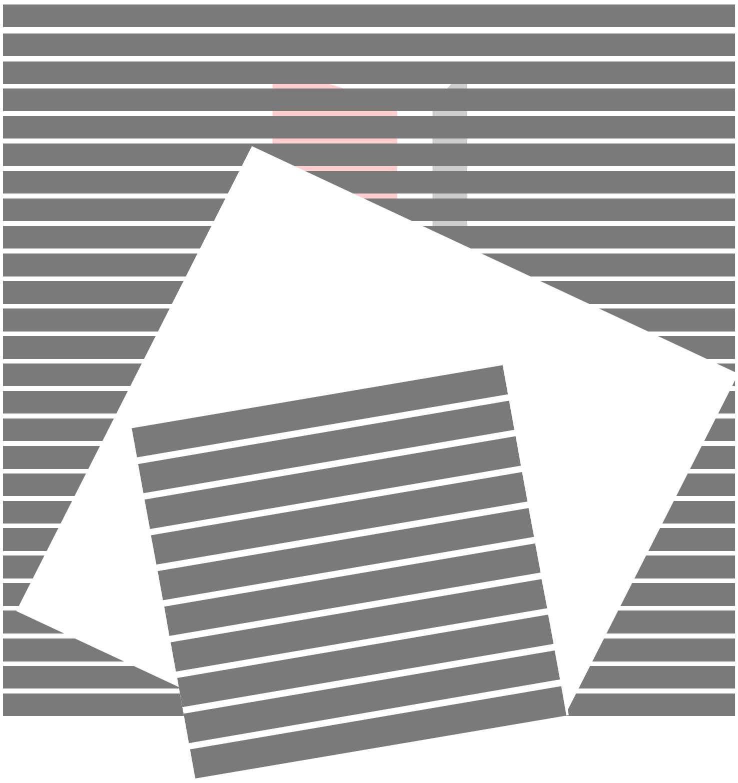
METI
Ministry of Economy,
Trade and Industry

Textbook for
Software Design & Development Engineers

NO. 3 SYSTEM DEVELOPMENT, OPERATIONS AND MAINTENANCE



Second Edition

REVISED AND UPDATED BY

Japan Information Processing Development Corporation
Japan Information-Technology Engineers Examination Center

Textbook for Software Design & Development Engineers

No. 3 SYSTEM DEVELOPMENT AND OPERATIONS

First edition first printed October 1, 2001
Second edition first printed August 1, 2002

Japan Information Processing Development Corporation
Japan Information-Technology Engineers Examination Center
TIME 24 Building 19th Floor, 2-45 Aomi, Koto-ku, Tokyo 135-8073 JAPAN

Table of Contents

Part 1 Software Engineering

1. Overview of Software Engineering

1.1 Origin of Software Engineering 1-2
- 1.1.1 Software Crisis 1-2

1.2 Definition of Software Engineering 1-4
- 1.2.1 Software production 1-4
- 1.2.2 Industrializing the management process 1-4

1.3 Achievements in Software Engineering 1-5
- 1.3.1 Methodology 1-5
- 1.3.2 Techniques in Software Engineering 1-6
- 1.3.3 Logic-oriented paradigm 1-8
- 1.3.4 Function-oriented paradigm 1-8
- 1.3.5 Object-oriented paradigm 1-8
- 1.3.6 Agent-oriented paradigm 1-8

Exercises 1-9

2. Process Models and Cost Models for Software Development

2.1 Process Models for Software Development 1-11
- 2.1.1 Waterfall model 1-11
- 2.1.2 Spiral model 1-12
- 2.1.3 Prototyping model 1-13

2.2 Cost Models for Software Development 1-15
- 2.2.1 Halstead model 1-15
- 2.2.2 FP (Function Point) model 1-16
- 2.2.3 COCOMO (COnstructive COst Model) model 1-17

Exercises 1-19

3. Defining Software Requirements

3.1 Software Requirements 1-21
- 3.1.1 Purpose of systematization 1-21
- 3.1.2 Configuration 1-21
- 3.1.3 Function 1-22

3.1.4 Performance 1-22
3.1.5 Constrains 1-22
3.2 Software Requirements Analysis 1-23
3.2.1 KJ (Jiro Kawakita) method 1-23
3.2.2 Functional analysis 1-24
3.2.3 Event response analysis 1-24
3.2.4 Structured analysis 1-25
3.3 Software Requirement Models 1-26
3.3.1 Functional hierarchical model 1-26
3.3.2 Dataflow model 1-27
3.3.3 Control flow model 1-28
3.3.4 Finite-state machine model 1-29
3.3.5 Petri-net model 1-29
3.3.6 Data-oriented model 1-29
3.3.7 Object-oriented model 1-30
3.3.8 Parallel process model 1-31
3.4 Techniques for defining Software Requirements 1-32
3.4.1 Structured analysis method 1-32
3.4.2 Object-oriented analysis method 1-34
Exercises 1-37

4. Software Design

4.1 Concepts of Software Design 1-44
4.1.1 Concepts of partitioning and integration 1-44
4.1.2 Concept of stepwise refinement 1-45
4.1.3 Concept of abstraction 1-46
4.1.4 Concept of information hiding 1-49
4.2 Software Design Methods 1-50
4.2.1 Structured design method 1-50
4.2.2 Jackson method 1-53
4.2.3 Warnier method 1-55
4.2.4 Object-oriented design method 1-56
Exercises 1-58

5. Programming

5.1 Procedural Programming Paradigm 1-63
5.1.1 Structured programming 1-64
5.1.2 Structured charts 1-71

5.2 Logical Programming Paradigm 1-75
5.2.1 Prolog (Programming in Logic) 1-75
5.2.2 Unification 1-76
5.2.3 Backtracking 1-77
5.3 Functional Programming Paradigm 1-78
5.3.1 Lisp (LIST Processor) 1-78
5.4 Object-oriented Programming Paradigm 1-79
5.4.1 Smalltalk 1-79
5.4.2 C++ 1-79
5.4.3 Java 1-81
5.4.4 Instantiation of classes 1-82
5.4.5 Class hierarchy and inheritance 1-83
5.4.6 Encapsulation as data abstraction 1-84
5.4.7 Message passing and polymorphism 1-84
Exercises 1-86

6. Software Quality

6.1 Software Quality Factors 1-92
6.1.1 Usability 1-93
6.1.2 Maintainability 1-94
6.1.3 Portability 1-95
6.1.4 Reliability 1-97
6.1.5 Efficiency 1-98
6.1.6 Updatability 1-99
6.1.7 Ease of testing 1-99
6.1.8 Operability 1-99
6.2 Program Testing 1-100
6.2.1 Technique for designing test cases 1-100
6.2.2 Technique for testing module integration 1-106
6.2.3 Static tests and dynamic tests 1-108
6.3 Quality Control of Software 1-109
6.3.1 Review method 1-109
6.3.2 Reliability estimation method 1-111
6.3.3 QC method 1-112
Exercises 1-113

7. Software Development Environment

7.1 Software Development Tools 1-120

7.1.1 Common functions 1-120
7.1.2 Design process tools 1-123
7.1.3 Development process tools 1-125
7.1.4 Test process tools 1-126
7.1.5 Maintenance process tools 1-126
7.2 CASE 1-128
7.2.1 upper-CASE 1-129
7.2.2 lower-CASE 1-129
7.2.3 component-CASE 1-129
7.2.4 integrated-CASE 1-130
7.2.5 Repository 1-130
7.2.6 Forward engineering 1-130
7.2.7Reverse engineering 1-131
7.2.8 Re-engineering 1-131
Exercises 1-133

8 Trends in Software Engineering

8.1 Software Tools 1-137
8.1.1 Component-orientation 1-137
8.1.2 JavaBeans 1-141
8.1.3 UML 1-144
8.1.4 COTS 1-145
Exercises 1-146

Answers for Part 1Exercises 1-147

Part 2 External Design

1. External Design Procedures

1.1 Preparations for External Design Work 2-2
- 1.1.1 Policies for system development 2-2
- 1.1.2 Systematization requirements definition 2-3

1.2 External Design Procedures 2-5
- 1.2.1 Designing the system functions 2-5
- 1.2.2 Designing the data model 2-5
- 1.2.3 Producing the external design 2-6
- 1.2.4 Design review 2-6

2. System Function Design

2.1 Selecting a System Architecture 2-8
- 2.1.1 Hardware configuration 2-9
- 2.1.2 Software configuration 2-9
- 2.1.3 Application packages 2-9
- 2.1.4 Scope of systematization 2-10
- 2.1.5 Candidate architecture 2-11

2.2 Subsystem Functional Specifications and Interface Design 2-14
- 2.2.1 Subsystem division 2-14
- 2.2.2 Defining subsystem functional specifications 2-16
- 2.2.3 Subsystem interface definition 2-17

2.3 Designing Security 2-19
- 2.3.1 Security policy 2-19
- 2.3.2 Security requirements 2-20
- 2.3.3 Security implementation method 2-21

2.4 Job Model Design 2-24
- 2.4.1 Details of job flow 2-24
- 2.4.2 Data flow refinement 2-28

Exercises 2-30

3. Data Model Design

3.1 Conceptual Models 2-35
- 3.1.1 ER model 2-35

3.2 Logical Data Model 2-42
- 3.2.1 Hierarchical model 2-42

3.2.2 Network model 2-43
3.2.3 Relational model 2-44
Exercises 2-45

4. Preparation of External Design Documents

4.1 Preparation of User Manual(Outline) 2-49
4.1.1 Participants 2-49
4.1.2 Review method 2-49
4.1.3 Format of user manual 2-49
4.2 Design of System Test Specifications 2-50
4.2.1 Policy planning for system testing 2-50
4.2.2 System test environment 2-53
4.2.3 Documentation procedure 2-53
4.3 External Design Documents 2-55
4.3.1 System Configuration Diagram 2-55
4.3.2 Subsystem Relationship Diagram 2-55
4.3.3 System Job Flow 2-56
4.3.4 System Function of Specifications 2-56
4.3.5 Input-output Specifications (screen and report specification) 2-57
4.3.6 Data Specification 2-57
4.4 Design Review 2-58
4.4.1 Review system 2-58
4.4.2 Organization 2-60
Exercises 2-64

Answers for Part 2 Exercises 2-68

Part 3 Internal Design

1. Procedure for Internal Design

1.1 Internal Design Procedure 3-2
1.1.1 Software component design 3-3
1.1.2 Input/output design 3-4
1.1.3 Physical data design 3-4
1.1.4 Creation and reuse of parts 3-5
1.1.5 Preparation of internal design documents 3-6
1.1.6 Design review 3-6

2. Software Component Design

2.1 Software Structure Design 3-8
2.1.1 Partitioning into components 3-10
2.1.2 Determining functional specifications 3-22
2.1.3 Interfaces between components 3-26
2.2 Use of Middleware and Tools 3-27
Exercise 3-29

3. Input/Output Design

3.1 Report design 3-36
3.2 GUI design 3-49
3.2.1 Screen design 3-49
3.2.2 Input and output check system and message design 3-59

4. Physical Data Design

4.1 File Design 3-78
4.1.1 Analysis of data characteristics 3-79
4.1.2 Determination of the logical data organization 3-81
4.1.3 Selection of data storage media 3-84
4.1.4 Record layout design 3-85
4.2 Physical data design 3-91
4.2.1 Mapping of the logical data structure 3-91
4.2.2 Capacity calculation 3-97
4.2.3 Performance improvement and optimization 3-99

Exercises 3-101

5. Creation and Reuse of Parts

5.1 Creating and Reusing Parts 3-109
5.1.1 Concept of creating and reusing parts 3-109
5.2 Using Software Packages 3-113
5.2.1 Subprogram libraries 3-113
5.2.2 Class libraries (combination with object-oriented languages) 3-114
Exercises 3-115

6. Creation of Internal Design Documents

6.1 Organization of Internal Design Documents 3-118
6.1.1 Diagram for partitioning into components 3-118
6.1.2 Diagram of interface between components 3-120
6.1.3 Component processing specifications 3-122
6.1.4 Screen design document 3-124
6.1.5 Form design document 3-126
6.1.6 File design document 3-128
6.1.7 Database design document 3-130
6.2 Design Review 3-132
6.2.1 Participants 3-132
6.2.2 Review types 3-134
6.2.3 Format of specifications 3-134
Exercises 3-137

Answers for Part 3 Exercises 3-140

Part 4 Program Design

1. Procedure for Program Design

1.1 Program Design Procedure 4-2
- 1.1.1 Confirmation of internal design documents 4-2
- 1.1.2 Partitioning into modules 4-3
- 1.1.3 Preparing module specifications 4-3
- 1.1.4 Program design documents 4-3
- 1.1.5 Partition of test specifications 4-4
- 1.1.6 Design review 4-4

2. Program Design Criteria

2.1 Choosing Module Partitioning Technique 4-6
- 2.1.1 Partitioning by data flow 4-6
- 2.1.2 Partitioning methods focusing on data structure 4-11

2.2 Module Partitioning Criteria 4-14
- 2.2.1 Module size 4-14
- 2.2.2 Module strength 4-14
- 2.2.3 Module coupling 4-17
- 2.2.4 Control scope and impact scope 4-20

2.3 Reusing Patterns and Parts 4-21
- 2.3.1 Extracting process pattern and parts 4-21
- 2.3.2 Reviewing the processing pattern and parts list 4-23
- 2.3.3 Reuse rate of processing pattern and parts 4-23
- 2.3.4 Standardizing the processing patterns and parts 4-23

Exercises 4-25

3. Creating of Program Design Document

3.1 Standardizing the Program Design Document 4-28
- 3.1.1 Contents 4-28
- 3.1.2 Modification criteria for maintenance and improvement 4-38
- 3.1.3 Contents of the user’s manual 4-40

3.2 Description of Processing Outline 4-41

3.3 Detailed Database Design 4-45
- 3.3.1 Clarifying database input-output conditions 4-45
- 3.3.2 Physical database design 4-48
- 3.3.3 Database creation 4-52

3.4 Standard Interface between Modules 4-60
3.4.1 Interface conditions within a program 4-61
3.4.2 Interface conditions within programs 4-64
3.4.3 Conditions for reusing program interfaces 4-66
Exercises 4-72

4. Creating Module Specifications and Test Specifications

4.1 Creating module specifications 4-75
4.1.1 Procedures 4-75
4.1.2 Methods 4-78
4.1.3 Main considerations 4-81
4.2 Creating test specifications 4-88
4.2.1 Types of tests and their objectives 4-88
4.2.2 Important considerations in testcase design 4-89
4.2.3 Testing methods 4-93
Exercises 4-96

5. Design Review

5.1 Program Design Documents Review 4-104
5.2 Test Specifications Review 4-113
5.3 User Manuals Review 4-114
Exercises 4-115

Answers for Part 4 Exercises 4-118

Part 5 System Operation and Maintenance

1. System Operations

1.1 What are managed in System Operations 5-2
1.1.1 Resource Management 5-4
1.1.2 Problem Management 5-6
1.1.3 Facility Management 5-9
1.1.4 Security Management 5-12
1.1.5 Performance Management 5-14
1.1.6 Cost Management 5-14
1.1.7 Other Operation Management 5-16
1.2 System Transition Steps 5-17
1.3 Operational Tests 5-18
1.4 Evaluation of System Operations 5-19
1.5 Operation Organization 5-21

2. System Maintenance

2.1 What is Maintenance? 5-23
2.1.1 Importance of Maintenance Work 5-23
2.1.2 Maintenance Cost 5-24
2.2 Planning for Future Maintenance during Development 5-25
2.3 Types of Maintenance during system operation 5-26
2.3.1 Preventive Maintenance and Post Maintenance 5-26
2.3.2 Recovery from Failure 5-29
2.3.3 Maintenance Organization 5-30
2.3.4 Hardware Maintenance and Software Maintenance 5-33
2.3.5 Discarding the system 5-36

