CS 306

SOFTWARE ENGINEERING I

GEORGE W. ZOBRIST

GEORGE W. ZOBRIST
COMPUTER SCIENCE DEPARMENT
UNIVERSITY OF MISSOURI-ROLLA
1870 MINER CIRCLE
ROLLA, MISSOURI 65409-0350

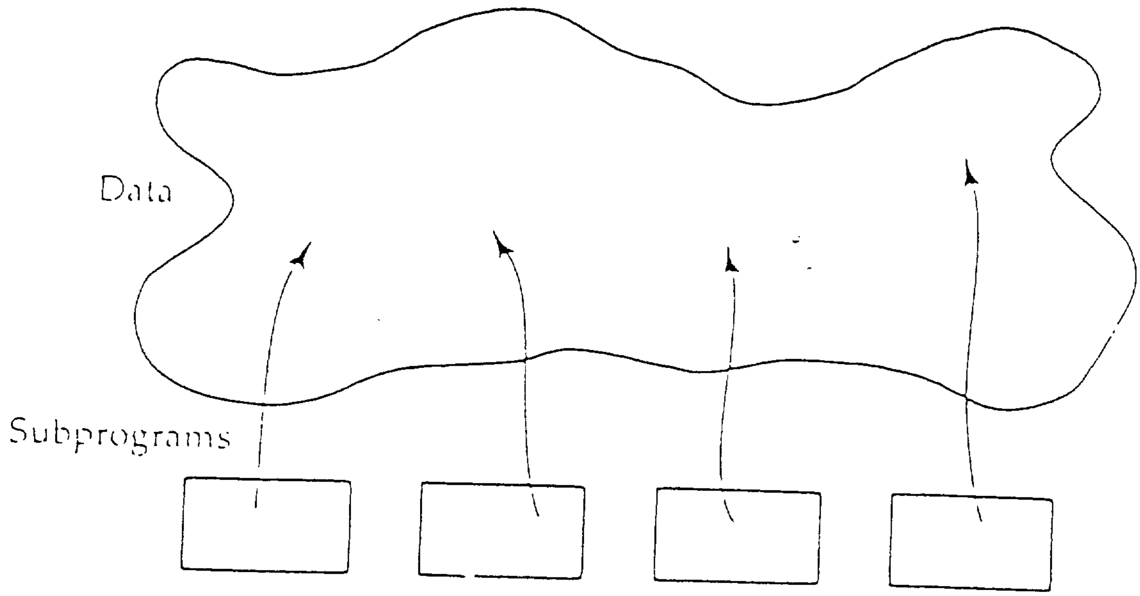


Figure 4-2 Topology of first- and second-generation languages

- *First-Generation Languages* (1954–1958)

FORTRAN I ALGOL 58 Flowmatic IPL V	mathematical expressions

- *Second-Generation Languages* (1959–1961)

FORTRAN II	subroutines, separate compilation
ALGOL 60	block structure, data types
COBOL	data description, file handling
LISP	list processing, pointers

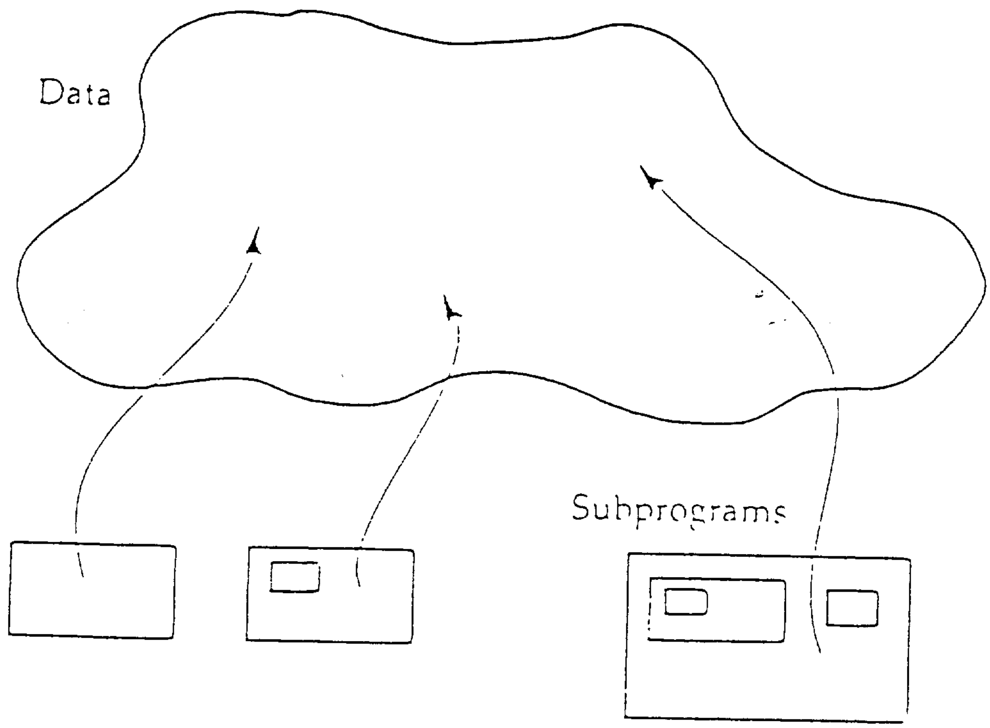


Figure 4-3 Topology of second- and third-generation languages

- *Third-Generation Languages* (1962–1970)

PL/1	FORTRAN+ALGOL+COBOL
ALGOL 68	rigorous successor to ALGOL 60
Pascal	simple successor to ALGOL 60
SIMULA	classes, data abstraction

- *The Generation Gap* (1970–1980)
  Many different languages, but none endured.

