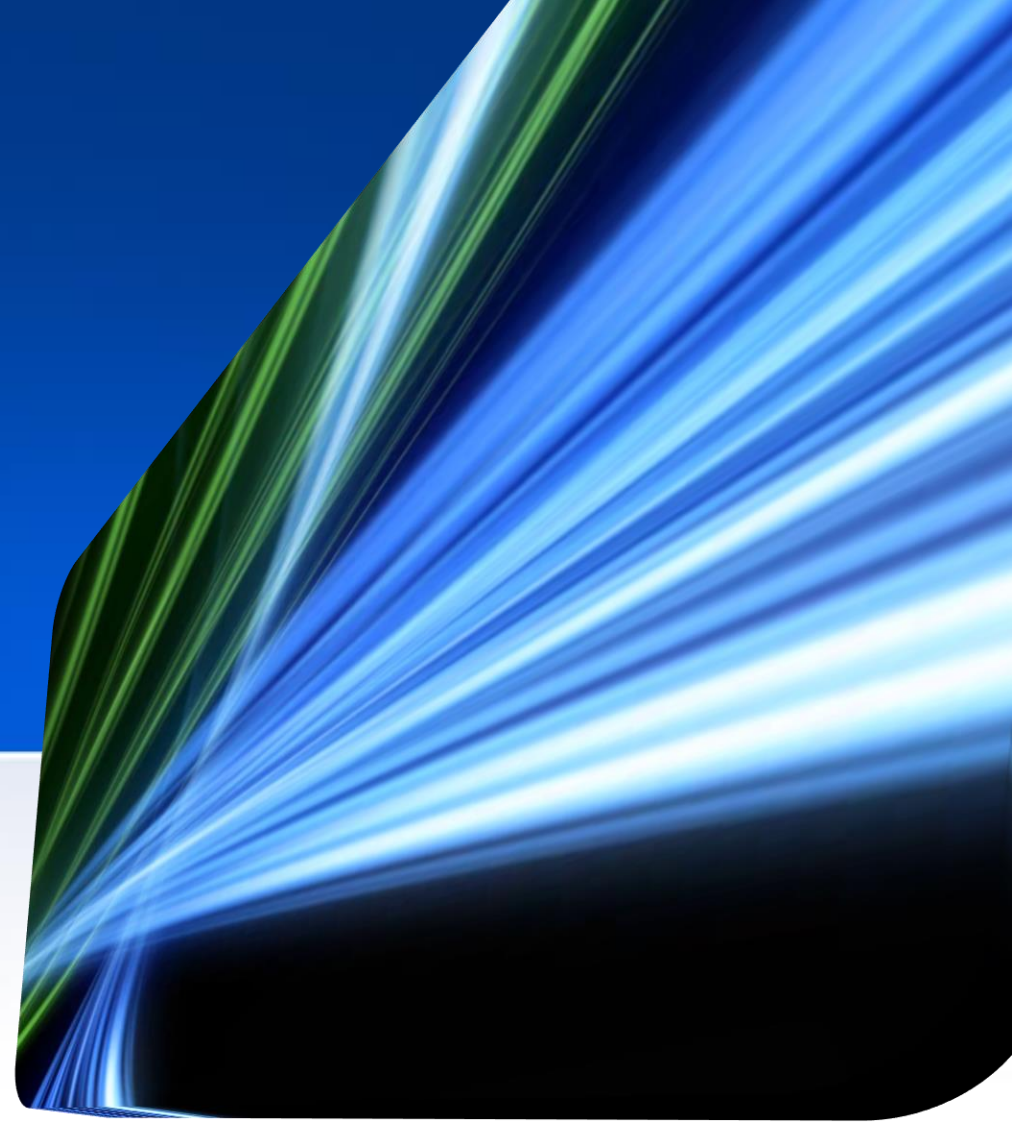
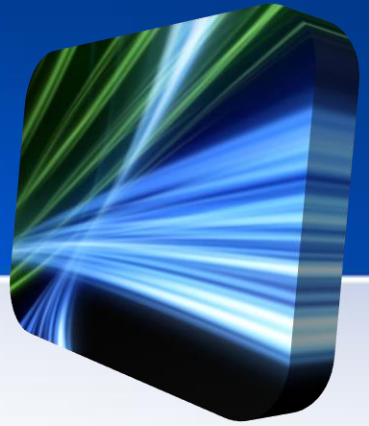
Design Concepts & Principles

Chapter 9

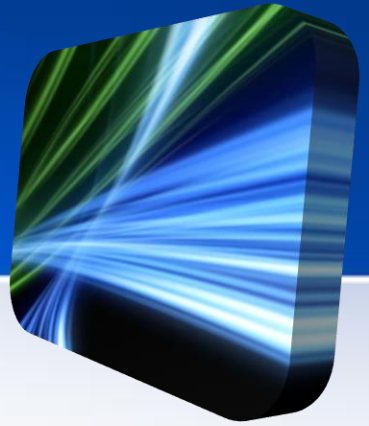


Lesson Objectives



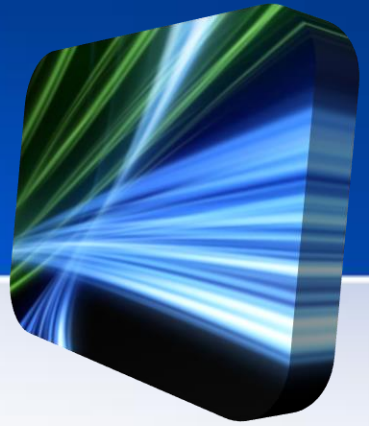
- Explain design process activities
- Discuss the general guidelines for a good design
- Discuss 4 common design quality metrics

