SOFTWARE ENGINEERING-18BIT41C

UNIT I: Introduction to Software Engineering: Some Definitions – Some Size factors – Quality and Productivity Factors – Managerial Issues. Planning a Software Project: Defining the Problem – Developing a Solution Strategy – Planning the Development Process – Planning an Organizational Structure – Other Planning Activities.

UNIT II: Software Cost Estimation: Software Cost Factors – Software Cost Estimation Techniques – Staffing-Level Estimation – Estimating Software Maintenance Costs.

UNIT III: Software Requirements Definitions: The Software Requirements Specification – Formal Specification Techniques – Languages and Processors for Requirements Specification.

UNIT IV: Software Design: Fundamental Design Concepts – Modules and Modularization Criteria – Design Notations – Design Techniques – Detailed Design Considerations – Real-Time and Distributed System Design – Test Plans – Milestones, Walkthroughs, and Inspections - Design Guidelines.

UNIT V: Verification and Validation Techniques: Quality Assurance – Static Analysis – Symbolic Execution – Unit Testing and Debugging – System Testing – Formal Verification.

Software Maintenance: Enhancing Maintainability During Development – Managerial Aspects of Software Maintenance – Configuration Management – Source-Code Metrics – Other Maintenance Tools and Techniques.

TEXT BOOK

1. “Software Engineering Concepts” – Richard Fairley – Tata McGraw - Hill Publishing Company Limited, NewDelhi 1997.

Introduction to Software Engineering: Some Definitions – Some Size factors – Quality and Productivity Factors – Managerial Issues. Planning a Software Project: Defining the Problem – Developing a Solution Strategy – Planning the Development Process – Planning an Organizational Structure – Other Planning Activities

SOFTWARE ENGINEERING -INTRODUCTION

Develops a software product

- User needs and constraints stated
- Product must accommodate implementers, users and maintainers
- Source code tested
- Maintenance, user manual, installation instructions, training aids and maintenance of documents must be prepared
- Software maintenance- change requests, redesign and modification of source code through testing of the modified code
- Appropriate user sites

Software products- 1960

Ex: airline, train, medical and other general purpose

Definition

Software Engineering is the technical and management discipline concerned with systematic production and maintenance of software products that are developed and modified on time within cost estimates.

Programmers- Implementing, packaging and modifying algorithms.

Software engineers- analysis design, verification and validation, testing, documentation, software maintenance and project management.

SIZE FACTORS

1. Total effort devoted to software:

- 1955- Hardware cost 80% high
- 1980- Reverse

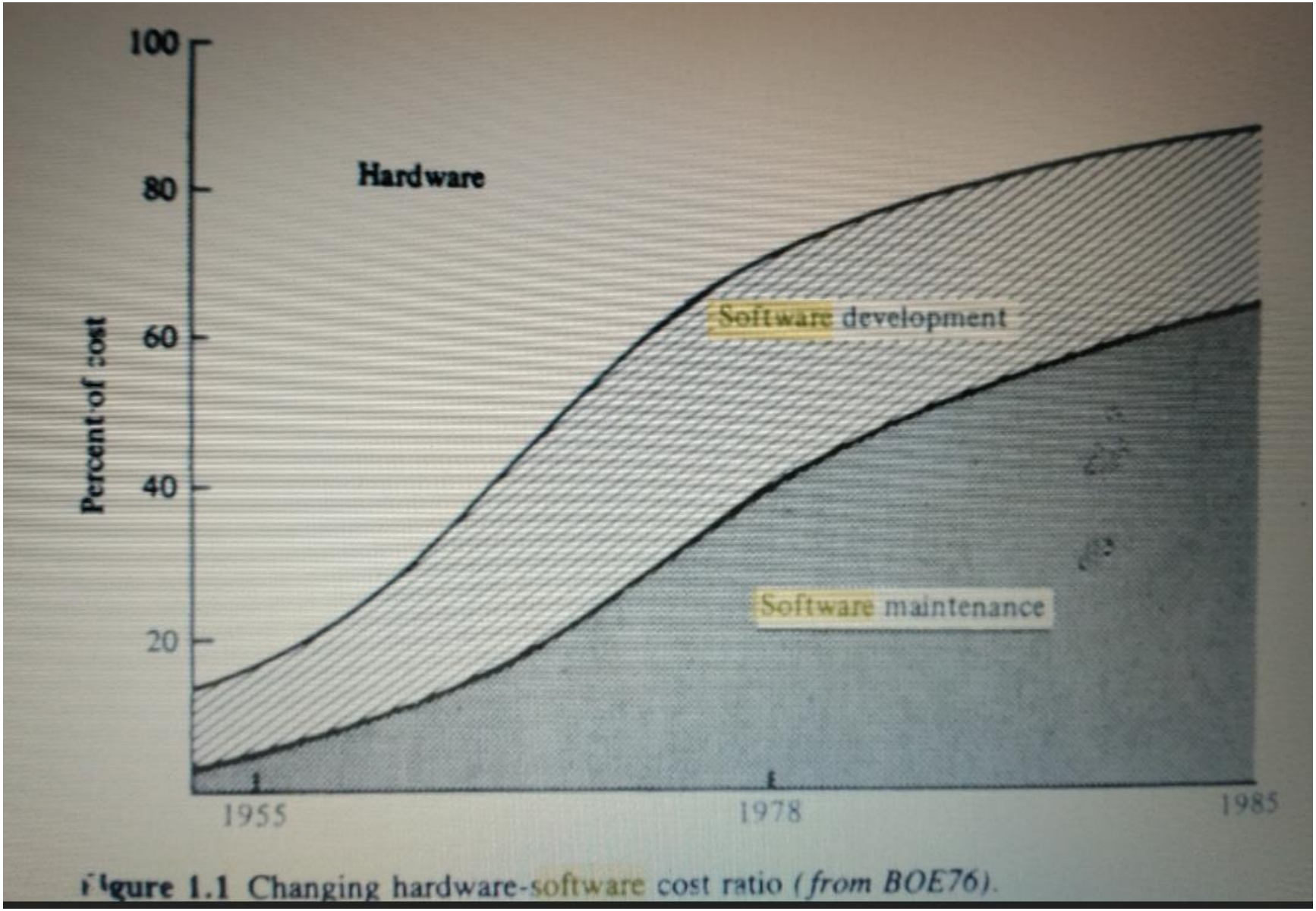


Figure 1.1 Changing hardware-software cost ratio (*from BOE76*).