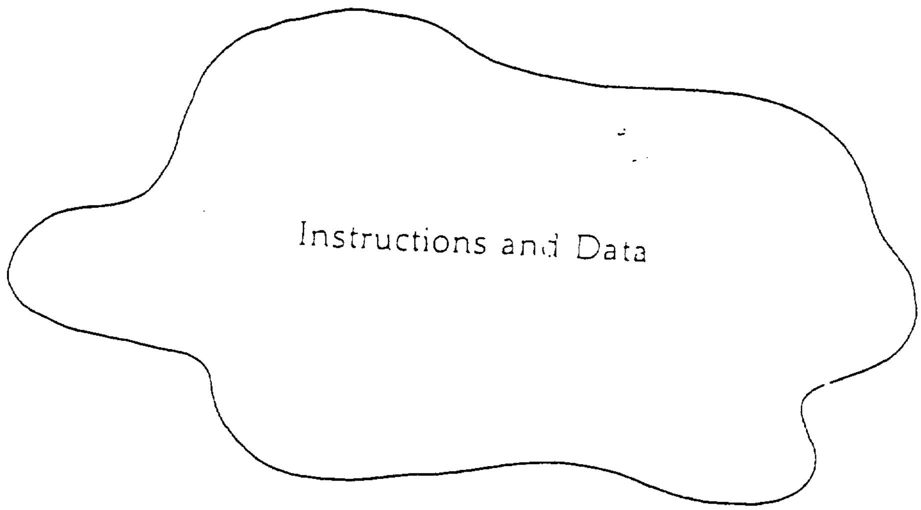


Figure 4-4 Topology of assembly languages

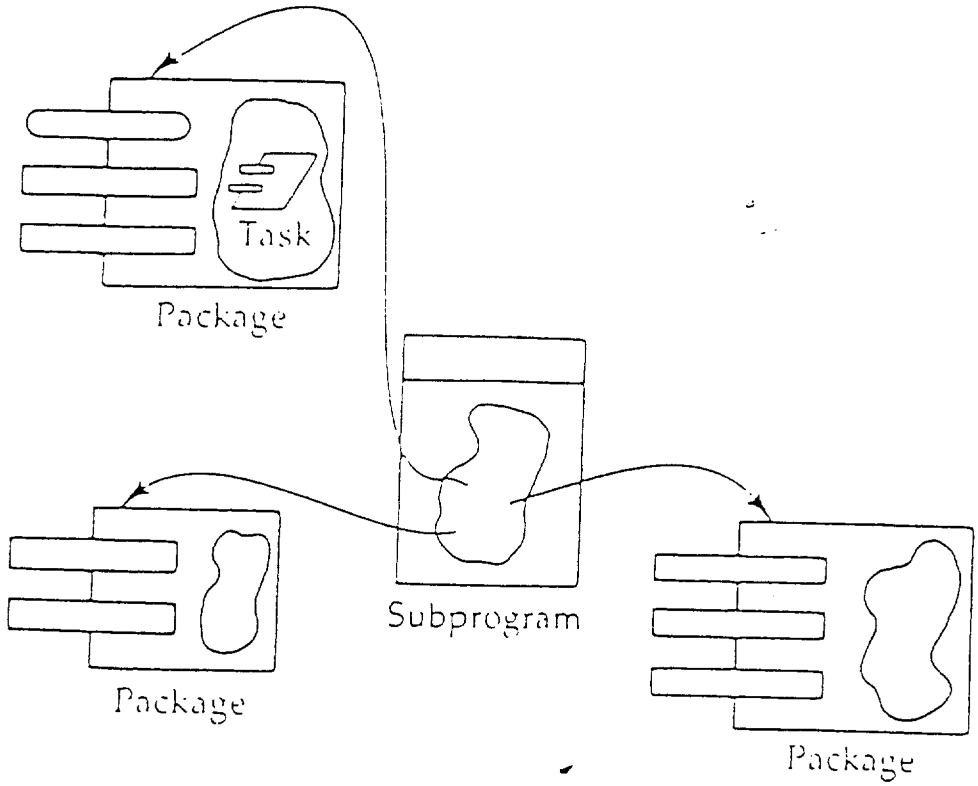


Figure 4-5 Topology of Ada

SOFTWARE LIFE CYCLE

- o REQUIREMENTS ANALYSIS
- o SYSTEM DESIGN
- o IMPLEMENTATION
- o VALIDATION/TESTING
- o OPERATION/MAINTENANCE

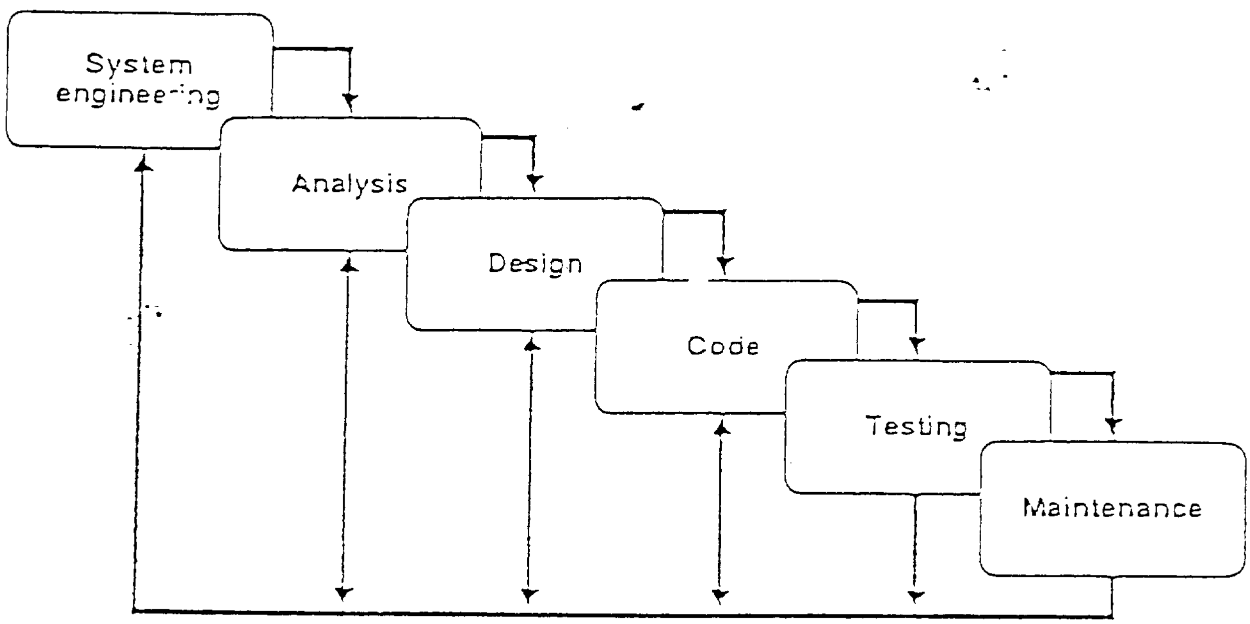


FIGURE 1.9
The classic life cycle.

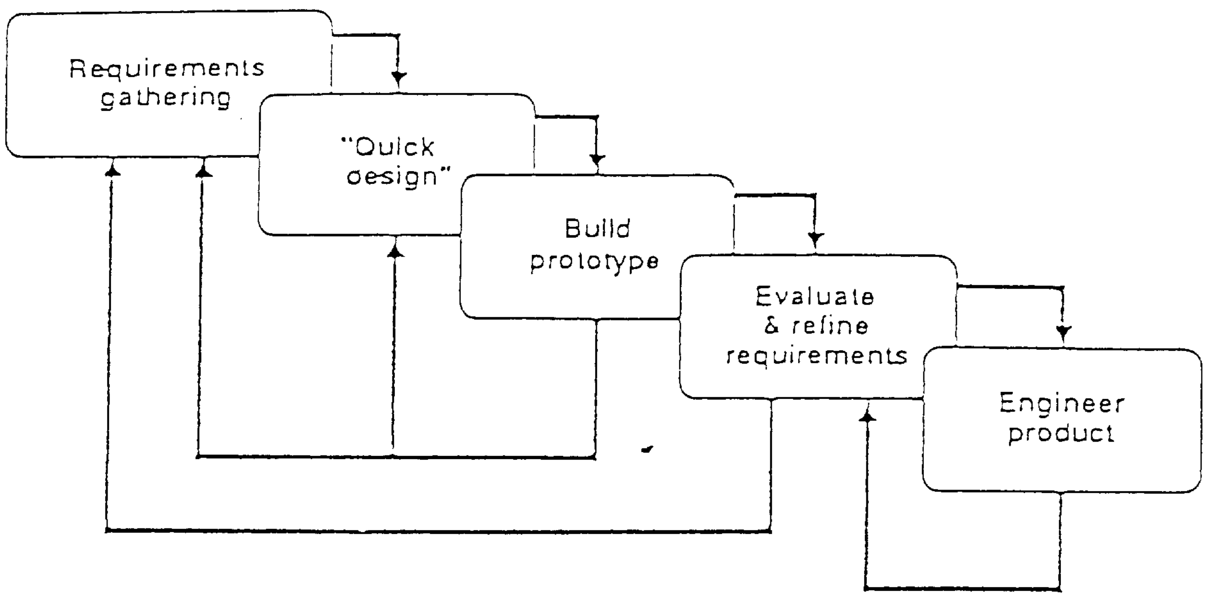


FIGURE 1.10
Prototyping.

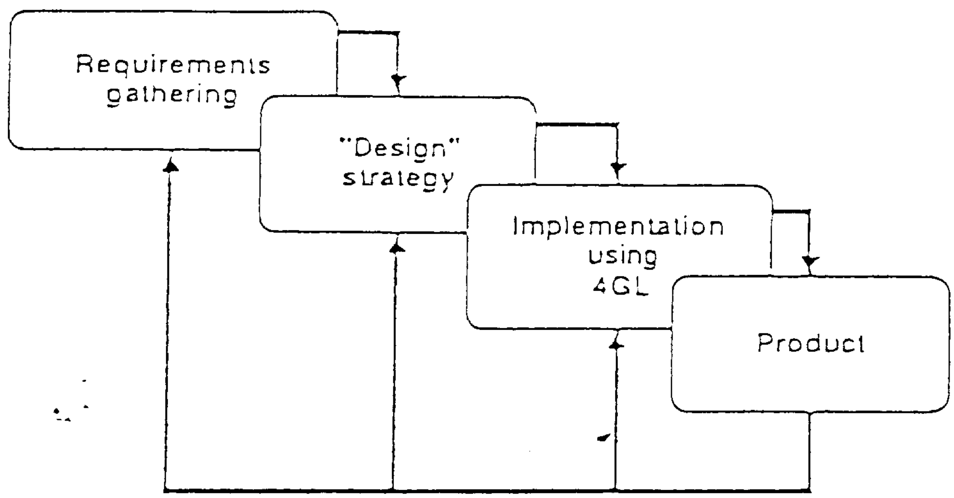


FIGURE 1.11
Fourth generation techniques.