Part 6 Project Management

1. Project Management

1.1 Overview of Project Management 6-2
1.1.1 What is Project Management? 6-2
1.1.2 Project Management Items 6-4
1.2 International Standards 6-12
1.2.1 PMBOK (Project Management Body of Knowledge) 6-12
1.2.2 Capability Maturity Model (CMM) 6-14
1.2.3 EVMS (Earned Value Management Systems) 6-17
1.3 Project Planning 6-19
1.3.1 Work Breakdown Structure (WBS) 6-20
1.3.2 PERT (Program Evaluation and Review Technique), CPM (Critical Path Method) and Gantt chart 6-22
1.3.3 Scheduling 6-27
1.3.4 Creation of project plans 6-27
1.3.5 Metrics 6-29
1.4 Managing Requirements 6-30
1.5 Risk Analysis and Management 6-31
1.6 Software Configuration Management (SCM) 6-32
1.7 Software Quality Assurance (SQA) 6-34
1.8 Project Management Software 6-35

Part 7 Additional Exercises

1. Exercises

Exercises for No.3 Part 1 (Software Engineering) 7-1
Exercises for No.3 Part 2 (External Design) 7-7
Exercises for No.3 Part 3 (Internal Design) 7-8
Exercises for No.3 Part 4 (Program Design) 7-9
Exercises for No.3 Part 5 (System Operation and Maintenance) 7-11
Exercises for No.3 Part 6 (Project Management) 7-13

2. Answers and Descriptions

Answers for No.3 Part 1 (Software Engineering) 7-15
Answers for No.3 Part 2 (External Design) 7-28
Answers for No.3 Part 3 (Internal Design) 7-30
Answers for No.3 Part 4 (Program Design) 7-34
Answers for No.3 Part 5 (System Operation and Maintenance) 7-40
Answers for No.3 Part 6 (Project Management) 7-45

Part 1

SOFTWARE ENGINEERING

1 Overview of Software Engineering

Chapter Objectives

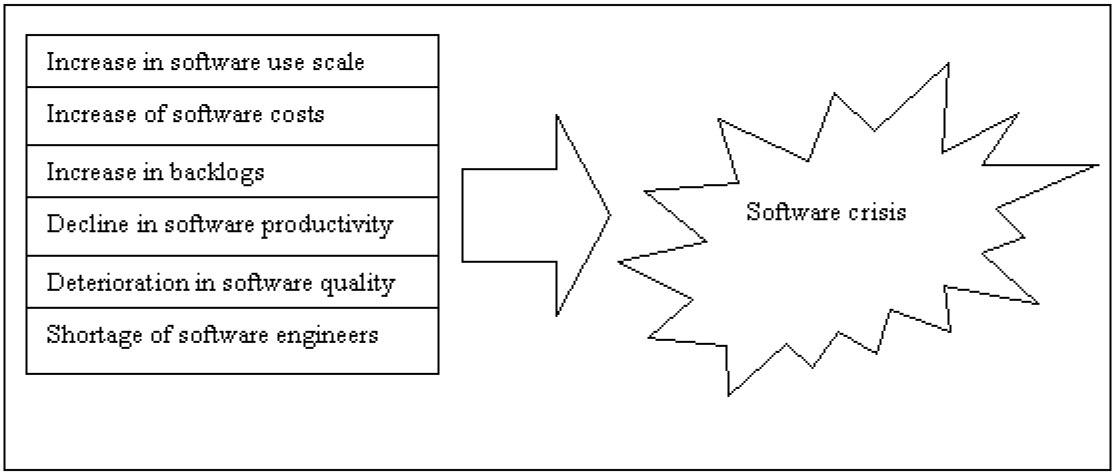
This chapter gives an overview of software engineering by presenting its birth and history.
It also explains what kind of engineering is defined as software engineering.
After studying this chapter, the student should be able to explain the methodologies and techniques that have been developed in this field.

1.1 Origin of Software Engineering
1.2 Definition of Software Engineering
1.3 Achievements in Software Engineering

1.1 Origin of Software Engineering

1.1.1 Software crisis

In the late 1960s, the use of mainframe computers increased, and mass production of software started. The resulting widespread application of software products brought about a variety of software-related problems. The emergence and continuation of these problems are called the software crisis. Software-related problems are summarized in the following figure:


Increase in software use scale
Increase of software costs
Increase in backlogs
Decline in software productivity
Deterioration in software quality
Shortage of software engineers
Software crisis

(1) Increase in software scale

As the use of mainframe computers increased and the scale of use of software products became larger, it became almost impossible to manage software manually.

(2) Increase of software costs

In relation with (1), as the scale of software products had increased, the costs for software development and maintenance have also increased.

(3) Increase in backlogs

The term "backlog" means unfinished development. Backlog in software development means a delay in the development of a new software product. Such delays are caused by increases in the workload for maintenance of software products.

(4) Decline in software productivity

Most of the work in any process of creating products depends on the experience of personnel. A lack of experience leads to a decrease in software productivity. As a result, it becomes more and more difficult to match new requests for system development.

(5) Deterioration in software quality

The creation of software products depends mostly on the work of people. As a result, the quality of software products can be unstable.

(6) Shortage of software engineers

As mentioned in (1) to (5), the expansion of use of the software products means that more human resources are required. Moreover, a deterioration of productivity and reliability of software products leads to a greater workload and higher costs because of increasing maintenance workload. At the same time, use of computers has been increasing, and the demand for software products has also been increasing. Consequently, this has led to a shortage of human resources for software development.

For these reasons, the software industry is in a crisis, and the awareness for the related problems has been growing.

This crisis of the software industry has triggered research on techniques to solve the related problems. The rest of this section describes the related developments in chronological order.

1969

IBM announced its unbundling policy.
Because of the reversal in the ratio of hardware to software costs, software products had to be priced separately. The entire computer industry then started to realize the value of software products.

Mid-1970s

The field of software engineering became widely recognized, and established itself in the industry.
After the first International Conference on Software Engineering (ICSE) was held in 1975, many different theories, methodologies, and techniques related to software engineering have been proposed.

1980s

A more wide-spread industrialization of software production became the target of the industry. The•(sigma) project started in Japan in 1985 to develop tools to aid productive software development.

1990s

Efforts were made to downsize computer resources and decentralize the software development environment. Computer aided software engineering (CASE) tools became popular for developing software products.

As can be seen, many different techniques have been proposed and discussed after the software crisis occurred. However, the problems related to the reliability of testing techniques and software evaluation have not been completely solved yet.

1.2 Definition of Software Engineering

1.2.1 Software production

In software engineering, software production involves both academic fields that are based on science, including the methodologies and techniques of computer science, and actual business domains.

The term methodology means a fundamental theory or concept on creating software. A technique is a technical method or procedure based on a theory.

By incorporating science, software engineering follows an engineering approach to software production.

1.2.2 Industrializing the management process

The software engineering industry aims to support software production and management from the engineering point of view. The first step of these efforts is an implementation of computer aided software engineering (CASE).

CASE uses computers to support software development and maintenance for the purpose of reducing reliance on the experience of personnel on one hand and improving productivity and reliability on the other hand.

1.3 Achievements in Software Engineering

1.3.1 Methodology

Methodologies proposed in the 1960s and 1970s are explained in the following table.

Methodology	Explanation
Top-down approach	Approach concretizing from the top level to the bottom for a system which has a hierarchical structure.
Data-oriented approach	Approach analyzing input and output data, and clarifying the relationship between them to determine the appropriate function to be implemented by software
Object-oriented approach	Approach combining data and processes into objects. The system is analyzed with a focus on the relationships between objects
Stepwise refinement	Approach refining step-by-step through hierarchy for a system which has a hierarchical structure
Abstraction	Approach abstracting the most distinctive element from an external in order to discover its characteristics
Information hiding	Approach hiding unnecessary information from mutually related components to improve data independence
Partitioning and integration	Approach determining software components, and integrating these components for operation in a system
Modularization	Approach partitioning software components hierarchically into modules, which are the minimum units of processing
Modularity	Approach treating software components as parts of a hierarchical structure

1.3.2 Techniques in Software Engineering

The following table explains software engineering techniques proposed in the 1960s and 1970s.

Technique	Explanation
Formal specification description	Technique to describe software specifications based on certain formal rules
SADT	Technique to define a model of requirements specifications based on structured analysis
PSL/PSA	System consisting of PSL, which can be used by a computer, and PSA, which is used to verify PSL descriptions with a computer.
SREM	System that is used to describe processing by real-time systems
Abstract data types	Technique to characterize data processing by defining data and procedures
Partial function partitioning method	Technique based on a principle that software functions can be divided into multiple smaller parts
Structured analysis method	Technique to transform a structured specification using DFD, a data dictionary, and mini-specifications.
Dijkstra method	Technique to convert a data structure step-by-step in a top-down approach
Structured design/composite design	Technique to convert a program into a hierarchical module structure
Jackson method	Technique to determine the algorithm of a program according to the input data structure
Warnier method	Technique to determine the algorithm of a program according to the mapping relationship between input and output data

Technique	Description
Structured theorem	Theorem that all algorithms can be described by using three types of basic control structures
Structured programming	Programming technique minimizing use of the goto statement
Structured chart	Algorithm diagram supporting structured programming
Structured coding	Technique of structured programming
Top-down programming	Technique of structured programming in which programming starts from the upper module
Module integration test	Technique for testing a program while integrating multiple modules
Black box test	Technique not to pay attention to the internal specification and to check whether specified functions are executed correctly or not
White box test	Technique to look at the internal specifications and to verify that a program runs according to the algorithm
Static and dynamic test	Technique which includes desk checking and testing with the computer

1.3.3 Logic-oriented paradigm

The logic-oriented paradigm describes the "logic" of relationships among characteristics and attributes of a target event by using facts and inference rules represented by predicate propositions. It is based on logic programming languages such as Prolog.

1.3.4 Function-oriented paradigm

The function-oriented paradigm describes a "function" that returns an output value corresponding to an input value. It is based on functional programming languages such as Lisp.

1.3.5 Object-oriented paradigm

The object-oriented paradigm describes the states and behaviors of an object. It is based on object-oriented programming languages such as Smalltalk, C++, and Java.

1.3.6 Agent-oriented paradigm

The agent-oriented paradigm is an extended type of the object-oriented paradigm. It has been attracting wide-spread attention in the industry. It develops a system with an agent, which is a software module, as a basic unit. An object starts a procedure passively upon receiving a message from outside. In contrast, an agent works actively according to the environment in which it is placed.

Exercise

1. The following paragraph is a text about the software crisis. Fill in each blank by choosing the correct phrase from the list below.

   In the late 1960s, the use of mainframe computers increased, and mass production of software started. The resulting widespread application of software products brought about a variety of software-related problems. The causes behind this software crisis are an expansion in the scale of software use, [***(1)***], [***(2)***], a decline in software productivity, a deterioration in software quality, and a shortage of software engineers.

   In software development, the term [***(2)***] refers to a delay in the development of a new software product.

   Answers:
   a. hardware crisis
   b. increase in backlogs
   c. increase of software costs
   d. software defects

2. The following paragraph is a text about software. Fill in each blank by choosing the correct phrase from the list below.

   Software used to be provided free of charge until IBM announced its [***(1)***] policy in 1969. After this announcement, software and hardware products have been priced separately.

   In 1975, [***(2)***] was held to provide an opportunity for an international discussion about software engineering.

   Answers:
   a. Capital liberalization
   b. TRON
   c. unbundling
   d. CASE
   e. IEEE
   f. ICSE
   g. SIGMA
   h. ISO

2 Process Models and Cost Models For Software Engineering

Chapter Objectives

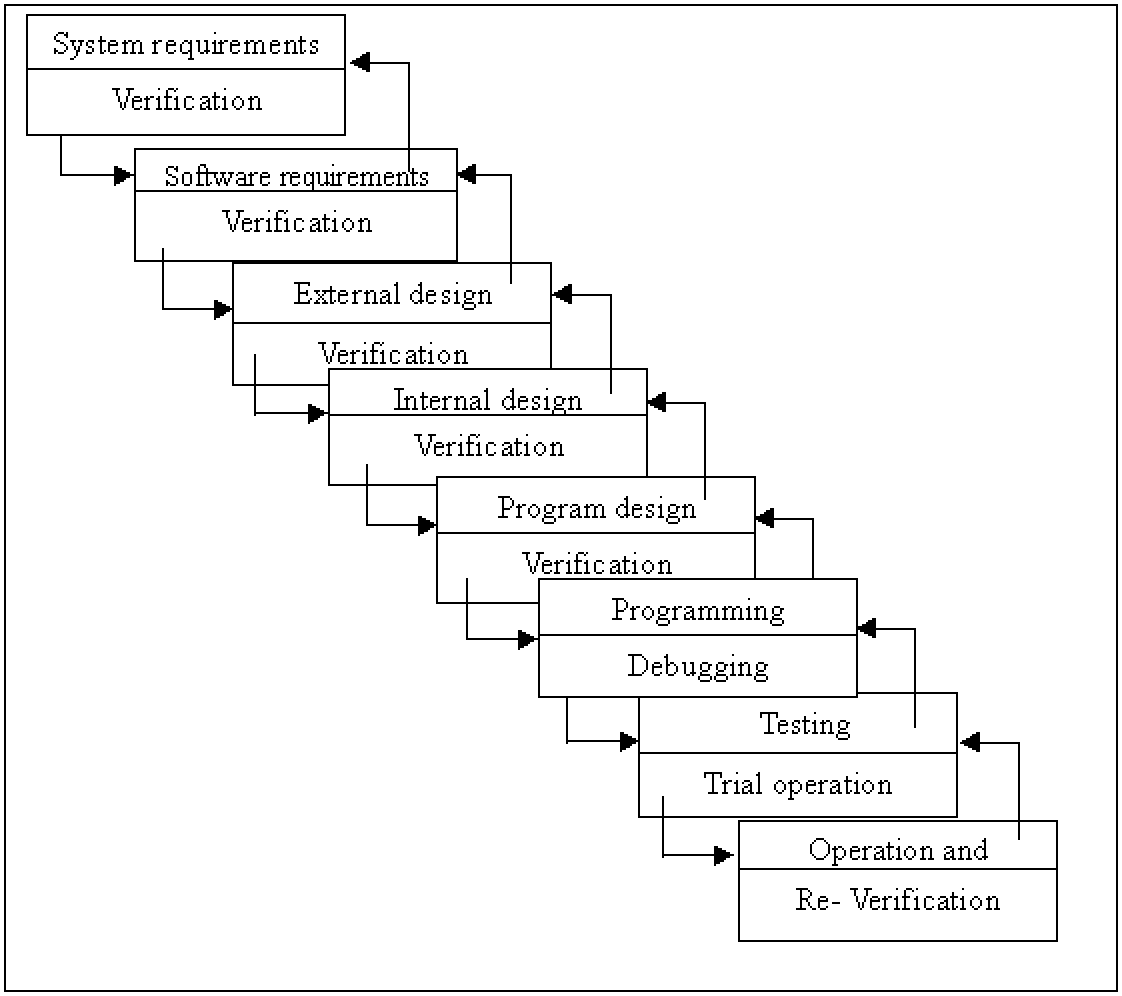
This chapter explains representative process models and cost models for software development and their characteristics. The goal of the study is to develop capability of proposing the optimum process model for the software development projects you participate in and the ability to explain the characteristics of cost models for software development.

2.1 Process Models for Software Development
2.2 Cost Models for Software Development

2.1 Process Models for Software Development

2.1.1 Waterfall model

This model divides software development into processes to be implemented sequentially from upstream to downstream. The implementation proceeds in one direction.
This model is shown below.



Advantages and disadvantages of the waterfall model are as follows.

Advantage	Suited for relatively large-scale development projects.
	Because the project is divided into processes, it is easier to manage the development.

Disadvantages	Whether the user requirements will be met is not known until late stages of development.
	Does not allow much flexibility for accommodating changes in specifications.