2. Distribution of Effort

- 1 to 3 years development and 15 years maintenance
- The distribution effort 40/60, 30/70, 10/90

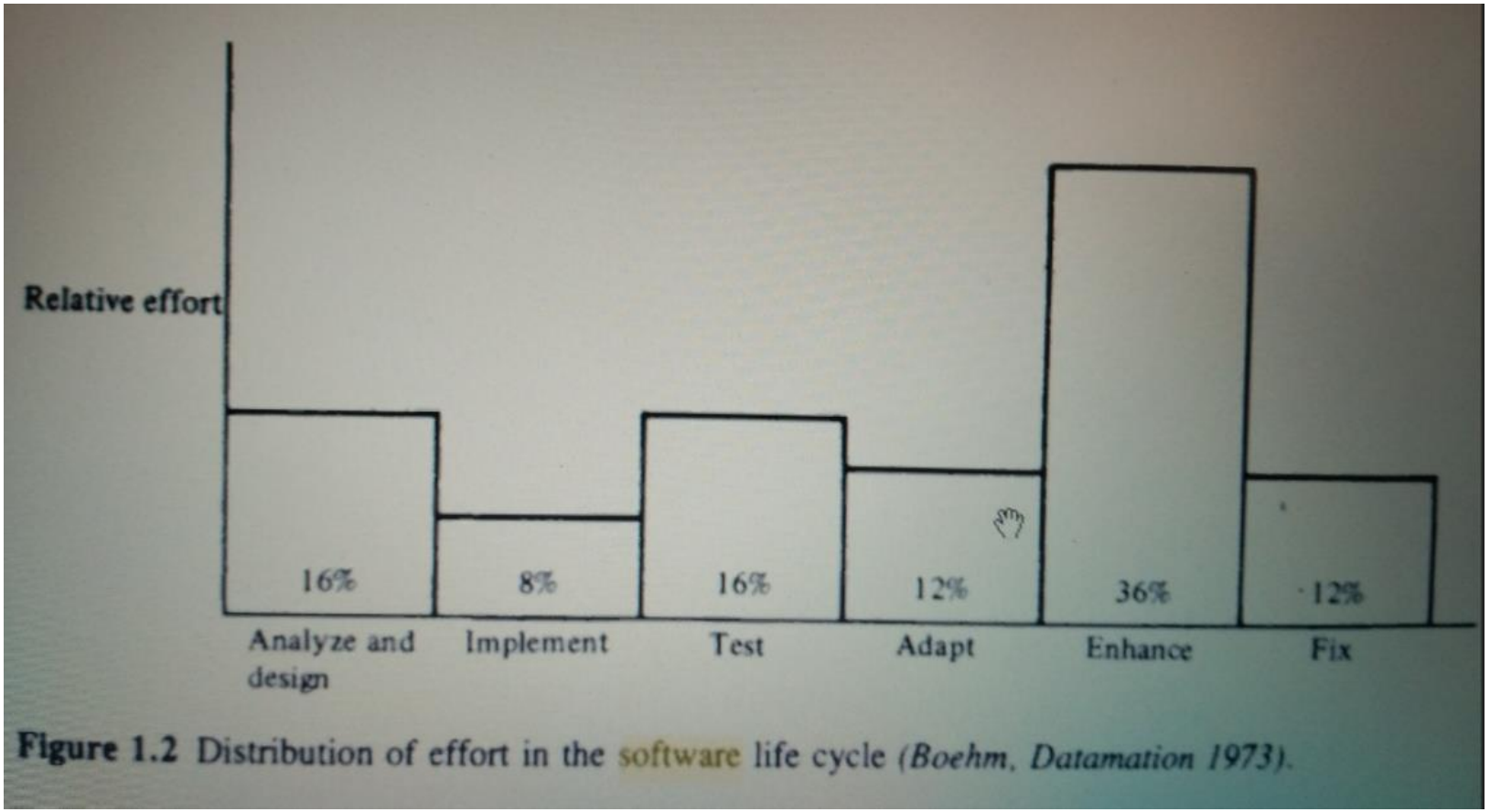


Figure 1.2 Distribution of effort in the software life cycle *(Boehm, Datamation 1973)*.

3. Project Size Categories

Determines the level of management control and the types of tools and techniques required on a software project.

Table 1.1 Size categories for software products

Category	Number of programmers	Duration	Product size
Trivial	1	1–4 wks.	500 source lines
Small	1	1–6 mos.	1K–2K
Medium	2–5	1–2 yrs.	5K–50K
Large	5–20	2–3 yrs.	50K–100K
Very large	100–1000	4–5 yrs.	1M
Extremely large	2000–5000	5–10 yrs.	1M–10M

Trivial: One programmer working part time, develop the software for the exclusive use of the programmer and one usually discarded after a few months.

Small projects: Scientific application written by engineers to solve numerical problems.

Medium Size projects: Assemblers, Compliers, Small management Systems, inventory, process control applications.

Large Projects : Large compilers, time-sharing systems, data base packages, graphics programs for data acquisition and display.

Very large projects: air traffic control, missile, defence and military command and control systems

4. How programmers spend their time:

Writing programs	18%
Reading programs and manuals	16%
Job communication	32%
Personal	13%

Miscellaneous	15%
Training	6%
Mail	5%

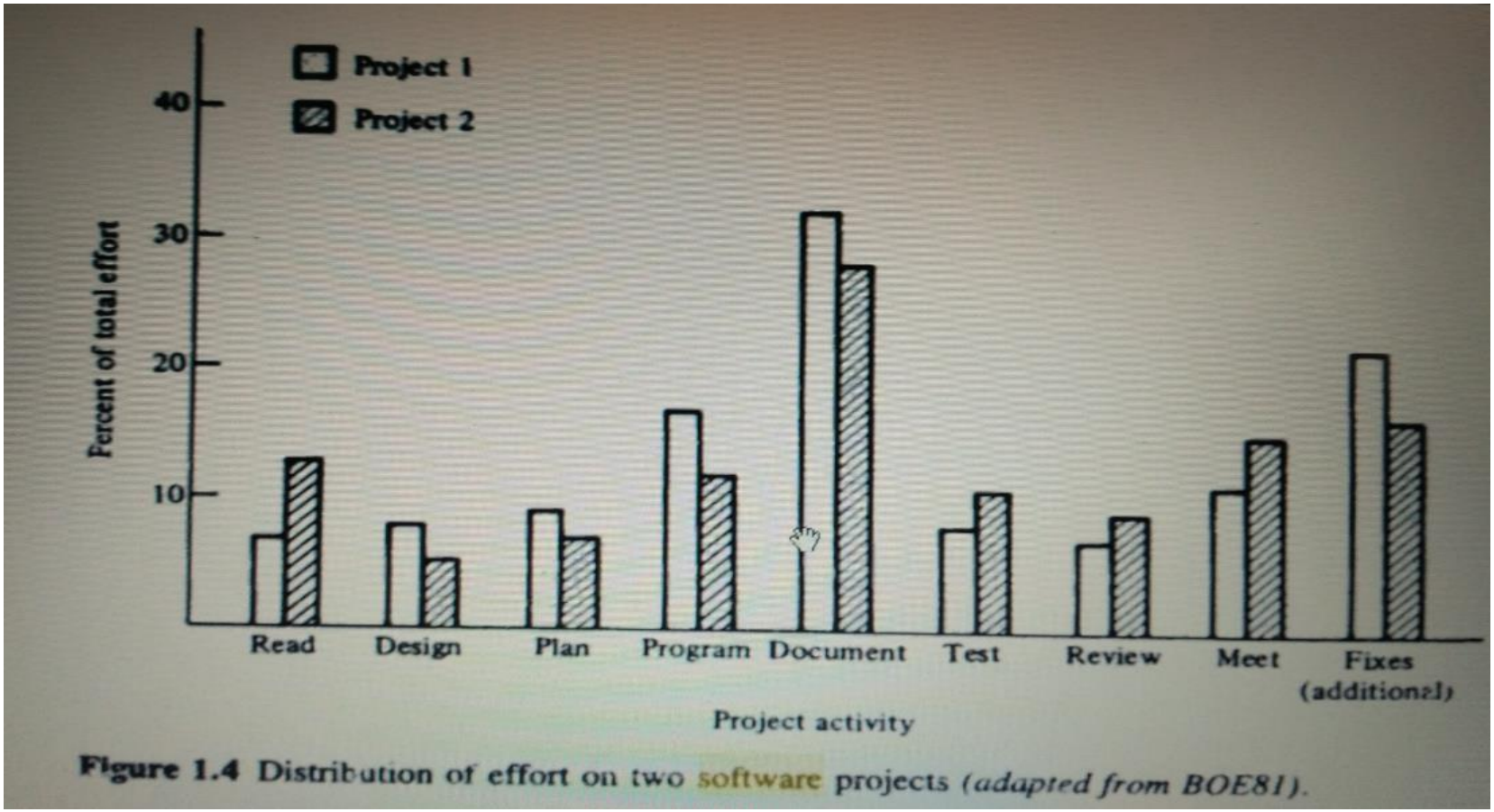


Figure 1.4 Distribution of effort on two software projects *(adapted from BOE81)*.