Good Design Characteristics



- Design must implement all of the explicit and implicit requirement
- Design must be readable, understandable for those using them for coding, testing and maintenance
- Design should provide a complete picture of the software, its data, functional and behavioural domains

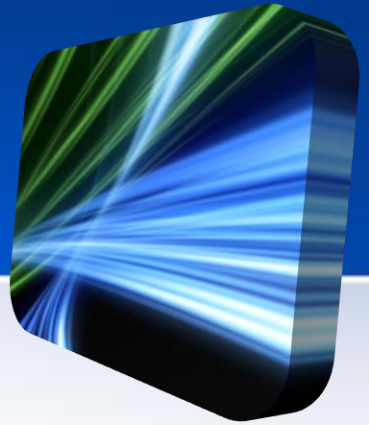
3 Stages of Tackling Design Problems



Study & Understand the Problem

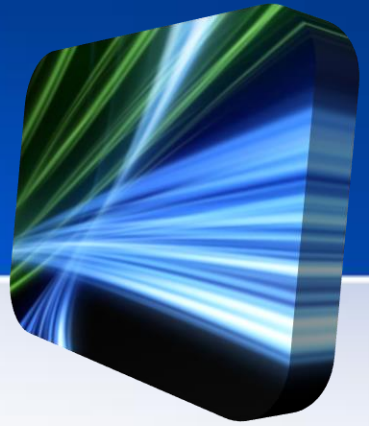
Identify gross features of at least ONE possible solution

Describe each abstraction used in the solution



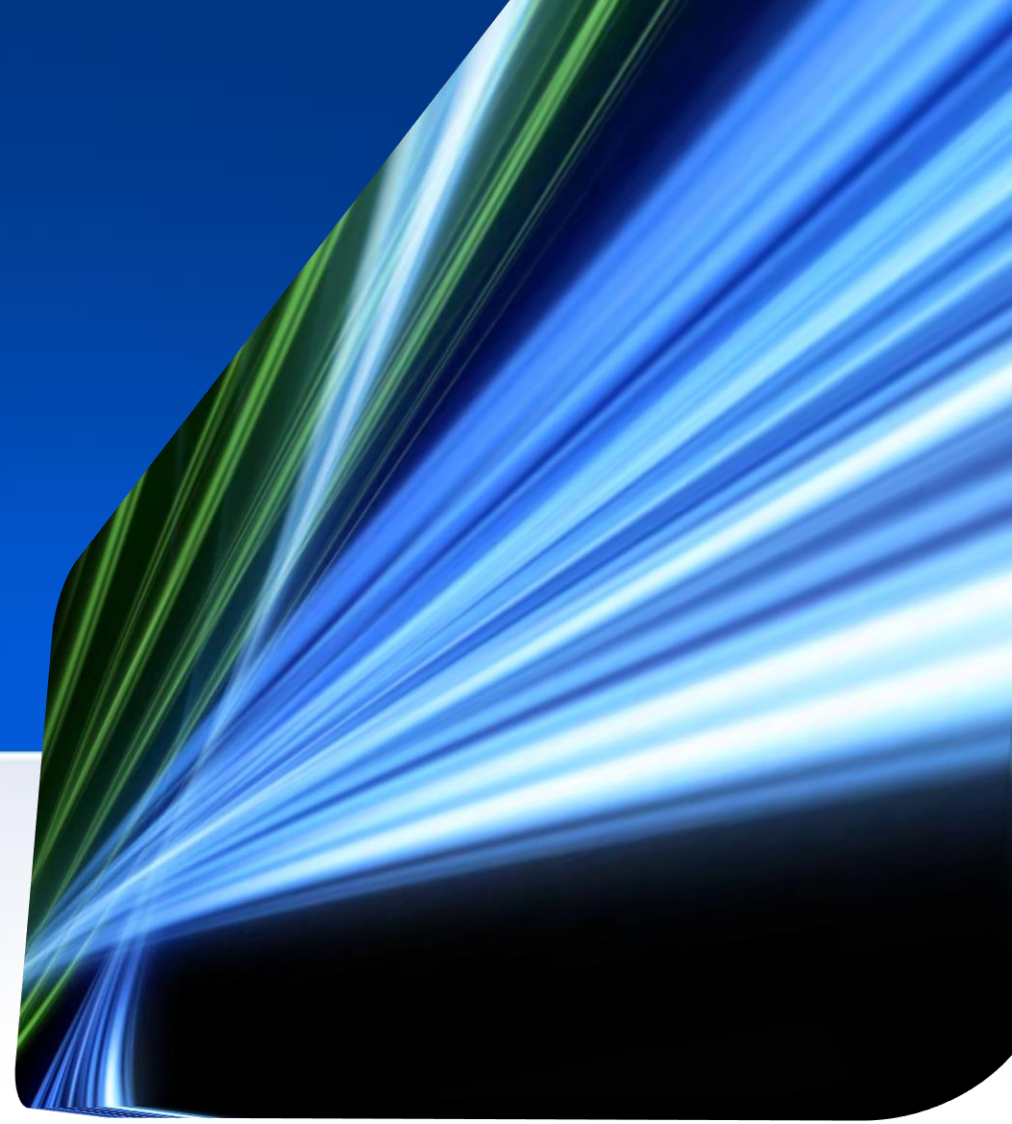
What ***metrics*** to be used to measure the design quality ?