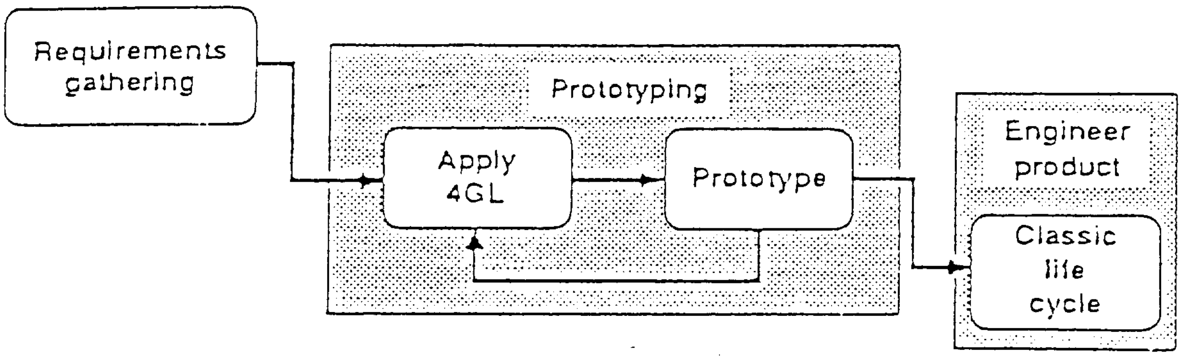


FIGURE 1.13
Combining paradigms.

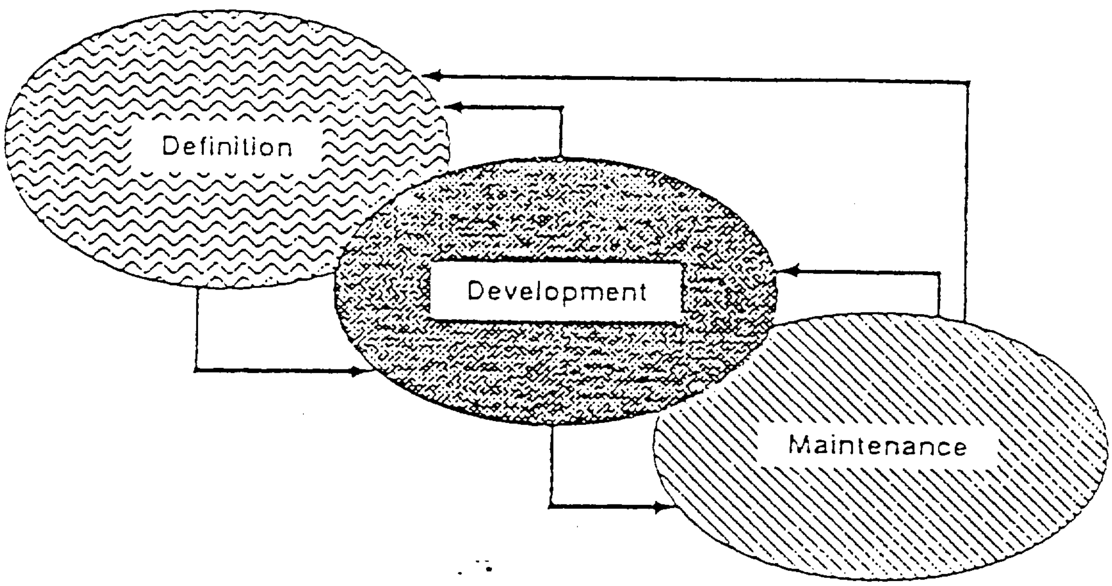


FIGURE 1.14
Paradigms: a generalized view.

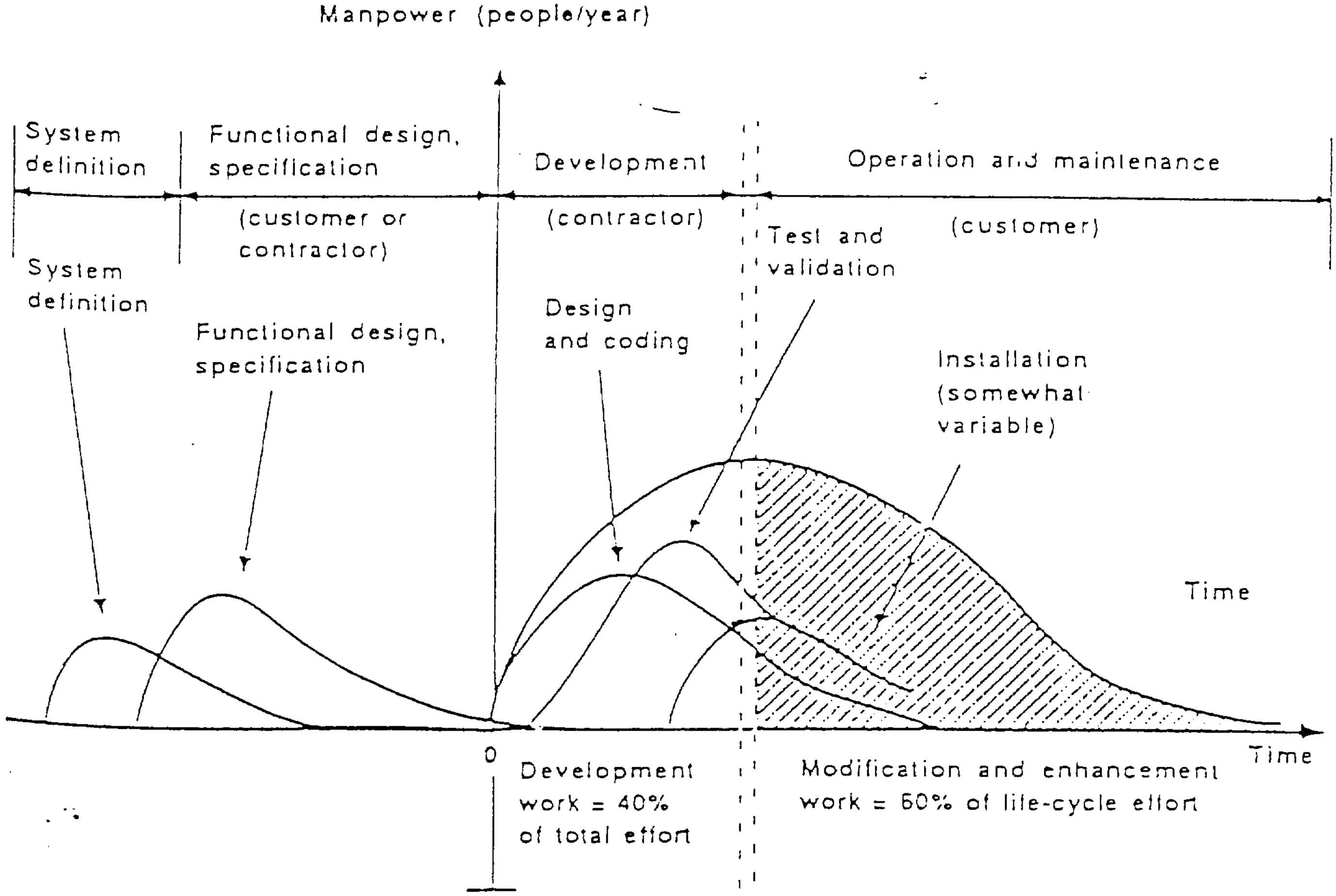


FIGURE 3.18
Effort distribution—large projects. (*Source*: L. Putnam. *Software Cost Estimating and Life Cycle Control*. IEEE Computer Society Press. 1980. p. 15. Reproduced with permission.)