QUALITY AND PRODUCTIVITY FACTORS

Writing small programs for personal use and developing or modifying a software product.

Ex: Electrical engineer assembling the components

High quality software → Technical and managerial skill

Factors that influence quality and productivity

1. Individual level:

Lack of Competence → Poor productivity and quality

Two Aspects → General Competence of individual → Ability to write correct programs

Two Aspects → Familiarity of the individual

Experiment by Sackman

12 programmers implemented 2 programs

	Performance measure	Ratio
Debugging hours	#1	28:1
Debugging hours	#2	26:1
Runtime	#1	5:1
Runtime	#2	13:1

2. Team Communication

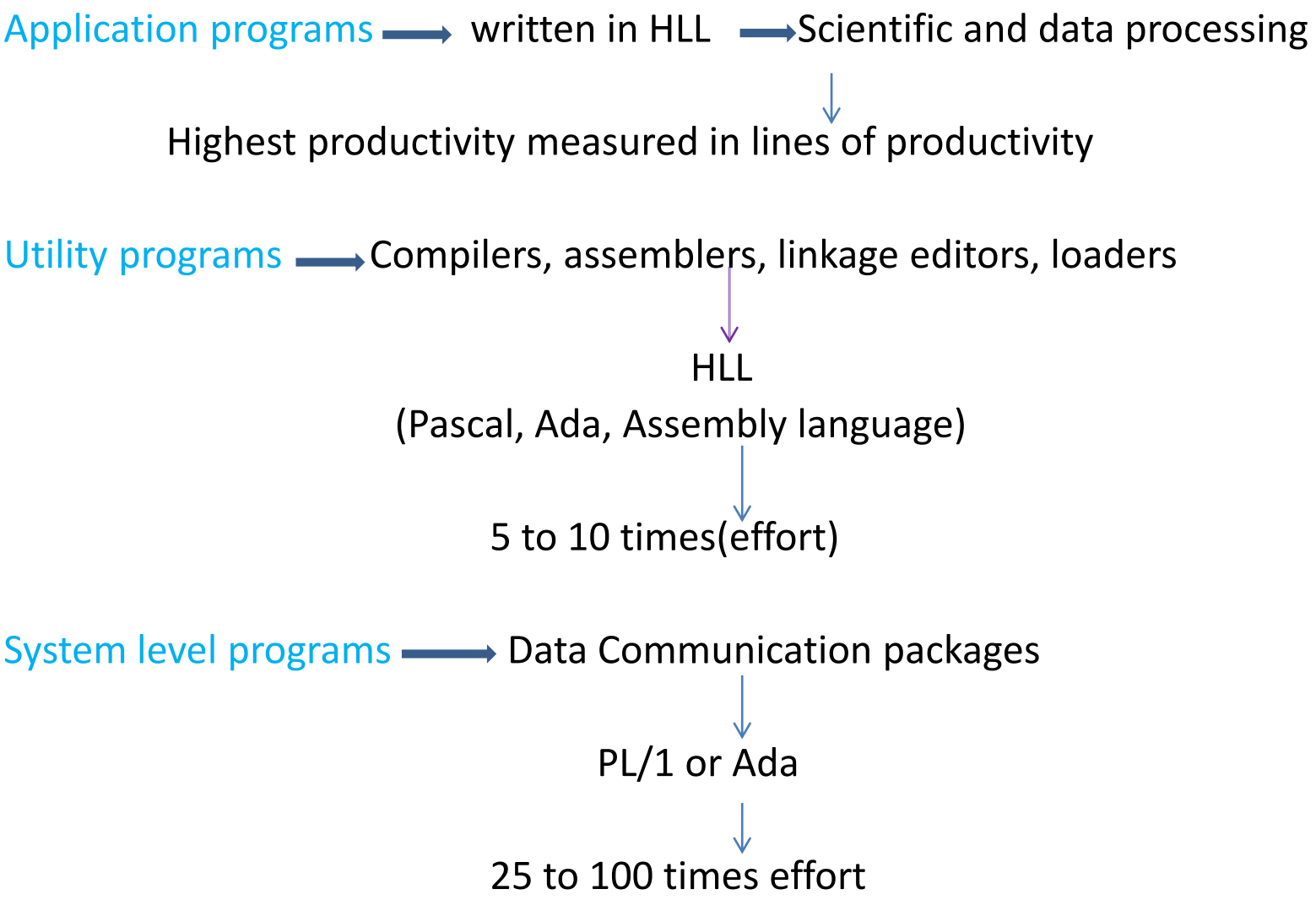
- Many programmers low social needs prefer to work alone
- Increased product size results in decreasing programmer
- Brooks- number of paths among the programmers is n(n-1)/2

n= number of programmers

If n=3 3(3-1)/2=6/2=3

3. Product Complexity:

Three level of product complexity.


Application programs
written in HLL
Scientific and data processing
Highest productivity measured in lines of productivity
Utility programs
Compilers, assemblers, linkage editors, loaders
HLL
(Pascal, Ada, Assembly language)
5 to 10 times(effort)
System level programs
Data Communication packages
PL/1 or Ada
25 to 100 times effort

4. Appropriate Notations:

Good notations clarify the relationships

Vehicle of communication among the project personnel

5. Systematic approaches

use accepted procedures and techniques

6. Change Control

Software adapts general purpose computing hardware to particular applications

7. Level of Technology

- Hardware and software facilities available for developing ,using and maintain a software product.
- Modern programming practices include use of systematic analysis and design techniques, appropriate notations, structured coding, systematic techniques for examining design documents and source code.

8. Level of Reliability:

- Basic level of reliability
- Extreme Reliability – Great care in analysis, design, implementation, system testing and maintenance of software product.

9. Problem Understanding:

- Not understanding problem leads to limitations
- Careful planning, customer interviews, task observation, user manual, product specifications can increase both customers and developer understand the problem.

10. Available time

Six programmer effort can be completed in 6 months or six programmers in one month.

11. Required skills

Good communication skills, writing text book-no misspellings.

No errors in syntax or punctuation.

Absolute consistency in cross referencing.

12. Facilities and Resources:

Good machine access, quiet place to work, reserved parking places, rest room, professional advancement.

13. Adequacy of training:

Entry level programmers

- Express oneself clearly in English
- Develop and validate software requirements and design specifications
- Work within application area
- Perform economic analysis
- Work with project management techniques
- Work in groups

14. Management Skills:

Software projects are often supervised by manager who have little, if any knowledge of software engineering. Hardware engineer should be trained software.

15. Appropriate goals: software product should provide, generality, efficiency, reliability.

16. Rising expectations

MANAGERIAL ISSUES

Managerial activities:

- managers have responsibility for ensuring that software products are delivered on time and within cost estimates.
- Developing business plans, recruiting customers, developing marketing strategies and recruiting and training employees.

Important management problems

- Planning for software engineering projects is generally poor.
- Procedures and techniques for the selection of project managers are poor.
- Responsibilities for various project functions.
- Ability to accurately estimate the resources.
- Success criteria of projects are frequently inappropriate.