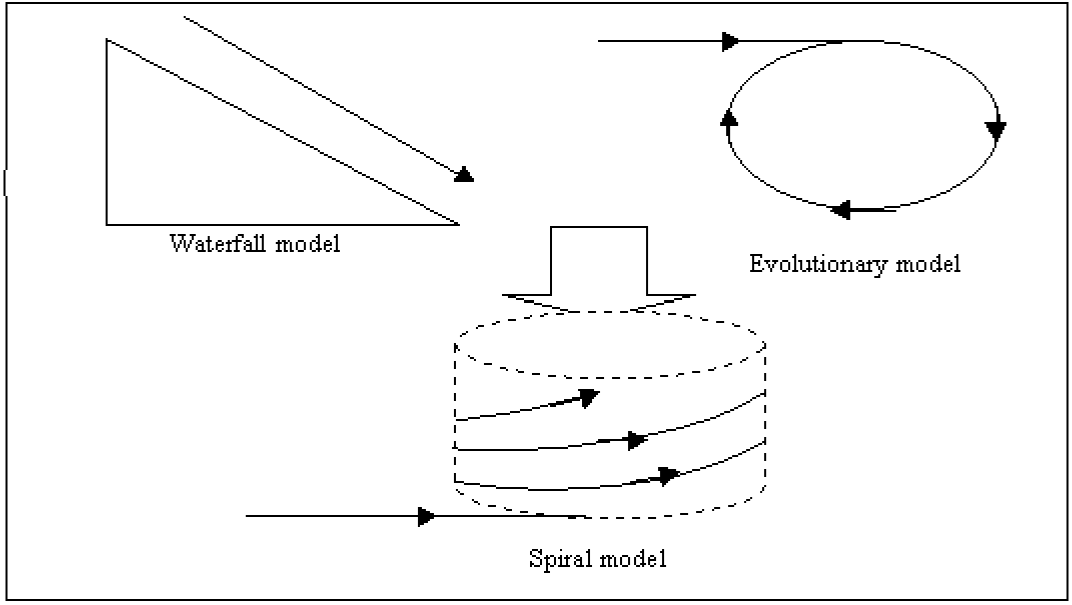
The waterfall model has been the most commonly used model since the birth of software engineering.

2.1.2 Spiral model

This process model is a combination of the waterfall model with the evolutionary model. As discussed in the previous section, software development proceeds in one direction from upstream to downstream with the waterfall model, With the growth model, however, development is implemented through cyclically growing processes. Advantages and disadvantages of the evolutionary models are as follows:

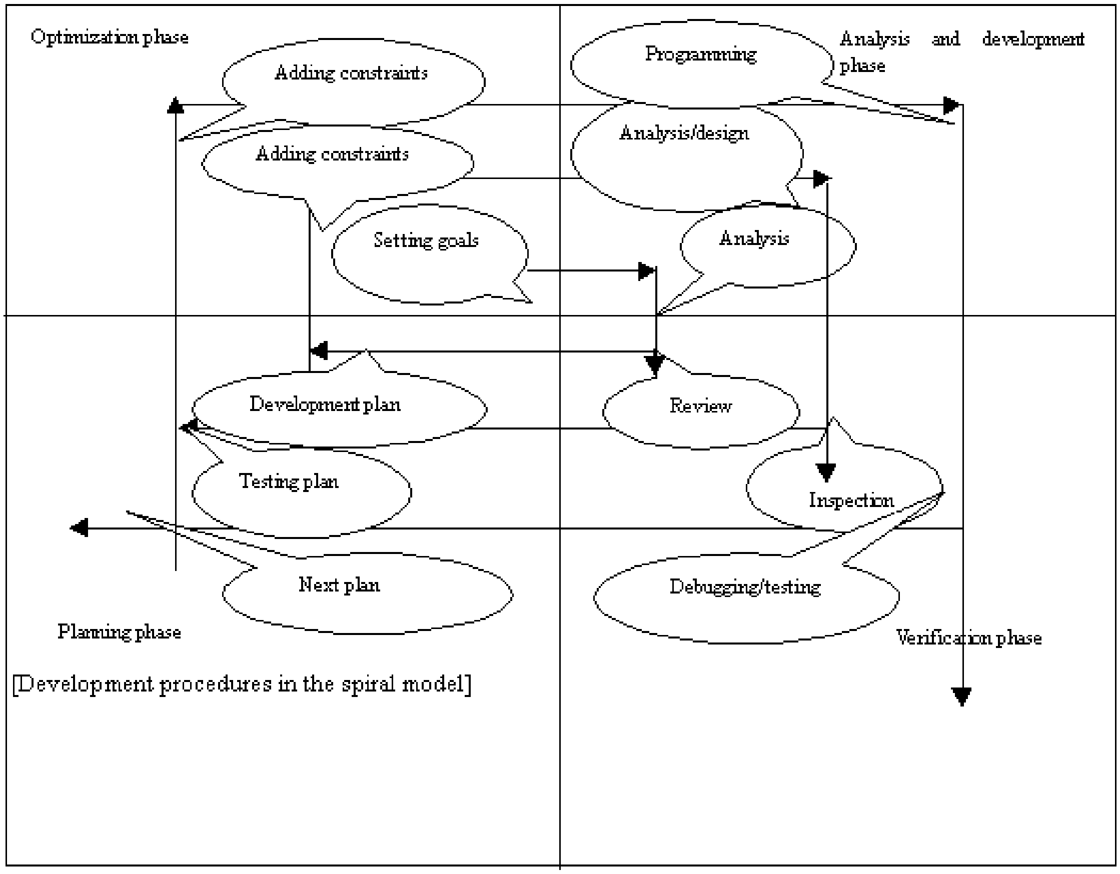
Advantage	Flexible in the sense that development can accommodate changes in specifications.
Disadvantage	Because this model includes repetitions of processes, it may be difficult to manage the project.

The spiral model has combined characteristics of the waterfall model with those of the growth model. What this means is that software development proceeds in one direction but goes through similar processes in cycles. This model is shown below and in the next page.



[Structure of spiral model]

[Development procedures in the spiral model]



[Development procedures in the spiral model]

2.1.3 Prototyping model

The prototyping model is a process model in which prototypes are created in early stages of software development. It is used by both the developer and the client to examine the requirements specifications for the system. In this way, the developer and the client can eliminate differences in their respective views concerning what they are trying to create.

One purpose of using this model is to ensure that the user's requests are met in system development by demonstrating with prototypes the required functions to be created at early stages of development.

Prototypes used with the prototyping model can be divided into those created and used by the developer to examine the development, and those used by both the developer and user to evaluate the functions of the system under development.

Classifications of prototypes		Description
Classification by purpose	Trial creation	Prototypes for components that are difficult to design. These are created to help obtain information required for design.
	Checking specs	Prototypes created to check the user's requirements in early stages (of defining requirements or analyzing processes)

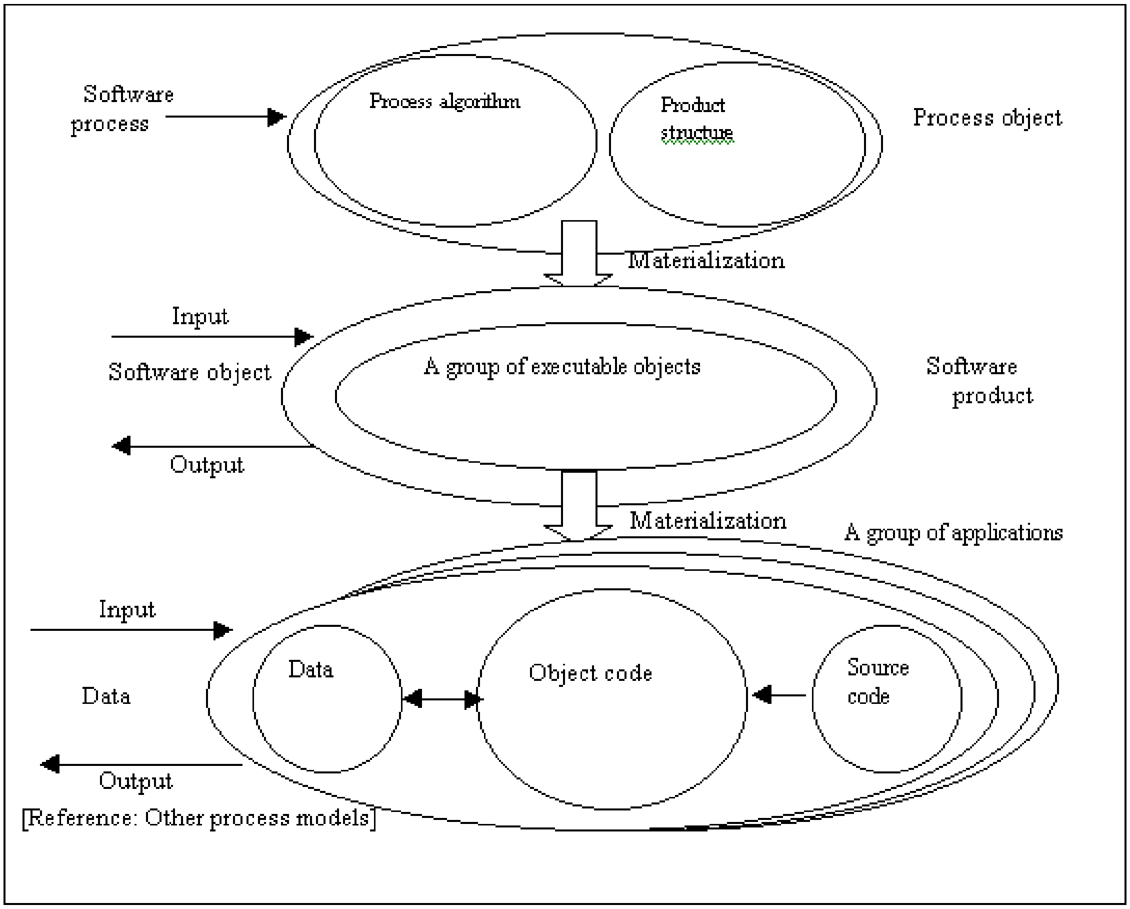
Classification by use	Thrown away	Prototypes are created and thrown away. The final product is created independently.
	Evolutionary type	Prototypes created are not thrown away, but are developed as a base for the final product. They are created in the same environment as the final product.

The prototyping model can have the following positive effects in system development.

(1) It allows to construct a system with more effective functions, because it helps to find out the user's potential needs.
(2) Because it allows to verify the user's exact needs in early stages of system development, many changes in the specifications that would otherwise be necessary in later stages can be avoided.
(3) Because the functions, performance and operability can be checked during system development, the total time for development can be shortened.

We have so far discussed the process models, which focuses on how to incorporate software development processes in models. There is another way of developing software called "process programming." Process programming focuses on the processes of software development, and defines each programming process in precise descriptions in the specification. With this method, each software product is understood as a software object.

In process programming, the first step is creating a process object. The process object consists of process algorithm and product structure. The process algorithm prescribes the steps to follow in creating objects, and the product structure is a model for objects that shows the structure of an object in detail. Software development by process programming is shown in the following.



[Reference: Other Process models]

Contract model
With this model, an agreement is concluded for each of the software development processes. To define requirements, for example, the party submitting the development order informs the party accepting the order of the client's requirements for the system to be developed. The party accepting the order then makes a proposal to the party submitting the order in accordance with such requirements. With this model, both parties conclude beforehand a formal agreement for each software development process.

2.2 Cost Models for Software Development

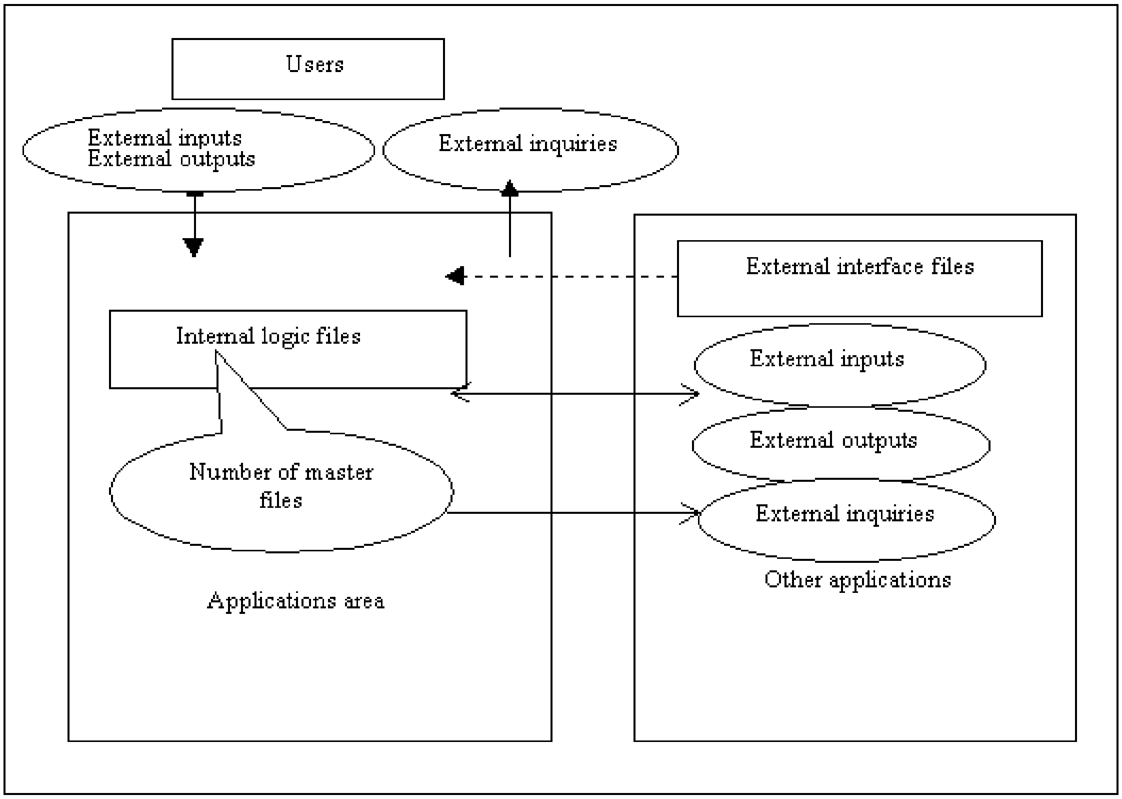
2.2.1 Halstead model

This is a model for estimating the quantity of intellectual work involved in programming by analyzing the source code of a program.

2.2.2 FP (Function Point) model

The FP model estimates the size of software by analyzing the sizes in units of function, for the functions included in the software. This model is based on the quantity of information that the client needs to have, so that it can produce estimates from the client's point of view.

This model distinguishes each of the functions of software on the basis of the number of inputs, the number of outputs, and the number of master files. Moreover, it considers the difficulty or ease of development of each file. The concept of this model is shown in the following:



The steps for an estimate with the FP model are as follows.

First, identify the functions of software. The FP model categorizes such functions into five types, which are listed in the following:

Function type	Description
External output (A1)	The number of points determined on the basis of the type and total quantity of input data
External input (A2)	The number of points determined on the basis of the type and total quantity of output data
External inquiry (A3)	The number of points determined on the basis of the type and total quantity of inquiries from the client
Internal logic file (A4)	The number of points determined on the basis of the type and total quantity of files which are accessed
External interface file (A5)	The number of points determined on the basis of the type and total quantity of interfaces to other systems

The function points (FP) of software can be obtained by using the points of these five function types as follows:

FP = C x (A1 + A2 + A3 + A4 + A5)
(C is a factor)

The characteristics of the FP model are as follows:
(1) FP is independent of the development language, so that the points can be used as a measure of improvement in terms of productivity or quality in software development.

(2) FP is not a measure for the quantity of development; it is a measure for the quantity of functions to be developed. Accordingly, it can indicate a value that is meaningful from the viewpoint of the user.

2.2.3 COCOMO (COnstructive COst Model) model

This model calculate effort of software based on statistical data, KDSI(thousands of delivered source instructions), and unit cost of a programmer by incorporating the difficulty of development into KDSI.

KDSI, which represents the size of programs, is calculated by using the following equation.

KDSI = total source lines of code - number of comment lines + number of lines in JCL (job control language)

With KDSI, the effort in person months is calculated as follows.

Effort in person months = a x $(\text{KDSI})^b$ x c

where a, b, and c are constants that are obtained based on some criteria and statistical data.

Criterion	Description
Level of estimate	Rough, ordinary, or detailed
Personnel	Highly skilled only, ordinary, or mixed staff

The duration of development is calculated as follows.

