MENOUFIA UNIVERSITY
جامعة المنوفية
منارة المعرفة فى قلب الدلتا

Software Engineering

Dr. Mohammed Badawy

mbmbadawy@yahoo.com
mohamed.badawi@el-eng.menofia.edu.eg

13 May 2016

Outline

- References
- Course Objectives
- Course Contents
- Software Engineering: Introduction
- Course Projects

SOFTWARE ENGINEERING & TESTING

B. B. Agarwal, S. P. Tayal, M. Gupta

COMPUTER SCIENCE SERIES

WHAT EVERY ENGINEER SHOULD KNOW ABOUT

SOFTWARE ENGINEERING

Phillip A. Laplante

Pankaj Jalote

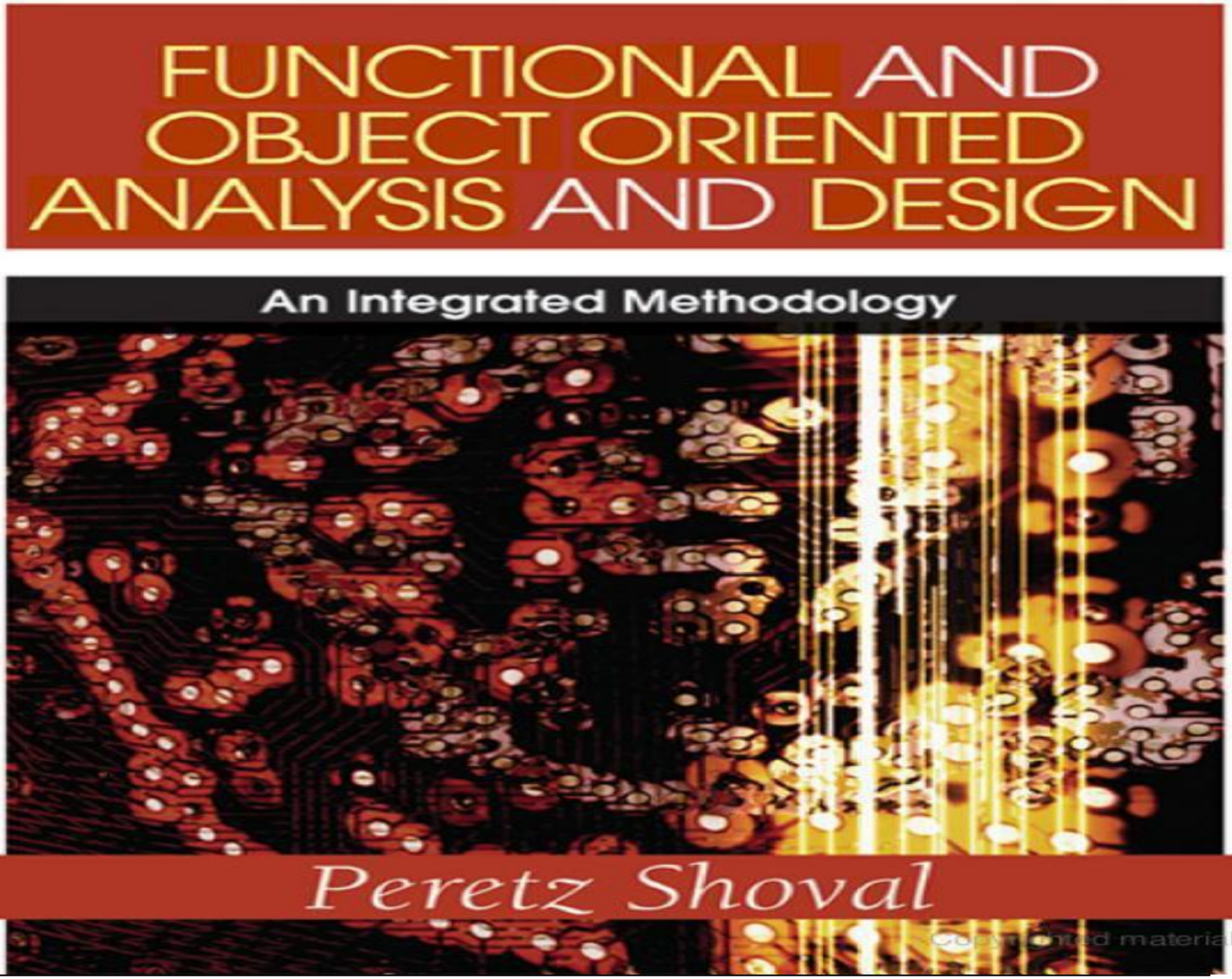
UNDERGRADUATE TOPICS IN COMPUTER SCIENCE

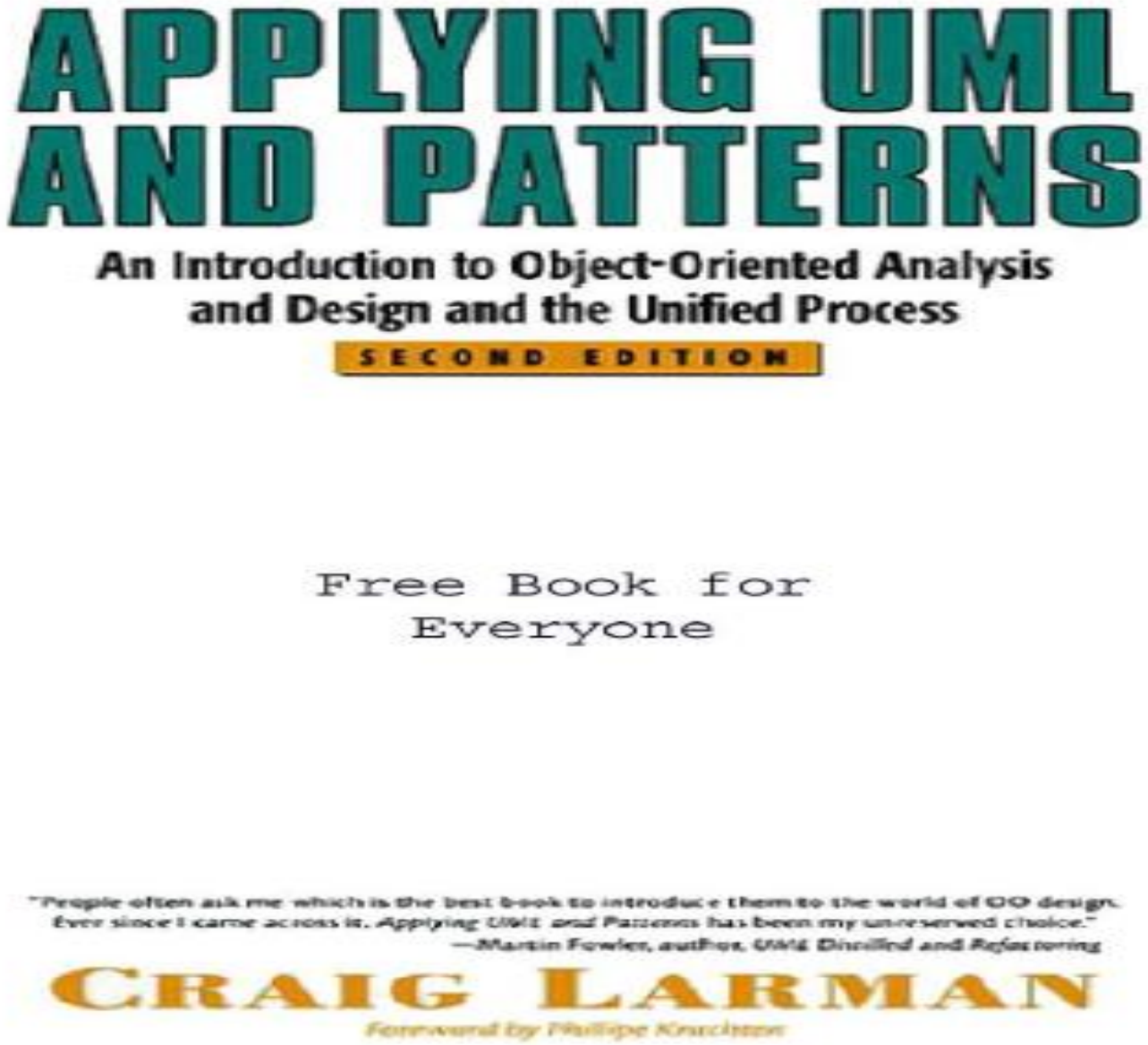
A Concise Introduction to Software Engineering

Springer

UTiCS


（英文版 · 第8版）
SOMMERVILLE
Software Engineering
8
Ian Sommerville


FUNCTIONAL AND OBJECT ORIENTED ANALYSIS AND DESIGN
An Integrated Methodology
Peretz Shoval
Copyrighted material


APPLYING UML
AND PATTERNS
An Introduction to Object-Oriented Analysis
and Design and the Unified Process
SECOND EDITION
Free Book for
Everyone
"People often ask me which is the best book to introduce them to the world of OO design.
Ever since I came across it, Applying UML and Patterns has been my unreserved choice."
—Martin Fowler, author, UML Distilled and Refactoring
CRAIG LARMAN
Foreword by Philippe Kruchten

NEW AGE

Software Engineering

(A Lifecycle Approach)

Pratap K.J. Mohapatra

NEW AGE INTERNATIONAL PUBLISHERS

An Integrated Approach to Software Engineering

Third Edition

Pankaj Jalote
Indian Institute of Technology Kanpur

Course Objectives

Upon completion of the course, students will be able to :