Tackling Design Problems

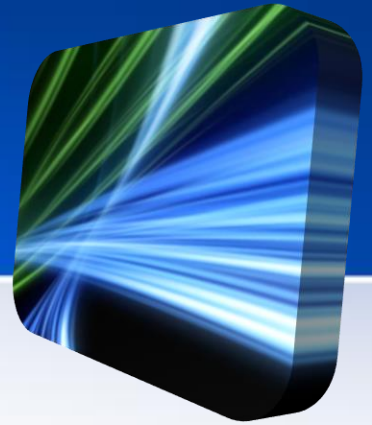


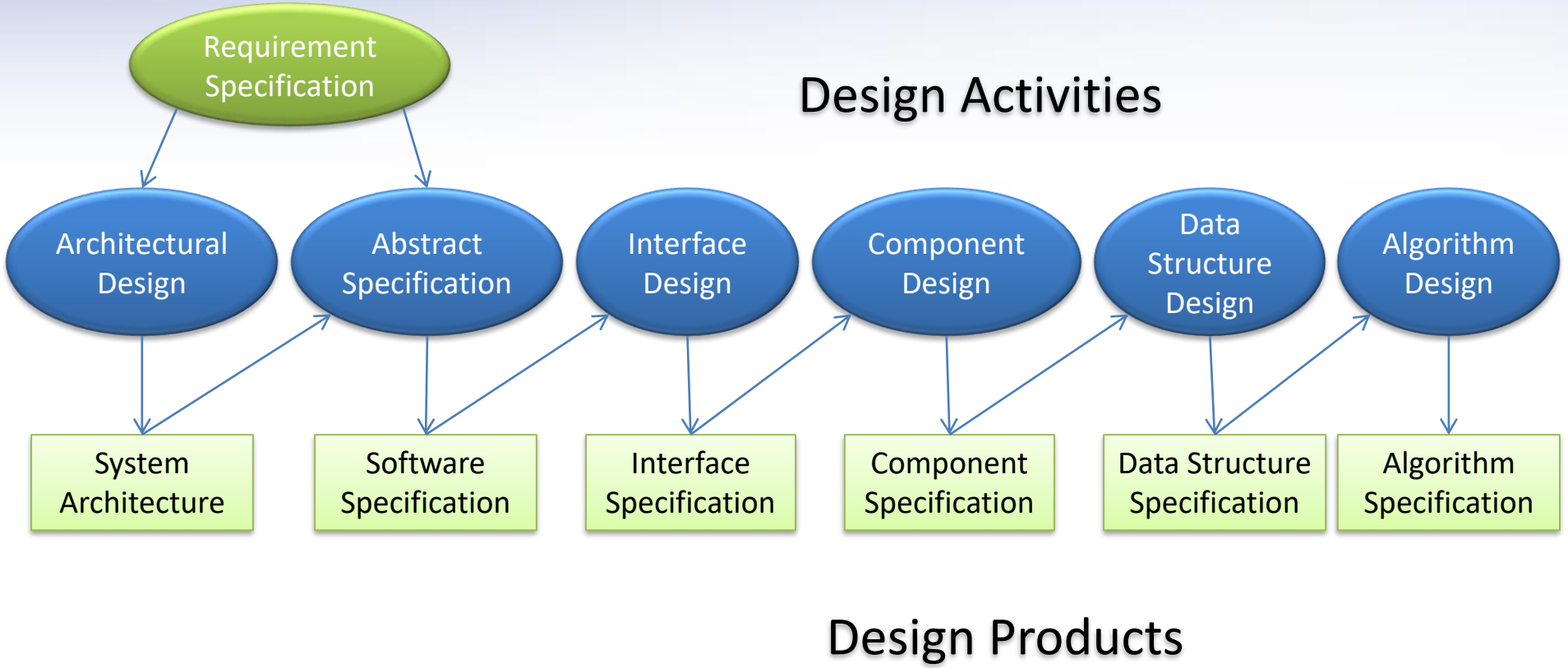
- *Top-down design* is one way of tackling a design problem.
- The problem is recursively partitioned into sub-problems until tractable sub-problems are identified.

Design Activities

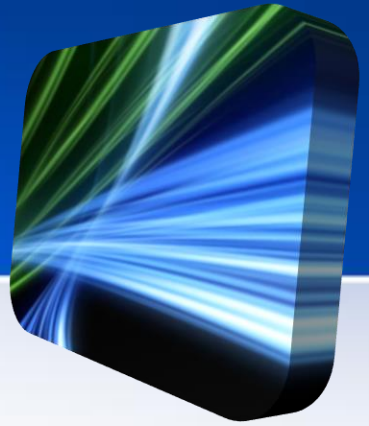


Parallel Design Activities





Parallel Design Activities



Architectural Design:

sub-systems & their relationships (communication) – chp 6

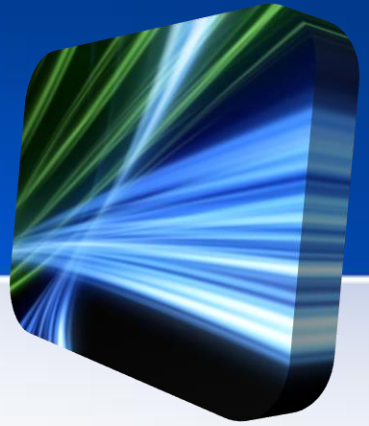
Abstract Specification:

Services & constraints of each sub-system

Interface Design:

Interfaces with other sub-systems (internal interface) , external interface (devices etc), user interface.

