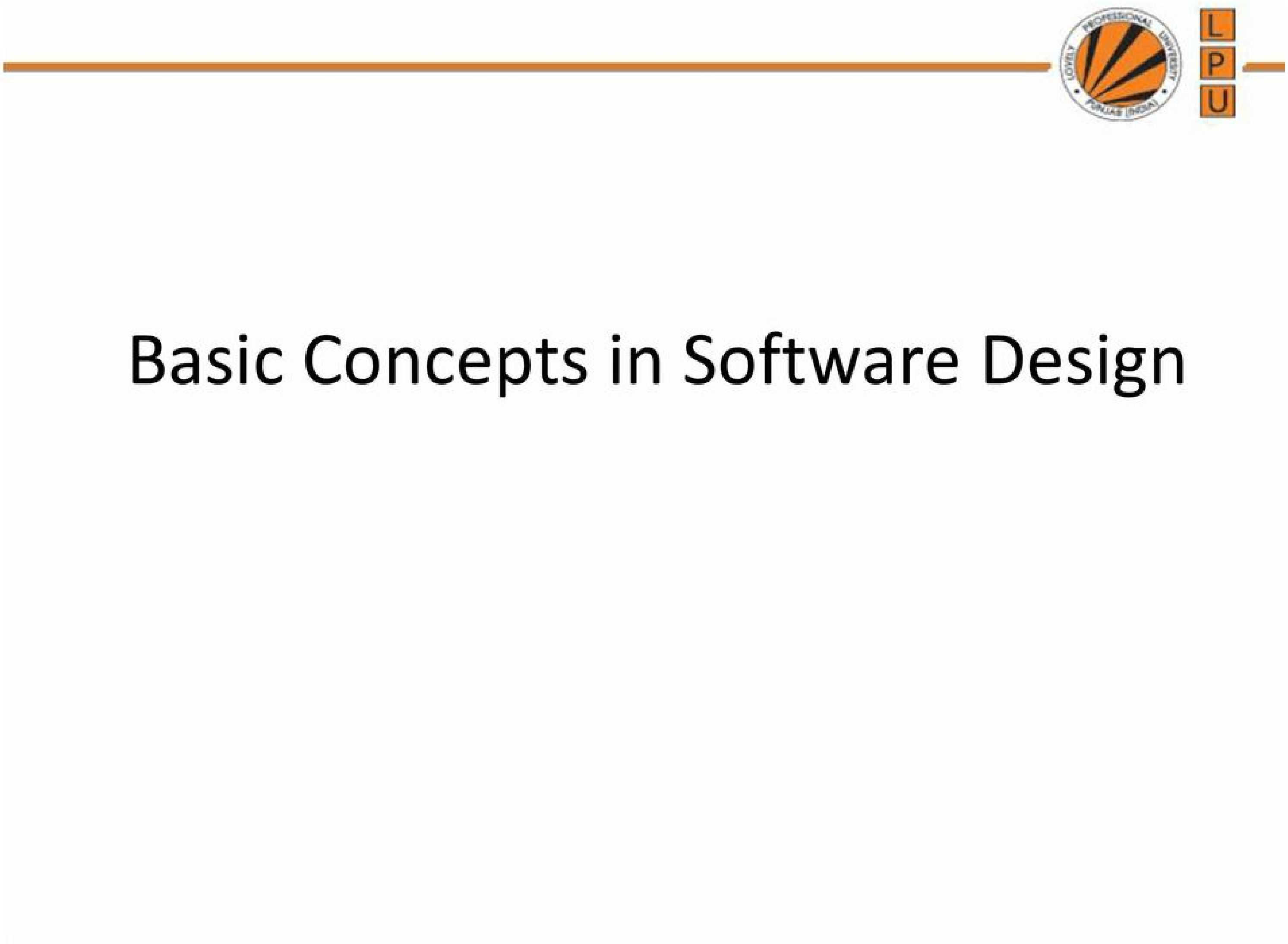


Basic Concepts in Software Design

1 / 24

Basic Concepts in Software Design

Published by Rafe Allen

Modified over 5 years ago

Embed

Download presentation

Similar presentations

Presentation on theme: "Basic Concepts in Software Design"— Presentation transcript:

1 **Basic Concepts in Software Design**

2 **Basic Concepts in Software Design**

Software design and its activities

Software design deals with transforming the customer requirements, as described in the SRS document, into a form (a set of documents) that is suitable for implementation in a programming language

3 **Basic Concepts in Software Design**

Design activities can be broadly classified into two important parts:

•Preliminary (or high-level) design and

• Detailed design.

