Computer Fundamentals

Pradeep K. Sinha
Priti Sinha

Chapter 10

Computer Software

Learning Objectives

In this chapter you will learn about:

- Term "Software" and its relationship with "Hardware"
- Various types of software and their examples
- Relationship among hardware, system software, application software, and users of a computer system
- Different ways of acquiring software
- Various steps involved in software development
- Software Engineering and CASE tools
- Firmware and Middleware

Software, Firmware and Middleware

Software

- ***Hardware*** refers to the physical devices of a computer system.
- ***Software*** refers to a collection of programs, procedures, and associated documents describing the programs, and how they are to be used.
- ***Program*** is a sequence of instructions written in a language that can be understood by a computer
- ***Software package*** is a group of programs that solve a specific problem or perform a specific type of job (for example, word-processing package)

Relationship Between Hardware and Software

- Both hardware and software are necessary for a computer to do useful job. They are complementary to each other
- Same hardware can be loaded with different software to make a computer system perform different types of jobs
- Except for *upgrades*, hardware is normally a one-time expense, whereas software is a continuing expense
- Upgrades refer to renewing or changing components like increasing the main memory, or hard disk capacities, or adding speakers, modems, etc.

Types of Software

Most software can be divided into two major categories:

- ***System software*** are designed to control the operation and extend the processing capability of a computer system
- ***Application software*** are designed to solve a specific problem or to do a specific task