- Aid in selecting the proper organizational structure are not available.
- Procedures, methods, and techniques for designing a project control system are not readily available.
- Standards and techniques for measuring the quality of performance of production expected from the programmers are not available.

Solution to solve above mentioned problems

- ❖ Educate and train top management, project managers and software developers.
- ❖ Enforce the use of standards, procedures and documentation.
- ❖ Analyse data from prior software projects to determine effective methods.
- ❖ Define objectives in terms of quality desired.
- ❖ Define quality in terms of deliverables
- ❖ Establish success priority criteria
- ❖ Allow for contingencies
- ❖ Develop truthful, accurate cost and schedule estimates

- ❖ Select project managers based on ability to manage software projects, rather than on technical ability or availability.
- ❖ Make specific work assignment to software developers and apply job performance standards.

PLANNING A SOFTWARE PROJECT

I. DEFINING THE PROBLEM

1. The problem statement should be phrased in the customers terminology
2. Justify a computerised solution strategy for the problem
3. Identify the functions to be provided – hardware and software subsystems
4. Determine system level goals and requirements
5. Establish high level acceptance criteria for the system

Developing a solution strategy

6. Outline several solution strategies, without regard for constraints

7. Conduct a feasibility study for each strategy

8. Recommend a solution strategy, indicating why other strategies were rejected.

9. Develop a list of priorities for product characteristics

Planning the development process

10. Define a life-cycle model and an organizational structure for the project.

11. Plan the configuration management, quality assurance and validation activities.

12. Determine phase-dependent tools, techniques and notations to be used.

13. Establish preliminary cost estimates for system development

14. Establish a preliminary development schedule

15. Establish preliminary staffing estimates

16. Preliminary estimates to operate and maintain the system

17. Prepare a glossary of terms

18. Identify sources of information, and refer to them throughout the project plan.

A. GOALS AND REQUIREMENTS

- ✓ After identification of problem preliminary goals and requirements formulated.
- ✓ Goals ⟶ Targets of achievement

Goals	
Qualitative	Quantitative
System enhance the professional skill	System should be delivered within 12 months
System should make users job interesting	The system should reduce the cost of a transaction by 25%

- ✓ Some goals apply to every project and every product
- ✓ Software product should be reliable, useful, understandable and cost effective
- ✓ Other goals ex: ease of use of nonprogrammers depend on the particular situation

Requirements: System requirements

Functional requirements

Performance requirements

Requirements of hardware, firmware and software

Requirements	
Quantified requirements	Qualitative requirement
Phase accuracy shall be within 0.5 degrees	Accuracy shall be sufficient to support mission
Response to external interrupts shall be 0.25 second maximum	System shall provide real-time response
System shall reside in 50k bytes of memory	System shall make efficient use of primary memory
	System shall be 99% reliable

High level goals expressed in terms of quality attributes

Software Quality characteristic tree

II. DEVELOPING A SOLUTION STRATEGY

- ✓ Several strategies should be considered. One solution strategy must be chosen by the planners to do feasibility study.
- ✓ Feasibility study examine the solution constraints
- ✓ Feasibility of a solution include

  case studies

  Worst case analysis

III. PLANNING THE DEVELOPING PROCESS

Software life cycle models

a. phased life-cycle model

b. prototype model/milestones, documents and reviews

c. Cost model

d. Successive versions model

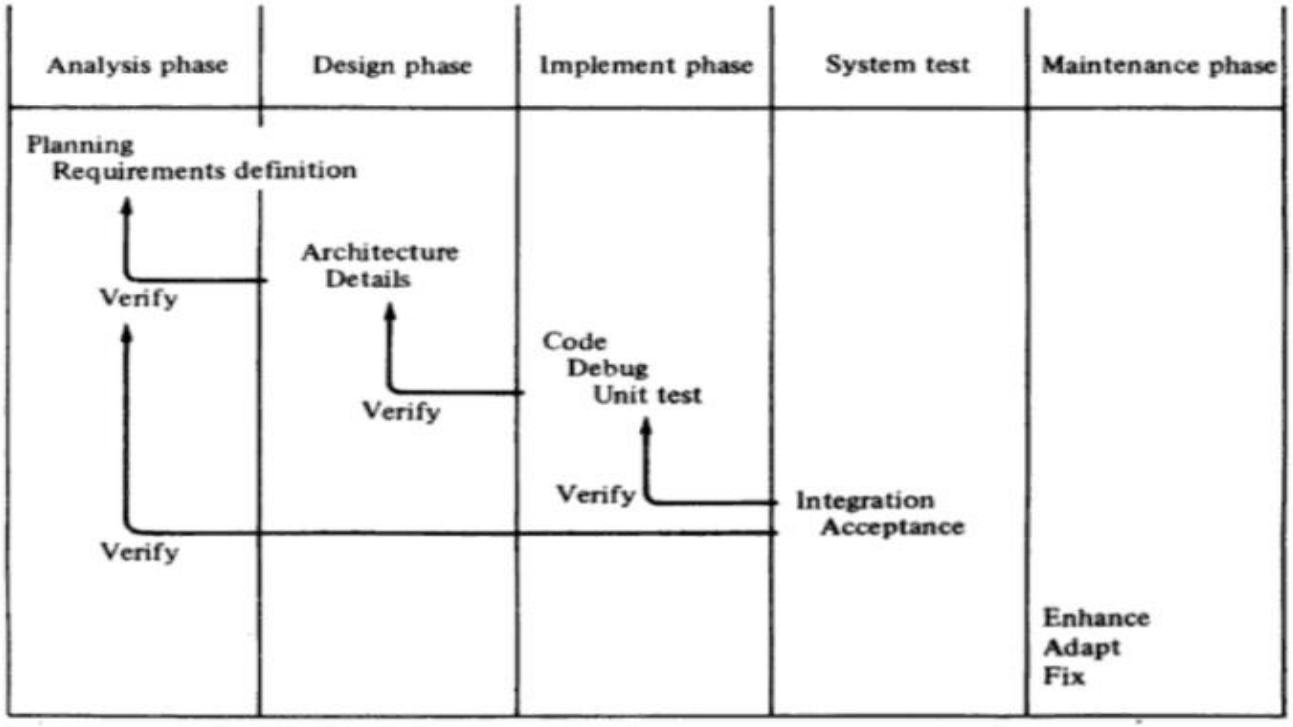
a. Phased life-cycle model

Phased model segments software ⟶ into successive activities

Phase ⟹ Well-defined processes ⟹ Well-defined products

↳ Resources are required to complete each phase

The phased model of software life cycle



The phased model consists of the above phases.

The products cascade from one level to the next in smooth progression

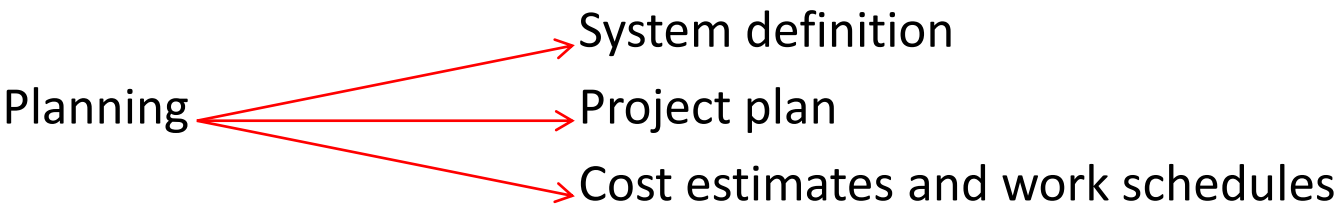
Analyses two sub phases

planning a. understanding customer problem

Requirement b. perform feasibility study

c. Develop recommended strategy

d. Determine the acceptance criteria



- ✓ In English/some other languages
- ✓ Charts, figures, graphs, tables and equations
- ✓ Life cycle model is used
- ✓ Organizational structure of the project
- ✓ Preliminary development schedule
- ✓ Preliminary cost
- ✓ Preliminary staff requirement
- ✓ Tools, techniques and standards

Requirement Definition ⟶ identifying the basic functions of hardware, software and people system.