4 **Basic Concepts in Software Design**

High-level design means identification of different modules and the control relationships among them and the definition of the interfaces among these modules.

The outcome of high-level design is called the program structure or software architecture.

A popular way is to use a tree-like diagram called the structure chart to represent the control hierarchy in a high-level design.

5 **Basic Concepts in Software Design**

During detailed design, the data structure and the algorithms of the different modules are designed.

The outcome of the detailed design stage is usually known as the module-specification document.

6 **Difference between analysis and design**

The aim of analysis is to understand the problem with a view to eliminate any deficiencies in the requirement specification such as incompleteness, inconsistencies, etc.

The model which we are trying to build may be or may not be ready.

The aim of design is to produce a model that will provide a seamless transition to the coding phase, i.e. once the requirements are analyzed and found to be satisfactory, a design model is created which can be easily implemented..

7 **Items developed during the software design phase**

Different modules required to implement the design solution.

Control relationship among the identified modules.

Ch:8 **Design Concepts**

•S.W Design should have following quality attribute:

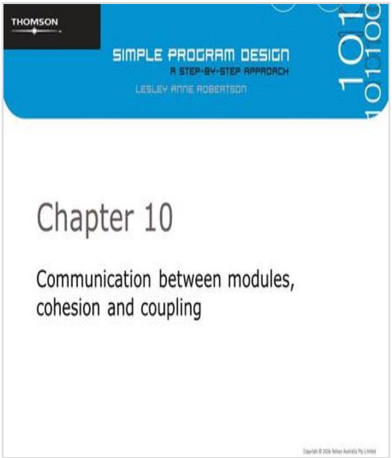
–Functionality

–Usability

–Reliability

–Performance

–Supportability (extensibility, adaptability, serviceability)



THOMSON

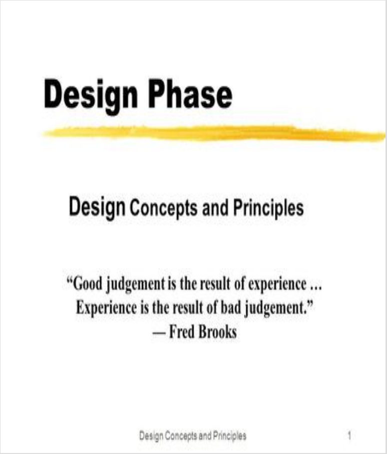
SIMPLE PROGRAM DESIGN

A STEP-BY-STEP APPROACH

LESLEY ANNE ROBERTSON

Chapter 10

Communication between modules, cohesion and coupling

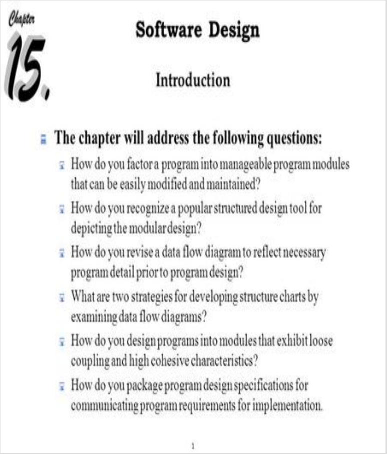


Design Phase

Design Concepts and Principles

"Good judgement is the result of experience … Experience is the result of bad judgement."
— Fred Brooks

Design Concepts and Principles 1



Chapter 15.