Development duration = 2.5 x $(\text{effort in erson months})^b$

A characteristic of the COCOMO model is to have three levels of accuracy for the estimate: low, middle, and high. These levels are described below.

Level of precision	Description
Low (basic)	Estimate effort in person months solely on estimated program size.
Middle (intermediate)	Improve accuracy by using 15 attributes in addition to KDSI.
High (detailed)	Applies the intermediate version at the module level, and then uses phase-based model is used to build up an estimate of the complete project.

The fifteen attributes affecting development, to be used in the "intermediate" level are listed in the following.

Affecting factor	Attribute
Software product	Required software reliability
	Database size
	Complexity of the software product
Computer	Execution time constraints
	Main storage constraints

	Frequency of changes of OS and/or hardware
	Program development environment
Personnel	Capability of analysts (designers)
	Experience with similar applications
	Capability of programmers
	Experience in OS and/or hardware
	Familiarity with the programming language used
Process	Use of the software development method
	Use of software tools
	Required development schedule

As can be seen from the table, the COCOMO model sets importance on the capability and organization of the development personnel and organization, as well as on the software and hardware.

Exercise

1. Choose the model that corresponds to each of the following descriptions from the answer choices.

(1) This process model has aspects of both the waterfall model and evolutionary model. This means that the software is developed to the final product in one direction, going through similar processes in cycles.

(2) Software development proceeds along with a single line from upstream to downstream, so that it is easy to obtain the entire picture of the development. This makes it easy to manage large-scale projects.

(3) The software development processes are repeated every time there is a change in the specifications. The software is developed through such repetitions.

(4) The software development process is divided into units of work, and an agreement is concluded for each unit of work, between the party submitting the order and the party accepting the order, so the development will proceed.

Answers:

a. Contract model

b. Spiral model

c. Waterfall model

d. Evolutionary model

2. Choose the answer that corresponds to each of the following descriptions on the cost models for software from the answer choices.

(1) The quantity of intellectual work in software development is obtained by analyzing source code.

(2) The total size of software is estimated by analyzing the size of each software function. Accordingly, the size is estimated from the client's point of view.

(3) The programmers' effort in person months in software development is obtained on the basis of statistical data and KDSI.

Answers:

a. Halstead model

b. COCOMO

c. Function Point model

3 Defining Software Requirements

Chapter Objectives

This chapter explains the requirements for software, typical examples of software requirement analysis, features of software requirement models, and object-oriented analysis. It also explains how to determine which software requirement model to adopt and how to use structured analysis to create software specification requirements.

3.1 Software Requirements
3.2 Software Requirements Analyzing
3.3 Software Requirement Models
3.4 Techniques for defining Software Requirements

3.1 Software Requirements

Software requirements are customers' requirements for software. Following five points should be considered to meet the software requirements:
(1) Why is systematization necessary?
(2) What is the configuration of the system?
(3) What are the functions necessary for implementing the system?
(4) What is the required level of system performance?
(5) What are the restrictions on developing a system?

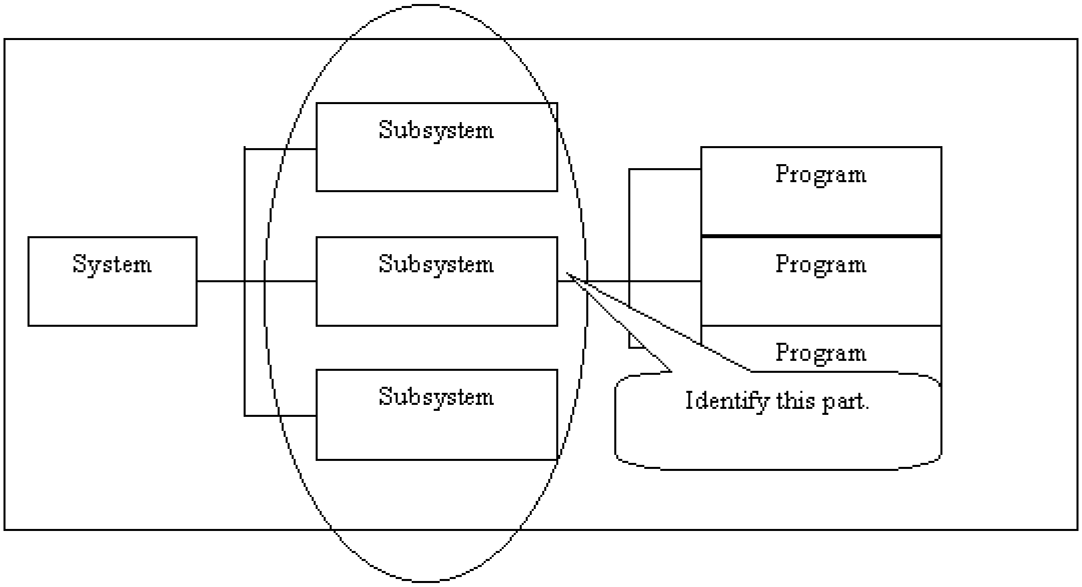
The following sections explain each of these items.

3.1.1 Purpose of systematization

The purpose of systematization can be viewed from two standpoints: that of a customer and that of a developer. From a customer's standpoint, systematization aims to improve performance in the application in terms of time and money.
Improved performance in time means less time is required for tasks, because of the introduced system. Improved performance regarding costs refers to reduced costs by introducing a system. Performance concerning both time and cost is called system efficiency. In contrast, from a developer's standpoint, systematization aims to meet customer requirements. That is, system efficiency is adjusted to suit customers' needs.
However, increasing performance in time leads to decreasing performance in costs and vice versa. It is thus important that the customer and the developer have agreed understanding.

3.1.2 Configuration

After the purpose of systematization is clarified, requirements for the system configuration must be identified and clarified. In this process, subsystems and functions required in the system configuration are identified based on the requirements for system implementation. The following figure shows a system configuration:



3.1.3 Function

Function is an effect of implementing a system, and it refers to criteria used to determine the capabilities of a system. If the criteria of a function are unreasonable, performance generally declines.

3.1.4 Performance

Performance is a system utility, and it refers to criteria for determining the extent of system performance. If unreasonable demands are placed on performance, functions are generally restricted. The effect of implementing a system is evaluated based on the functions and performance. The following table summarizes the effect of the system:

Effect of the system	
Type	Description
Function	Evaluation criteria for what a system can do
Performance	Evaluation criteria for the extent of the performance of a system

3.1.5 Constraints

Constraints are various agreements on systematization. Common examples are financial constraints, time constraints, legal constraints, and technical constraints.

3.2 Software Requirements Analysis

Software requirements analysis refers to a clarification on how to represent customer requirements after first identified. The following sections explain the KJ method, functional analysis, event response analysis, and structured analysis with some examples.

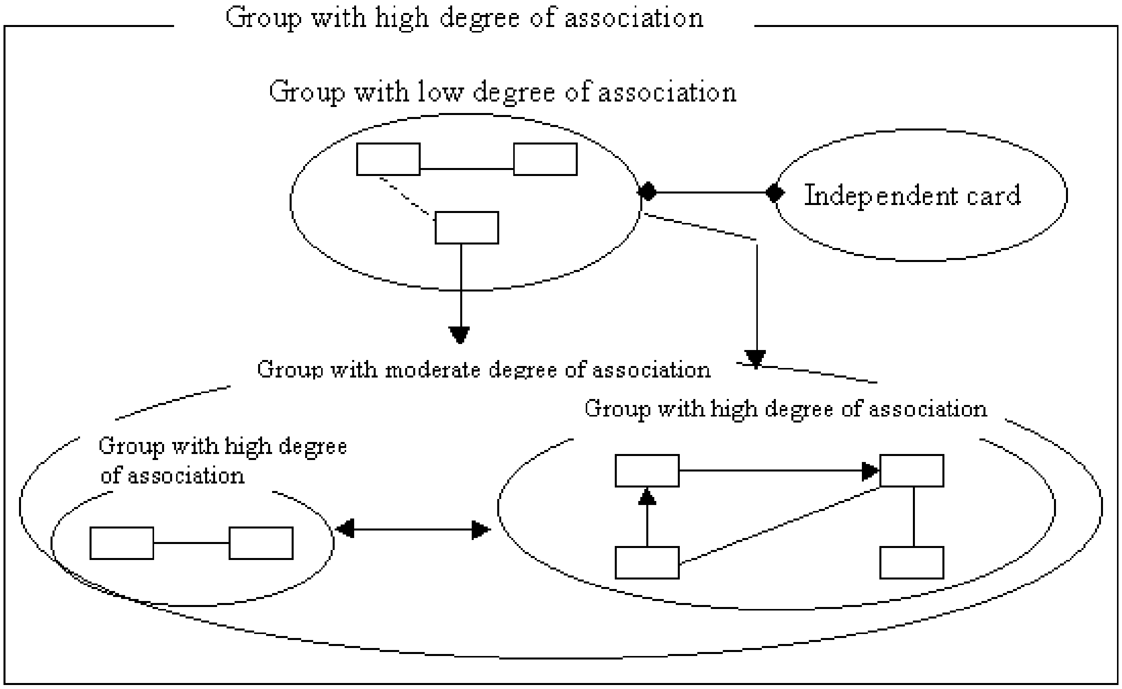
3.2.1 KJ (Jiro Kawakita) method

The KJ method is a software requirements analysis method proposed by Jiro Kawakita. This method supports the formulation of new ideas out of initial ideas, where the new ideas are derived from conceivable ideas for particular events. This method is suitable for determining requirements based on results from the current-status-oriented approach. In the current-status-oriented approach, existing problems are already known and a solution to the problems is sought.

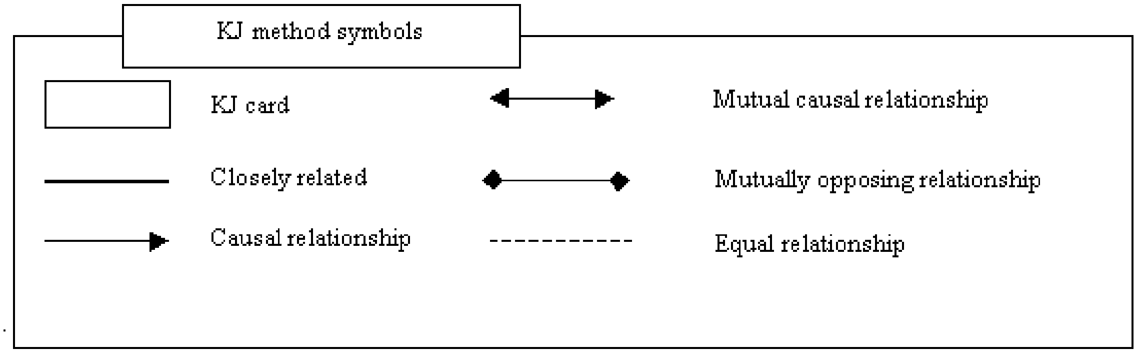
The procedure of the KJ method is as follows:

(1) A moderator is selected.
(2) The moderator points out existing problems.
(3) Each participant thinks about possible solutions to the existing problems pointed out by the moderator.

(4) Each participant writes their solutions devised in step (3) on distributed cards.
(5) The cards with the written solutions are collected and then grouped, so that solutions of the same categories are together. This makes the relationships among groups clear. Note that solutions that do not belong to any group are handled independently instead of forcibly designating such solutions as the belonging to specific groups.


Group with high degree of association
Group with low degree of association
Independent card
Group with moderate degree of association
Group with high degree of association
Group with high degree of association

(6) Create a group relationship diagram, and show it to the participants.



(7) Consider solutions from other standpoints again based on the group relationship diagram, and then repeat step (4) to (7).

That is the last step in the KJ method. Note that this method requires the participation of customers as well as developers. That is, it is clear in the KJ method enhances the emergence of new ideas by allowing developers and customers to exchange their opinions at the same place from different standpoints.

3.2.2 Functional analysis

Functional analysis is an analysis method for identifying system functions in terms of both input and output data. That is, relationships among the four elements of input data, output data, functions, and conditions are defined. Functional analysis is applied to data flow models for software requirement models.

3.2.3 Event response analysis

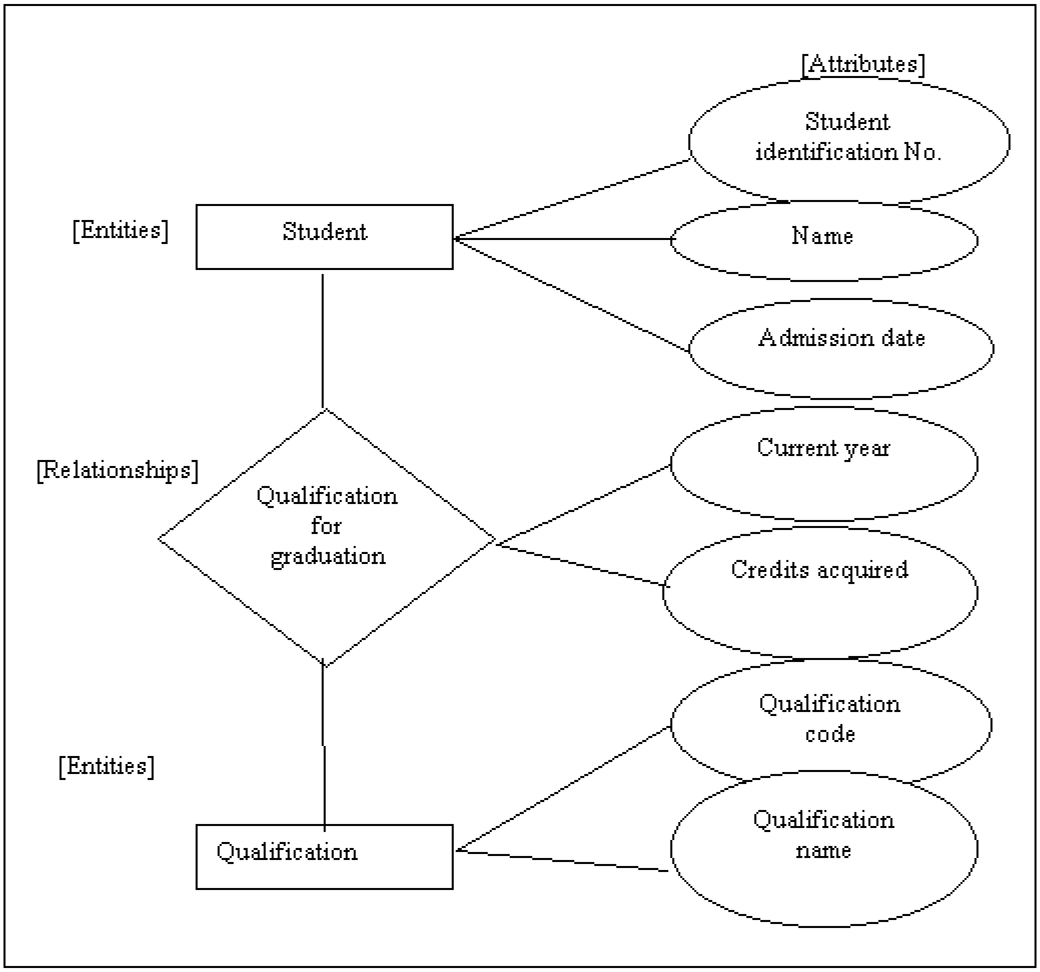
Event response analysis is a method of analyzing system responses to events from outside the system with the passage of time. Event response analysis is applied to the control flow model, finite-state machine model, and Petri-net model for software requirement models. For information about the control flow model, finite-state machine model, and Petri-net model, refer to Section 3.3, "Software Requirement Models."

An example of relationships between events and responses can be found in GUI behaviors. An event in a GUI is created by clicking an icon, and clicking the icon starts a procedure encapsulated with the icon.

3.2.4 Structured analysis

Structured analysis refers to creating graphs that represent model entities with nodes and their relationships with arrows. With this analysis method, the structures of groups of entities can be clarified by analyzing external models and creating associated graphs. A typical example of this method is clustering. This is a method for classifying components of a model by grouping them while focusing on their similarities.