Architectural design identifies the software components, decoupling and decomposing them into processing modules and conceptual data structures and specifying the interconnections among components.

Detailed design is adaptation of existing code, modification of standard algorithms, invention of new algorithms, data representations and packaging of the software product.

Implementation phase is translation of design specifications into source code, debugging, documentation and unit testing of source code.

Errors in implementation phase – logical errors, errors in data structure layout, failure to capture customer needs, design error that reflect that failure to translate requirement into correct design.it is expensive to remove analysis and design errors from source code.

System testing – integration testing and acceptance testing

Acceptance testing involves planning and executing various types of tests in order to demonstrate the implemented software system satifies the requirements stated in requirement document.

Software system is released for production work and enters the maintenance phase of the life cycle model.

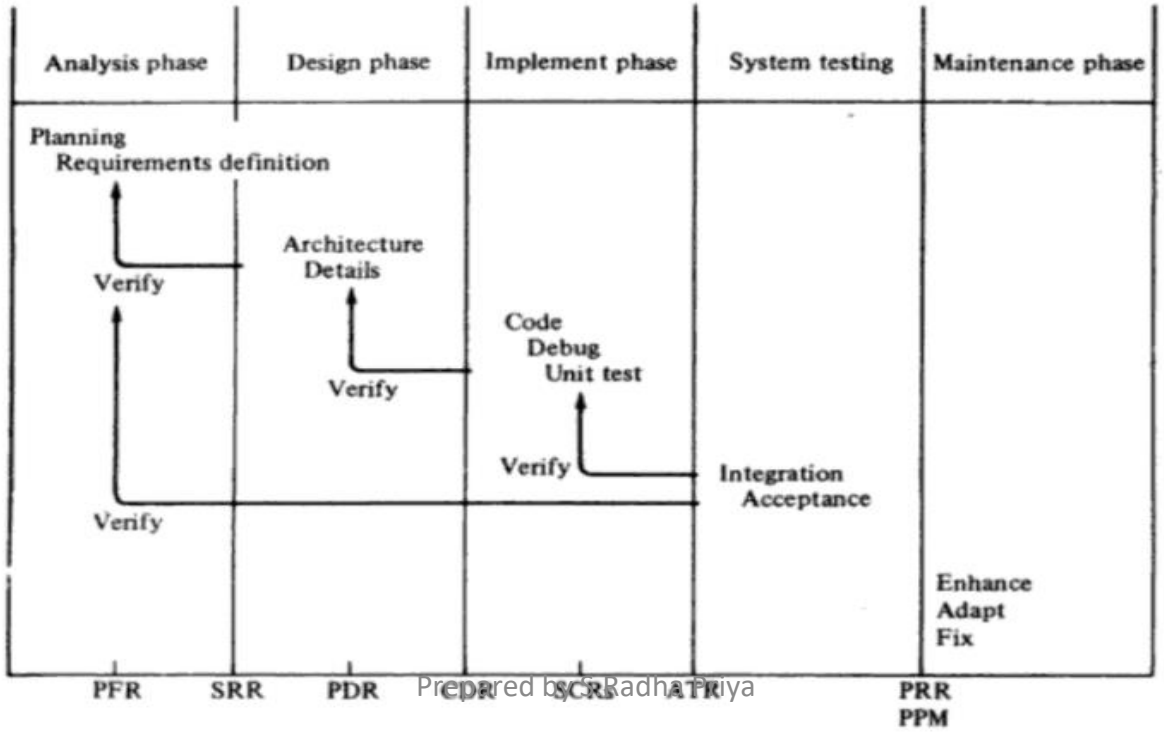
Maintenance activities

a. enhancement of capabilities

b. adaptation of software to new processing environment

c. correction of software bugs

b. Milestones, documents and reviews



	REVIEW	*WORK PRODUCTS REVIEWED*
PFR:	Product Feasibility Review	System Definition Project Plan
SRR:	Software Requirements Review	Software Requirement Specification preliminary User's Manual preliminary Verification Plan
PDR:	Preliminary Design Review	Architectural Design Document
CDR:	Critical Design Review	Detailed Design Specification User's Manual Software Verification Plan
SCR:	Source Code Review	Walkthroughs & Inspections of the Source Code
ATR:	Acceptance Test Review	Acceptance Test Plan
PRR:	Product Release Review	All of the Above.
PPM:	Project Post-Mortem	Project Legacy

The following reviews and milestones in the phased life-cycle model

1. System definition and project plan are prepared
2. User manual is prepared
3. Software requirement specification is prepared
4. Software verification plan is prepared
5. Software requirement review for all the 4 points above
6. Software design specification – architectural design, and detailed design
7. Software product specification
8. Detailed design- critical design review is held

9. Design phase- software verification plan in expanded to include methods that will be used to verify that the design is complete and consistent.

10. Actual test plan includes, actual test cases and expected results and capabilities to be demonstrated by each test.

11. During the implementation phase, source code is written, debugging unit tested.

12. The user manual, installation and training plans and software maintenance plane are completed during implementation phase.

13. Prior to product delivery, final acceptance review is performed.

14. Project legacy is written, what went wrong and what went well.

C. The Cost Model

- ✓ Cost of performing the various activities in a software project
- ✓ Cost- sum of cost incurred in conducting each phase of the project.