- Understand and use basic concepts of software engineering applicable to software systems
- Plan software projects, gather requirements, create a software architecture and design, implement code, perform systematic testing,
- Select and apply appropriate tools, and have experience of working as a member of a team on a software engineering project

Course Contents

- Software Development Processes
- Requirements Engineering
- Software Architecture
- Software Design & Coding
- Software Maintenance & Testing
- Tools [MS Project – Power Designer]

توزيع درجات المقرر

- ✓ إجمالي الدرجات ١٥٠ درجة توزيعها كالاتي:
- ✓ الامتحان النهائي ٩٠ درجة
- ✓ (الحضور بالمعمل والمحاضرة -انشطة- اختبار منتصف الفصل) ٣٠ درجة
- ✓ - (امتحان عملي وشفهي - مشاريع ***Projects*** ٣٠ درجة على الاقل

Introduction

- Virtually all countries now depend on complex computer-based systems
- National infrastructures and utilities rely on computer-based systems and most electrical products include a computer and controlling software
- Industrial manufacturing and distribution is completely computerised, as is the financial system
- **Therefore, producing and maintaining software cost-effectively is essential for the functioning of national and international economies**

What Is Software ?

- The term software refers to the set of computer programs, procedures, and associated documents (flowcharts, manuals, etc.) that describe the programs and how they are to be used

IEEE Definition

- Software is the collection of computer programs, procedure rules and associated documentation and data