Structured analysis has been applied to the ER model in the schema design model of databases. The ER model uses a diagram based on concepts of entities and their relationships, and external targets are abstracted for modeling. In the ER model, entities are represented with rectangles, relationships are represented with diamonds, and attributes are represented with ellipses. Attributes refer to the properties of the kind of components that compose entities and relationships. The following figure shows an example of the ER model:



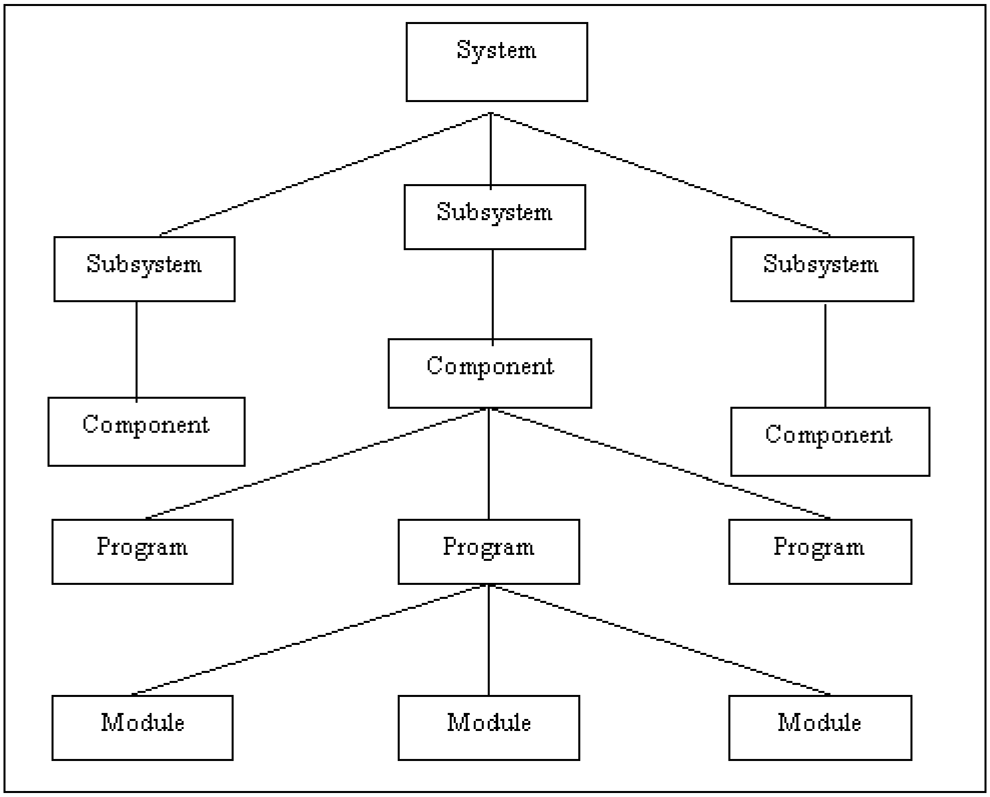
3.3 Software Requirement Models

This section explains the software requirement model, which reflects results of a software requirement analysis. The following types of models are explained as typical software requirement models in the following sections: functional hierarchical model, data flow model, control flow model, finite-state machine model, Petri-net model, data-oriented model, object-oriented model, and parallel-process model. Examples of the models are given.

3.3.1 Functional hierarchical model

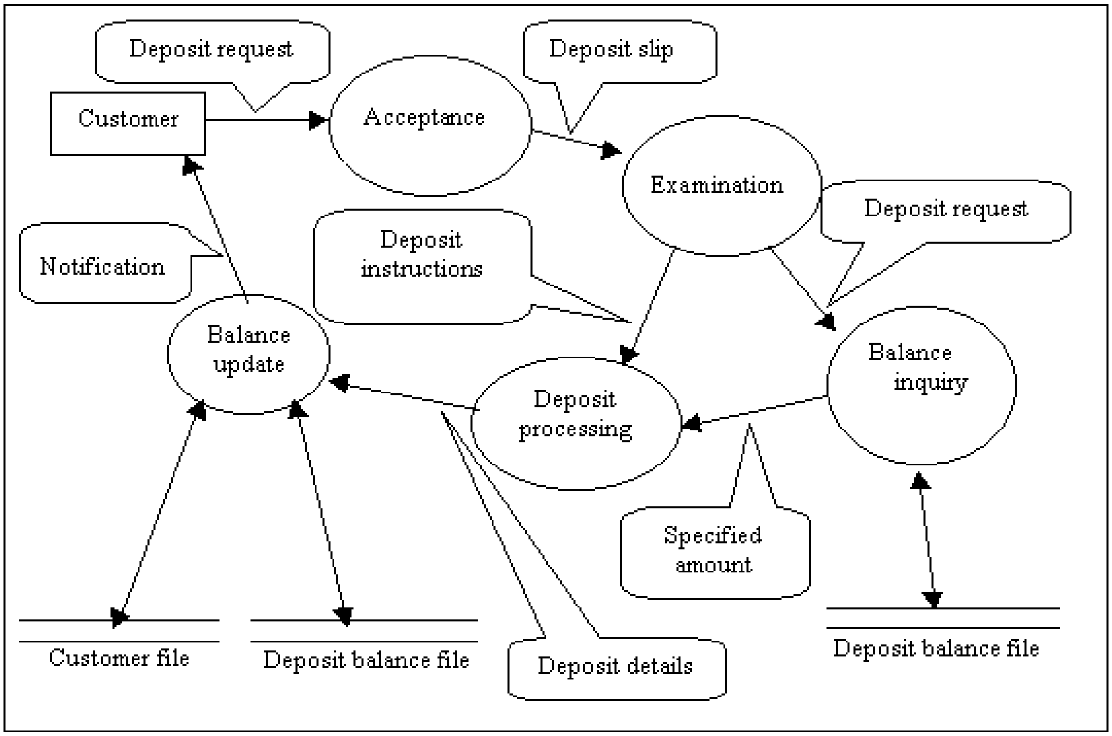
The functional hierarchical model is a software requirement model created in a hierarchical configuration by stepwise subdivision with a focus on system operations, which can be considered to be a function that converts input data into output data. As explained in Section 3.2.2, "Functional analysis," a system function can be defined as an operation to create output data from input data. In the functional hierarchical model, system functions are subdivided and then represented in a hierarchical structure, such as from a system to subsystems and from a subsystem to programs. The following figure shows an example of the functional hierarchical model:

The functional hierarchical model is created by subdividing the system for each function, as shown above. This model is the software requirement model most suitable for the waterfall life-cycle model.



3.3.2 Dataflow model

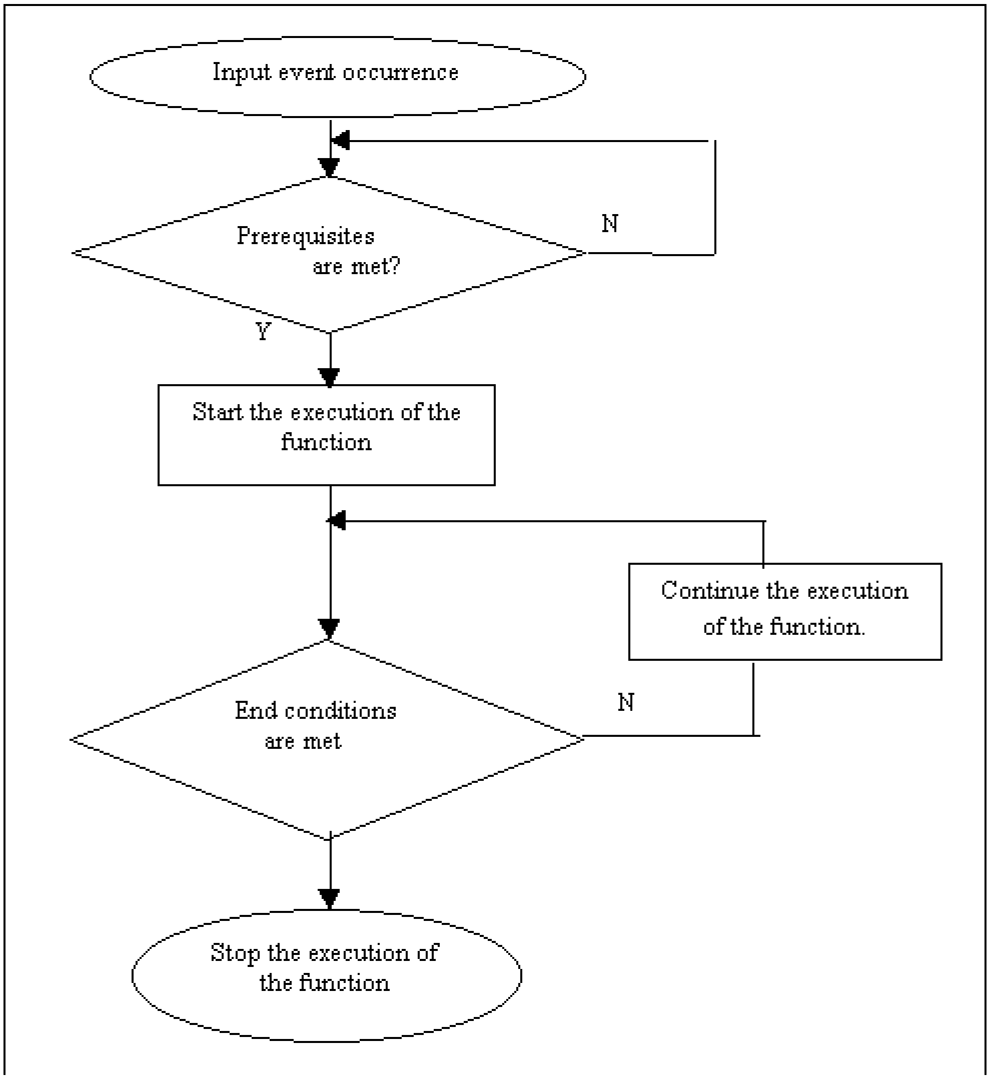
The dataflow model is a software requirement model based on functional analysis, and system functions are analyzed according to the relationships of input, functions, storage, and output. This model is suitable for analyzing requirements of application systems which are mainly business processing by flow of data. With the data flow model, models are developed by using a diagram called the data flow diagram (DFD). In this diagram, input data and output data are represented with arrows, each processing operation with circle, storage data with double lines, and entities (data source/sink) with rectangles. The following figure is an example of DFD:



DFDs may give impression of ease of understanding not only to developers but also to customers, leading to smooth communication.

3.3.3 Control flow model

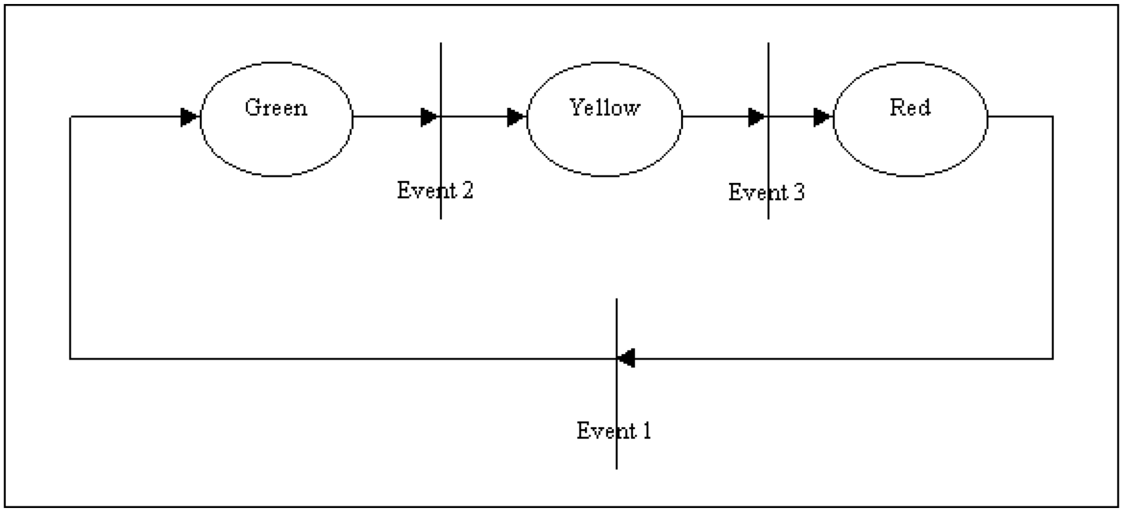
The control flow model is a software requirement model based on event response analysis. In this model, the system state and relationship between events depend on the passage of time. This model is most suitable for analyzing requirements of systems that require dynamic control. A typical example is a real-time traffic signal control system. In addition, events and functions are related in this model. The following figure shows relationships between events and functions.


Input event occurrence
Prerequisites are met?
N
Y
Start the execution of the function
Continue the execution of the function.
End conditions are met
N
Stop the execution of the function

Different kinds of control flow model have been proposed, Such as the logical control flow model and R net (Requirement Net: method of representing transaction flows in a network) adopted by the Software Requirement Engineering Methodology (SREM). SREM refers to a methodology for describing the requirement specifications for real-time systems.

3.3.4 Finite-state machine model

The finite-state machine model is a software requirement model based on event response analysis, and relationships between data and control are considered in order to represent them properly. This model is suitable for analyzing requirements of control systems.



3.3.5 Petri-net model

A Petri-net model is a software requirement model based on event response analysis. Because this model can represent synchronization of functions operating in parallel, it is suitable to express control-oriented system requirements analysis. Petri-net itself is a directed graph based on a binary tree, and it consists of two nodes and a direction arrow connecting them. The following figure shows an example of the Petri-net model.

The figure represents the operation of traffic lights. The figure shows the control that turns on the green light when event 1 occurs, and turns on the yellow light when event 2 occurs. Then, when event 3 occurs, the yellow light is turned off and the red light is turned on.

3.3.6 Data-oriented model

The data-oriented model is a software requirement model that focuses on data analysis. The basic ideas of this model are as follows:

(1) Software requirement analysis for a system can be conducted more smoothly by focusing on data handled by the system.
(2) By defining clear relationships among data, the functional configuration is also determined.

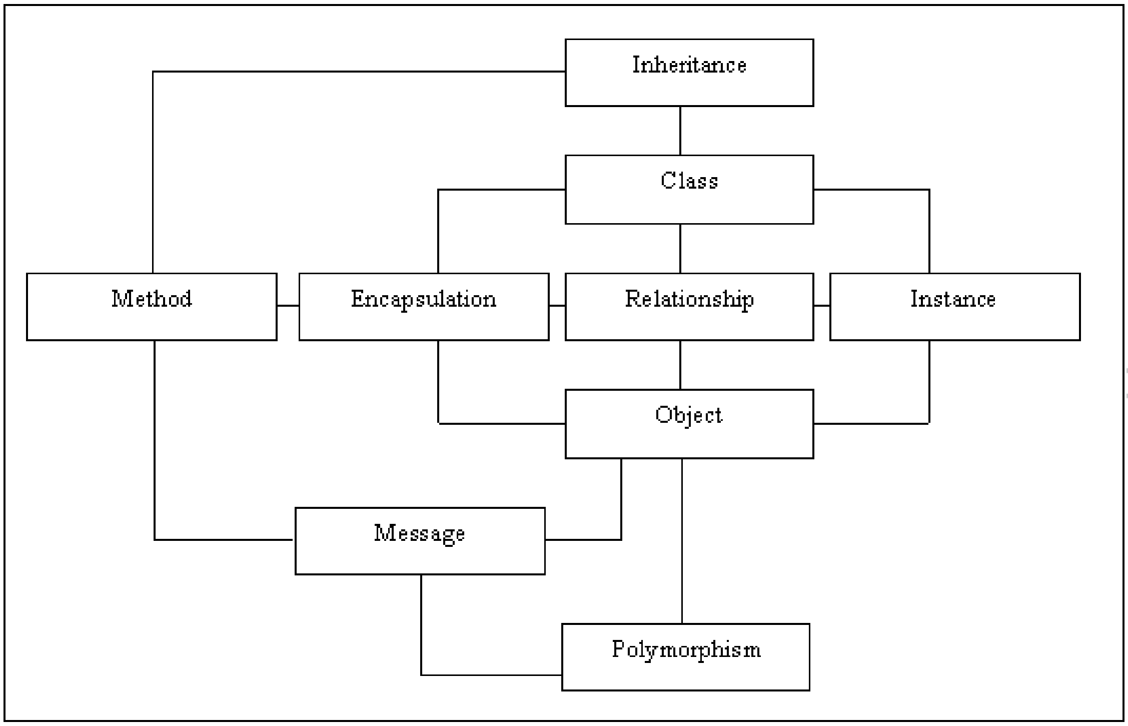
The basis of idea (1) is that the analyses of current jobs start in many cases with a close examination of input-output data. Also, existing input-output data is one of the most fundamental information.