Software Design

Introduction

- The chapter will address the following questions:
  - How do you factor a program into manageable program modules that can be easily modified and maintained?
  - How do you recognize a popular structured design tool for depicting the modular design?
  - How do you revise a data flow diagram to reflect necessary program detail prior to program design?
  - What are two strategies for developing structure charts by examining data flow diagrams?
  - How do you design programs into modules that exhibit loose coupling and high cohesive characteristics?
  - How do you package program design specifications for communicating program requirements for implementation.

1

Data structures of the individual modules.

8 Characteristics of a good software design

Correctness:A good design should correctly implement all the functionalities identified in the SRS document.
• Understandability:A good design is easily understandable.
Efficiency:It should be efficient.
• Maintainability:It should be easily amenable to change.

9 Features of a design document

It should use consistent and meaningful names for various design components.
The design should be modular. The term modularity means that it should use a cleanly decomposed set of modules.
It should neatly arrange the modules in a hierarchy, e.g. in a tree-like diagram.

10 Software Design Issues

Cohesion
Most researchers and engineers agree that a good software design implies clean decomposition of the problem into modules, and the neat arrangement of these modules in a hierarchy.
The primary characteristics of neat module decomposition are high cohesion and low coupling.

11 Cohesion Cohesion is a measure of functional strength of a module.

A module having high cohesion and low coupling is said to be functionally independent of other modules.
By the term functional independence, we mean that a cohesive module performs a single task or function.
A functionally independent module has minimal interaction with other modules.

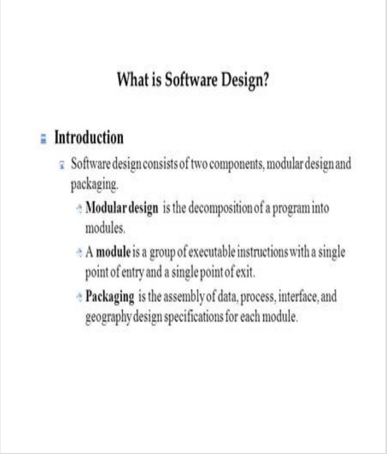
12 Classification of cohesion

13 Coincidental cohesion

A module is said to have coincidental cohesion,
if it performs a set of tasks that relate to each other very loosely if at all.
In this case, the module contains a random collection of functions.
For example, in a transaction processing system (TPS), the get-input, print-error, and summarize-members functions are grouped into one module. The grouping does not have any relevance to the structure of the problem.

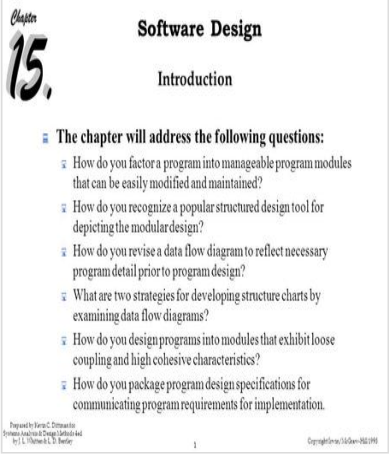
14 Coincidental cohesion

A module that only has coincidental cohesion is one supporting tasks that have no meaningful relationship to one another.
Page-Jones gives, as an example, a module (not necessarily one that can be implemented using software!) supporting the following tasks.



What is Software Design?

- Introduction
  - Software design consists of two components, modular design and packaging.
    - **Modular design** is the decomposition of a program into modules.
    - A **module** is a group of executable instructions with a single point of entry and a single point of exit.
    - **Packaging** is the assembly of data, process, interface, and geography design specifications for each module.



Chapter 15.