REQUIREMENTS ANALYSIS/DEFINITION

- o FEASIBILITY
- o ANALYSIS
- o DEFINITION

SYSTEM DESIGN

- o HIGH LEVEL DESIGN
- o INTERFACE DESIGN
- o DESIGN SPECIFICATION
- o DETAIL/ALGORITHMIC

IMPLEMENTATION

o CODE DESIGN

VALIDATION/TESTING

- o UNIT
- o SUBSYSTEM TESTING/INTEGRATION
- o OPERATIONAL
- o ACCEPTANCE

OPERATION/MAINTENANCE

- o HONEYMOON
- o ERROR CORRECTION
- o EVOLUTIONARY
- o PERFECTIVE MAINTENANCE

SOFTWARE ENGINEERING GOALS

- o MODIFIABILITY
- o EFFICIENCY
- o UNDERSTANDABILITY
- o RELIABILITY

RELIABLE SOFTWARE

- o FOUR APPROACHES
  - AVOIDANCE
  - DETECTION
  - CORRECTION
  - TOLERANCE
- o COMPLEXITY
- o DESIGN CORRECTNESS

SOFTWARE PRINCIPLES

- o ABSTRACTION
- o INFORMATION HIDING
- o MODULARITY
- o LOCALIZATION
- o UNIFORMITY
- o COMPLETENESS
- o CONFIRMABILITY

DESIGNING THE SOLUTION

- o DERIVE DATA, PROGRAM STRUCTURE, AND PROCEDURAL DETAIL
- o UNDERSTAND STRUCTURAL ELEMENTS THAT COMPRISE SOFTWARE ARCHITECTURE
- o UNDERSTAND DATA STRUCTURES AND IMPACT ON SOFTWARE DESIGN
- o DIFFERENCE BETWEEN ARCHITECTURAL AND PROCEDURAL
- o IMPORTANCE OF REFINEMENT
- o IMPORTANCE OF MODULARITY
- o IMPORTANCE OF INFORMATION HIDING
- o MODULARITY AND FUNCTIONAL INDEPENDENCE
- o USE ABSTRACTION

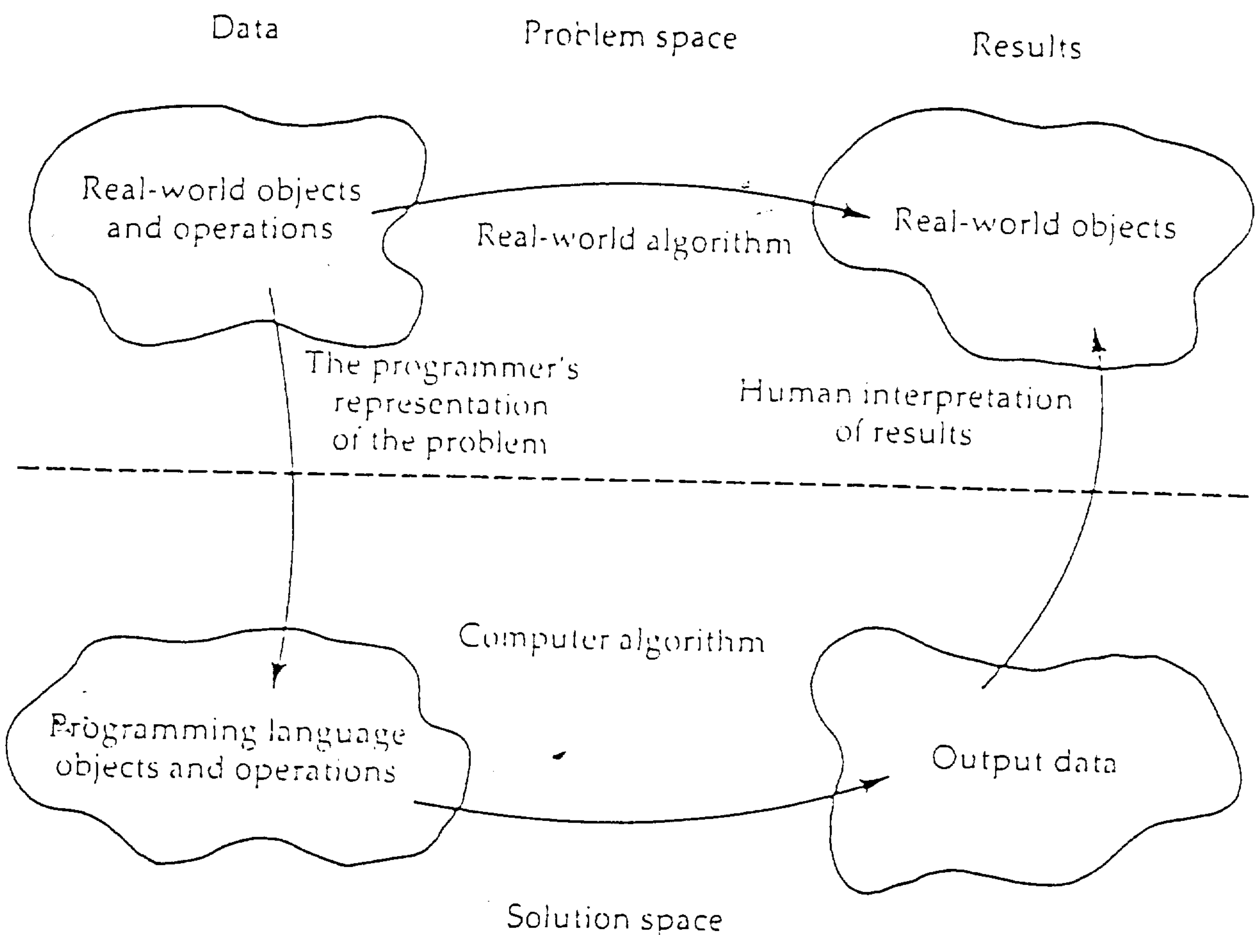


Figure 5-1 Model for a typical programming task. (From *The Programming Language Landscape* by Henry Ledgard and Michael Marcotty © 1981 Science Research Associates, Inc. Reproduced by permission of the publisher.)