The basis of idea (2) is that the functions required between input data and output data can be determined by mapping the relationships between input data and output data.

As explained, the data-oriented model is a software requirement analysis method focused on data.

3.3.7 Object-oriented model

The object-oriented model is a software requirement model which has evolved from the abstract data model, and the concept of class that indicates common features in a data structure is used for modeling. The abstract data model is a model that combines the structure of data and data operations that make up system processing targets. Objects in the object-oriented model behave actively, in contrast to the abstract data types. The following shows an object-oriented model.



[Object-oriented model]

Explanation of the terms used in the figure:

(1) Object
A code entity stored at an address within the storage area. An object is created when the related program is started, and it disappears when the program terminates.
(2) Class
From various objects those which have the same characteristics are grouped together and given a new name.

A class continues to reside in memory even after a program terminates.

(3) Instance
Embodiment of an object belonging to a class

(4) Method
Description of a behavior which an object has

(5) Encapsulation
Putting together data and behavior

(6) Message
Unit of request for activating the behavior of an object

(7) Relationship
Relationships between classes and between objects. More precisely, hierarchical relationships between classes are called an is-a relationship. The upper class is called a super-class, and the lower class is called a subclass.

Hierarchical relationships between objects are called a part-of relationship. The upper object is called a parent object, and the lower object is called a child object.

(8) Inheritance
Inheriting particular properties

(9) Polymorphism
In message exchange among objects, an inherent behavior defined in each object can be initiated, even if messages with the same name are sent to multiple child objects.

3.3.8 Parallel process model

The parallel process model is a model for events in which multiple processes operate in parallel simultaneously.

3.4 Techniques for Defining Software Requirements

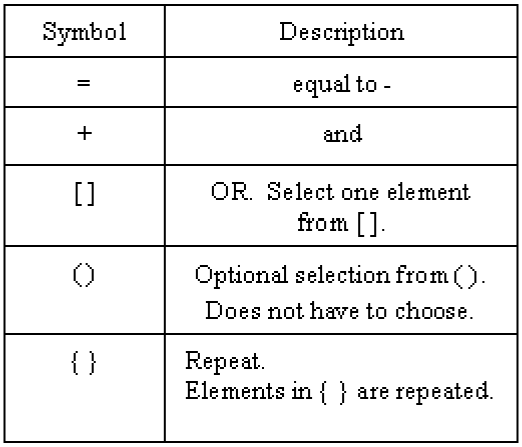
This section explains how to clarify and define software requirements. The structured analysis method and object-oriented analysis method are explained as examples of the software requirement definition.

3.4.1 Structured analysis method

The structured analysis method is a method for determining software requirements proposed by Tom DeMarco. The basic idea of this method is to define the system structure by identifying system functions according to the data flow and subdividing these functions into layers. The purpose of structured analysis is to create structured specifications using tools such as data flow diagrams, a data dictionary, and mini-specifications. Since the data flow diagram is explained in Section 3.3.2, "Data flow model", the data dictionary and mini-specifications are explained here.

(1) Data dictionary

The data dictionary is a dictionary created after system functions and data flows are clarified with the data flow diagram. This dictionary is created to determine the logical structure for each type of data for a common management approach. The data dictionary contains definitions of all files used by the system and data shown in the data flow diagram. Its contents follow the logical structures of data and files. These logical structures involve only the data structure and do not include data types and attributes. Backus-Naur Form (BNF) is used for creating the data dictionary, and the following table explains BNF notations.



Symbol	Description
=	equal to -
+	and
[ ]	OR. Select one element from [ ].
( )	Optional selection from ( ). Does not have to choose.
{ }	Repeat. Elements in { } are repeated.

(2) Minispecifications

Minispecifications defines the system functions clarified in DFDs. Note that minispec is created to cover not only normal processes but also exception handling and abnormal processes. It is also important to itemize them by function. Structured languages, decision tables, and decision trees are available as tools that can be used for creating minispec. Each of these tools is explained in the following:

1) Structured language
A structured language is a tool for creating minispecs in a subset of a natural languages based on structured specifications, where "structured specifications" mean that they have the following three constructions:

Construction	Description
Sequence construction	Configuration in which statements are connected in sequence
Selection construction	Configuration in which different statements are executed depending on conditions
Iteration construction	Configuration in which statements are repeated depending on conditions

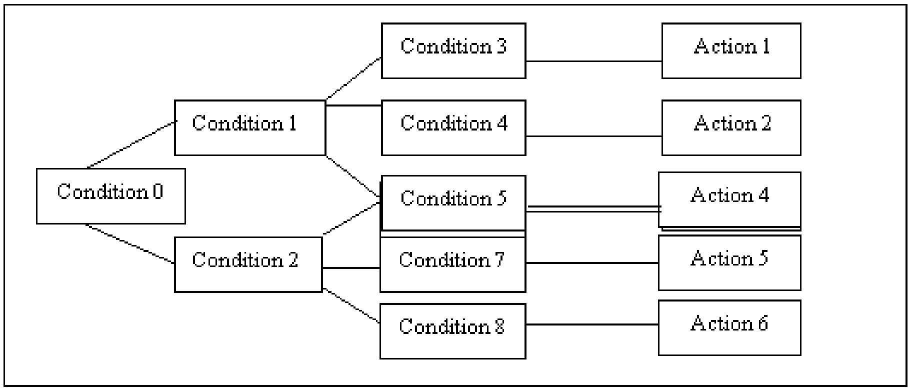
2) Decision table
A decision table is a tabular form tool used when complex conditions are combined. It consists of four matrices. The following figure shows an example of a decision table.

Condition	Case				
	1	2	3	4	5
1 .Condition 1	N	Y	Y	N	N
2. Condition 2	N	Y	N	N	N
Action	1	2	3	4	5
1. Action 1	Y	Y	Y	N	-
2. Action 2	Y	N	Y	N	N

Y : The condition occurs
N : The condition does not occur
- (hyphen): The condition does not occur

3) Decision tree

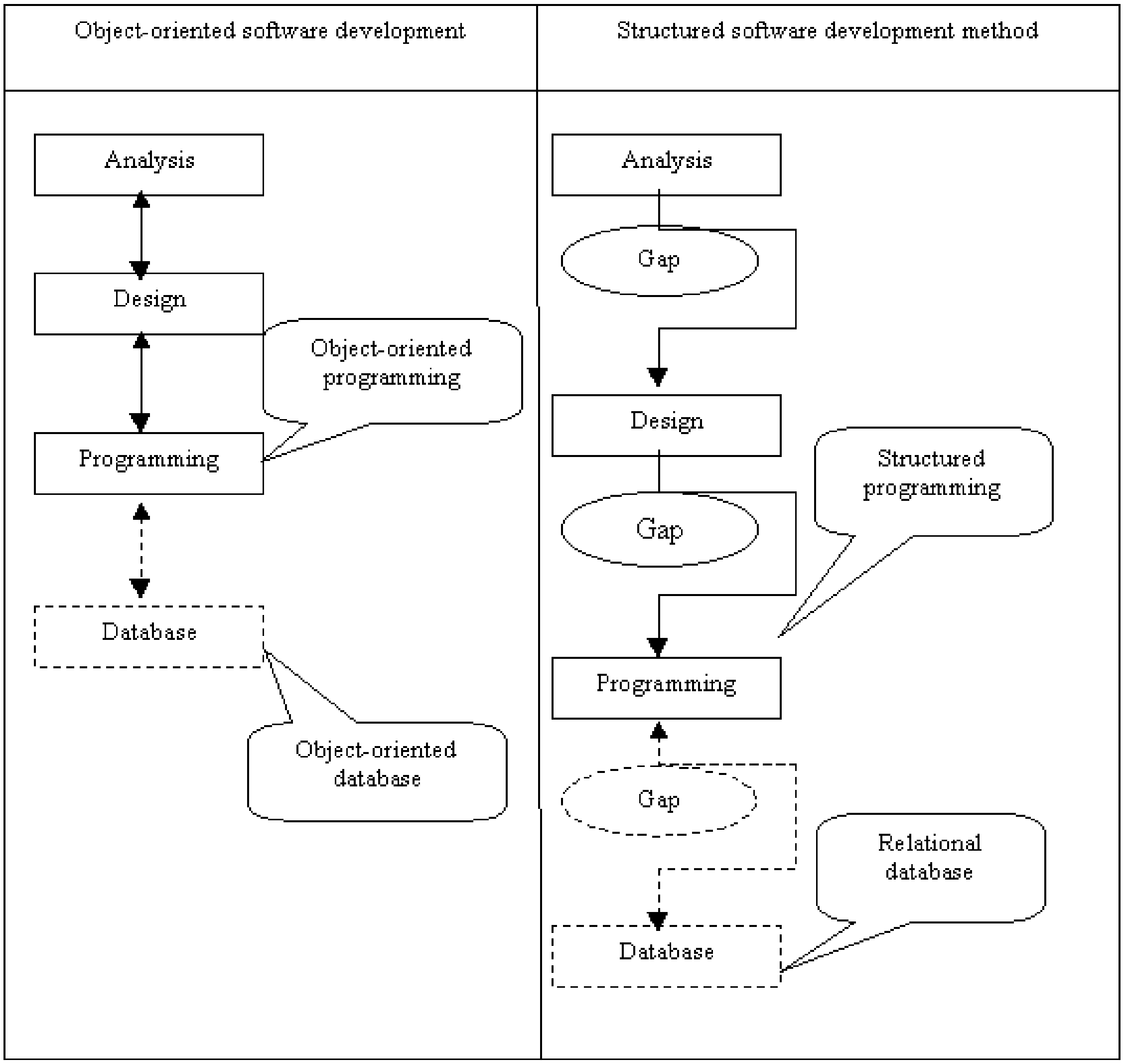
A decision tree is a tree-form tool representing a decision table in a tree structure. The following figure shows an example of a decision tree.



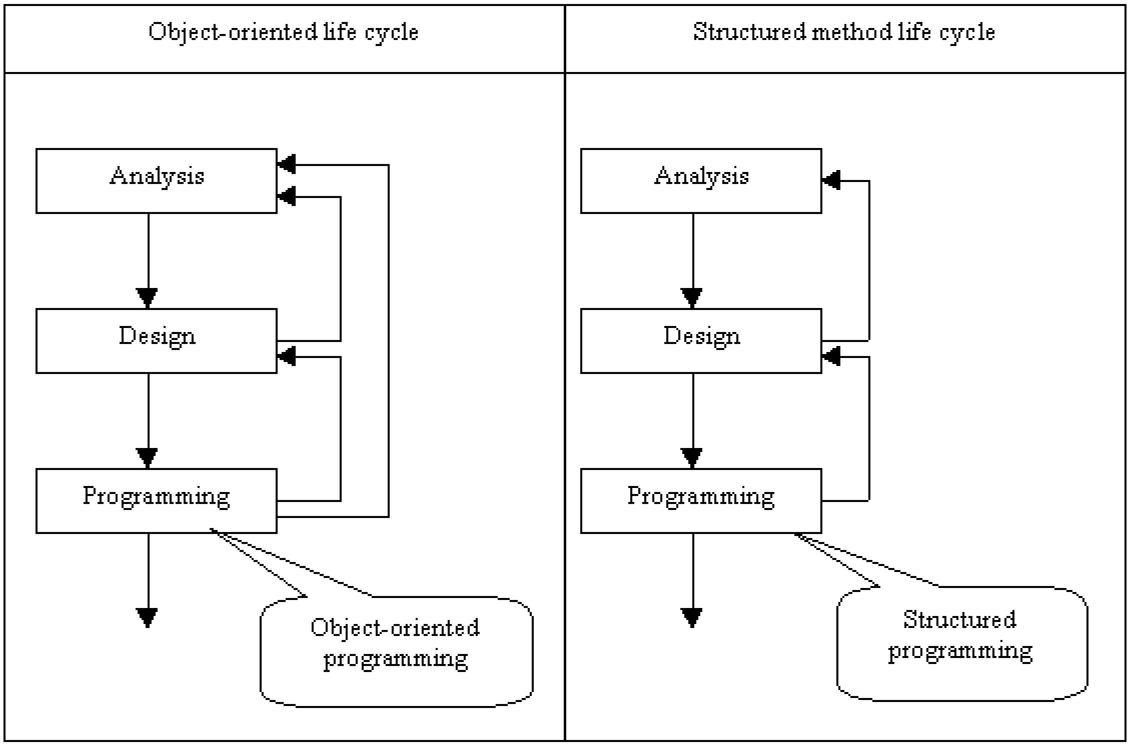
[Decision tree]

3.4.2 Object-oriented analysis method

The object-oriented analysis method corresponds to the system analysis process method in the object-oriented paradigm. In the object-oriented paradigm, everything is represented as an object. Examples of available analysis methods of the same kind are the Coad & Yourdon method, Shlaer & Meller method, Booch method, and OMT method. The object-oriented analysis method has undergone similar development as that of the structured analysis method. However, there is a crucial difference between the object-oriented analysis method and structured analysis method. The difference is that the object-oriented analysis method has almost no gaps between the analysis, design, and programming phases. Here, a gap refers to delimitation between processes. The following figure shows the difference in software development between the object-oriented analysis method and structured analysis method.



While structured analysis method is the waterfall model, the object-oriented analysis method takes a form similar to the spiral model, which means it is possible to go back from any process to another process. The figure on the following page shows the difference in life cycles between the object-oriented analysis method and structured analysis method.


Object-oriented life cycle
Structured method life cycle
Analysis
Design
Programming
Object-oriented programming
Analysis
Design
Programming
Structured programming

Exercises

1. Select the best description of the KJ method.
(1) Method for identifying system functions in terms of both input data and output data
(2) Method for clarifying the structure of group entities by analyzing external models and creating associated graphs
(3) Method for creating ideas based on different kinds of conceivable ideas for particular events
(4) Method for analyzing system responses of the system with the passage of time to the events.

2. Which of the following descriptions of the KJ method is incorrect?
(1) During analysis of requirements with the KJ method, the participation of not only developers but also customers is necessary.
(2) The KJ method is a software requirement analysis method suitable for determining requirements following a goal-oriented approach.
(3) In the KJ method, a moderator is necessary.
(4) The KJ method starts from understanding existing problems.

3. Select the best description of functional analysis.
(1) Software requirement analysis method for clarifying the structure among entities by analyzing a model of an external world.
(2) Software requirement analysis method for defining relationships among four elements: input data, output data, functions, and conditions
(3) Software requirement analysis method for analyzing system responses to events with the elapse of time
(4) Software requirement analysis method for creating ideas based on a variety of ideas

4. Use the best terms to fill in the [***square***] in the following description about event response analysis.

Event response analysis is a method of analyzing [***(1)***] responses [***(2)***]. An event is an example of this method is an event-driven GUI.

5. Use the best terms to fill in the [***square***] in the following description about structured analysis.

With the structured analysis method, the structures of groups of [***(1)***] can be clarified by analyzing a model of an external world. A typical example is [***(2)***]. This is a method for classifying components of a model by grouping them while focusing on their similarities.

6. From the group of choices listed below, select the software requirement analysis method indicated by each of the following descriptions.

(1) Method of clarifying the structure entities by analyzing a model of an externalworld

(2) Method of analyzing system responses over the passage of time

(3) Analysis method for identifying system functions in terms of both input data and output data

(4) Analysis method for creating ideas resulting from exchanges of opinions from the different standpoints of developers and customers meeting together

Answers:

a. KJ method

b. Functional analysis

c. Event response analysis

d. Structured analysis

7. Use the best terms to fill in the [***square***] in the following description about the functional hierarchical model.