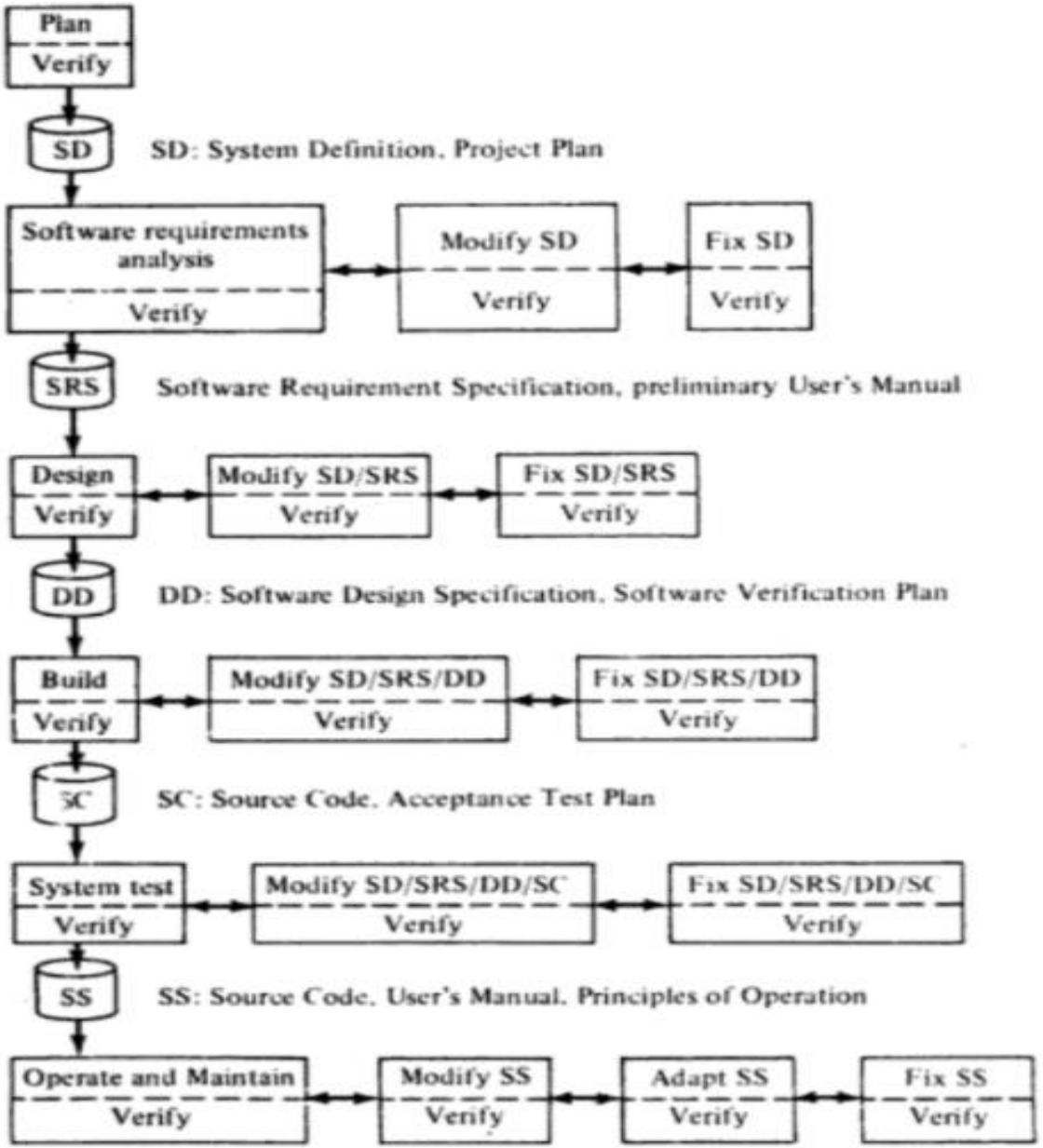


Figure 2.4 The cost model of the software life cycle *(ALF82)*.

- ✓ The system definition and project plan in the cost of performing the planning and preparing the documents, cost of verifying the customer needs.
- ✓ Software requirement specification includes the cost complete with respect to the system definition and the customer needs.
- ✓ Cost of design activities and preparing the design specification and the test plan, plus the cost of modifying the correcting the system definition, project plan and software requirement specification.
- ✓ The cost of product implementation is cost of user manual, system definition, project plan, software requirement specification, design specification, verification plan plus the cost of verifying that the implementation is correct, complete and consistent.
- ✓ Finally, the cost of software maintenance is the sum of the costs of performing product enhancement, making adaptations to new processing requirements and fixing bugs.

d. The prototype life-cycle model

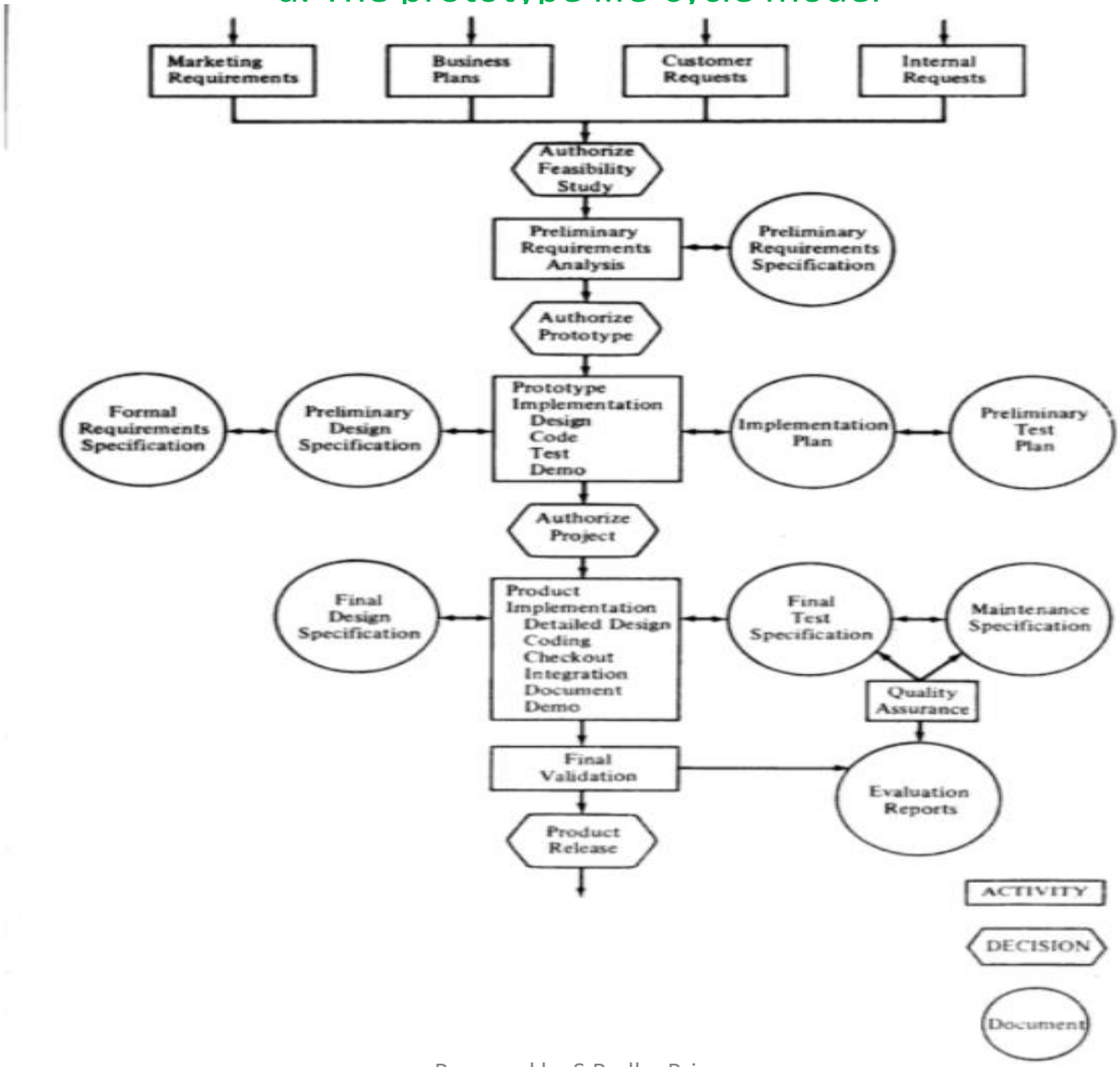


Figure 2.6 An alternate life-cycle model.

A prototype is a mock-up or model of a software product.

Reasons for developing a prototype:

- Illustrate input data formats, messages, reports and interactive dialogues for the customer.
- Gaining better understanding of the customer needs
- Explore technical issues in the proposed product.
- Major design decision will depend on response time of the device controller/efficiency of a sorting algorithm.
- In situations where analysis, design, implementation is not appropriate.
- The nature of prototyping to be performed on a particular software depend on the nature of the product.

e. Successive Versions

Product development by the method of successive versions is an extension of prototyping in which an initial product skeleton is refined into increasing levels of capability.