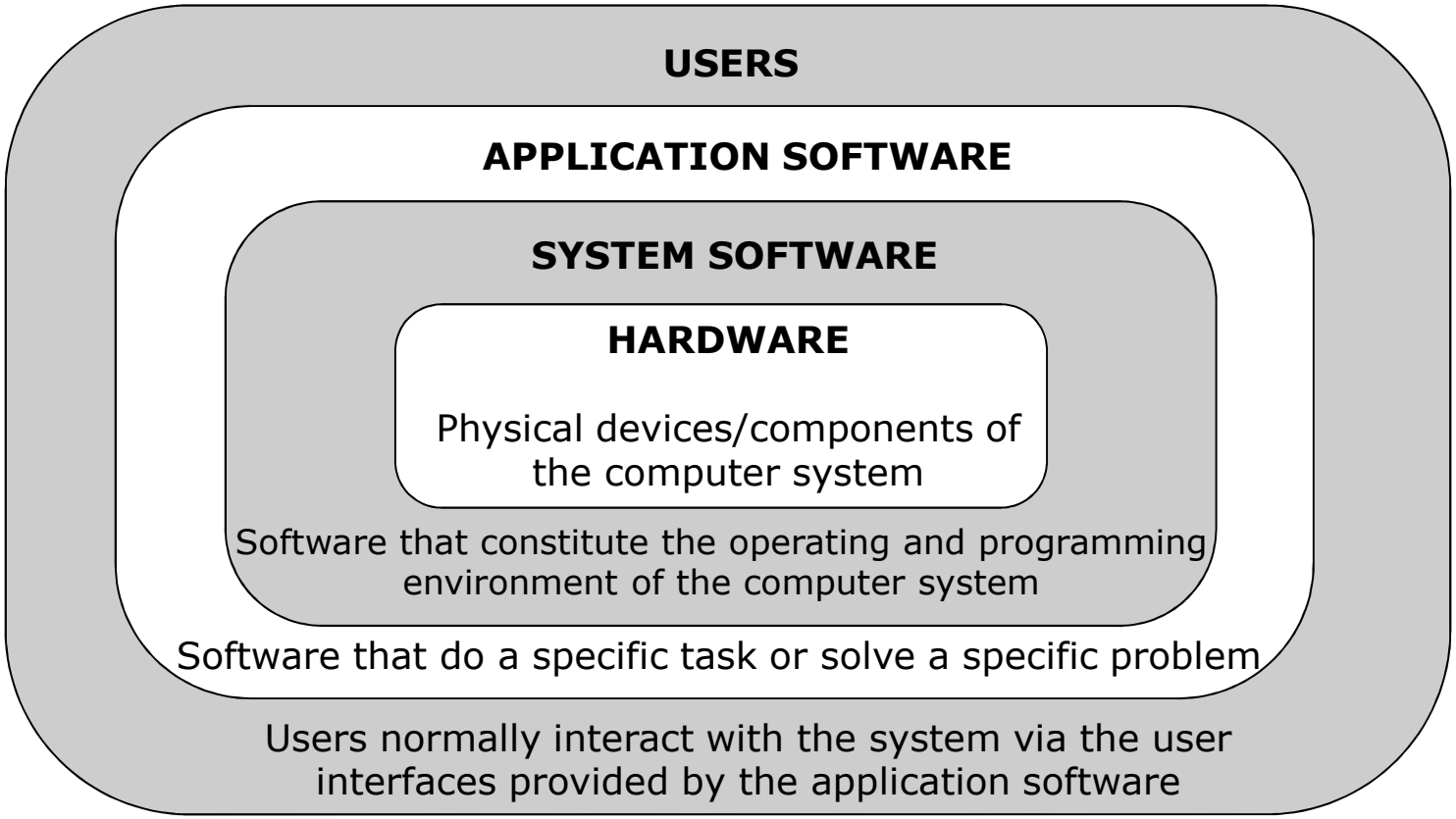
System Software

- Make the operation of a computer system more effective and efficient
- Help hardware components work together and provide support for the development and execution of application software
- Programs included in a system software package are called ***system programs*** and programmers who prepare them are called ***system programmers***
- Examples of system software are operating systems, programming language translators, utility programs, and communications software

Application Software

- Solve a specific problem or do a specific task
- Programs included in an application software package are called ***application programs*** and the programmers who prepare them are called ***application programmers***
- Examples of application software are word processing, inventory management, preparation of tax returns, banking, etc.