Parallel Design Activities



Component Design:

Services are allocated to different components & design their interfaces

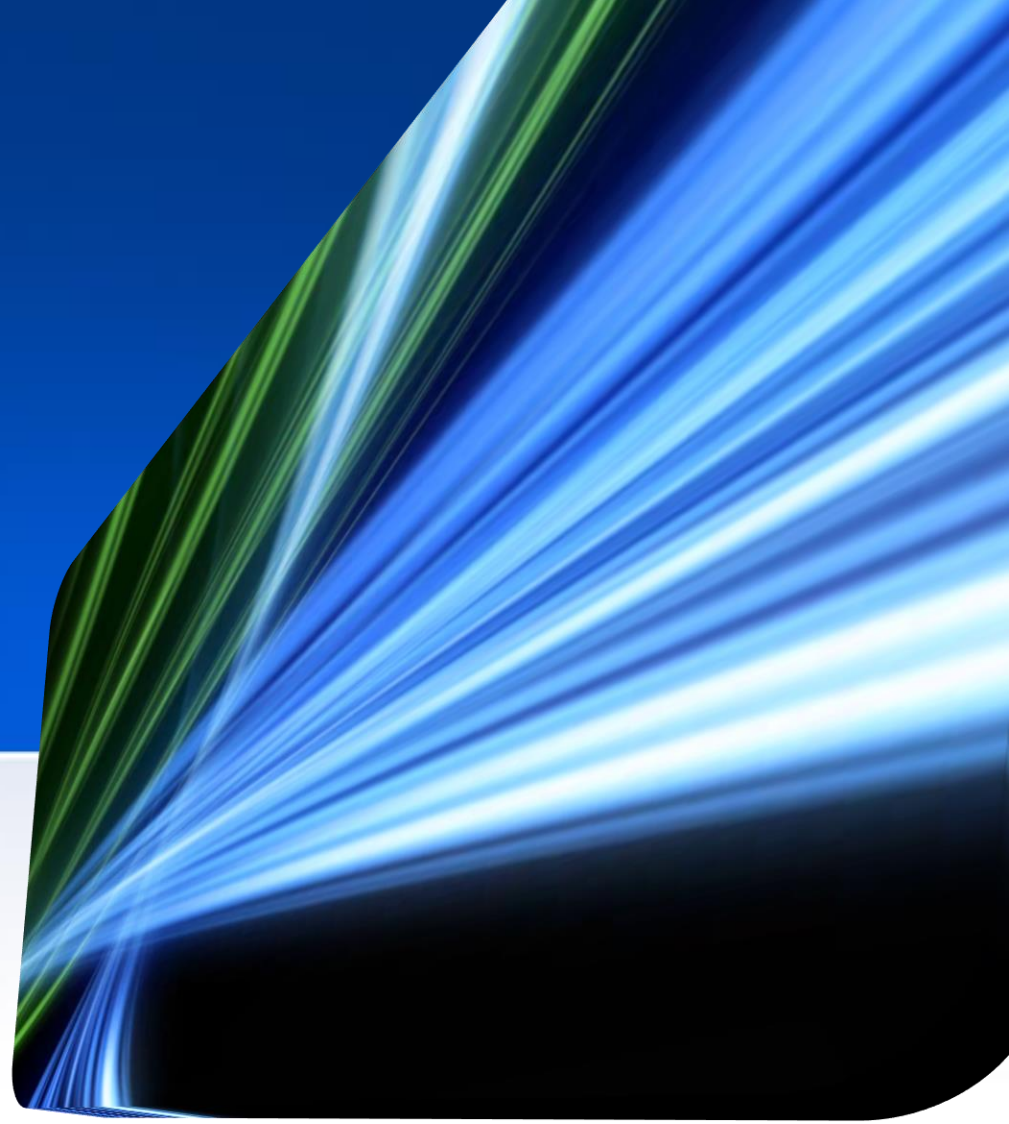
Data Structure Design:

Design data structures used (e.g. link list, records, tables etc)

Algorithm Design:

Design algorithms used to provide the services.

Notation used in Design Documents

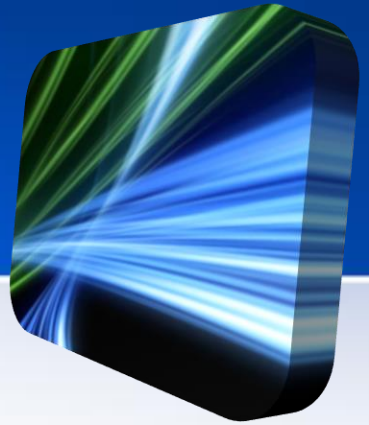


Notations used in Design Documents

Graphical Notations

overall picture of the system & the r/ship between the components.

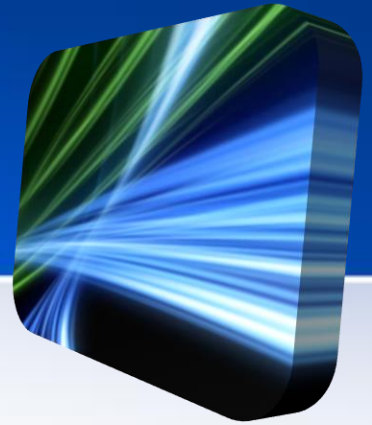
Notations used in Design Documents



Program Description Language

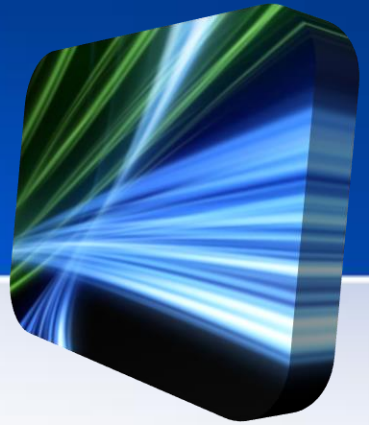
- ✓ use control and structuring constructs based on programming language constructs. Allow the intention of the designer to expressed but not the details of how the design is to be implemented.