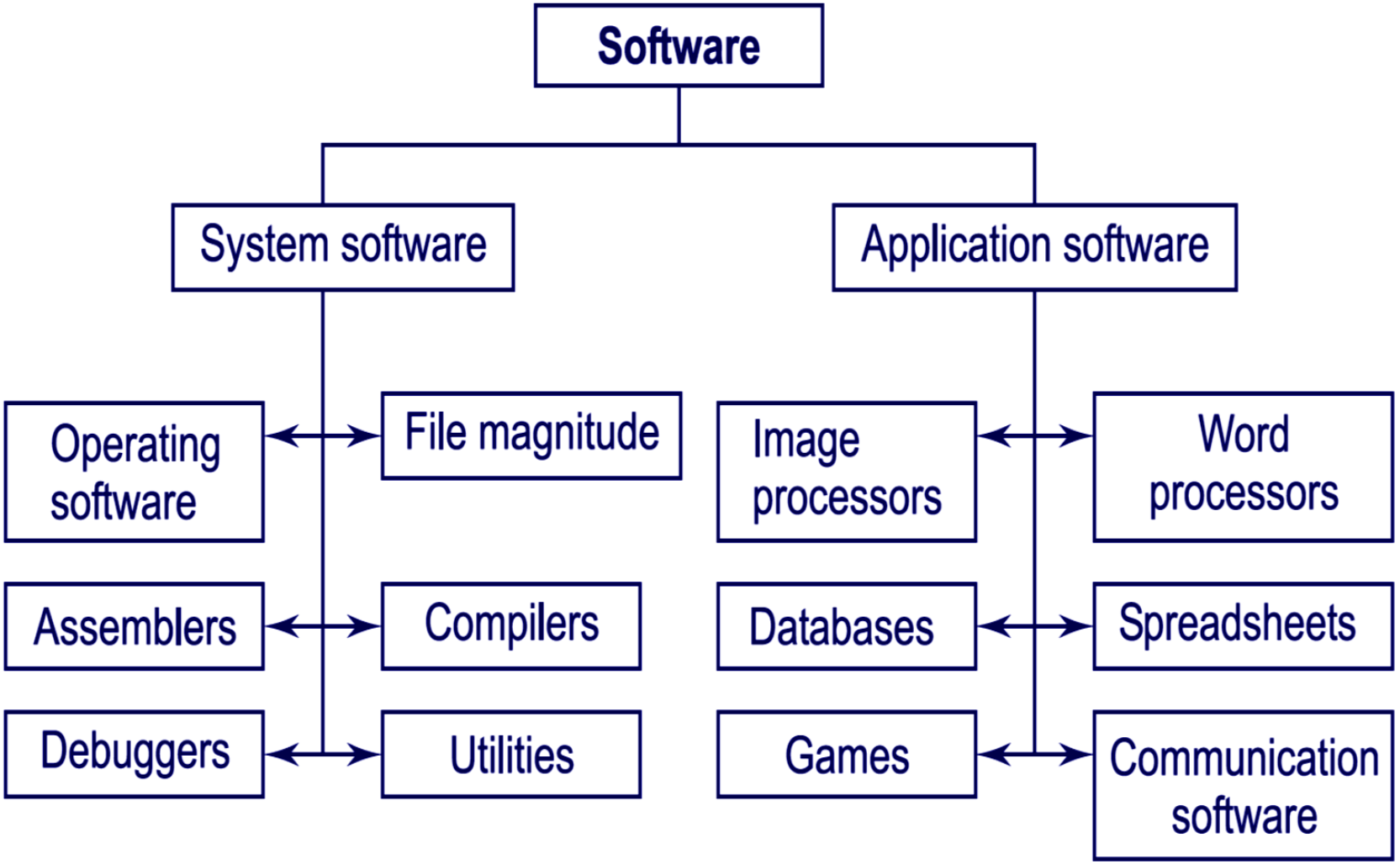
Importance of Software

- It is the engine that drives business decision-making
- It serves as the basis for modern scientific investigation and engineering problem-solving
- It is embedded in all kinds of systems, such as transportation, medical, telecommunications, military, industrial processes, entertainment, office products, etc.

Categories of Software

- **System Software**: as the Operating Systems, Compilers, Assemblers, Debuggers, and Utilities
- **Application Software**: Programs that do real work for users, as word processors, spreadsheets, and database management systems

Categories of Software



Classes of Software

- ***Generic Software***: Designed for a broad customer market whose requirements are very common, fairly stable, and well-understood by the software engineer
- **Customized Software**: Developed for a customer where domain, environment, and requirements are unique to that customer and cannot be satisfied by generic products
  - Examples: process control systems, traffic management systems, hospital management systems

WHAT IS SOFTWARE ENGINEERING ?

- Software engineering is an engineering discipline which is concerned with all aspects of software production

IEEE Definition

- Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, i.e., the application of engineering to software

Software Engineering Principles

- Software-engineering principles deal with both the process of software engineering and the final product.
- The right process will help produce the right product, but the desired product will also affect the choice of which process to use.
- Both are important.

