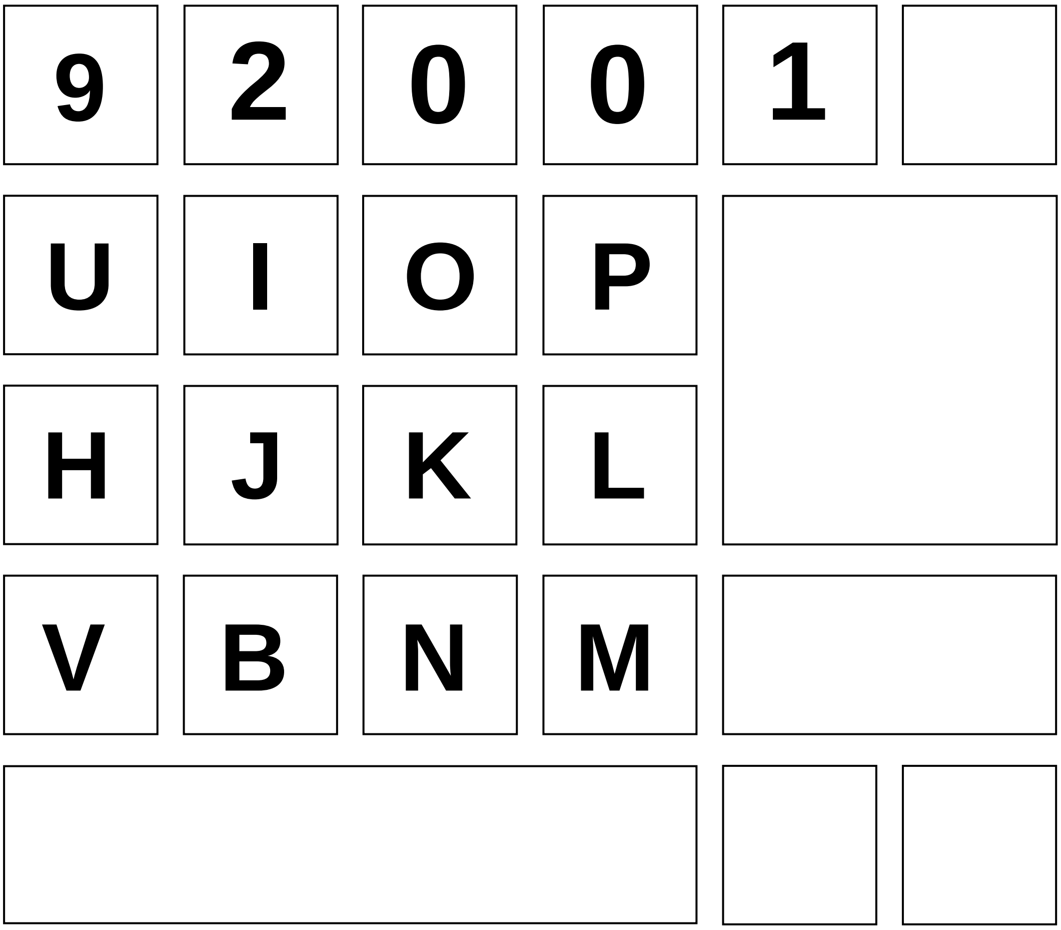
METI
Ministry of Economy,
Trade and Industry

Textbook for
Fundamental Information Technology Engineers

NO. 2 SYSTEM DEVELOPMENT AND OPERATIONS



Second Edition

REVISED AND UPDATED BY

IPA

Japan Information-Technology Engineers Examination Center
INFORMATION-TECHNOLOGY PROMOTION AGENCY,JAPAN

Contents

1. System Development

Textbook for Fundamental Information Technology Engineers

No. 2 SYSTEM DEVELOPMENT AND OPERATIONS

First edition first printed September 1, 2001
Second edition first printed August 1, 2002
Third edition first printed August 1, 2003

Japan Information-Technology Engineers Examination Center
INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN
Center Office 15F, Bunkyo Green Court 2-28-8, Hon-Komagome, Bunkyo-Ku,
Tokyo, 113-8663, JAPAN

1 System Development

Chapter Objectives

For corporations, it is essential to pursue profit. The current economic situation, however, is chaotic and it is difficult to predict future trends. In such a situation, information-processing systems become vitally important to business activity, and therefore crucial to the existence of corporations. Information processing engineers, all designing and developing information-processing systems, have become increasingly important.

This chapter is about development flows that use the waterfall model, which provides a base for system development, development environments, management systems, and the use of software packages. In detail, the following items will be learned:

① Types and outlines of system development methodologies, and development processes using the waterfall model.
② Diagramming for requirement analysis, and various design technologies.
③ Programming languages and programming techniques.
④ The importance of test, and various design methods for test cases.
⑤ Tools used in development environments, and techniques and systems for development management.
⑥ Types of software packages and how to use them.

Introduction

About 50 years have passed since the machine called "computer" came into being. Initially, no theory or methodology existed for systems development. So, the development depended solely on engineers' "craftsmanship." It is quite similar to the situation in the old days when carpenters built houses based on their "intuition" and "experience." However, systems have gradually grown larger, and the expansion of and modification to existing systems have become necessary, together with new developments. In addition, development productivity has become highly sought after. The situation has reached the point where the use of "craftsmanship" is totally insufficient, leading to the postulating of system development theories, and the engineering methods to implement them.

In addition to conventional methods, new development methods and theories have recently been introduced that provide higher productivity and more user-friendliness.

In this chapter, basic theories for system development and their methodologies are learned. The effective use of recent software packages and methods of review essential to development works are also covered.

1.1 Development Methodologies

Relationships between enterprise systems, information technology, and major system development methods are also described.

1.1.1 Role of System Development Organization

(1) Enterprise Activities and Information Systems

Various efforts are required for continuous growth of an enterprise. Effective use of information technology (IT) is vital.

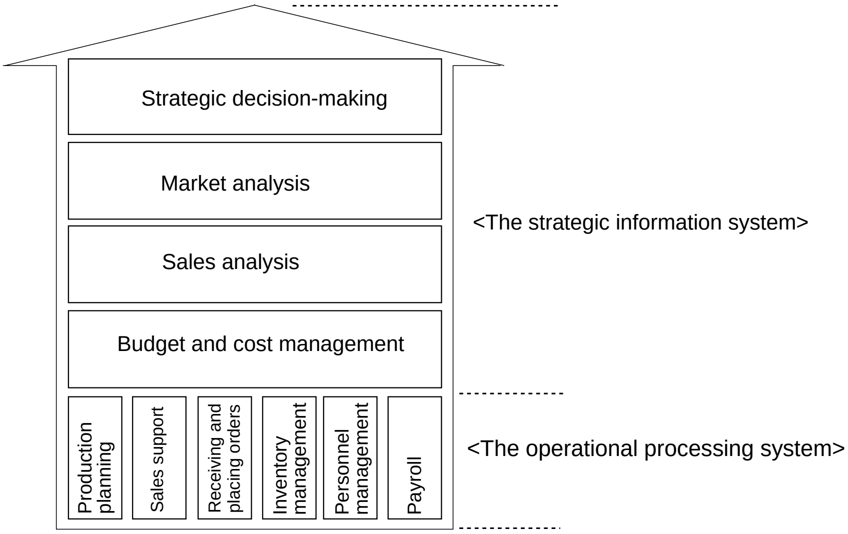
For enterprises, there are two types of information:

- Internal information : Information that is generated through business activities, and includes various slips, forms and management documents that are used in sales, production ordering and accounting.
- External information : Information generated depending on the economic situations surrounding corporations, including sales of product, trends in the industries concerned, moves of competing companies, and transactions with related companies.

Enterprises effectively use these pieces of information in their daily activities, with information systems as the means. Information systems are classified into the following two categories:

- The operational processing system : Used to support daily operations and to provide business management data. Periodical information processing is conducted for increasing productivity, and improving operation efficiency.
- The strategic information system : To achieve various business objectives, it is important to use the resources available, humans, materials, and money, in the most effective way. The system provides information required to manage these resources. The system is mainly used to generate reports from management viewpoint.

Figure 1-1-1
The strategic information system and the operational processing system


Strategic decision-making
Market analysis
Sales analysis
Budget and cost management
Production planning
Sales support
Receiving and placing orders
Inventory management
Personnel management
Payroll
<The strategic information system>
<The operational processing system>

The objectives of each system are as follows.

<The operational processing system>

- Reduction in manpower for operations concerned
- Reduction of inventory
- Reduction of business processing
- Realization of paper-less operations
- Reduction of delivery time

<The strategic information system>

- Increase in sales
- Improvement of customer satisfaction
- Improvement of sales efficiency
- Creation of new markets

Generally, an organization in charge of developing information processing systems is called the system development organization or something similar, and system engineers (SEs) and programmers belong there. On the other hand, an organization using developed systems is called the user organization.

(2) Progress in Information Technology and the System Development Organization

① Progress in information technology

Recent advances in technology are astonishing. Of the various advancements, the following are considered as factors affecting system development significantly.

a. Progress in computer technology

Performance improvement of both personal computers and workstations, and the extent of price reduction are amazing. Operations that once could be performed only with mainframes can now be executed with these smaller computers.

b. Wide-spread use of software packages

Because of the rapid development of software packages, those for RDBMS (Relational Data Base Management System) and spreadsheets are easily available, and it is also possible to integrate them into some of systems.

c. Progress in network technology

Technology innovation in the information and communication area, including the combination of LAN (Local Area Network) and WAN (Wide Area Network) and the construction of intranets and extranets (networks made by expanding intranets to the outside of the company), is remarkable.

d. Progress in system construction technology

The technology has shifted from the process-oriented approach to the data-oriented approach (DOA). As a result, notations such as Data Flow Diagram (DFD), Entity-Relationship Diagram (ERD), and Hierarchy plus Input Process Output (HIPO) created to represent structured designs, have been widely used. In addition, CASE (Computer Aided Software Engineering) tools have become available to support development efforts, and have become widely used as well.

② Present situations of system development organizations

With the increase of system scales and the introduction of multimedia, system development organizations are confronted with the following problems:

a. Increase in the amount of backlogs

The number of backlogs, which indicate systems whose development can't be started immediately following a development request from a user organization, has been increasing. Every company reportedly has 2 to 3 years worth of backlogs on average.

b. Introduction of multimedia systems, and an increase in the number of large-scale systems

Multimedia systems, in which various data, including voice and video as well as text are used, have come into being. In addition, WAN and LAN have been combined and intranets has been constructed. Therefore, they are now required to handle increasingly more complex and larger-scale systems.

c. Increase in maintenance work

With the scale of systems enlarging, the amount of maintenance work increases. The amount grows because system modification requests from user departments increase, work to modify existing systems becomes necessary, and work to modify the entire system due to the finding of bugs also increases.

③ New roles of the system development organizations

In addition to the conventional works of system development and maintenance, the system development organizations is now expected to perform the following related work as well:

a. Development of systems and their operation and maintenance

In addition to the conventional works, the organizations must adopt new technology and methods aggressively.

b. Construction and operations of databases and networks

Various databases are essential to business operation and management. In addition, it is now impossible to consider business activities without the use of networks. The construction, operation and management of these databases and networks are considered essential work of the organization.

c. Planning and coordination for company-wide informationalization

The planning of large information systems covering the entire company, and the reflection of users' or company executives' views, are also an important work of the organization.

d. Tying-up with users

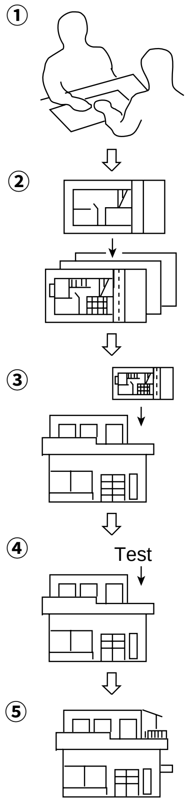
It is expected that the organization accepts users' demands positively and develops ever more realistic systems. It is also necessary to willingly offer the technical support necessary to conduct end user computing (EUC), and to off training sessions and on-the-job training (OJT) on information processing in the company.

1.1.2 Software Development Model

For system development, various models are used that depend on the scale of the company and the ways of work prevalent in a company. Here, these methods are described briefly. In particular, a detailed description on the waterfall model is given in 1.1.3.

As shown in Figure 1-1-2, examining the building of a house makes it easier to understand the way system development progresses.

Figure 1-1-2
A procedure for building a house



- A client's request for a house will be made to the builder. Based on the requests, the builder offers cost estimates, schematic drawings and schedules, and talks with the client. In addition, he applies for a building permit from the government offices concerned.
- Based on the schematic designs, the builder produces detailed designs that take into consideration necessary constraints such as the geographical and social conditions of the location and costs. Finally, the designs are broken down to the level that allows for actual construction.
- Assembly is carried out by carpenters based on the designs.
- Checks are conducted each time the assembly of a part is completed. At the end, checks are performed with the client in attendance. Then, the house is handed over to the client.
- Various kinds of maintenance will be conducted according to requests from the client.

With schematic diagrams alone, it is difficult for the client to have a clear image of the house and how it will look when the building has been completed. Therefore, checks with a model house, or 3-D images on a computer have recently been offered, as well.

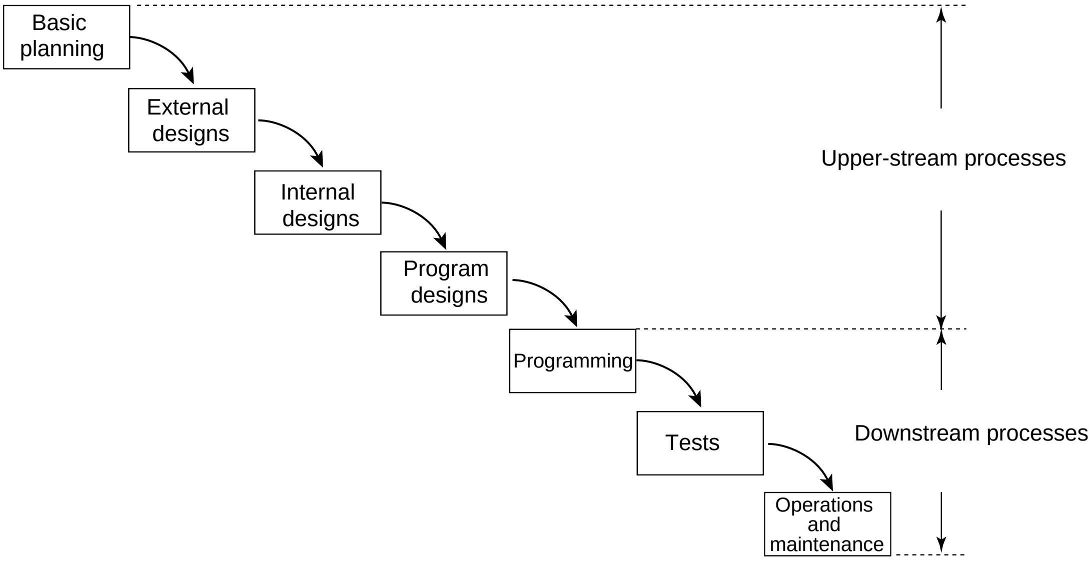
Similar handling is also used in system development.

(1) Waterfall Model

The waterfall model, a system development technology, is still the most widely used one. In the model, the work is divided into a number of phases, and management is conducted for each phase.

As indicated by the name 'waterfall,' work progresses in the model from the upper-stream (the basic planning) to the downstream (testing), never flowing backwards.

Figure 1-1-3 The waterfall model



(2) The Prototype Model

The waterfall model involves the following problems.

- With the waterfall model, it is extremely difficult to grasp users' requirements in the basic planning phase for a system. Sometimes even customers do not know such requirements.
- Design diagrams and verbal explanations are sometimes insufficient.

To solve these problems, the prototype model has been devised. With the prototype model, a system to be constructed is roughly modeled with a simplified programming language like SQL (Structured Query Language), to help the understanding of the customer. Then, intended development work is started.

The prototype includes various models.

<The classification by construction methods>

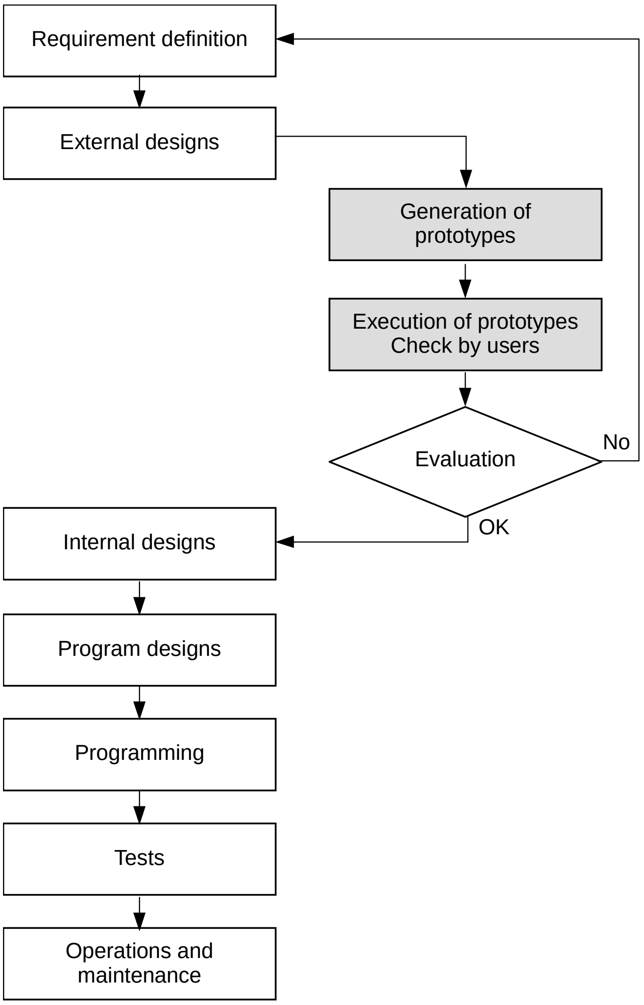
- "Throw-away type": Trial pieces are disposed of after their objectives are attained.
- "Skeleton type": Details are subsequently added to a trial piece to gradually expand it to the intended system.

<Classification by the extent of coverage>

- "Partial use type": The model is used in the phases of requirement definition and external designs.
- "Total use type": The model is used for all the phases.

Figure 1-1-4 shows a flowchart example of the prototype model of the partial use type.

Figure 1-1-4 A flowchart example of the prototype model of the partial use type



Use of the prototype model brings about a sense of participation to the users, preventing errors in upper-stream phases that significantly affect subsequent work.
However, the prototype model involves the following problems yet to be solved.

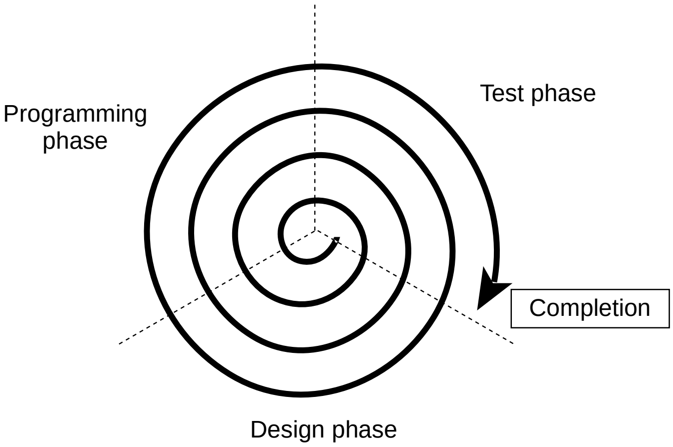
- The development cost exceeds that of the waterfall model.
- Schedule adjustment is difficult.

(3) The Spiral Model

In the spiral model, a series of processes composed of design, programming and test are repeated for each sub-unit of a system, with the development made iteratively and multiplicatively (see Figure 1-1-5).
<Characteristics>

- Use of the model is suitable for cases where sub-units of a system to be developed are comparatively independent of each other.
- It partially follows ways the waterfall model performs.
- It allows the prototype model to be used when the need arises.
- Used in object-oriented type development and in others.

Figure 1-1-5
The spiral model



(4) Object-oriented Development

Recently, attention has been focused on object-oriented development. In this model, a system is considered as a set of objects, and development is conducted on an object basis. In an object development, the process of analysis to design to implementation is repeatedly performed, constituting a kind of spiral model.

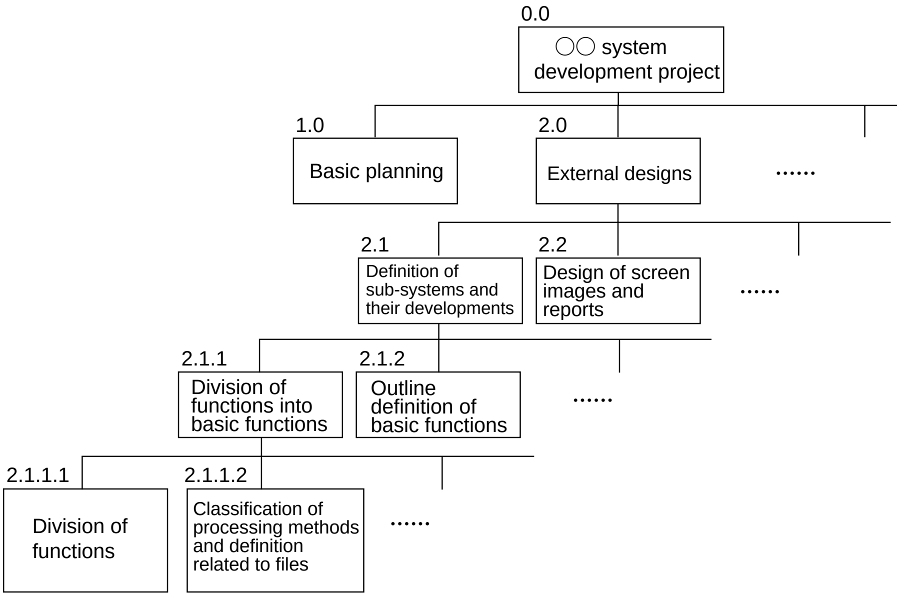
(5) Work Breakdown Structure

To achieve its objectives a system development project is broken down into the following levels, in this order, depending on the progress of the development.

1. The level deciding the major structure of the project
2. The work level that constitutes the framework of each phase
3. The detailed current work level

The Work Breakdown Structure (WBS) is derived by adding concrete objectives, work schedules, and progress management specifying details at a level as fine as possible to what is obtained by these breaking-down operations. WBS is represented with a hierarchical structure as shown in Figure 1-1-6.

Figure 1-1-6
A work breakdown structure


0.0
○○ system development project
1.0
Basic planning
2.0
External designs
......
2.1
Definition of sub-systems and their developments
2.2
Design of screen images and reports
......
2.1.1
Division of functions into basic functions
2.1.2
Outline definition of basic functions
......
2.1.1.1
Division of functions
2.1.1.2
Classification of processing methods and definition related to files
......

Use of WBS provides the following merits.

- Cost estimates and data for the cost analysis are provided.
- Work structure and work coverage of a project and responsibility for the work are clarified.
- Grasping actual progress for each work unit, and the planning of work are made easier.

- Names of broken-down work units and the classification system are a piece of expertise know-how.

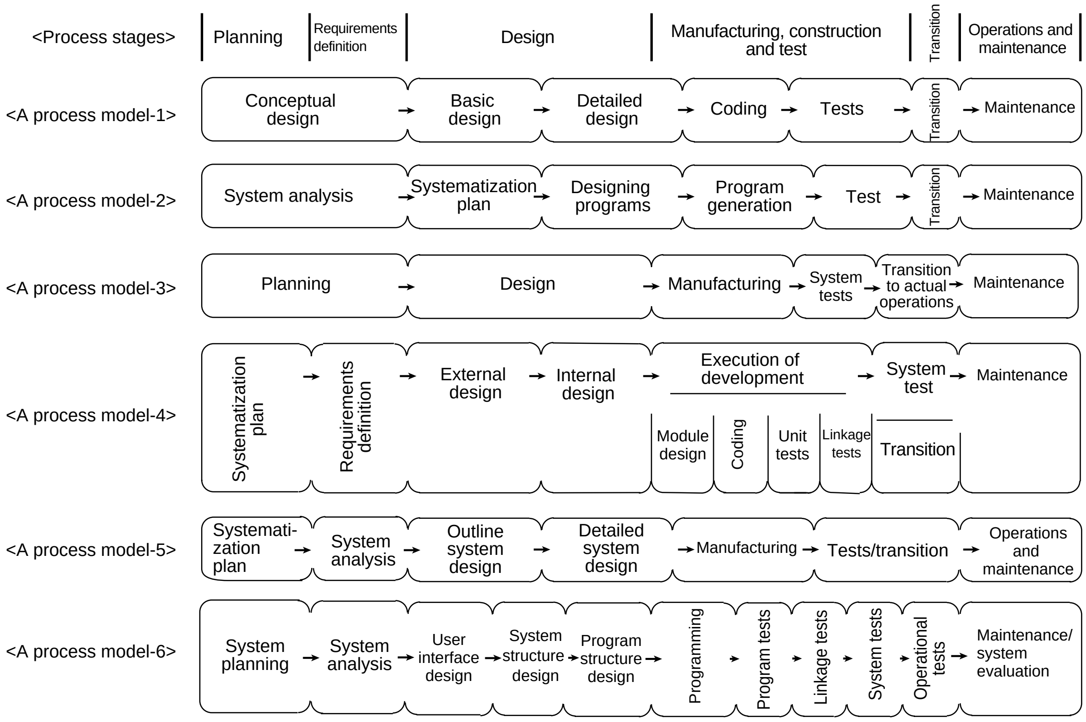
With WBS, targets of work, such as quality, cost, and time, are given on a work unit basis. So, work is performed with the targets as the references.

(6) Process and Process Model

The process is defined as what work units, such as analysis, design and manufacturing, are needed in producing products (including software products), and are arranged in a time series. Each component of a process is called "process stage." With WBS described above, work is represented hierarchically, but not in a time series. This makes a big difference.

A process is roughly designed in the basic planning phase. In this design, a process model shown in Figure 1-1-7 may be used as a reference.

Figure 1-1-7 A process model

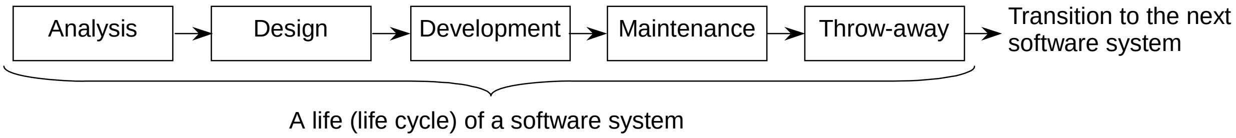


Software products are produced through each of the development processes. Therefore, designs of each process, giving a base of the process, largely affect the quality and cost of the software product.

1.1.3 Software Life Cycle

The life cycle is the process from the birth to the death or the life span of a living thing and a product. Likewise, with the concept of the software life cycle (SLC), the period between the start of a project to develop a system, and the time when the system update is finished is considered a life of the system. Then, activities occurring during that period are represented with the actual life used, as a model to express relationships between processes.

Figure 1-1-8 The life cycle of a software system

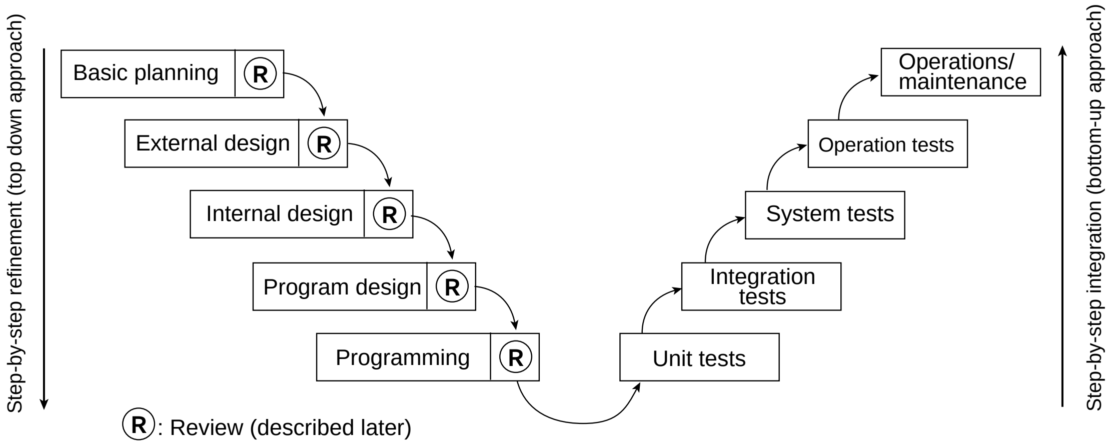

Analysis
Design
Development
Maintenance
Throw-away
Transition to the next software system
A life (life cycle) of a software system

In the following, the waterfall model which is the most typical is explained.

(1) Characteristics of the Waterfall Model

Figure 1-1-9 shows an overall image of the waterfall model.

Figure 1-1-9 An overall image of the waterfall model (V-shaped structure)



In the waterfall model, the following techniques are used. Therefore, it is possible to visualize the model as the V-shaped structure shown in Figure 1-1-9.

- The basic planning phase to the programming phase	:	Step-by-step refinement method (top-down approach)
- The unit test phase to the operation test phase	:	Step-by-step integration method (bottom-up approach)

The characteristics of the Waterfall Model are summarized below:

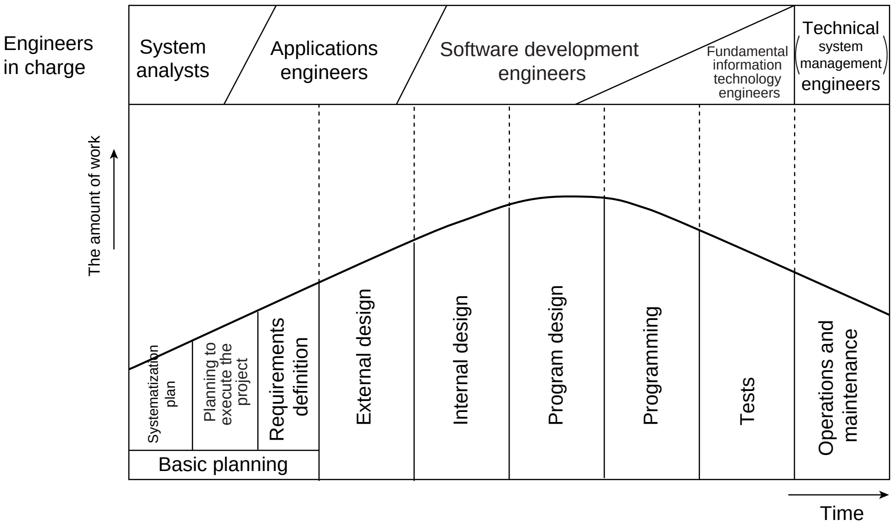
< Characteristics >

- A system development work is divided into a number of phases for management.
- When work for a phase is completed, work products (including various design documents) from the phase are reviewed to check correctness.
- Work products (including various design documents) from a phase are then turned over to the following process, where work is carried out with those outputs received. In this way, consistency in a system development is maintained.
- Basically, it is not permitted to return work to an upper phase.
- How to organize the project is of critical importance.

As described above, dividing a system development process into a number of phases for management is one of the characteristics of the waterfall model. Figure 1-1-10 shows a relationship between each phase and the amount of work there.

In (2) to (8) below, an outline of each work phase in a system development is described.

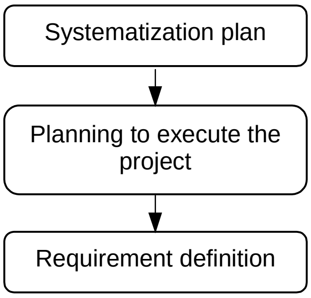
Figure 1-1-10 Relationships between each phase in the waterfall model and the amount of its work



(2) Basic Planning

Basic planning is the first step of a system development. It is necessary to have a thorough knowledge of the existing operations that are targeted for computerization. Otherwise, it is impossible to develop a system that can satisfy the users. Consequently, basic planning starts with an analysis of the existing system, and the identification of its problems.

Figure 1-1-11 Basic planning



Detailed procedure in the basic planning are described below:

① Systematization plan

The systematization planning is a work to draw up basic plans for a system.

<Tasks>

a. Investigation and analysis of problems in operations targeted for systematization.
b. Based on results from item a, the best solution is explored and the necessity for a system development is reexamined. If it is found that a better alternative solution exists, the alternative is adopted.
c. If it is found from the work on item b that a new system development is suitable, a systematization plan is created and proposed to the person in charge.

<Documents>

- A systematization plan

② A project implementation plan

After the person in charge approves the systematization plan, an implementation plan (a project implementation plan) is generated.

<Procedure>

a. A project is organized (including the assignment of the person in charge)
b. A system resource plan (estimates) is worked out.
   - Personnel plan

  - Hardware for the system development
  - Estimates of the development scale (including manpower and budget)
  - Financing

  And others

c. A work process plan and the highest-level schedule are drawn up.
  There are the following types of schedules.
  - Highest-level schedule : The schedule for an entire system
  - Middle-level schedules : Schedules for each phase in a system development
  - Lowest-level schedules : Schedules for each individual person concerned

  It is desirable, if at all possible, to work out all of the schedules correctly at this phase, but in practice it is difficult. So, it is at least necessary to compile the highest-level schedule at this phase.

<Documents>

- Development plans

③ Requirement definition

In requirement definition, targeted operations constituting inputs to system development, and requirements for the information system are analyzed and defined in detail more than in the development plans, with structured methods such as DFD (Data Flow Diagram) and ERD (Entity Relationship Diagram).

<Tasks>

a. Information about the system, such as the work in the targeted operations (a work model generated here is called an existing logic model), forms used and objectives, is collected.
b. Requirements for the system as a whole, including the functions, performance and operation requirements, are defined.
c. Requirements for both hardware and software are clarified.

<Documents>

- Requirement specification

④ Summary of the basic planning

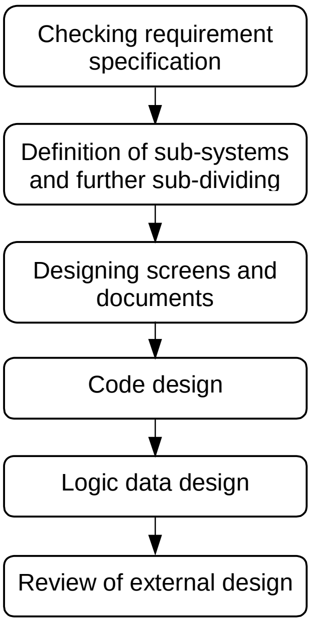
The basic planning is the phase where an outline of an information system is designed, including the analysis of a system to be developed and the drawing-up of the highest-level schedule. Results of it affect following processes significantly. In addition, it is the first phase in a system development. Therefore, it is necessary that the work there is performed thorough confirmation with users, mainly by systems analysts having advanced skills and core knowledge.

(3) External Design

External design is designs for externally visible parts or those for user interfacing. Here, the system is designed purely from the users' viewpoint without considering the constraints of hardware (such as computers). In addition, system structures required to achieve such designs are clarified.

In external design, the work indicated in Figure 1-1-12 is carried out.

Figure 1-1-12 External design


Checking requirement specification
Definition of sub-systems and further sub-dividing
Designing screens and documents
Code design
Logic data design
Review of external design

<Tasks>

a. Checking requirement specification
   After checking requirement specifications included in the basic plan, the overview of the system is represented by the use of diagrams so that the processing and flows of data can be easily understood. Based on the representations, the division into sub-systems and input/output designing are performed. DFDs (Data Flow Diagram) or HIPOs are used for diagramming.
b. Definition of sub-systems and further sub-dividing
   The entire system is divided into a number of sub-systems on a functional basis, then a sub-system is further divided into smaller units.
c. Designing screens and documents
   In designing screens and documents, rough designs of screens and screen transition maps, and rough designs of input/output documents are produced. (Detailed work to implement the designs is conducted in the internal design.)
d. Code design
   Here, the designing of the code, such as the determination of a coding system, is conducted.
e. Logical data design
   In logical data design, relationships between the data are analyzed, and candidates for databases and files are picked up. (Detailed work for this is conducted in the internal design.)
f. Review of external design
   External design documents are reviewed.

<Documents>

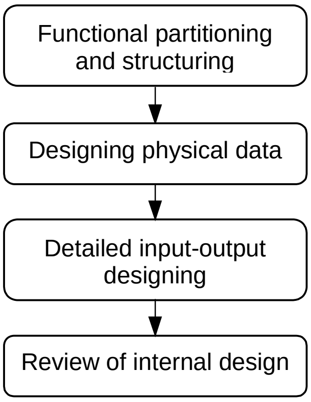
- External design documents
- External design documents review reports

(4) Internal Design

The internal design is for the invisible parts of a system, and deals with designs that are viewed from the computer or system development side. In the external design, the system is designed from the user's viewpoint. In the internal design, however, the details are designed by considering how efficiently these external designs are implemented on the computer, or by taking into account the hardware constraints as well as software aspects.

In the internal design, the work indicated in Figure 1-1-13 is carried out.

Figure 1-1-13 Internal design


Functional partitioning and structuring
Designing physical data
Detailed input-output designing
Review of internal design

<Tasks>

a. Functional partitioning and structuring (structured design)
In functional partitioning and structuring (structured design), each sub-system is partitioned into programming units, and the flows of data and processes among programs are clarified.

b. Designing physical data (designing files)
In designing physical data (designing files), for the effective use of hardware characteristics physical designing, of files and databases is conducted based on logical data designs which were performed in the external design phase.

c. Detailed input-output designing
In detailed input-output designing, details of screens and input-output documents are designed using specially provided forms.

d. Review of internal design
Review is carried out for internal design documents

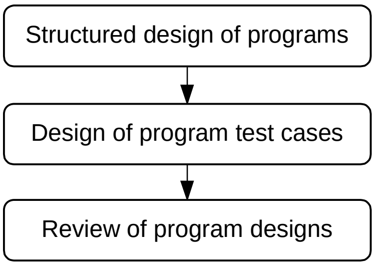
<Documents>

- Internal design documents
- Internal design review reports

(5) Designing Programs

In designing programs, internal structures of each program are designed. Each program that is derived through the division in the internal design phase is further divided hierarchically into functional units called modules. Then, interfaces (input/output data) between modules are designed.
In addition, plans for program (integration) tests are prepared, and the test cases are set up as well.
In designing programs, the work indicated in Figure 1-1-14 is carried out.

Figure 1-1-14
Designing programs


Structured design of programs
Design of program test cases
Review of program designs

<Tasks>

a. Structured design of programs (module partitioning)
In structured design, each program is partitioned into functional units called modules to allow easy maintenance, and the flow of data and processes among programs are clarified. In addition, the functions of each module and interfaces between the modules are determined.

b. Design of program (integration) test cases
In designing program (integration) test cases, plans for program tests are prepared and test cases are designed.

c. Review of program designs

In the review of program designs, review of program design documents is conducted.

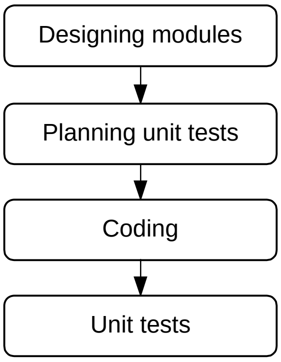
<Documents>

- Program design documents
- Program design review reports
- Program (integration) test plans

(6) Programming

In programming, the designing of the logical structures of modules defined in the program design phase, and the coding of the modules are conducted. In addition, plans for module (unit) tests are prepared, and the test cases is set up.

Figure 1-1-15 Programming



<Tasks>

a. Designing modules
   In designing modules, the internal logical structures of modules (detailed processing procedures within each module) are designed with the use of various structured techniques.
b. Planning unit tests
   Unit test plans are prepared. (Appropriate test data are produced, and test schedules are determined.)
c. Coding
   Each module is coded in a programming language.
d. Unit tests
   Unit tests are conducted for each module.

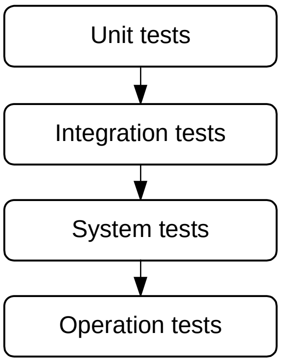
<Documents>

- Module design documents
- Module design review reports
- Unit test plans
- Source program lists
- Source program list review reports
- Unit test reports

(7) Test

Testing work is conducted to detect errors in the behaviors and structures of modules, programs or a system as a whole (see Figure 1-1-16). If errors are found, a feedback is made, if necessary, to the designing or programming phase for correction. Then, tests are conducted again to check whether the errors have been actually corrected or not.

Figure 1-1-16 Various tests



<Tasks>

a. Unit tests (conducted in the programming phase)
In unit tests, each module are checked to see whether it is executed correctly or not.

b. Integration tests
In integration tests, tests are conducted for each program produced by linking modules. Operations of programs and interfaces between modules are checked.

c. System tests
In system tests, the operation of the system as a whole is totally checked from the viewpoints of required objectives and performance. Then, the start of actual operations is decided based on the results.

d. Operation tests
In operation tests, operation groups from the user department conduct tests under the same conditions and environments as in the actual operations.

<Documents>

- Unit test reports
- Integration test reports
- System test reports
- Operation test reports

(8) Operations and Maintenance

The production run of the developed system is started. Maintenance operations are conducted when a defect (such as difficulty in use) or a bug is found, or when a change is unavoidable. In some cases, the system must be modified.

1.1.4 Software Reuse

Systems have been developed on built-to-order basis. Based on a user's requirements, developers analyze targeted operations, and design, program, and test the system before completing its development. Then, the user facilitates his operations using the system which has been built meeting his specific requirements.

Now, however, many companies are annoyed by their backlogs (development items yet to be started). The built-to-order base production is causing the increase in backlog. With the built-to-order base production, the development of a system takes between several months and several years. In addition, the participation of experts like system engineers (SEs) is required to develop a system. However, many of these system engineers' efforts are actually consumed in maintenance work of existing systems, which means the number of personnel who can be spared for new development projects is small. These situations have produced the idea of reusing existing software.

In software reuse, portions or the whole of a piece of software are made parts, or existing software is analyzed, and modified, and as a result new software is obtained. In such a way, a piece of software can be used repeatedly in various systems.

There are the following ways of reusing software:

- Reuse as parts
- Reuse through reengineering

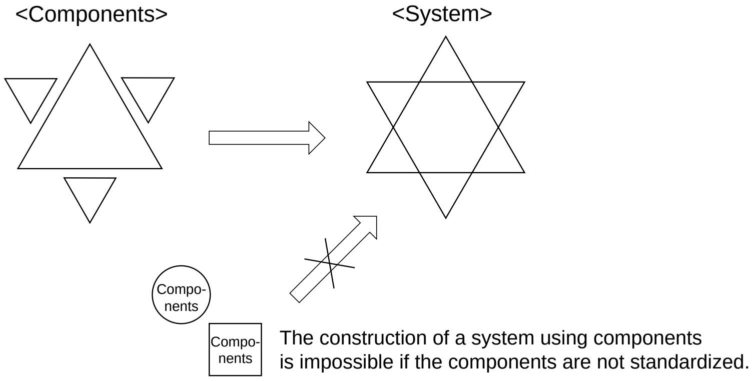
(1) Reuse as Components

The method of creating components from existing software and then assembling them to build a new system offers cost reductions and an increase in system quality. In addition, development periods can be shortened.