Logical System Architecture



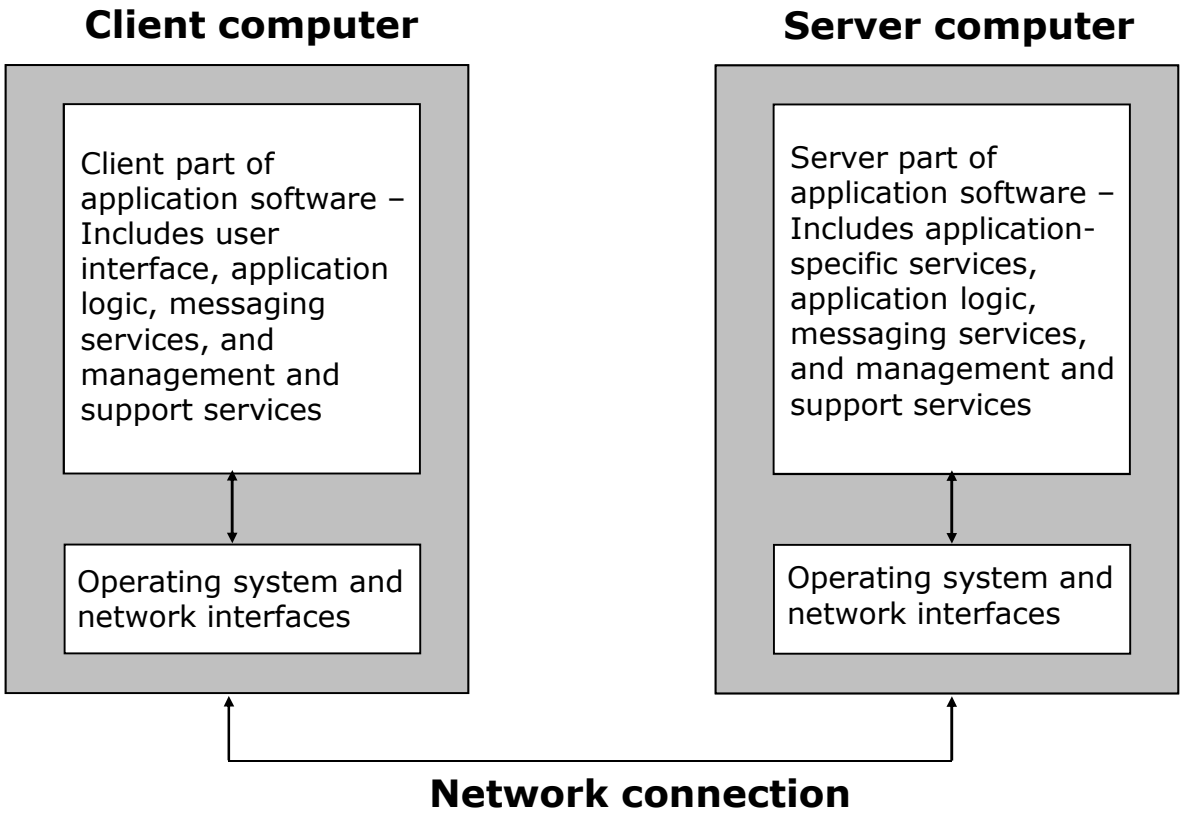
Firmware

- *Firmware* refers to a sequence of instructions (software) substituted for hardware
- This software is stored in a read-only memory (ROM) chip of the computer
- Initially, vendors supplied only system software in the form of firmware
- Many vendors now supply even application programs as firmware
- Firmware is frequently a cost-effective alternative to wired electronic circuits
- Firmware has today made it possible to produce smart machines of all types