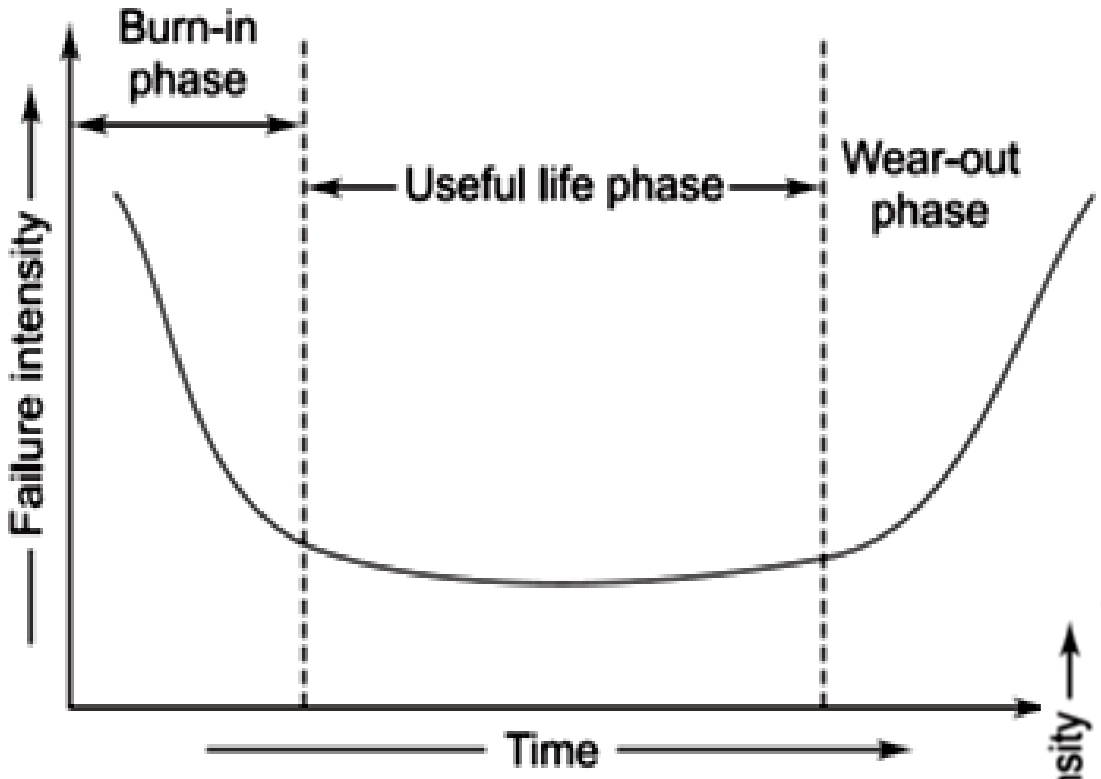
Software Engineering Principles

- In principle, we should distinguish between **methods** and **techniques.**
- **Methods** are general guidelines that govern the execution of some activity; they are rigorous, systematic, and disciplined approaches.
- **Techniques** are more technical and mechanical than methods; often, they also have more restricted applicability.
- In general, however, the difference between the two is not sharp. We will therefore use the two terms interchangeably.

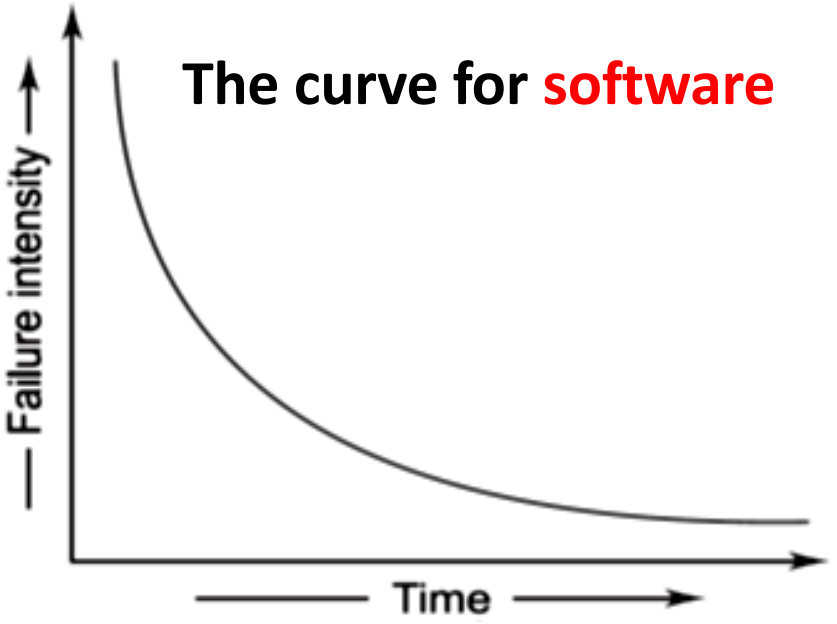
Software Characteristics

- Most software is custom-built, rather than assembled from existing components
- Software is flexible
- Software doesn't wear out

Software Characteristics



Failure rate as a function of time for hardware



The curve for software

Software Crisis

- 31% of the projects get cancelled before they are completed
- For every 100 projects, there are 94 restarts
- During software development, then, many problems are raised and that set of problems is known as the software crisis.

Software Crisis: Problems

- Schedule and cost estimates are often grossly inaccurate
- The “productivity” of software people hasn’t kept pace with the demand for their services
- The quality of software is sometimes less than adequate
- Communication between the customer and software developer is often poor
- Software maintenance tasks devour the majority of all software funds

Software Crisis: Causes

- If there is delay in any process or stage (i.e., analysis, design, coding & testing) then scheduling does not match with actual timing
- Communication between managers and customers, software developers, support staff, etc., can break down because the special characteristics of software and the problems associated with its development are misunderstood

Software Myths

- If we get behind schedule, we can add more programmers and catch up.
- Project requirement continuously changes, but changes can be easily accommodated because software is flexible.
- The only deliverable work product for a successful project is the working program.
- Once we write the program and get it to work, our job is done.

Software Myths

- Until we get the program running, we have no way of assessing its quality.
- Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down
- A general statement of objectives is sufficient to begin writing programs; we can fill in the details later.

Software Applications

- System Software
- Real-time Software
- Embedded Software
- Business Software
- Personal Computer Software
- Artificial-Intelligence Software
- Web-based Software
- Engineering and Scientific Software

Software Engineering Processes

- A process is a series of steps involving activities, constraints, and resources that produce an intended output of some kind

Processes Characteristics

- The process prescribes all of the major process activities
- The process uses resources, subject to a set of constraints (such as a schedule), and produces intermediate and final products
- The process may be composed of sub-processes that are linked in some way
- The process may be defined as a hierarchy of processes, organized so that each sub-process has its own process model

Processes Characteristics

- Each process activity has entry and exit criteria, so that we know when the activity begins and ends
- The activities are organized in a sequence, so that it is clear when one activity is performed relative to the other activities
- Every process has a set of guiding principles that explain the goals of each activity
- Constraints or controls may apply to an activity, resource, or product. For example, the budget or schedule may constrain the length of time an activity may take or a tool may limit the way in which a resource may be used

Software Processes

- A software process is the related set of activities and processes that are involved in developing and evolving a software system

OR

- A set of activities whose goal is the development or evolution of software

Software Processes' Activities

- **Software Specifications**: Definition of the functionality of the software and constraints on its operation
- **Software Development:** Software that meets the specifications must be produced
- **Software Validation:** The software must be validated to ensure that it does what the customer wants
- **Software Evolution:** The software must evolve to meet changing customer needs

Software Processes' Activities

- Some processes are more suitable than others for some types of applications
- If an inappropriate process is used, this will probably reduce the quality or the usefulness of the software product to be developed
- A software process is a set of activities
- When those activities are performed in specific sequence in accordance with ordering constraints, the desired results are produced

Evolution of Software

- **Early Era:** *1950 – 1960*

– General-purpose hardware became commonplace - Software was custom-designed

- **Second Era:** *1961 – 1972*

- Multi-users – Databases - Real-time software - Product software - Multi-programming