Software Design

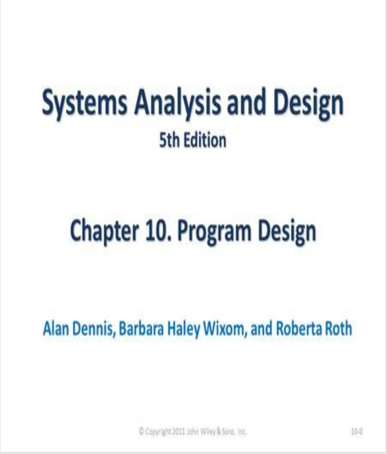
Introduction

- The chapter will address the following questions:
  - How do you factor a program into manageable program modules that can be easily modified and maintained?
  - How do you recognize a popular structured design tool for depicting the modular design?
  - How do you revise a data flow diagram to reflect necessary program detail prior to program design?
  - What are two strategies for developing structure charts by examining data flow diagrams?
  - How do you design programs into modules that exhibit loose coupling and high cohesive characteristics?
  - How do you package program design specifications for communicating program requirements for implementation.

Prepared by Kevin C. Dittman for Systems Analysis & Design Methods 4ed by J. L. Whitten & L. D. Bentley

1

Copyright Irwin/McGraw-Hill 1998



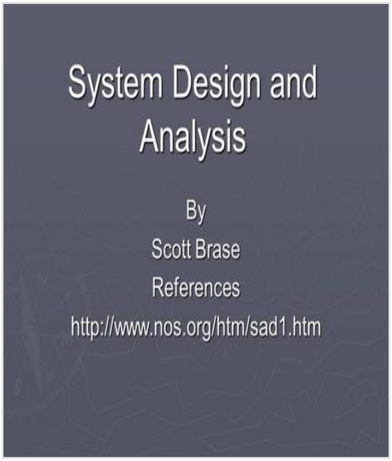
Systems Analysis and Design
5th Edition

Chapter 10. Program Design

Alan Dennis, Barbara Haley Wixom, and Roberta Roth

© Copyright 2011 John Wiley & Sons, Inc.

10-0



System Design and Analysis

By
Scott Brase
References
http://www.nos.org/htm/sad1.htm

15 **Logical cohesion: A module is said to be logically cohesive, if all** elements of the module perform similar operations.

An example of logical cohesion is the case where a set of print functions generating different output reports are arranged into a single module.

16 Logical cohesion:

Again, quoting Page-Jones: ``A logically cohesive module is one whose elements contribute to activities of the same general category in which the activity or activities to be executed are selected from outside the module.”

Keeping this definition in mind, consider the following example. Someone contemplating a journey might compile the following list: .

17 Logical cohesion:

Go by Car
Go by Train
Go by Boat
Go by Plane

18 Temporal Cohesion

A temporally cohesive module is one supporting tasks that are all related in time.

Ex1:
Turn off TV
Brush Teeth

Ex2:A module whose name is ``Do All Startup Activities,'' or ``Do All Shutdown Activities,'' might only have "temporal cohesion."

19 Procedural Cohesion

A module with (only) procedural cohesion is one supporting different and possibly unrelated activities, in which control passes from one activity to the next.

This is a bit better than ``temporal cohesion,'' since we know that there's a fixed ``linear ordering'' of the activities.

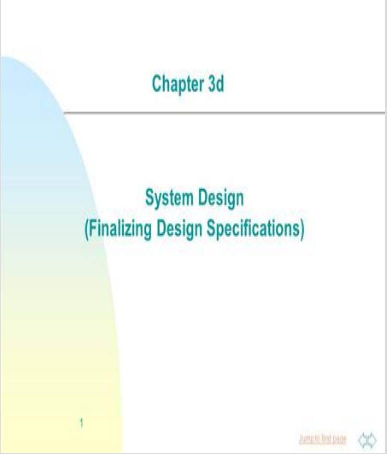
20 **Communicational Cohesion**

A module exhibits communicational cohesion if all the activities it supports use the same input or output data - or access and modify the same part of a data structure.

Find Title of Book
Find Price of Book
Find Publisher of Book
Find Author of Book

21 Sequential Cohesion

Again, quoting Page-Jones: ``A sequentially cohesive module is one whose elements are involved in activities such that output data from one activity serves as input data to the next.”



