Component Level Design

Ruchita Shah

Component Design

- Occurs after data, archi & i/f design
- high level of abstraction
- program at low level of abstraction
- conversion introduce bugs
- Follow a design guidelines

Structured Programming

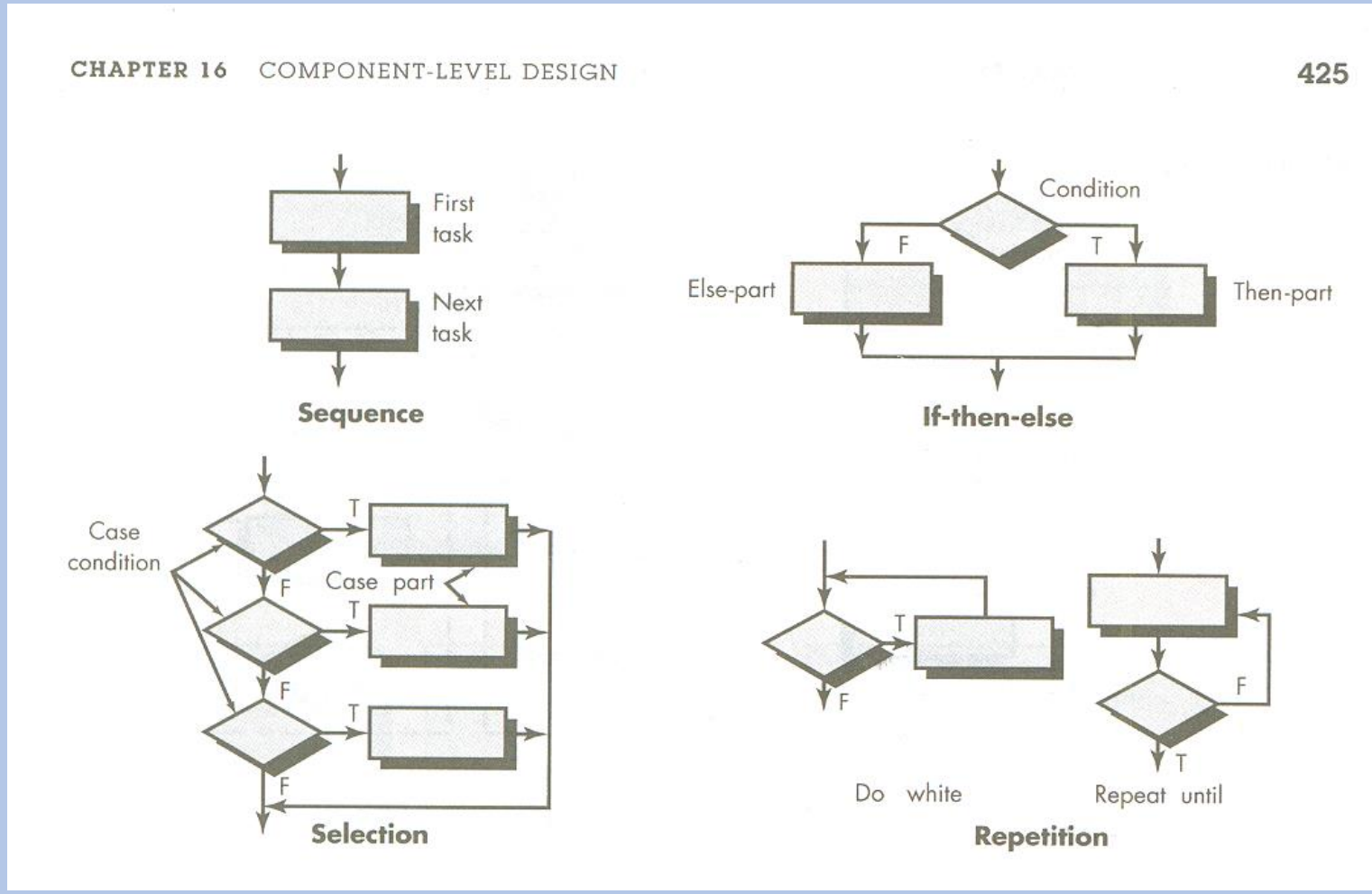
- A set of logical constructs to form program
- Emphasize on maintenance of functional domain
- Constructs are sequence, condition & repetition
- Sequence implements processing steps
- Condition provide facility to select process
- Repetition allows looping