① Methods to create components

Parts are created by users themselves or supplied by software vendors. Vendor-supplied parts are standardized. However, the standardization of specification formats, and representations are also necessary for user-created parts. The reason is that components which are not standardized can not be assembled properly. Furthermore, while what languages are used to create components is basically not an issue, it is certainly better to use the same language considering that source code may then have to be modified.

Figure 1-1-17
Standardization of components



The construction of a system using components is impossible if the components are not standardized.

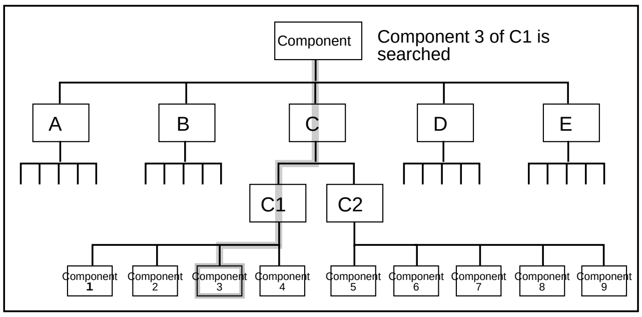
Parts, whose source programs are supplied, can be customized (modified), if necessary, for use in a more proper way.

② Component libraries and search systems

To construct systems using software components, the components are stored and maintained in a component library. The larger the number of the components stored in the library, the more flexible the construction of systems becomes.

Furthermore, when the search system is improved, the component library will be used more frequently, improving the components database and increasing the opportunities of using the library.

Figure 1-1-18
Component library



• The existence of an enriched component library and easy-to-use tools produces a synergistic effect.

③ Improving components-creating tools and components-searching tools

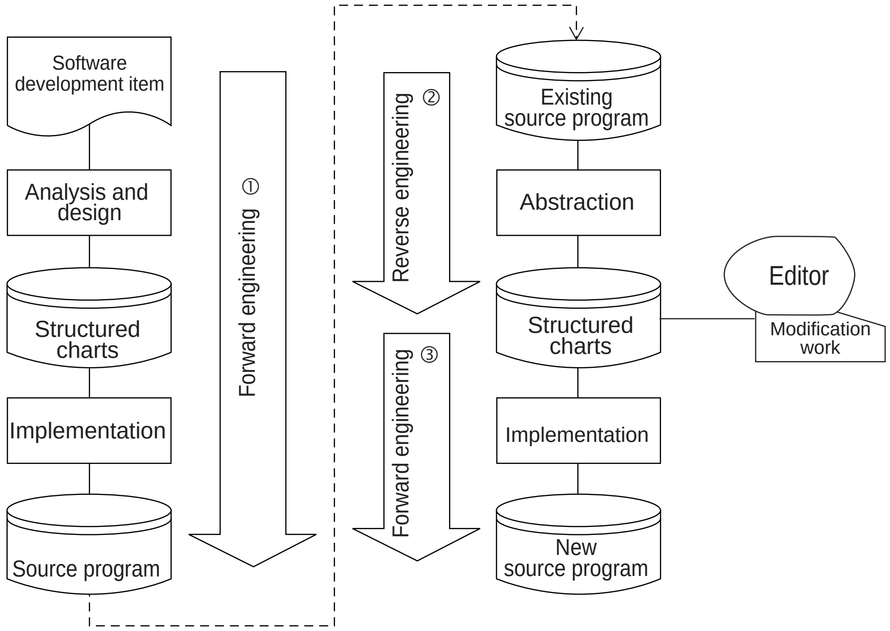
When software components are created and easy ways of searching components from component libraries are offered, software reuse will be accelerated because of the ease of use. Consequently, it will become possible to provide more improved components-creating tools and components-searching tools.

Methods or tools to develop application software by assembling software components that are created, based on standard specifications to enable reuse, are called "componentware."

(2) Reuse through Reengineering

Creating new software from existing software is called "reengineering."
Reengineering is the technology that is used to create a new system by utilization of a system that is actually in use.
To perform reengineering, CASE tools (such as maintenance CASE tools) are often used.

Figure 1-1-19
Reengineering



① Forward → ② Reverse → ③ Forward = Reengineering

In ordinary reengineering, specifications of existing software are derived as the first step. The technology used in this part of the work is called "reverse engineering." Then, specifications for new software are created by modifying the derived specifications. Based on the new specifications, a new system is produced. The traditional development technology used here is called "forward engineering" in contrast with "reverse engineering" (see to Figure 1-1-19).

1.2 Requirement Analysis and Design Method

After the development project implementation plan for a development item requested by a company executive or an end user is approved, the first thing conducted in the development project is to analyze requirements. Here, typical analysis and design methods are described.

1.2.1 Diagramming Methods

In analyzing requirements, functions of a targeted application system are picked up by creating an outline model of the system, in order to construct an information system for the business operation. Here, DFD, ERD, state transition diagrams and UML, all used for aiding the analysis works, are described.

(1) DFD (Data Flow Diagram)

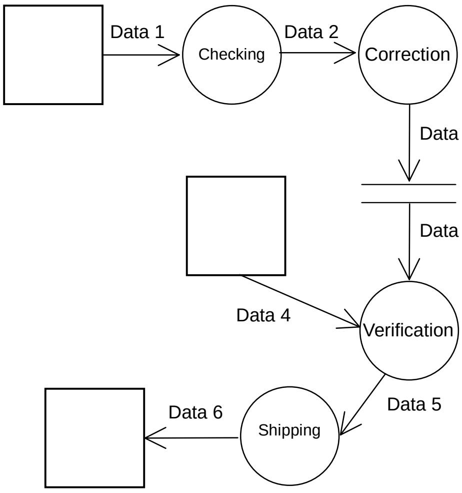
DFD (Data Flow Diagram) is an analysis method done by the use of diagramming, and represents business processes visually to aid understanding by paying attention to the data flow.

Figure 1-2-1 DFD

- Symbols used in DFD

Symbol	Name	Meaning
→	Data flow (An arrow with a name)	The flow of data is indicated
○	Process (Bubble)	Processing/conversion of data is indicated
=	Data storage (Two parallel lines)	Stored data are indicated (Files, databases and others)
□	External (A square)	A source or destination of data is indicated (Ordinarily, persons or organizations outside of analysis)

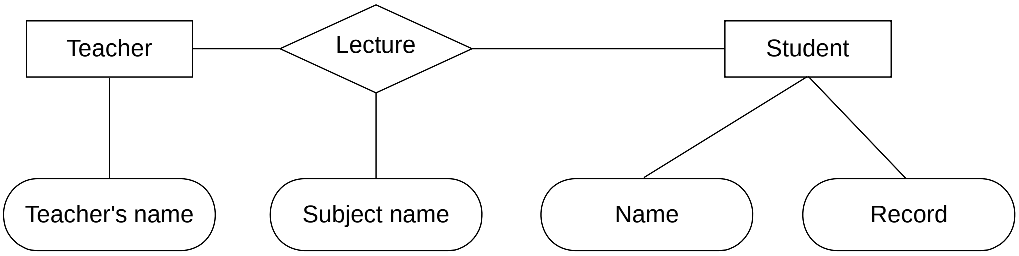
- An image of DFD



(2) ERD (Entity Relationship Diagram)

ERD offers a data model in which a targeted world is represented with two concepts of "entity," and "relationship" between entities. ERD is also called an "business operation model" because it is used for creating business operation models.

Figure 1-2-2 ERD



ERD is composed of the following three elements:

- Entity : Indicates an entity targeted for management; represented with a rectangle.
- Relationship : Indicates relationships between entities, and relationships including those between an entity and a relationship; represented by a diamond.
- Attribute : Indicates characteristics and properties of an entity or relationship; represented with an ellipse.

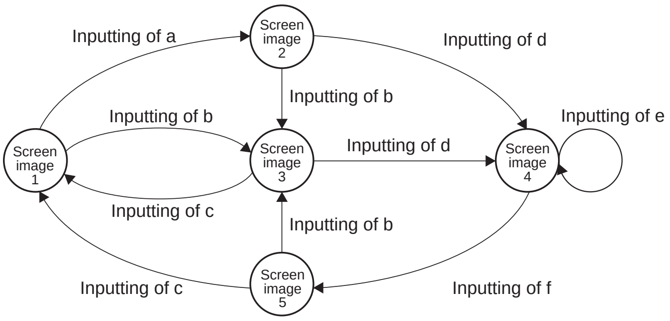
ERD in Figure 1-2-2 indicates the followings:
- "Teacher" and "Student" are connected with a relationship named "Lecture."
- "Teacher" has a "Teacher's name."
- "Student" has a "Name" and a "Record."
- "Lecture" has a "Subject name."

Three types of relationships exist: "1 to 1," "1 to N" and "N to N" where N indicates an integer larger than 1. In this example, if a teacher gives a lecture to more than one student and a student attends different lectures given by more than one teacher, respectively, the relationship is N to N.

(3) State Transition Diagrams

The state transition diagram is used to schematize situations whose states vary depending on time and behaviors. The state transition diagram of Figure 1-2-3 indicates how screens change by operations on the screens (such as) inputting data and pushing a function key.

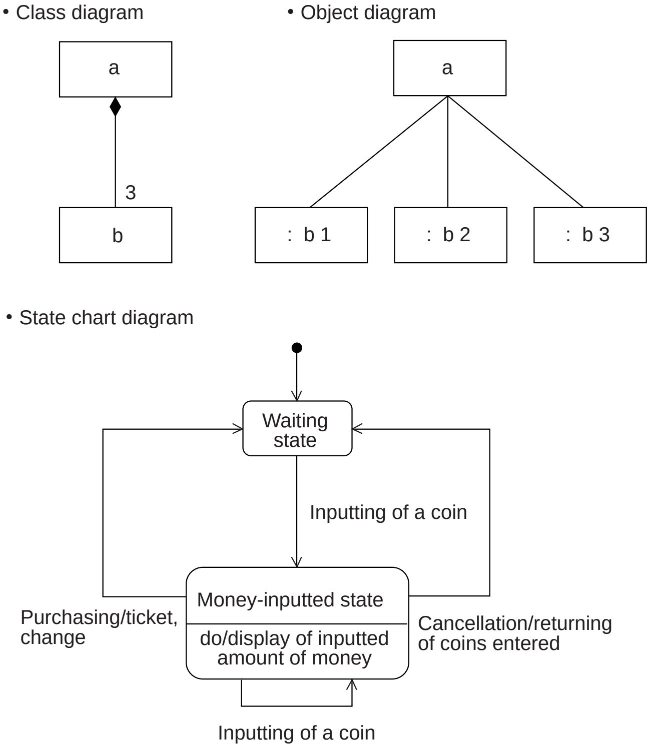
Figure 1-2-3
A state transition diagram example



(4) UML (Unified Modeling Language)

UML is a modeling language used in object-oriented analysis. There have been several representation methods each depending on the person who proposed it, and they have not been unified. UML has been devised to standardize them.

Figure 1-2-4 UML



1.2.2 Analysis/Design Diagramming

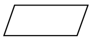
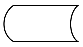
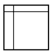
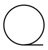
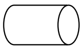
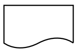
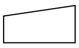
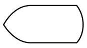
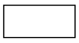
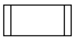
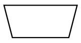
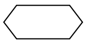
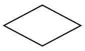
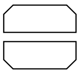
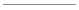
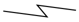
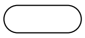
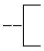
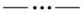
System functions defined in upstream processes are achieved with a number of programs. The flowchart is the most typical diagramming method used to analyze the logical structures. Here, the flowchart, structured charts and decision table are described as diagramming methods for analysis and design.

(1) Flowcharts

Based on the definition/analysis of complex problems and processing procedures as well as range of work, the flowchart is one of the methods, and the most typical one, that is used to represent ways to solve them and the flows of the necessary operations, with unified symbols.
Figure 1-2-5 shows symbols used in flowcharts, which are specified in JIS X 0121.
The flowchart is comprised of the following two types:

- The system flowchart (process chart)
- The program flowchart

Figure 1-2-5 The symbols used in flowcharts

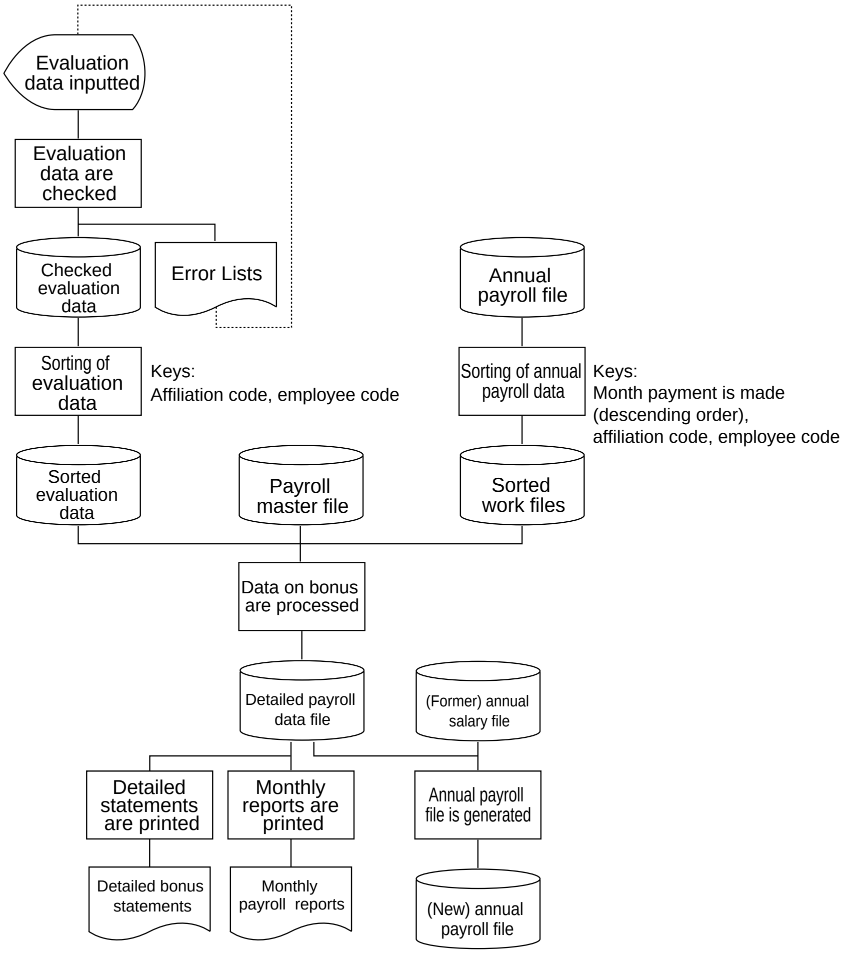
Symbol	Name	Descriptions
	Data	Indicates data for which media is not specified.
	Stored data	Indicates data stored in formats suitable for processing (media are not specified).
	Internal storage	Indicates data stored in an internal storage medium.
	Sequential access storage	Indicate only sequentially accessible data, such as those on magnetic tapes, and cassette tapes.
	Direct access storage	Indicates directly accessible data, such as those on magnetic disks, and floppy disks.
	Documents	Indicates data on media human beings can read.
	Manual input	Indicate data on any medium that is inputted by hand-operated means, such as online keyboards, switches, push buttons or bar code.
	Display	Indicates data on any medium that displays information used by human beings, such as those on display screens or online indicators.
	Process	Indicates any type of processing functions.
	Predefined process	Indicates processing composed of one or more operations or instructions defined, for example, a subroutine.
	Manual operation	Indicates any operation handled manually.
	Preparation	Indicates the modification of an instruction or a group of instructions that, like the setting-up of switches or the initial setting-up of a routine, affects later operations.
	Decision	Indicates a decision function that, with an input port and output ports only one of which can be selected at a time, selects one output port depending on the result of the conditional evaluation depicted in the symbol.
	Loop limit	Composed of two parts; conditions for termination and a loop name are written in the symbols indicating the start and the end of the loop, respectively.
	Line	Indicate a flow of data or control.
	Communication link	Indicates that data is transferred on a communication line.
	Connector	Indicates an output port to another place on a flowchart or an input port from another place.
	Terminator	Indicates an output port to, or an input port from, an external environment.
	Comment, annotation	Used to add explanations or notes for clarification.
	Ellipsis	Used instead of a line symbol, and indicates that a symbol or a group of symbols are omitted.

*JIS X0121

① The system flowchart (process chart)

The system flowchart (process chart) indicates a flowchart for a targeted system as a whole.

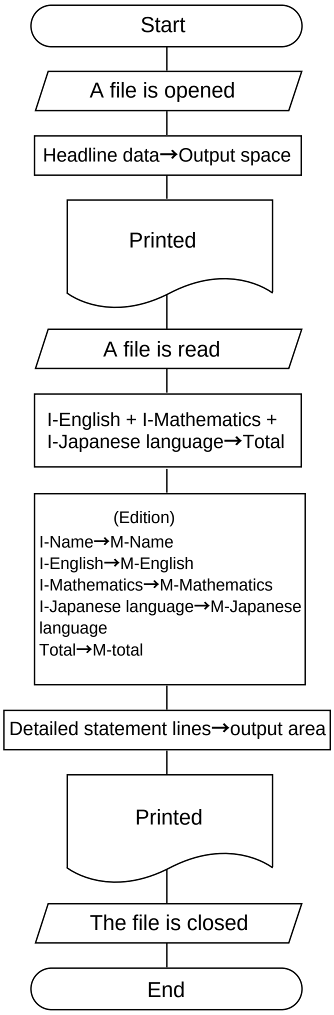
Figure 1-2-6 A system flowchart example


Evaluation data inputted
Evaluation data are checked
Checked evaluation data
Error Lists
Annual payroll file
Sorting of evaluation data
Keys:
Affiliation code, employee code
Sorting of annual payroll data
Keys:
Month payment is made (descending order),
affiliation code, employee code
Sorted evaluation data
Payroll master file
Sorted work files
Data on bonus are processed
Detailed payroll data file
(Former) annual salary file
Detailed statements are printed
Monthly reports are printed
Annual payroll file is generated
Detailed bonus statements
Monthly payroll reports
(New) annual payroll file

② The program flowchart

When programs are developed, the program flowchart is used to describe processing procedures based on analysis results with flowcharts.

Figure 1-2-7
A program flowchart example


Start
A file is opened
Headline data→Output space
Printed
A file is read
I-English + I-Mathematics + I-Japanese language→Total
(Edition)
I-Name→M-Name
I-English→M-English
I-Mathematics→M-Mathematics
I-Japanese language→M-Japanese language
Total→M-total
Detailed statement lines→output area
Printed
The file is closed
End

(2) Structured Charts

With flowcharts, it is possible to describe algorithms based on the structure theorem. However, use of arrow symbols in flowcharts (in other words, allowing use of GO TO statements) may lead to the generation of a non-structural algorithm. On the other hand, in structured charts, algorithms are described without use of any arrow symbol (or GO TO statements). Therefore, descriptions based on the structure theorem (refer to 1.3.3) are possible.

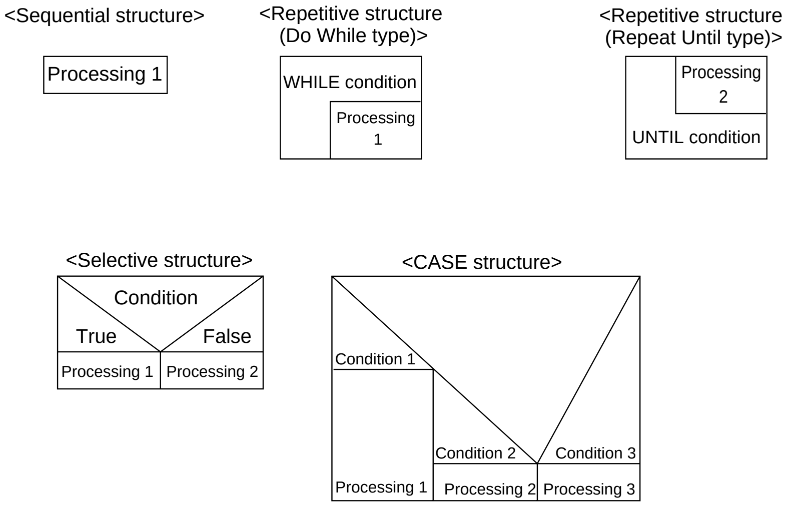
Structured charts include the following different types:

- NS chart
- PAD
- SPD
- HCP
- YAC II

① NS chart (Nassi-Shneiderman chart)

The NS chart uses special symbols having basic control structures for representation.

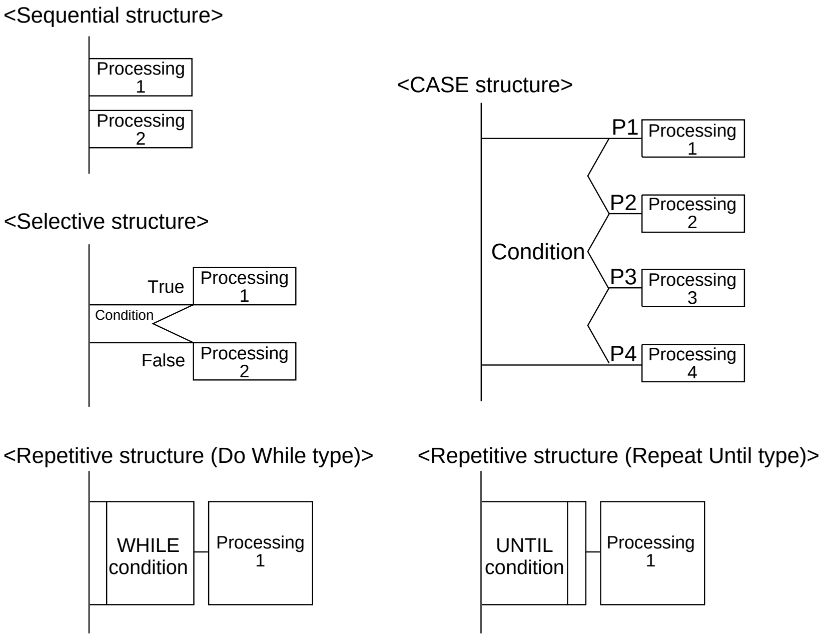
Figure 1-2-8
An NS chart example



② PAD (Problem Analysis Diagram)

PAD describes logic structures of algorithms with tree structures.

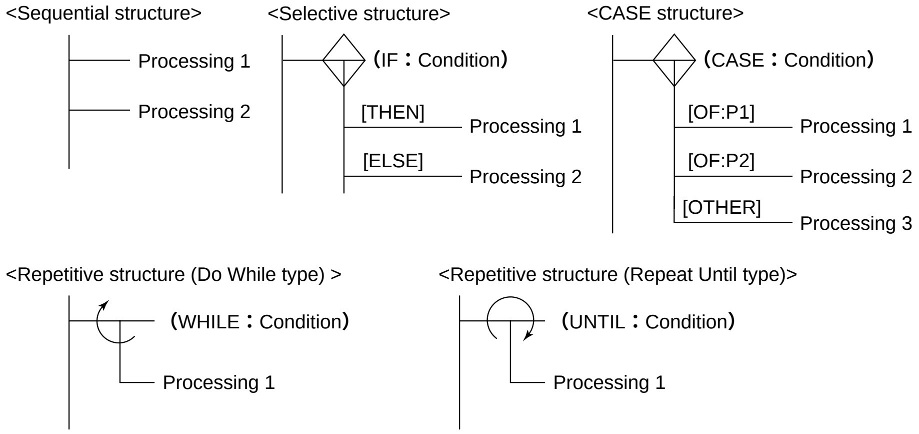
Figure 1-2-9
A PAD example



③ SPD (Structured Programming Diagram)

SPD describes logic structures of algorithms with tree-structured charts.

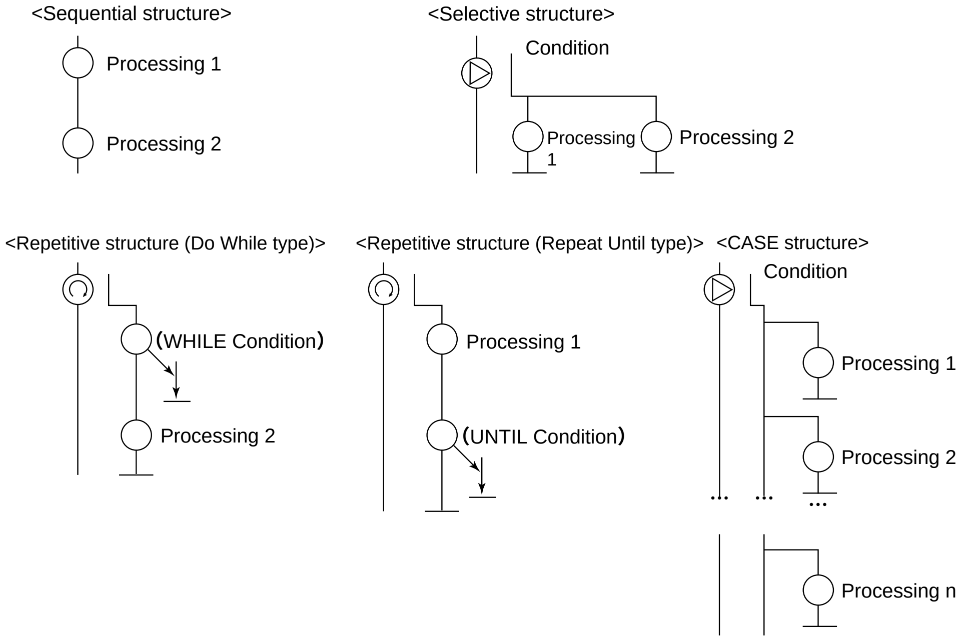
Figure 1-2-10
An SPD example



④ HCP (Hierarchical and ComPact description chart)

HCP describes, in the same way as SPD, logical structures of algorithms with tree-structured charts.

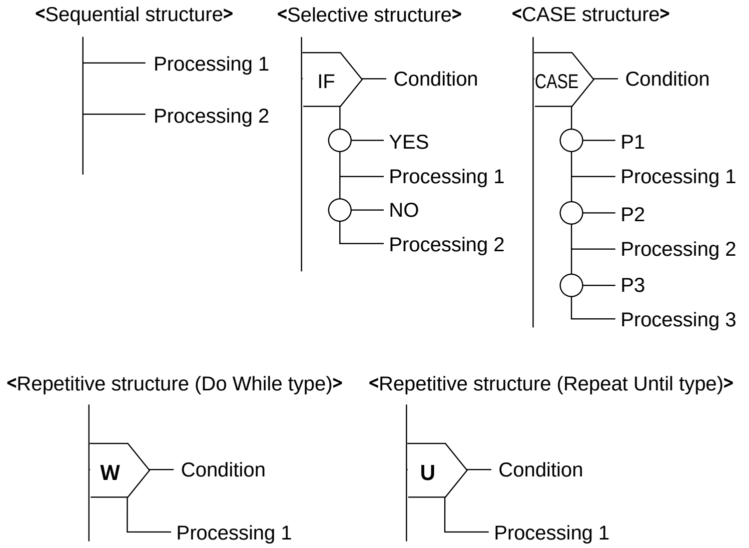
Figure 1-2-11 A HCP example



⑤ YAC II (Yet Another Control chart II)

YAC II also describes logic structures of algorithms with tree-structured charts.

Figure 1-2-12
A YAC II example



(3) Decision Table

The decision table is used to analyze and arrange logical structures of modules, and describes all factors from the definition of problems to their solutions with simple 2-D tables. The decision table allows the simple arrangement of relationships between conditions and processing, even for problems that are complex and include many factors. It enables a smooth transition to coding based on the structure theorem.

Figure 1-2-13
A decision table example

Condition	Condition entry
Conditions to be checked for problems that occur are made in the form of queries.	The answer to a query from a condition stub is described with Y (Yes) or N (No).
Action stub	**Action entry**
All possible actions to be taken for problems defined in a condition stub are described neatly.	Instead of Y or N for the condition entry, X is written only when an action is taken.

Work on weekdays	Y	Y	N	N
Work on holidays (Sundays and national holidays)	N	N	Y	Y
Work in: 09:00 to 21:00	Y	N	Y	N
Work in: 21:00 to 00:00	N	Y	N	Y
Hourly payment: ¥1,000	X	X		
Hourly payment: ¥1,200			X	X
Allowance: ¥1,000 per attendance		X		X

1.2.3 Design Method

(1) Structured Analysis and Design

The structured method is a term combining both structured analysis and structured design. With the structured analysis (SA), requirements for a system targeted for development are analyzed by paying special attention to the functions of the system, and flows of data between the functions. This analysis method uses the following three documents that diagrammatically represent an application business world with data flows for analysis.

- Data flow diagram (DFD) : Allows hierarchical descriptions, which enables the hierarchical division of system functions.
- Data dictionary : Used to define data structures of data whose names are added to data flows in DFD.
- Mini-spec : Continued hierarchical division of the functions leads to basic operations. These basic operations are defined in mini-spec.

The structured analysis was originally meant only for describing data and processes diagrammatically. However, De Marco added the use of data dictionaries and mini-spec to the original method. Consequently, the structured analysis has now been used as a methodology.

The objectives of the structured design (SD) method is to convert processes in DFD generated by the structured analysis into program modules, and conduct a division into modules with a top down approach. The STS (Source Transform Sink) division method, or the transaction division method, is used for the division into modules.

(2) Process-oriented Design

Software is composed of two elements, functions and information (data). In traditional software development, analysis and design have been conducted centered on functions performed by the software. This is called process-oriented design or the process-oriented approach. In this method, no clear standard exists to define function units, and so the definition can be easily influenced by a designer's own way of thinking. In addition, when functions to meet the users' requirements are pursued, it is difficult to adjust requirements for more than one user, and the use of overlapped functions is unavoidable. As a result, a system includes many overlapped functions, which is one more factor that makes maintenance difficult.

(3) Data-oriented Design

Data does not vary because of users' requests or requirements for business operations, but exists in the real world regardless of types of, or relationships among, operations. Thus, in the data-centric design, attention is paid to data, and a system is designed based on data process structures. Furthermore, this design method has evolved into the Data Oriented Approach (DOA). Analysis and design are conducted for data, separated from the users' requests or requirements for operations, and then processes are defined and integrated into data-specific procedures.

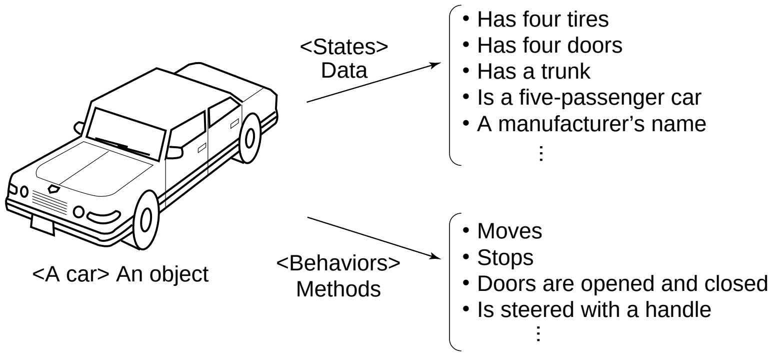
(4) Object-oriented Design

The object-oriented design is derived by further advancing the concept of the data-oriented approach.

① The concept of object-oriented design

Let's consider the object-oriented concept with a car as an example.

Figure 1-2-14 Objects, data and methods (with a car as an example)



Even using a simple word of "a car," it involves each car having its specific states, quite different from any other cars, that includes a type, a year, a model, a door shape, an amount of displacement. In addition, a car involves various behaviors such as "move," "stop," "open a door" and "close a door." Thus, using the fundamental structures and behaviors of an "object" for system development is an essential characteristic of the object-oriented design.

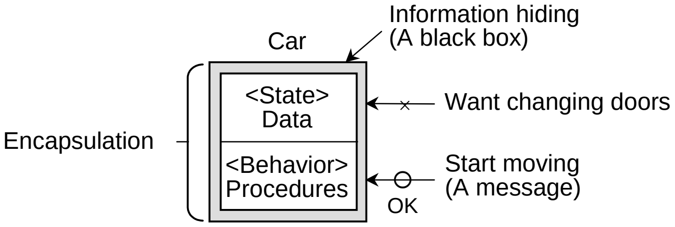
The following relationships are obtained when terms used in the object-oriented design are applied to the example in Figure 1-2-14.

- The template of a car: Class
- An individual car: Instance or object
- States: Data (Attributes)
- Behaviors: Methods (Procedures)

Integrating the "states" and "behaviors" is called "encapsulation."

Furthermore, hiding "states" and "behaviors" by containing them in a black box is called "information hiding." This leads to the prohibition of direct access to data, increasing the independence of objects and making reuse easier.

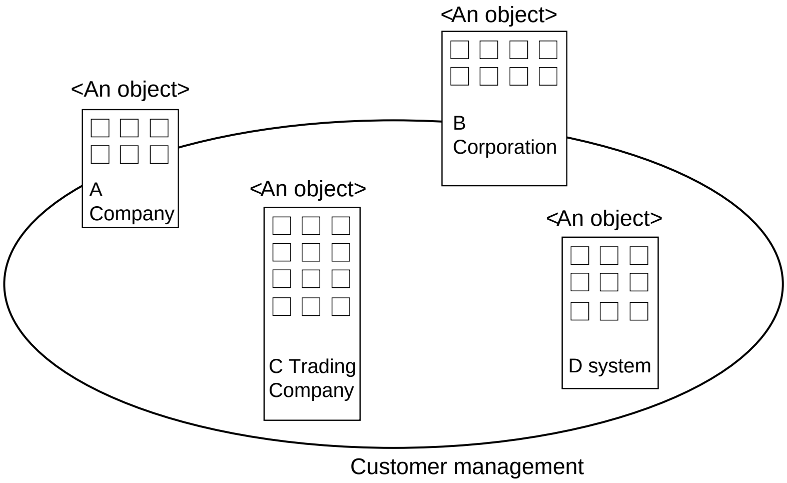
Figure 1-2-15
An example of encapsulation and information hiding



② Object/class/instance

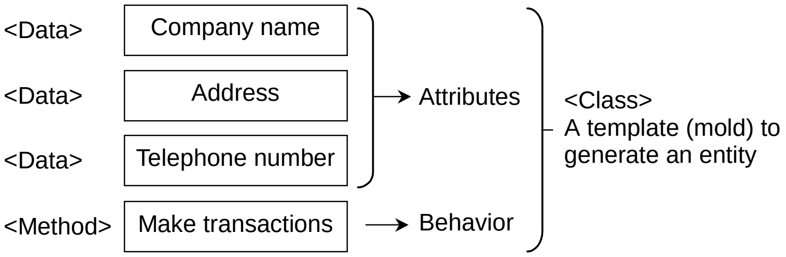
For example, when a customer management application is developed with the object-oriented design, each customer is handled much like an object.

Figure 1-2-16 Object examples (customers)



What is defined as the general attributes and behavior of customers is called "Class." A class is defined by picking up portions (characteristics) which more than one similar object has in common. So, a class can be said to be a kind of template (mold). On the other hand, a behavior specific to an object is called a "Method."

Figure 1-2-17
A class example



Each entity generated from a class is called an "Instance."

Figure 1-2-18 Examples of instances (entities)

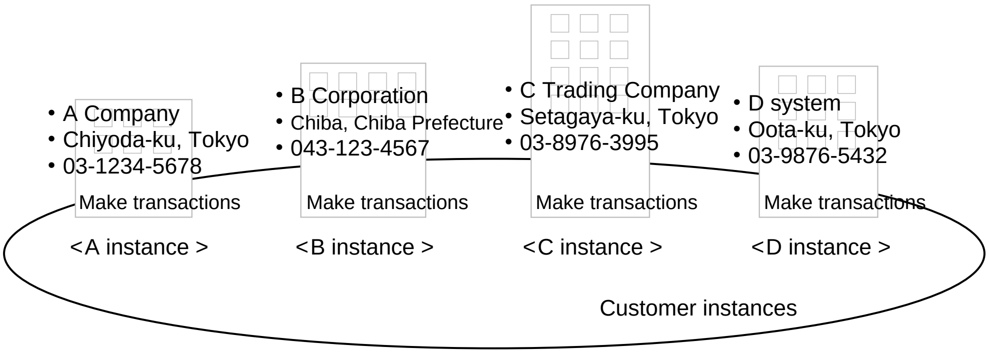
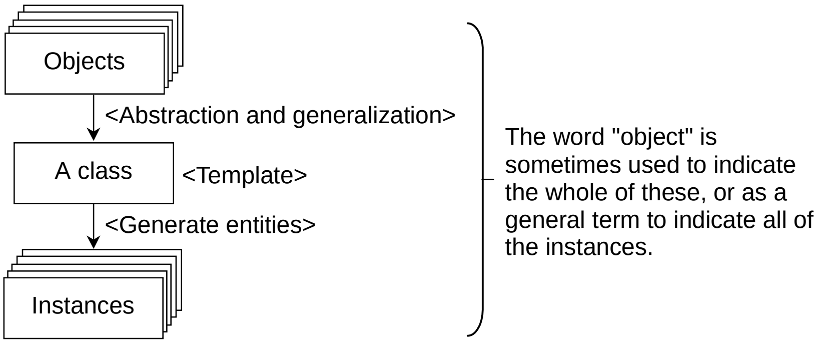


Figure 1-2-19 indicates relationships among objects and a class, and instances described above.

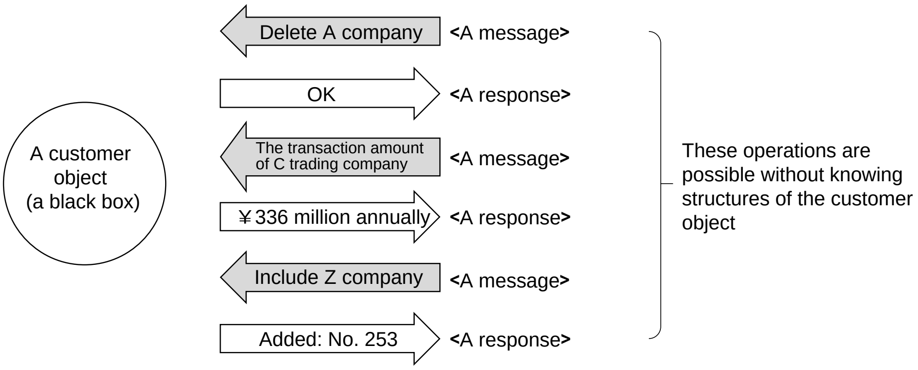
Figure 1-2-19 Relationships among objects and a class and instances


Objects
<Abstraction and generalization>
A class
<Template>
<Generate entities>
Instances
The word "object" is sometimes used to indicate the whole of these, or as a general term to indicate all of the instances.

③ Message

The message is an only means available for giving an instruction to an object. In the object-oriented design, accessing an object by any other means than messages is prohibited, thus increasing independence and reliability of objects. In addition, it is possible to consider an object as a black box, enabling one to perform necessary operations without knowing the contents of an object.

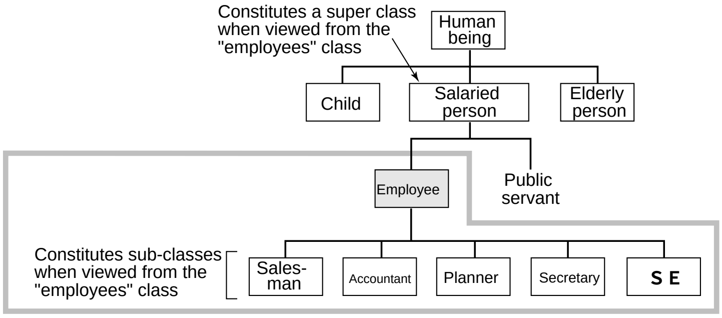
Figure 1-2-20 Message examples


Delete A company
<A message>
OK
<A response>
A customer object (a black box)
The transaction amount of C trading company
<A message>
¥336 million annually
<A response>
Include Z company
<A message>
Added: No. 253
<A response>
These operations are possible without knowing structures of the customer object

④ Inheritance

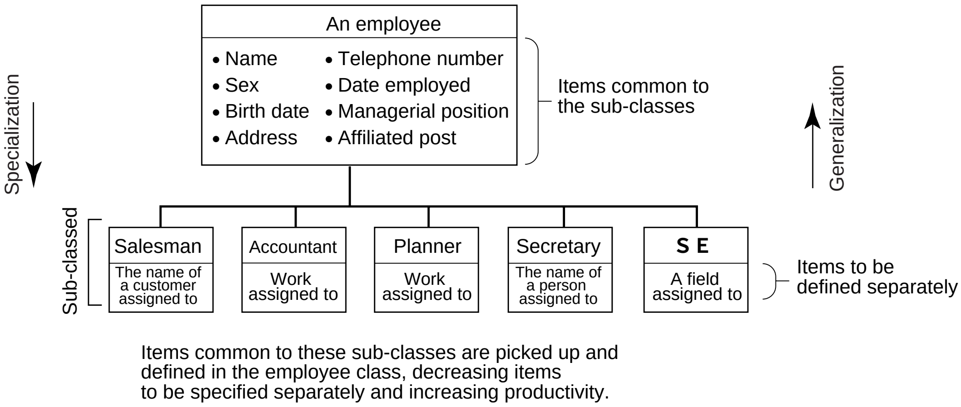
If the increase of productivity is considered an objective in using the object-oriented design, "Inheritance" offers a useful means. The inheritance indicates inheriting data (states) and procedures (behaviors) from other classes (mainly upper classes).
A class placed above a class is called a "Super class," while that placed under a class a "Sub-class."

Figure 1-2-21 A hierarchical example of classes


Constitutes a super class when viewed from the "employees" class
Human being
Child
Salaried person
Elderly person
Employee
Public servant
Constitutes sub-classes when viewed from the "employees" class
Sales-man
Accountant
Planner
Secretary
S E

In the example of Figure 1-2-21, the salaried person class placed above the employee class constitutes a super class, while salesman, accountant planner, secretary and SE classes located under the employees class constitute sub-classes. Each of the sub-classes includes the same items such as name, sex, and address, with only a few items remaining to be defined separately. Use of inheritance is aimed at increasing productivity by defining only the items necessary to be specified separately in each class (programming for this is called difference programming) and by inheriting items common to the sub-classes from the super class "employee."