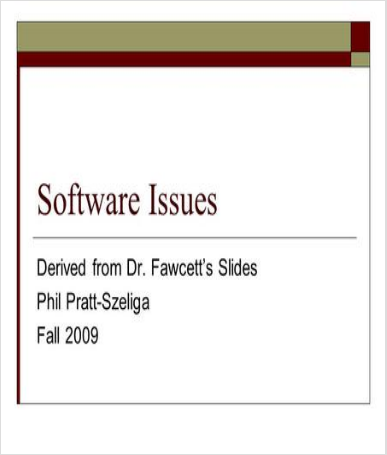
Chapter 3d

System Design
(Finalizing Design Specifications)

1

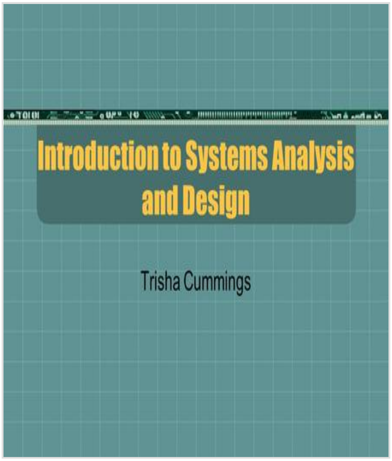
Chapter 1

Program Design



Software Issues

Derived from Dr. Fawcett's Slides
Phil Pratt-Szeliga
Fall 2009



Introduction to Systems Analysis and Design

Trisha Cummings

Clean Car Body
Fill in Holes in Car
Sand Car Body
Apply Primer

22 Functional Cohesion

A module exhibits ``functional cohesion'' if it supports activities needed for the execution for one and only one problem-related task.

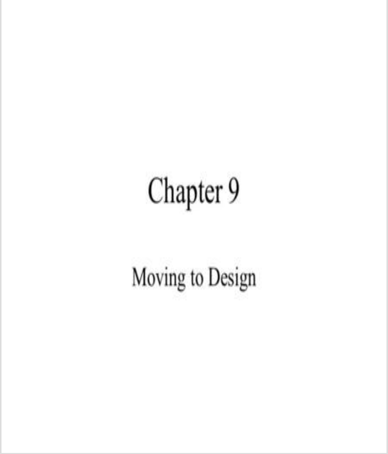
23 **Software Design Procedural Cohesion Temporal Cohesion**

Procedural Cohesion occurs in modules whose instructions although accomplish different tasks yet have been combined because there is a specific order in which the tasks are to be completed.
Temporal Cohesion
Module exhibits temporal cohesion when it contains tasks that are related by the fact that all tasks must be executed in the same time-span.

24 **Software Design Functional Cohesion Sequential Cohesion**

A and B are part of a single functional task. This is very good reason for them to be contained in the same procedure.
Sequential Cohesion
Module A outputs some data which forms the input to B. This is the reason for them to be contained in the same procedure.

Download ppt "Basic Concepts in Software Design"