Structured Programming

- Fundamental for structured design are used to limit design to small no of operations
- Reduces complexity
- Enhances readability, testability & maintainability
- Enables understanding process called as chunks

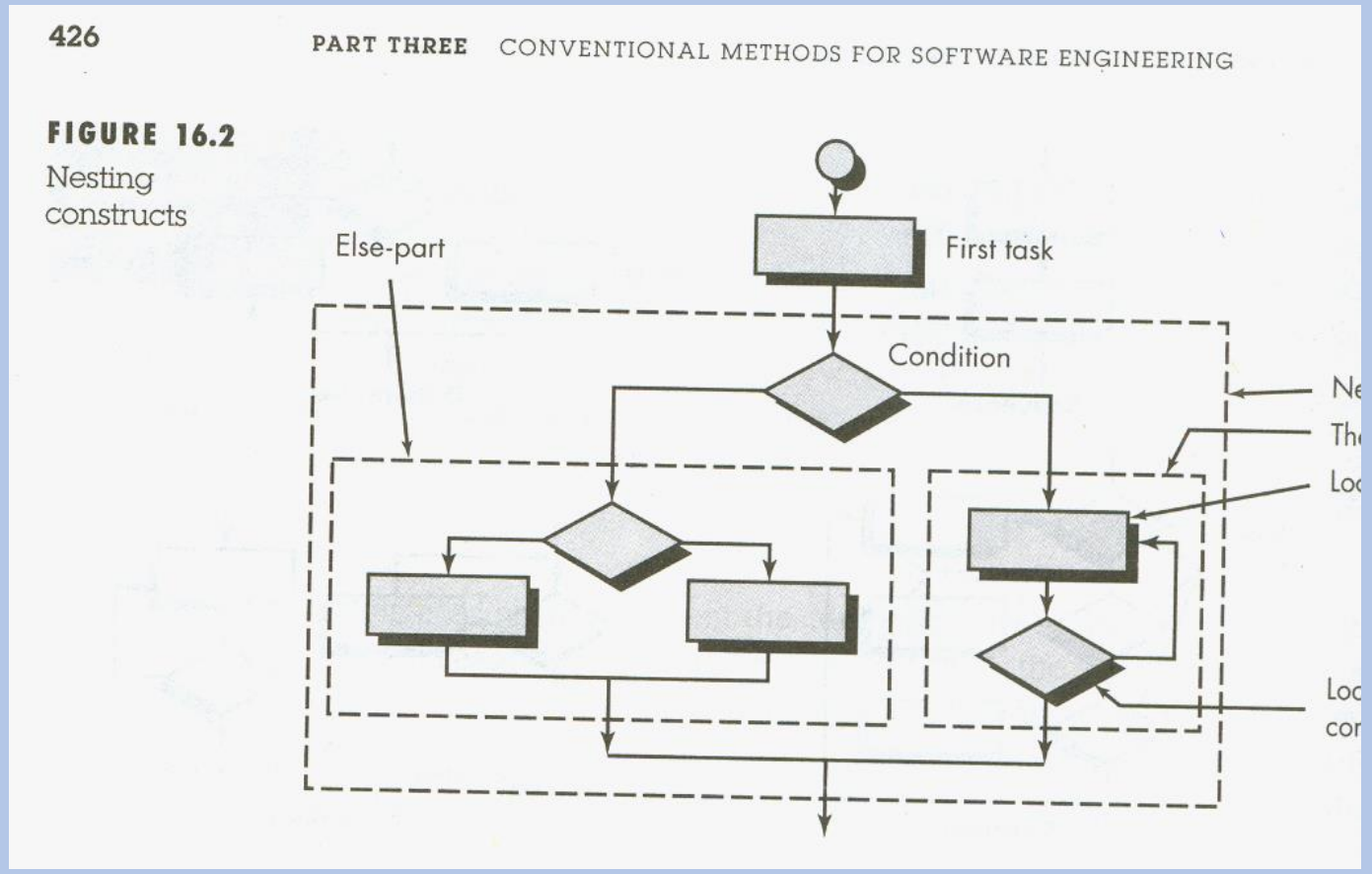
Graphical Design Notation

- Depict procedural detail
- misuse lead to wrong s/w
- ex flowchart



Graphical Design Notation

- Nesting may be applied



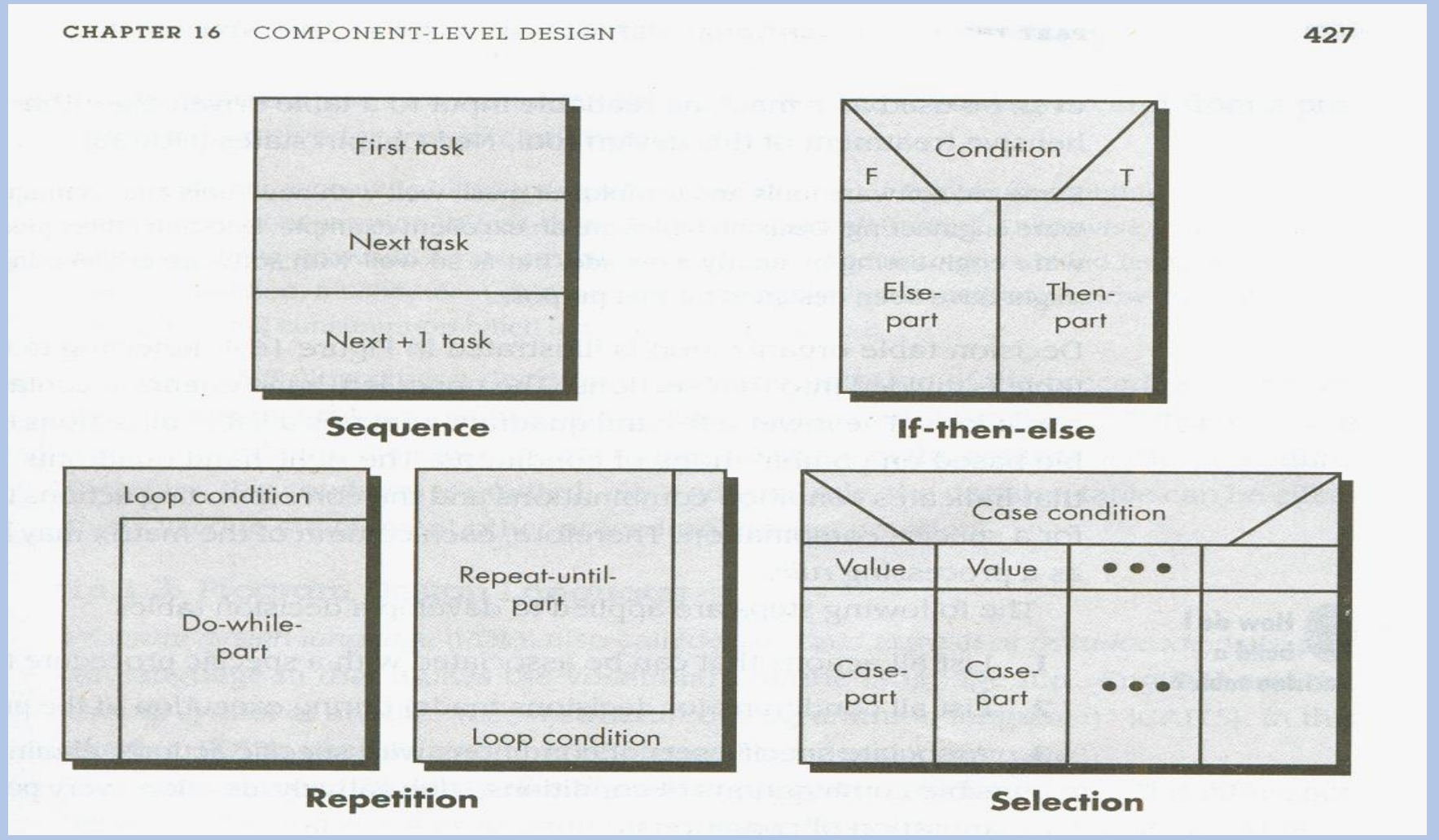
Graphical Design Notation

- Dogmatic of constructs introduce inefficiency when escape from nested loop or nested conditions required
- complex testing, increase in error, negative impact on readability & maintainability
- 2 options
  - Redesigning of procedural representation so escape branch not needed
  - Structured constructs violated in controlled manner, a constrained branch out of nested flow
- Ideally 1st option
- 2nd option accommodated without violating spirit of structured programming

Box Diagram

- Do not allow to violate structured construct
- Characteristics are :
  - Functional domain well defined & clearly visible
  - Arbitrary transfer of control impossible
  - Scope of local & global data easily determined
  - Recursion easily represented