Notations used in Design Documents



Informal Text

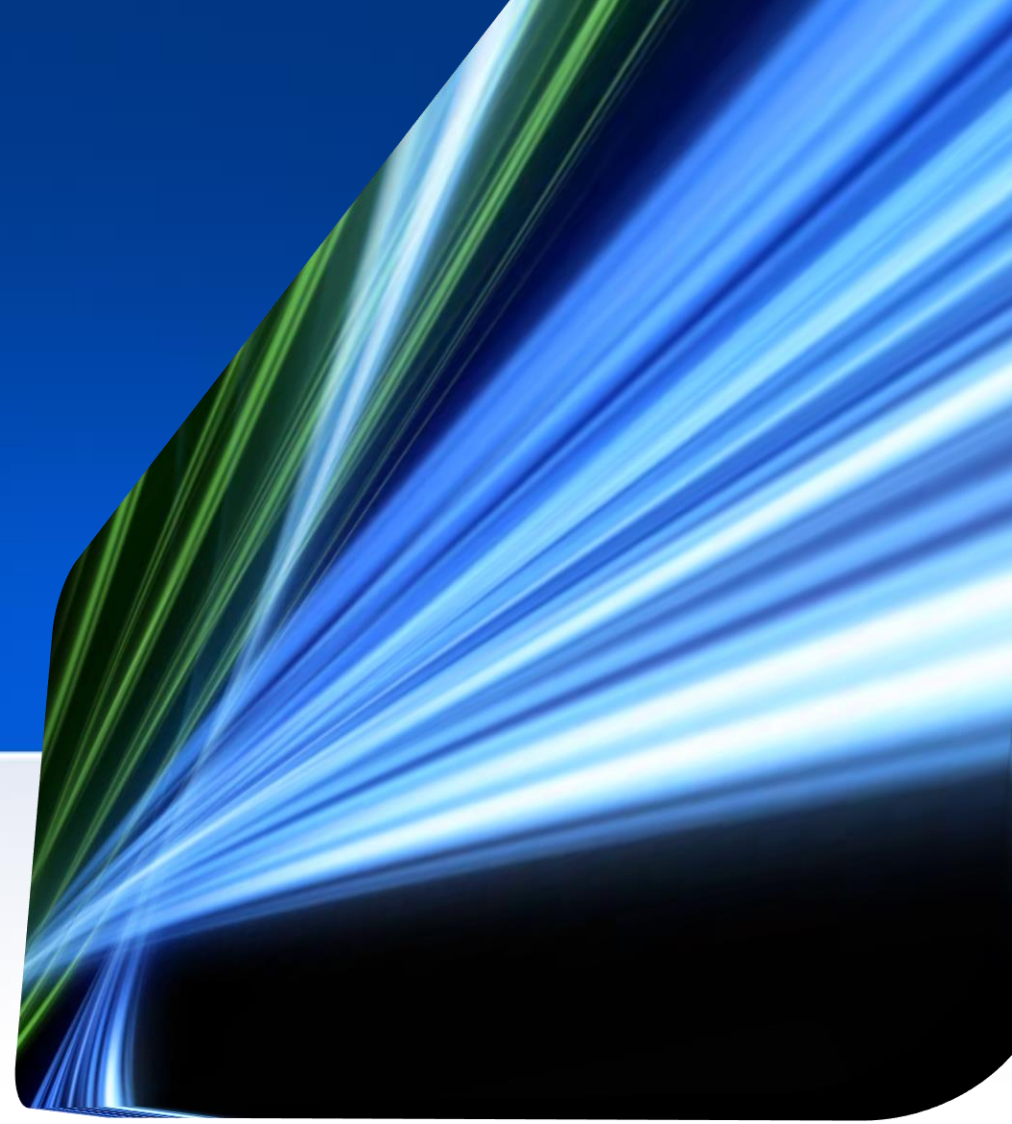
- ✓ for design that can't express formally like non-functional requirements



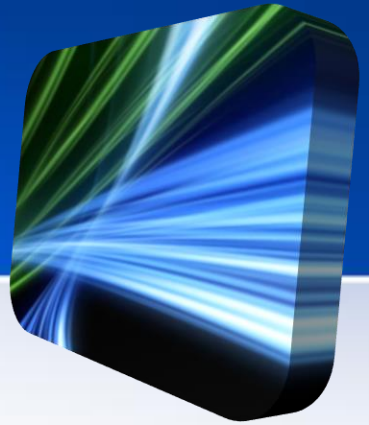
- Notation(s) used in various design activities:

Design Activities	Notation used
Architectural Design	Graphical
Abstract Spec.	Informal Text
Interface Design	Graphical, Informal Text
Component Design	Graphical, PDL
Data Structure Design	Graphical, PDL
Algorithm Design	PDL, Informal Text