Chapter 9

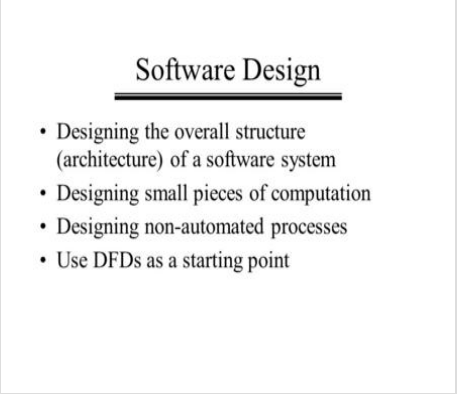
Moving to Design

Similar presentations

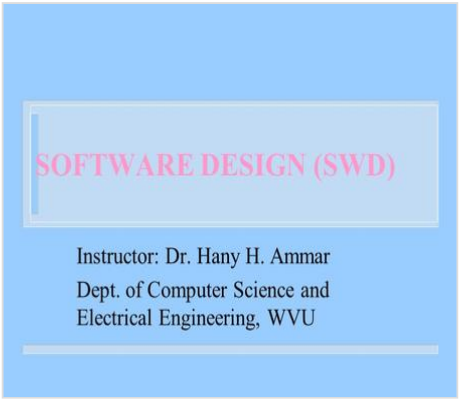
9

Program Design

Simple Program Design
Third Edition
A Step-by-Step Approach

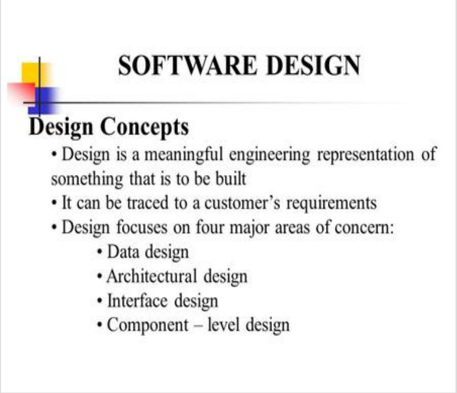

Software Design

- Designing the overall structure (architecture) of a software system
- Designing small pieces of computation
- Designing non-automated processes
- Use DFDs as a starting point


SOFTWARE DESIGN (SWD)

Instructor: Dr. Hany H. Ammar
Dept. of Computer Science and Electrical Engineering, WVU

SOFTWARE DESIGN



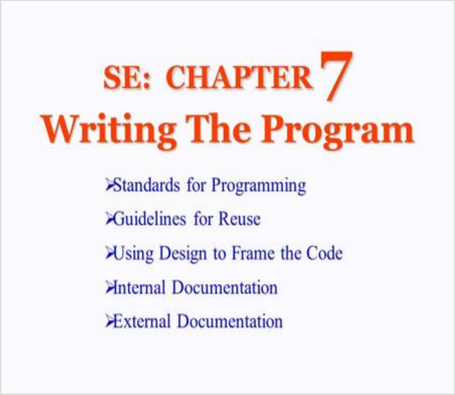
SOFTWARE DESIGN

Design Concepts

- Design is a meaningful engineering representation of something that is to be built
- It can be traced to a customer's requirements
- Design focuses on four major areas of concern:
  - Data design
  - Architectural design
  - Interface design
  - Component – level design

Software Design

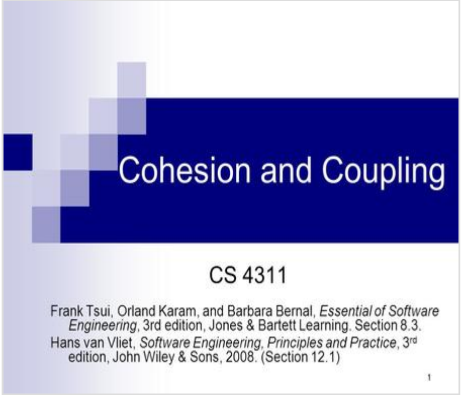
Deriving a solution which satisfies software requirements



SE: CHAPTER 7

Writing The Program

- ➢Standards for Programming
- ➢Guidelines for Reuse
- ➢Using Design to Frame the Code
- ➢Internal Documentation
- ➢External Documentation



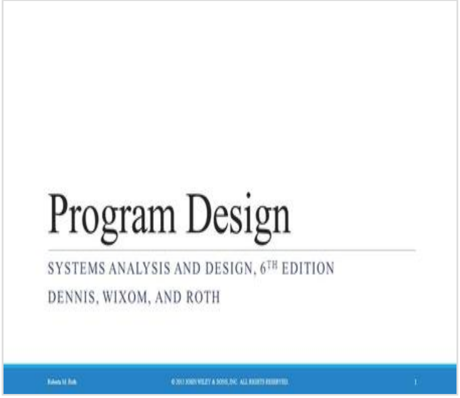
Cohesion and Coupling

CS 4311

Frank Tsui, Orland Karam, and Barbara Bernal, *Essential of Software Engineering*, 3rd edition, Jones & Bartett Learning. Section 8.3.