Figure 1-2-22
An inheritance example

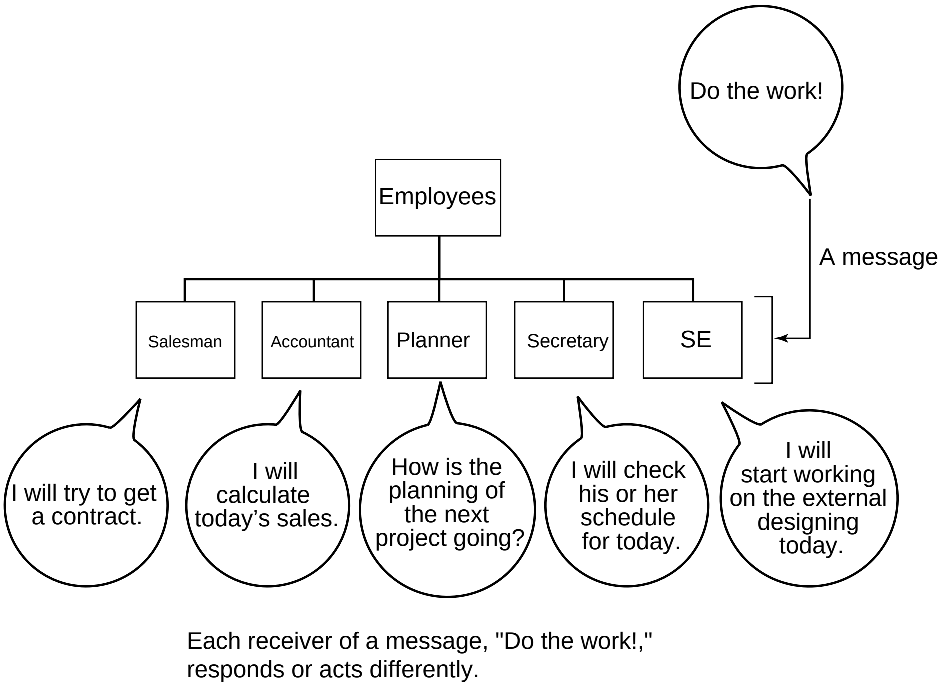

An employee
• Name
• Sex
• Birth date
• Address
• Telephone number
• Date employed
• Managerial position
• Affiliated post
Items common to the sub-classes
Specialization
Generalization
Sub-classed
Salesman
The name of a customer assigned to
Accountant
Work assigned to
Planner
Work assigned to
Secretary
The name of a person assigned to
S E
A field assigned to
Items to be defined separately
Items common to these sub-classes are picked up and defined in the employee class, decreasing items to be specified separately and increasing productivity.

Defining items common to the sub-classes in the super class is called "generalization," while dividing a class and subsequently defining sub-classes is called "specialization." These relationships are called "generalization and specialization relationships" or "is_a relation" (Figure 1-2-21 shows such an example).

⑤ Polymorphism

Polymorphism indicates that each object receiving the same message responds by behaving differently.

Figure 1-2-23
An example of polymorphism

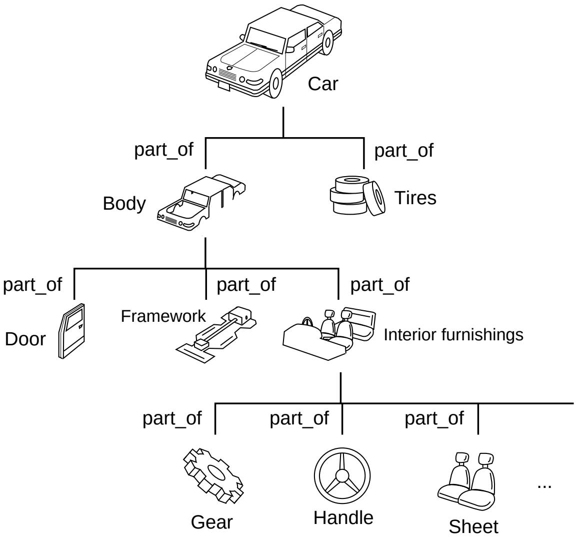

Do the work!
A message
Employees
Salesman
Accountant
Planner
Secretary
SE
I will try to get a contract.
I will calculate today's sales.
How is the planning of the next project going?
I will check his or her schedule for today.
I will start working on the external designing today.
Each receiver of a message, "Do the work!," responds or acts differently.

Polymorphism has a relationship to the encapsulation of objects. In the example in Figure 1-2-23, what the sender of a message has to do is just issue a message, "Do the work!," regardless of the objections of the receivers. In addition, if another type of work is to be added to the side receiving the message, necessary modifications are limited to the receiver side, with no modification required for the sender. Thus, polymorphism, ensuring independence for either side, offers significant favorable effects.

⑥ Compound object

An object made by combining more than one object is called a "compound object."

Figure 1-2-24
A compound object example



In the example of Figure 1-2-24, a car is composed of various parts. If each part is considered an object, the car can be said to be a compound object. Such a structure is called an "Aggregation and decomposition relationship," or a "part_of relation."

⑦ Class library

A class library literally means a library of classes. An objective of the object-oriented design is to increase productivity. Therefore, it is necessary to promote reuse by standardizing class designs as much as possible. the Consequently, the class library offering high-quality classes becomes important.

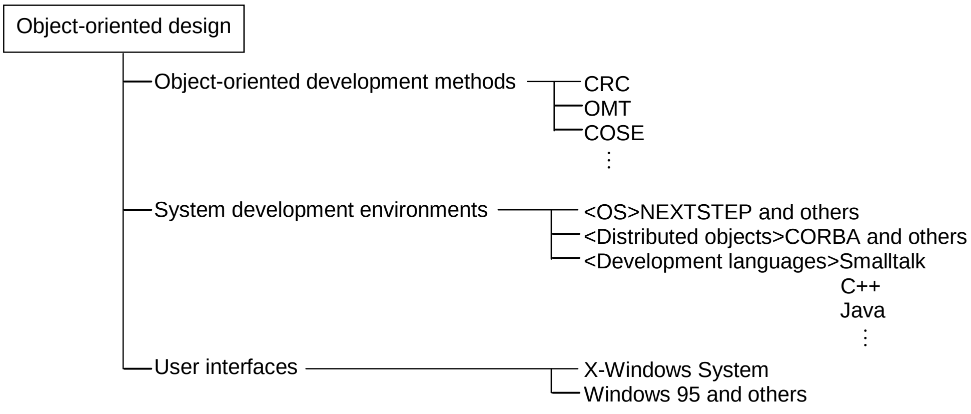
⑧ Delegation

If an object receiving a message cannot handle what is specified by the message, the object sends the message to another object for the processing. Such an action is called "Delegation." In an object-oriented language that uses no inheritance structure, delegation is used in place of the inheritance structure. Since destinations of delegation can be altered dynamically during processing, delegation is more general than the inheritance structure.

⑨ An example effectively using the object-oriented design

Recently, systems in various fields have been developed by using the object-oriented design. As a result , outputs from these developments have been effectively used in various fields. The windows system is a familiar example that effectively uses the object-oriented design. This is because the object-oriented design is suited for constructing "event-driven" systems.

Figure 1-2-25 An example using the object-oriented design effectively


Object-oriented design
Object-oriented development methods
CRC
OMT
COSE
⋮
System development environments
<OS>NEXTSTEP and others
<Distributed objects>CORBA and others
<Development languages>Smalltalk
C++
Java
⋮
User interfaces
X-Windows System
Windows 95 and others

In Figure 1-2-25, CORBA (Common Object Request Broker Architecture) indicates standard specifications in order to use object-oriented technology again in order to coordinate operations of applications on various computers (servers), placed and distributed over networks to perform a piece of processing. This can be said to be a kind of client server system.

(5) Module Design

In the structured design, functional partitioning is performed based on an analysis of the requirement specifications, and each partitioned function is defined as a program. A defined program is further partitioned into a number of modules in module designing. The module is defined as follows:

- The module is a collection of instructions that are used to achieve a function and can be compiled independently.
- Compiled modules can be called upon by other modules, and arguments are used for the interface.

Furthermore, criteria to partition programs into modules are made in order to increase independence amongst modules. The following effects are expected by reducing the extent of dependency amongst the modules and increasing the degrees of their respective independence.

- A structure is introduced amongst a group of modules, and relationships among them become easily understandable.
- The degree in which modifying or correcting a module affects other modules is reduced, minimizing the extent of the resultant effect.
- It can be used as an effective guiding principle to create high-quality software.

The following two measures are used as criteria to judge the degree of independence of software (see Figure 1-2-26):

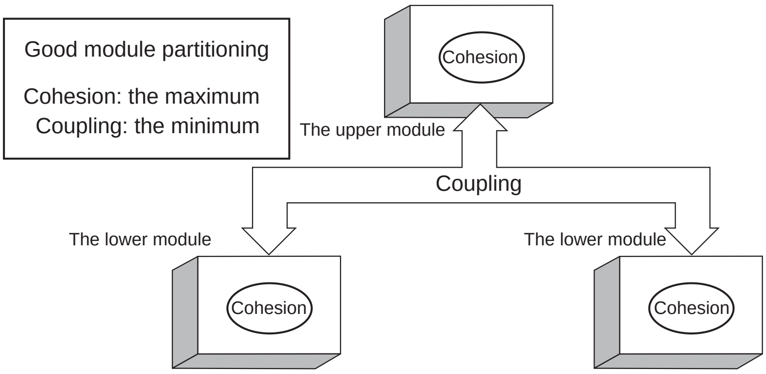
① Module cohesion

Module cohesion indicates the strength in which components composing a module are related to each other, and modules with stronger internal relationships are considered better. The configuration of processes is evaluated here.

② Module coupling

Module coupling indicates the strength to which modules are related. The weaker the relationship, the better the modules are. Ways of handling arguments are evaluated.

Figure 1-2-26 Independence of modules



(6) Screen Design

Of the human interface designs, the screen design is particularly important.

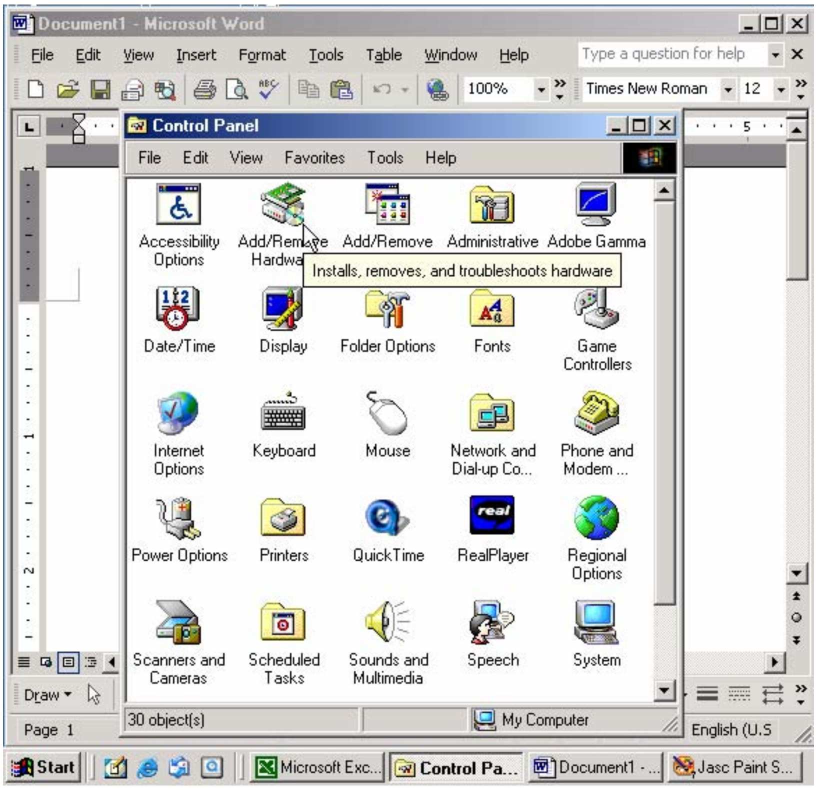
The screen design is so important for users that system quality is sometimes evaluated solely by how easily the screen can be used. Here, the definition of the screen design and its procedure is studied.

① What is the screen design?

In a system, the screen (a place to input data) is the part most familiar to users, because they have frequent contact with it. Recently, screen designs have been made using GUI (Graphical User Interface) abundantly. GUI provides an interface that allows interactive processing by use of easily recognizable icons, dropdown and popup menus. With a pointing device such as a mouse, an arrow or other symbols

which can move to any place on a screen, processing is then carried out by pointing an icon or an item on a menu.
In personal computers, GUI functions are built into operating systems, while, with UNIX, it is necessary to provide GUI functions separately. Icons themselves, their meanings and ways of handling them somewhat vary from one GUI to another. However, they somewhat vary for each GUI.

Figure 1-2-27
A GUI example



② Procedures for designing screens and actual designing work

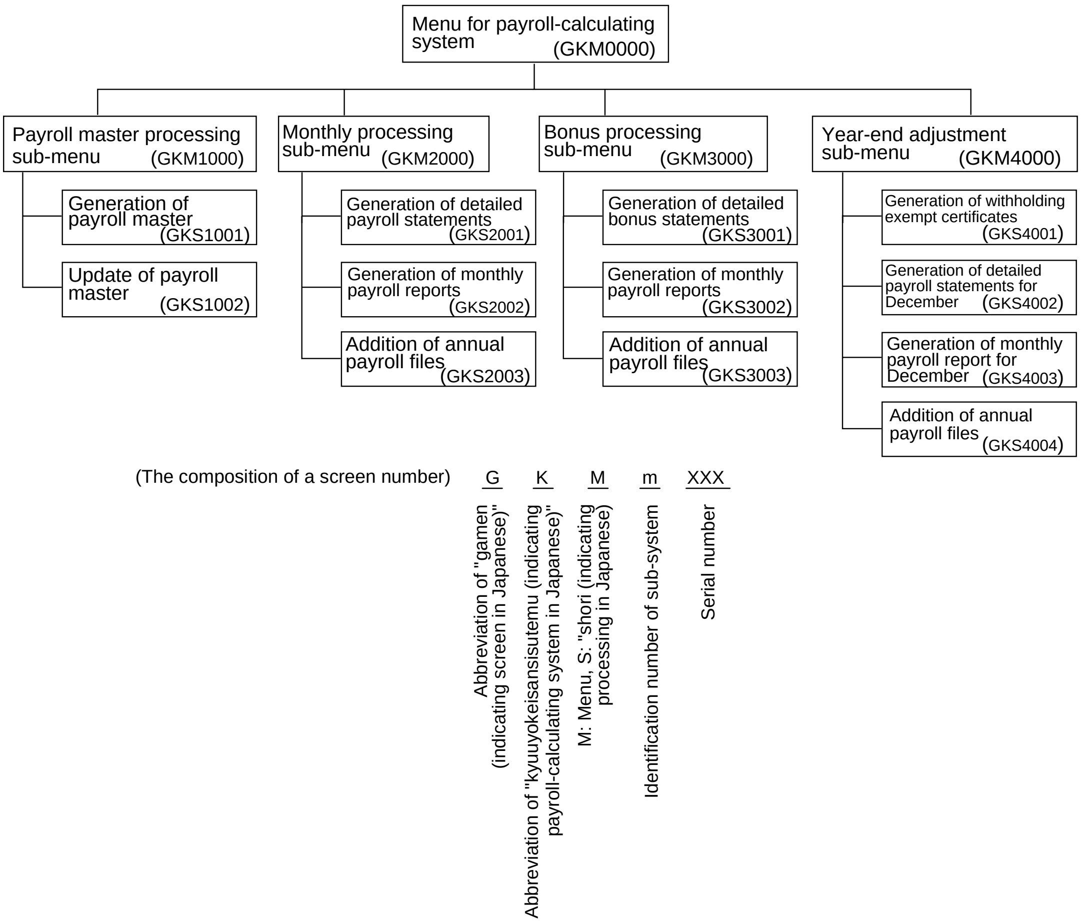
Screen designing is carried out by using the following procedures:

1. An overall picture is created (The entire organization diagrams of all the screens to be used is created).
2. Screen designs are standardized (Screen layouts and flows are standardized).
3. Screen flows are designed (The sequences in which interactive processing is handled are designed).
4. Ways of displaying items on the screen is studied (The frequency of each screen to be used, and the users' level of experience are considered).
5. Screen layouts are designed (Organizations and arrangements on screens are designed).

a. Creation of an overall picture

All the screens used in a system targeted for development are picked up considering their relationships, and an entire organization chart of these screens is created (see Figure 1-2-28). Generally, the chart is mostly composed of menus, sub-menus and processes. Giving a number to each screen makes management easier.

Figure 1-2-28 Creation of an overall picture


Menu for payroll-calculating system (GKM0000)
Payroll master processing sub-menu (GKM1000)
Generation of payroll master (GKS1001)
Update of payroll master (GKS1002)
Monthly processing sub-menu (GKM2000)
Generation of detailed payroll statements (GKS2001)
Generation of monthly payroll reports (GKS2002)
Addition of annual payroll files (GKS2003)
Bonus processing sub-menu (GKM3000)
Generation of detailed bonus statements (GKS3001)
Generation of monthly payroll reports (GKS3002)
Addition of annual payroll files (GKS3003)
Year-end adjustment sub-menu (GKM4000)
Generation of withholding exempt certificates (GKS4001)
Generation of detailed payroll statements for December (GKS4002)
Generation of monthly payroll report for December (GKS4003)
Addition of annual payroll files (GKS4004)
(The composition of a screen number)
G
K
M
m
XXX
Abbreviation of "gamen (indicating screen in Japanese)"
Abbreviation of "kyuuyokeisansisutemu (indicating payroll-calculating system in Japanese)"
M: Menu, S: "shori (indicating processing in Japanese)
Identification number of sub-system
Serial number

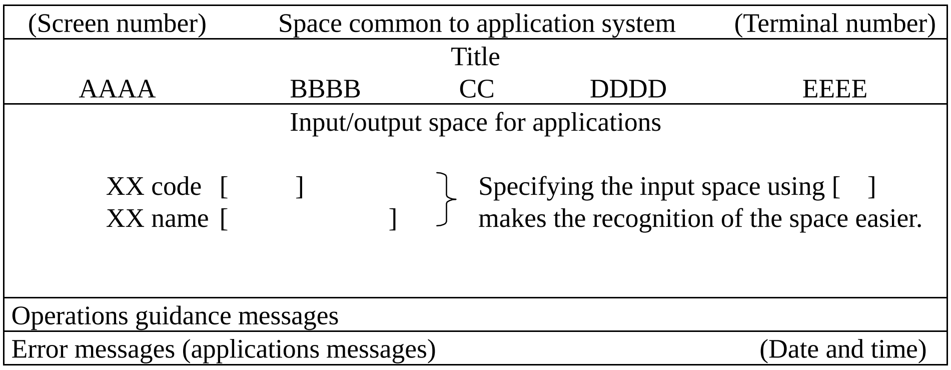
b. Standardization of screen designs

Standardizing screen layouts (locations to place items and others) and screen flows enables increase in productivity of both the development side and user side.

<Items to be standardized>

- Displaying locations (including locations of titles and content items)
- Ways of displaying (including unification of ways of using colors and words and phrases)
- Ways of inputting (selection methods, direct entry, etc.)
- Use of PF keys (program function keys)

Figure 1-2-29 Standardization of screen layouts (an example)


(Screen number) Space common to application system (Terminal number)

Title

AAAA BBBB CC DDDD EEEE

Input/output space for applications

XX code [ ]
XX name [ ]

Specifying the input space using [ ] makes the recognition of the space easier.

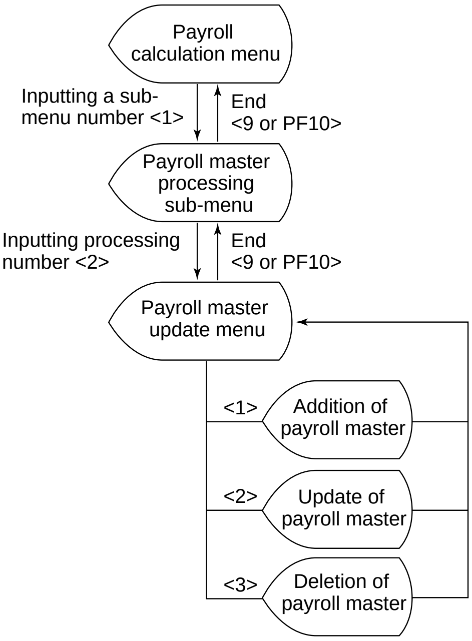
Operations guidance messages

Error messages (applications messages) (Date and time)

c. Designing of screen flows

Application processing needs several screens. In screen flow designs, the order to display each of these screens is designed to execute the processing.
In the designing process, the productivity of both developers and users can be increased, if unified patterns are used for screen flows, for processing and for such things as registration, search, update, deletion and others.

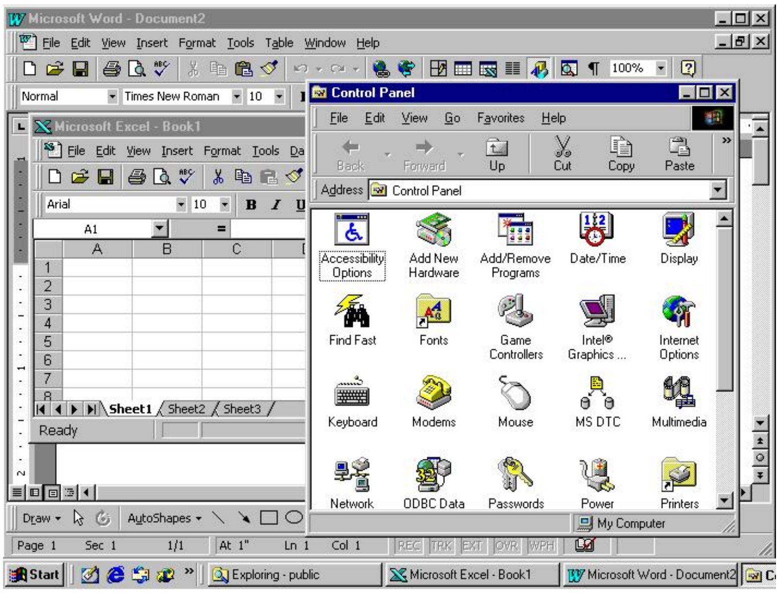
Figure 1-2-30
A screen flow example


Payroll calculation menu
Inputting a sub-menu number <1>
End <9 or PF10>
Payroll master processing sub-menu
Inputting processing number <2>
End <9 or PF10>
Payroll master update menu
<1>
Addition of payroll master
<2>
Update of payroll master
<3>
Deletion of payroll master

d. Study of ways of displaying on the screen

Ways of displaying items on the screen are determined by considering the frequency of screens being used and the level of the user's experience. For example, if novices operate the screens, use of a command-entry method (inputting commands directly into screens) will not increase productivity. Designing based on the user's standpoint is essential in any case.

Figure 1-2-31
An example of multi-windows

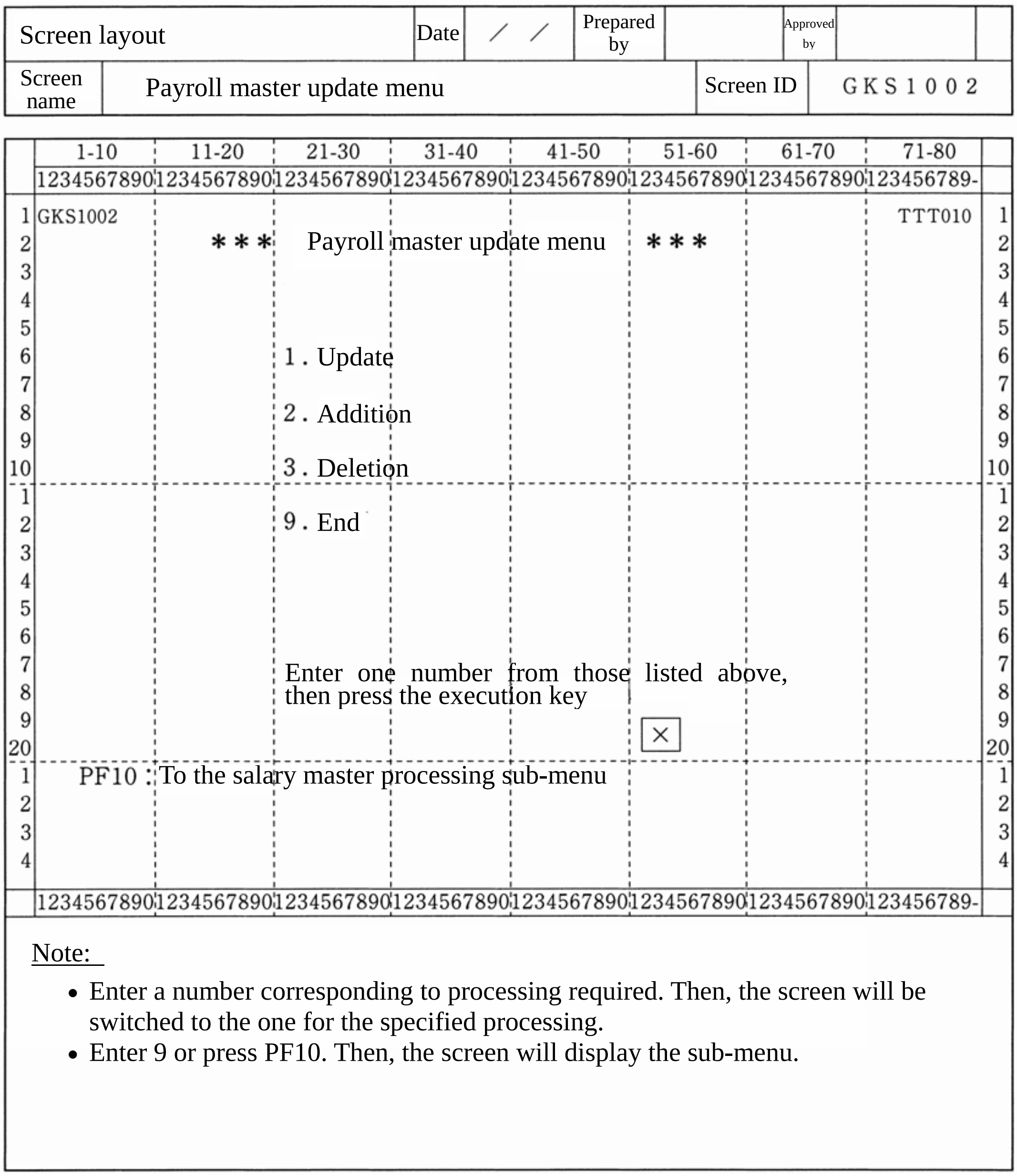


Recently, many multi-window systems, in which more than one screen is displayed simultaneously to enable processing in parallel, have also been used. This type of system is easy to use. However, since it operates with a pointing device, it is necessary to train users to use the device when courses for system operations are held.

e. Screen layout design

Layouts on the screen are designed in this phase. Figure 1-2-32 shows such an example. In some cases, only screen images are checked in external design, and actual designing is conducted later in internal design using specially prepared forms.

Figure 1-2-32 A screen layout example



Screen layout	Date	/ /	Prepared by		Approved by		
Screen name	Payroll master update menu			Screen ID	GKS1002		

```
GKS1002                                                        TTT010
                 ***  Payroll master update menu  ***

                      1. Update

                      2. Addition

                      3. Deletion

                      9. End

                      Enter one number from those listed above,
                      then press the execution key
                                                       ×
      PF10 : To the salary master processing sub-menu
```

Note:

- Enter a number corresponding to processing required. Then, the screen will be switched to the one for the specified processing.
- Enter 9 or press PF10. Then, the screen will display the sub-menu.

Today, prototype models are created with GUI tools and visual languages for evaluations, and the layouts are decided by gaining the user's approval.

(7) Designing Reports

When users use a system, report designs are as important as screen designs. The designs must be easily understandable and easy to use.

Report designs are likely to be taken for designs for documents to be printed, but it should be remembered that results displayed on the screen are reports as well.

① What are report designs?

Reports are used to present results of processing by a system in concrete and easily understandable ways. Therefore, the formats and other things must be considered from the users' standpoint. In external design phase, output images, such as sizes of reports, and the layouts of items in reports are designed, while the designing of those things depending on hardware constraints (like printing methods) and detailed things are conducted in the internal design phase.

② Procedures to design reports and necessary work

Reports are designed with the following procedures:

- Investigation of outputs from overall viewpoints
- Determination of output methods and storage media
- Layout creation

Details of each work item are as follows:

a. Investigation of outputs from overall viewpoint

Various items, such as objectives and timings, necessary to create reports are investigated from overall viewpoint. Items for the investigation include:
Report titles, use objectives, production cycles and timings, deadlines, distribution addressees, quantities, formats, and positions of output items.

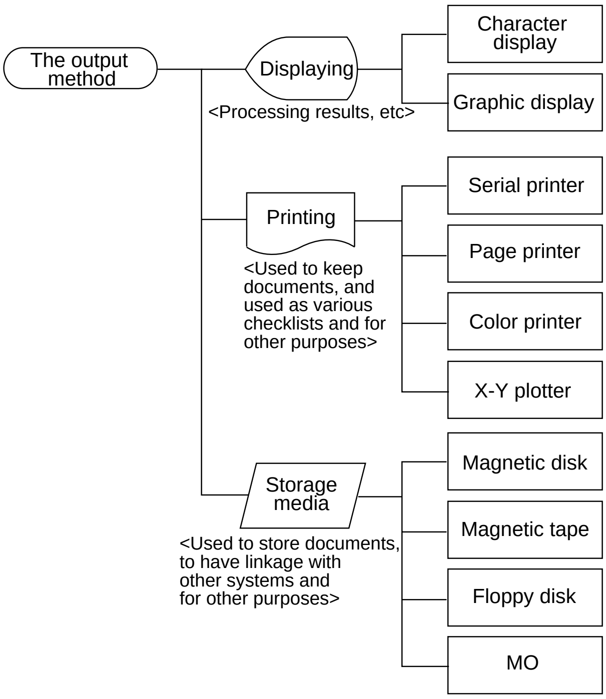
- Report titles
  Titles appropriate to the contents of the reports are given.
- Purposes
  It must be determined whether each report is for external or internal use. If the report is to be submitted externally, there may be designated forms already determined. So, careful attention is necessary.
- Production cycles and timings
  It must be investigated when reports are needed (daily, weekly, monthly, at the timing of business settlements or other occasions).
- Deadlines
  It is investigated when reports must be outputted.
- Distribution addressees
  Destinations of each report must be defined clearly.
- Quantities
  The numbers of pages or screens are estimated for each report.
- Formats
  The number of digits and types (numerals, kanji, etc.) of printed (displayed) items (fields) in reports are determined.
- Positions of output items
  Output views, such as location of items, are clarified.

b. Determination of output methods and storage media

The most suitable output method and medium are determined for each report by considering the nature of the reports and their purposes. There are many cases where reports are recorded onto storage media as data.

Figure 1-2-33

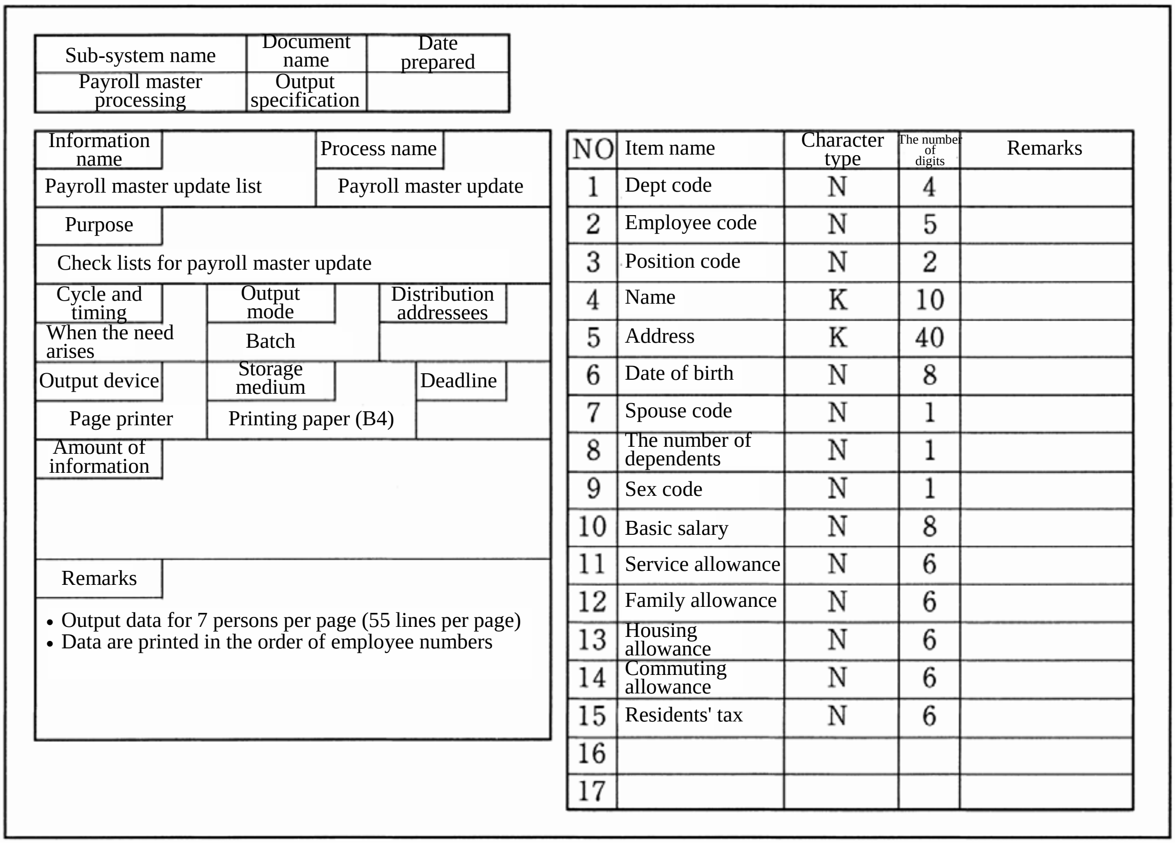
Output methods and storage media


The output method
Displaying
<Processing results, etc>
Character display
Graphic display
Printing
<Used to keep documents, and used as various checklists and for other purposes>
Serial printer
Page printer
Color printer
X-Y plotter
Storage media
<Used to store documents, to have linkage with other systems and for other purposes>
Magnetic disk
Magnetic tape
Floppy disk
MO

c. Creation of output layouts

Output specifications are compiled based on results of items a and b described above, and output layouts (output images) are created based on the specifications (see Figures 1-2-34 and 1-2-35). Designs based on user viewpoints are also necessary for output layouts (output images), such as space between characters, item layouts, inserting year, month and date data automatically.

Figure 1-2-34 An output specification example



Sub-system name	Document name	Date prepared
Payroll master processing	Output specification	

Information name: Payroll master update list

Process name: Payroll master update

Purpose: Check lists for payroll master update

Cycle and timing: When the need arises

Output mode: Batch

Distribution addressees:

Output device: Page printer

Storage medium: Printing paper (B4)

Deadline:

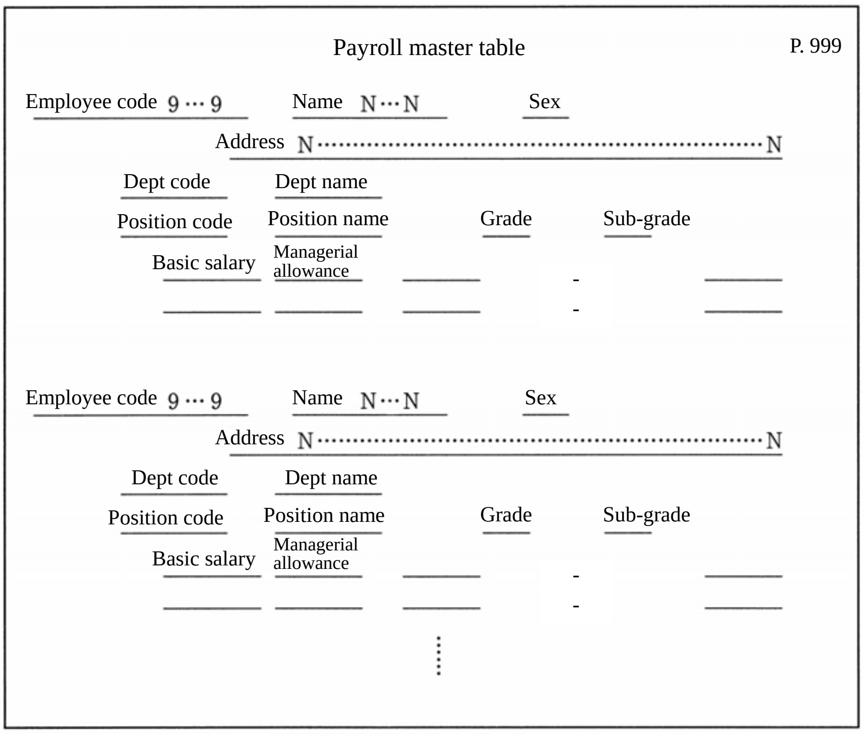
Amount of information:

Remarks:

- Output data for 7 persons per page (55 lines per page)
- Data are printed in the order of employee numbers

NO	Item name	Character type	The number of digits	Remarks
1	Dept code	N	4	
2	Employee code	N	5	
3	Position code	N	2	
4	Name	K	10	
5	Address	K	40	
6	Date of birth	N	8	
7	Spouse code	N	1	
8	The number of dependents	N	1	
9	Sex code	N	1	
10	Basic salary	N	8	
11	Service allowance	N	6	
12	Family allowance	N	6	
13	Housing allowance	N	6	
14	Commuting allowance	N	6	
15	Residents' tax	N	6	
16				
17				

Figure 1-2-35 An output layout (output image) example



(8) Code Design

We find numerous codes, such as prefecture codes, postal codes, vehicle numbers, and product numbers. This is because the proper use of codes facilitate the processing and management of information in computers.
Here, items on coding are described.

① What is code design?

a. Objectives of coding

Coding information makes the identification, classification and arrangement of data easier. However, it is not enough to simply do coding. In code design, it is necessary to include flexibility such as expansion and modification capabilities as well. If a wrong code design is used, all related systems (programs) and data must be modified. Therefore, a cautious attitude must be taken in designing codes.
Input/output codes for data used in external design are specifically called external codes.

b. Meanings of codes

Generally, codes must be provided with the following four kinds of meanings that must be considered in designing codes:

- Identification meanings

Meanings to differentiate a data from others (uniqueness).
For example, customers with the same family and given names can be identified separately by giving each of them a different customer number.

- Classification meanings

Meanings to classify data.
For example, data can be arranged and classified systematically, as by age group or by sex basis.

- Arrangement meanings

  Meanings to determine (rearrange) orders of data.
  For example, data can be analyzed and arranged by birth date order, "Kana (Japanese character)" order or a prefecture order.

- Checking meanings

  Meanings to check whether codes are inputted correctly.
  Ordinarily, an operation such as modulus 11 is applied to each original code and resultant data (check digit data) is added to the lowest digit of the code.

② Points to be considered in designing codes

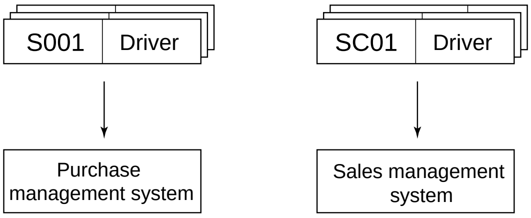
A wrong code design may lead to big problems that affect the entire system. Therefore, code design must be made cautiously by taking into consideration the following items:

a. Coverage areas and use periods of codes

- Coverage areas

  Code systems depend on what areas the codes are used in. For example, when linkage with external systems is required (connected with other company's systems through networks), systems used as standard in the industry must be used. Even when used in the same organization, it is desirable to design codes usable throughout the company.

Figure 1-2-36 A product code example (an example where a purchase management system and the corresponding sales management system use different codes)

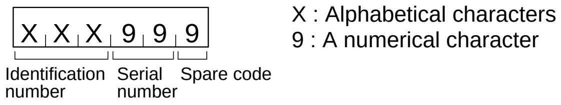


If different codes are used for the same product, unnecessary operations increase.

- Use periods

  If codes are designed based on wrong estimates for code-using periods and/or increases in the amount of work to be handled in the future, code shortage may be caused. If such a case occurs, the code system must be reconsidered, causing the entire system to be modified. A code design requires a sufficient amount of codes.
  To cope with such a situation, spare code for expansion may be provided to the end of each code.

Figure 1-2-37
A code design example with an expansion capability



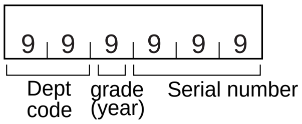
b. Understandability

One important point to be considered in designing codes is understandability.

- "Understandable" means: easy to handle (shorter and simpler)
- Systematic (allowing classification into groups)
- Clear (basically numerical characters are used and alphabetical characters used additionally when needed).

Figure 1-2-38

A systematization example of codes



③ Procedures to design codes and necessary work

Codes are designed in the following procedures:

1. Selection of items to be coded
2. Clarification of coding objectives
3. Estimates of use periods and amount of data
4. Determination of use areas
5. Coding work and creation of code tables
6. Creation of code files

Details of each work are as follows.

a. Selection of items to be coded

Candidate items to be coded are selected through examination and analysis of input/output data and from screen layouts which have been created. Then, items to be coded are determined with a view to business processes and computer processing.

b. Clarification of coding purposes

Code systems to be used depend on coding purposes. Therefore, coding purposes must be clarified first. For example, is the coding required to refer to (or search) data or to classify a large quantity of data?

c. Estimates of use periods and amount of data

The number of data for each item to be coded is estimated with the present volume, and an increase estimate in the future. This estimation is very important. If the number of data increases over the estimate in the system's life, the modification of the entire system, such as modifications of files and programs, becomes necessary. A design containing some allowance in consideration of use period is required.

d. Determination of use (application) areas

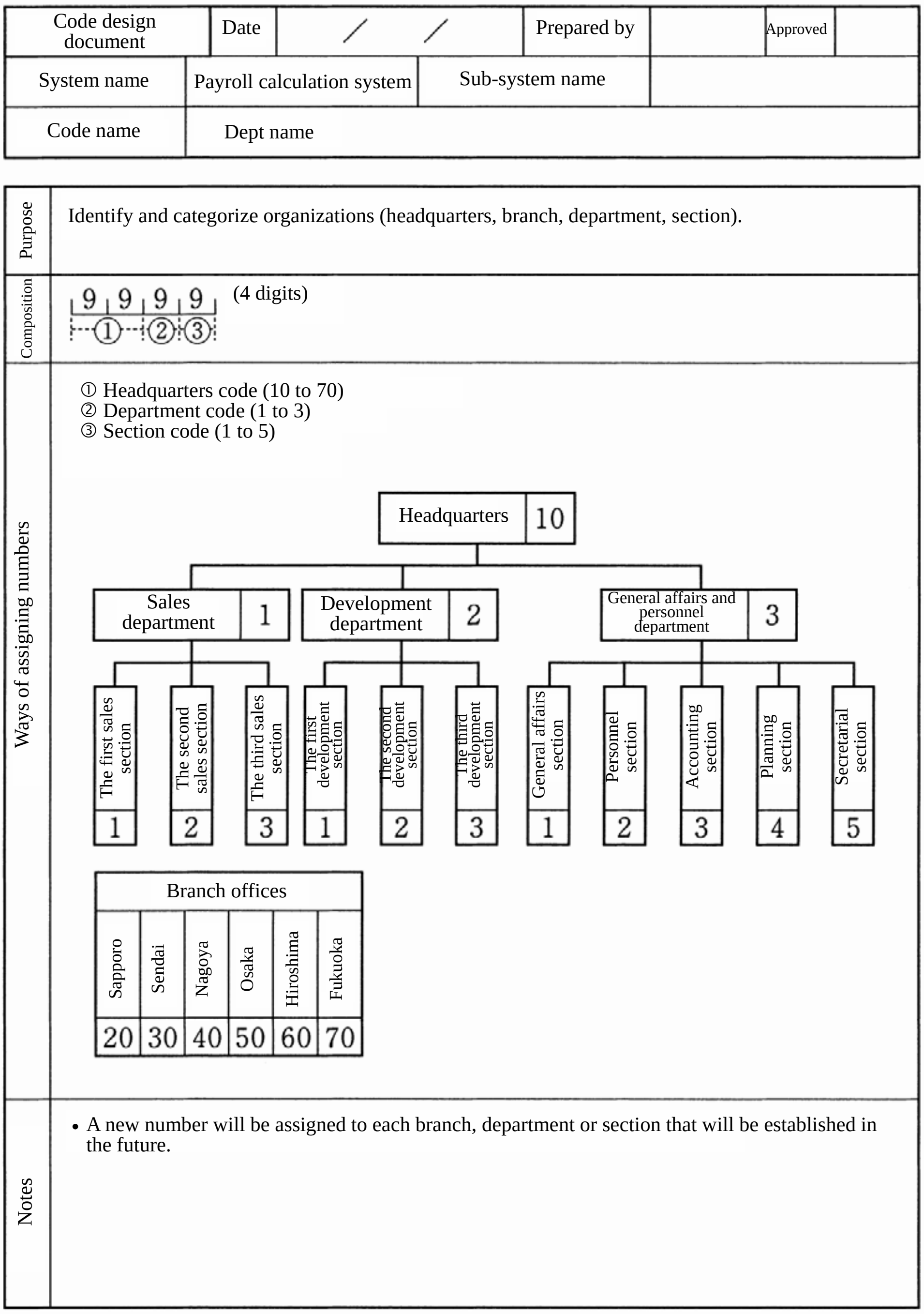
Use areas of codes designed should be determined by specifying the work they are used in and by examining whether they are used in other work or not. This is because, for use in some areas, the number of digits may have to be increased. As described above, it is better to use standard codes like JIS code and JAN code for systems that have a linkage with other company's systems (for example through network connections).

e. Coding work and the generation of code tables

Actual codes should be designed with knowledge about types and characteristics of ordinary codes, based on the results of the items a to d above, and by considering the number of digits (determined in consideration of the number of data) used in the code system and the check digits. Consequently, code design documents are prepared. Then, code tables are drawn up that actually assign codes to data. A document putting together these tables is called "Code book."