DESIGN QUALITY METRICS

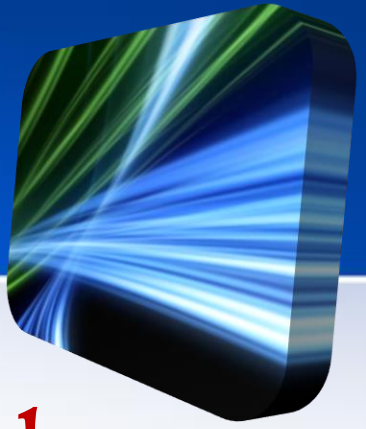


Design Quality Metrics



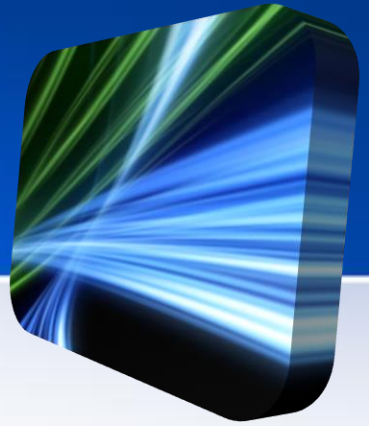
- Cohesion
- Coupling
- Understandability
- Adaptability

Design Quality Metrics

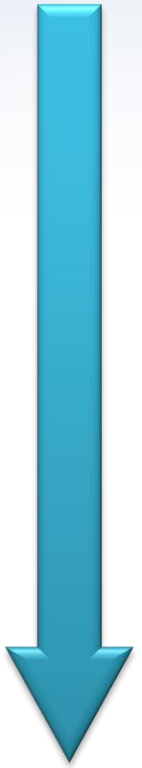


- The cohesion of a component is a measure of the **closeness** of the relationships between its components.
- A measure of the degree to which parts of a program module are closely functionally related

Design Quality Metrics - Cohesion



- Coincidental cohesion
- Logical cohesion
- Temporal cohesion
- Procedural cohesion
- Communication cohesion
- Sequential cohesion
- Functional cohesion

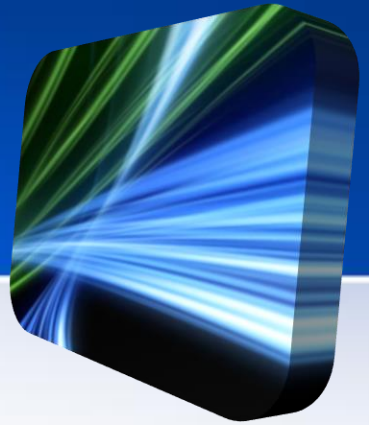


Worst

Best

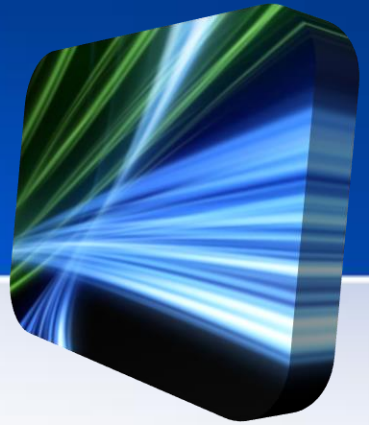
7 levels of Cohesion suggested by Constantine & Yourdon (1979)

7 levels of Cohesion suggested by Constantine & Yourdon (1979)



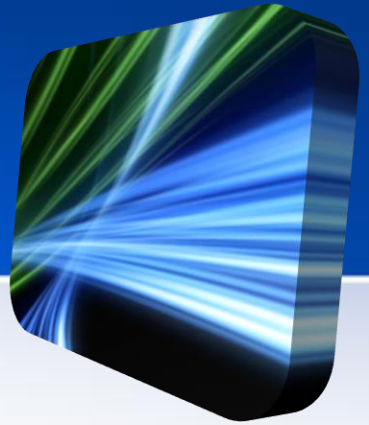
- Coincidental cohesion
  - Parts of a module are grouped arbitrarily
  - E.g. “Utilities” class

7 levels of Cohesion suggested by Constantine & Yourdon (1979)



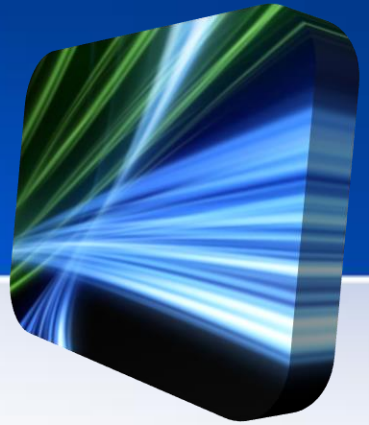
- Logical cohesion
  - Logically categorized to do the same thing
  - E.g. mouse, keyboard → input handling routines

7 levels of Cohesion suggested by Constantine & Yourdon (1979)



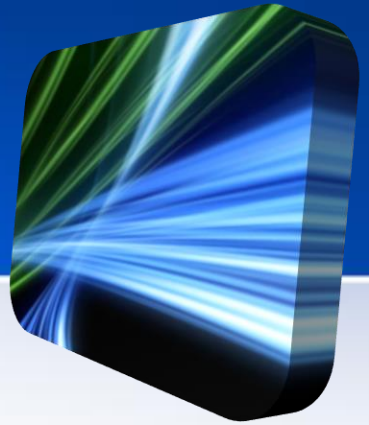
- Temporal cohesion
  - Parts of a module are grouped by when they are processed at a particular time in a program execution
  - E.g. exception → closes open files, creates error log, notifies the user

7 levels of Cohesion suggested by Constantine & Yourdon (1979)