Box Diagram


CHAPTER 16 COMPONENT-LEVEL DESIGN
427
First task
Next task
Next +1 task
Sequence
Condition
F
T
Else-part
Then-part
If-then-else
Loop condition
Do-while-part
Repeat-until-part
Loop condition
Repetition
Case condition
Value
Value
Case-part
Case-part
Selection

Box Diagram

- Fundamental element is box
- Sequence by connecting 2 boxes
- Condition represented by a condition box followed by then-part & else-part
- Repetition by bounding pattern that enclose process to be repeated
- Selection using graphical form
- Box diagram layered on multiple pages
- Call to subordinate module represented within box by module name enclosed in oval

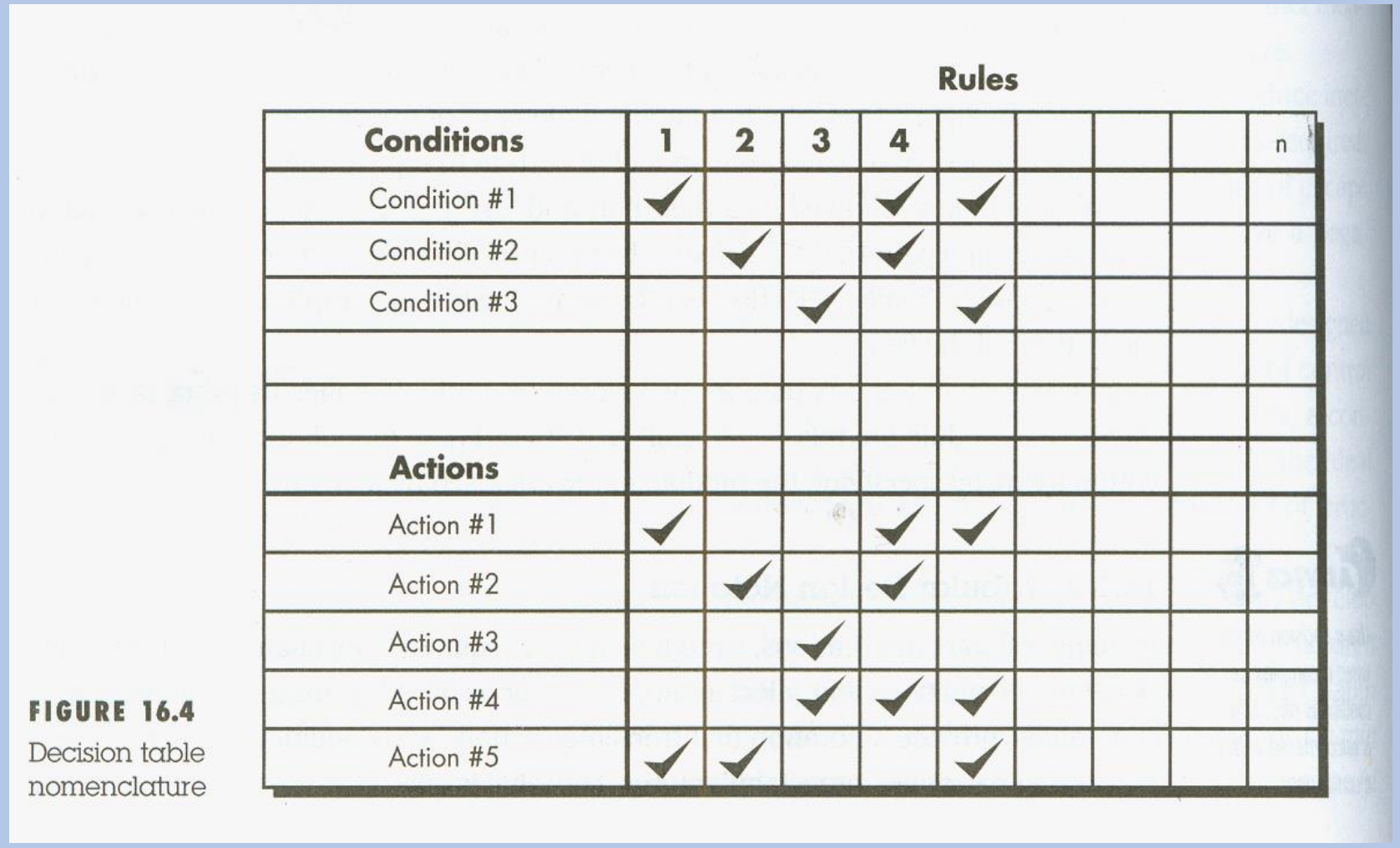
Tabular Design Notation

- Module evaluated for complex combination of conditions & select appropriate action
- Decision table translated actions & conditions into tabular form
- Easy to interpret
- Used as machine readable i/p