Figure 1-2-39 indicates a code design document example.

Figure 1-2-39 A code design document example



Code design document	Date	/ /	Prepared by		Approved	
System name	Payroll calculation system	Sub-system name				
Code name	Dept name					

Purpose

Identify and categorize organizations (headquarters, branch, department, section).

Composition

9 9 9 9 (4 digits)
① ② ③

Ways of assigning numbers

① Headquarters code (10 to 70)
② Department code (1 to 3)
③ Section code (1 to 5)

Headquarters 10
Sales department 1
Development department 2
General affairs and personnel department 3
The first sales section 1
The second sales section 2
The third sales section 3
The first development section 1
The second development section 2
The third development section 3
General affairs section 1
Personnel section 2
Accounting section 3
Planning section 4
Secretarial section 5

Branch offices					
Sapporo	Sendai	Nagoya	Osaka	Hiroshima	Fukuoka
20	30	40	50	60	70

Notes

- A new number will be assigned to each branch, department or section that will be established in the future.

f. Creation of code files

Data on the code tables are stored on storage media (such as magnetic disks) for use in actual work. Thus the created file is called code file.

④ Widely used code examples

Widely used code examples are shown in Figure 1-2-40.

Figure 1-2-40 A code system example

Code name	Descriptions	Application example
Sequence code	A number is assigned sequentially. Though the work is simply done, codes cannot be systematized.	<Prefecture codes in JIS> 01 Hokkaido, 02 Aomori, ---, 47 Okinawa
Block code	A number is assigned to each block. Then, a number is assigned sequentially within a block. Use of this code is convenient for classification, but inconvenient to add data to, or when the number of items is large.	<Codes for universities and higher professional schools in JIS> National universities 0001 - Public universities 1001 - Private universities 2001 - ⋮ ⋮
Decimal code	Objects for coding are first coded with 0 to 9. Then, 0 to 9 codes are given for each of these coded numbers. This process is continued for finer coding.	<Decimal classification method in Japan (book-classifying codes)> 000 General 010-----Libraries 020-----Books, Bibliography ⋮ 100 Philosophy 110-----General on philosophy 120-----Oriental ideologies ⋮ 200 Histories ⋮
By-digit code (Group-classifying code)	Each digit of a code has a meaning assigned. Use of this code is convenient to aggregate data for each classification item, but needs increasingly more figures to handle a larger number of data.	<Industry classification codes in JIS> ① ② ③ ④ 12 | 84 | 1 | 1 The software industry 12 | 84 | 1 | 2 The information processing service industry 12 | 84 | 1 | 3 The information-providing service industry 12 | 84 | 1 | 9 The other information service industry ① The main category code (service industry) ② The medium category code (information service, research and advertisement industries) ③ The minor category code (information service industry) ④ The subdivided category code
Mnemonic code	Abbreviations or symbols of products are used as codes. So, they are easily memorized, but use of them may be inconvenient for classification.	18CTV (18-inch color television receivers) YY. MM. DD (year, month and date)
Check digit code	Indicates codes with check digit to check the code itself. Used for code to which the inclusion of any error cannot be permitted.	<Bank account numbers> 1 2 6 5 ③ ← Check digit ×) 1 4 2 3 1 8 12 15 (1+8+12+15) ÷ 11 = 3 Remainder 3

⑤ Code-checking methods

The check digit method is used to find errors in codes, but does not offer an error-correction capability.

In computer memory, an error correcting code (ECC) is generally used that provides both bit error detection, and error correction capabilities.

Code errors are mostly generated when code data are inputted. For example, errors may occur between 3 and 8 that have a visual similarity or in inputting orders, like inputting 123 instead of 132.
Figure 1-2-41 describes the way of calculating a check digit.

Figure 1-2-41 A way of calculating a check digit

< The calculation of a check digit by use of modulus 10>

① Weighting is applied (the code is assumed to be "1011").

1 0 1 1
5 4 3 2 < Each integer indicates the weight given to the corresponding digits.>
* Smaller weights are given to lower digits.

② Multiplying operations are performed for each digit, and resultant data are accumulated.

$$\begin{array}{cccc} 1 & 0 & 1 & 1 \\ \times & \times & \times & \times \\ \underline{5} & \underline{4} & \underline{3} & \underline{2} \end{array}$$

* If a multiplying operation gives a two-digit result, each digit is accumulated separately.

5+0+3+2 = 10

③ Division (by use of modulus calculation)

$10 \div \underline{10} = 1$ The remainder is 0, which becomes the check digit.
(M: modulus) (Data derived by subtracting the remainder from the modulus data may be used as the check digit).

④ The inclusion of the check digit data

$1\ 0\ 1\ 1\ \underline{0}$
Check digit

(9) Human Interface Design

In designing man-machine systems rationally, the human engineering has previously placed human beings as a subsystem of an entire system. In other words, the human system and the machine system were handled with an equal weight. However, it has recently been seen that human-centered system designs, based on the versatile and flexible natures of human beings, are necessary. This means that it now aims to make man-machine interfaces of systems as natural as possible for human use. Consequently, the term "human interface" has been used in place of "man-machine interface."
To be concrete, the human interfaces are portions where users have contact with systems. GUI, assisting in operations by the use of screens, and document layouts, both described above, are human interface examples. In particular, in input/output operations through a screen of personal computers, as exemplified by Windows, user interfaces with icons and a mouse are widely used. Therefore, systems must now be designed by using these functions effectively. Figure 1-2-42 shows typical human interface factors.

Figure 1-2-42 Typical human interface factors

- Windows
- Icons
  - Push buttons
  - Check boxes
  - Radio buttons
- Menus
  - Popup menus
  - Pull-down menus
  - Drop-down menus
  - Tool bars
  - Scroll bars
- Pointing devices

1.3 Programming Language

1.3.1 Program Attributes

Programs executed on computers have several characteristics. Different specifications should be used depending on the characteristics of programs executed. Therefore, careful attention is required.

(1) Reusable

To execute a program, the program ordinarily must be re-stored to the main memory unit with a loader (this operation is called "reload"). Programs that, once loaded, can be used repeatedly without the re-storing operation are called "reusable programs." In reusable programs, initialization and other related operations are performed prior to the activation of programs to maintain the integrity of processing. Therefore, a program cannot be used simultaneously by more than one process.

(2) Reentrant

Contrary to reusable programs that do not permit simultaneous use, programs that can produce correct results even when used by more than one process simultaneously are called "reentrant programs."

In reentrant programs, the portion for variables and the portion for instructions are provided separately. Then, the portion for variables is provided for each process separately, while the portion for instructions are shared by the process concerned to simultaneous processing.

(3) Recursive

A program that enables itself to be called (executed) while executing itself is called a "recursive program." Being able to call a program by itself does not guarantee that other processes can call it. Therefore, a recursive program is not always a reusable program.

(4) Re-locatable

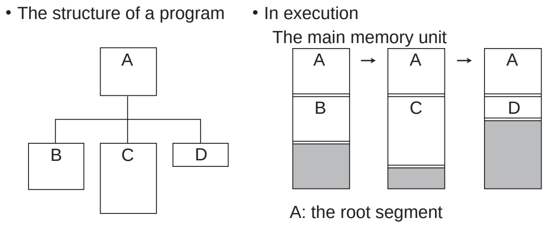
A program to be executed is stored onto the main memory unit with a loader. Programs that can be stored at any location on the main memory space for execution are called "re-locatable programs."

A program whose location in the main memory can be altered even while it is executed is called a "dynamically re-locatable program."

(5) Overlay

"Overlay" is a method used to execute a program that occupies memory larger than the main memory capacity. In this method, a program is divided into a number of segments, each of which is not executed simultaneously. Then, in execution, the root segment loads one of these exclusive segments alternately from a secondary memory device (see Figure 1-3-1).

Figure 1-3-1 The overlay method



A: the root segment
B to D: Exclusive segments

1.3.2 Data Type

The data types are the fundamental elements of the data structures processed by software, and can largely be classified into the basic data type and the structure type. Use of a type is declared with a specification statement of a programming language.

(1) The Basic Data Type

The basic data type is defined as the set of data types for single data items, such as integer type, real number data type, logical type and character type.

- Integer type : The set of integers. 1 2 3
- Logical type : "True" or "false." 1 (or 0), T (or F)
- Character type : The set of characters. A B C , a) b) c) d) e)
- Pointer : An index to point a data. 109 → 109 Data

← The address to store a data

(2) Structure Type

A structure type is derived by combining basic data types.

- Array type: indicates an array of a finite number of data items which have the same type and the same size.

1	2	3	4	5	6	7	8
12	101	1	45	32	128	64	512

- Record type: indicates an aggregate of a finite number of different data items each has a pre-determined same size and type.

Number	Name	Age	Birthplace	Hobby
6873	Sakaya Senta	45	Hokkaido	Chanting of Chinese poems
7245	Yuuki Shinobu	31	Tokyo	Travel

(3) Abstract Data Type

The abstract data type is defined for entities which contain data and operations on the data. The operations are functions or procedures. In using an abstract data type, only a reference to the type name and calling of an operation are permitted.

Operations	Data
push down	Stack pointer value
pop up	

1.3.3 Control Structure

The control structure of a program, or an algorithm, is designed based on the structure theorem, because designs using the theorem are effective in developing programs with the following characteristics resulting in increased productivity and software quality.

- Clear logical structures
- Accurate
- Easy-to-read
- Easy-to-maintain

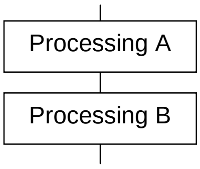
"Structure theorem" is defined as follows:
"For programs (called proper programs) designed to be composed of structural units each of which has an entry and an exit, any logic can be described by combining the three basic structural units."
Details are described below:

(1) Basic Structure

① Sequential structure (sequential type)

In the sequential structure, functions (instructions in programs) are executed sequentially in a one-way direction.

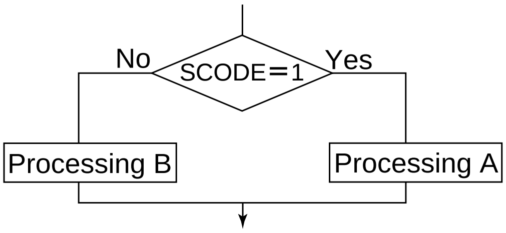
Figure 1-3-2 A sequential structure example



② Selective structure (If-then-else type)

In the selective structure, either of two functions is selected depending on whether a condition is met or not.

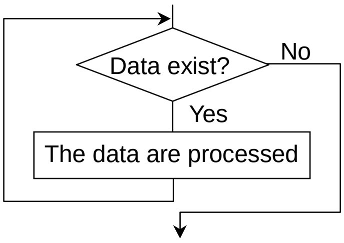
Figure-1-3-3
The selective structure



③ Repetitive structure (Do-While type)

In the repetitive structure (Do-While type), if a condition is found "true," the same function is executed repeatedly. If the condition is found "false," the processing goes out of the loop.

Figure-1-3-4
The repetitive structure (Do-While type)



If any algorithm can be described with the three structures above, use of GO TO sentences, which is a biggest cause for making programs complex, should be unnecessary. This point is described later.

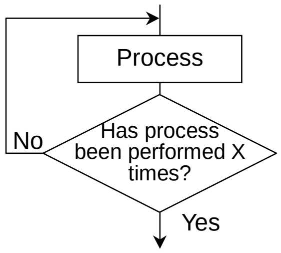
(2) Additional Structures

In addition to the three structures above, the following two structures are used to make programs more easily understood.

① Repeat-Until type

In the Repeat-Until type, a condition is checked after executing a function. Then, if the condition is found "false," the execution of the function is repeated, while, if "true," the operation goes out of the loop.

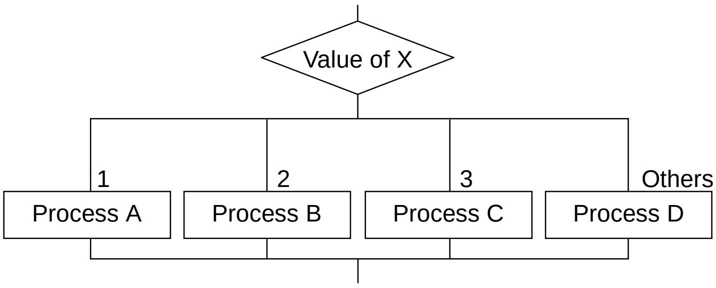
Figure-1-3-5
The repetitive structure (Repeat-Until type)



② Multi-branch structure (CASE)

The multi-branch structure is used when one of many functions is performed depending on the result of checking a condition.

Figure-1-3-6
The multi-branch structure (CASE)



(3) GO TO-less Programming

Use of the GO TO statements is the largest factor in making programs complex. Use of the GO TO statements seems convenient. So, they are often used carelessly. However, if the statements are abused, the structure theorem no longer holds.
Therefore, the use of the GO TO statements should be avoided if possible (this does not means that the use of them is prohibited), and algorithms should be described by combining the basic structures.

(4) Procedure and Function

Units which compose a program are procedures and functions. There are procedures and functions provided by language processing systems, and also those created by users. Both are called for execution by programs. With a procedure, processed results are returned through arguments, while, with a function, results are returned as values of the function.

1.3.4 Syntactic Analysis

Syntactic rules to describe programs are provided for a programming language. Based on the rules, the compiler analyzes source programs created by users and performs the translation. Analyzing structures of a program based on a language's syntactic rules is called syntactic analysis. The language we ordinarily use in human communications is called a natural language, while the programming language is an artificial language.

(1) Formal Language

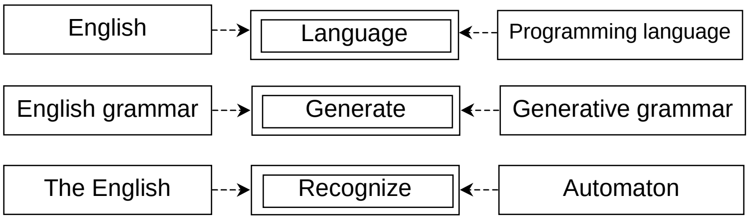
In natural language, words and sentences have a number of different meanings. In addition, some degree of freedom exists in sentence structures. Therefore, it is difficult to mechanically analyze natural sentences. It is difficult for the computer to do interpretation with due consideration to indirect factors such as the atmosphere and context of conversation, human feelings, surrounding environments, etc. Meaning of the sentence, "watashiwa torida (I take a piece of bird meat)," said in front of a meat shop, is different from the meaning of the same sentence said in response to a question about the zodiac sign of your birth year. In the latter case, the sentence means "The zodiac sign of my birth year is the Cock." With the present-day computer, it is still difficult to determine meaning. A formal language is designed so that no ambiguity in interpretation is left.

① Definition of a formal language

A formal language is defined with a generative grammar and automatons. The generative grammar is the generic name of the grammars proposed by Chomsky. Put simply, it means rules to generate languages. On the other hand, the automaton can recognize only formal languages.

Figure-1-3-7

Definition of a language



a. Types and grammars of formal languages

Chomsky proposed the following four formal languages. An automaton corresponds to each of the four languages.

Figure-1-3-8

Types of formal languages

Types of languages		Accepting automaton
Phrase structure grammar (0-type language)		Turing machine
Context-dependent language (1-type language)		Linear finite automaton
Context-free language (2-type language)	BNF	Push-down automaton
Regular grammar (3-type language)	Theoretically simplest	Finite automaton

b. Operations in languages

The phrase structure grammar is described with four elements of N, T, P and S. Four example, real numbers are generated as follows.

```
N: non-terminal symbol = { <numeral>, <numeral sequence>, <integer>, <exponent>, <real
   number>}
T: Terminal symbol     = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, E,. }
P: Generative rules    = { <numeral >    → 0
                           <numeral>     → 1
                           <numeral>   → 2                                        ⑦
                           <numeral>   → 3                                        ⑤
                           <numeral>   → 4                                        ③
                           <numeral>   → 5
                           <numeral>   → 6
                           <numeral>    →7
                           <numeral>   → 8
                           <numeral>   → 9
                           <numeral sequence> → <numeral>                         ④
                           <numeral sequence> → <numeral sequence> <numeral>      ②
                           <integer> → <numeral sequence>                         ⑥
                           <exponent> → <integer>
                           <real number> → <integer>. <numeral sequence>          ①
                           <real number> → <integer>. <numeral sequence> E <exponent>
                           <real number> → <integer> E <exponent>
                           <numeral>   → 0
                           }

S: start symbol        = { <real number>}
```

```
Example    Generation of a real number 1.23 <real number> → <integer>. <numeral sequence>   ; from
①
                        → <integer>. <numeral sequence> <numeral>     ; from ②
                        → <integer>. <numeral sequence> 3             ; from ③
                        → <integer>. <numeral> 3                      ; from ④
                        → <integer>. 2 3                              ; from ⑤
                        → <numeral sequence>. 2 3                     ; from ⑥
                        → <numeral>. 2 3                              ; from ④
                        → 1. 2 3                                      ; from ⑦
```

The non-terminal symbols appear only during the generation (rewriting), while terminal symbols appear at the end.

(2) Regular Expression

The regular expression is a way of describing regular languages. When use of only character sequences is insufficient, the expression is effective to describe the pattern compactly.

```
N: non-terminal symbol = { <N2>, <N3>, <N4>, <N5>, <real number>}
T: Terminal symbol      = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, E,. }
P: Generative rules     = { <N5>   → { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }               ④
                            <N5>   → <N5>  { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }
                                          ; " | " indicates a delimiter for the right side data
                            <N4>   → <N5>.
                            <N4>   → <N4>  { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }
                            <N3>   → <N4>  { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }
                            <N2>   → <N5>.                                                  ③
                            <N2>   → <N5>  E
                            <N2>   → <N3>  E
                            <N2>   → <N2>       { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }   ②
                            < real number > → <N2> { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }  ①
                            }

S: start symbol         =   { <real number> }
```

```
Example   Generation of a real number 1.23 <real number>  → <N2>  3       ; from ①
                                                         → <N2> 23       ; from ②
                                                         → <N5>.23       ; from ③
                                                         → 1.23          ; from ④
```

(3) BNF (Backus Naur Form)

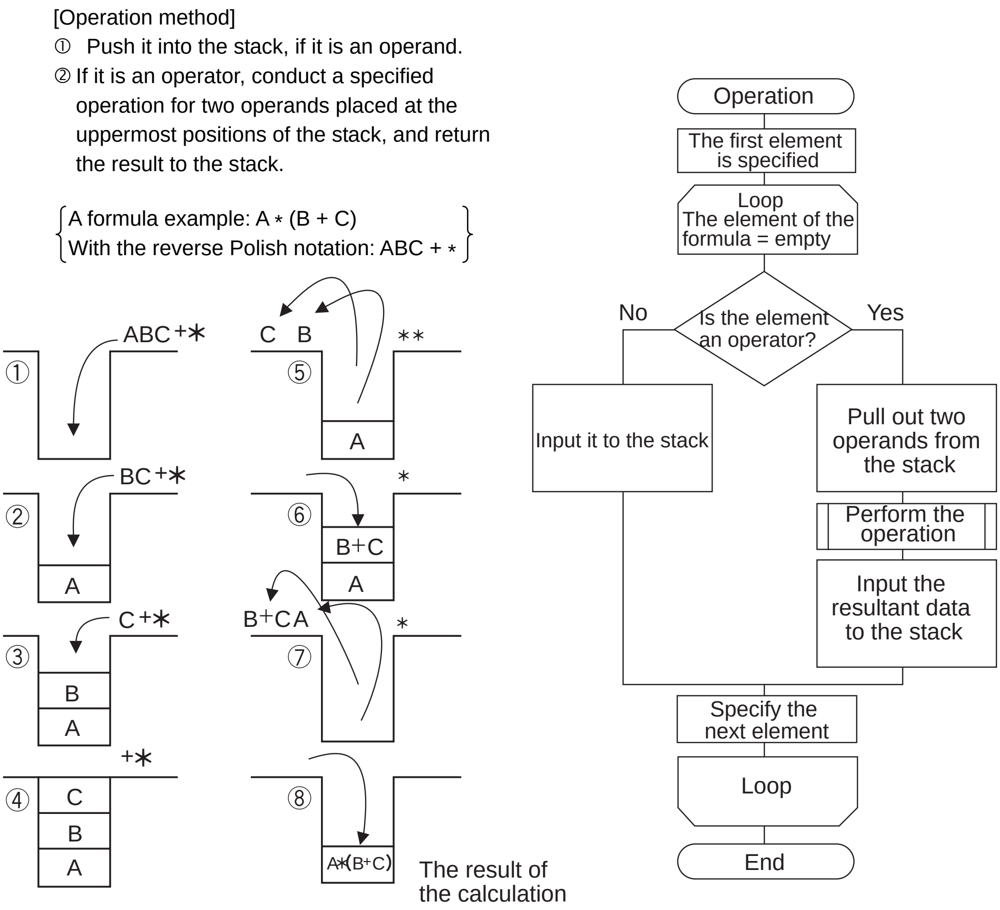
BNF, also called Backus Naur Form, offers a way of description to define formats of a context-free grammar. Real numbers are described as follows.

<numeral> :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 8 | 9 ; Here, " | " indicates「or」.
<numeral sequence> :: = <numeral> | <numeral sequence> <numeral>
<integer> :: = <numeral>
<exponent> :: = <integer>
<real number> :: = <integer>. <numeral sequence> | <integer>. <numeral sequence> E <exponent> | <integer>
E <exponent>

(4) Polish Notation

The Polish notation, a way of expressing formulas, places an operator before an operand. This notation is also called prefix notation. For example, "1 + 2" is expressed as "+ 12." On the other hand, the notation ("1 + 2") usually used is called infix notation. In compiler, postfix notation ("12 +"), also called reverse Polish notation, is used (Figure 1-3-9). With this notation, an operator is placed after an operand. In compiler, reverse Polish notation is most used, because, with the notation, operands and operators, constituting components of a formula, are placed in the order of operations, consequently making the operations effective. Therefore, postfix notation is sometimes called simply Polish notation.

Figure 1-3-9 An example of operations using the reverse Polish notation



[Operation method]
① Push it into the stack, if it is an operand.
② If it is an operator, conduct a specified operation for two operands placed at the uppermost positions of the stack, and return the result to the stack.

A formula example: A ∗ (B + C)
With the reverse Polish notation: ABC + ∗

①
ABC+∗
②
BC+∗
A
③
C+∗
B
A
④
+∗
C
B
A
⑤
C B
∗∗
A
⑥
∗
B+C
A
⑦
B+CA
∗
⑧
A∗(B+C)
The result of the calculation
Operation
The first element is specified
Loop
The element of the formula = empty
Is the element an operator?
No
Yes
Input it to the stack
Pull out two operands from the stack
Perform the operation
Input the resultant data to the stack
Specify the next element
Loop
End

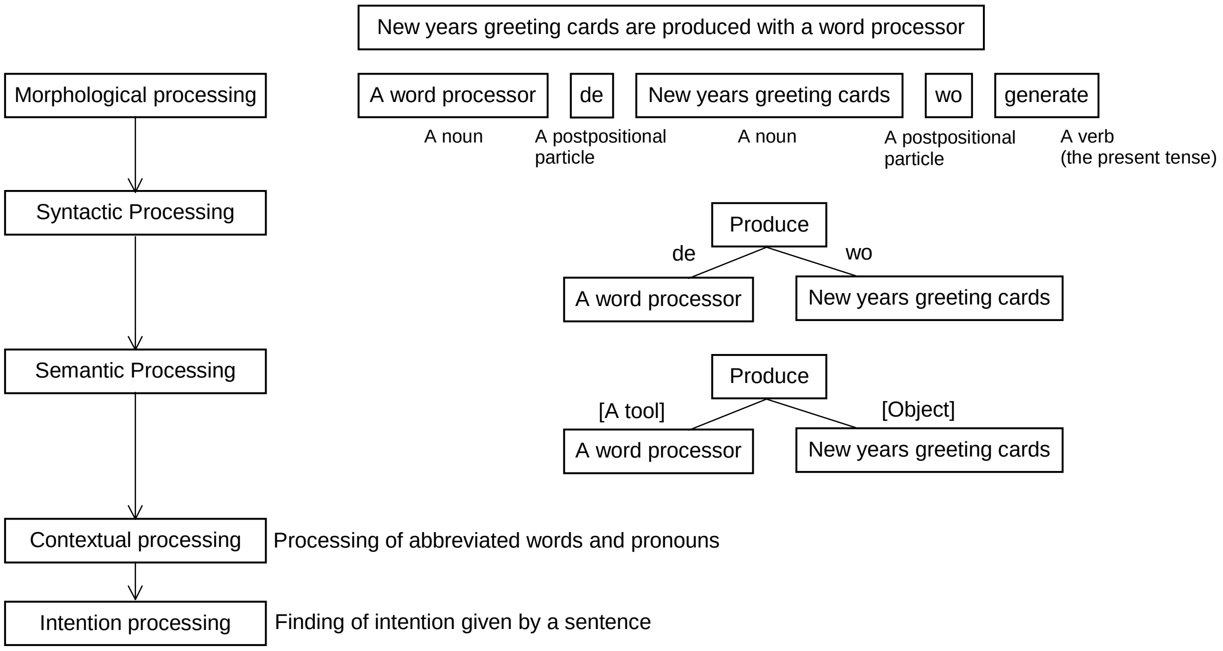
(5) Natural Language Processing

The natural language includes many ambiguous expressions. Therefore, today only a part of it is processed with the computer. The processing of natural language is still mostly at a research stage. In natural language processing, the analysis of "meaning," "intention" and "context" is required in addition to the processing of syntax and words and phrases. So, the processing is more complex than that of the formal languages. In particular, the use of knowledge bases becomes essential to handle the ambiguity.
A rough flow of the natural language processing is as follows (see Figure 1-3-10).

1. Morphological processing : The processing to find words (morphemes) from a given sentence, and determine their parts of speech.
2. Syntactic processing : Processing to apply syntactic rules and to find the syntactic structure.
3. Semantic processing : Processing to find the semantic structure of a sentence by using dictionaries and knowledge bases.
4. Context processing : Processing to interpret relationships between a sentence and other sentences and pronouns such as "it," "that" etc. and to analyze abbreviated words and cause-effect relationships between sentences.
5. Intention processing : Processing to find the intention of a sentence by also taking into consideration situations, habits and cultural backgrounds.

The conversion of a Japanese sentence inputted with simple "Kana" characters or Roman letters, into an equivalent sentence including Chinese characters, is a typical example of the natural language processing.

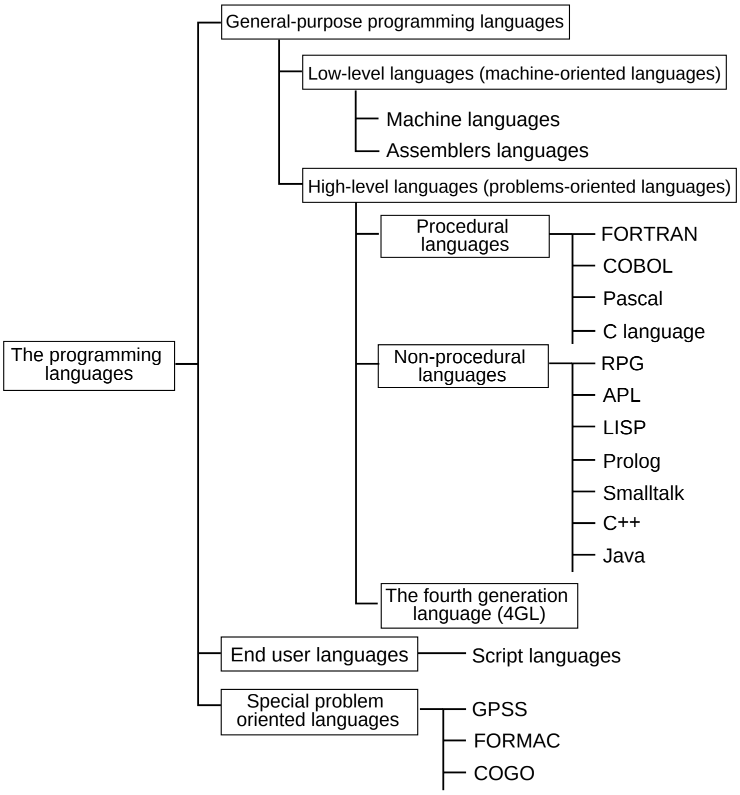
Figure 1-3-10 An example of natural language processing



1.3.5 Classification of Programming Languages

Figure 1-3-11 is a rough classification of programming languages.

Figure 1-3-11 Classification of programming languages


General-purpose programming languages
Low-level languages (machine-oriented languages)
Machine languages
Assemblers languages
High-level languages (problems-oriented languages)
Procedural languages
FORTRAN
COBOL
Pascal
C language
The programming languages
Non-procedural languages
RPG
APL
LISP
Prolog
Smalltalk
C++
Java
The fourth generation language (4GL)
End user languages
Script languages
Special problem oriented languages
GPSS
FORMAC
COGO

(1) Low-Level Language

Machine languages, and programming languages that have one-to-one correspondence between their instructions and those of corresponding machine languages, are called low-level language. Assembler languages belong to the low-level language. Programs written with an assembler language have the following characteristics:

- Offers high program execution speeds.
- Offers lower software productivity, compared to the high level language, because it uses alphabetical symbols to describe programs.
- Programmers have to have knowledge about hardware, because the language has a close relationship with hardware functions.
- Programs have to be rewritten when hardware is modified.

With such characteristics provided, assembler languages are used only in control processing and other areas where high-speed processing is required. High-level languages are mostly used in software development.

(2) High-Level Language

The high-level language has been developed to solve problems the low-level languages pose. It is also called the problem-oriented language.

The high-level language has the following characteristics:

- It offers ease in describing processing procedures and is suited for solving problems.
- Compared with the assembler, it has structures close to natural languages.
- It is less dependent on special hardware (suited for use in a variety of applications).
- An instruction can include more than one machine language instruction. So, the number of program steps (the number of instructions) can be shortened.

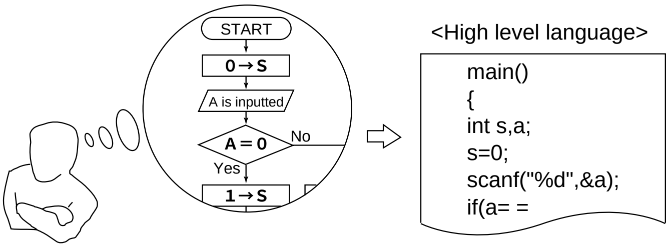
Use of a high-level language reduces program creation time, makes program modification and the addition of functions easier, consequently leading to an increase in software development productivity. However, the high-level language needs software to directly operate hardware functions, sometimes leading to a decrease in processing efficiency.

General-purpose high-level languages currently being widely used are described in detail in the next section.

(3) Procedural Language

To describe algorithms that indicate processing procedures to solve a problem, instructions using a procedural language are utilized. Many of the high-level languages are procedural languages.

Figure 1-3-12
Procedural Language



```
main()
{
int s,a;
s=0;
scanf("%d",&a);
if(a= =
```

(4) Non-procedural Language

The non-procedural language allows programs to be generated without describing algorithms. With the language, a program is generated by providing input, output and processing conditions in order to solve a problem, and by selecting necessary processing routines that are provided in a language in advance.

The non-procedural language has the following characteristics:

- Without using any algorithms, the language allows persons with inadequate programming technique to create programs.
- It offers high productivity due to applying pre-determined patterns to processing procedures, but less flexibility in generating programs.

The non-procedural language is classified into the following languages depending on description formats used and other factors:

- Tabular form language
  Writing processing requirements in predetermined formats with a tabular form language automatically generates corresponding programs.
- Functional language
  With a functional language, the combining of functions generates programs. LISP for list-processing and APL for scientific and engineering computation are functional languages.
- Logical language
  With the logical language, programs are generated from logical expressions based on predicate logic. An example of this type of languages is Prolog, a programming language for artificial intelligence.
- Object-oriented language
  With an object-oriented language, programs are created by considering things handled by programs as objects. Typical object-oriented languages include Smalltalk, C++, and Java suited for use on networks.

(5) Fourth Generation Language (4 GL)

Users often complain, "Requested functions are not provided.", when the development of software for systematizing operations is at last completed.

Main reasons behind this situation include:

- System developers do not understand sufficiently the operations to be systematized.
- The client did not correctly provide contents of the operations to be systematized to system developers creation.

The creation of programs by persons in the user department, who are versed in the operations, is one solution to the problem. However, the handling of a high-level language to generate programs requires a considerable amount of technical education.

To cope with such a situation, a programming language, allowing persons not having knowledge and experience about programs to create programs easily, has been developed. This language is called the fourth generation language.

The fourth generation language is the generation succeeding the high-level language (the third generation language). With the former high-level language, procedures to solve problems in application operations are described with instructions, while, with the fourth generation language, the processing of operations is done by instruction with parameters. In other words, the new language intends to generate processing procedures to solve problems automatically as much as possible, reducing the number of instructions to generate a program.

(6) Script Language

With the continuing wide spread use of computers, the move for "end-user computing," in which a user department develops systems by itself has also grown. The script language is a programming language used by end users to develop such programs.

The event-driven nature is the most important characteristic of the script language. The program waits for data to be inputted by a user with a keyboard or mouse. On detecting an input, the program starts processing.

Today, advanced development environments using GUI have been provided for the language. It is also used as a tool to customize application software. HTML (Hyper Text Markup Language), a text-editing language to create files to be displayed on a WWW (World Wide Web) browser on the Internet, has been used widely.

(7) Specific Problem Oriented Language

Today, we develop computer programs with programming languages, and use computers to solve problems in a variety of areas. However, some processing in some areas is so complex and advanced that it takes from several hours to several tens of hours to complete the necessary calculations. The specific problem oriented language has been developed targeting processing only for operations used in these special areas.

The specific problem oriented language has the following characteristics:

- Since the areas programs are used in are limited, it allows necessary programs to be developed more quickly and precisely within the limited areas, compared with the use of general-purpose languages.
- Ordinarily, software makers supply such languages as software packages. Therefore, the use of them is advantageous to users in reliability and development cost.
- Specialized knowledge in the areas where the language is used is necessary.
- The language is not used for areas other than those it has been developed for.

The following are major languages in the category of specific problem oriented language:

- GPSS (General Purpose Simulation System)
  A language used for simulating a phenomenon on a computer is called a simulation language. GPSS is a simulation language developed by IBM and is widely used for discrete simulation purposes.
- FORMAC (FORmula MAnipulation Compiler)
  This is a programming language for performing complex and advanced formula manipulation mostly in hydrodynamics and astronomy. In the language, advanced formula manipulation functions, such as the handling of differential operation and the transformation of formulas, are added to FORTRAN, a programming language for scientific and technical computation.
- COGO (COordinate GeOmetry)
  This is a programming language to analyze and design the structures of buildings.

1.3.6 Types and Characteristics of Programming Languages

Various high-level languages have been developed. The number of them exceeds several hundred. The upgrading of programmers is easier for high-level languages, compared with low-level languages. Therefore, the language is used in many areas.

(1) FORTRAN (Formula Translator)

FORTRAN was designed in 1954 by John Backus (an American) and others. Then, it was actually developed, based on the design documents, by IBM and others with IBM playing the principal role. FORTRAN was announced as the world's first compiler in 1957. In the days when FORTRAN was developed, computers were mostly used for scientific and engineering computation, aimed at processing complex computations at high speed. Therefore, FORTRAN has established its position as a programming language for scientific and engineering computation.

<Characteristics>

- It enables the expression of numerical problems and algorithms like logic operations, both of which are required for scientific and engineering computation.
- Groups of functions, such as trigonometric functions, exponential functions and logarithmic functions, are also provided.
- The structure of the language is simple.
- It employs a non-recursive and static structure.

Figure 1-3-13

A program example with FORTRAN

```
C      SAMPLE PROGRAM
       INTEGER I,J,K
   10  CONTINUE
       READ(5,100)I,J,K
  100  FORMAT(3I5)
       IF(I.EQ.0) GO TO 20
       L = I ∗ J − K
       WRITE(6,200) L
  200  FORMAT(1H,I8)
       GO TO 10
   20  STOP
       END
```

(2) COBOL (COmmon Business Oriented Language)

The development of COBOL was started in 1959 as a compiler for business processing by CODASYL (COnference on DAta SYstems Language), an organization of computer manufacturers and computer users, with the Department of Defense of the United States at the center of the research and development efforts. The specifications were announced in 1960, and the ANSI (American National Standards Institute) specifications were established in 1968. In Japan, JIS COBOL was established in 1972. It has been widely used as a compiler for business processing using general-purpose computers.

<Characteristics>

- The use of expressions close to natural English sentences offers high document readability.
- It allows structures of files used for inputting/outputting to be defined rigorously.
- Supporting various file organization methods enables efficient file-inputting/outputting operations.
- Many business processing functions, such as classifying and merging functions and report-generating functions, are provided.

Figure 1-3-14

A program example using COBOL

```
IDENTIFICATION DIVISION.          } Identification division
PROGRAM-ID.  SAMPLE.
ENVIRONMENT DIVISION.    ·········Environmental division
DATA DIVISION.
WORKING-STORAGE SECTION.
01  INPUT-AREA.
    03 I              PIC 9(3).
    03 FILLER         PIC X(1).
    03 J              PIC 9(3).        Data division
    03 FILLER         PIC X(1).
    03 K              PIC 9(3).
    03 FILLER         PIC X(69).
01  INSATSU.
    03 L            PIC······,···9.
PROCEDURE DIVISION.
HAJIME.                                Procedure division
      .
      .
```

(3) BASIC (Beginner's All purpose Symbolic Instruction Code)

BASIC was developed in 1965 as a programming language for student education by Dartmouth College of the United States. When personal computers came into general use BASIC was adopted as a high-level language allowing ease of use.

<Characteristics>

- It employs an interpretive method in which the translation of a program, and the execution of the program are done simultaneously.
- The use of an interpretive method makes the execution speed slower compared to the use of a compiler language.
- A line number is placed in front of each instruction line.
- The use of descriptions close to English words makes the learning of the language easy.
- Default values are automatically inserted, even when descriptions for declarations of variable types or output formats are omitted.

Figure 1-3-15

A program example using BASIC

```
10  INPUT I,J,K
20  IF I = 0 THEN END
30  L = I ＊J－K
40  LPRINT L
50  GOTO 10
```

(4) The C Programming Language

The C programming language is a compiler language for system description developed in 1972 by W. Kernighan and D. Ritchie at Bell Laboratories of AT&T as the language for the development of an operating system called UNIX.

Initially, the language was used exclusively for UNIX on the PDP-11, mini-computers of DEC. Because of its usefulness, however, C language today is widely used on many computers including workstations and personal computers. The C language based on the manual, "The C Programming Language," by Kernighan and Ritchie is called K&R C language. On the other hand, there is ANSI, established in 1989 as a standard specification of the language that includes improved and expanded functions as well. A C language complying with that specification is called "ANSI-C."

<Characteristics>

- It allows simple expressions.
- A variety of operators, data structures and control structures are provided.
- It allows bit-operations close to those of the low-level language.
- Few constraints are imposed on program formats, or a format-free method is employed.
- High portability is provided.

The C++ language, a descendent of C which includes additional object-oriented functions, is widely used.

Figure 1-3-16

A C language program example

```
#include<stdio.h>
main()
{
    int x,y;
    printf(“input Number = ”);
    scanf(“%d”,&x);
    y = x % 2 ;
    if(y == 0)
            printf(“Even number¥n”);
    else
            printf(“Odd number¥n”);
}
```

(5) Other High-Level Languages

In addition to FORTRAN, COBOL, BASIC and the C language, various programming languages have been developed for a variety of objectives. Characteristics of major programming languages are described below:

① ALGOL (ALGOrithmic language)

ALGOL is a programming language developed for scientific and engineering computations.

<Characteristics>

- Programs are described with syntactic rules called the Backus notation.
- Recursive calling is permitted.
- The language is designed based on the structured programming concept.

Though the language has the superior characteristics described above, ALGOL is hardly used because FORTRAN serves the same purpose.

② Pascal

Pascal is a programming language for programming education and it includes ALGOL's characteristics.

<Characteristics>

- It provides control structures based on structured programming.
- Programs themselves are easily readable, because the language was developed for programming education.
- A variety of data structures are provided.

③ Ada

Ada, a programming language developed by the U.S. Department of Defense, has the following characteristics:

<Characteristics>

- The language is based on Pascal with various improvements added to it.
- A high reliability provides for the development of large-scale software.
- Easily maintained and good execution efficiency are provided.

④ LISP (LISt Processor)

LISP, a programming language, has the following characteristics:

<Characteristics>

- It is a non-procedural programming language.
- Both data and programs have list structures.
- Programs are described by combining standard functions provided in the system and those defined by users.

⑤ Prolog (Programming in logic)

Prolog is a language for the artificial intelligence research and development.

<Characteristics>

- It is a predicate logic language.
- Inference functions are provided.

⑥ APL (A Programming Language)

APP, developed by IBM, has the following characteristics:

<Characteristics>

- It provides notations based on ordinary mathematical notations.
- Arithmetic operations and logic operations can be expanded to vectors, matrices and tree structures.
- Programs can be described by combining special characters and symbols.

⑦ PL/I (Programming Language/One)

PL/I, jointly developed by IBM and its user organization, is for both scientific and engineering computation and business processing.

<Characteristics>

- It uses a nesting structure with the block used for the program unit.
- It allows bit operations and list operations for data.

(6) Object-oriented Languages and Other Programming Languages

① Visual Basic

Visual Basic, a typical visual language, is BASIC that can run in Windows environments. User interface is important for interactive programs. Visual Basic offers an environment to develop applications that provide ways of interaction suitable for Windows' graphical environments.