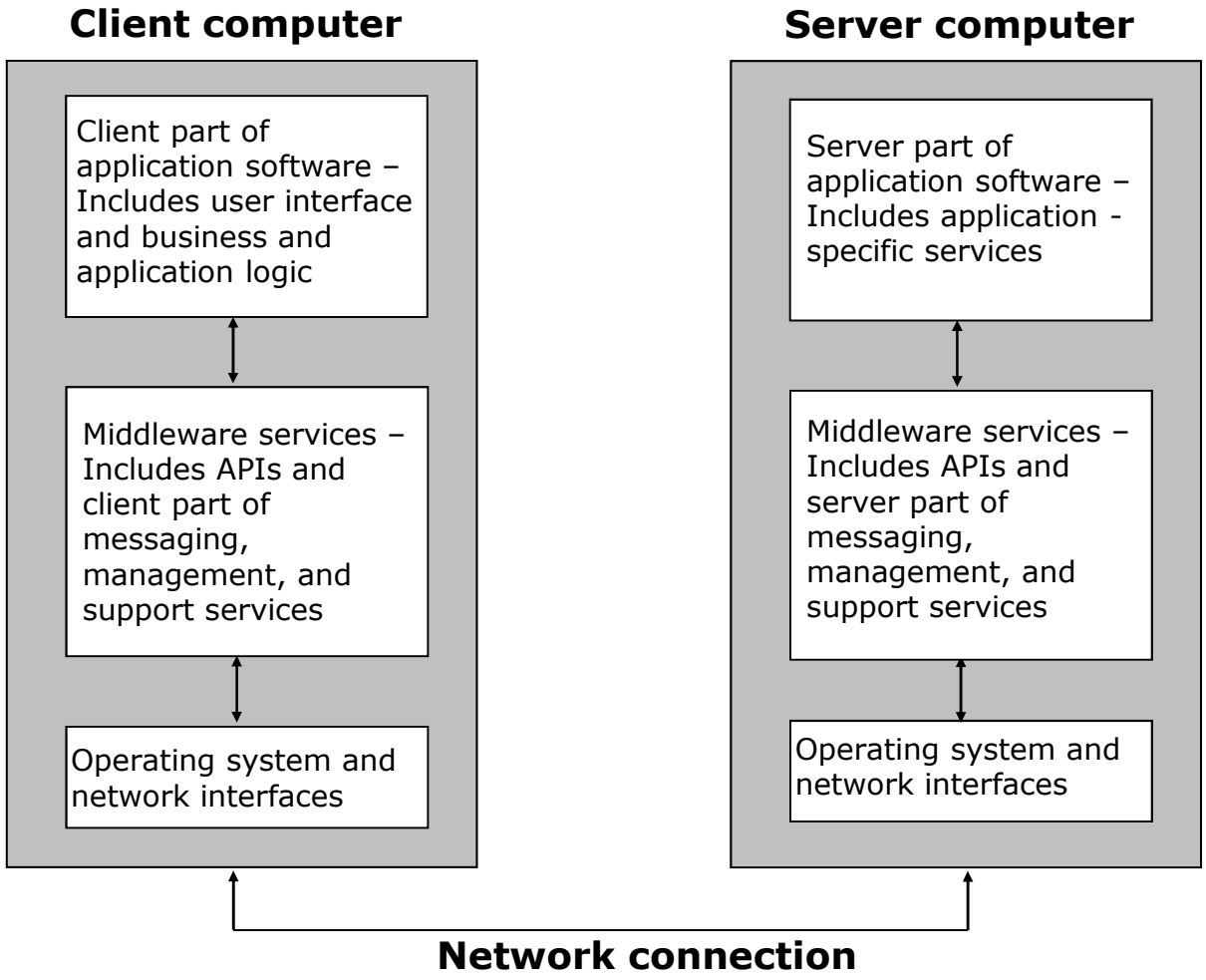
Middleware

- Basic idea here is to have a separate software layer that acts as "glue" between the client and server parts of an application
- Provides a programming abstraction as well as masks the heterogeneity of underlying networks, hardware, and operating systems
- This software layer is known as middleware because it sits in the middle, between the operating system and applications
- *Middleware* is defined as a set of tools and data that helps applications use networked resources and services

Two-tier, Client-server Architecture



Three-tier, Client-server Architecture



Dr. Sinha's

Acquiring Software

Ways of Acquiring Software

- Buying pre-written software
- Ordering customized software
- Developing customized software
- Downloading public-domain software

Each of these ways of acquiring software has its own advantages and limitations

Buying Pre-written Software

- Prepare a list of all available software packages that can perform the desired task
- Select those software packages only that meet the system specifications
- Choose the best one
- Find out the source from where you can purchase the finally selected software at the cheapest price

Advantages and Limitations of Buying Pre-written Software

- Usually costs less
- Planned activity can be stared almost immediately
- Often, operating efficiency and the capability to meet specific needs of user more effectively in not as good for pre-written software packages as for in-house developed software packages

Ordering Customized Software

- If none of the available pre-written software packages meet the specific requirements of a user it becomes necessary for the user to create a customized software package
- If a team does not exist in-house, the user must get it created by another organization by placing an order for it
- Following steps are followed for this:
  - User prepares a list of all user requirements carefully
  - User then floats a tender for inviting quotations for creation of the requisite software
  - After receiving the quotations, the user selects a few of them for further interaction based on the cost quoted by them, their reputation in the market, their submitted proposal, etc.
  - User then personally interacts with the representative(s) of each of the selected vendors

Ordering Customized Software

- User makes a final choice of the vendor to offer the contract for creation of the requisite software
- Selected vendor then creates the software package and delivers it to the user
- Vendor need not develop everything from scratch
- User may choose to place the order for both hardware and software to a single vendor
- Vendor develops the software on the chosen hardware, and delivers the software along with the hardware to the user
- This is referred to as an *end-to-end solution* or a *turnkey solution*

Advantages & Limitations of Ordering Customized Software

- User need not maintain its own software development team, which is an expensive affair
- User needs to always depend on the vendor for carrying out the changes and the vendor may separately charge for every request for change

Developing Customized Software

- If no pre-written software package meets the specific requirements, and if the organization has in-house software development team, it may choose to develop a customized software package in-house
- Following steps are followed for in-house development of a software package:
  - Organization first constitutes a project team to develop the software
  - Team studies the requirements and plans functional modules
  - It then analyzes which of the functional modules need to be developed, and which of the functional modules' requirements are met with existing pre-written software
  - Team next plans their programs and does coding, testing, debugging, and documentation

(Continued on next slide...)