Evolution of Software

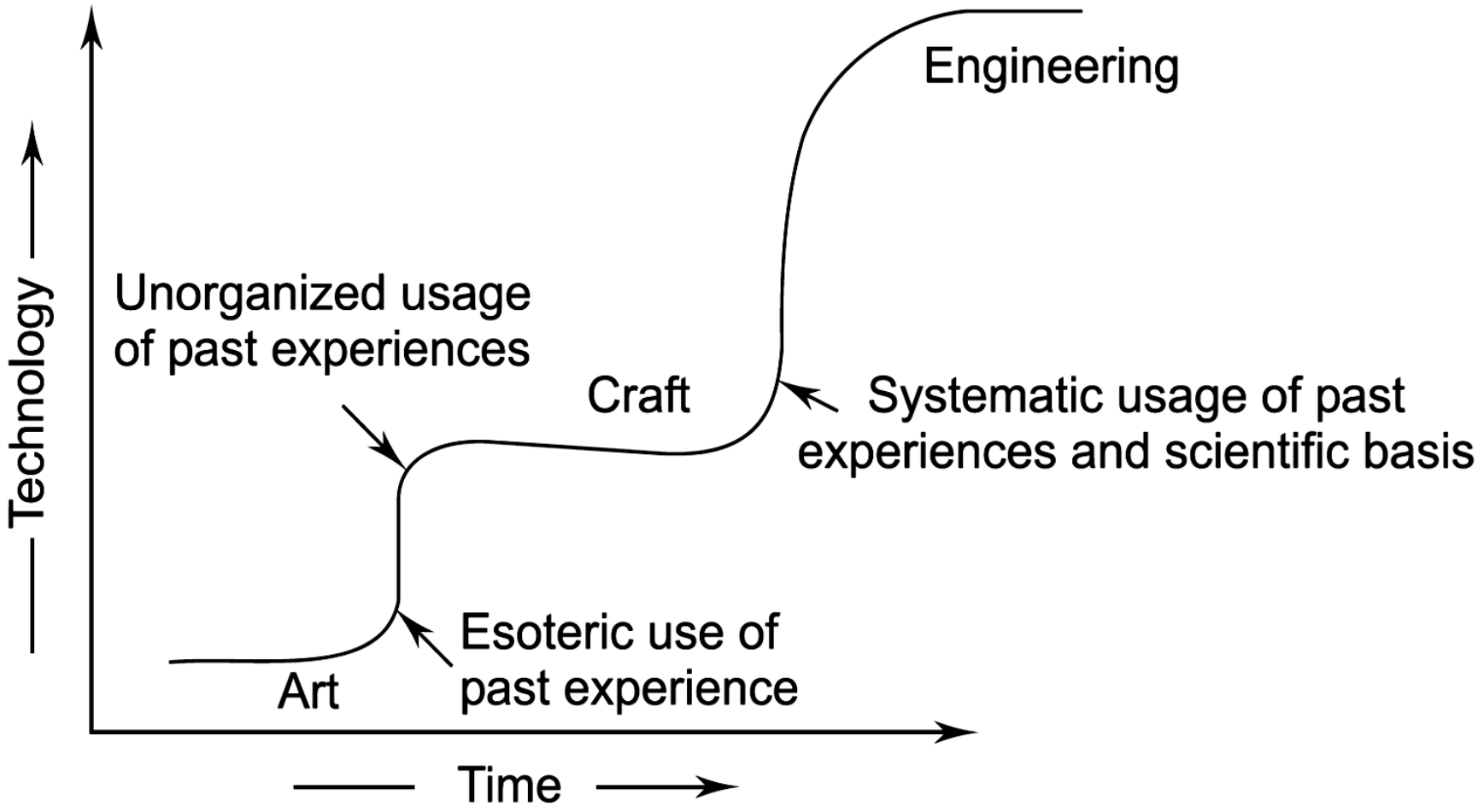
- **Third Era:** *1973 – 1985*

- Embedded intelligence - Consumer impact - Distributed systems - Low-cost hardware

- **Fourth Era:** *1986 – Present*
- Moves us away from individual computers and computer programs and toward the collective impact of computers and software
- Includes: Powerful desktop systems - Expert systems - Artificial intelligence - Network computers - Parallel computing - Object-oriented technology

Evolution of Software

Software-engineering principles have evolved over the past more than 50 years from art to an engineering



Some Terminologies

- Deliverables and Milestones
- Product and Process
- Measures, Metrics, and Indicators

Some Terminologies

Deliverables and Milestones

- Different deliverables are generated during software development. The examples are source code, user manuals, operating procedure manuals, etc.
- The milestones are the events that are used to ascertain the status of the project. Finalization of specifications is a milestone. Completion of design documentation is another milestone.

Some Terminologies

Product and Process

- What is delivered to the customer is called the product. It may include source-code specification documents, manuals, documentation, etc.
- A process is the way in which we produce software. It is the collection of activities that leads to (a part of) a product.

Some Terminologies

Measures, Metrics, and Indicators

- **Measures** provide a quantitative indication of amount, dimension, capacity, or size of a given attribute of a product *(comparing to a standard, e.g. five centimeters)*.
- **Metrics** are a quantitative measure of the degree to which a system, component, or process possesses a given attribute (*e.g. only two user-discovered errors in the first 18 months of operation*)
- **An indicator** is a device or variable that can be set to a prescribed state based on the results of a process or the occurrence of a specified condition (*e.g. a flag)*
- An indicator is something that draws a person's attention to a particular situation

Programs versus Software Products

- ***Programs:*** A program is a subset of software.
- ***Software Products:*** A software product consists not only of the program code but also of all the associated documents, such as the requirements specification documents, the design documents, the test documents, and the operating procedures which include user manuals and operational manuals.