Tabular Design Notation

- Table divided in 4 sections
- Upper left-hand quadrant – list conditions
- lower left-hand – list of all actions on combination of conditions
- right-hand – a matrix indicating conditions & corresponding actions for specific combination

Tabular Design Notation



	Rules								
Conditions	1	2	3	4					n
Condition #1	✓			✓	✓				
Condition #2		✓		✓					
Condition #3			✓		✓				
Actions									
Action #1	✓			✓	✓				
Action #2		✓		✓					
Action #3			✓						
Action #4			✓	✓	✓				
Action #5	✓	✓			✓				

FIGURE 16.4
Decision table nomenclature

Tabular Design Notation

- To develop a decision table, steps are :

1. List all action associated with specific procedure
2. List all condition of procedure
3. Associate set of conditions with actions, eliminate impossible combinations
4. Define rules, indicate actions occurring for a set of combinations

Tabular Design Notation

- Ex. Electricity billing system – if fixed rate charge, minimum monthly charge for < 100 KWH, otherwise apply schedule A rating, if variable rate applied, schedule A rate for consumption < 100KWH, if > 100 KWH then schedule B rate

- Five rules are five conditions

Tabular Design Notation

	Rules				
Conditions	**1**	**2**	**3**	**4**	**5**
Fixed rate acct.	T	T	F	F	F
Variable rate acct.	F	F	T	T	F
Consumption <100 kwh	T	F	T	F	
Consumption ≥100 kwh	F	T	F	T	
Actions					
Min. monthly charge	✓				
Schedule A billing		✓	✓		
Schedule B billing				✓	
Other treatment					✓

Program Design Structure (PDL)

- Structured English or pseudo code
- Combines syntax of programming language with vocabulary of English
- like modern programming language
- Difference b/w PDL & programming lang is use of text embedded in syntax
- Can not be compiled
- Tools to translate PDL to programming lang

Program Design Structure (PDL)

- A design lang should have following characteristics :

1. Fixed syntax of keywords for all structured constructs, data declaration & modularity characteristics
2. Free syntax of natural lang
3. Data declaration facility, includes simple & complex DS
4. Subprogram definition & calling techniques

Program Design Structure (PDL)

- should include
  - construct for subprogram definition
  - I/f description
  - Data declaration
  - Block structure
  - Condition construct
  - Repetition &
  - I/O
  - can include keywords for multitasking, concurrent processing, interrupt handling, inter-process synchronization etc.

Program Design Structure (PDL)

- Ex – design of SafeHome Security system, following PDL describes its procedure

```
PROCEDURE security-monitor;
INTERFACE RETURNS system. status;
TYPE signal IS STRUCTURE DEFINED
        name IS STRING LENGTH VAR;
        address IS HEX device location ;
        bound.value IS upper bound SCALAR;
        message IS STRING LENGTH VAR;
END signal TYPE;
TYPE system.status IS BIT (4);
```

Program Design Structure (PDL)

.

.

.

.

initialize all system ports & reset all hardware;

CASE OF control.panel.swotches (cps)

WHEN cps = “test” SELECT

CALL alarm PROCEDURE WITH “on” for test.time in seconds;

WHEN cps = “alarm-off” SELECT

CALL alarm PROCEDURE WITH “off”;

.

Comparison of Design Notation

- Number of different tech for procedural design
- Comparison for best result
- procedural representation easy to understand & review
- code derived from notation as a byproduct of design
- easily maintainable
- attributes of design notations are :

1. Modularity : support modular s/w & i/f facility
2. Overall simplicity : simple to learn, easy to use & read
3. Ease of editing :

Comparison of Design Notation

3. Machine readability : can be i/p to a computerized development system
4. Maintainability : s/w maintenance, most costly, means maintenance of procedural design
5. Structure enforcement : structured programming concept, good design
6. Automatic processing : can be processed for better insight for correctness & quality
7. Data representation : local & global data – essential element of component-level design, directly
8. Logic verification : automatic verification of design logic
9. “code-to” ability : code generation easy, less effort & errors