Hans van Vliet, *Software Engineering, Principles and Practice*, 3rd edition, John Wiley & Sons, 2008. (Section 12.1)

1



Program Design

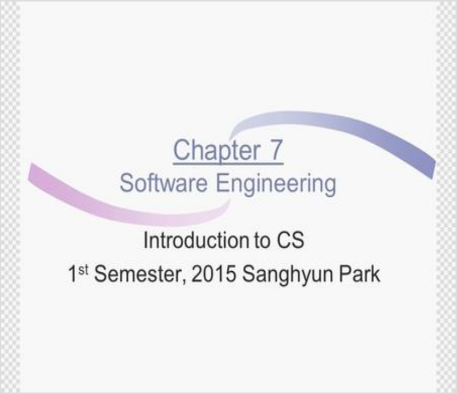
SYSTEMS ANALYSIS AND DESIGN, 6TH EDITION

DENNIS, WIXOM, AND ROTH

Roberta M. Roth © 2015 JOHN WILEY & SONS, INC. ALL RIGHTS RESERVED. 1

Coupling & Cohesion

CMSC 201- Fall '11



Chapter 7

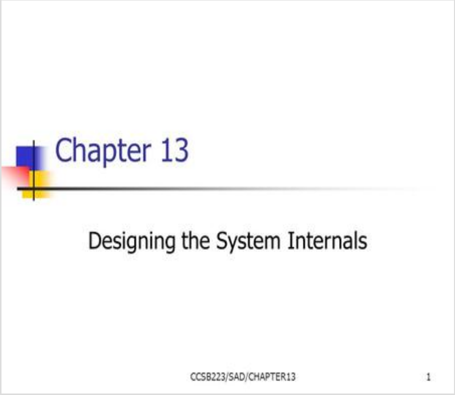
Software Engineering

Introduction to CS

1st Semester, 2015 Sanghyun Park

Design Concepts

By Deepika Chaudhary



Chapter 13

Designing the System Internals

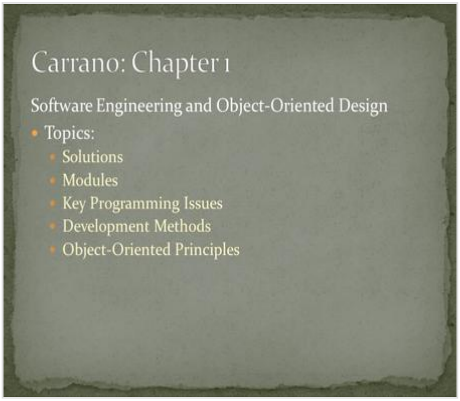
CCSB223/SAD/CHAPTER13 1

CMPT 275

High Level Design Phase

Modularization

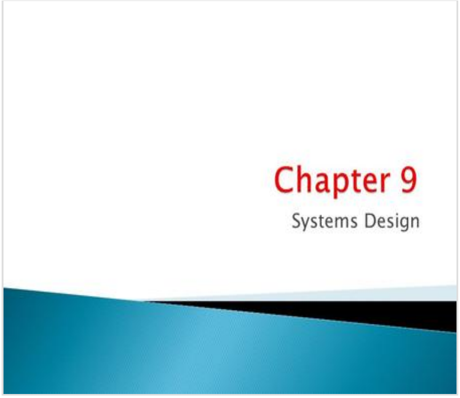
1



Carrano: Chapter 1

Software Engineering and Object-Oriented Design

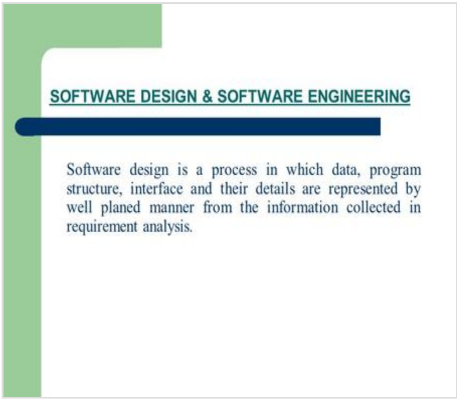
- Topics:
  - Solutions
  - Modules
  - Key Programming Issues
  - Development Methods
  - Object-Oriented Principles