The functional hierarchical model is a software requirement model created in a hierarchical configuration by stepwise refinement with a focus on [***(1)***]. In this model, functions are subdivided, such as from a system to [***(2)***] after system functions are divided. This model is the optimum model for the [***(3)***] life-cycle model.

8. Use the best terms to fill in the [***square***] in the following description about the dataflow model.

The data flow model is a software requirement model based on [***(1)***], and system functions are analyzed according to the relationships of input, [***(2)***], storage, and output. With the data flow model, models are developed hierarchically by using a diagram called the [***(3)***].

9. From the following sentences, select the one that does not describe a feature of the data flow model.

(1) The data flow model is suitable for analyzing requirements of business processing application systems that mainly handles data flows.

(2) Because customers cannot easily understand DFD, DFD may not facilitate good communication with customers.

(3) The data flow model is a software requirement model based on functional analysis.

(4) The data flow model is represented with four symbols: arrows, circles, double lines, and rectangles.

10. Which of the following phrases describes the control flow model?

(1) Software requirement model based on event response analysis, and relationships between data and control are considered to represent them properly

(2) Software requirement model focusing on data analysis

(3) Model for events in which multiple processors operate in parallel simultaneously

(4) Requirement model based on event response analysis, and this model is suitable for analyzing requirements of systems that require dynamic control

11. From the group of choices listed below, select the best terms for the [***square***] in the following description.

The Petri-net model is a software requirement model based on [***(1)***]. Because this model can represent synchronization of functions operating [***(2)***], it is suitable to focus on [***(3)***] in analyses of requirements for systems.

Answers:

a. functional analysis
b. event response analysis
c. in parallel
d. in series
e. business processing
f. control

12. From the group of choices listed below, select the best terms for the [***square***] in the following description about the data-oriented model.

The data-oriented model is a software requirement model that focuses on [***(1)***] analysis. The basic ideas of this model are as follows:

- Software requirement analysis for a system can be conducted more smoothly by focusing on data handled by the system.
- By defining clear relationships among [***(1)***], the [***(2)***] configuration is also determined.

Answers:

a. functional
b. element
c. job
d. data

13. Use the best terms to fill in the [***square***] in the following description about the object-oriented model.

The object-oriented model is a software requirement model developed from the [***(1)***] model, and the concept of [***(2)***] that indicates common features in a data structure is used for modeling.

14. Use the best terms to fill in the [***square***] in the following description about the parallel process model.

The parallel process model is a model for events in which multiple [***(1)***] operate [***(2)***] simultaneously.

15. From the group of choices listed below, select the best terms for the [***square***] in the following description about the structured analysis method.

The structured analysis method is a method for determining software requirements proposed by DeMarco. The purpose of structured analysis is to create structured specifications using tools such as [***(1)***], [***(2)***], and [***(3)***]. [***(1)***] is the diagram that clarifies system functions and the data flow. [***(2)***] is created after system functions and the data flow are clarified. [***(4)***] is used to create it. [***(3)***] define the system functions clarified in DFDs.

Answers:
a. minispec
b. data flow diagram (DFD)
c. a data dictionary
d. BNF

16. From the group of choices listed below, select the best term for each of the following descriptions about minispecs.
(1) Tool for creating minispecs in a natural languages based on specifications with three syntaxes
(2) Representation of (3) in a tree structure
(3) Tabular form tool used when complex conditions are combined
Answers:
a. decision table
b. data flow diagram
c. decision tree
d. structured language

17. From the group of choices listed below, select the best terms for the [***square***] in the following description about the object-oriented analysis method.

The object-oriented analysis method corresponds to the analysis process method in the object-oriented paradigm. The object-oriented analysis method has undergone similar development as that of the [***(1)***]. However, there are crucial differences between them. The differences are as follows:

- In the object-oriented analysis method, there are almost no [***(2)***] between the analysis, design, and programming phases.
- While the [***(1)***] is the waterfall model, the object-oriented analysis method takes a form similar to the [***(3)***].

Answers:
a. structured analysis method
b. structured programming
c. gaps
d. spiral model

18. For the following sentences on systematization, mark a circle next to it if it is correct. Otherwise, mark an X next to it.

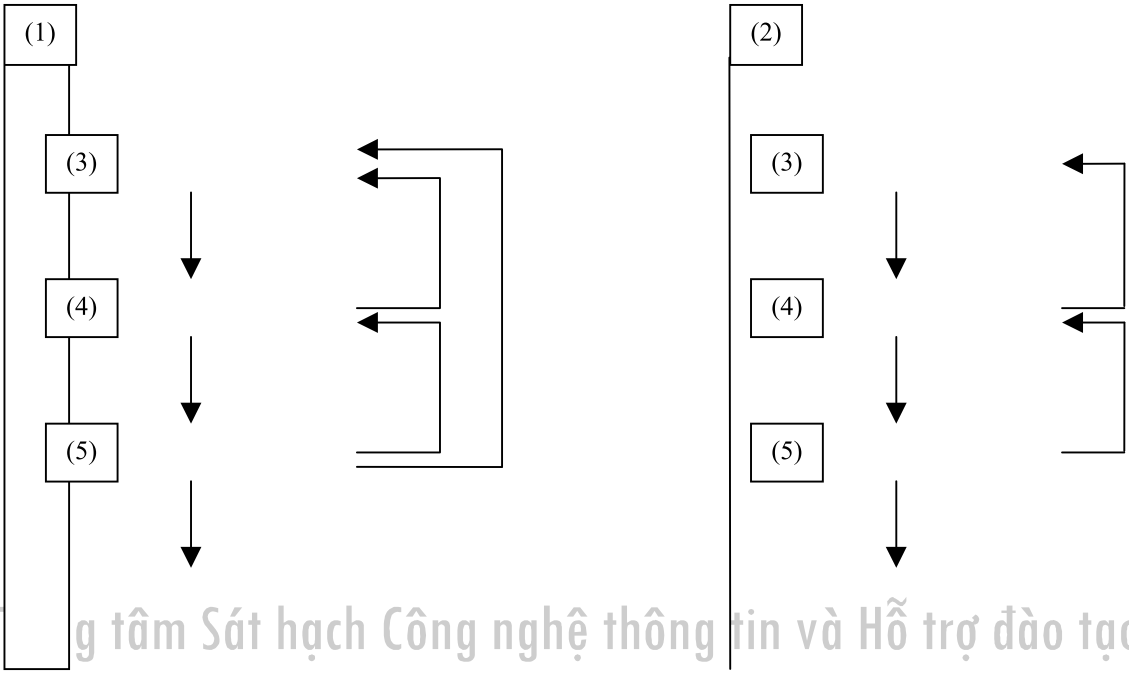
(1) Systematization is an idea from the viewpoint of the developer.
(2) One of the purposes of systematization is to improve system efficiency.
(3) By carrying out systematization, performance concerning both time and cost is improved.

19. Use the best terms to fill in the [***square***] in the following description about system effectiveness.

System effectiveness refers to [***(1)***] and [***(2)***] of a system. [***(1)***] indicates what the system can do and [***(2)***] indicates to the extent of the performance of a system. If the criteria of [***(1)***] are unreasonable, performance generally declines. If unreasonable demands are placed on [***(2)***], functions are generally restricted.

20. List three constraints on systematization.

21. Enter suitable names (words) in the life cycle flow shown below.



4 Software Design

Chapter Objectives

This chapter explains the basic concepts of software design. It also explains how to design software using structured design, the Jackson method, and the Warnier method.

4.1 Concepts of Software Design
4.2 Software Design Methods

4.1 Concepts of Software Design

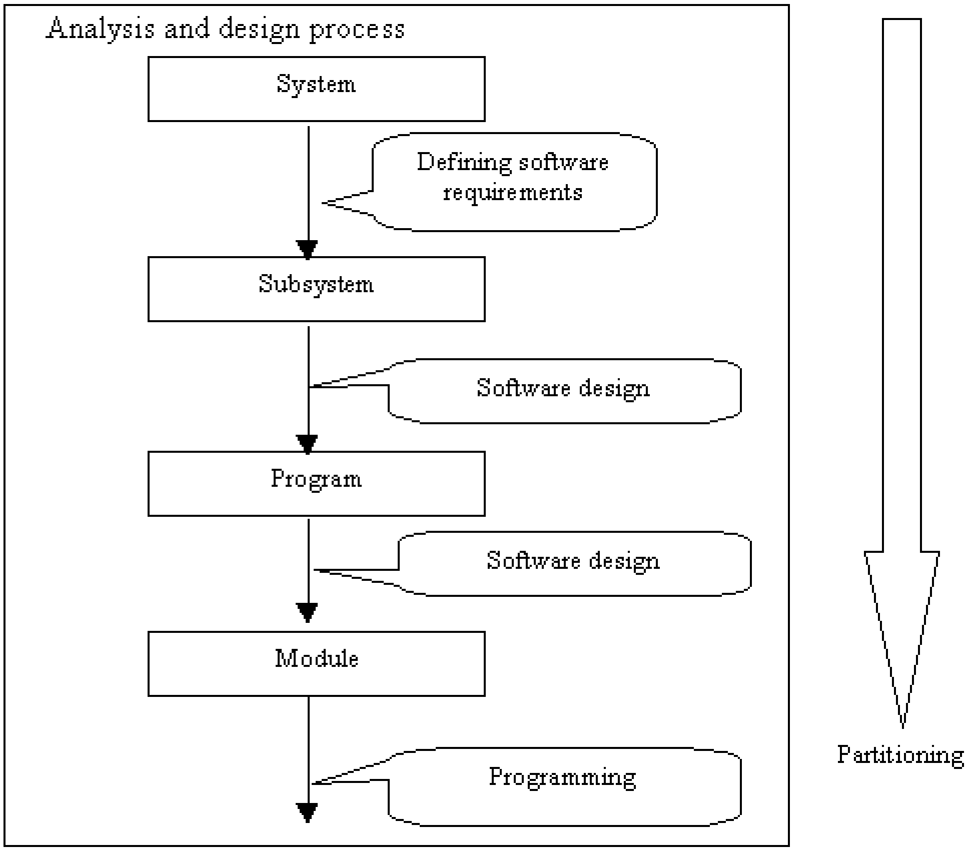
The functions required in software can be identified by partitioning a system during the design process, and the system can then be implemented by combining the identified functions. The concepts of division and integration are thus important in software design. This section explains software division and integration and the techniques used for software division.

4.1.1 Concepts of partitioning and integration

The concepts of partitioning and integration are utilized in software analysis, design, development, and implementation. According to these concepts, software can be designed by dividing a system into functional components, and software can be developed by integrating the components. Each concept is explained in the following.

(1) Partitioning

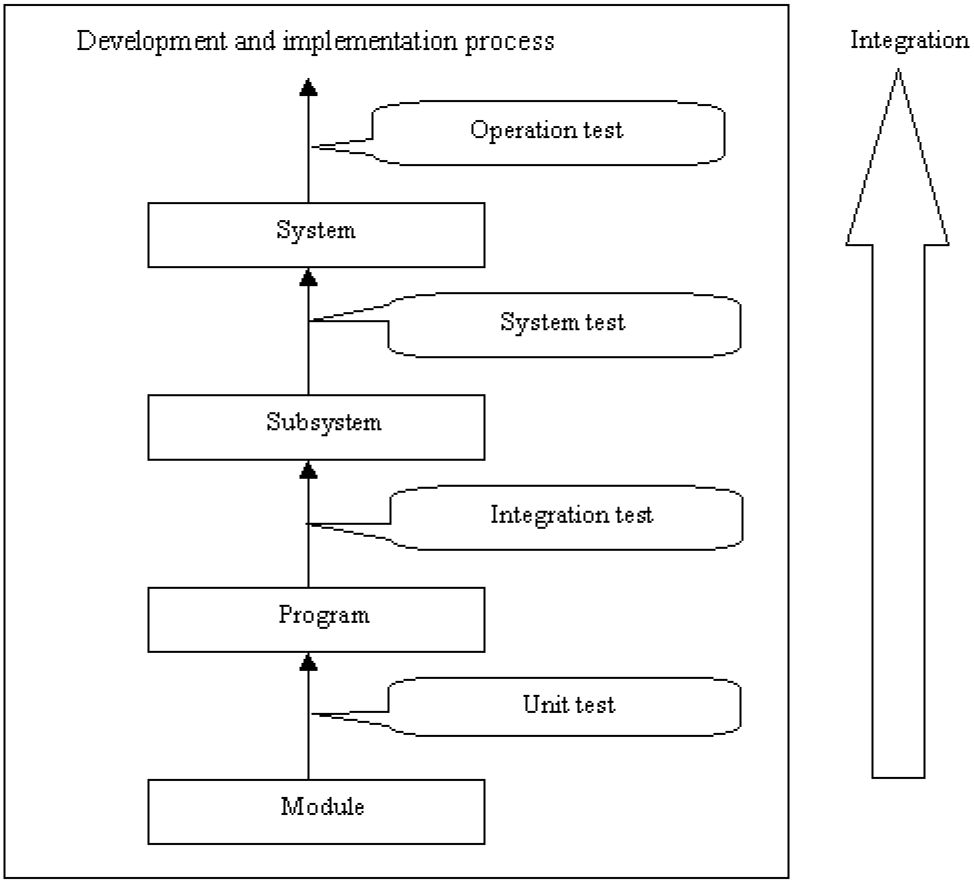
The concept of division is a principle used in the software analysis and design process. More precisely, the structure of a system can be divided step by step to organize the system into subdivisions. The following figure shows the analysis and design process:



As shown in the figure, the definition of software requirements is a phase in which a system is partitioned into subsystems, whereas software design is a phase beginning from subsystem to program development or from program to module development.

(2) Integration

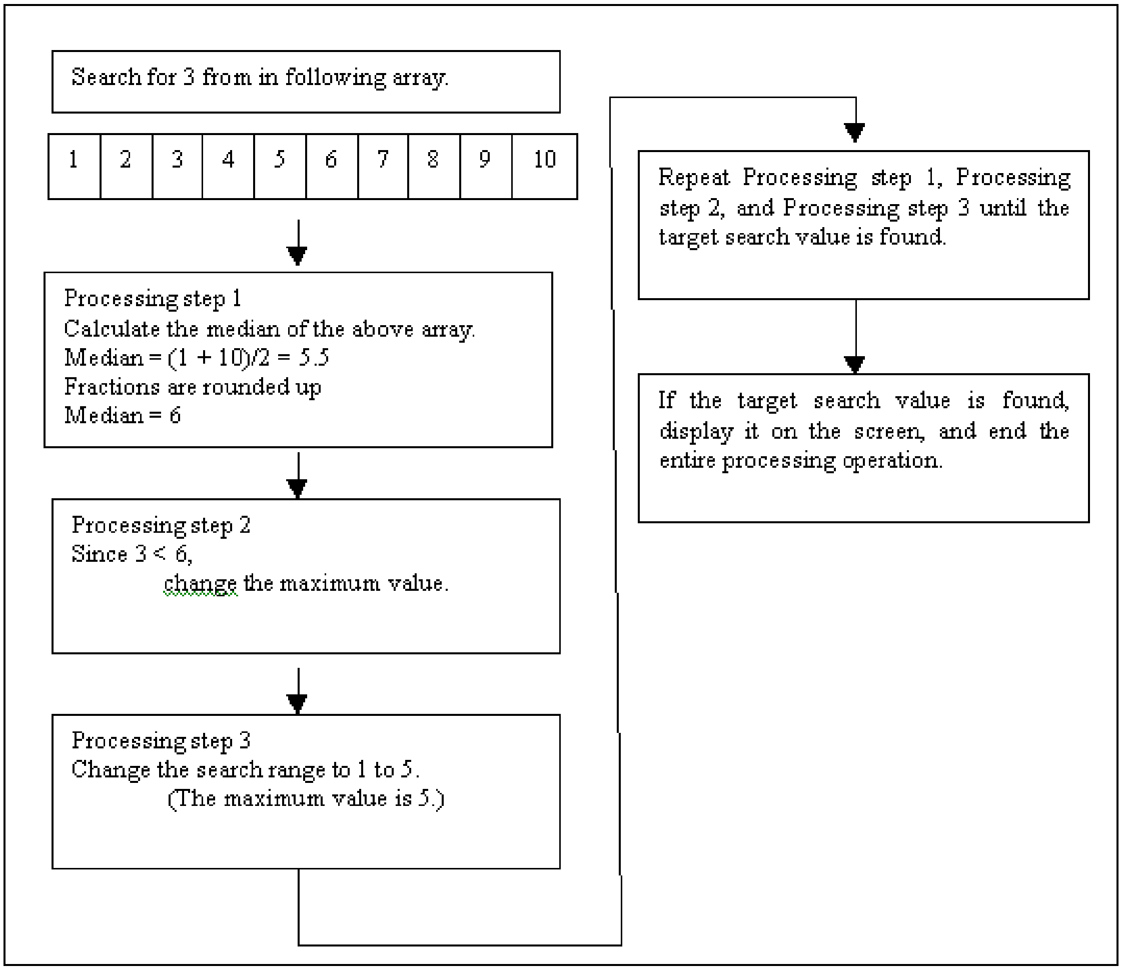
The concept of integration is a principle used in a series of development and implementation processes ranging from programming to verification. More precisely, a system can be created by integrating divided modules, programs, and subsystems. The following figure shows the development and implementation processes:



As shown in the figure, integration is a phase in which modules resulting from dividing the system are combined step by step so that they can be implemented as one system.