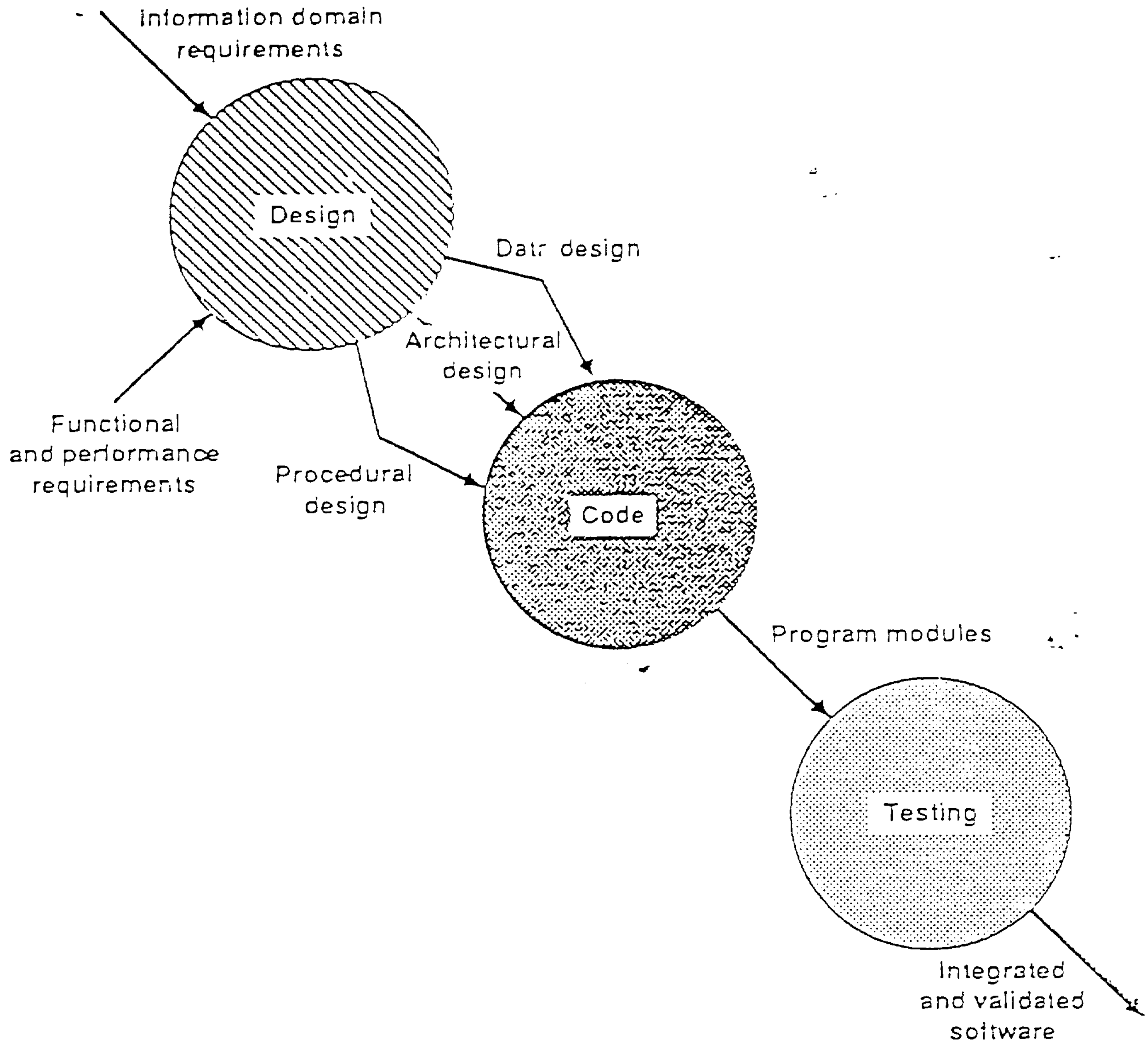
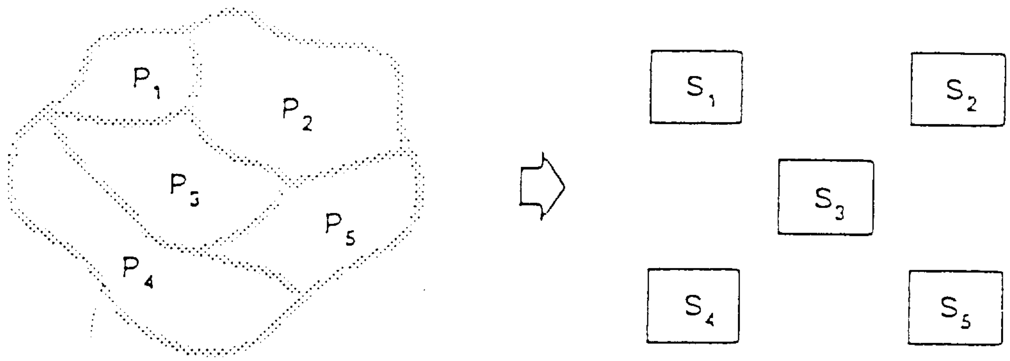


FIGURE 6.1
The development phase.



"Problem" to be solved via software

Software "solution"

FIGURE 6.4
Evolution of structure.

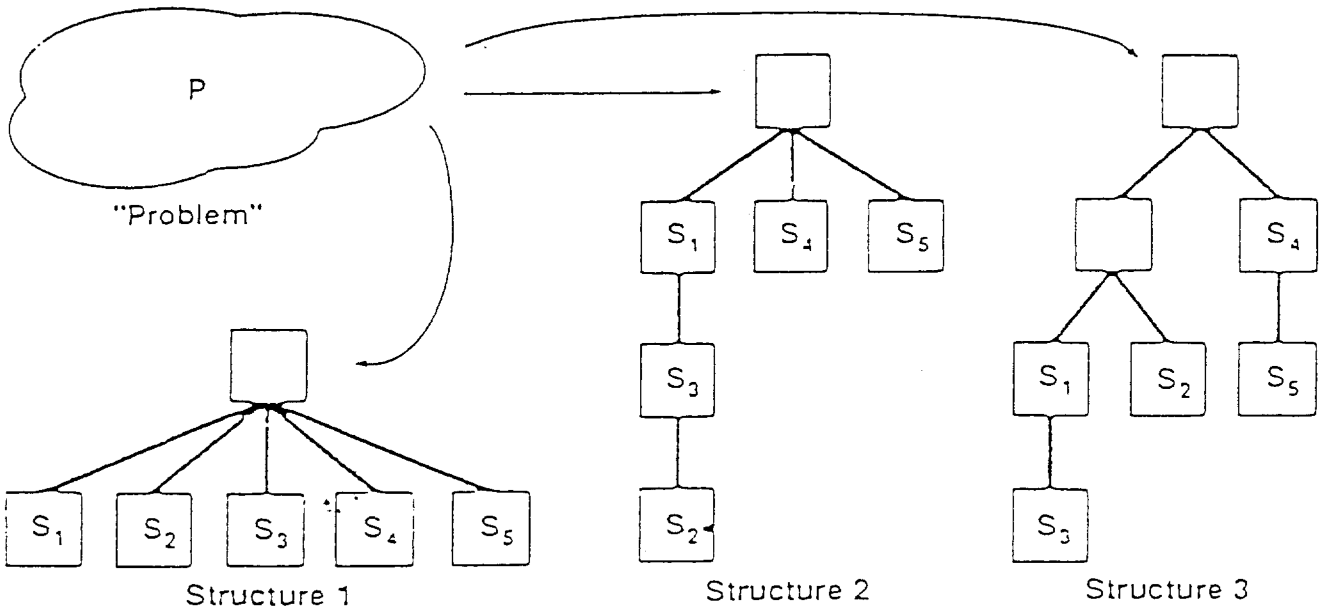


FIGURE 6.5
Different structures.

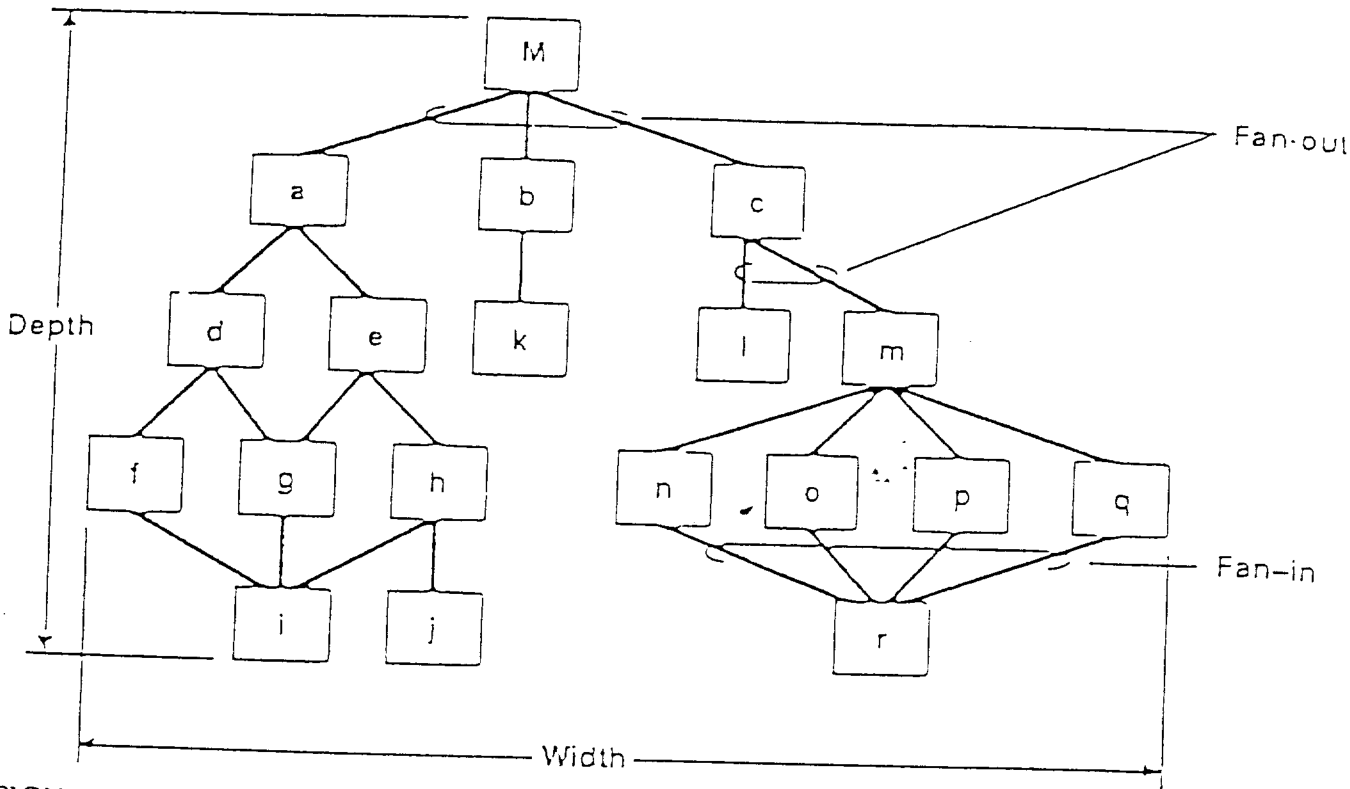


FIGURE 6.6
Structure terminology.