Chapter 9

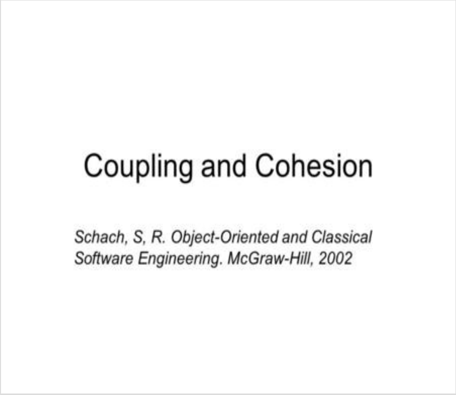
Systems Design

Systems Development Lifecycle

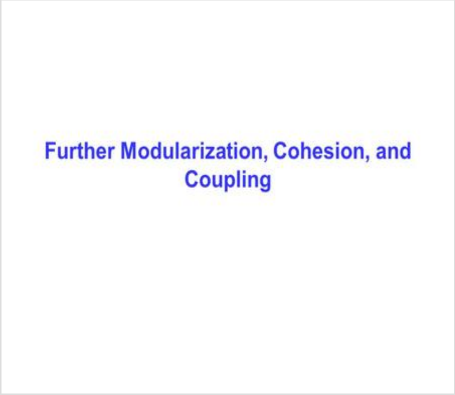
Design CS 470 – Software Engineering I Sheldon X. Liang, PH.D.

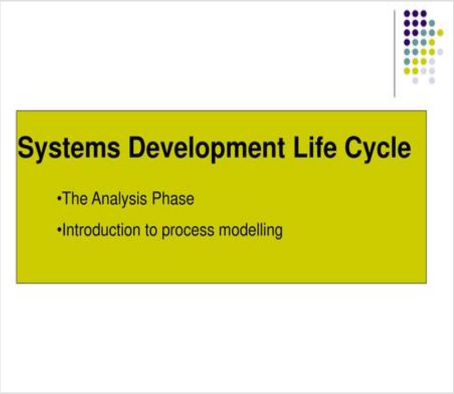

SOFTWARE DESIGN & SOFTWARE ENGINEERING

Software design is a process in which data, program structure, interface and their details are represented by well planed manner from the information collected in requirement analysis.


Coupling and Cohesion

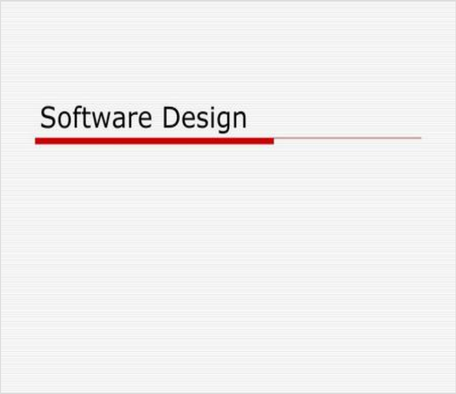
Schach, S, R. Object-Oriented and Classical Software Engineering. McGraw-Hill, 2002

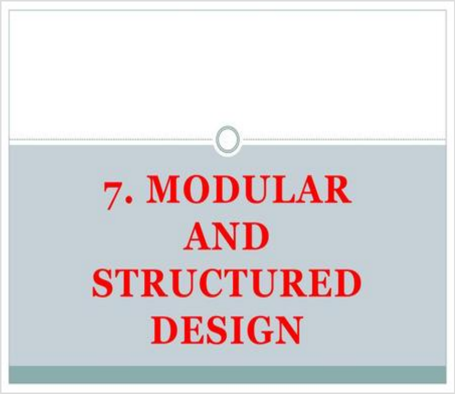

Further Modularization, Cohesion, and Coupling


Systems Development Life Cycle

•The Analysis Phase

•Introduction to process modelling


Software Design


7. MODULAR AND STRUCTURED DESIGN