Chapter 5

How to build the software based on the SPEC

I. Fundamental Design concepts
   (a) Abstraction
   (b) Information hiding
   (c) Modularity
   (d) Functional independence
   (e) Cohesion
   (f) Coupling

II. AD
    (a) Data How oriented AD
    (b) Data Structure AD
    (c) Object oriented AD

III. DD

I. Fundamental concepts in Design

   a. Abstraction:
      The way of looking at the solution at some level of generalization without regard to irrelevant low-level details.

      2 types: Procedure ~ / data ~
      like DFD

      1) Ex: of Procedure abstraction
         Open the door
         - walk to the door
         - grab the knob
         - turn the knob
         - pull the knob

      2) Data abstraction:
         Ex: Door is abstraction for
         - Door type
         - Swing Direction
         - Weight
         - Dimension

   b. Information Hiding (Encapsulation)
      Procedures Data contained in a module should be inaccessible to the other modules that have no need for such information.
c. Modularity
Divide the solution into smaller pieces (modules) with minimum cost.

Eg.
Problem: P
Cost of solving C(P)
P = P1 + P2
C(P1) + C(P2) < C(P) Divide & conquer

![Diagram of cost integration and total cost vs. number of modules]

d. Functional independent modules
F.I. suggests that we build modules “single-minded” & avoid excessive interaction with other modules.
⇒ cohesion & coupling
⇒ high cohesion, low coupling

e. Cohesion: is a measure of the relative functional strength of a module.
There are 7 measures of cohesion:
bad
1. Coincidental cohesion – perform multiple functions, no relations b/t these functions.
2. Logical ~
3. Temporal ~ - Does functions based on the time, Initialize all variables open all files.
4. Procedural ~
5. communicational ~
6. Informational ~ A module has I.C. if it performs a number of functions, each with its own entry & exit point, all performed on the same data.
7. Functional ~ module performs exactly one function.
good

Recommendation: at least 6 & 7

Ex. Function IC
begin
Entry \[f_1\] \[f_2\] \[f_3\] \[f_4\] exit
fi \(i=1, \ldots, 4\) perform on the same data.
end
f. Coupling is a measure of interconnection / interdependence among modules. Lower is better
- side effect
- Reusability considerations

5 measures of coupling

good
1. Data coupling
2. Stamp coupling
3. Control coupling
4. Common coupling
5. Content

Bad
1. Pass data / using simple data (Integer, Ptr….)
2. Pass record structure, array
3. Pass control to the other module

Ex. Func A
Begin
: B (flag, …)
: 
end

Func B (flag, …)
begin
: Computations based on the flag
: 
end

4. Modules access same global variables
    Use “goto”

II. A.D. Techniques

a) Data flow oriented AD Technique
⇒ DFD (Data Flow Diagram)

Specification

DFD
 REQ⇒

translate
⇒
PSD

DFD ⇒ PSD Program Structure Diagram

Translate DFD into PSD
3 Steps

Step 1. Establish the type of the flow in DFD
- Transform > type
- Transaction

Step 2. Map DFD ⇒ PSD based on the type
Step 3. Apply heuristics (Refinement) [Imploide
    Explode]
(Step 1): Establish the type of the flow in DFD.
1. Transform type (sausage)

(Step 2): Translate DFD to psd based on the type
(a) Transform type
   i. isolate the flow boundaries. (I, P, O)
   ii. Perform “first-level” factoring
   iii. Perform “second-level” factoring

Ex. (i)

Diagram showing the flow of input, process, and output.
Create
3 control modules

software

CI

1
2

3

CP

4
5
9

6
7
8

9

10
11

12

CO

I

P

O

S.W.
(ii) First level factoring
   (1) Create 3 control modules one per boundry
   (2) Attach modules such that the lower-modules perform the work

(iii) Second level factoring
      Map individual transforms

(b) Transaction mapping

Ex.

(i) Identify incoming transaction center & paths
(ii) Perform First-level factoring
     - create 2 control modules
     - Develop the incoming flow as in transform flow
(iii) Perform 2\textsuperscript{nd} factoring map individual path

Software (main)

Eliminate this in (Step 3)
Step 3: Apply heuristics

(i) Eliminate control modules with 0 or 1 module attached to them

(ii) Coupling → if high coupling between 2 modules, → implode the modules

(iii) Low cohesion → explode the module (After change, need to change DFD as well to keep consistency)

(iv) Create common module (explosion)

Ex. ATM machine

Level 1

1. Read card info
2. Read user input
3. Validate & Prompt
4. Withdraw
5. Deposit

Level 2: Withdraw

4.1 Determine Type
4.2 Process Quick Cash
4.3 Normal withdraw
4.4 Process ATM action
4.5 Display msg
4.6 Control cash Dispenser
b) Data Structure Oriented AD
We look at the structure of Data, try to create structure of processes.

Ex. A software for a c.c. accounting system (simplified).
We have to generate the following report.

<table>
<thead>
<tr>
<th>CNO.</th>
<th>Date</th>
<th>Pmt</th>
<th>OLD BAL</th>
<th>NEW BAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZ</td>
<td>10/10/98</td>
<td>100.-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XYZ</td>
<td>10/12/98</td>
<td>200.-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.....</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>300.-</td>
<td>1000.-</td>
<td>700.-</td>
</tr>
<tr>
<td>ZZZ</td>
<td></td>
<td>----</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Grand. Total

1. Structure of data (reports)
c) OOAD 3 Steps
1) Construct Interaction diagram
   - sequence
   - collaboration
2) Construct the detailed class diagram (include methods)
3) Design AD in terms of clients of objects

Step 1. Construct interaction diagram
2 types (sequence, collaboration)

i) Sequence diagram emphasis the chronological order of event

Ex. Of sequence diagram (elevator prob)
Elevator

Events: (behaviors)
1. user presses the F-Button (up)
2. up floor button is on
3. elevator arrives
4. up-floor button goes off
5. door opens
6. press E-button
7. E-button is on
8. door closes
9. Elevator goes up / arrives
10. E-button goes off
Step 2. Detailed class Diagram

**Button**
- Light: Boolean
- Turn on / off

**E - Button**

**F - Button**

**Elevator**
- Direction: up, down
- Move( )

**E - controller**

**Door**
- Is-open: Boolean
- Open / close
Step 3. AD based on the client objects

III. Detailed Design (DD)
Is an iterative process where the control relationship of the processes are specified in an algorithmic manner, it is based on the concept called structured programming. (The algorithm can be written using 3 component sequence, iteration, selection).

1) Flow Chart

2) Box Diagram

3) PDL (Program design language)
Program construct + English

Module A is
Require: …
Provide: …
Begin:
end