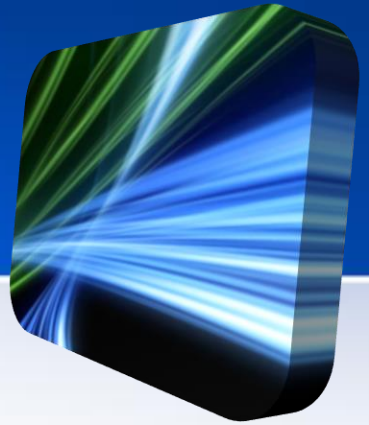
- Procedural cohesion
  - Follow a certain sequence of execution
  - E.g. a function which checks file permissions and opens the file

7 levels of Cohesion suggested by Constantine & Yourdon (1979)



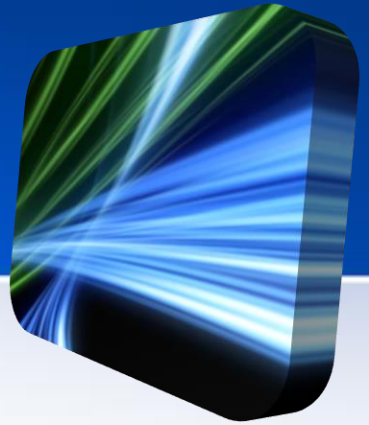
- Communication cohesion
  - Operate on the same data
  - E.g. a module which operates on the same record of information

7 levels of Cohesion suggested by Constantine & Yourdon (1979)



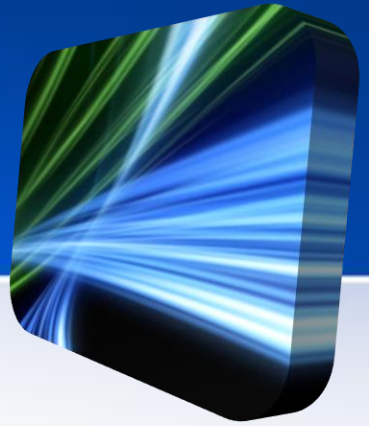
- Sequential cohesion
  - Output from one part is the input to another part like an assembly line
  - E.g. function which reads data from a file and processes the data

7 levels of Cohesion suggested by Constantine & Yourdon (1979)



- Functional cohesion
  - Single well-defined task of the module

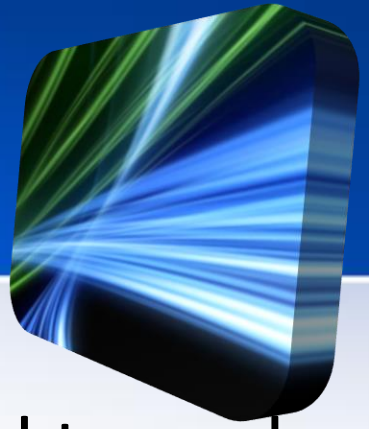
Design quality metrics - Coupling



- Coupling is related to cohesion. It is an indication of the **strength of interconnections** between the components in a design

Highly coupled systems	Loosely coupled systems
have strong interconnections, with program units dependent on each other.	are made up of components, which are independent or almost independent.

Difference between Cohesion and Coupling

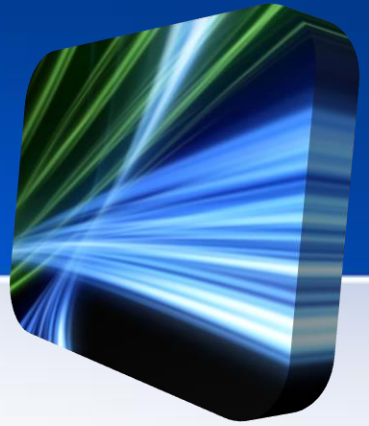


- Cohesion focus on how much the functionality are related to each other within the module, while coupling deals with how much one module is dependent on the other program modules within the whole application.

Cohesion

Coupling

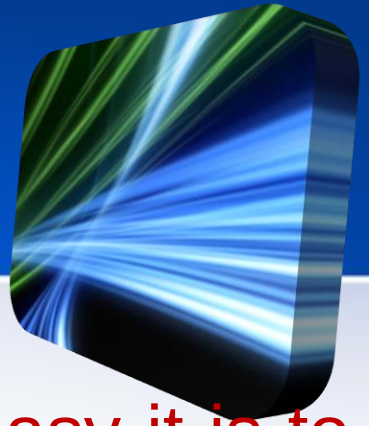
Design Quality Metrics - Understandability



- The understandability of a design is important because anyone changing the design must first understand it

- There are a number component characteristics that affect understandability including: -
  - Cohesion and coupling
  - Naming
  - Documentation
  - Complexity