<Characteristics>

- Visual Basic, a product from Microsoft, is closely related with Windows.
- Programs generated with Visual Basic can be run directly on Windows.

② C++

C++, developed by B. Straustrap and others at Bell Laboratories, was produced by adding object-oriented concepts to the C language.

<Characteristics>

- It offers complete compatibility with the C language.
- It includes concepts of "class," "inheritance," and "virtual function," used in object-oriented programming.

③ Java

Java is an interpreter-type object-oriented language based on the C and C++ languages. Running Java on a WWW browser on the Internet enables interactive operations.

<Characteristics>

- The language can be run on any platform.
- Enhanced security and networking functions are provided.
- Dynamic functions are provided.

④ Perl (Practical Extraction and Report Language)

Perl, an interpreter language developed by Larry Wall for text processing, is used on UNIX-family operating systems and on Windows. It has been strongly influenced by the C language and shell scripts.

<Characteristics>

- It is used as a standard language for CGI (Common Gateway Interface) programs run on WWW servers.
- It is an interpretative script language.

⑤ SGML (Standard Generalized Markup Language)

SGML is a language to describe logic structures and semantic structures of documents with simple tags (marks).

<Characteristics>

- It was developed as a language to allow all electronic documents to be handled by computers. However, the ensuing complexity has prevented it from being widely used.

⑥ HTML (Hyper Text Markup Language)

HTML is a language to describe hypertext used on the WWW and other applications.

<Characteristics>

- It is an expanded version of SGML.
- The use of tags enables easy specification of image files or links.

⑦ XML (eXtensible Markup Language)

XML, positioned as the language to succeed HTML, is a page description language in the process of being standardized. XML is a language combining the strong points of both HTM and SGML where linking functions of HTML are expanded and SGML is optimized for Internet use. The standardization work by the WWW consortium for XML was completed in Dec. 1997.

<Characteristics>

- Compared to HTML, it provides higher levels of freedom and flexibility to users.
- Applications are expected to include data exchange in electronic commerce, enterprise document storage, and digital BS broadcasting.

⑧ PostScript

PostScript is a page description language developed by Adobe Systems. PostScript is used as the de facto standard in desktop publishing.

<Characteristics>

- Descriptions are independent of performances of output devices.
- Embedding an interpreter of PostScript in a printer enables the printing of various types of fonts at high speeds.
- When software that interprets PostScript is used, an ordinary printer can produce the same quality as a PostScript-embedded printer.

1.4 Programming Techniques

1.4.1 Procedural Programming

The programming technique which uses "procedures" to describe solutions is called procedural programming. Procedural programming was the first programming technique used in high level languages. It is the conventional programming technique most used today.

Languages supporting procedural programming include FORTRAN, COBOL, PL/1, Pascal, ALGOL, BASIC and the C language. In these languages, a syntax is provided for each instruction executed by a computer. In other words, a procedural program is considered an aggregate of sentences. It can also be said that, "the program is composed of data structures and algorithms." This is most useful for the stored program method, a characteristic of the von Neumann-type computer.

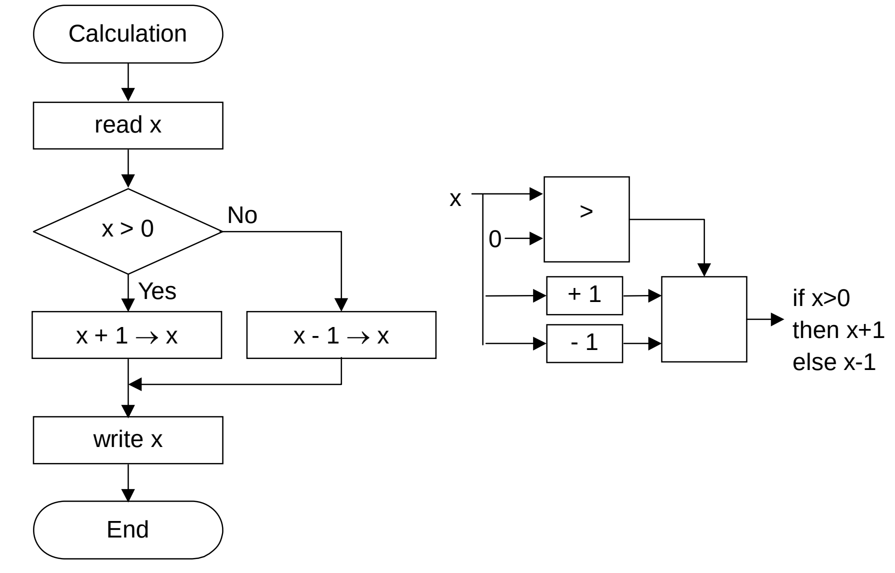
The efficiency and maintainability of procedural programming is enhanced with the introduction of structured programming and structured carts.

1.4.2 Functional Programming

A programming technique that uses functions to describe solutions is called functional programming. LISP, developed to make the handling of characters and symbols easier, is a typical language supporting functional programming. LISP was originally developed for use in research on artificial intelligence.

With procedural programming, values in data structures in the main memory are varied when a program is executed, indicating that dynamic side effects take place. With functional programming, however, functions process input data and output function values. So, it involves no side effects. The rewriting of values in data structures is not permitted. It is also called declarative programming, since programs are described declaratively. Control flows are described with procedural programming, while data flows are described with functional programming, and therein likes the difference between the two programming methods (see Figure 1-4-1).

Figure 1-4-1 Procedural programming and functional programming



The procedural method
(Control flows)

The functional method
(Data flows)

1.4.3 Logic Programming

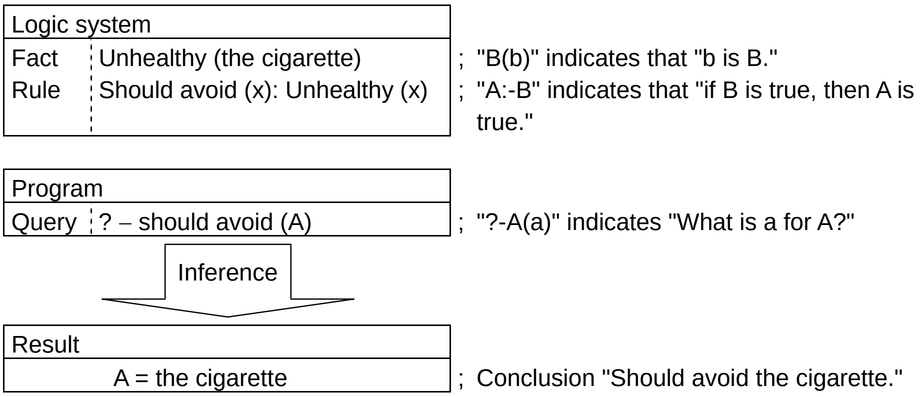
The programming method, which describes solutions with "an aggregate of declarations" is called logic programming. Logic programming is composed of logic expressions and inference mechanisms of the base-constituting logic system, and involves no side effects. Prolog is a typical programming language supporting logic programming.

With logic programming, relationships inherent in problems and constraints are described declaratively. Input-output relationships are described with logic expressions. The result of calculation (inference) is either "successful" or "unsuccessful." Inputting a logic expression and consequently proving that the expression coincides with the conclusion of a given logic expression as a program is equivalent to the execution (calculation) of a program.

Figure 1-4-2 A logic programming example

An inference example with Prolog

"Should avoid" and "Unhealthy" are both predicates. "X" indicates a variable. "The cigarette" indicates a constant.



Logic system

Fact Unhealthy (the cigarette) ; "B(b)" indicates that "b is B."

Rule Should avoid (x): Unhealthy (x) ; "A:-B" indicates that "if B is true, then A is true."

Program

Query ? – should avoid (A) ; "?-A(a)" indicates "What is a for A?"

Result

A = the cigarette ; Conclusion "Should avoid the cigarette."

As exemplified in Figure 1-4-2, the inference is conducted with the syllogism.

"Smoking the cigarette is unhealthy." → "Taking a thing unhealthy should be avoided." → "Smoking the cigarette should be avoided."

1.4.4 Object-Oriented Programming

The programming method to describe solutions with states of objects and their behaviors is called object-oriented programming. Smalltalk-80 is a typical language supporting object-oriented programming. However, the following programs, all expanded versions of existing languages, are also widely used.

- The expanded versions of the C language → C++ and Objective-C
- The expanded version of Prolog → ESP
- The expanded versions of LISP → Flavor and CLOS
- The expanded version of Pascal → Object Pascal

As described in 1.2.3, with object-oriented programming, program development is done effectively by using the following facilities:

Object oriented	Instance	:	To create objects from classes
	Encapsulation	:	To combine data and methods
	Message passing	:	Message-sending/receiving
	Inheritance	:	Enabling children to inherit characteristics from a parent

1.5 Test and Review Methods

Tests and reviews are essential to enhance software quality.

1.5.1 Test Methods

Test methods include the white box test and the black box test.

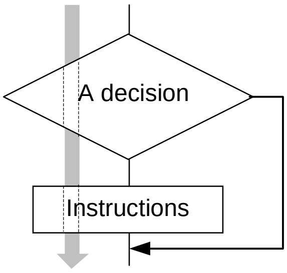
(1) The White Box Test

For the white box test, test cases are designed by giving particular consideration to internal structures of modules, and logic and control flows. In tests, it is desirable to review all the instructions (statements) included in programs in detail. However, it is difficult because of the amount of work required. Therefore, designs should be made considering a balance between "the degree of coverage" and "productivity."
The degrees of coverage used in designing test cases are described in the following:

① Instruction coverage

Instruction (statement) coverage refers to the design of test cases that enable every instruction (statement) in programs to be executed at least once. Instruction coverage does not take into consideration effects of decisions (paths).

Figure 1-5-1 Instruction coverage

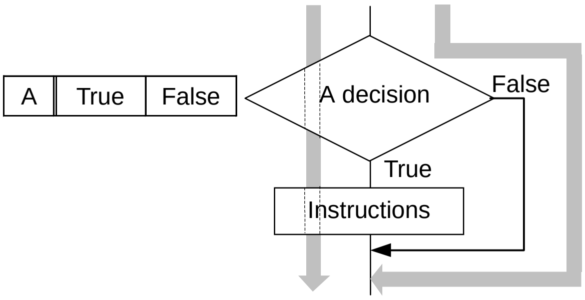


In the example shown in Figure 1-5-1, all the instructions to be executed are included in a single path. Testing the path involves all the instructions. Therefore, providing only one test case is sufficient for this test.

② Decision coverage

The decision coverage refers to the design of test cases that include the execution of the "true" condition and the execution of the "false" condition at least once for decision (see Figure 1-5-2).

Figure 1-5-2
Decision coverage

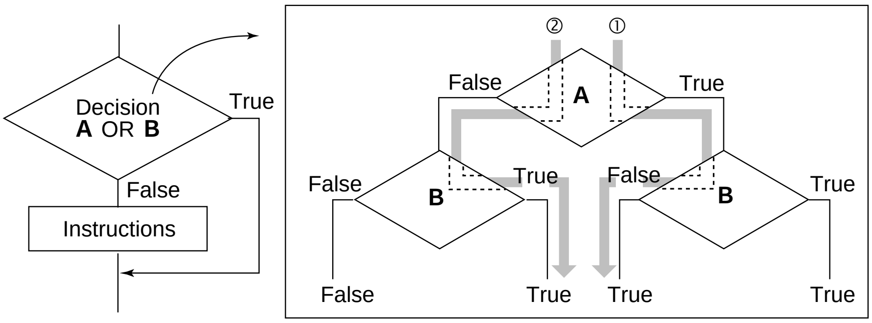


In the example shown in Figure1-5-2, the decision (branch) is covered if the two branch paths are tested. However, decision is made on compound conditions, there may be conditions which are not tested in.

③ Condition coverage

The condition coverage refers to the design of test cases enabling the testing of the combination of conditions in which the "true" and "false" states of each condition are executed at least once.

Figure 1-5-3
Condition coverage



For example, if the decision condition in Figure 1-5-3 is "if A = true OR B = true," then all the paths to be taken when either A or B is true must be covered by the test cases. So, data which cause the execution of the two paths shown in Figure 1-5-4 constitute the test cases.

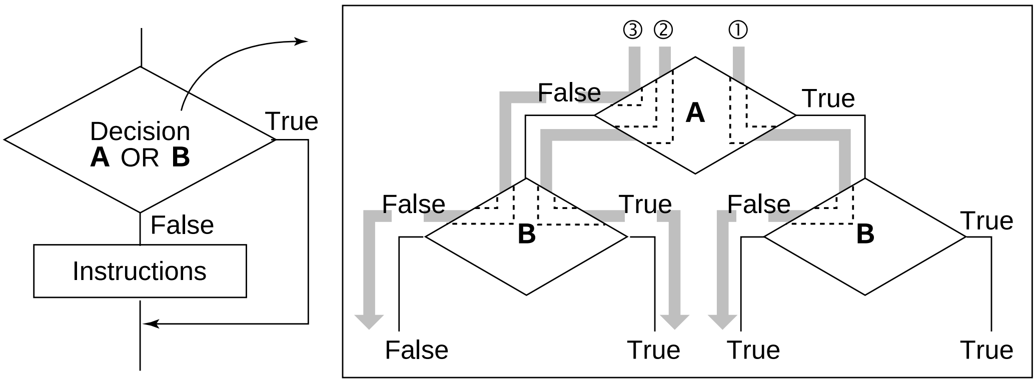
Figure 1-5-4
A test case of condition coverage

	①	②
A	True	False
B	False	True

④ Decision/condition coverage

In case of the condition coverage, "true" and "false" states of each of the conditions A and B of the multiple conditions are tested. However, the path for "A = "true" and B = "true"" and the path for "A = "false" and B = "false"" are not tested. On the other hand, the decision/condition coverage is applied to additionally combined with the condition coverage (see Figure 1-5-5).

Figure 1-5-5 An example of the decision/condition coverage



In the example shown in Figure1-5-5, the path executed when the condition is "false" is covered by a test case. Therefore, the three paths shown in Figure 1-5-6 are covered by the test cases.

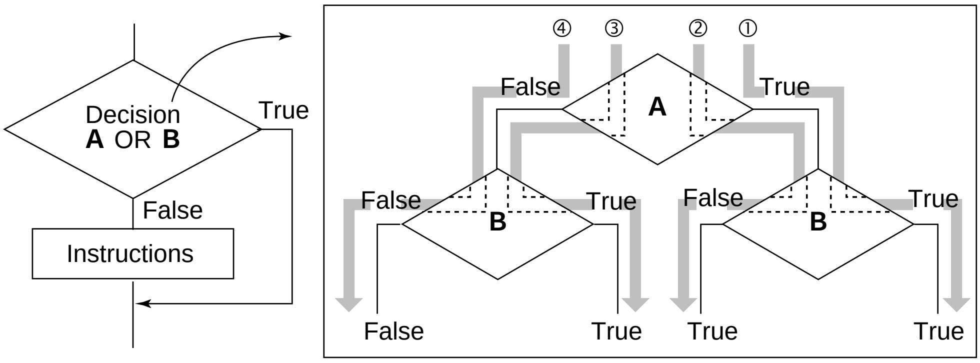
Figure 1-5-6
Test cases for the decision/condition coverage

	①	②	③
A	True	False	False
B	False	True	False

⑤ Multiple condition coverage

The multiple condition coverage employs test cases which cover all the paths which are executed depending on all the combinations "true" and "false" conditions and which makes every instruction included executed at least once.

Figure 1-5-7 Multiple condition coverage



In the example shown in Figure 1-5-7 where the decision condition is "if A = true or B = false, " paths to be taken for all the values of A and B are considered to prepare the test cases. Consequently, the following four paths in Figure 1-5-8 form the combinations to be executed by the test cases.

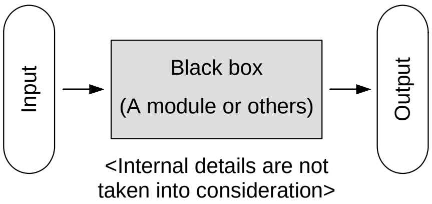
Figure 1-5-8
A test case of multiple condition coverage

	①	②	③	④
A	True	True	False	False
B	True	False	True	False

(2) Black Box Test

In the black box test, no consideration is given to internal structures and logic structures of modules. In other words, a module is considered a black box. Then, test data are designed by only paying attention to interfaces (inputs and outputs) of each module.

Figure 1-5-9
The black box test



Test cases are designed in the following ways:

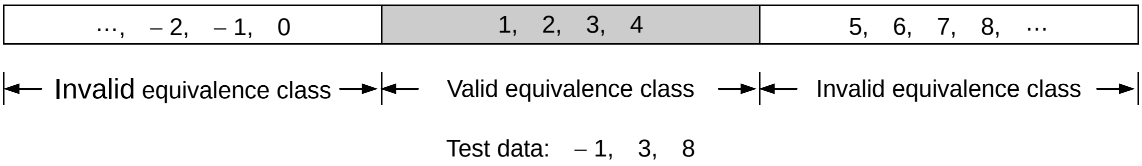
① Equivalence partitioning

In equivalence partitioning, data are partitioned into several different groups (equivalence classes), each element of which has the same property. Then, a piece of the data in each group is used as a representative value in each group. In equivalence partitioning, data are partitioned into either of the following two classes:

- Valid equivalence class : The class of data that are in the valid range.
- Invalid equivalence class : The class of data that are in the invalid (error) range.

Then, a typical data for each class is selected for a test case.

Figure 1-5-10 Equivalence partitioning

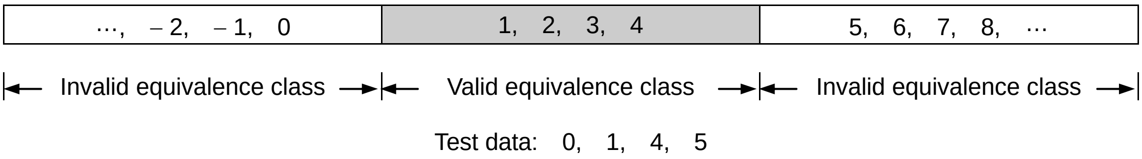


In Figure 1-5-10, if the valid equivalence class includes 1 to 4, then the set of data representing each class, for example, "-1, 3, 8" can be used as the test data.

② Boundary value analysis

The boundary value analysis is an analysis method that uses boundary values of valid equivalence classes as test data.

Figure 1-5-11 Boundary value analysis



In Figure 1-5-11, if the valid equivalence class includes data of 1 to 4, then the set of boundary data of each class, for example, "0, 1, 4, 5" can be used as the test data.

③ Cause-effect graph

In equivalence partitioning and boundary value analysis, data are classified for analysis. However, the cause-effect graph (cause-result graph) provides a method to design effective test cases for modules for which partitioning into classes is difficult.
<How to create cause-effect graphs>

1. All of the causes (inputs) and results (outputs) are picked up based on specifications.
2. Relationships between causes (inputs) and results (outputs) are expressed in graphs to make the logical relationships clear.
3. Decision tables are prepared based on the graphs. Test data are produced based on these tables.

An example

(Descriptions of specifications)

- The number of examination subjects is 5.
- "Pass" is the output, if the results of four or more subjects are successful.
- "Temporary pass" is the output, if the results in two or three subjects are unsuccessful.
- "Failure" is the output, if the result in only one subject is successful.

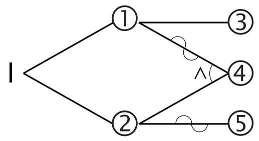
(Procedure 1) Pick up all relationships between causes (inputs) and results (outputs) based on the specifications.

Figure 1-5-12
Relationships between causes (inputs) and results (outputs)

Causes (inputs)	Results (outputs)
① Results in four or more subjects are successful ② Results in two or more subjects are successful	③ Pass ④ Temporary pass ⑤ Failure

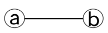
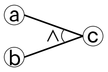
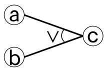
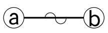
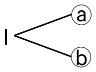
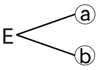
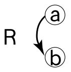
(Procedure 2) Express relationships between causes (inputs) and results (outputs) in graphs to make the logical relationships clear.

Figure 1-5-13
A cause-effect graph



The numbers ① to ⑤ correspond to those in Figure 1-5-12 (Procedure 1).

Meanings of symbols used in case-effect graphs are as follows:

	Symbol	Meaning	Descriptions
(1)		Equivalent	If ⓐ is true, ⓑ is true.
(2)		AND (product)	If ⓐ is true and ⓑ is true, ⓒ is true.
(3)		OR (sum)	If ⓐ is true or ⓑ is true, ⓒ is true.
(4)		NOT (Negation)	If ⓐ is false, ⓑ is true.
(5)		Include	ⓐ includes ⓑ.
(6)		Exclusive	If ⓐ is true, ⓑ is false, or if ⓐ is false, ⓑ is true.
(7)		Required	ⓐ requires ⓑ.

(Procedure 3) Decision tables are prepared based on the graphs.

Figure 1-5-14 A decision table

Cause and result \ Test case		1	2	3
Cause	① Passes in four or more subjects	Y	N	N
	② Passes in two or more subjects	—	Y	N
Result	③ "Pass" is outputted	X	—	—
	④ "Temporary pass" is outputted	—	X	—
	⑤ "Failure" is outputted	—	—	X

Rules used in decision tables are shown in the following:

Table name	Rule header			
	1	2	…	n
Condition 1	Y	Y		Y
Condition 2	Y	—		Y
Condition 3	Y	—		N
⋮	⋮	⋮		⋮
Condition n	—	—		Y
Condition 1	X	X		X
Condition 2	—	X		X
Condition 3	X	—		X
⋮	⋮	⋮		⋮
Condition n	—	—		X

- Table name: Indicates logic name
- Rule header: Indicates the number to differentiate logic decision rules
- Rows for conditions: Each row includes a condition to make decision in programs.
  Y: "true"
  N: "false"
  ―: No decision is made.
- Rows for actions: Each row indicates whether the processing is performed or not.
  X: Processing is performed.
  ―: No processing is performed.

④ Experimental design method

It is ideal to design every conceivable test case and to conduct tests using them. However, this requires a great amount of testing work. Therefore, the experimental design method, a form of statistical analysis, is used for tests requiring a large amount of data. Analyzing test data with this method and using the results as test cases makes it possible to conduct efficient testing.

An example The checking of data on students-- validity check of student registration numbers and codes of high schools graduated from, and the numerical check of grades-- are conducted.

(Procedure 1) Eight test cases are conceivable, if all the test cases are assumed for all the combinations of the three items: the validity check of student registration numbers and the validity check of codes of high schools graduated, and the numerical check of grades.

Figure 1-5-15
Student data

Item Test number	Student registration number	High school code	Grade
1	Valid	Valid	Number is entered
2	Valid	Valid	Number is not entered
3	Valid	Invalid	Number is entered
4	Valid	Invalid	Number is not entered
5	Invalid	Valid	Number is entered
6	Invalid	Valid	Number is not entered
7	Invalid	Invalid	Number is entered
8	Invalid	Invalid	Number is not entered

(Procedure 2) As for the three items, the following table (matrix) showing relationships among them is created:

Figure 1-5-16 A table showing relationships

Student registration number Code of high school graduated from	Valid	Invalid
Valid	Number is entered	Number is not entered
Invalid	Number is not entered	Number is entered

Combined

<Test cases>

Test No.	Code of high school graduated from	Student registration number	Grade
1	Valid	Valid	Number is entered
4	Valid	Invalid	Number is not entered
6	Invalid	Valid	Number is not entered
7	Invalid	Invalid	Number is entered

(Procedure 3) When compared with data on students in Figure 1-5-15, it is found that four of them are picked up. It can be concluded that, if all of the four tests give successful results, the other four tests should give equally successful results.

1.5.2 Review Methods

To perform system development in a top-down way, it is necessary to detect errors as early as possible. This is because hidden errors will significantly affect work at later stages. Errors generated in outline system designs, such as basic planning and external design phases affect more considerably, costing heavily to correct the errors. The correction work of errors in maintenance work reportedly costs several tens more than the correction work in coding.

Even though tests are conducted by covering all theoretically conceivable tests cases, eliminating all errors is difficult from the viewpoint of the constraints on the amount of time needed and other factors. However, reducing errors as much as possible has large effects on quality and error-correcting costs.

Therefore, as much as possible efforts should be made for reviews that offer effective measures to find errors.

Reviews indicate discussion and evaluation that, in each phase of basic planning through to programming, are made to pick up ambiguous or problematic points included in outputs (various design documents and source codes) from each phase. In particular, the review conducted in the design phase is called "design review," while that for program codes n the development phase "code review."

The following effects are expected from conducting reviews.

<Effects of reviews>

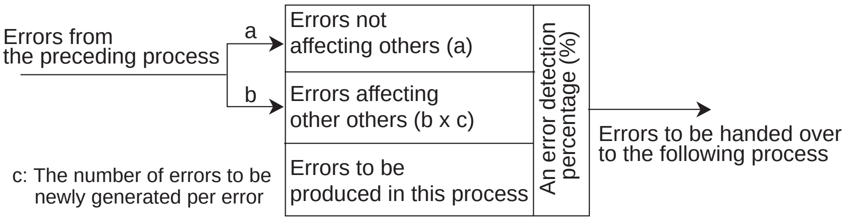
- Opportunities to reconsider ambiguous points are given, allowing participants to share the same understanding.
- Problem (errors), that actual developers in charge cannot find, can be detected by persons other than developers.
- Functions, performances and quality (including reliability, operability and maintainability) can be improved.

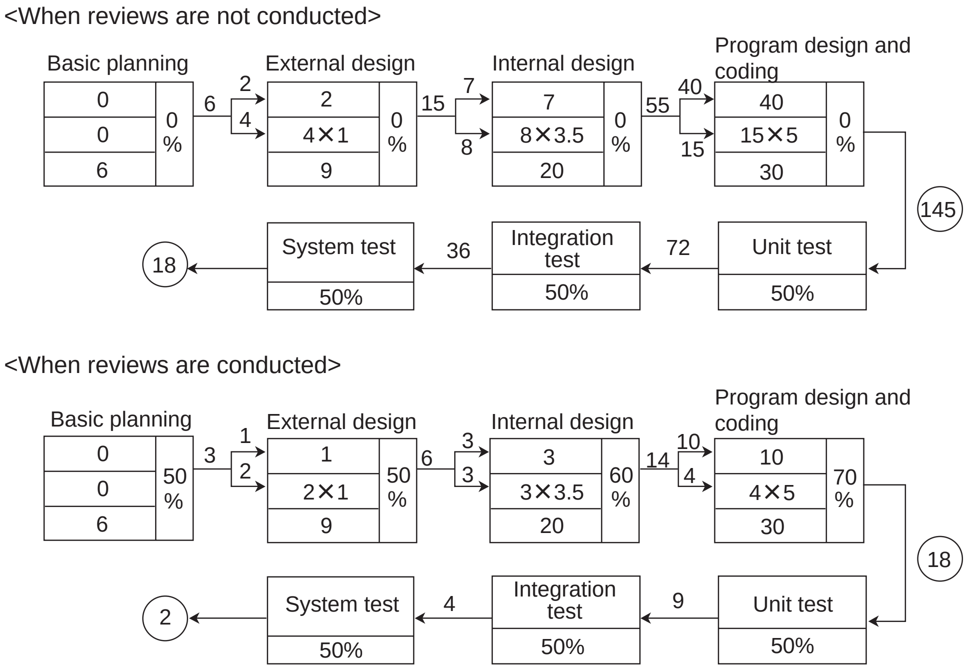
Among the items described above, reviews are particularly powerful in detecting problems (errors).

Let's compare, with the example given in Figure 1-15-17 as a model, economical aspects for a case where reviews are conducted and another case where reviews are not conducted.

Figure 1-5-17

A model for comparison in economical aspects





<When reviews are not conducted>

(Basic planning)

Six errors are injected and handed over to the following external design.

(External design)

Nine new errors are injected, and 15 errors in total are handed over to the following internal design.

(Internal design)

Of the 15 errors, 7 are passed through the internal design. It is assumed here that the remaining 8 affect others, injecting an average of 3.5 errors per received error. In addition, 20 other new errors are injected in this phase, bringing the total number of errors to be handed over to the program design and coding phase to 55.

(Program design and coding)

Of the 55 errors in total, 40 pass through the program design and coding phase. However, it is assumed here that the remaining 15 affect others, injecting an average of five errors per a received error. In addition, 30 new errors are created, bringing the number of errors to be handed over to the next phase to 145.

(Unit tests, integration tests and system tests)

It is assumed that the error-detecting rate is 50% for each test.

(The result)

The completed system includes 18 errors.

<When reviews are conducted>

(Basic planning)

Six errors are injected. Assuming that the review here enables errors to be detected at a rate of 50%, the number of errors handed over to the external design phase becomes three.

(External design)

Nine new errors are injected. Assuming that the review here enables errors to be detected at a rate of 50%, the number of errors handed over to the internal design phase becomes six.

(Internal design)

Of the six errors, three pass through the internal design phase. It is assumed here that the remaining three affect others, injecting an average of 3.5 errors per received error. In addition, assuming that 20 other new errors are created and that the review enables the errors to be detected at a rate of 60%, the number of errors handed over to the program design and coding phase becomes 14.

(Program design and coding)

Of the 14 total errors, ten pass through the program design and coding phase. It is assumed here that the remaining four affect others, injecting an average of five errors per received error. In addition, assuming that 30 other new errors are created and that the review enables the errors to be detected at a rate of 70%, the number of errors handed over to the test phase becomes 18.

(Unit tests, integration tests and system tests)

It is assumed that the error-detecting rate is 50% for each test.

(The result)

The system developed includes two errors.

Various cases are considered. However, as described above, the total number of errors remaining in the case where reviews are conducted is completely different. Therefore, effective reviews should be conducted. Typical review methods include walkthroughs and inspection.

(1) Walkthrough

Walkthroughs conducted by persons who produce artifacts, and those concerned (peers) aim to detect errors in designing and coding as soon as possible to improve quality.

(2) Inspection

Inspection, a method that formalizes the walkthrough, aims to detect errors in designing and coding as soon as possible to increase quality. The inspection, chaired by a person, called the "moderator," is in charge of the meeting, reviewing processes that include error correction and the checking of correction results, and each participant's roles are defined in detail.

(3) Points to be Considered in Conducting Reviews

When reviews are conducted, the following points should be taken into account:
<Points to be considered>

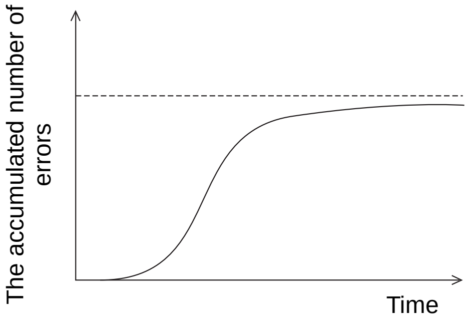
- Documents to be used in the meeting should be distributed two to three days in advance, and the participants should check them before discussing them with other participants.
- Concentrate on detecting errors and make the meeting as short as possible. Holding such meetings frequently reduces the amount of work to be redone.
- Multiple, but only developers should participate in the meeting.
- Managers should avoid attending the meeting. The meeting is held to detect errors. If a manger participates in the meeting, participants will hesitate to express their own opinions freely, diminishing the error-detecting potential that the meeting has.

1.5.3 Test Design and Management Methods

(1) Bug Curve

The bug curve offers a model that uses the number of bugs to quantitatively estimate the reliability of a program. The curve, also called a "reliability growth curve" and a "quasi-mathematical model," is prepared by placing the accumulated number of errors on the longitudinal axis and time on the horizontal axis. An inflection point is located midway along the curve and convergence is achieved ultimately. The logistic curve and the Gompertz curve are often used for this purpose.

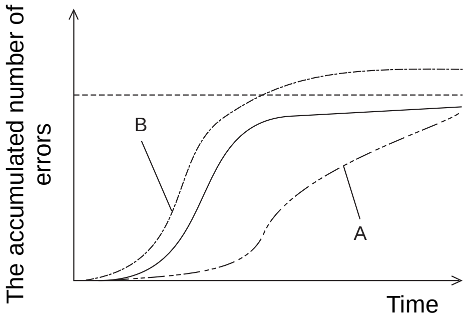
Figure 1-5-18
A bug curve (reliability growth curve) example ARC



(2) Bug Management Diagram

The bug management diagram manages bugs by using bug curves (see Figure 1-5-19).

Figure 1-5-19
A bug management diagram example



Actual data plotted are compared with a bug curve, and it is possible to conduct the following judgments.

- Build-up is slow (A) : The use of defective test cases and unsatisfactory test environments is considered.
- Convergence is not achieved (A) : Programs must be reconsidered.
- The accumulated number of errors detected remain lower than the bug curve (A) : The quality may be high or the test cases used may not be good.
- The accumulated number of errors detected goes above the bug curve (B) : The test cases used may be good or the quality of programs may not be good.

Taking into account actually obtained data and the level of the developers' skill, it is possible to make a judgment about the status of test processes and the quality of programs.

(3) Coverage

"Coverage" is also called "test coverage." The coverage for a program means the percentage of the paths covered out of all the paths included in the program, and the phrase indicates a quantitative expression for the coverage method described on the white box test. In Japan, software developer generally use 100% coverage as the standard, but there are many as well that target coverages less than 100%. The higher, the better.

(4) Test Design

It has already been learned that the test method includes the white box test and the black box test and that there are several methods for designing test cases. The use of design cases generated only with one of the two methods is insufficient. Generally, it is effective to conduct tests based on the black box test centered on testing functions, and supplement them with the white box test.

① Boundary conditions for inputs and outputs and conversion conditions are checked based on external design documents for programs. Then, test cases are generated. The test cases should always be limited to those that check whether necessary functions (WHAT) are achieved.

② Based on program source lists and detailed specifications (related to algorithms described on a statement basis, such as program specifications and module specifications), branching patterns of the programs are checked. Then, test cases enabling both "true" and "false" conditions of each decision-making condition to be executed are produced.

③ Check that test cases ① and ② described above can execute all of instructions included. For loops, test cases for getting-out-of-loop conditions of 0, 1, and the maximum times are created.

④ After producing the test cases for ①, ② and ③, test results for each test case are assumed theoretically, generating test condition documents.

(5) Fault Injection

It is impossible to eliminate all defects in software. This is because it is impossible to verify that no error is included. Dijkstra, who proposed structured programming, said:

"Tests can show that errors exist, but can not prove that no error is included."

Myers, who proposed structured design, had this to say about tests:

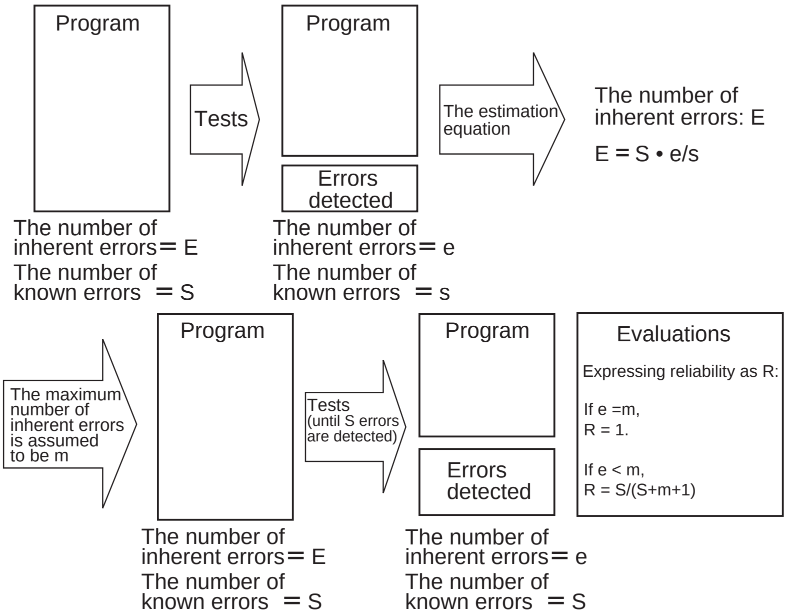
"The test is a process to check that programs are proper, or, in other words, are not to show that no error is included."

"The test is a process to execute programs to detect errors."

After all, nobody can say "the program is error-free" even after tests are conducted. However, a delivery time is specified for system development. Therefore, it is necessary to complete the system development and to start actual operations. Consequently, there comes a time when designers are forced to declare that the test operations have been completed.

The fault injection, is a method to judge the date of the completion of test operations numerically; known errors are injected intentionally, and the number of latent inherent errors is estimated based on the ratio of the number of inherent errors detected to the number of known errors detected.

Figure 1-5-20
The fault injection


Program
Tests
Program
Errors detected
The estimation equation
The number of inherent errors: E
E = S • e/s
The number of inherent errors = E
The number of known errors = S
The number of inherent errors = e
The number of known errors = s
The maximum number of inherent errors is assumed to be m
Program
Tests (until S errors are detected)
Program
Errors detected
Evaluations
Expressing reliability as R:
If e =m, R = 1.
If e < m, R = S/(S+m+1)
The number of inherent errors = E
The number of known errors = S
The number of inherent errors = e
The number of known errors = S

1.6 Development Environments

Needs for enterprises have been more and more diversified and advanced. Market situations are changing rapidly. Companies will be threatened if they don't respond quickly as well as consistently to these changes. Therefore, information systems, supporting all aspects of business activities, must be developed in short periods of time, and must be high in quality as well. If the development of a system needs several years, situations surrounding the system will become completely different when it is completed, and lots of investment in the development will be wasted. Therefore, such long development periods are not permitted even for large-scale, highly important systems.

In this section, development environments mostly aiming at "increases in development productivity" are discussed. If work is automated as much as possible and software components are used to the maximum extent possible, as in the production of industrial products, high and uniform quality products are generated.

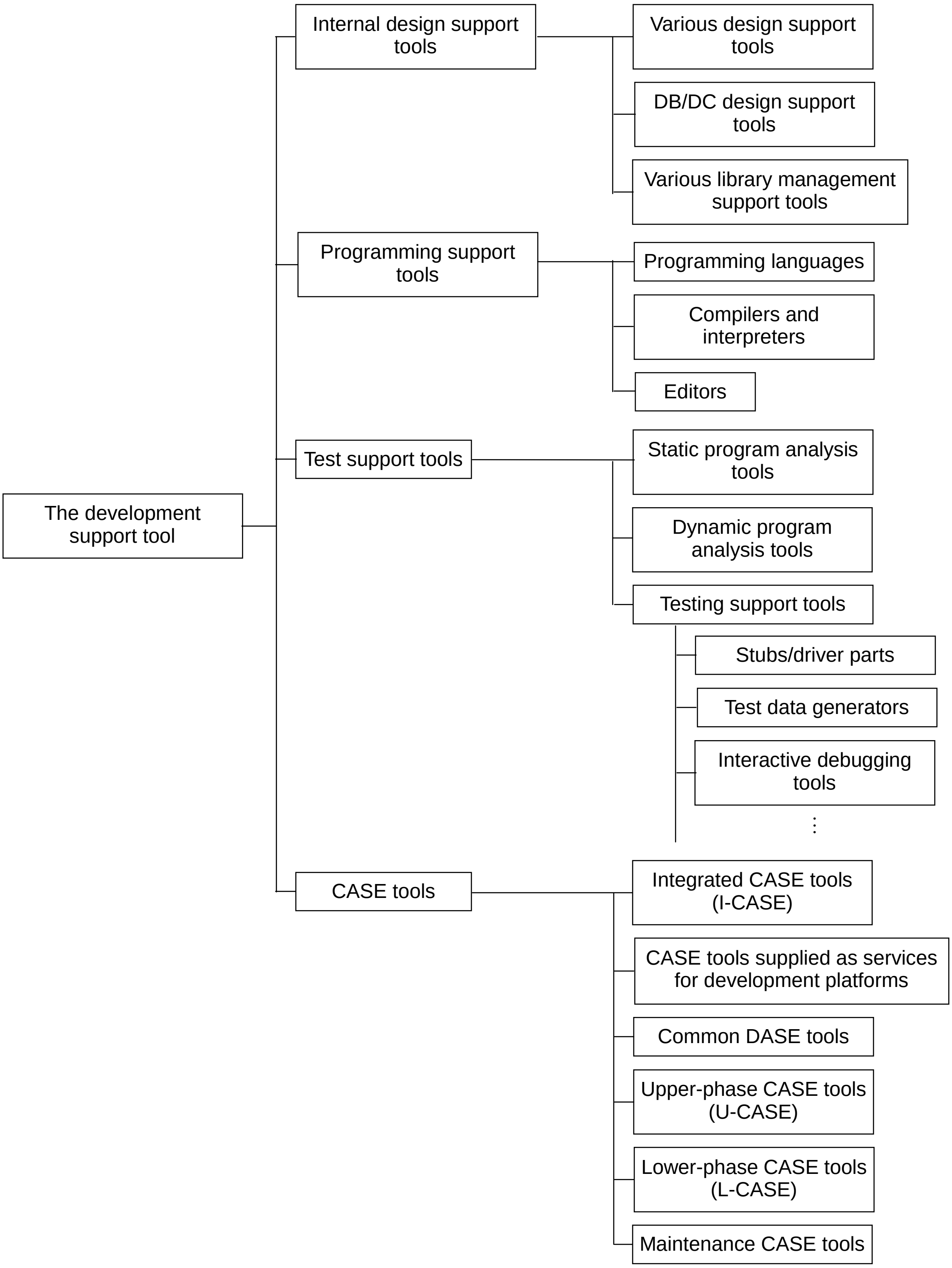
1.6.1 Development Support Tools

Various types of system development support tools are available as shown in Figure 1-6-1.

Development support tools include everything from text editors and programming languages both already in use to CASE tools described later. There are many options, and they effectively enable gigantic increase in productivity.

Below, types, functions and characteristics of development support tools are described.

Figure 1-6-1 A classification of development support tools

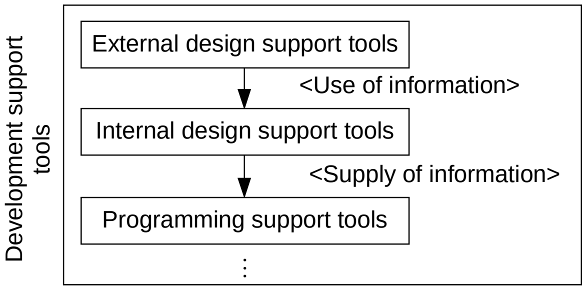


The development support tool
Internal design support tools
Various design support tools
DB/DC design support tools
Various library management support tools
Programming support tools
Programming languages
Compilers and interpreters
Editors
Test support tools
Static program analysis tools
Dynamic program analysis tools
Testing support tools
Stubs/driver parts
Test data generators
Interactive debugging tools
⋮
CASE tools
Integrated CASE tools (I-CASE)
CASE tools supplied as services for development platforms
Common DASE tools
Upper-phase CASE tools (U-CASE)
Lower-phase CASE tools (L-CASE)
Maintenance CASE tools

(1) Internal Design Support Tools

Internal design support tools, tools to support internal design work, are mostly supplied as a part of an integrated development support system. Since development using the waterfall model is made in a top-down method, later process work is carried out by using information registered and generated with an internal design support tool.

Figure 1-6-2
The placement of the internal design support tool


Development support tools
External design support tools
<Use of information>
Internal design support tools
<Supply of information>
Programming support tools
⋮

The internal design support tool is mostly used to support the following work.

