SOFTWARE DESIGN QUALITY

COHESION
and
COUPLING
(Part I)

Goals and Concepts of Structured Analysis and Design:

- structured development of software:
  - methodology
  - notation (DFD, SC)
- integration of analysis and design
- quality control (cohesion, coupling)
- based on the flow of information (data)
- provides proven heuristics

Quality Characteristics for Design

- modular design achieved by
  - abstraction
  - decomposition/aggregation
  - specialisation/generalisation
  - stepwise refinement [N. Winth]
  - information hiding [D.L. Pamas]

  ==> independent modules

Quality Characteristics for Modules

- independent modules
  - avoid aggregation of multiple functions per module
    - single-minded function

  ==> high COHESION
  - minimal interaction with other modules
  - simple interface

  ==> low COUPLING

[Steven, Myers, Constantine]

Module COHESION

- quality of a single module
- based on module’s internal functionality
  - cohesion should be high

- types of cohesion (low to high):

Coincidental Cohesion

- characteristics:
  - functions relate very loosely to each other
    - "random" selection of multiple functions
- example(s):
  - result of post-facto "modularization"
  - result of poor maintenance
- problems:
  - multiple functions
  - no logical connection between functions
Logical Cohesion

- characteristics:
  - combines similar functions
  - implements a set of functions for "all purposes"
- example(s):
  - pack/unpack records for telecommunication
  - central error-handling module
- problems:
  - multiple functions
  - wide interface (with control parameters)
  - bad reusability, maintainability

Temporal / Classical Cohesion

- characteristics:
  - combines similar functions and all of them are always executed at the same time
- example(s):
  - open-all-files module
  - initialise-all-variables
- problems:
  - multiple functions
  - bad reusability, maintainability
  - violation of the principle of "local scope"

Procedural Cohesion

- characteristics:
  - tasks must be executed in a specific sequence and are related by the flow of control
- example(s):
  - combine decision making & task execution, e.g. error-detection and error handling
  - local optimisation
- problems:
  - fixed combination and order of tasks
  - bad reusability, maintainability

Communicational Cohesion

- characteristics:
  - tasks that use the same data (parameters)
  - sharing portions of a data structure
- example(s):
  - monthly-employee-statistic and write-paycheque, both functions using the employee-record
- problems:
  - wide interface (with stamp data coupling)
  - code sharing is tempting
  - bad reusability, maintainability

Sequential Cohesion

- characteristics:
  - tasks execute in a specific sequence and are related by the flow of data
  - tasks transform data in a multitude of steps
- example(s):
  - calculate-taxes and write-paycheque
- problems:
  - hidden interface(s)
  - fixed combination and order of tasks
  - bad reusability, maintainability
Sequential Cohesion

- characteristics:
  - single-minded function
  - cannot be decomposed any further (in a meaningful way)
  - not based on control or data flow
  - algorithmic behaviour / description
  - simple interface = good parameters
    (no stamp coupling, no control)

- problems:
  - cannot always be achieved

End of Section 8a

coming up:
coupling