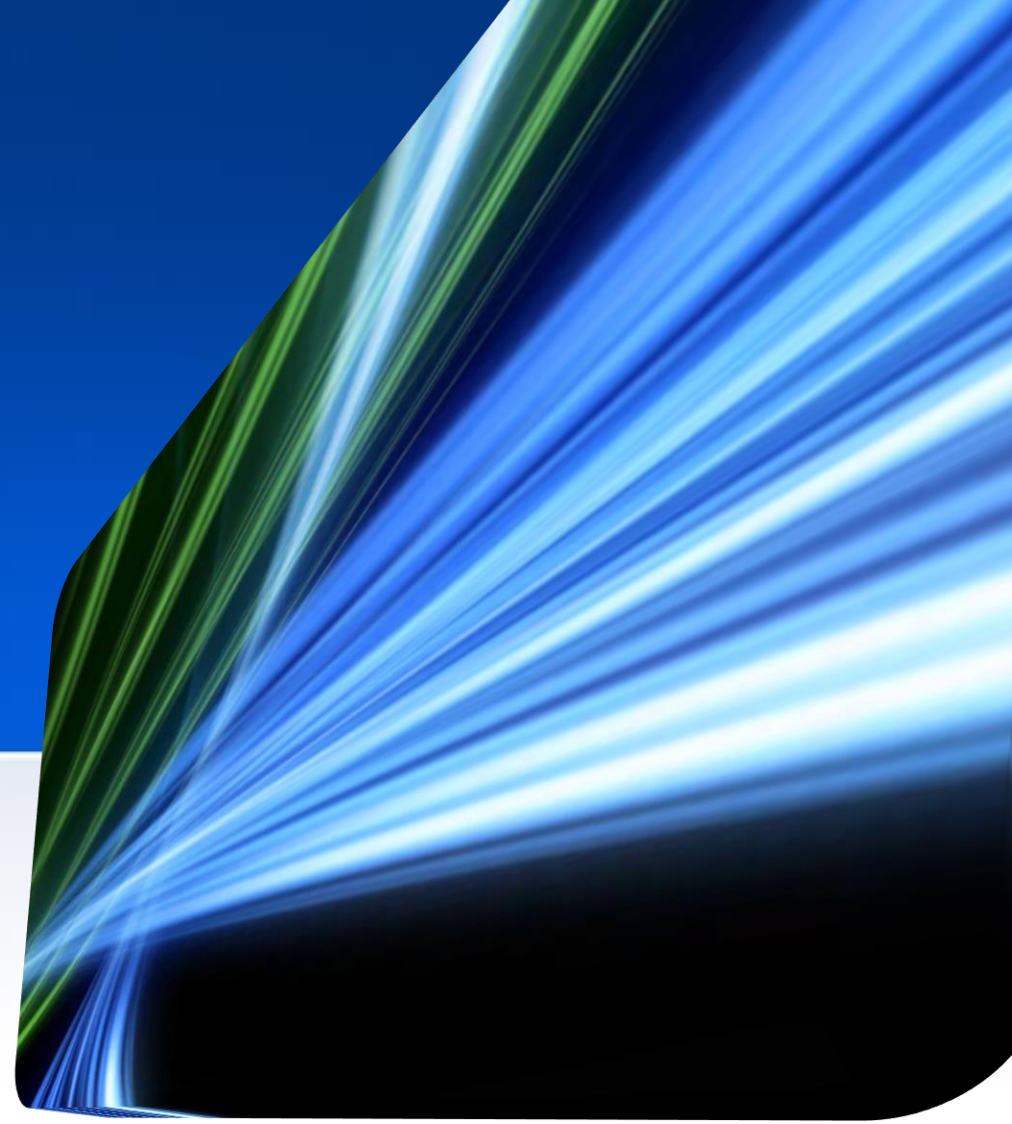
Design Quality Metrics - Adaptability

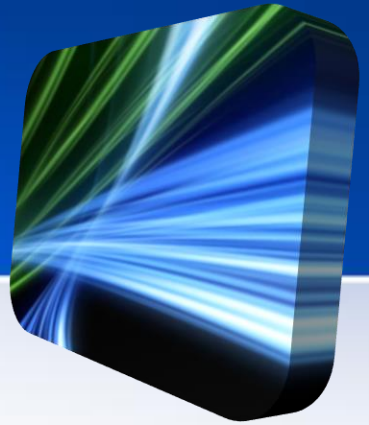


- The adaptability of a design is a general estimate of how easy it is to change the design.

- How to achieve Adaptability?
  - Its components should be loosely coupled
  - Design should be well documented
  - The component documentation should be easily understood.
  - The design should be consistent with the implementation
  - The program implementing the system should be written in a readable way

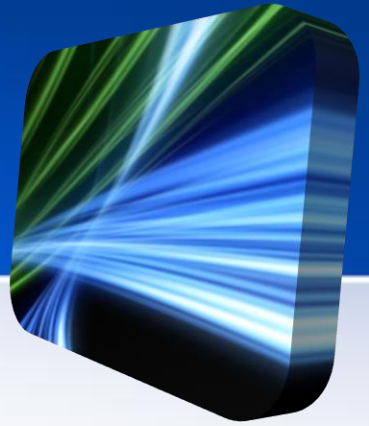
GUIDELINES FOR GOOD DESIGN



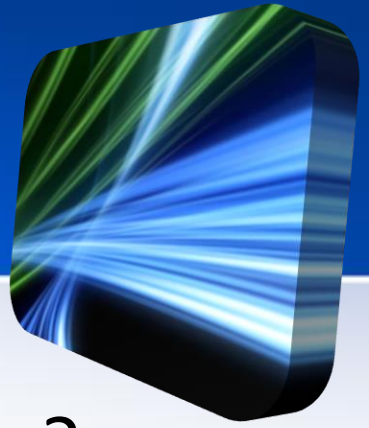


What are the Guidelines for Good Design?

Guidelines for Good Design



- Design is not coding, coding is not design
- The design should be assessed for quality as it is being created, not after the fact
- The design should be reviewed to minimize conceptual errors (e.g. Omission, ambiguity, inconsistency)
- The design should be structured to accommodate change
- The design should be traceable to the analysis model



- How do Cohesiveness and coupling related to Adaptability?

Explain in your answer in ONE sentence

- The lower the coupling, the higher the adaptability
- The higher the cohesiveness, the higher the adaptability