Developing Customized Software

- Team then tests all the modules in an integrated manner
- Team then deploys the software for use by users
- Users then use the software and the project team members responsible for maintenance activities maintain it

Advantages & Limitations of Developing Customized Software

- Easier to carry out changes in the software, if it is developed in-house
- Developing software in-house means a major commitment of time, money, and resources
- In-house software development team needs to be maintained and managed

Downloading Public-domain Software

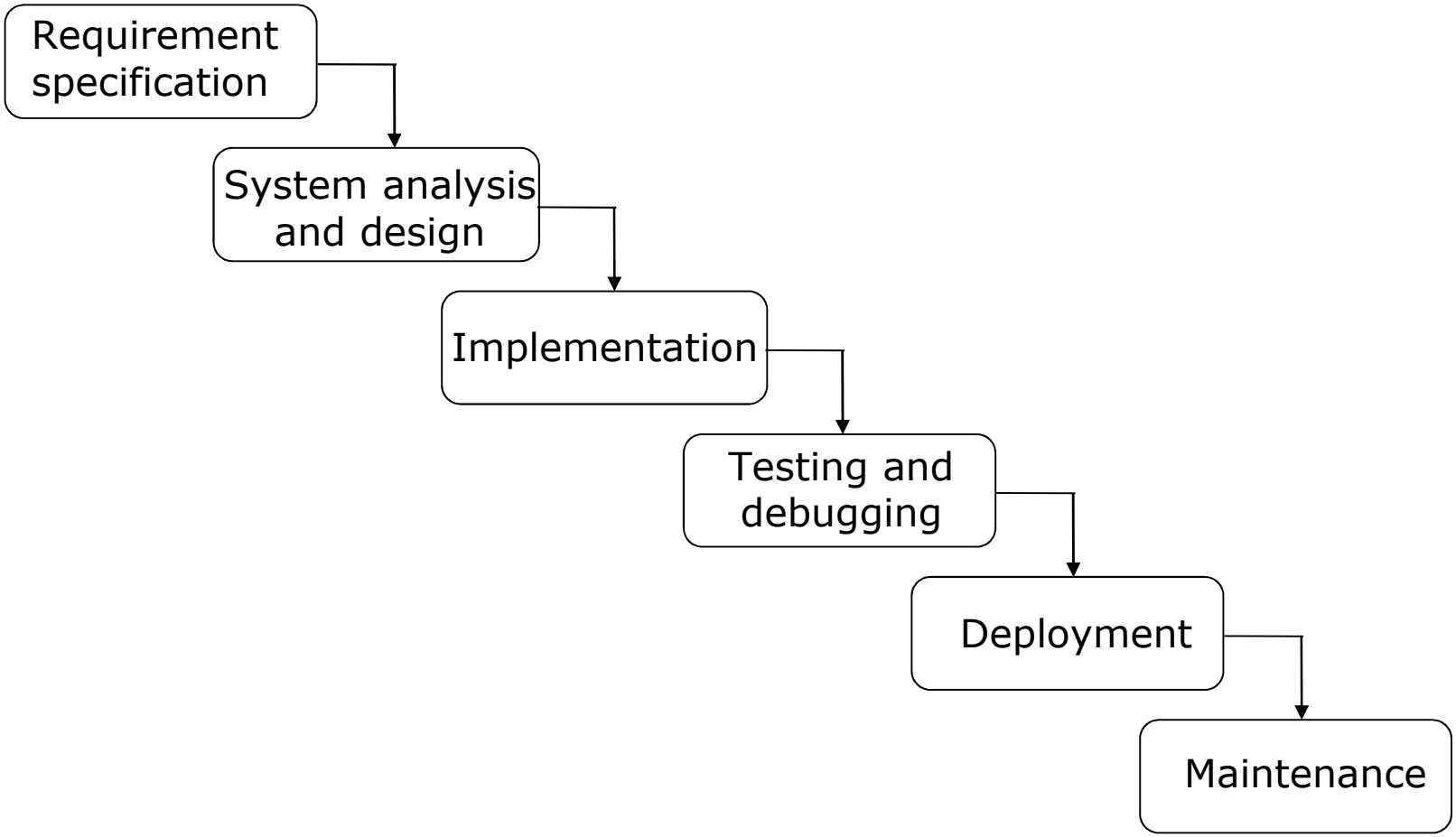
- *Public-domain software* is software available free or for a nominal charge from the bulletin boards or user-group libraries on the Internet
- Public-domain software is also referred to as *shareware/ freeware*
- They are also known as *community-supported software* as mostly the authors do not support the product directly and users of the software support and help each other
- *Open Source Software* (*OSS*): Usually, OSS allows a user to download, view, modify, and distribute modified source code to others

Advantage & Limitations of Downloading Public-domain Software

- Available for free or as shareware, and are usually accompanied with source code
- Usually community-supported as author does not support users directly
- Can be downloaded and used immediately
- They may not be properly tested before release
- Open Source Software (OSS) are becoming popular due to:
  - Allows any user to download, view, modify, and redistribute
  - User can fix bugs or change software to suit needs
  - Copyright is protected for both original and subsequent authors
- Not all open source software are free and vise-verse

Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC)



(Continued on next slide...)