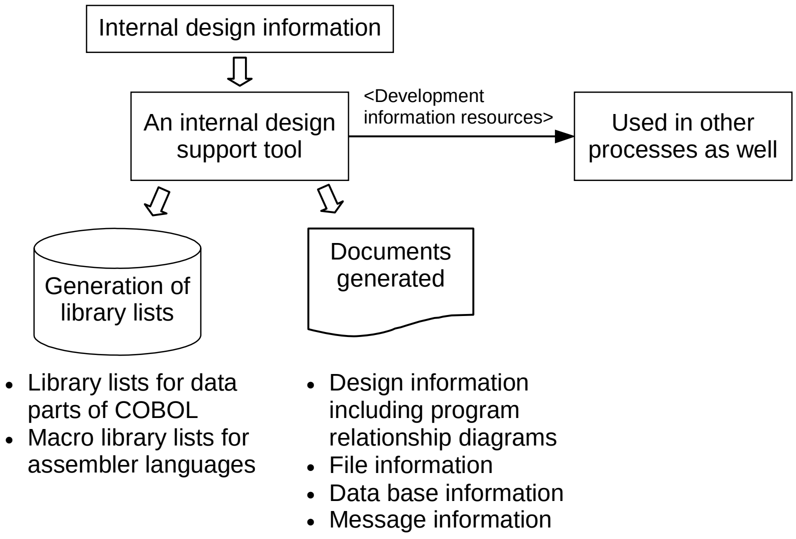
① Various design supports
- Functional partitioning, and structured design
- Screen design
- Physical data (files) design
- Various document design
- Message design

② Design supports of DB/DC (DataBase/Data communication)
- Designing DB
- Designing DC (including messages between terminals and the host in online systems)

③ Management of various libraries
- Library management of data parts of COBOL
- Management of macro libraries of assembler languages

Figure 1-6-3 shows relationships between input to an internal design support tool, the execution of the tool, and outputs from the tool.

Figure 1-6-3
Input to an internal design support tool and outputs from the tool


Internal design information
An internal design support tool
<Development information resources>
Used in other processes as well
Generation of library lists
Documents generated
• Library lists for data parts of COBOL
• Macro library lists for assembler languages
• Design information including program relationship diagrams
• File information
• Data base information
• Message information

(2) Programming support tools

The programming support tool includes software such as service programs used in programming. It also includes:
- Programming tools
- Language processors
- Editors

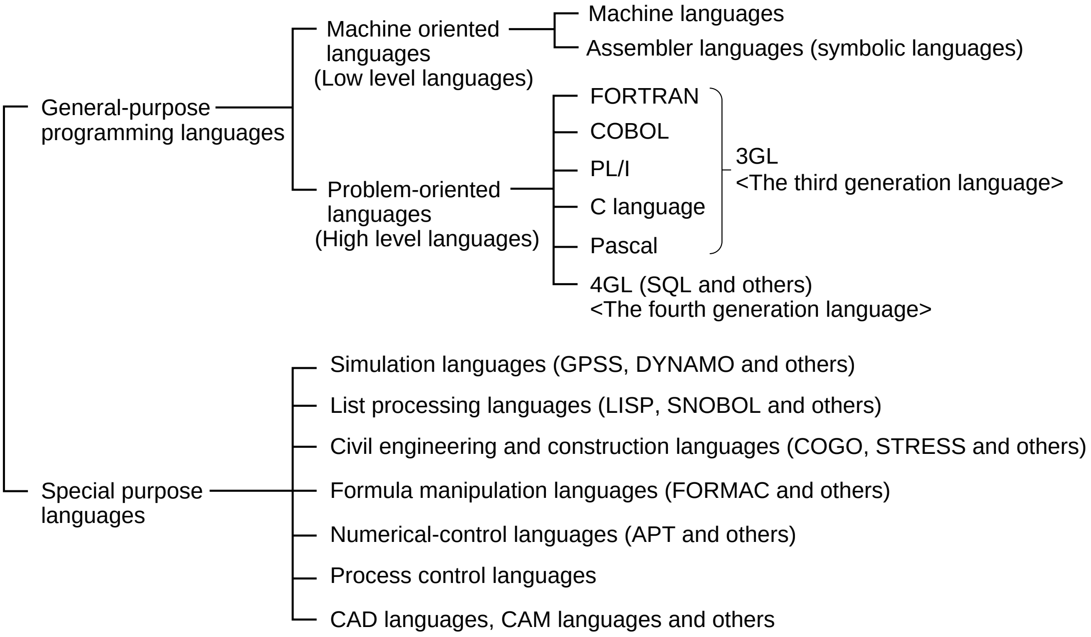
Below, details on each of these items are provided.

① Programming languages

The programming languages can be classified as in Figure 1-6-4.

Recently, non-procedural languages, typically the fourth generation language (4GL) and including SQL, have been used frequently. This is because 4GL offers higher productivity than 3GL (the third generation language).

Figure 1-6-4 Types of programming languages


General-purpose programming languages
Machine oriented languages (Low level languages)
Machine languages
Assembler languages (symbolic languages)
Problem-oriented languages (High level languages)
FORTRAN
COBOL
PL/I
C language
Pascal
3GL <The third generation language>
4GL (SQL and others) <The fourth generation language>
Special purpose languages
Simulation languages (GPSS, DYNAMO and others)
List processing languages (LISP, SNOBOL and others)
Civil engineering and construction languages (COGO, STRESS and others)
Formula manipulation languages (FORMAC and others)
Numerical-control languages (APT and others)
Process control languages
CAD languages, CAM languages and others

② Language processor

a. Compiler

The compiler, a language processor used in compiler languages like COBOL, translates the entire source program, as a whole, into objects (machine language programs).

b. Interpreter

The interpreter, a language processor used in interpreter languages like BASIC, executes source codes line by line. It allows easy debugging, because even the codes of incomplete programs can be executed.

③ Editor

The editor is software to generate and edit programs. The editor includes the text editor and the structure editor.

a. Text editor

The text editor is used to generate and edit programs.

b. Structure editor

The structure editor, an editor enabling programs to be edited, with their structures being made clear as well, allows various specifications in addition to programs to be edited.

(3) Test support tools

The main objective of tests is to remove bugs, and utmost efforts must be made to achieve the objective. However, the work to generate test data alone is quite laborious. It is not wise to take much time in such work.

To remove bugs efficiently as well as to make each test evenly weighted and impartial, it is necessary to use tools as much as possible.

There are various types of test support tools. They are broadly classified into the following two types:

- Program test support tools

- Test execution support tools

① Program test support tool

The program test support tool includes the static program analysis tool that is used to analyze programs without executing them and the dynamic program analysis tool that is used to analyze programs while executing them.

② Test execution support tool

Test execution support tools include tools to provide environments for dummy modules required in linkage tests, and test data generators that, with parameters given, generate test data automatically.

Figure 1-6-5 Various test support tools

Tools				Functions
Program test support tools	Static program analysis tools	Code analysis tools		Used to increase readability of source programs.
		Structure analysis tools		Used to generate test cases.
		Module interface-checking tools		Used to check interfaces between modules.
	Dynamic program analysis tools	Monitoring tools Online simulators		Control flows are traced. Then, data on the number of times each path is used and on execution time are outputted to files.
		System-logging tools		Use rates of resources are outputted to files.
		Automatic test data generation and execution tools		Used to generate test data and to execute programs.
Test execution support tools		Stub/driver tools		Used to set up environments where modules and programs are run.
		Test data generators		Used to generate program test data automatically.
		Interactive debugging tools		Source programs are displayed on a screen, and execution is conducted interactively.
		Utility (service) programs	Dump	Data in a main memory unit and registers are outputted to output devices and memory devices.
			Tracers	Execution states of a program are traced successively to get necessary data.

(4) CASE tool

CASE is the abbreviation of Computer Aided Software Engineering.
The CASE tool indicates development support tools aimed to automate software development work or to increase efficiency of the work, and uses various development techniques combined with the structured program design, a software engineering technology, as the core.

① Characteristics of the CASE tool

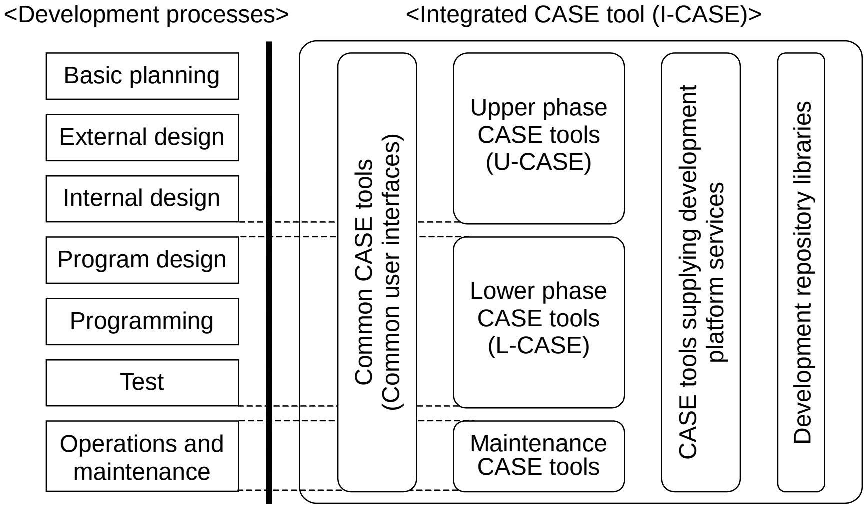
The CASE tool includes the following characteristics:

- The CASE tool is developed with the aim of centralizing system development management and automating various works. Therefore, in the CASE tool, various data on system development is stored in a software development information data base called a "repository." In other words, data stored in the repository is used any time to centralize system development management and to automate various works.
- Various CASE tools are used together to cover the entire processes of system development.
- User interfaces are unified to make works easier.
- The tool offers data, not only on development works such as analysis and design, but all data on system development, including data on quality management and project management.

② Types of CASE tools

The CASE tool includes tools shown in Figure 1-6-6.

Figure 1-6-6 Types of CASE tools


<Development processes>
<Integrated CASE tool (I-CASE)>
Basic planning
External design
Internal design
Program design
Programming
Test
Operations and maintenance
Common CASE tools (Common user interfaces)
Upper phase CASE tools (U-CASE)
Lower phase CASE tools (L-CASE)
Maintenance CASE tools
CASE tools supplying development platform services
Development repository libraries

a. Upper phase CASE tool (U-CASE)

The upper phase CASE tool covers the planning phase to the internal design phase. With structured methods such as DFD, the tool supports various design works including those for various modeling, requirement definitions, databases and networks.

b. Lower phase CASE tool (L-CASE)

The lower phase CASE tool covers the program design phase to the test design phase. It supports automatic programming and the automation of various tests.

c. Maintenance CASE tool

The maintenance CASE tool supports the modification and correction of systems, and maintenance work.

d. Common CASE tool (Common user interfaces)

The common CASE tool supports management work covering the entire system development processes, including document management and project management.

e. CASE tool supplying development platform services

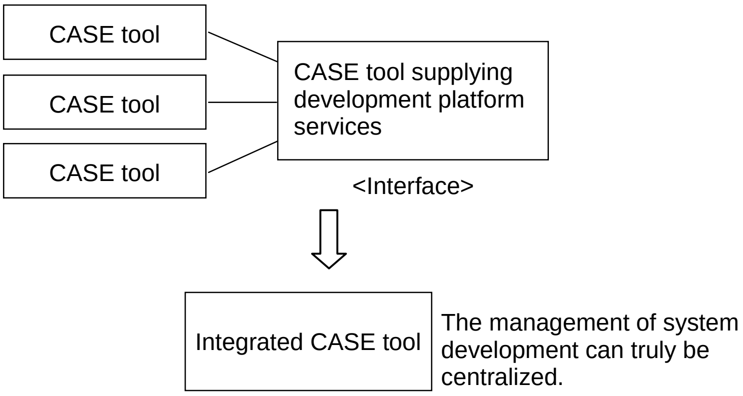
The CASE tool supplying development platform services provides interfaces between CASE tools and supports the integration of tools.

f. Integrated CASE tool (I-CASE)

The integrated CASE tool, attracting the most attention among the CASE tools, integrates functions provided by CASE tools used in each phase from the upper phase to the maintenance phase, offering support to every phase of a system life cycle consistently.

RAD (Rapid Application Development) is a method to enable effective system development by forcing the user's participation in a phase earlier in a system development and by automating development work by fully using an integrated CASE tool.

Figure 1-6-7
Integrated CASE tool


CASE tool
CASE tool
CASE tool
CASE tool supplying development platform services
<Interface>
Integrated CASE tool
The management of system development can truly be centralized.

1.6.2 EUC and EUD

Formally, system development was work to be performed exclusively by the information system department. However, as workload for the maintenance of existing systems increases the amount of backlog, the department cannot afford to handle new development projects readily. In addition, end users now want to use data freely. In such situations, the concepts of EUC (End User Computing) and EUD (End User Development), indicating that users themselves develop their own systems, have been gaining support. Behind reasons that make EUC and EUD possible, there exist the facts that personal computers have been used widely and their functions have also advanced, and that user-friendly software packages, such as spreadsheet and simple database software, have become available.

(1) Use of Spreadsheet Software

Spreadsheet software has been developed for use in aggregating data and generating tables. The software also enables aggregated data to be expressed in visual forms like graphs. With these functions used as a business tool, it is possible to generate simple business systems. In addition, the spreadsheet software is provided with functions to take in data from backbone systems. The functions allow users to take in necessary data from backbone systems and to manipulate it freely.

Recently, GUI has been supported in user interfaces, providing better development environments and easier operational capabilities.

(2) Use of Database Software

Database software as large as that used in backbone systems is not required for EUC. However, database software can take in data from backbone systems. Therefore, the software enables the taking in of only necessary data from backbone systems, then the manipulation and analysis of it. Generally, data that is taken from a backbone system and manipulated in a departmental system is not rewritten to the backbone system again. Needless to say, it is possible to generate databases by inputting the department's own data and to use it.

① Normalization of databases

Database design is important in creating a database. The normalization of databases is a burdensome work, but some database software is provided with functions that normalize database automatically when necessary data is inputted.

② Data manipulation

Data is manipulated interactively by use of a command format or menu format, and SQL statements are automatically generated. Therefore, the making of simulation-like use, such as analysis, is quite easy, requiring no special knowledge. When skill improves, more advanced use, such as coding and executing SQL statements, becomes possible also.

1.7 Development Management

1.7.1 Project Planning

To successfully complete a development project, planning in advance, and management based on the plans provided thus, is important. Preparing plans helps make the entire view of a development project clear, enabling checking and investigating in advance of project problems and risks, in addition to objectives, functions and coverage of the system, and the amount of work and the number of workers needed for the project.

(1) Project Planning

Project plans are produced for the following items:

- Outputs, work, schedules, quality, risks and others
- Developers : The organization, bringing-up of human resources, communications means and others
- External procurement : Where to make procurement, ways to procure, delivery time, quality (or skill) and others
- Development environments : Hardware, software and others
- Development cost : Personnel expenses, the cost of equipment, expenses and others

Detailed plans are created for these items. These plans provide bases to judge whether the project should be inaugurated or not. In addition, they are also used as targets for management after the project is approved.

(2) Profitability of Projects

Since system development is performed as a part of business activity, profitability, naturally, is sought after the development. This means that a so-called effects-to-cost evaluation is necessary. If a system development is known to be unprofitable at the planning phase, the project would not be approved without a compelling reason, for example, that the system is needed to meet requirements of laws or regulations.

Even if profitability is expected at the planning phase, it often happens that the project is found utterly unprofitable at completion. For example, the following scenarios are conceivable: The customer changes requirements at the design phase, increasing the amount of work needed. Functions assumed in advance for the development environments are actually supported unsatisfactorily. What is procured externally is poor in quality, generating lots of rework and affecting the delivery time. Some scenarios show up only when actions are taken. It can be said that projects are more likely *not* to proceed as planned.

Therefore, the reconsideration of plans in each phase becomes necessary, as well, to remain profitable. In particular, modifications of specifications must be reflected correctly. So, to secure quality, it becomes important to carry out modification management.

1.7.2 Quality Planning, Management, and Evaluation

(1) QFD (Quality Function Deployment)

Tests of software are only to remove defects and to maintain designed quality. On the other hand, QFD is a technology to generate higher quality software. QFD is devised to increase the quality of hardware design. The outline of QFD is as follows: With QFD, quality of software including subsystems and modules is represented as quality characteristics that allow tangible evaluation and the same method of evaluation is used systematically for the quality of subsystems and modules as well.

(2) Software Quality

As is well known, the computer is inflexible. It is a lie to say, "This computer is smart." It is more proper to say, "This software (program) is smart." However, the quality of software never improves without the creation of better software. In general-purpose computers, entering character data instead of numerical items causes the computer to come to an abnormal end. To avoid this, it is possible, with a program, to check data entered and to ask that data be entered again if the data entered first is other than a numeral. In other words, providing an error-checking function makes the program smarter, improving the software quality.

Such a function should not depend on the sense of programmers. It must be taken into consideration without fail in the design phase where user requirements are itemized in detail. If the necessity of the checking function is confirmed as a user's latent need, but is not described in specifications in the design phase, the function will never be achieved. Defects originating in requirement definitions in the upper-most phase may only be found in operation tests, the lower-most phase. Consequently, the correction could, most likely, be impossible, even after the defects are found.

In other words, if enough time is allocated in the upper phases for verification and correction, then the amount of correction needed in the lower phases is reduced. Actually, it is said, "One hour used to prevent defects in the upper phases eliminates three to ten hours of correction work in the lower phase." To increase software quality, execution of the so-called Plan-Do-Check-Action (PDCA) cycle -- planning, executing the plan, evaluating outputs and taking actions based on evaluations -- is required.

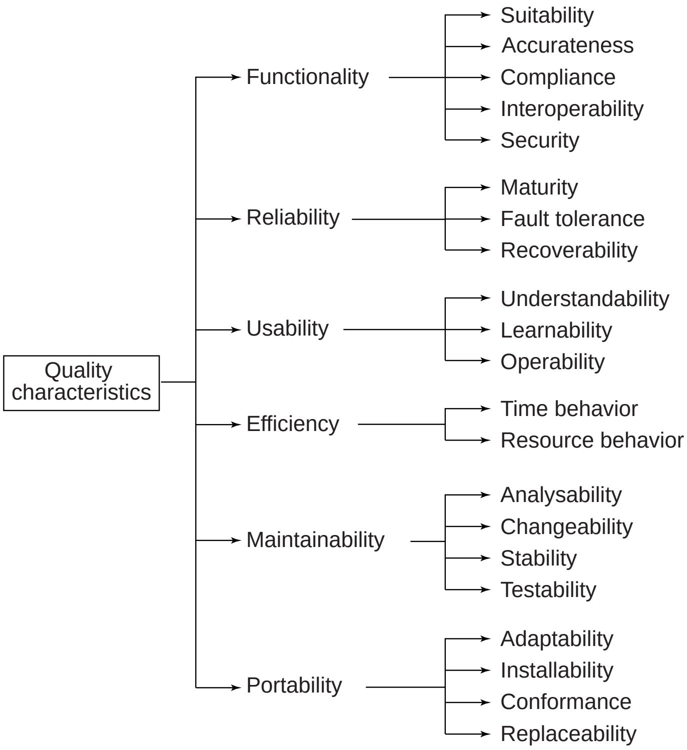
(3) Software Quality Characteristics

There are many software evaluation methods. Here, six quality characteristics listed in ISO/IEC 9126 shown in Figure 1-7-1 are described.

① Functionality (functional characteristics)

- Functions required for the system are implemented (suitability)
- Functional accuracy is provided (accurateness)
- Functions meet specifications (compliance)
- Ease of connecting with other systems is provided (interoperability)
- Substantial security is provided (security)

Figure 1-7-1 Quality characteristics of ISO/IEC 9126



② Reliability (reliability characteristics)

- Software includes no bug: maturity.
- A certain system level is maintained even when a trouble occurs: fault tolerance.
- Normal operations are restored readily, when a failure occurs: recoverability.

③ Usability (ease-of-use characteristics)

- Easily operational: understandability.
- Easy to remember: learn-ability.
- Allowing easy operation management: operability.

④ Efficiency (performance characteristics)

- Providing good responses and high throughput: time behavior.
- Allowing effective use of system resources: resource behavior.

⑤ Maintainability (maintenance characteristics)

- Allowing easy analysis of design documents and programs when a bug is found: analyze-ability.
- Allowing easy expansion and modification of the system: changeability.
- The modification of the system does not affect others: stability.
- Laborious tests are not required after a modification is made: testability.

⑥ Portability (characteristics of porting programs to other computers)

- Having adaptability: adaptability.
- Providing easy installation work: install-ability.
- Complying with porting specifications: conformance.
- Allowing easily replacement with other software: replace-ability.

1.7.3 Process Management

Process management is classified into process planning and progress management. Here, characteristics of each, and methods used for them are explained.

(1) Outlines of Process Planning and Progress Management

① Process planning

System development is completed through various processes. Some big works take several years to complete. Therefore, precise process planning is an important work management item.

In particular, new development projects involve many uncertain factors that cannot be determined definitely in the process-planning phase. Therefore, as work process advances, the flexible handling of situations, such as making delivery dates earlier (depending upon situations) or minimizing delays, becomes necessary.

② Progress management

Progress management is managing the progress of work. It is necessary to check the progress of work and to take some actions for work whose progress is behind schedule. System development work performance is shared by more than one person. Therefore, one person's delay in work leads to a delay in the progress of the project as a whole. Consequently, thorough progress management is required to minimize the effects of delays as well as to detect problems as early as possible.

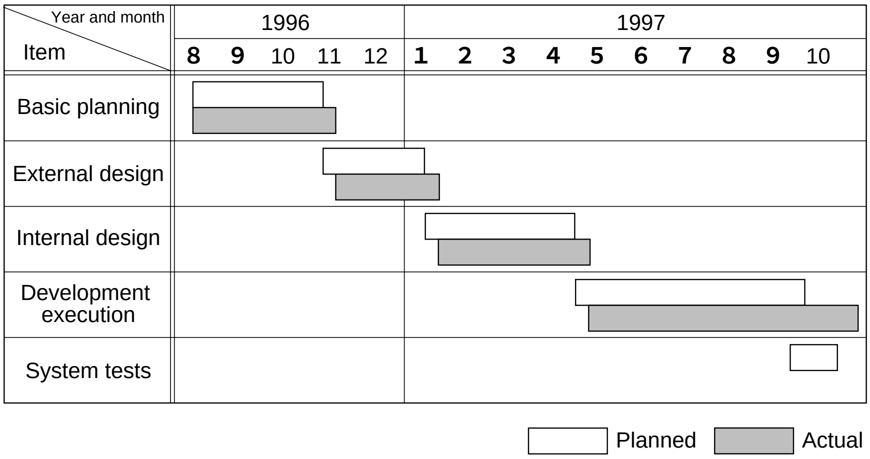
(2) Process Planning

The Gantt chart and PERT are typically used as process planning methods.

① Gantt chart

The Gantt chart is also called "bar chart."

Figure 1-7-2 A Gantt chart example



<Characteristics>

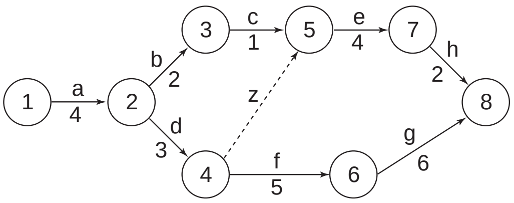
- It provides easily understandable diagrams in which the schedule of each work section is indicated with a horizontal line (bar).
- In the chart, scheduled start and finish timing of each work section and present status the work are shown clearly.
- Priorities of work sections are not shown.
- Degrees in which delay in each work section affects other works are not shown.

② PERT (Program Evaluation and Review Technique)

PERT offers an engineering technique to generate schedules for work sections (processes) of a project, and then to manage them.

Figure 1-7-3

A PERT diagram example (the first of two diagrams)



* a to h: Each indicates work to be done there. Numerals: Each indicates the number of days needed for the work.

---➤ : A dummy line (showing only the order of work to be done).

<Characteristics>

- PERT can handle the development of large-scale and complex systems as well.
- It enables the total number of days required (the minimum period necessary) to be calculated.
- The order in which work is to be done is made clear, enabling important management points to be clarified.
- The number of days included in each work section as a margin is easily calculated.
- It can be applied to calculations to reduce development costs and to decrease the number of days needed for work (based on CPM (Critical Path Method) or other methods).

<Procedures>

1. The estimated number of days needed for each work section is determined. Then, the results are put together in a table like that in Figure 1-7-4.

Figure 1-7-4

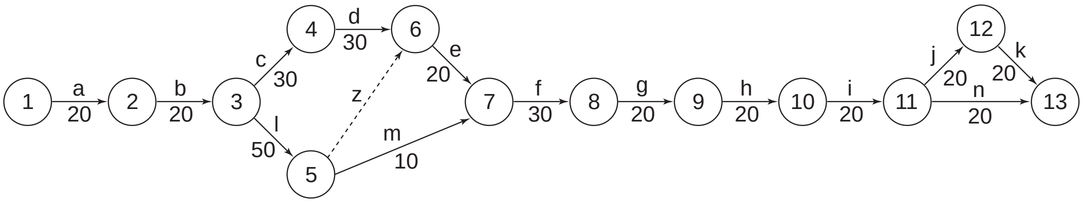
A work estimate table example

Symbol	Work item	The number of days needed	Work or works to be done in advance
a	System requirement analysis and definition	20	–
b	System design	20	a
c	Detailed operation design	30	b
d	Software requirement analysis and definition	30	c
e	Software system design	20	d, l
f	Detailed software design	30	e, m
g	Programming	20	f
h	Software linkage tests	20	g
i	Total software tests	20	h
j	System linkage tests	20	i
k	System tests	20	j
l	Installation of hardware	50	b
m	Setting up of development environments	10	l
n	Education and training of persons concerned	20	i

2. Based on the work estimate table, the PERT diagram shown in Figure 1-7-5 is drawn (take care about the order of works).

Figure 1-7-5

A PERT diagram example (the second of two diagrams)



3. The following numbers of days are decided at each node.
   - The earliest possible linkage time : Indicates the earliest possible time, before which work cannot be started.
   - The latest possible linkage time : Indicates the latest time by which the work must be completed.
4. The path generated by connecting the nodes, for each of which the earliest possible linkage time and the latest possible linkage time are the same (indicating that no margin is provided), is called "critical path." Works on the line are the most important for management.

(3) Progress Management

Progress management is conducted from the following two viewpoints;

- The starting and finishing timing of each work section
- Progress status of each person's individual work

① Management of the starting and finishing timing of each work

Management is conducted so that each work section is started as specified in the process plan and is finished as specified by all means, by checking intermediate progress status of each work section and by taking appropriate measures based on that status.

The following are considered reasons for delays in work:

- Skill of engineers concerned is insufficient.
- Planning and the setting of targets are not well considered.
- Personnel-related problems (including the redeployment of development members, the transfer of some of the members to other locations, and some of the members' leaving the company)
- Budgetary problems (including that a development support tool to be purchased may not actually be bought).
- Hardware and/or software troubles

When a delay in work is found, the manager, such as the project leader, must investigate measures to handle the situation and take concrete measures, such as modifying processes, as soon as possible.

To enable taking such measures, every workforce member must report the progress status of his or her work regularly by means of a work diary or weekly work report. In particular, when an unexpected situation occurs, he or she must report it as soon as possible.

② Scheduling of each workforce member

Scheduling is used to allocate the work of each process to each workforce member, to decide the order in which each of the work sections is conducted, and to manage work progress status on a daily basis. Scheduling is also effective to make delivery timing earlier and to minimize delays.

An example The external design, for designing the outline of a system, is composed of many work sections, in which a large number of system engineers (SEs) are engaged.

In this example, scheduling becomes as follows when considered from the leader's viewpoint and the workforce member's viewpoints, respectively:

<The project leader>

1. The system design work is broken down into some small works.
2. Each work is allocated to each member depending on his or her skill level.

3. Scheduling for each member (each work) is generated.
4. Instruction to assign work is given to each member.
5. The completion of each work is checked by reading each member's work diary or weekly report.
6. Flexible measures are taken, such as modifying the plan if necessary, when a delay in work is found.

<Workforce members>

1. He or she manages himself or herself by comparing the work progress status with the scheduling of work allocated and tries to keep the finish timing planned.
2. He or she reports work progress status by means of a work diary or weekly report, in addition to a report when the work is completed.

As described above, progress management is conducted by combining ① the rough process plans (deciding work-starting and work-finishing dates) and ② detailed scheduling.
In particular, reporting to the manager is quite important for both ① and ②.
Figure 1-7-6 shows relationships between plans and reports.

Figure 1-7-6
Relationships between plans and reports

	The middle level	The middle level	The lower level
Plans	• A plan covering a project as a whole is draw up	• Process plans for each sub-system are created	• Work schedules for each workforce member are created.
Reports	• Work completion report	• Process completion reports	• Work reports • Weekly work reports

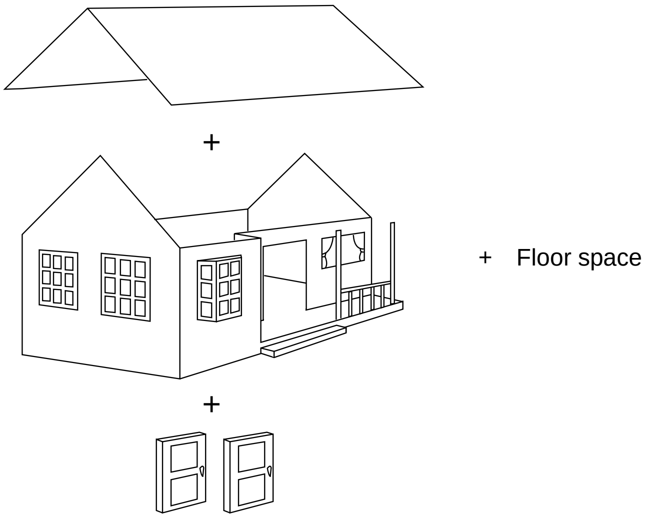
1.7.4 Software Productivity

To evaluate software productivity, the scale of software must allow evaluation. Software development involves various outputs depending on each process, such as proposals, requirement specifications, design specifications, program specifications, and source programs. However, many of them are created by handicraft-like work, depending largely on human experience and sense. Therefore, cost estimates have also depended mostly on experience and sense. To improve such situations, software engineering was proposed. Then, several cost-estimating methods were proposed, and some of them are now in use.

(1) Estimation Outline

For example, Figure 1-7-7 shows an example of estimation when building a house.

Figure 1-7-7
A house cost estimation example



House cost estimation is comparatively easy

Roofs, doors and windows are all visible. Therefore, if an outline is designed, the cost can be estimated (based on the floor space and other factors).
However, files, databases and programs are all invisible. Therefore, estimates in detailed design phases become greatly different from those in the basic planning phase. Consequently, the estimation of system development should be conducted in the various phases described is the following:

1. In the basic planning phase (when systematization is planned)
2. In the external design phase (when the partitioning to sub-systems is carried out)
3. In the internal design phase (when programs are designed).

(2) Ways of Making the Estimation

There are the following methods to estimate development scales:

- Estimation based on data in the past
- Estimation based on the number of lines of code
- Standard task method
- FP (function point) method
- Various estimation models (COCOMO model, etc.)

In the following, each of the methods mentioned above is briefly described:

① Estimation based on data in the past

In this method, estimates of a system to be developed are derived based on actual data of similar systems developed in the past. There are two ways to make the estimation.

- The entire system development process is partitioned to some steps, and estimates are derived based on actual data for similar work.
- The system is partitioned into some program modules, and estimates are derived based on actual data for similar program modules.

<Characteristics>

- With similar systems in the past, basic errors need hardly be included.
- The estimation task is comparatively simple.
- The amount of error in the estimates becomes larger if an appropriate past system is not selected for the estimation.
- The application of the method is impossible if there has been no similar system in the past.

② LOC-based method

The LOC-based method is most frequently used as a method to estimate development size. With this method, a development size is estimated with the lines of code (for example, XXXX kilo COBOL-equivalent LOC), and, based on the data, amounts of necessary resources are estimated.

<Procedures>

1. A system is expressed as a set of program modules

   System functions are partitioned into program modules, with relationships among them indicated by a structural block diagram or other means.

2. The size of each program is evaluated

   The number of LOC in each program module in the diagram is estimated. Then, the total number of LOC is calculated.

3. Manpower for all the programs required is evaluated
   The total number of LOC is converted to the amount of manpower, such as data of man-months (the number of persons required times the number of months required). For example, if a system development requires 2 years' efforts by 20 persons, the manpower is 20 persons × 24 months = 480 man-months.

4. Evaluations on a process basis

   An amount of manpower is allocated to each process, such as the basic planning and various designs, with an allocation percentage decided based on past data.

5. Evaluation of indirect manpower

   A weight for manpower for SE works, such as system analysis and design, and a weight for manpower for administrative work, is decided.

6. Total manpower is estimated

   The total manpower is calculated by aggregating manpower data for each process.

<Characteristics>

- It offers the most typical method.
- If clear standards to estimate program LOC and to convert them to the amount of manpower exist, the calculations involved are relatively simple.
- It is a prerequisite that an outline of the functions of the programs to be developed can be grasped.

③ Standard task-based method

With the standard task-based method, work is broken down on an output basis or on a processing basis with WBS (Work Breakdown Structure). Then, detailed estimation is made for each unit, and resultant estimates are accumulated in a bottom-up way. Refer to 1.1.2 for WBS.

<Procedures>

1. Checking outputs and work required

   A system is broken down into a hierarchical structure based on WBS, and all of the outputs to be developed in the project are listed. Then, all work required to generate these outputs is picked up.

2. The size of each work unit is converted to an amount of manpower

   The amount of manpower required for each work unit picked up is estimated with certain standards, such as actual data for standard work in the past.

3. The aggregation of total manpower

   The amount of manpower estimated for each work unit is totaled.

<Characteristics>

- With this method, estimation is made after work is broken down into the detailed output level or processing level, then the estimates are accumulated in a bottom-up way. Therefore, grounds for estimates are made clear.
- If a difference is generated, the identification of the cause is easy.
- Actual data for standard work is required. In addition, the estimation work requires much effort.

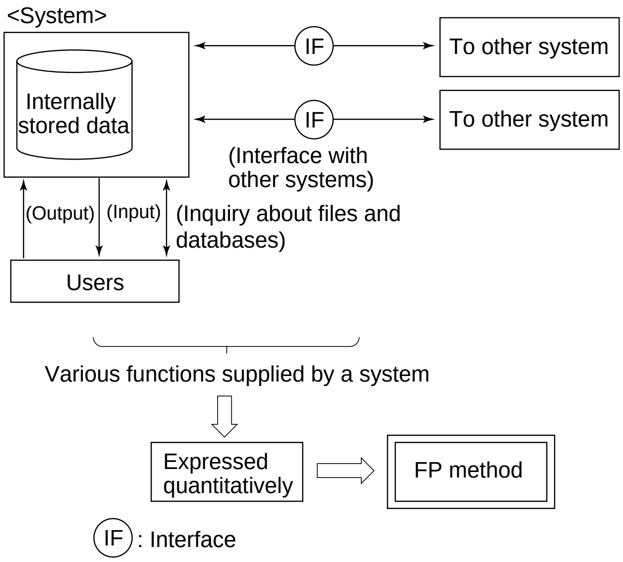
④ FP (Function Point) method

With the FP (Function Point) method, each function to be included in the system is expressed quantitatively with a certain method, and thus quantitatively expressed data is used as a measure for estimation (see Figure 1-7-8).

This method is fundamentally different from the three methods ① to ③ described above in using each function supplied to customers as the unit of measurement (or this method is said to be a customer-oriented estimation method). For example, the following functions utilized by users are

selected as the units to be expressed quantitatively.
<Units to be used as standards>
- Inputs
- Outputs
- Files and databases (internally stored data)

Figure 1-7-8
An FP method example



- Inquiries about files and databases
- Interfaces with other systems

<Procedures>
1. Checking functions (the "units to be used as standards" described above) supplied by a system
2. Functions selected in Item 1, above, are classified into categories of "simple," "average" or "complex." Then, a weight is given to each of the categories based on certain standards.
3. Values given in Item 2 above are aggregated (thus derived data constitutes FP before an adjustment is made).
4. System-specific coefficients are derived depending on the characteristics of the targeted system.
5. A final FP is calculated by multiplying data from Item 3, above, by data from Item 4, above.
6. The FP value is converted to the amount of project manpower.

<Characteristics >
- The data is easily understandable to users, because estimation is made for items visible to users.
- The adjustment is made based on actual data accumulated in the past. Therefore, the accumulation of data is necessary.
- Standardizing evaluation criteria in applying the estimation method is required.

⑤ COCOMO (COnstruction COst MOdel) model

The COCOMO model, an estimation method proposed by Boehm, is suited for estimation of middle- to large-scale systems.
With the COCOMO model, systems are classified based on the following three modes. Then, for each mode, the total development manpower and development period are calculated from the number of statements expected at the time when the system is handed over to the user.

<Three modes>
- Organization mode (small-scale system development)
- Semi-embedded mode (development of systems for ordinary operations)
- Embedded system (development of large-scale and constraint-abundant systems)

< Characteristics >
- In addition to being used by itself, the method is also used to check estimates in other methods.
- It is necessary to accumulate and analyze actual data that includes development manpower data for various systems.

1.7.5 Development Organization

There are many points that contribute to the success of a system development, including various work management-related items as in the following:

- User active participation (the establishment of a development organization)
- Thorough process management and progress management of each member's work
- Thorough system quality management

A large-scale system development takes a long development time (sometimes up to several years), and requires an enormous amount of money and human resources.

However, it is not permissible that a system development fail in the basic planning or design phase or that, though the development is completed, the quality is too poor to be used in actual operations.

Development work, naturally, is managed by a manager, for example, a project manager. However, each development member's proper management of his or her own work and creation of high quality outputs are important factors to lead system development to success.

The first important thing to lead system development to success is to establish a solid development organization. Making the work progress just as in the system development schedule (the highest-level schedule) created in the basic planning phase is the largest prerequisite to success. Furthermore, whether work proceeds smoothly or not can also be said to be dependent largely on the cooperative work of workforce members.

(1) Organization Styles

What type of development organization styles should be employed depends on the development scale or who constitutes the core of the development. However, participation by the users is indispensable in any such organization. In many successful system developments, user organizations participated in basic planning and external design.

Furthermore, system developments are frequently made in cooperation with external development companies. For example, the work up to design phase is carried out internally, and programming work is sub-contracted to other companies.

As examples of development organizations, Figure 1-7-9 shows one for a small-scale project, while Figure 1-7-10 shows one for a large-scale project.

Figure 1-7-9
A small-scale development organization example

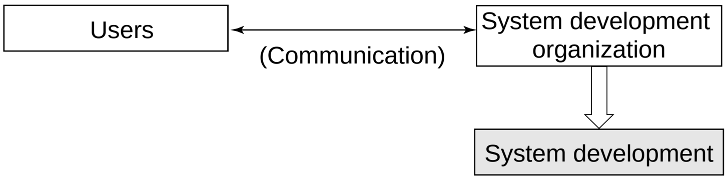
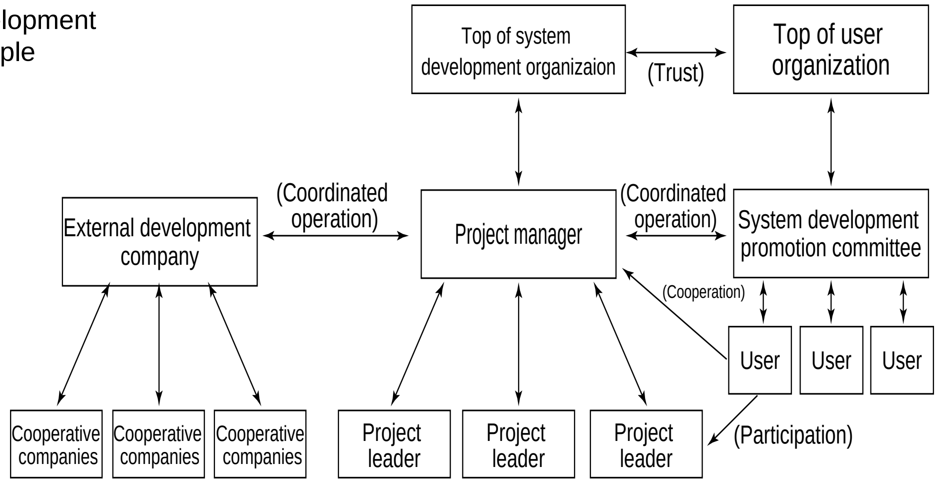


Figure 1-7-10
A large-scale development organization example

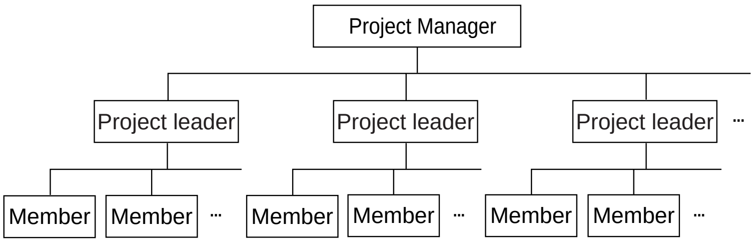


(2) Development Organization

The development organization receives systematization requests from users, and conducts work for basic planning, various designs, programming and various tests. Recently, in many cases, internal project teams do the work of basic planning and various designs, and programming and tests are commissioned to external software development companies. However, the development organization ordinarily conducts attesting work after tests are completed.

Figure 1-7-11 shows a development organization example.

Figure 1-7-11 A development organization example (hierarchical team)



① Types of the development organization

Many system developments are conducted as projects. The definition by NASA (National Aeronautics and Space Administration) of the project is "Tasks that are taken on in many organizations last for one to five years and are related to each other." In other words, the project indicates an organization with definite objectives lasting a limited time period.

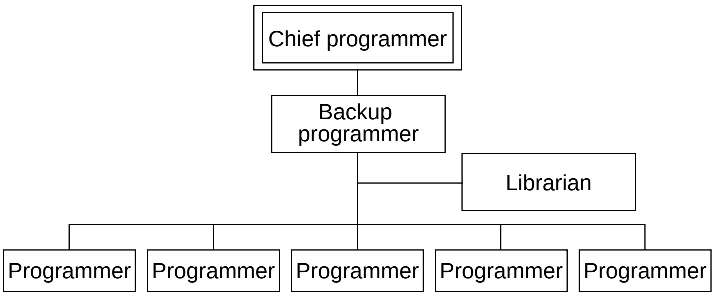
There are three typical types of project teams.

- Chief programmer team
- Specialist team
- Hierarchical team

a. Chief programmer team

The chief programmer team is a project team including the relatively small number of up to ten members, where the chief programmer with full team responsibility exercises leadership in assigning work to each member clearly and increasing productivity and quality.

Figure 1-7-12
A chief programmer team example



The backup programmer: works as an assistant to the chief programmer.
The Librarian: performs document management and administrative work other than the actual development work.