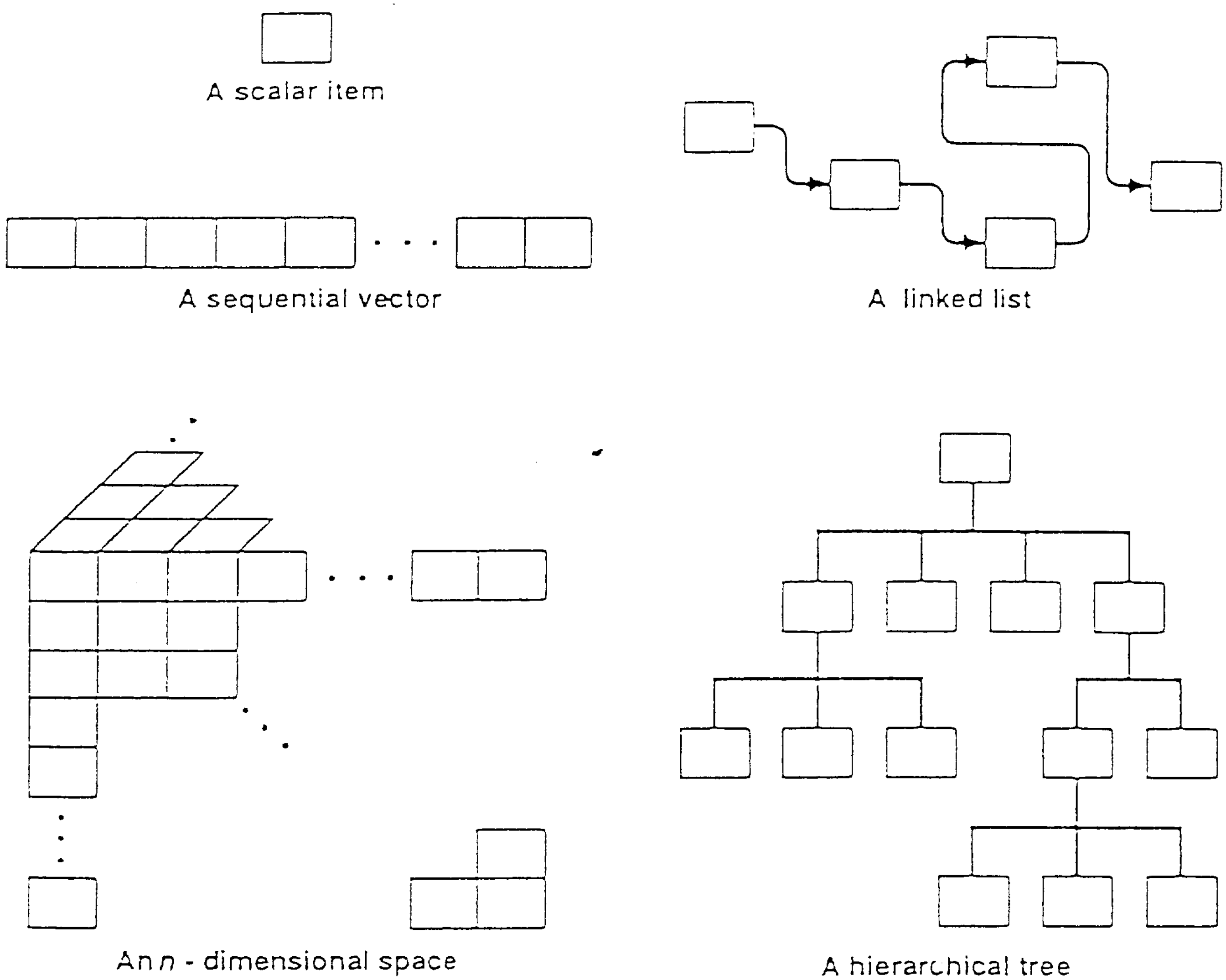


FIGURE 6.7
Classic data structures.

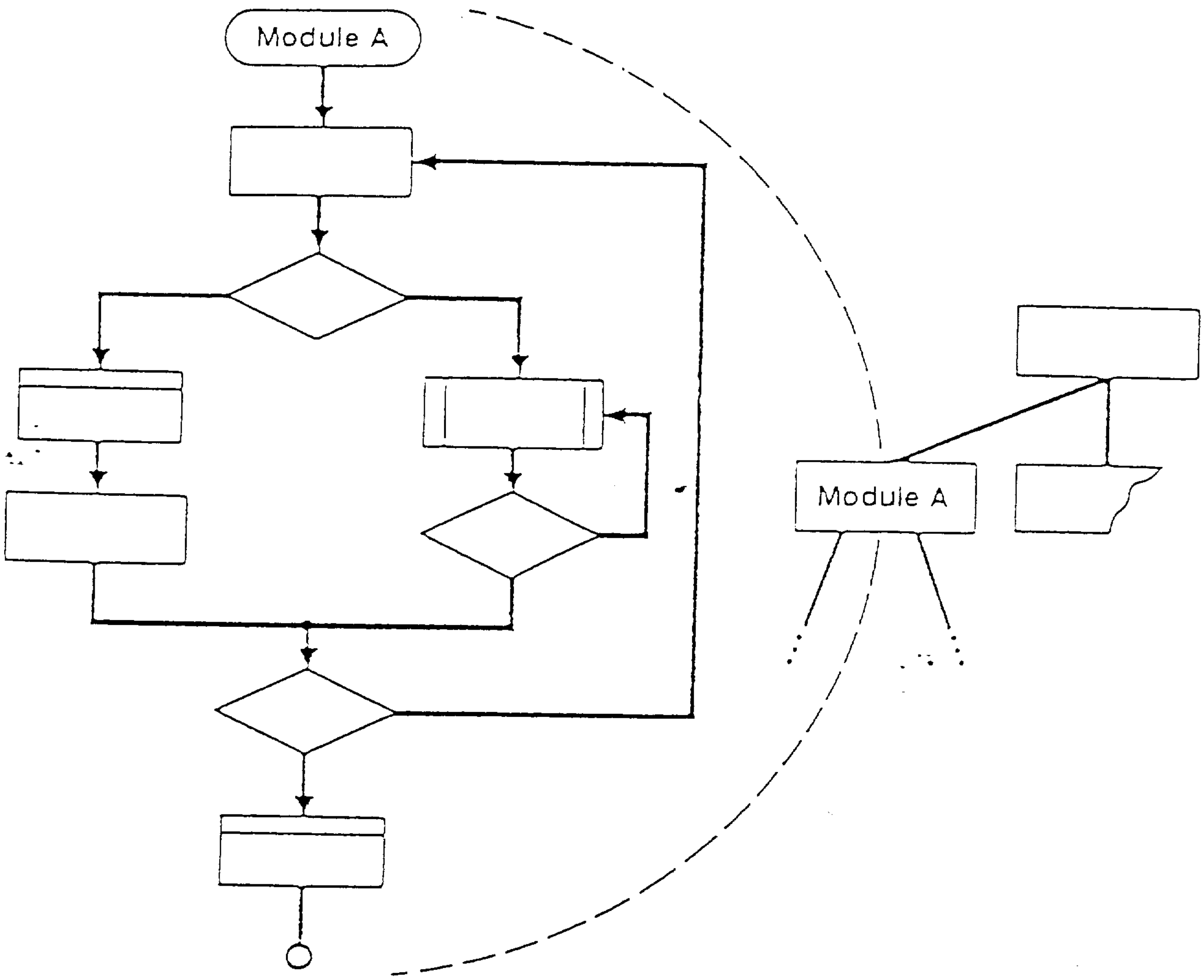


FIGURE 6.8
Procedure within a module.