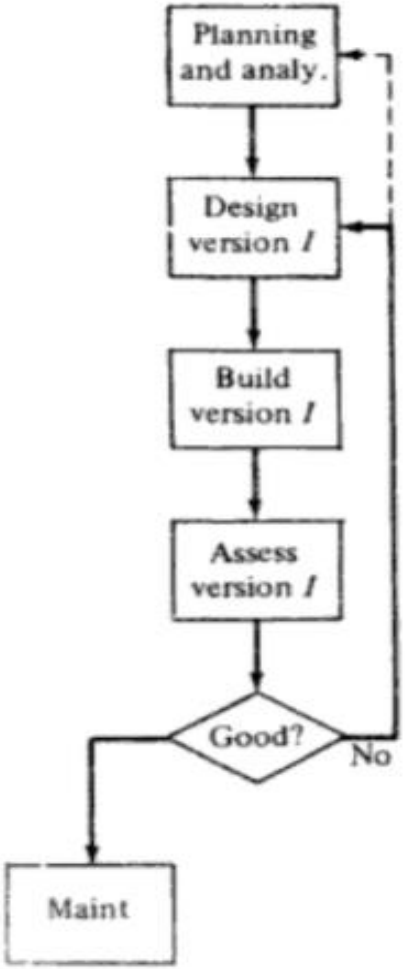


Figure 2.7a Design & implementation of successive versions.

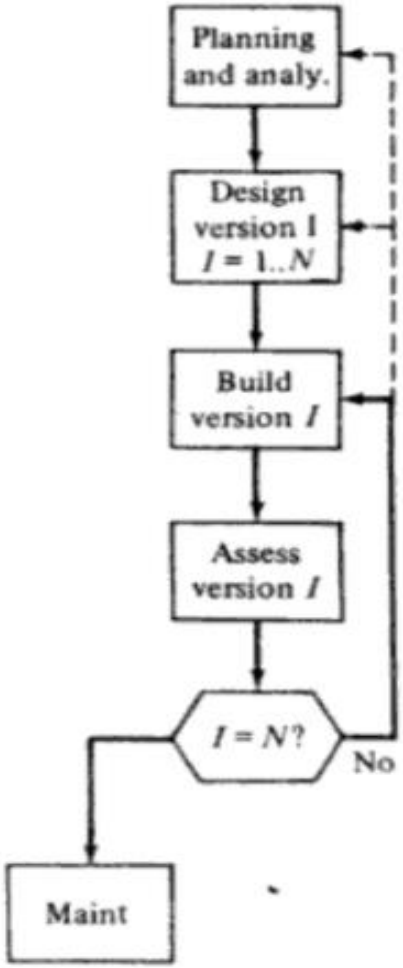


Figure 2.7b Analysis & design followed by implementation of successive versions.

Design and implementation of successive versions

- Analysis phase followed by design, implementation, building and assessment of successive versions.
- The dashed line indicates that the assessment of version I may dictate the need for further analysis before designing version I+1.

Analysis and design followed by implementation of successive versions

- Version 1 through N of the product are designed prior to any implementation activities. In this case the characteristics of each successive design will have been planned during analysis phase.
- Dashed line indicate the implementation of the I^{th}version may reveal the need for further analysis and design before proceeding with implementation of version $I + 1$.

PLANNING AN ORGANIZATIONAL STRUCTURE

1. Planning structure

a. Project format

- Involves a team of programmers who conduct a project from start to finish.
- Team members do project definition, design, implement, test, conduct

Project reviews and prepare supporting documents.

- Some members may stay with product installation and maintenance.
- Typically work for 1 to 3 years are assigned new projects on completion of the current one.

b. Functional Format

- A different team of programmers perform each phase of the project and the work products pass from team to team as they evolve planning and analysis team(system definition and plan)

Product definition team(s/w req. analysis and specification)

Implementation team(implements, debugs, unit test)

Quality Assessment team(certifies the quality)

Maintenance team

c. Matrix Format

- Each of the functions have its own management team and a group of specialist personnel who are concerned only with that function.

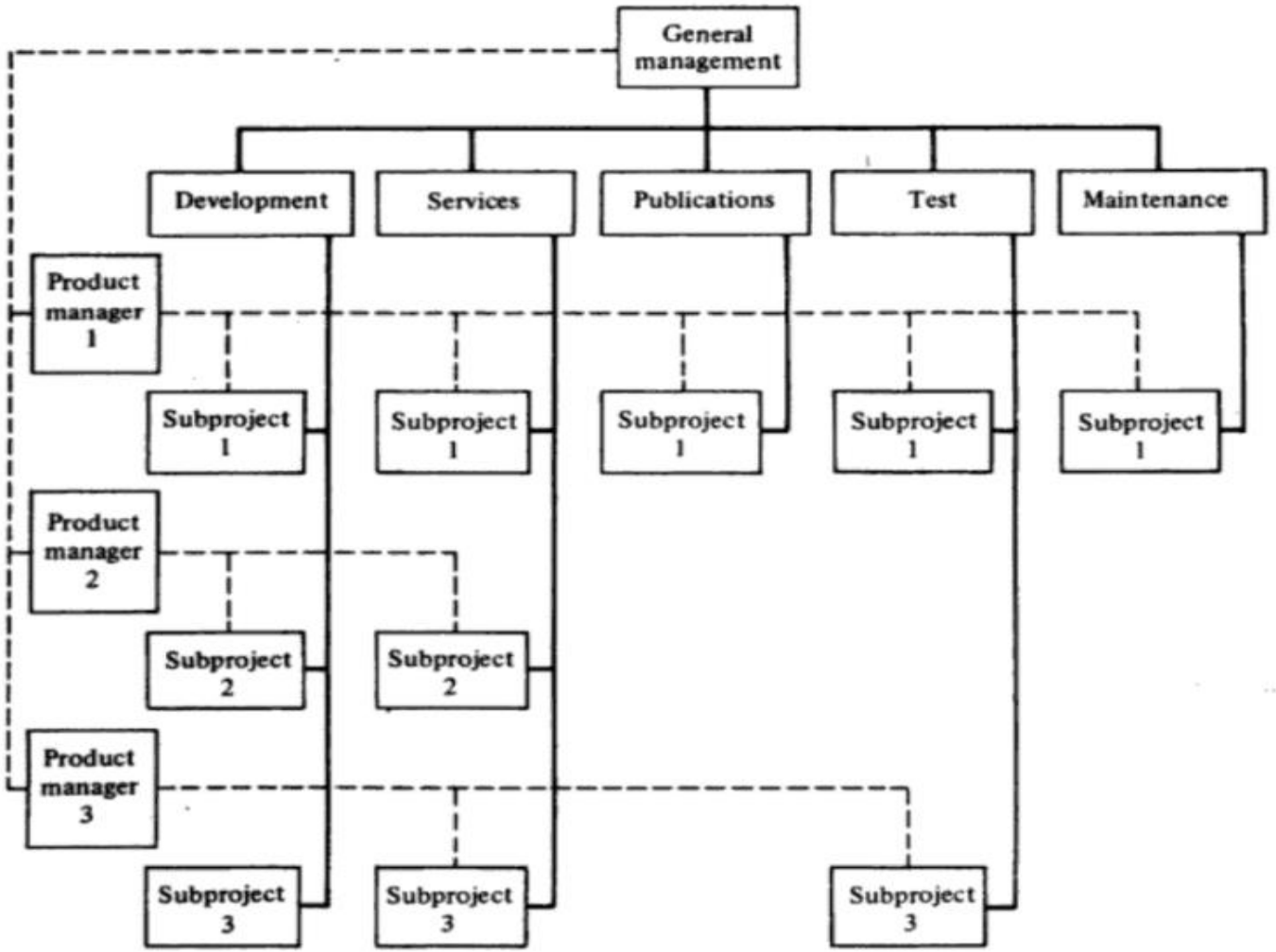


Figure 2.8 Product-function matrix organization *(adapted from (GUN78)).*

- Product function matrix organization, each functional group participates in each project. For ex: software development team member belong originally to the development function but work under supervision of a particular project manager.