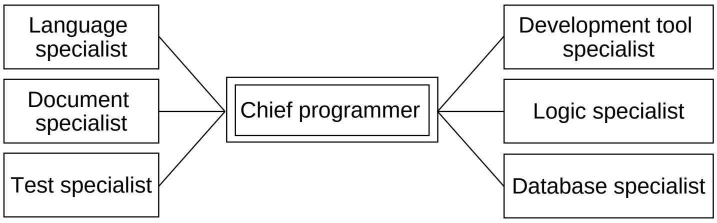
< Characteristics >

- Comparatively small-scale projects are likely to adopt this type of team organization.
- The most significant characteristics of this organization is the existence of the backup programmer and librarian.
- It is suitable for bringing up leaders (burden carried by the chief programmer is heavy).
- It tends to cause deterioration in programmers' morale.

b. Specialist team

The specialist team is a modified type of the chief programmer team, and is composed of a chief programmer and several technical specialists.

Figure 1-7-13
A specialist team example



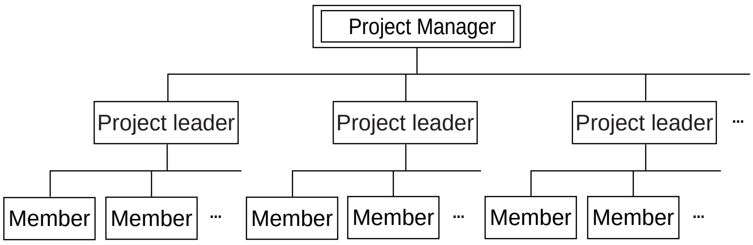
<Characteristics>

- The chief programmer creates all the programs.
- Technical specialists in charge of special areas (such as development tools, tests, documents, databases, etc.) help in the chief programmer's work, extending the programmer's ability to the maximum possible.
- It is essential that the members have high-level skills.

c. Hierarchical team

The hierarchical team is composed of a project manger, several project managers and workforce members.

Figure 1-7-14
A hierarchical team example



<Characteristics>

- This type of team organization is the most widely used in Japan.
- It is adopted in relatively large-scale system developments.
- Communication becomes less adequate, compared to the chief programmer team.

② Team members' roles

a. Person in charge of development organization (project manager)

In many cases, the person in charge of a development organization becomes a project manager. The manager, in an important position, is fully responsible for the development project.

The project manager must have not only high-level information technology skills, but also project-managing capabilities and planning capabilities. In addition, maintaining proper communication within the company and with parties outside the company is an important role of the manager.

<Roles>

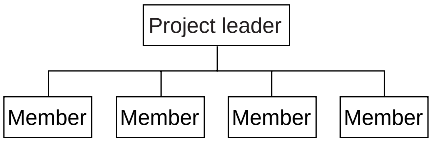
- Planning, drafting plans, execution and evaluation of projects.
- Communication with user and related organization (including parties outside the company)
- Vitalization of project work (including personnel deployment and transfer of power)
- Other management work

b. Project leader

The project leader plays the role of assisting project manager, putting team operations in order, or acts as a go-between workforce members and the project manager.

An organization led by a project leader is called a "sub-project team," which performs actual development work or development-supporting work (technical, for tests, for standardization or others).

Figure 1-7-15
A sub-project team example



- Development work on a sub-system basis is performed.
- Development support work is performed.

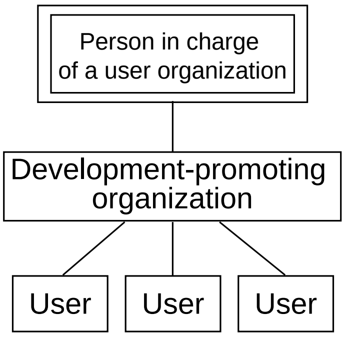
c. Member

Instructed by a project leader, members perform actual development work (designing, programming, etc.) or development-supporting work.

(3) User Organization

System development is conducted at the request of the user organization, and the user organization uses the developed system. Therefore, though the development organization performs system development, system development cannot be successful without the user organization's cooperation.
Figure 1-7-16 shows a simplified organizational structure of a user organization.

Figure 1-7-16
A simplified organizational structure of a user organization


Person in charge of a user organization
Development-promoting organization
User
User
User

① Person in charge of a user organization

The person in charge of a user organization has the greatest power in all phases from budgetary aspects to the promotion of the actual system development project. For the person in charge of a user organization, it is also requested that he or she make the utmost effort to increase the effects-to-investment ratio by executing various plans (such as holding training courses) as leader.

② Development-promoting organization

The development-promoting organization is organized with responsible persons (those in a managerial position) in a user organization as the core, and gives approval to outputs from system development. However, it is not concerned in system details.

③ User

Users actually utilize systems. So, users should actively participate in system development and give advice to various operations. Therefore, user participants should be well familiar with business processes.

1.8 Software Packages

The software package is software product used by more than one user. Today, software package business is thriving, and a variety of package products are available, so they should be used effectively as well as intensively. Here, the entire aspect of the software package is described.

1.8.1 Outline of Software Packages

Roles of software packages and their usefulness are described.

(1) Significance and Roles of Software Packages

There are the following differences between the software package and individual pieces of software:

- The software package places more weight on common parts of work.
- Individual pieces of software include differences existing in work as well.

In other words, software created for general use is said to be the software package.

Recently, it has been said that the amount of backlog (development work that cannot be started) has increased. Statistically, there is reportedly an average of a two- to three-year backlog. The following causes are considered as causing the backlog:

- The system development organization cannot keep up with increase in demands for system development.
- System development productivity is low (development techniques and development-supporting tools are insufficient).
- The supply of system development engineers too low and their skill levels are insufficient (due to long-lasting sluggish economic situations).

The software package provides an effective means to solve these problems.

There was a time when the introduction of software packages was received with resistance. Behind the feeling, there was a sense that a system should be ordered and developed individually. However, software packages with high performance-to-cost ratios have become available recently. Some products can meet operation needs even at significantly high levels.

With such a trend, many software makers manufacture and ship software packages, just as manufacturers of industrial products manufacture and ship their products. Now, the software package business is a main business area in the software industry.

(2) Usefulness of Software Packages

A software package is sold as a CD-ROM set and a number of manuals. The packages cover quite vast areas, including those for system development engineers and those for use by ordinary users.

In addition, they offer quite high performance. There are a variety of software packages available in Japan and abroad. The benefits brought by these software packages to information systems are immeasurable.

<Use merits>

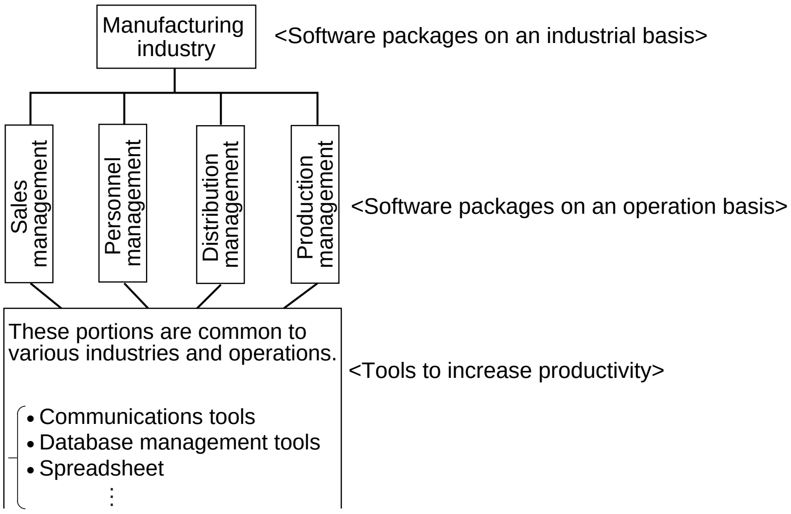
- Development time is shortened.
- Development size is reduced (only portions not covered by software packages are developed).
- Quality is increased (software packages include significantly less bugs, because they are developed under strict quality management).
- Upgrade services are provided (ordinarily, the new version package is available inexpensively, when functions of a software package are expanded).

However, use of software packages in developing a system may cause inconvenience in user interfaces. In such a case, the packages concerned are customized (modification of internal logic or others), but careful handling is necessary because, when the version of a package is updated, operations are not guaranteed for customized versions of the package.

1.8.2 Classification of Software Packages

Software packages in a variety of sizes and types are available. Some are small-sized packages, like communications software provided with electronic mailing functions, and some are large-sized ones like ERP (Enterprise Resource Planning) software.
Operations in a company are considered hierarchical as in Figure 1-8-1. Software packages are provided to support each of the operations.

Figure 1-8-1
Classification of software packages


Manufacturing industry
<Software packages on an industrial basis>
Sales management
Personnel management
Distribution management
Production management
<Software packages on an operation basis>
These portions are common to various industries and operations.
<Tools to increase productivity>
• Communications tools
• Database management tools
• Spreadsheet
⋮

(1) Software Packages on an Industrial Basis

The software package on an industrial basis is software product in view of common characteristics shared by each company in an industry, and products for an entire company operation or for each work unit (manufacturing industry, distribution industry, financial industry, etc.) of an organization are sold.

(2) Software Packages on an Operation Basis

The software package on an operation basis is product in view of operation units allotted for each organization of a company. However, recently, products used shared by each organization are available as well.

① ERP (Enterprise Resource Planning)

ERP (Enterprise Resource Planning) indicates a concept or method to make business activities effective by managing business resources integrated from the viewpoint of using them productively.
Software packages supporting activities to achieve the concept are called ERP (packages for integrated operations). Products from SAP or Oracle Application are well known.

② CRM (Customer Relationship Management)

CRM is an expanded version of ERP. ERP is developed by placing a priority on putting in order processing systems, such as sales, procurement, production, inventory management, services and accounting, as the information infrastructure of a company. CRM combines operations of a center-managed customer database with ERP, enabling marketing activities with care, increasing customer satisfaction levels, eventually firming support from good customers.

③ SFA (Sales Force Automation)

SFA is a concept originating in BPA (Business Process Automation). BPA is the concept of increasing customer satisfaction levels, while achieving reduction in cost, by increasing productivity by automating business processes. With SFA, it is intended that intellectual productivity be increased by automating all of the simple office works and by using the time generated thus for customer services and/or planning and providing proposals. For such activity, use of notebook-type personal computers or PDA (Personal Digital Assistant) is essential.

④ SCM (Supply Chain Management)

SCM, also an expanded version of ERP, is a package to support the achieving of plans provided across organizations using an information infrastructure (resources) put in order. With SCM, it is intended that decision-making be supported to increase the speed of achieving plans.

⑤ CTI (Computer Telephone Integration)

CTI (Computer Telephone Integration) is a software package to support the integration of computers and telephony. The objective is to expand the ways in which computer systems are used. There are many applications, including order-receiving operations, customer inquiry operations, help desks and marketing campaigns, all of which use voice response units.

(3) Tools to Increase Productivity

The tools to increase productivity are software products for work, such as mailing, database management, spreadsheet and document generations, that is for units smaller than operations and common to industries and operations. In some areas, the tool to increase productivity is classified as OA (Office Automation) and design tools to support design work.
<Types>

- Communication tools (file transfer, electronic mailing, groupware, communications software, etc.).
- Database management tools (relational database management system (RDBMS), and various utilities)
- Spreadsheet/document generation tools
- Project management tools
- System development support tools (CASE tools, GUI-building tools, various test tools, etc.).

1.8.3 Production Management Tools

(1) CAD (Computer Aided Design)

CAD is a system with which designers make designs by getting a computer's support through a graphic display. CAD, based on image processing, is progressing into 3-D views from the 2-D views used for drawing.

(2) CAM (Computer Aided Manufacturing)

In a broader sense, CAMs are systems to support manufacturing processes, such as process management, preparation for production, processing and assembly tests. However, ordinarily it is a system to support the generation of data to instruct numerical-control units. Major CAM tools today generate instruction data from outputs from CAD systems.

(3) CAE (Computer Aided Engineering)

CAE is a system to support a series of works included in the designing of products, performance tests and manufacturing. CAE is used to make work efficient by analyzing performances required for products, designing based on analysis results and simulating experimental products on a computer.
The name "the CAD/CAM/CAE system" is sometimes used to cover both design and manufacturing systems.

1.8.4 Examples of Software Packages Effective Use

As described above, use of software packages brings businesses big benefits. Here, typical examples of using software packages effectively are described.

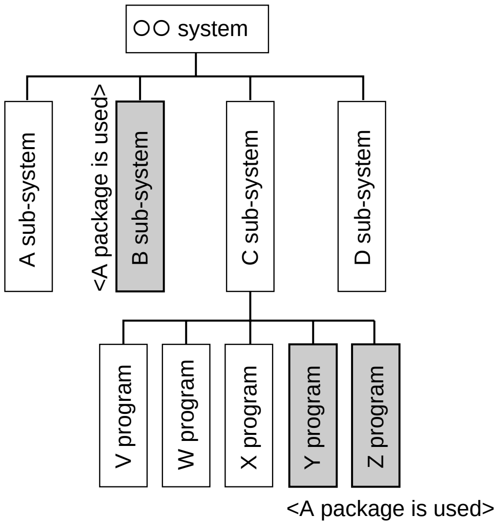
(1) Use of Software Packages as Part of a System

The software package includes not only applications but also program packages. Therefore, use of software packages at the sub-system level and program level is also possible (see Figure 1-8-2). Appropriate setting or slight modification of parameters allows many software packages to be used for these purposes. Therefore, software packages should be used as much as possible.

Recently, class libraries in object-oriented programming have been combined into software packages. Therefore, cases have appeared where only individually necessitated portions have been created and libraries are used for the remaining portions.

Figure 1-8-2

A system-embedding example



(2) Increase in System Development Productivity

Many tools to support system development are available as software packages. Therefore, use of them enables efficient system development.

<Product examples of development support software packages>

- CASE tools: U-CASE, L-CASE, I-CASE, etc.
- GUI/document generation tools
- Project management tools
- Testing/debugging tools

(3) Promotion of Office Automation (OA)

Businesses have introduced software packages to promote office automation with the intention to include the following:

- Making all data be managed as databases.
- Managing the flows of operations-related information, such as electronic mailing and workflow
- Schedule management

RDBMS is mostly used as databases, while groupware (software to support work performed as a group) is widely used to promote OA promotion.

(4) Use as Business

In the software industry, the business of a significantly large number of companies is to develop and sell software packages. If a company can develop and sell products meeting customers' needs, a tremendous growth is expected for the company. However, system development for software packages is slightly different from that for individual software development. Figure 1-8-3 shows differences between software package development and individual software development.

Figure 1-8-3 Comparison between software package development and individual software development

	Software package development	Individual software development
Basic planning	Based on marketing, work to develop software needed in society is performed. In some cases, a new software package is generated by integrating existing software packages.	Through meetings with users, analysis and definition are made to systematize what a user needs.
Test/shipment	The software is installed in various types of computers so thorough quality management is conducted to eliminate bugs. Furthermore, customization is performed for customers, then the contents are recorded on CD-ROM or other media, which is shipped.	Ordinary tests, including unit tests to operation tests, are conducted, then the system is handed over to the user.
Maintenance	The version of a product is updated, considering market trends and relations with other software that works with the product.	Maintenance work is conducted based on user's requests or change of situation.

From now on, it is expected that, in addition to the development of new software packages, many integrated software packages integrating existing software packages will be produced.

1.8.5 Groupware

To support human work, computers' human interfaces have been improved and advanced. However, human work is mostly conducted as group work. System development is a typical example.

With the advance in network technology in recent years, more than one computer is connected through networks and information exchange between them has become possible. Groupware is a system to support joint work through networks that have thus been developed.

(1) Communications

Communications functions are the most important functions of groupware. Communication within a group can be maintained through electronic mail and electronic bulletin boards without convening a meeting somewhere.

(2) Data sharing

Sharing data makes it possible to consider computers a common workplace. Compiling a document in cooperation with a number of persons offers a typical example. The writers who are located at different places from each other can share a document file placed on a pre-determined server.

In addition, since groupware allows centralized management of data, it becomes unnecessary for different sections to keep the same data.

(3) Schedule Management

The function of centrally managing group members' schedules, provided with groupware, enables schedules of all the group members, for example 10 members, to be checked instantly without need of checking separately. The function eliminates the work needed to make an adjustment of meeting time.

Achieving these functions requires document processing functions like Japanese word processing, and database functions in addition to network functions. In some cases, these functions are realized by providing interfaces with existing products. From now on, more advanced support systems taking in the concept of PDM (Product Data Management), managing various data centrally, will come into use.

1.8.6 OA Tools

Personal computers have so rapidly and widely been used in businesses and homes because software allowing easy use by even inexperienced persons has been installed in them. Formerly, computers were handled only by information processing engineers versed in information technology. However, with advances in information technology, processing functions have been enriched, and computers have now become products as familiar as electronic home appliances.
In particular, the following three tools play principal roles in office automation (OA).

- Word processor
- Spreadsheet software
- Presentation software

(1) Word Processor

The largest factor that has enabled wide spread use of personal computers is the introduction of word processors. The introduction of word processor functions has transformed computers from "data processing machines that only specialists can handle" to "familiar information processing machines."

① Formatting

The most important objective of using the word processor is to generate documents, through inputting data, in which formatting functions are used to decide layouts of documents to be printed.
Formatting mostly specifies paper sizes used and appearances of printed texts.

- Setting the type of papers used
  The size of papers used is specified.
- Setting a printing direction
  Whether printing in the longitudinal or horizontal direction is specified.
- Setting the number of characters to be printed per line
  The number of characters to be printed per line is specified, which determines space between adjacent characters printed.
- Setting the number of lines to be printed per page
  The number of lines to be printed per page is specified, which determines space between adjacent lines printed.

With the word processor, it is possible to specify a set of formatting and to print with that formatting to see how the printed appearances and image look, then to make re-specifications.

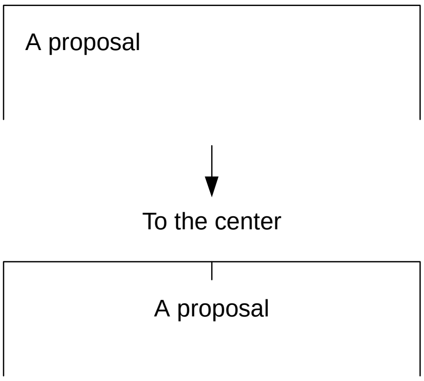
② Editor functions

When sentences directly written on paper are corrected, the portions to be corrected are erased with an eraser, then new words or phrases are over-written. However, sentences generated with the word processor exist on the main memory of the computer. Therefore, correcting them is quite easy.
In addition to the correcting function, the word processor offers various editing and processing functions.

a. Centering

The word processor provides the ability to place a title of sentences or other character sequence at a horizontally or longitudinally central position based on the number of characters per line that is set in the formatting. This function is called "centering."

Figure 1-8-4
A centering example



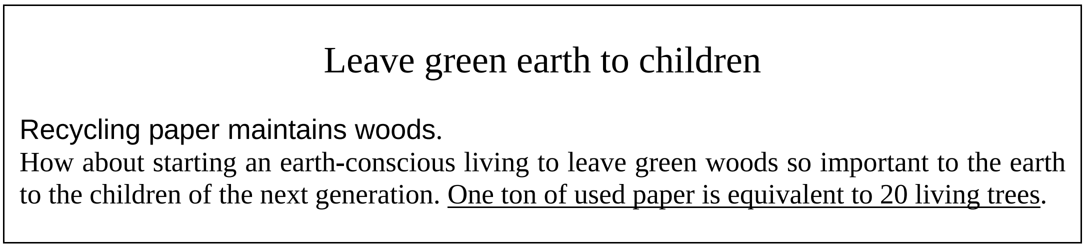
b. Moving character strings to the right-most or left-most position

The word processor offers the ability to move character strings to the right-most or left-most position. Without this function, it is necessary to input "space" (through the space key) to move the strings to the right-most or to use the delete key to move the character string to the left-most position of a page.

c. Character decoration

The word processor allows character strings to be underlined, or a Gothic font to be used for them, or the font for a title to be altered, or an italic font to be used, when printed. Such functions are enabled because font patterns are stored in the software used.

Figure 1-8-5 A character decoration example



Leave green earth to children

Recycling paper maintains woods.
How about starting an earth-conscious living to leave green woods so important to the earth to the children of the next generation. One ton of used paper is equivalent to 20 living trees.

d. Copying and move

Sentences generated with the word processor are stored on the main memory. Therefore, phrases or character strings in these sentences can be easily copied or moved to different locations.

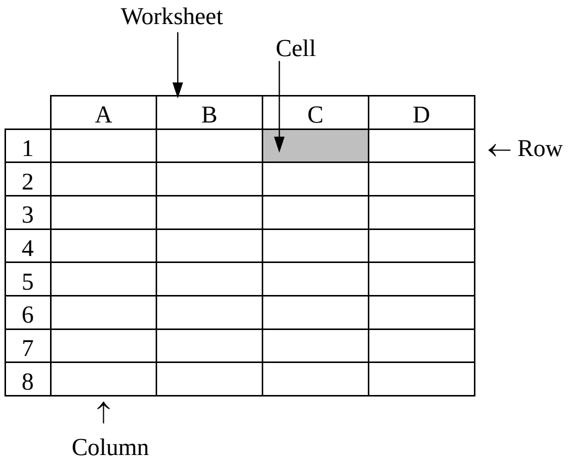
(2) Spreadsheet Program

The spreadsheet program is to generate tables (with two dimensions) or to aggregate data. It is also possible to rearrange aggregated data or express them as graphs.

① Worksheet and cell

The table displayed on a screen is called a worksheet, while each unit of rectangular space is called a cell. In a worksheet, a set of cells arranged horizontally on the same level is called a row, while a set of cells arranged longitudinally on the same line is called a column. A row is identified with numerals, like 1,2,3,---, and a column with alphabetical characters, like A,B,C,---.

Figure 1-8-6
Worksheet and cell



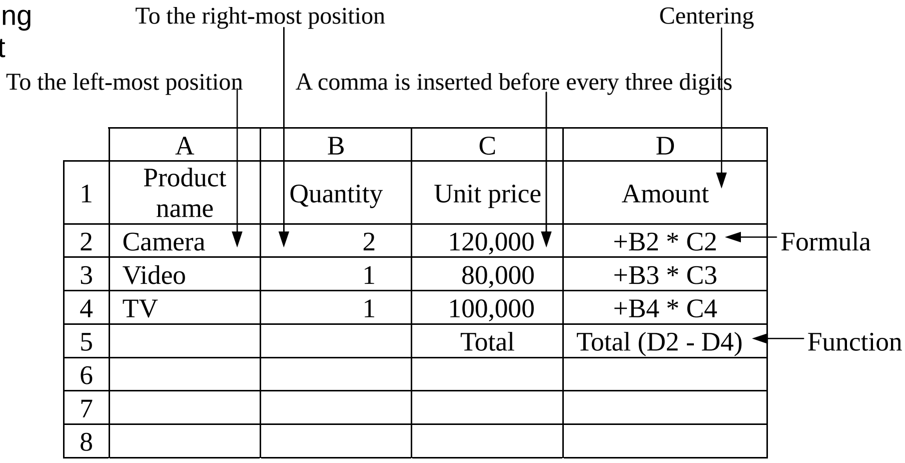
Data is inputted to each cell. In addition to numerical data and characters, a formula or a function can also be inputted to a cell (see Figure 1-8-7). Furthermore, each cell can have its own formatting (displaying data to the right-most or left-most position), and data in a cell can be displayed in its own formatting (see Figure 1-8-8).

The spreadsheet program identifies with the initial character, whether an entered character string indicates data or a formula. If the initial character is one of the symbols to specify a formula, such as '+,' '@,' '=,' etc. (each program uses different symbols for the purpose), the characters which follow are considered as constituting a formula.

In addition, it allows the use of functions in place of formula. For example, in Figure 1-8-7, "Total (D2 - D4)" indicates a function to aggregate data from cell D2 to cell D4. In some programs, the expression of "Total (D2, D4)" is used instead of "Total (D2 - D4)."

Figure 1-8-7

An example: Inputting data to a worksheet


To the right-most position
Centering
To the left-most position
A comma is inserted before every three digits
Formula
Function

	A	B	C	D
1	Product name	Quantity	Unit price	Amount
2	Camera	2	120,000	+B2 * C2
3	Video	1	80,000	+B3 * C3
4	TV	1	100,000	+B4 * C4
5			Total	Total (D2 - D4)
6				
7				
8				

Figure 1-8-8

A displayed worksheet example

	A	B	C	D
1	Product name	Quantity	Unit price	Amount
2	Camera	2	120,000	240,000
3	Video	1	80,000	80,000
4	TV	1	100,000	100,000
5			Total	420,000
6				
7				
8				

② Recalculation function

If data on a worksheet is modified, outputs of formulas or functions that use this data will vary. The results will also vary when a column and/or row is inserted or deleted. This function is called the "recalculation function," and is an important characteristic of the spreadsheet program.

With this function, it is possible to run a simulation by changing data variously.

③ Editor functions

a. Copying or moving data, and inserting or deleting data

Copying data is done in the same way as in the word processor. First, the area to be copied is specified. Then, the area where the data is to be copied is specified. After that, the copying is executed. Moving data can be executed in the same way. Moving differs from copying in that, with moving, original data is lost from the original area.

Both copying and moving can be executed for data in a cell or for an area specified.

On the other hand, insertion and deletion can be executed only for a row or column.

b. Automatic adjustment of cell numbers

The superiority of the spreadsheet program is that it allows the data of cells affected by an editing operation to be automatically adjusted. This is called an automatic adjustment function of cell numbers.

For example, assume that there are many cells in which the same calculation is performed. With this editing function, it is possible that the formula is inputted to only one of these cells, with the formula copied to the remaining cells. Data calculated with the same formula is entered for each of these cells.

c. Rearrangement of data

Data inputted not considering any order can be rearranged on a row basis, for example, in the order of larger value data placed before smaller, in the order of smaller value data placed before larger, in the order of Japanese "kana" characters, alphabetical order, or the ascending or descending order of JIS.

④ Formula and function

The spreadsheet program allows various calculations other than the four fundamental rules of arithmetic to be made. In addition, functions are embedded to support complex calculations.
Various functions depending on applications are provided, including typical ones listed as follows:

- Arithmetic functions: calculations of rounding off, absolute values and others
- Logic functions: AND, OR and others
- Statistical functions: calculations of total, average, maximum and minimum values, and others
- Dating functions: displaying date data, time calculation, and others
- Financial functions: calculations of profit ratios, investment, depreciation expenses, and others

(3) Presentation Software

Formerly, major presentation tools were slide projectors and overhead projectors. However, recently they have been replaced by software packages called presentation software that offers functions which are much more than those of projectors. Present software offers animation functions and sound functions as well, and enables effective presentation by bringing into screens results of spreadsheet programs and database software. Typical presentation software includes PowerPoint from Microsoft, Freelance from Lotus and Appleworks for Macintosh.

Exercises for No.2 Chapter1(System Development)

Q1 **Which of the following is the most appropriate as the characteristic of information technology use in businesses today?**

a. In view of profitability, businesses have increased the degree of manufacturing internally without outsourcing systems or operations to external companies.

b. So-called "end-user computing," in which users by themselves build systems and have access to or process information for their own applications, is gaining support.

c. To cut costs or to shorten software development time, applications have been developed based on orders received rather than by use of package software.

d. Widely spreading use of networks reduces the range affected by a system failure, making security management easier.

Q2 **Which of the following is the most appropriate as the explanation of the waterfall model, a system development methodology?**

a. An application is broken down into sub-units. Then, each of them is designed and manufactured sequentially one after another.

b. System development is made in the order of processes, without work being returned to a higher level process.

c. An operational experimental product is generated, and checking requirement specifications and evaluation are made in an early phase.

d. Development time is shortened by participation by users, by development by a fewer number of engineers and by effective use of development tools.

Q3 **Which of the following is the most appropriate as the description of prototyping, a system development technique?**

a. Work is performed in the order of basic planning, external design, internal design, program design, programming and test. Therefore, with the technique, a perspective of the work as a whole is gained, making the determination of schedules and allocation of resources easier.

b. An experimental product is produced in an early phase of system development, enabling the removing of ambiguities and differences in recognition between the user and development organizations.

c. The software is classified into software whose specifications are fixed and do not require modification, and software whose specifications require modification. Then, for the software whose specifications require modification, the process of development, reconsideration and modification is repeated.

d. A large-scale application is broken down into sub-units, each of which is highly independent. Then, for each sub-unit, the process of design, programming and test is repeated, gradually expanding the development area.

Q4 **Each of the following sentences describes work for one of the system development processes. Which gives the correct order of the development processes?**

a. **Present problems are investigated and analyzed, then the requirements for a targeted system are defined.**

b. **Functions required for building a system are partitioned into programs to make process flow clearer.**

c. **Detailed processing procedures are designed, coded, and corrected.**

d. **Tests are conducted.**

e. **The structured design of each program is made based on internal design documents.**

f. **Based on requirements for a system, functions necessary for the system are defined.**

a. a–f–b–c–e–d b. a–f–b–e–c–d
c. a–f–e–b–c–d d. a–f–e–c–b–d

Q5 **Which of the following is the most appropriate as the explanation of reverse software engineering?**

a. Design specifications are produced from implemented software. Then, software is developed based on specifications produced thus.

b. Software is designed in the order of output, processing and input.

c. Functions having been implemented with software are achieved with hardware.

d. A development language and development tools are selected depending on the processing characteristics of the software.

Q6 **With a structured analysis method, data and functional flows are expressed with symbols which respectively indicate data flows, processing (functions), data storage, and externals (data sources and data destinations). Which of the following is the method?**

a. DFD b. ERD c. NS chart
d. State transition diagram e. Warnier diagram

Q7 **Which of the following is the diagram that is used in structured programming and expresses the entire structure of a program in the form of a hierarchical structure?**

a. NS chart b. PERT diagram c. State transition diagram d. Bubble chart

Q8 **What is it called to make details of object implementations invisible by putting together data and methods in object-oriented programming?**

a. Instance b. Encapsulation
c. Clustering d. Abstraction

Q9 Which of the following is most appropriate as the description of object-oriented programming?

a. Data exchange among objects is executed through instances.

b. The object indicates descriptions of class characteristics.

c. Encapsulation indicates putting together classes as a library.

d. A class can inherit methods from its parent class.

Q10 Which is/are inappropriate item(s) to be considered in screen designs of external or internal designs?

a. In screen transitions, a direct selection method intended for experienced users should be used instead of step-by-step selection by use of menus.

b. Input items on screens should be enclosed with □ or [ ] to make it clear that the spaces are for input fields.

c. Screen layouts should be designed so that items to be referred to can be arranged from left to right or top to bottom.

d. To complete a processing operation, designs should be made so that the suspension of data inputting and returning to a previous screen cannot be allowed.

e. Standardized screen layouts, for example, unified places for displaying titles and messages, should be used.

Q11 Which of the following is the most appropriate as the description of code design and code management?

a. Codes inevitably vary, so it is important to put code books in order and to manage them.

b. It is desirable that codes are understandable by themselves. Therefore, the use of longer code is better.

c. Numerals should mainly be used as codes, and Chinese characters should not be used.

d. Codes should be assigned to make data classification easier, but the addition or expansion of codes should not be considered.

Q12 Assume that there is a four-numeral code $N_1N_2N_3C$. The right-most C indicates the check digit, which is calculated with the formula below.

$C=\mathrm{mod}\,(\,(N_1 \times 3 + N_2 \times 2 + N_3 \times 1\,),\,10)$

Here, mod (a, b) indicates the remainder of a/b.

Then, what is the figure for □ in the following four figure code "81□6?"

a. 0 b. 2 c. 4 d. 6 e. 8

Q13 **Which of the following is the most appropriate as the explanation of re-entrant programs?**

a. The program gives correct results even if it is re-executed without being re-loaded after being once executed.

b. Placed at any addresses on a real memory, the program can be executed.

c. The program is partitioned into more than one segment, and can be loaded and executed on a segment basis.

d. Even if more than one task executes the program in parallel, correct results can be obtained.

Q14 **Which of the following is the most appropriate as the explanation of structured programming, an important factor in the logic design of modules in program generation?**

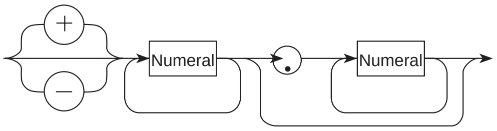
a. It means that indenting rules for coding are provided to make source lists more easily readable.

b. It means that notes are effectively used to enable understanding of process methods just by reading them.

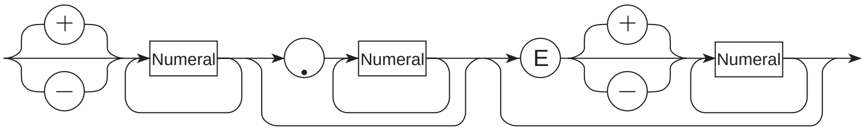
c. It is described with the three basic structures of "sequential," "selection" and "repeating."

d. A standard size of a module should be 50 to 150 steps.

Q15 **There is a syntax which is described with the following syntactic structure. Expressions like -100, 5.3, and +13.07 meets the syntax.**



When this description method is used, which of the following numeric expressions meet the syntax specified by the following syntactic diagram?



a. 5.2E – 07 b. + 1.E4 c. – .9 d. 9.89E

Q16 **Which of the following gives the correct answer for the formula described below with reverse Polish notation? Here, xy– indicates that y is subtracted from x, while xy÷ indicates the quotient from an xy÷ operation.**

Formula: 4 3 5 – ÷

a. –2 b. –0.2 c. 0.2 d. 0.5 e. 5

Q17 **Which of the following is correct as the explanation of Java?**

a. It is a communications protocol used on the Internet.

b. It is a browser for the Internet.

c. It is an object-oriented programming language.

d. It is a coding technique for statistic color image data.

Q18 **Which of the following is the test case generation method used in the white box test?**

a. The cause-effect graph
b. The experimental design method
c. Condition coverage
d. Equivalence partitioning

Q19 **Which of the following is the most appropriate test to be used in boundary value analysis?**

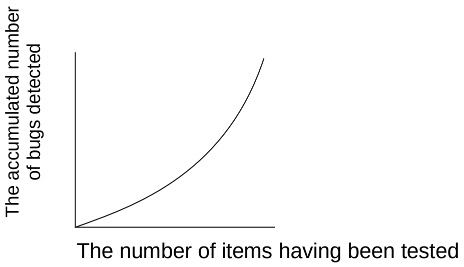
a. The minimum and maximum values
b. The minimum and maximum values, and those adding 1 to each of the values
c. The minimum value and that adding 1 to the value
d. The maximum value and that adding 1 to the value

Q20 **To maintain the quality of design documents, reviews are conducted in each phase of development. Which of the following is the most appropriate as the explanation of the inspection, a review technique?**

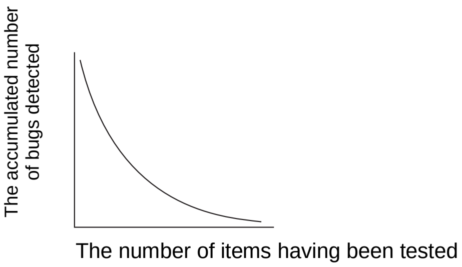
a. Reviews as a whole are conducted with each participant playing the responsible role in turn.
b. For reviewing, part of the target software is experimentally produced and is actually executed.
c. Items to be reviewed are selected in advance. Then, documents are reviewed quickly by checking an item at a time.
d. The person who authored the design documents to be reviewed chairs the review meeting.

Q21 **The relationship between the number of items having been tested and the accumulated number of bugs is used as a management item to check quality status of the test process. Which of the following graphs indicates that the quality is becoming stabilized?**

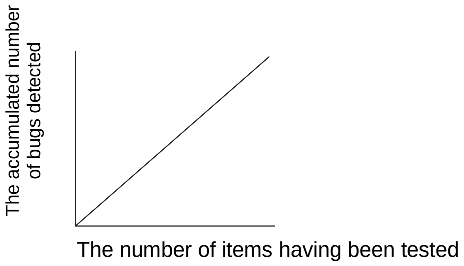
a.



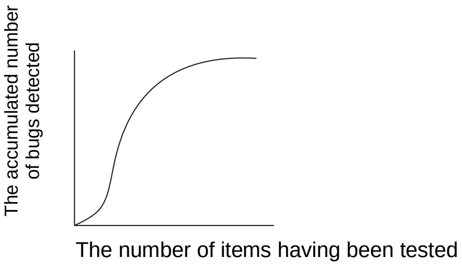
b.



c.



d.



Q22 **Which of the following is classified as a function of upstream CASE tools?**

a. Source program-analyzing function
b. System analysis and definition function
c. Test-supporting function
d. Automatic program generation function
e. Project management function

Q23 **Software quality characteristics include reliability, usability, maintainability and portability. Then, which of the following explains reliability?**

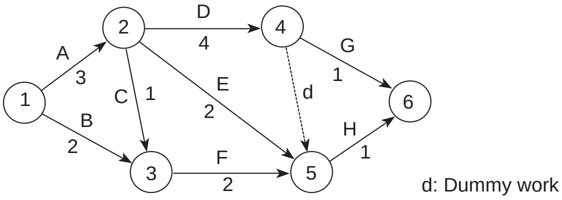
a. It indicates how easily operations can be mastered.

b. It indicates whether functions required for software can always be maintained normally under designated conditions.

c. It indicates the degree of modification which becomes necessary when software is to be used in a different computer environment.

d. It indicates the degree of ease in which modification requests from users or troubles can be handled.

Q24 **There is a project described with the following PERT diagram. The symbol above each arrow indicates a work name, while the number indicates the number of days needed for the work. What is the earliest starting date for the work H? Assume that the project start day is day 0.**



a. 4 b. 5 c. 6 d. 7 e. 8

Q25 **Which of the following is the most appropriate as the description of the characteristics of the chief program team system compared with the hierarchical system?**

a. The amount of burden on the leader is heavier in the hierarchical system than in the chief program team system.

b. The hierarchical system is less appropriate for large-scale systems than the chief program team system.

c. Communications within a team is easier in the hierarchical system than in the chief program team system.

d. The chief program team system does not need a back-up programmer.

e. The chief program team system needs a librarian.

2 System Operations and Maintenance

Chapter Objectives

It is called "operations" to actually run a developed system, while the work to handle problems in operations is called "maintenance." These are works required continuously until the system is scrapped, imposing a heavy burden on information processing engineers.
In this chapter, an outline of operations and maintenance is given to consider how systems should be developed, and to reduce the burden of operations and maintenance as much as possible.

① An outline of operations and the contents of each management item are taught.
② An outline of maintenance, and types and contents of maintenance work are taught.

Introduction

Operations and maintenance of a system constitute the final phase of a software life cycle. They concern the phase where a developed system is actually run, accounting for the largest portion of the software life cycle. Therefore, a big difference is seen between a system that offers effective operations and maintenance and one that offers less effective operations and maintenance.

To achieve effective operations and maintenance, it is too late to make plans for them after a system has been developed. Measures for operations and maintenance should be included in system development efforts.

In this chapter, contents of operations and maintenance are taught aiming at enabling the preparation of effective operations and maintenance plans.

2.1 System Operations

System operations are performed based on operation service standards. Management items required for system operations include:

- Resource management
- Problem management
- Facility management
- Security management
- Performance management
- Cost management

2.1.1 Resource Management

Resource management occupies an important place among the operation management items. To use system resources effectively, it is necessary to have correct knowledge about the resources required for operations.

System resources include:

- Hardware resources
- Software resources
- Data resources
- Network resources

(1) Hardware Resource Management

Hardware resource management indicates the management of computers and their peripheral devices. Hardware resources must be maintained properly.

Actually, operation managers should confirm that resources are used effectively by checking how hardware equipment is being used. In addition, if unevenness in use is found, the rearrangement of hardware resources must be considered to better distribute the load. The basic concept is to use resources effectively to increase the rate of operation of each hardware device equally.

The life of hardware resources should also be taken into account. Generally, equipment in use for more than a certain period becomes likely to cause problems more frequently. Considering the replacement of devices by checking the problem-generating rate is an important item in managing hardware resources as well.

In hardware resource management, following data are collected and their results should be evaluated and analyzed regularly:

- Response performances
- Processing capabilities (the number of processed items per hour)

(2) Software Resources

Software resource management indicates the management of programs running on a system. Contrary to hardware resources, many parts of software resources are invisible. Therefore, keeping to pre-determined standards is important for management.

① Library management

The items to be conducted in library management include:

- Where libraries (including backup libraries) are physically stored should be clarified.
- The version data in a library should be managed (The co-existence of new and older versions of the same software should be avoided).
- Libraries should be protected (for security and from computer viruses).

② Prevention of illicit use

The following measures should be taken to prevent illicit use of software resources:

- Whether illegitimate copying is made or not should be managed.
- How software resources are used should be managed.

(3) Data Resource Management

Data resource management indicates the management and adjustment of data used in a system from overall organizational viewpoints. Users themselves manage much of the data. However, the objective of data resource management is to manage this data systematically, and to select important data for special management aiming at securing perfection.

In many of the systems today, databases are used. Therefore, database resource management on an operational basis should be covered.

The following should be performed in data resource management:

- Securing perfection
- Ensuring security (prevention of illegitimate use)
- Systematic management of data resources

In addition, data audit, conducted to investigate and analyze methods to carry out data resource management and to perform the management at a higher perfection level, is an important item as well.

(4) Network Resource Management

It is no exaggeration to say that computer systems today are always connected to networks. In network resource management, equipment making up networks, such as CCU (Communication Control Unit, DCE (Data Circuit Terminating Equipment), etc. is managed. Network resource management is basically performed under hardware management. However, as for communications circuits, the handling of re-routing circuits, in addition to backbone circuits, is involved. Therefore, a management organization including telecommunications carriers must be established.

2.1.2 Problem Management

It is desirable that no problem occurs in system operations. However, in actuality there is no system where no problem occurs. Therefore, an important aspect is how speedily a system can be restored after a problem has occurred. Problem management is for measures to be taken when a system problem occurs.

Standard procedures to be taken when a problem occurs are as follows:

- Finding and reporting trouble
- Creation of trouble reports
- Analysis of trouble
- Work to recover from a problem
- System restoration work

After the restoration work is completed, it is necessary to evaluate whether the measures taken were appropriate and to reflect the results to measures to be taken thereafter.

(1) Finding and Reporting Trouble

The earlier a problem is found, the smaller the effect on the system as a whole and the easier the measures to be taken will be. Therefore, to find a problem as early as possible, it is important to take always care about data collected in resource management and to grasp ordinary operating situations.

In addition, the establishment of an organization allowing trouble found to be reported swiftly to the manager in charge of problems is important.

(2) Creation of Trouble Reports

Trouble reports must be produced immediately when a report that trouble has occurred is received. The trouble report has two uses. One is for analyzing the problem and taking measures swiftly as well as correctly; the other is as one of the statistical data utilized to prevent problems in advance.

In addition, at this time, the area to be affected by the problem is identified and notification is made to the sections concerned. In particular, problems that are considered to greatly affect system operations need support for the work of restoration from the problem. Therefore, immediate notification is essential.

(3) Analysis of Trouble

To investigate the causes of a problem, analysis is conducted into the situations where the problem occurred. To investigate the causes, logged data at the time when the problem occurred and dump listings are generally used. Many of the causes of hardware problems are found with these means. However, for software-related problems, finding the causes sometimes takes much time. In such a case, temporary measures may be taken, with a search for the real causes left until later.

The following methods are used as measures, if log and dump listing data are insufficient to find the causes:

- The situation when the problem occurred is re-generated artificially.
- A measure is taken that, if a similar trouble occurs again, allows detailed data to be obtained.