Software Development Life Cycle (SDLC)

- **Requirement specification**
  - Team defines all possible requirements of the software in this phase
- **System analysis and design**
  - Team studies the requirements specified in the first phase with respect to available hardware and software technologies and prepares a system design document
- **Implementation**
  - This phase is also known as *construction* or *code generation* because in this phase, the team constructs the various software components specified in system design document

Software Development Life Cycle (SDLC)

- **Testing and debugging**
  - Team integrates the independently developed and tested modules into a complete system
  - Team then tests the integrated system to check if all modules coordinate properly with each other
- **Deployment**
  - Team deploys the software at user(s) site on (or along with) associated hardware for use by intended user
- **Maintenance**
  - Team fixes problems in the system in this phase

Dr. Sinha's

Software Engineering

What is Software Engineering?

- *Software* is a set of computer programs, procedures, and associated documents
- *Engineering* is systematic application of scientific knowledge in creation and building of cost-effective solutions
- *Software engineering* is systematic application of principles of computer science and mathematics in creation and building of cost-effective software solutions

Need for Software Engineering

- With software products growing in scale and complexity, number of software developers involved in a software development project has been increasing proportionately
- Managing the development of large software products and maintaining them is a difficult task
- Progressively larger software products in sensitive applications are being used
- Required correctness and reliability of software products is increasing
- Quality and productivity demands for software products led to the introduction of systematic practices (later on known as *software engineering practices*)

Goals of Software Engineering

- **Correctness should be of very high degree**
  - Correctness refers to the degree to which a software product performs its intended functions properly, consistently, and predictably
- **Usability should be of very high degree**
  - Usability refers to the ease with which a software product and its associated documentation are usable
- **Should be cost-effective**
  - Cost-effectiveness means that the total development and operational costs of a software product should be as low as possible

Principles of Software Engineering

- **Precise requirements definition**
  - Designer of a software product must define its requirements precisely
- **Modular structure**
  - Designer of a software product must structure it in a modular fashion
  - Modular design helps in distribution of development task of different modules to different programmers
- **Abstraction**
  - Software product should use abstraction and information hiding
  - Object-oriented programming takes care of this aspect of software engineering
  - Abstraction helps in easy reusability of existing modules

Principles of Software Engineering

- **Uniformity**
  - Software product should maintain uniformity in design, coding, documentation, etc. Uniformity ensures consistency
- ***CASE (Computer Aided Software Engineering)*** **tools** provide a wide range of features for creation of better and more reliable software products
  - *Design specification tools*
    - Allow programmers to design visually screens, menus, database tables, reports, dialog boxes, and other key components of a program
  - *Code generation tools*
    - Generate source code (programs) from the design specification of a software product

(Continued on next slide…)

Principles of Software Engineering

- *Testing and debugging tools*
  - Help programmers in testing and debugging of their programs
- *Source-code analysis tools*
  - Help in optimizing a program by pointing out unreachable lines of code and functions that the program never uses (calls)
- *Documentation tools*
  - Assist in automatic generation of technical documents for a software product

Key Words/Phrases

- Application programmers
- Application programs
- Application software
- CASE tools
- Computer program
- Customized software
- Database
- Education software
- End-to-end solution
- Entertainment software
- Firmware
- Graphics software
- Hardware
- Middleware
- Open Source Software
- Personal assistance software
- Pre-written software
- Public-domain software
- Shareware
- Software
- Software Development Life Cycle (SDLC)
- Software Engineering
- Software package
- Spreadsheet
- System programmers
- System programs
- System software
- Turnkey solution
- User-supported software
- Utilities
- Waterfall model
- Word-processing