Cost, Schedule, and Quality

- The software should be produced at reasonable cost, in a reasonable time, and should be of good quality.
- Industrial-strength software is very expensive
- The productivity in the software industry for writing fresh code generally ranges from few hundred to about 1000+ LOC per person-month
- Software companies often charge the client for whom they are developing the software between $3000 - $15,000 per person-month

Cost, Schedule, and Quality

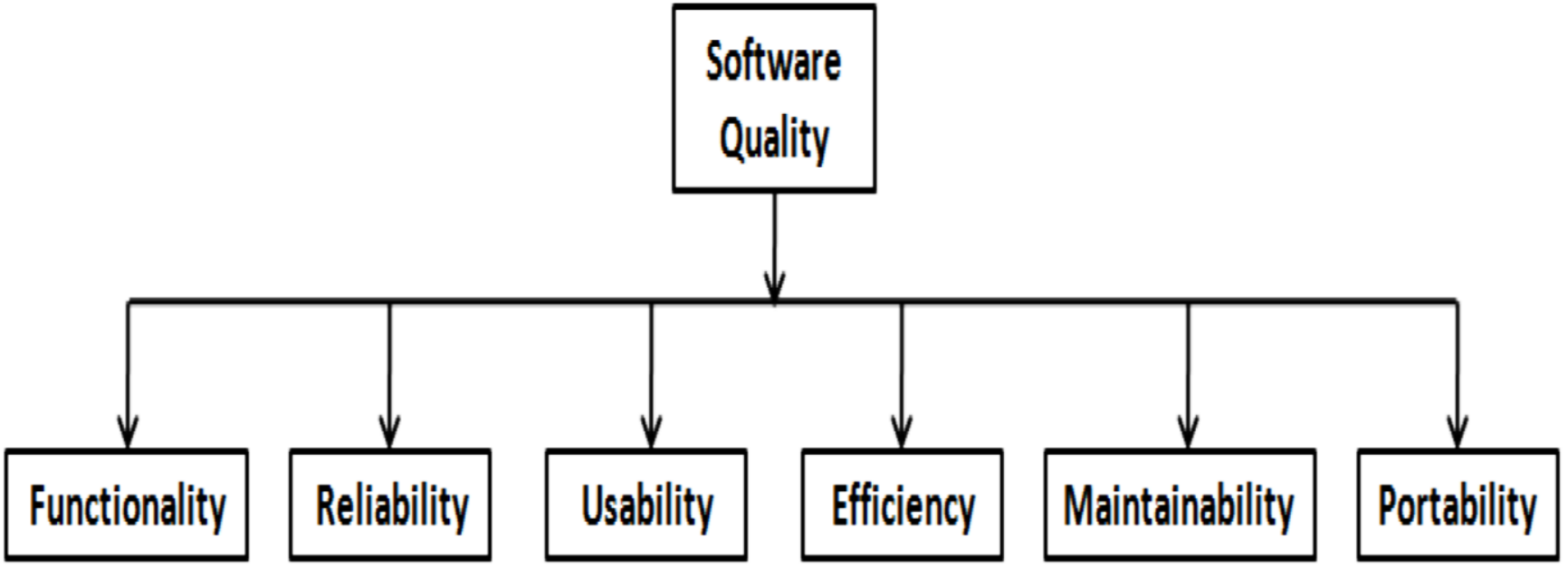
- With a productivity of 1000 LOC per person-month, it means that each line of delivered code costs between $3 and $15!
- And even small projects can easily end up with software of 50,000 LOC.
- With this productivity, such a software project will cost between $150,000 and $750,000!

Cost, Schedule, and Quality

- Schedule is another important factor in many projects
- Business trends are dictating that the time to market of a product should be reduced; that is, the cycle time from concept to delivery should be small

Cost, Schedule, and Quality

- Developing high-quality software is another fundamental goal of software engineering
- The international standard on software product quality suggests that software quality comprises six main attributes



Cost, Schedule, and Quality

- **Functionality**: The capability to provide functions which meet stated and implied needs when the software is used
- **Reliability**: The capability to provide failure-free service
- **Usability**: The capability to be understood, learned, and used
- **Efficiency**: The capability to provide appropriate performance relative to the amount of resources used
- **Maintainability**: The capability to be modified for purposes of making corrections, improvements, or adaptation
- **Portability**: The capability to be adapted for different specified environments without applying actions other than those provided for this purpose in the product

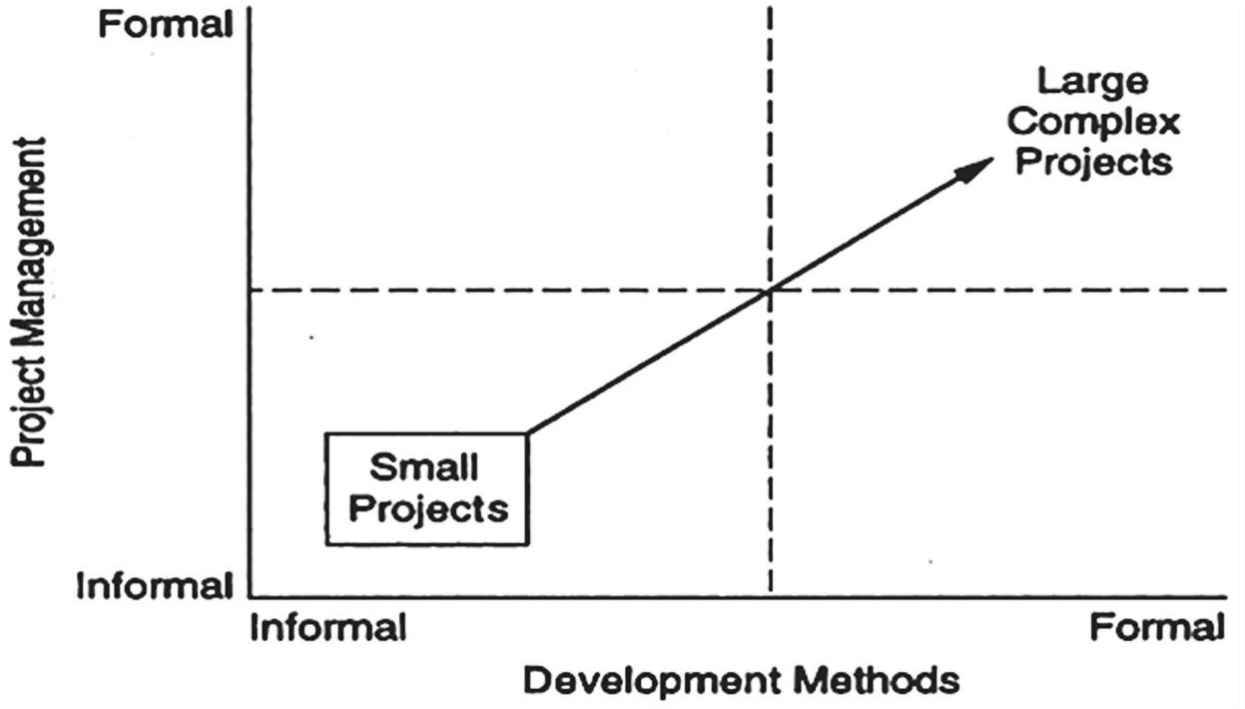
Scale and Change

- Methods that one can use to develop programs of a few hundred lines cannot be expected to work when software of a few hundred thousand lines needs to be developed.
- A different set of methods must be used for developing large software.