  disadvantage: more than one boss

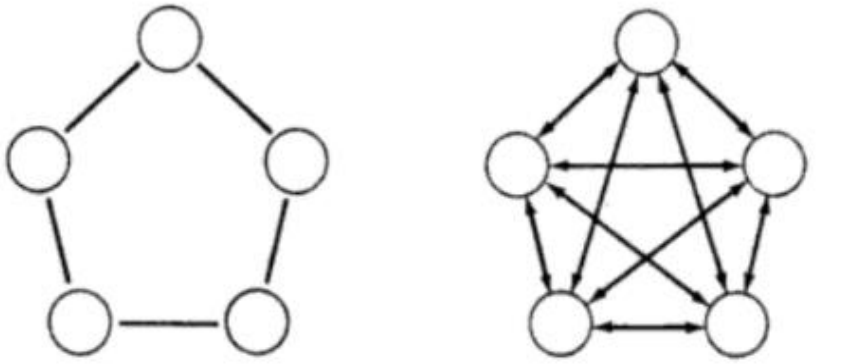
- Matrix organizations are increasingly popular because special expertise can be concentrated in particular functions.

2. Programming team structure

- Every programming team have an internal structure. The best team structure for any particular project depends on the nature of the project and the product.

a. Democratic teams(egoless team)

- In egoless team goals are set and decisions made by group consensus. Group leadership rotates from member to member based on the tasks to be performed and the differing abilities of the team members.
- Democratic team differs from an egoless team



(a) Structure (b) Communication paths

Figure 2.9 Egoless programming team structure and communication paths *(MAN81)*.

Advantages

- ✓ Opportunity to each team member to contribute to decisions
- ✓ Learn from one another
- ✓ Increased job satisfaction

Disadvantages

- ✓ All team members must work well together
- ✓ Weakening individual responsibility and authority

b. Chief programmer teams

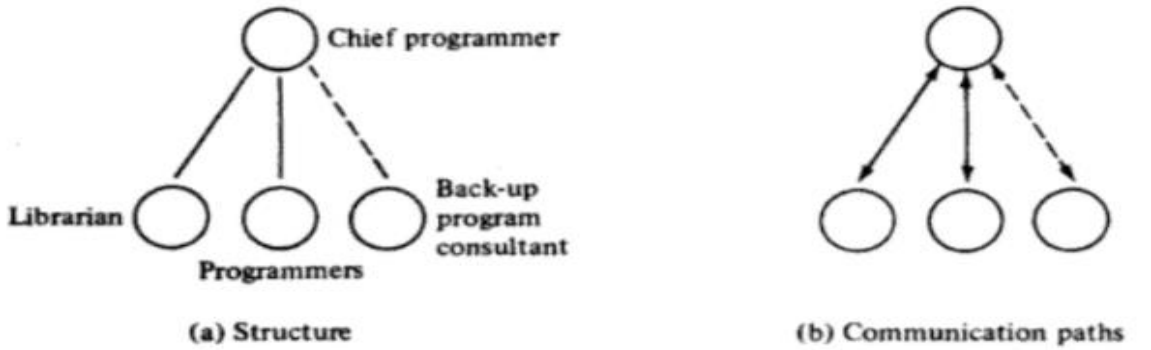


Figure 2.10 Chief programmer team structure and communication paths *(MAN81)*.

- ✓ Highly structured
- ✓ The work is allocated by the chief programmer
- ✓ The programmers between two and five, write code, debug, document and unit test
- ✓ Librarian maintain program listings, design documents and test plans
- ✓ Back-up programmer serves as a consultant to the chief programmer on various technical problems

⇓

Liason with customer publications group quality assurance group perform some analysis, design.

c. Hierarchical team structure

In a hierarchical team, the project leader assigns tasks, attends reviews and walkthroughs, detects problem areas, balances the workload and participates in technical activities.

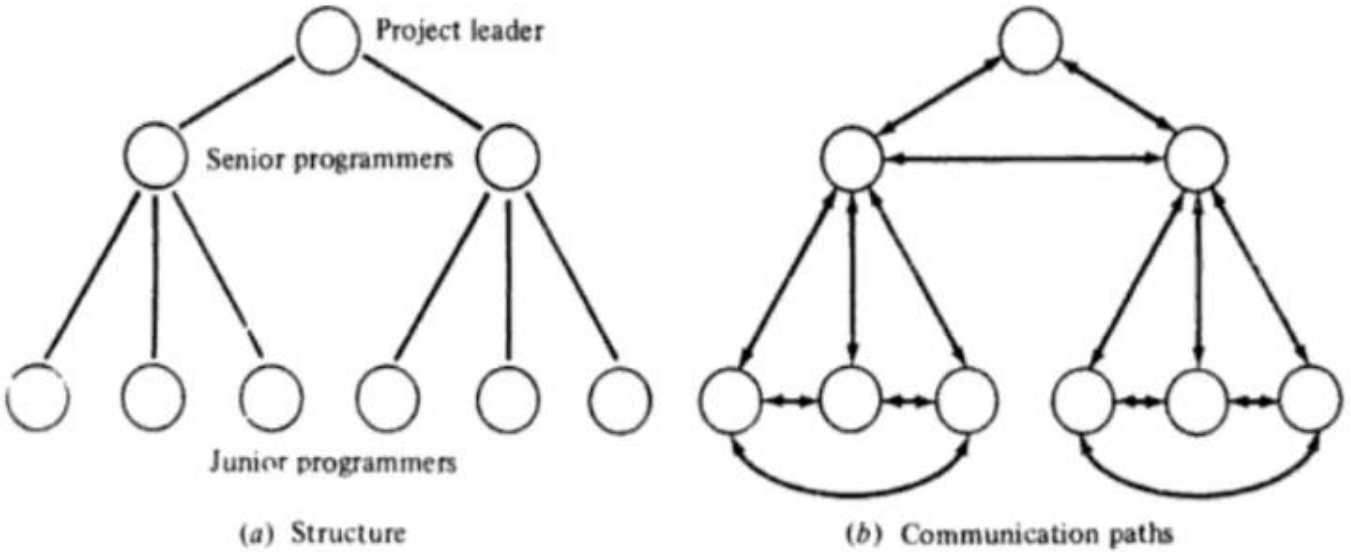


Figure 2.11 Hierarchical programming team structure and communication paths *(MAN81)*.

- ✓ It limits the number of communication paths
- ✓ It is reasonable to have 3 teams in hierarchical fashion, with each team leader reporting to the project leader.
- ✓ The number of immediate subordinates at each level should be limited to five to seven people.

d. Management by objectives(MBO)

- ❖ Using MBO, employees set their own goals and objectives with the help of their supervisor, participate in the setting of their supervisors goals.
- ❖ Objectives are set for periods of 1 to 3 months
- ❖ MBO is well suited to the milestones and intermediate work products of software engineering.

Other planning activities

1. Planning for configuration management and quality assurance
2. Planning for independent verification and validation.
3. Planning phase-independent tools and techniques
4. Other planning activities cost estimates, schedule, preliminary staffing level.