Making the causes of problems clear enables preventing the occurrence of similar problems again.

(4) Work to Recover from a Problem

Based on the causes of problems, the methods to restore the system are determined and restoring operations are performed. The methods are various depending on the problem causes.

① Hardware trouble

- Back-up devices are put in use.
- Problem devices are separated. Then, they are repaired (for example, by the manufactures of these devices).

② Software trouble

- The software is re-activated.
- The older version software is restored in place of the present version software.
- Correction is made for the present software.

③ Data trouble

- Data that caused the problem is removed or modified.
- Roll-back or roll-forward operations are performed.

In addition, keeping trouble reports on how the restoration work was performed enables them to be used as documents to consider measures for similar problems occurring later.

(5) System Restoration Work

A system whose operations are stopped is restored. It is checked to see whether the work to recover from a problem enables the system to function normally. Then, the system is restored with ordinary services offered again.

Depending on the ways used to recover from a problem, the following cases should also be taken into consideration:

① Hardware trouble

The following should be considered, if recovery is made by use of backup hardware:

- Performances in comparison with the main hardware
- Restoration work when the repair of the main hardware is completed

② Software trouble

The following should be considered, if recovery is made by use of an older version of the software:

- Drop in the level of functions available (such as services available)
- The limitation of use in consideration of response capabilities

③ Data trouble

The following item should be considered, if data is corrected:

- Corrected data is consistent with that not corrected.

System restoration work continues until all the functions of a system are fully restored.

2.1.3 Facility Management

To operate a computer system, the facility and equipment of the computer center must be maintained above a certain quality level. As for the installation standards for facilities and equipment, the Economy, Trade and Industry Ministry (formally the International Trade and Industry Ministry) has put out a guideline entitled "Standards for Information System Safety Measures."

In investigating operations (designs) of equipment, the three aspects of reliability, expandability and cost must be taken into account. Concerning expandability, designs must include margins in consideration of recent advancements in technology and variations in external factors.

Facilities to be considered in system operations include the following:

- Power supply-related facilities
- Air conditioning facilities
- Disaster prevention facilities
- Crime prevention facilities
- Storage facilities

(1) Power Supply-Related Facilities

Computer systems cannot be operated without power supply. Therefore, facilities continuously supplying stable power must be provided.

① Primary power supply

Ordinarily, a commercial power supply system is used for the primary power supply. However, a mechanism to secure stable power supply is required. Actually, to handle quality degradation in supply voltage from a commercial power supply system, measures to maintain supply voltage at a constant level, including the use of an AVR (Automatic Voltage Regulator), are taken.

② Non-utility electric facilities

A non-utility electric facility is used as a backup when the primary power supply has a problem (for example, due to a power outage). The facility may be used as the primary power supply if a commercial power supply system is unavailable. However, ordinarily it is used only when a problem occurs. Therefore, inspections to check the storage of power generation fuel and failure of the facility must routinely be conducted.

③ UPS (Uninterrupted Power Supply)

UPS (uninterrupted power supply) is temporarily used from the time when power supply from the primary power supply is stopped to the time when power supply from a non-utility electric facility is started. UPS also plays the role of compensating short breaks in the primary power supply.

④ Others

- Battery
  Ordinarily, batteries are charged for each device while power is being supplied. In some cases, batteries charged with battery chargers or other means are provided separately from the devices.
- Power distribution facilities
  It is also important to regularly inspect distribution boards, breakers, surge protectors and other related equipment.

(2) Air Conditioning Facilities

Most equipment used in a system generates heat in operation. Even though the heat from each device is small, the total amount of heat generated in a computer room, where many devices operate, becomes significant. Ordinarily, temperatures and humidity enabling stable operation are specified for each piece of system equipment. Therefore, providing air conditioning facilities is important to maintain the proper conditions.

There are two types of air conditioning facilities: the centralized type and the distributed type. With the centralized type, a large air conditioner is installed in a computer center, controlling the temperature of the room as a whole. This type of air conditioning facility requires a small installation space. However, once the air conditioning facility fails, the operations of the center as a whole may be stopped. With the distributed type however, an air conditioner is equipped with each device. Therefore, this type of facility offers high expandability and also the merit that danger due to power failure is distributed. However, this type is slightly inferior in the efficiency of facility investment to the centralized type.

Water-cooled cooling equipment may be used for a facility generating lots of heat, such as a large host computer. Today, much equipment generates less heat and can be used in ordinary office environments. Consequently, the use of air conditioning facilities will likely be limited to center facilities.

(3) Disaster Prevention Facilities

Stable operations of a system require the use of disaster prevention facilities provided in case of disasters, such as fire, earthquake, etc. Typical disaster prevention facilities include the following:

① Fire prevention facilities

Fire prevention facilities include automatic fire alarms (the heat-sensing type and the smoke-sensing type) and fire-extinguishing equipment. Sprinkler systems cannot be used as fire-extinguishing equipment in center facilities. Therefore, fire-extinguishing equipment using halide or carbon dioxide is used instead. The type of fire extinguishers for electricity-caused fires is different from that for ordinary fires. Therefore, fire-extinguishing exercises should be conducted routinely.

② Anti-earthquake facilities

As a direct measure for earthquakes, fixing equipment with anchor bolts, or topple-preventing equipment, using floor vibration-absorbing equipment, is used. However, what is feared in earthquakes is secondary fire. To prevent secondary fire, equipment to shut down power supply by sensing vibration is also used.

③ Emergency announcement equipment

When a disaster occurs, the role of emergency announcement equipment announcing correct information is important.

Disaster prevention facilities are not used in ordinary times. Therefore, conducting inspections to check that the facilities work in emergencies without failure is important. At the same time, it is desirable to provide a mechanism (such as a disaster drill) to check measures to be taken in emergencies as well as communications measures and contact routes.

(4) Crime Prevention Facilities

Crime prevention facilities are for protecting systems from man-made menaces such as information stealing and destructive activities. Since man-made menaces are brought in by outside persons, the facilities are, in many cases, for preventing the entrance of such persons.

① Entry/exit control equipment

Entry/exit control equipment allows managing persons to enter and exit from a center facility. To identify whether a person is entitled to enter or depart from a center facility, the authentication of a person is performed by means of a password or passwords, a card for enter/exit, such as a magnetic card or IC card, a fingerprint, a voiceprint, or a retina pattern.

It is also important to manage entry/exit logs (names of persons entering or exiting from a center and their entry/exit times. This measure is effective to prevent man-made menaces by persons who have a genuine right to enter a center).

② Monitor

Monitors are equipped in the vicinity of a center facility to find suspicious persons and to prevent those persons from entering the facility. Monitors are also installed inside a facility to check whether there is a person behaving suspiciously. Formerly, more than one monitor had to be installed, because stationary monitors were used. However, many monitors using a wide lens and having a variety of functions, including direction-moving and zooming functions, have recently become available. Therefore, a sufficient effect can be expected with a small number of monitors installed.

(5) Storage Facilities

Even though the highest level security functions are used to prevent data from being stolen, they are meaningless if backup or some other data are taken out easily. Sufficient care must be taken about facilities to store backup data and other data outputted from a system.

A storage facility is for storing important data. Therefore, the facility must be equipped with disaster prevention functions, such as fireproof and waterproof functions, in addition to security functions provided to prevent stealing.

Basically, a storage facility must be built at a place far apart from its center facility. In addition, entry/exit control must be conducted in the same way as for center facilities.

2.1.4 Security Management

The objectives of security management are to prevent the illicit use of a system and information leakage in system operations.

Security management includes the following:

- User management
- Access management
- Use management

The typical technology used includes:

- Encryption

(1) User Management

The management of users of a system constitutes the base for all of security management. To use a system, it is necessary to have a user ID issued by the system administrator. The user ID is information given to each user with the objective of enabling checking the authentication of a user's right to use a system and grasping the user's use status.

To manage user ID, the following should be considered:

- Only a necessary number of user IDs should be issued. User IDs having become unnecessary should be deleted as soon as possible to prevent the illicit use of those IDs.
- Sharing user ID must be prohibited. Different ID must be issued, even to each member of a development team.
- The amount of authority set to user ID should be as small as possible.

Passwords to authenticate users must also be managed. The password is most widely used as a means to authenticate a user.

To manage passwords, the following should be considered:

- Use of a password which cannot be easily guessed should be encouraged. The manager must check passwords regularly, and ask for a change if a password allowing an easy guess is found.
- A mechanism to change passwords regularly should be provided, for example, by limiting the term of validity for them.
- The security level of files recording passwords must be increased by use of encryption or other means. In addition, reference of the files by general users must be prohibited.

(2) Access Management

In access management, different access rights can be set for each user. By so doing, even for users of the same system, information and/or services available can be set differently, based on a user's position.

(3) Use Management

In use management, collecting the use data of a system increases the security level of the system as a whole.

Data to be collected include the following:

- User name (user ID)
- Use date
- Use time (log-in/log-out times)
- Terminals used
- Systems used
- Resources used

(4) Encryption

Corporate secrets handled by systems should be encrypted to prevent them from being altered. It is desirable for data resident in the systems to be encrypted as well.

2.1.5 Performance Management

One objective in managing the performance of system operations is to check whether services offered to users meet the required standards or not. At the same time, finding local performance degradation leads to prevention of a systems failure.

Items to be managed include:

- Response times and turn-around times
- Throughput
- Available time (starting time and ending time)
- The maximum number of terminals operating
- Quality of output data
- SLA (Service Level Agreement) of networks

Collecting and analyzing these data enables one to determine whether performance expected for a system is maintained.

Attention should also be paid to user complaints, because it may potentially involve a performance degradation, which cannot be identified with simple measurement.

In addition, changes in external factors (such as an increase in the number of transactions) may make it impossible to maintain system performance unless some measures are taken. Performance management should also be used routinely to enable the prediction of such a situation occurring, and to allow the making of proposals for new equipment and for changing up software versions.

2.1.6 Cost Management

Systems operation naturally means cost. Cost management is very important for enterprises seeking to increase profit, and due consideration must be given to it. Therefore, collecting cost data correctly, and cutting unnecessary costs as much as possible are important in running the systems.
TCO (Total Cost of Ownership) used for cost management is classified into the following two items.

- Initial cost
- Running cost

(1) Initial Cost

The initial cost is one-time cost in the system installation phase, and not generated after the system operation is started.
The types of initial costs include:

- Purchasing cost of equipment (system equipment, network equipment, terminals, etc.)
- Purchasing cost of software (basic software, software packages, etc.)
- Software development cost

These costs are not always generated. For example, equipment-purchasing cost is not required for rental equipment. Therefore, careful consideration should be given to these costs for installing a system.

(2) Running Cost

The running cost becomes necessary once a system is in operation. It is a persistent cost which is generated periodically and fixed.
The types of running cost include:

- Rental cost of equipment (system equipment, network equipment, terminals, etc.)
- License fees of software (basic software, software packages, etc.)
- Maintenance cost (for hardware and software maintenance)
- Maintenance cost of equipment (communication live cost, heating and lighting expenses)
- Expendables cost
- Personnel expenses

The running cost is classified into the fixed cost that is generated constantly, and the variable cost and expenses which depends on conditions. The variable costs and expenses should be managed on a monthly basis, to find the differences between the actual and budget data. Then, proper measures should be taken, if necessary.
A charging method may be introduced in which users (user organizations) pay the running cost. The charging method includes the proportional allocation method, in which cost is allocated depending on the amount of use, and the base allocation method in which the amount to be allocated is determined depending on the ratios of profits gained by the use of a system.

2.1.7 Other Operation Management

(1) System Operation

In view of the use, systems must be operated with consideration to the following:

- Operation manuals, describing system operation methods and operation procedures, are provided.
- Job control statements (job scheduling) are provided to enable the automatic processing of jobs.
- Integrity measures for inputting data are provided in addition to the management of inputting and outputting data.

(2) System Operation Tools

Various system operation tools are used to make smooth and easy system operations possible.
Typical system operation tools include:

- Tools for automatic operations
- Monitoring tools
- Diagnostic tools.

(3) System Transition

The transition means the work to switch the operation of a system from an old version to a new version. A system transition is conducted with the following procedures:

- Preparing a transition plan
- Preparing a transition procedure manual
- The execution of transition work
- Operation test
- Transition to the operation phase (work is handed over)

In transition work, attention must be paid to the management of version data. In distributed processing systems, transition work may take several days. In such a case, it must be checked to see that operations are possible without trouble even while old and new versions are both being used.

2.2 System Maintenance

The "Y2K problem" attracted much in the attention of the software industry. This is because programs, having operated normally until the end of 1999, might have generated an error on the arrival of 2000. Needless to say, software had to be corrected before an error cropped up, but it was not easy to detect programs needing correction from such an enormous amount of software. However, this was a problem that had to be handled. Therefore, software to help solve the problem had been developed and marketed. This Y2K problem has had an effect on information processing engineers as a warning about the difficulty of maintenance work.

The amount of maintenance work has been increasing year after year, with maintenance costs now accounting for 60 to 70 percent of the cost for the entire life cycle of a system.

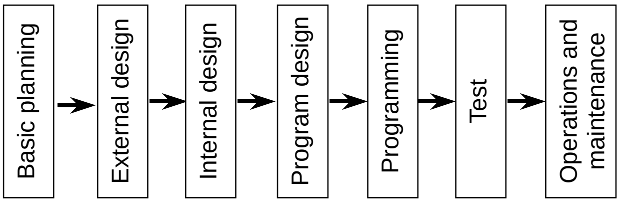
The first step to cope with these situations is to simplify maintenance work. The simplification of maintenance work is one of the measures directly leading to a significant reduction in development cost. To achieve it, close communication and frequent contacts with the users and the defining of measures to be taken when trouble occurs, and the production of manuals describing details to prevent troubles are important.

What is ideal is to develop easily maintainable systems that include very few defects, and enables the handling of various problems expected in the future.

2.2.1 What is Maintenance?

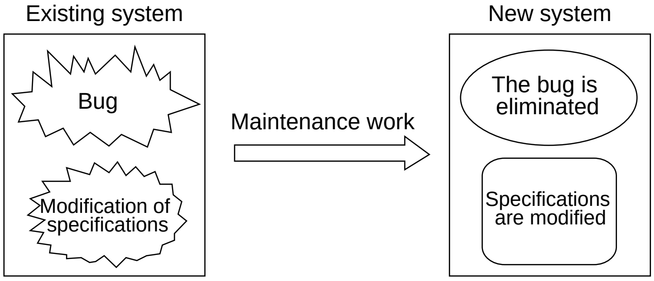
A large-scale system is mostly developed with the following waterfall model as shown in Figure 2-2-1:

Figure 2-2-1 The water fall model



Actual operations of a developed system are started after comprehensive operation tests are completed. However, a system (programs) must be corrected if a bug (error) hiding in the system is found, or when the user requests the modification of the system specifications. Work associated with these corrections and modifications is called maintenance (see Figure 2-2-2).

Figure 2-2-2 Maintenance



2.2.2 Importance of Maintenance Work

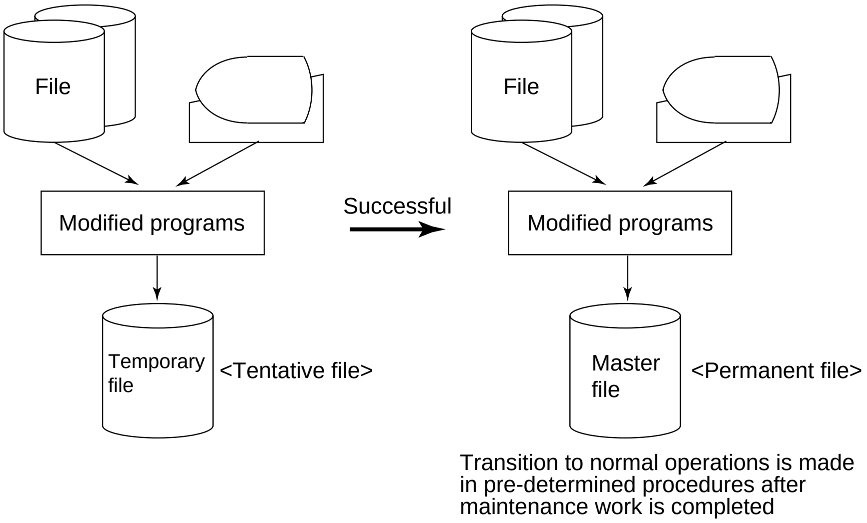
Maintenance is an important work that includes the restoration of a system to a normal status by correcting a bug, and the flexible handling of user's modification requests. Some maintenance work is required immediately after the completion of a development. However, some is still required several years later. In addition, modifications must be made to the existing system. Therefore, the work includes various risks, requiring lots of care (see Figure 2-2-3). In addition, as described above, the cost of maintenance and the amount of maintenance work have kept increasing year by year. Prior to starting maintenance work, thorough investigation, considering the following points, must be conducted:

< Issues to be considered for maintenance work>

- Are user's modification requests reasonable?
- Are other users affected?
- Does a system configuration need to be changed?

In particular, the test to check whether modifications by maintenance work have affected other systems is called a "regression test."

Figure 2-2-3 Modification of an existing system



Transition to normal operations is made in pre-determined procedures after maintenance work is completed

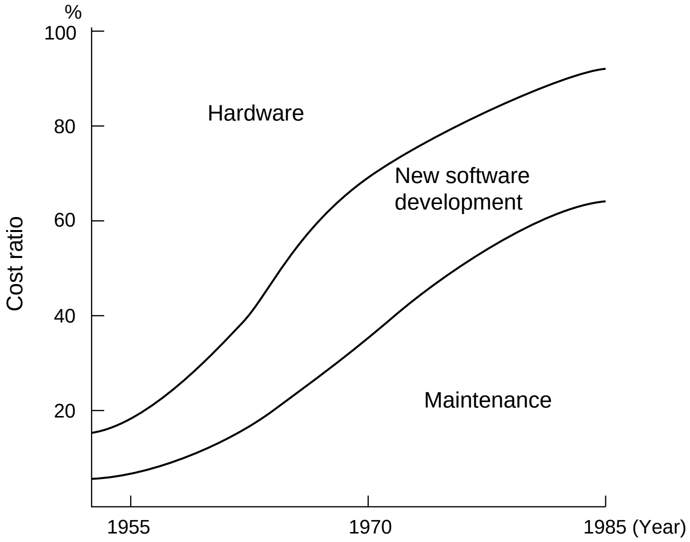
In the example of Figure 2-2-3, outputs from processing using modified programs are stored in a temporary file, instead of the master file. Then, after it is confirmed that the temporary file functions normally, the outputs are transferred to the master file used in the actual operations.

Figure 2-2-3 shows an example of maintenance work that should be conducted carefully.

2.2.3 Maintenance Cost

Figure 2-2-4 shows Boehm's transition curves. This curve indicates the yearly ratios of maintenance cost occupies in the total development cost.

Figure 2-2-4
Boehm's transition curves



Formerly the cost of hardware was dominant. Today, however, the maintenance cost has become dominant as the hardware cost has become increasingly lower.

Maintenance work almost inevitably becomes necessary. Therefore, maintenance cost must be reduced as much as possible. To achieve this reduction, ordinary operations must be conducted based on users' system manuals. In case troubles occur, manuals that describe their situations, causes and countermeasures must be provided.

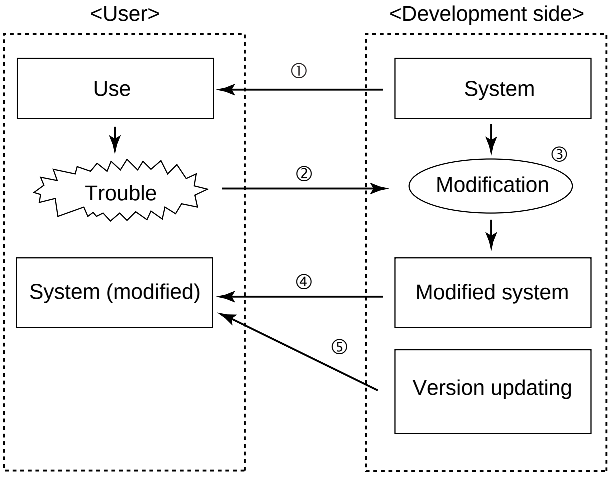
2.2.4 Maintenance Tasks

Here, maintenance work as a whole is described centered on relationships between the user side and the development side.

(1) Communication between the user side and the development side

Figure 2-2-5 shows a series of communication flows between the user side and the development side in the installation of a new system and its maintenance.

Figure 2-2-5 Communications between the user side and the development side



Basically, the following exchanges are made between the user side and the development side.

① The development side hands over a system (including manuals) to the user side.

② The user uses the system, and notifies the development side every time trouble occurs, and requests modification of the specifications.

③ The development side investigates these requests, and conducts modification work for troubles that have occurred, and in answer to the user's requests.

④ The development side sends to the user side, modified data and/or modified programs.

⑤ The development side periodically sends new version (version-up) programs to the user.

(2) Measures to be taken by the Development Side and by the User Side

What measures should the user side and the development side take respectively, in the event that the maintenance costs keep increasing? Here, the measures are described step by step.

① Efforts to reduce maintenance work must be made

a. The development side: The extent of inconvenience and difficulty the user suffers from troubles, and poor performances of a system must be understood

As an example, in August 1997, the operations of the stock-exchanging system of Tokyo Stock Exchange stopped for about two hours, halting the selling and buying of more than 1,700 stocks for half a day (see Figure 2-2-6).

The Tokyo Stock Exchange is one of the dominant securities exchanges in the world, as well as in Japan. The halting of such a system is an international problem, affecting the world significantly.

The cause was reportedly the runaway of programs due to an occurrence of an accidental event.

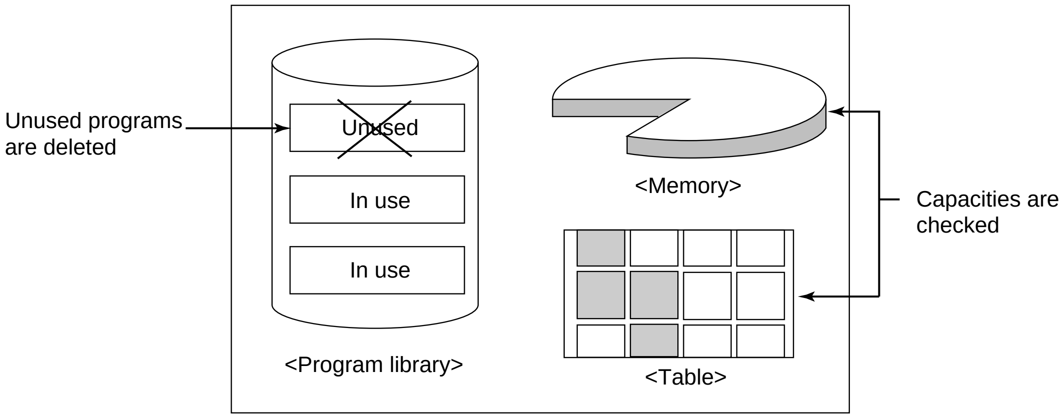
For ordinary operations, the development side must conduct, extensive tests covering all the cases conceivable. However, the deletion of unused programs and checking memory and table capacities, are also an important work to be performed routinely.

Figure 2-2-6 A series of events between a trouble occurrence in the Tokyo Stock Exchange and the ending of maintenance work

August 1	
6:50	The stock exchange system was put in operation.
8:20	Receiving stock orders was started.
8:21	Operations of TRS (a stipulation related system) stopped abruptly, and the system was switched to the standby system.
8:22	Operations of the stand-by system also halted, bringing the operations of TRS to a complete standstill. The programs were reloaded, and the standby system was reactivated with "quick IPL."
8:31	The standby system reactivated with "quick IPL" halted again. The "quick IPL" were repeated two more times. However, system operations failed each time.
8:50	Operations of the order-receiving system were brought to a stop.
9:00	Transactions for the morning session were started.
9:20	Operations to put the stock exchange system in use were conducted anew (after clearing the files), then the system was reactivated.
10:30	It was confirmed that the normal operations of the system were restored.
11:00	Transactions for the morning session were closed.
11:30	Order-receiving operations for the afternoon were started 45 minutes earlier than usual.
12:30	Transactions for all stocks were started for the afternoon session.
Middle of the night	By the middle of the night the investigation of the cause was completed. The program was modified to prevent the activation of "the median calculation of foreign stock values," having triggered the trouble.
August 2	
12:30	Tests of the system modified on a test machine were completed.
18:00	The modified system was operated in test environments (operating system) of the actual machine.
August 3	
15:30	Final confirmation of the actual machine operations was made in actual environments.

* IPL: Initial program loading

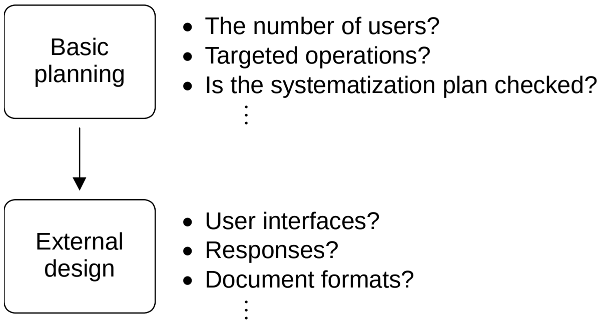
Figure 2-2-7 Deletion of programs and checking capacities



b. The user side: Request modifications to functions and operability after actual operations are started, and the improvement of performance due to an increase in the amount of use.

While using a system, requests to improve functions and operability, or those to improve responses that will degrade due to increase in the number of users and the amount of traffic, will be made. Modification in some degree would be inevitable due to social change or other factors. However, there should be some that will able to be avoided if thorough communication with the user side is made in basic planning and external design phases of a system development. Having sufficient meetings with the user side in view of the future is necessary in the system design phase.

Figure 2-2-8 Communication in the design phase



Thorough investigation is necessary with users participating.

In addition, if modification is unavoidable, it is important to sufficiently examine the validity of maintenance.

② Efforts to enable smooth maintenance work

a. Documentation of trouble in formation

The user must document the details of a problem even if it may be considered trivial. Describing troubles, for example, in formats unified throughout a company is better than just reporting verbally. This is because the descriptions will become quite helpful to take preventative measures against future troubles.

b. Overall management with a trouble register

A trouble register, enabling unified management of troubles in a system, must be provided. A register may be provided and managed for each user or each system. However, from the efficiency viewpoint, it is more desirable to provide a unified management method throughout a company.

c. Analysis of troubles, performance analysis and the modification improvement, and management of design documents and source programs

System maintenance work requires manpower more than other types of work. The amount of manpower per trouble reportedly is 10 person-days ordinarily. Needless to say, work for a trouble the restoration of which is expected to be difficult takes a longer time. In most of the restoration work, various design documents and source programs are used. Therefore, easily understandable design documents must be provided in the initial design phase in consideration of later use. Source programs must also be written in forms to allow easy debugging, for example, by utilizing a structured programming method.

d. Revision of work manuals

Systems are operated by human beings. Therefore, even though work is performed correctly following manual instructions, errors may be included in inputted data or an incorrect operation of computers may be performed. To avoid making such errors, manuals must be revised so that the occurrence of errors can be prevented as much as possible. In addition, care naturally must be taken for persons lacking in operational experience, such as new employees, and education within the company must be conducted as well.

e. Revision project to add functions and to improve performance

Systems, having fully supported user's needs initially, will become less advanced over time. Therefore, updating the software is essential. A project must be organized to survey how the system is being used, and the system must be updated to enhance the usefulness of the system.

(3) Maintenance Tasks

Actual maintenance tasks are classified into the following three tasks.

- Correction task : Bugs in programs and systems are removed. In particular, an urgent action is required for OLTP (OnLine Transaction Processing).
- Modification task : Data such as an era name, interest rates, tax rates and data items are modified depending on social changes and when the need arises.
- Improvement task : Systems are improved, for example, through the modification of specifications due to user's requests.

Among these tasks, the amount of improvement task has recently been increasing. It is because users' requests, such as an increase in processing speeds, and for the change or modification of interfaces to make them more easily usable, have been diversified.

2.2.5 Maintenance Organization

The maintenance organization means the organization responsible for performing maintenance work when maintenance is requested for a system.

To establish a maintenance organization, particular considerations must be given to the following:

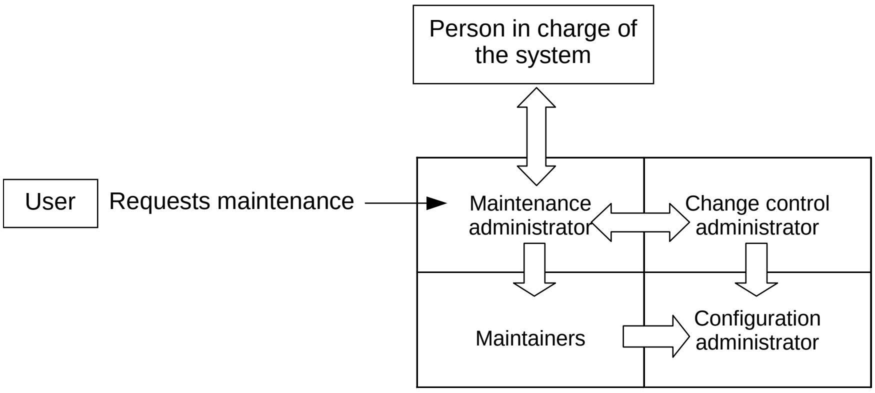
- Conducting the maintenance work as well as making development efforts at the same time is impossible. The development organization and the maintenance organization must be established separately.
- It is desirable that the person in charge of maintenance be involved in the system development.
- Outputs from a system development include documents such as design documents, in addition to the system itself. Therefore, a person or persons in charge of documentation management must be assigned to create some unified management of these documents.
- A person (maintenance administrator) liaison between users and the maintenance organization must be assigned. The number of such liaisons must be limited to one.

If a developer performs maintenance work at his own discretion, a serious problem may ensue. Therefore, a group must be organized to perform maintenance work.

Here, the actual maintenance organization and roles played by each member are described.

Figure the 2-2-9 shows a maintenance organization example.

Figure 2-2-9
A maintenance organization


Person in charge of the system
User
Requests maintenance
Maintenance administrator
Change control administrator
Maintainers
Configuration administrator

The maintenance organization is arranged as shown in Figure 2-2-9 to enable smooth maintenance work. With clearly separated roles assigned to each member, they can cooperate effectively in maintenance work.

(1) Roles (functions) of Maintainers and Maintenance Administrator

① Maintenance administrator

Maintenance administrator constitutes the core of maintenance work and plays the following roles:

- The administrator like attending a window to directly receive requests from users. Any request must be received through the administrator to unify maintenance work management. "Any developer can do maintenance work" is not permitted.
- The maintenance administrator works in cooperation with change control administrator and checks the feasibility of maintenance from various viewpoints (such as contents of maintenance, delivery time and cost). The system administrator may participate in the checking as well.
- The administrator gives instructions developers to perform work.

② Maintainers

Maintainers perform the following tasks:

- With instructions given by the maintenance administrator, they actually do the maintenance work.
- They regularly report to the maintenance administrator and configuration administrator, their work in progress status reports and work completion reports.

③ Change control administrator

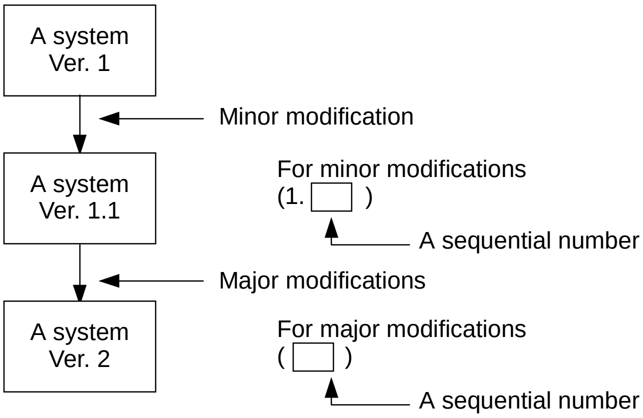
Change control administrator is also called a document administrator or data administrator.
Change control administrator plays the following roles:

- Change control administrator checks, in cooperation with the maintenance administrator, the validity of maintenance from various viewpoints (such as contents of maintenance, delivery time and cost).
- Each modification is documented and managed.

④ Configuration administrator

Configuration administrator manages versions of systems (see Figure 2-2-10).
If different versions are simultaneously used by different users, problems will surface in ease of use or operations in connection to other software. Therefore, the configuration administrator must accurately manage the changes of versions caused by changes.

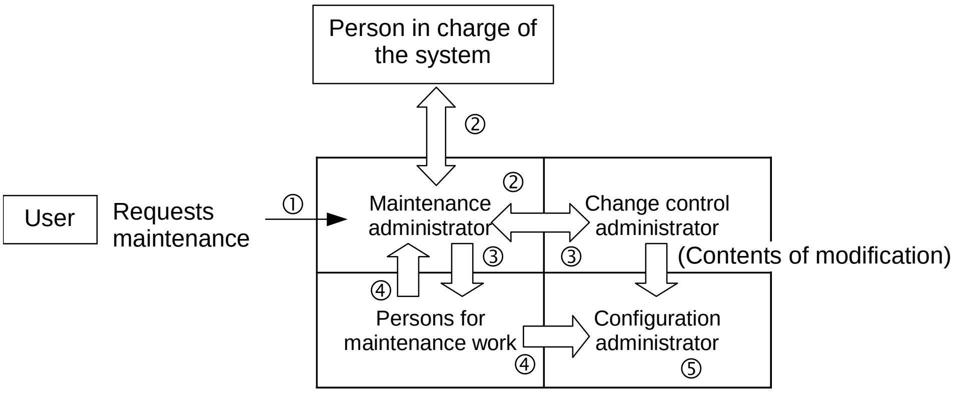
Figure 2-2-10
A version updating



(2) Maintenance Procedures

Figure 2-2-11 shows maintenance procedures.

Figure 2-2-11
Maintenance procedures


Person in charge of the system
②
User
Requests maintenance
①
②
Maintenance administrator
Change control administrator
③
③
(Contents of modification)
④
Persons for maintenance work
Configuration administrator
④
⑤

Actual works are as follows:

① Users request maintenance to the maintenance administrator.

② After receiving the request, the maintenance administrator checks, discusses the validity, and makes final decision together with the change control administrator and person responsible for the system.

③ Once the execution of the maintenance work is decided on, the maintenance administrator gives instructions to the maintainers to carry out the work. In addition, each system modification is documented and managed.

④ Maintenance workers report their work progress status to the maintenance administrator at appropriate times and their work completion to the maintenance administrator and configuration administrator.

⑤ The configuration administrator updates the version of the targeted system.

2.2.6 Types of Maintenance

Types of maintenance include the preventive maintenance that allows plans to be devised in advance, and the post maintenance that is performed when a failure occurs.

(1) Preventive Maintenance

Preventive maintenance is performed to prevent failures, and the maintenance plans are devised in advance. Preventive maintenance, allows for a planned acquisition of maintenance personnel, and enables efficient maintenance.
Preventive maintenance includes:

- Daily maintenance
- Scheduled maintenance

① Daily maintenance

The daily maintenance is conducted every day to check the states and performances of equipment making up the systems. Ordinarily, measures allowing checks to be conducted without stopping system operations are provided. Remote maintenance is widely in use in systems that involve locations far apart.

② Scheduled maintenance

The maintenance is conducted at pre-determined intervals.
This maintenance is a comparatively large-scale work in which system operations are either stopped or performed with alternative equipment. Since operations are stopped temporarily the work must be carried out quickly.

(2) Post Maintenance

Post maintenance is conducted when a failure has occurred, or may occur.
The maintenance is not a work which is scheduled in advance. Therefore, the acquisition of maintenance personnel for the work may be difficult. In case such a situation occurs, having extra emergency maintenance personnel is desirable.
Post maintenance includes the following:

- Tentative maintenance
- Emergency maintenance

① Tentative maintenance

Tentative maintenance is conducted only when an unusual situation is found in system operations, such as a sudden sharp deterioration in performance or an unusual sound is detected coming from the operating equipment.
The objective of this maintenance is to take measures before a failure actually occurs.

② Emergency maintenance

Emergency maintenance is conducted to make a recovery from the failure when it occurs. The work results in stopping systems temporarily without any notification in advance.
The necessity of such emergency maintenance work should be avoided as much as possible. Therefore, if the number of emergency maintenance works tends to increase, maintenance plans, including the reconsideration of the system as a whole, must be devised and executed.

Conducting maintenance naturally requires maintenance personnel. However, retaining the personnel may pose a problem with the cost. Therefore, maintenance work may be entrusted to external companies.
Benefits gained by entrusting the work to external companies with a maintenance contract include:

- The cost can be reduced.
- Personnel problems can be solved, because consideration for the personnel becomes unnecessary.
- It eliminates the need for the 24-hour duty service from employees.
- Detailed maintenance by specialists is expected.

However, the demerits include:

- Problems may be created from security aspects.
- Maintenance personnel may be changed at the discretion of the company contracted with.
- Not all the maintenance personnel assigned are talented.
- Raising issues on the operation is not done.

In concluding a maintenance contract, various aspects, including companies to be entrusted with the work, and maintenance mode (resident or on call), must be considered.

2.2.7 Hardware Maintenance and Software Maintenance

Though the same word, "maintenance," is used, hardware maintenance is quite different from software maintenance in issues to be paid attention to, and methods used.

(1) Hardware Maintenance

The hardware maintenance is conducted by paying special attention to the following:

- Reliability
- Performance balance
- Inventory status

① Reliability

Maintenance items in view of reliability include the following:

- Important phenomena monitoring
  Important events in systems are monitored.
- Failure tendency monitoring
  In this monitoring, data are sampled for a long period to analyze failure tendencies.
- Specified monitoring
  This monitoring is for important issues in consideration of the reliability characteristics of equipment making up a system.

② Performance balance

If a system is in use for a long time, the replacement of old devices with new ones may cause an imbalance between them. If the balance is made worse, devices with insufficient performances may affect the performance of a system as a whole. Therefore, performance balance between devices must be checked.

③ Inventory status

Expendables necessary for the systems, and the inventory of parts to be replaced or repaired must be checked. It is important that these items are checked and inspected routinely to prevent being forced to stop operating until parts arrive.

(2) Software Maintenance

The software maintenance is conducted by paying special attention to the following:

- Addition and improvement of functions
- Restoration
- Prevention

① Addition and improvement of functions

To meet requirement modifications of a system in operation, maintenance is conducted for modification requests for software specifications and/or for performance improvement.

② Restoration

Maintenance related to the restoration of software in operation includes the following:

- Corrective maintenance
  This maintenance is performed if performance requirements or requirement specifications are not

satisfied.

- Adaptive maintenance
  This maintenance is for the necessity of modifying software due to changes in external factors.
- Maintenance for perfection
  This maintenance is performed to increase the level of software perfection.

③ Prevention

This maintenance is performed on the software side to prevent system failures, or to support solving operational problems, and includes the following:

- Technical support
  This is the maintenance to solve operational problems.
- Preventive modification
  This is the maintenance to prevent the occurrence of failures in advance.

Exercises for No.2 Chapter 2 (System Operation and Maintenance)

Q1 **Which of the following is not included in resources to be managed in operation management?**

a. Computers
b. Databases
c. Programmers
d. Programs

Q2 **Which of the following is correct as the standard procedure to be performed in a system failure?**

a. Temporary measures are taken.
b. The use of temporary measures is terminated and they are removed.
c. Failure portions are identified, and failures are localized.
d. Failure portions are separated.

a. The occurrence of a failure→finding the failure→a→c→d→permanent measures are taken→b
b. The occurrence of a failure→finding the failure→b→c→a→permanent measures are taken→d
c. The occurrence of a failure→finding the failure→c→a→b→permanent measures are taken→d
d. The occurrence of a failure→finding the failure→c→d→a→permanent measures are taken→b

Q3 **Which of the following is the most appropriate for user ID management?**

a. All ID users participating in the same project use the same ID.
b. A user having more than one ID sets the same password for all the IDs.
c. If authority is given to a user ID, it should be limited to as minimal as possible.
d. The deletion of a user ID must be made long after the abolition of the ID has been notified.

Q4 **Which of the following is the most inappropriate for the handling of passwords and password files in the system management organization?**

a. Whether passwords can be guessed easily or not is regularly checked, and the use of different passwords is urged for problematic ones.
b. In order to reduce the extent that passwords are referred to, it is recommended that users record their passwords on pocket books or elsewhere.
c. If effective terms can be set for passwords, the functions must be used.
d. Reference by ordinary users to password files must be prohibited, even if the passwords are encrypted.

Q5 Which of the following is <u>unrelated</u> to service standards for online system operations?

a. Response times

b. Operation starting time

c. Failure recovery time

d. Debugging time

Q6 Which of the following graphs, with the number of years elapsed as the horizontal axis and monthly payments for the vertical axis, is the most appropriate if computers are introduced under the following conditions?

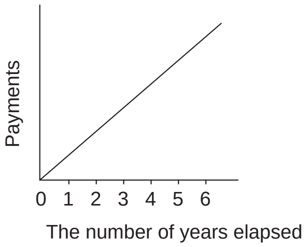
① Computer cost

- **For the initial five years, the fixed monthly amount (of lease payment) calculated, based on purchasing prices of the computers and their lease rates, is paid.**
- **In the sixth and later years, the fixed monthly amount calculated, based on the one-tenth purchasing prices of the computers and their lease rates, is paid.**

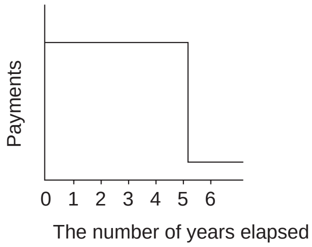
② Maintenance cost

- **The fixed monthly amount is paid as maintenance charges to a maintenance company.**

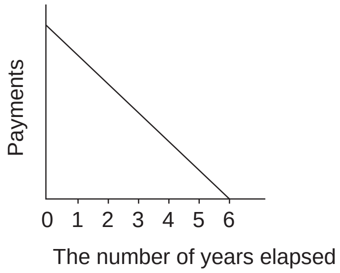
a.



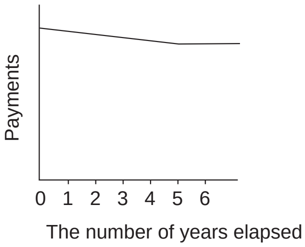
b.



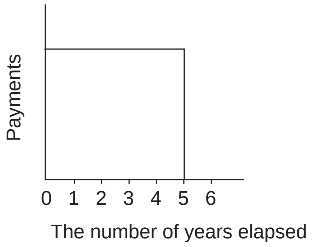
c.



d.



e.



Q7 Which of the following software test methods is used to check whether modifications made for software maintenance have affected other portions or not?

a. Operation tests

b. Linkage tests

c. System tests

d. Regression tests

Q8 Which of the following is <u>inappropriate</u> for the descriptions of maintenance work for the application software which was developed in house?

a. The operation manager starts using new software after modifications have been approved, and removes old software based on a pre-determined plan.

b. Programmers who have developed the original programs perform program modification associated with the modification of specifications made after a development has been completed. Then, new software is quickly put in use in actual environments.

c. Persons conducting tests must analyze areas affected by a modification, conduct testing of those parts which are related to modified programs, and make evaluations.

d. In performing maintenance, standards, methodologies and procedures regarding document administration, maintenance methods and program modifying procedures must be provided in advance.

Index