Size (KLOC)	Software	Languages
980	gcc	ansic, cpp, yacc
320	perl	perl, ansic, sh
200	openssl	ansic, cpp, perl
100	apache	ansic, sh
65	sendmail	ansic
30,000	Red Hat Linux	ansic, cpp
40,000	Windows XP	ansic, cpp

Scale and Change

- In small projects, informal methods for development and management can be used. However, for large projects, both have to be much more rigorous, as illustrated in the next figure



Scale and Change

- Change is another characteristic of the problem domain which the approaches for development must handle.
- As the complete set of requirements for the system is generally not known or stated, as development proceeds and time passes, additional requirements are identified, which need to be incorporated in the software being developed.
- This need for changes requires that methods for development embrace change and accommodate it efficiently.
- Change requests can be quite disruptive to a project, and if not handled properly, can consume up to 30 to 40% of the development cost.

Software Engineering versus Computer Science

- Essentially, computer science is concerned with the theories and methods that underlie computers and software systems
- Whereas software engineering is concerned with the practical problems of producing software.
- Some knowledge of computer science is essential for software engineers in the same way that some knowledge of physics is essential for electrical engineers.

Software Engineering versus System Engineering

- System engineering is concerned with all aspects of the development and evolution of complex systems where software plays a major role.
- System engineering is therefore concerned with hardware development, policy and process design and system deployment as well as software engineering.
- System engineers are involved in specifying the system, defining its overall architecture and then integrating the different parts to create the finished system.
- System engineering is an older discipline than software engineering

How do software engineers spend their time on the job?

- Software engineers probably spend less than 10% of their time writing code
- The other 90% of their time is involved with other activities that are more important than writing code. These activities include:
  - Eliciting requirements
  - Analyzing requirements
  - Writing software requirements documents
  - Building and analyzing prototypes
  - Developing software designs
  - Writing software design documents

How do software engineers spend their time on the job?

- Researching software engineering techniques or obtaining information about the application domain
- Developing test strategies and test cases
- Testing the software and recording the results
- Isolating problems and solving them
- Learning to use or installing and configuring new software and hardware tools
- Writing documentation such as users manuals
- Attending meetings with colleagues, customers, and supervisors
- Archiving software or readying it for distribution

Computer-Aided Software Engineering (CASE)

- CASE is the name given to software used to support software process activities such as requirements engineering, design, program development and testing.
- CASE tools therefore include design editors, compilers, debuggers, system building tools and so on.

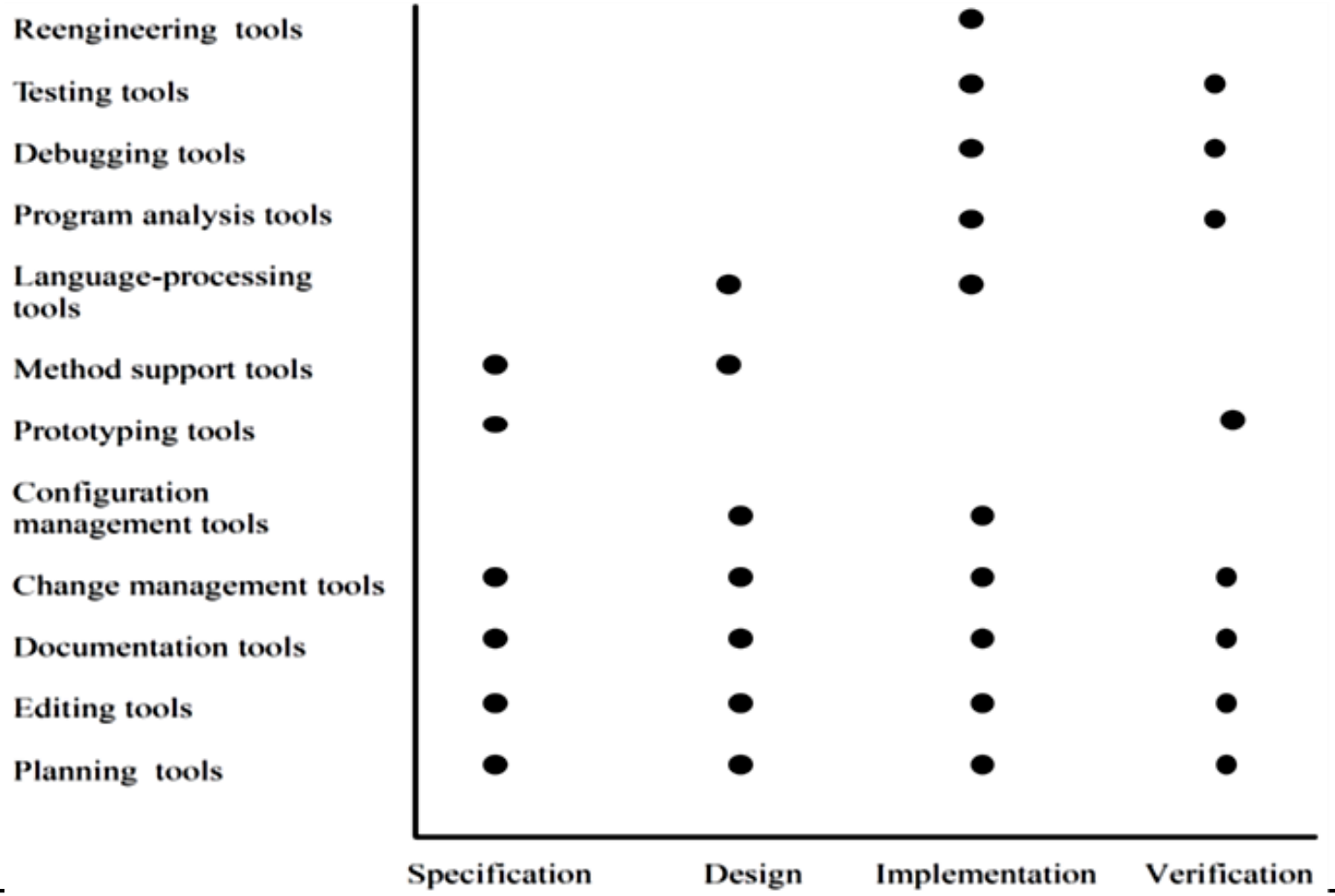
Computer-Aided Software Engineering (CASE)