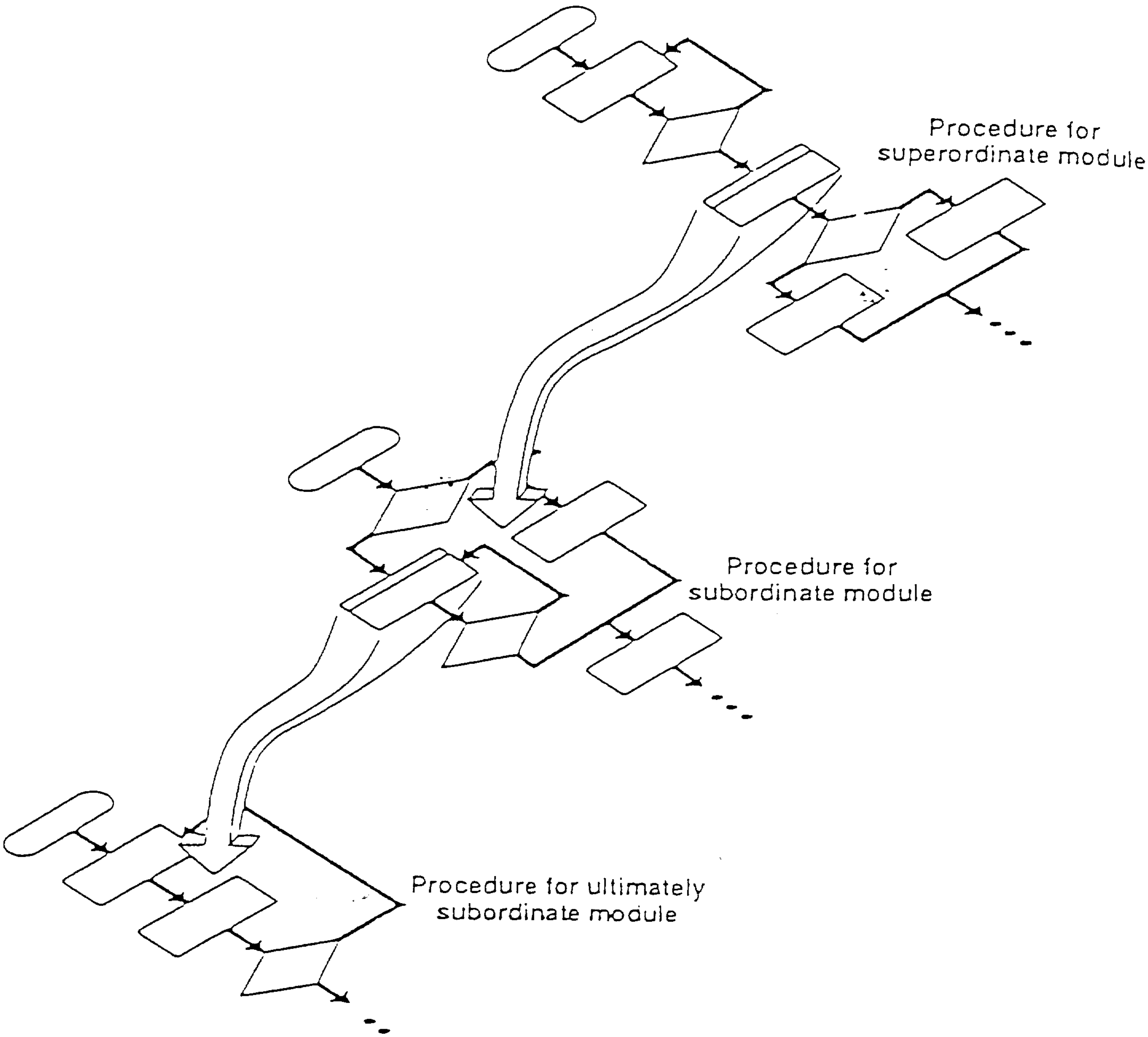


FIGURE 6.9
Procedure is layered.

ABSTRACTION 1. The software will incorporate a computer graphics interface that will enable visual communication with the draftsperson and a digitizer interface that replaces the drafting board and square. All line and curve drawing, all geometric computations, and all sectioning and auxiliary views will be performed by the CAD software.

ABSTRACTION 2.

```
CAD software tasks:
    user interaction task;
    2-D drawing creation task;
    graphics display task;
    drawing file management task;
end.
```

ABSTRACTION 3.

```
procedure: 2-D drawing creation;
    repeat until (drawing creation task terminates)
    do while (digitizer interaction occurs)
        digitizer interface task;
        determine drawing request case;
            line: line drawing task;
            circle: circle drawing task;
            .
            .
            .
end;
do while (keyboard interaction occurs)
    keyboard interaction task;
    process analysis/computation case;
        view: auxiliary view task;
        section: cross sectioning task;
        .
        .
        .
    end;
    .
    .
    .
    end repetition;
end procedure.
```

INFORMATION HIDING

- MODULES HIDE SOME DESIGN DECISION
- DESIGN CHANGES LATER NO USERS OF MODULE NEED TO CHANGE

Module:	Sequence
Secret:	Representation of sequence
Change:	Alternate between linked-list and array implementations
Module:	Sort routine
Secret:	Algorithm for sorting
Change:	Use different algorithm tuned to size of data or expected degree of sortedness
Module:	Disk request handler
Secret:	Policy for picking next disk access
Change:	Favor certain kinds of requests to improve overall system performance
Module:	Panel indicator interface
Secret:	How indicator display modes (on, off, blinking) are implemented
Change:	Replace device that can blink with one that cannot, so you must simulate blinking by sequences of on/off

Figure 5–3: Some Typical Secrets

```
stack : record sp:integer; arr:array[1..Max] of element end;
procedure Push(e:element) is
   begin stack.sp := stack.sp + 1; stack.arr[stack.sp] := e; end;
procedure Pop is
   begin stack.sp := stack.sp-1; end;
function Top return element is
   begin return stack.arr[stack.sp]; end;
procedure Init is
   begin stack.sp := 0; end;
```

Figure 5–4: Stack Implemented as an Array

```
type stackptr;
type stackelement is record elem:element; next: stackptr; end record;
type stackptr is access stackelement;
stack : stackptr;
procedure Push(e:element) is
      item : stackptr;
   begin
      item := new stackelement; item.elem := e;
      item.next := stack; stack:=item;
   end;
procedure Pop is
   begin stack := stack.next; end;
function Top return element is
   begin return stack.item; end;
procedure Init is
   begin stack := null; end;
```

Figure 5–5: Stack Implemented as a List

INFORMATION HIDING VS ABSTRACTION

- o INFORMATION HIDING FOCUSES ON WHAT TO HIDE
- o ABSTRACTION FOCUSES ON WHAT TO REVEAL

PROGRAM STRUCTURE DESIGN

- o MODULE INDEPENDENCE
- o MODULE STRENGTH
- o MODULE COUPLING

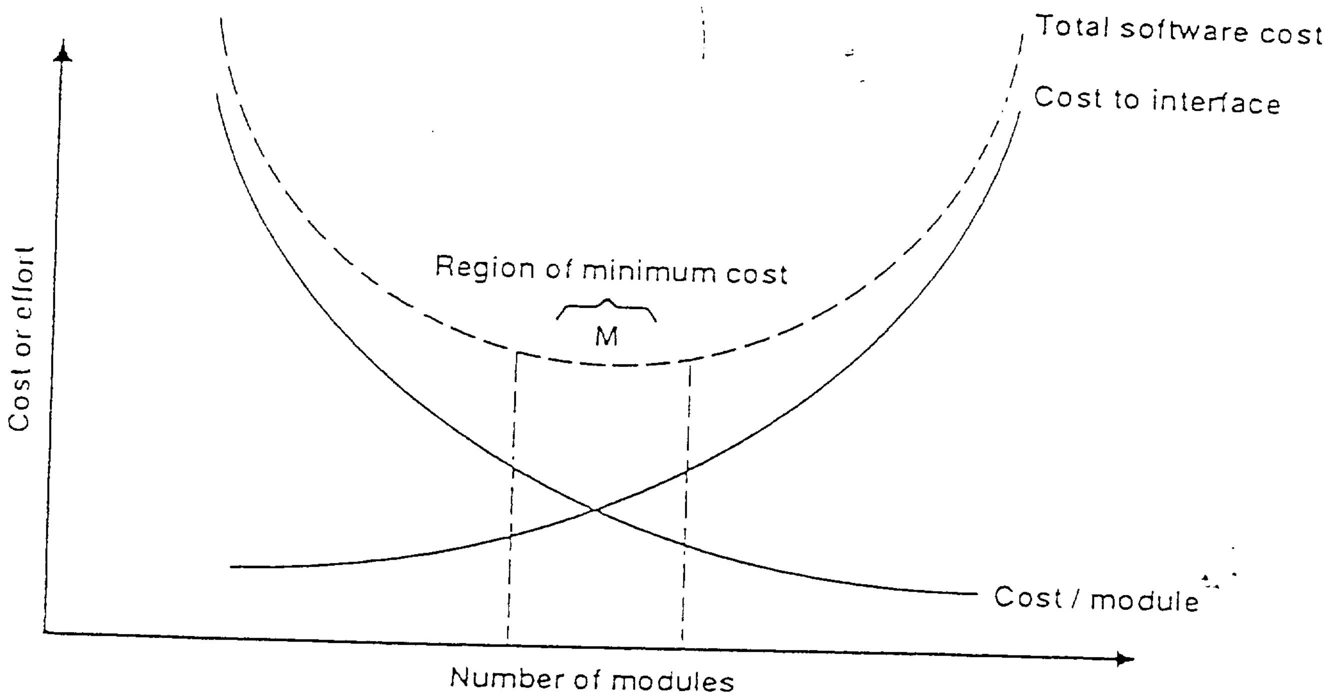


FIGURE 6.10
Modularity and software cost.

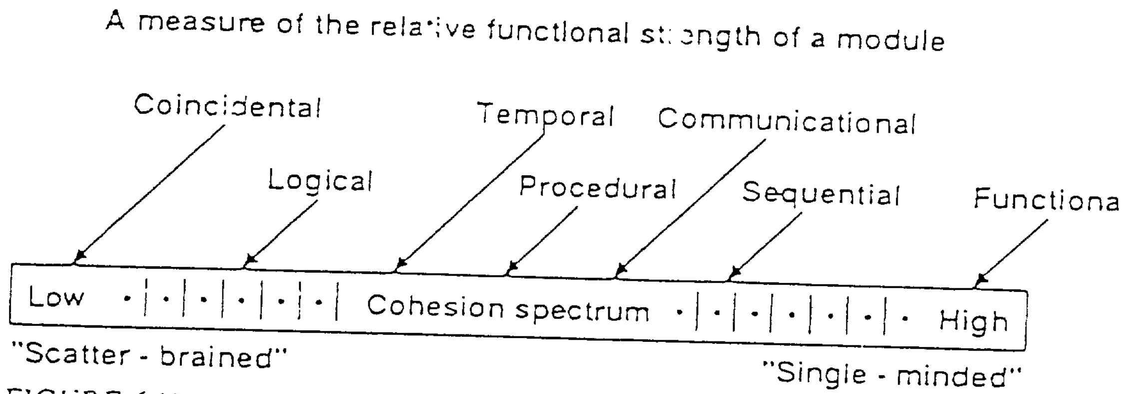
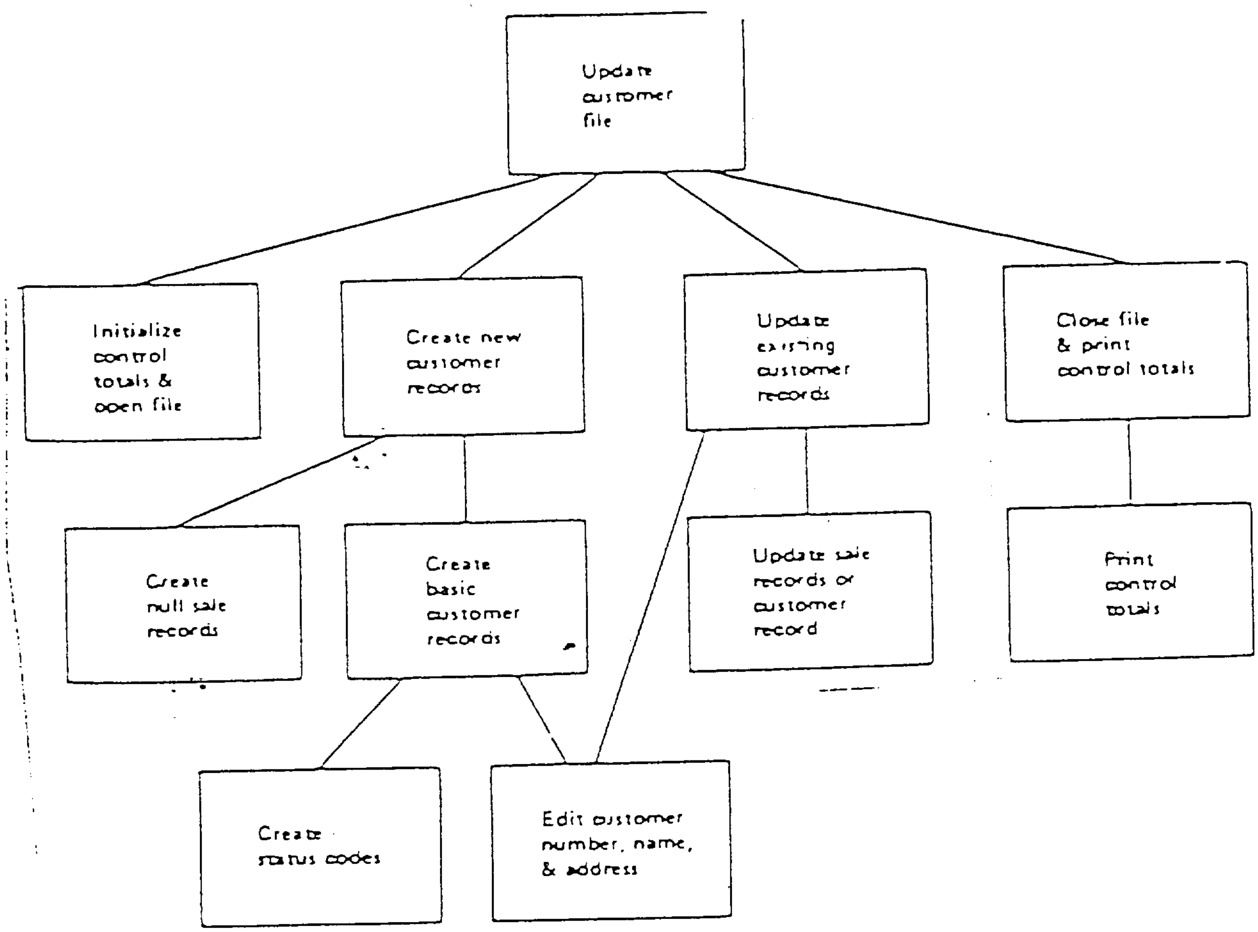


FIGURE 6.11
Cohesion.


Update customer file
Initialize control totals & open file
Create new customer records
Update existing customer records
Close file & print control totals
Create null sale records
Create basic customer records
Update sale records or customer record
Print control totals
Create status codes
Edit customer number, name, & address