- Classification of CASE tools according to function

Tool type	Examples
Planning tools	PERT tools, estimation tools, spreadsheets
Editing tools	Text editors, diagram editors, word processors
Change management tools	Requirements traceability tools, change control systems
Configuration management tools	Version management systems, system building tools
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

Computer-Aided Software Engineering (CASE)

- Activity based classification of CASE tools

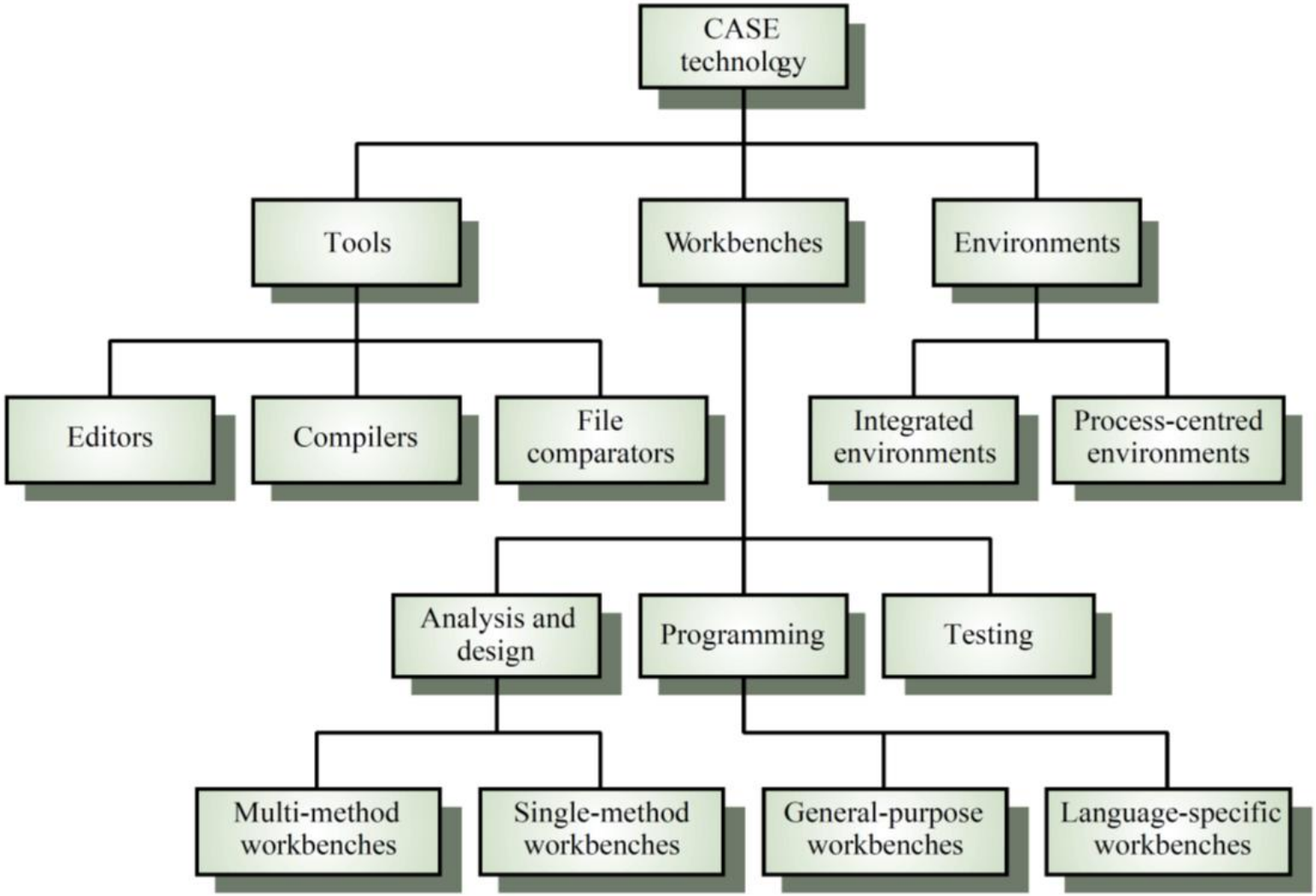


Computer-Aided Software Engineering (CASE)

CASE systems may be classified in three categories:

- ***Tools*** support individual process tasks such as checking the consistency of a design, compiling a program and comparing test results.
- Tools may be general-purpose standalone tools or grouped into workbenches.
- ***Workbenches*** support process phases or activities such as specification, design, etc.
- They normally consist of a set of tools with some greater or lesser degree of integration.
- ***Environments*** support all or at least a substantial part of the software process.
- They normally include several integrated workbenches.

Computer-Aided Software Engineering (CASE)



Phased Development Process

- A development process consists of various phases, each phase ending with a defined output
- The phases are performed in an order specified by the process model being followed:

Phased Development Process

- **Requirements Analysis**: The output is an SRS document
- **Software Design**: The output is an SDS document
- **Coding**: The output is the code
- **Testing**: unit testing - integration testing - acceptance testing: the outputs are test reports and error reports