4.1.2 Concept of stepwise refinement

Stepwise refinement is based on the idea of minimizing complexity, and in this process, each problem is solved by dividing it into different layers and subdividing them step by step to make clear details of the problem. In the example of this concept illustrated in the following, a program searches for a specific value from 10 items of numeric data arranged and stored in ascending order in an array and then displays the found value on the screen. The following figure shows the flow of processing using a binary search:


Search for 3 from in following array.
1 2 3 4 5 6 7 8 9 10
Processing step 1
Calculate the median of the above array.
Median = (1 + 10)/2 = 5.5
Fractions are rounded up
Median = 6
Processing step 2
Since 3 < 6,
change the maximum value.
Processing step 3
Change the search range to 1 to 5.
(The maximum value is 5.)
Repeat Processing step 1, Processing step 2, and Processing step 3 until the target search value is found.
If the target search value is found, display it on the screen, and end the entire processing operation.

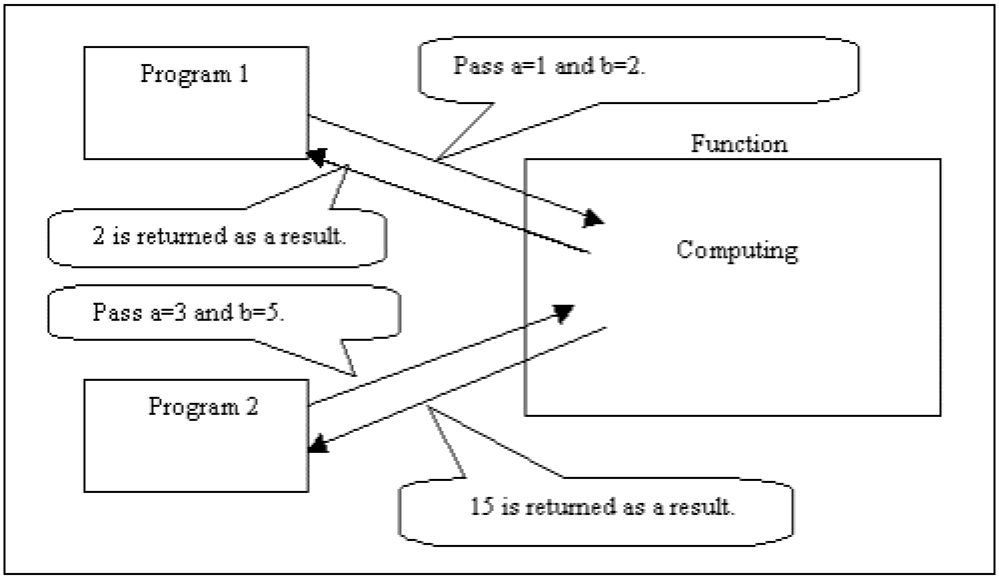
As mentioned in the diagram, subdividing a target section step-by-step is called stepwise refinement.

4.1.3 Concept of abstraction

Abstraction is an idea which enables the characteristics of a target to be revealed by extracting some of the most typical elements of the target. Some examples and explanations of abstraction in software development are given in the following.

(1) Procedure abstraction

Examples of procedure abstraction include the modules, subroutines, and functions of any procedural programming language. In each of them, multiple smaller units of processing are put together into one larger unit to hide details of the processing flow. In other words, to use such processing units from outside a system, one has to know only the structure of the data to be passed without having to know details of the processing flow in the system. The following figure shows a procedure abstraction:



The computation function shown in the figure performs its task based on data passed by each program and then returns a computation result to each program. Thus, the operation to make processing results available even if the calling source does not know the internal processing flow of the called destination is referred as procedure abstraction.

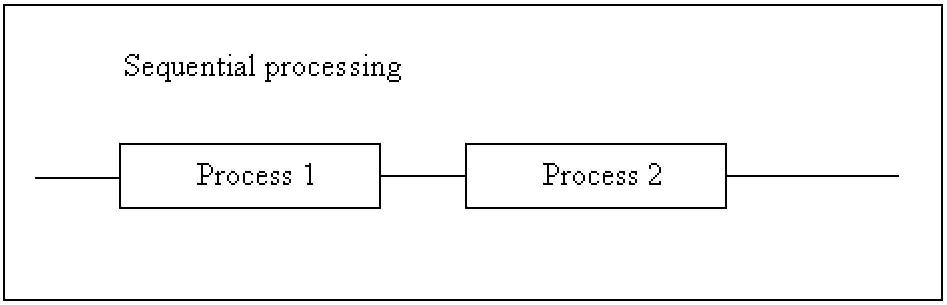
(2) Control abstraction
Control abstraction is imposing restrictions on control patterns by structuring algorithms described in a procedural programming language. The following table lists more concrete descriptions:

Program flow	Description
Control flow	Computation that changes with the passage of time, indicating the program has a dynamic structure
Text flow	Placement of statements that make up a source program, indicating the program has a static structure

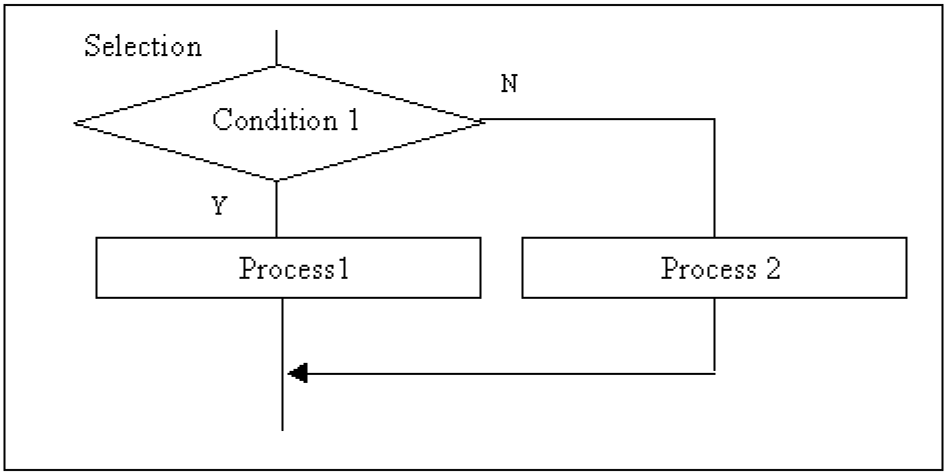
The flows of control and text described in the table are related to each other for the ease of understanding. That is, it is easy to understand an algorithm described in a program, if the flow of control matches that of text. Other principles for facilitating creation of easily understandable programs are to use a basic control structure and, to avoid using unnecessary GO TO statements. Structured programming is based on the idea that the flow of control can be matched with that of text by following these principles.

The following shows flowcharts illustrating the basic control structure of structured programming.

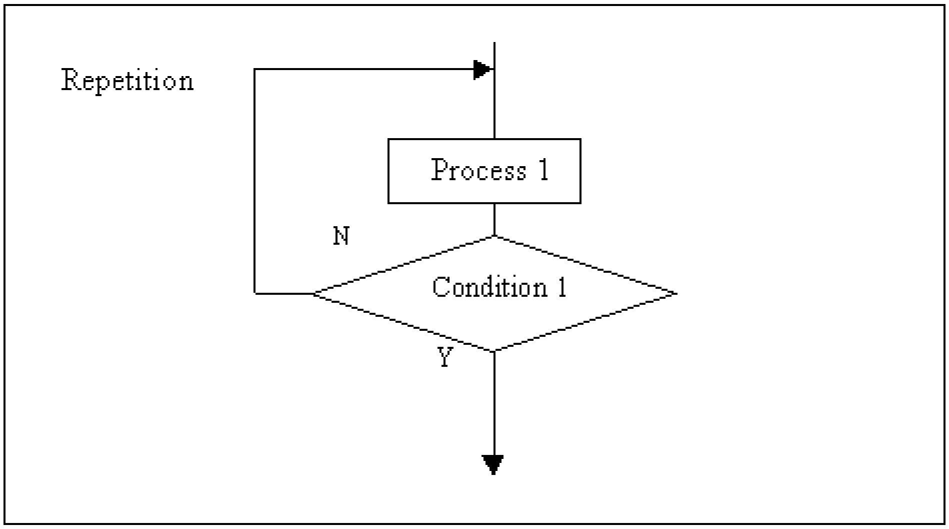
[Basic control structure of structured programming 1]



[Basic control structure of structured programming 2]



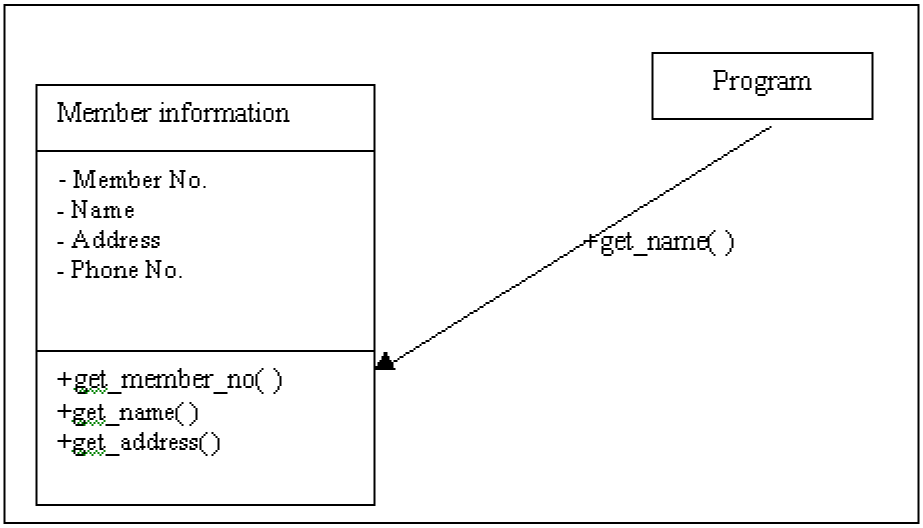
[Basic control structure of structured programming 3]



(3) Data abstraction
Data abstraction is an idea that the structure of a program can be defined according to data and operations with the data.

4.1.4 Concept of information hiding

Information hiding is a concept, proposed by David Parnas, that dependencies of mutually related components can be eliminated by hiding as much unnecessary information among the components as possible. This idea can be applied as described above to module division. The following figure illustrates this concept:



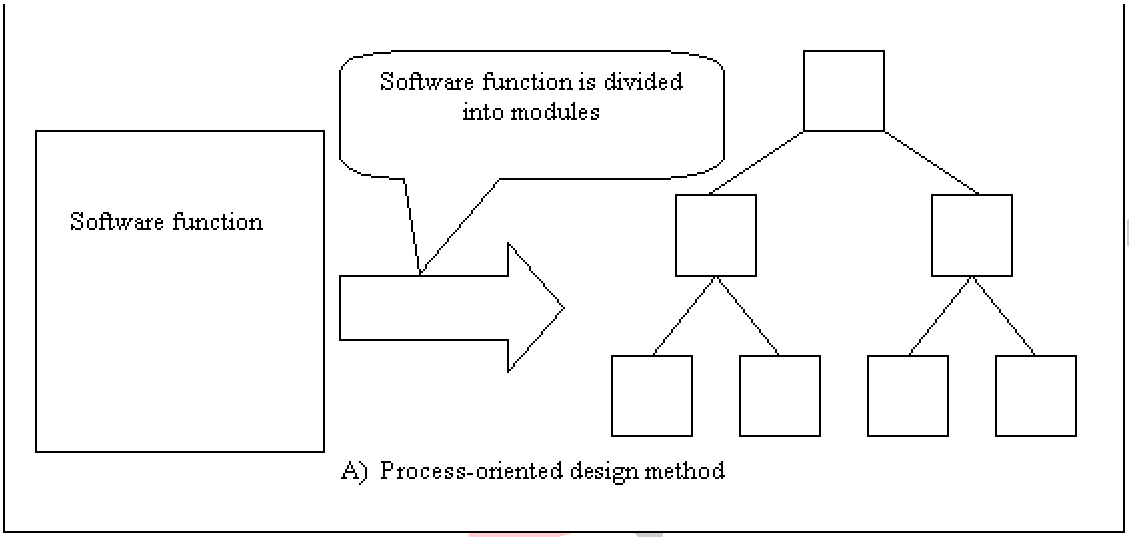
As shown in the figure, data can be hidden from programs so that it cannot be directly referenced. This is called information hiding. In this example, a program can refer to data by calling "+get_name()".

4.2 Software Design Methods

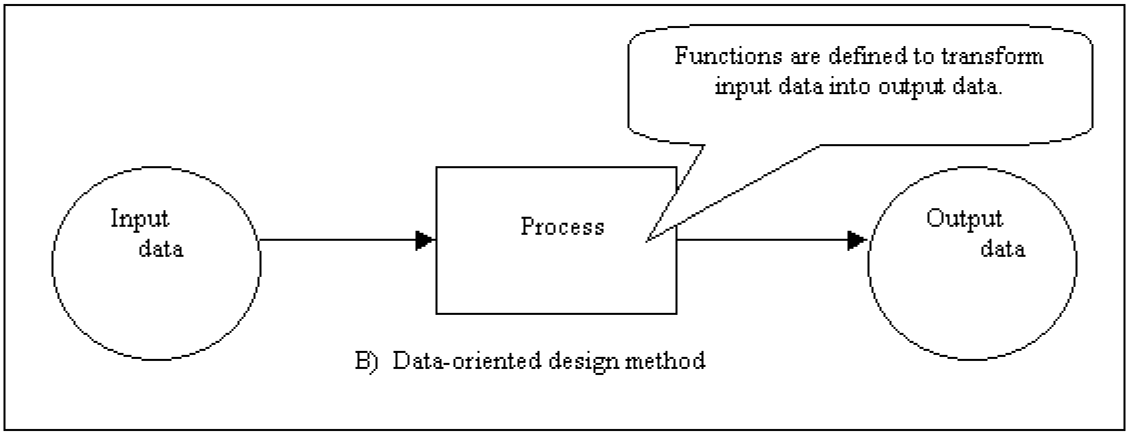
This section explains techniques that can be used, depending on software requirements, for the internal design of software. Structured design is taken as an example of a process-oriented software design method, and the Jackson method and Warnier method are taken as examples of data-oriented software design methods. Also an object-oriented design method is explained as an example.

4.2.1 Structured design method

Software design methods can roughly be categorized into two approaches: one approach focuses on the process itself, and another approach focuses on input data and output data. The following figure illustrates both approaches:



A) Process-oriented design method


Functions are defined to transform input data into output data.
Input data
Process
Output data

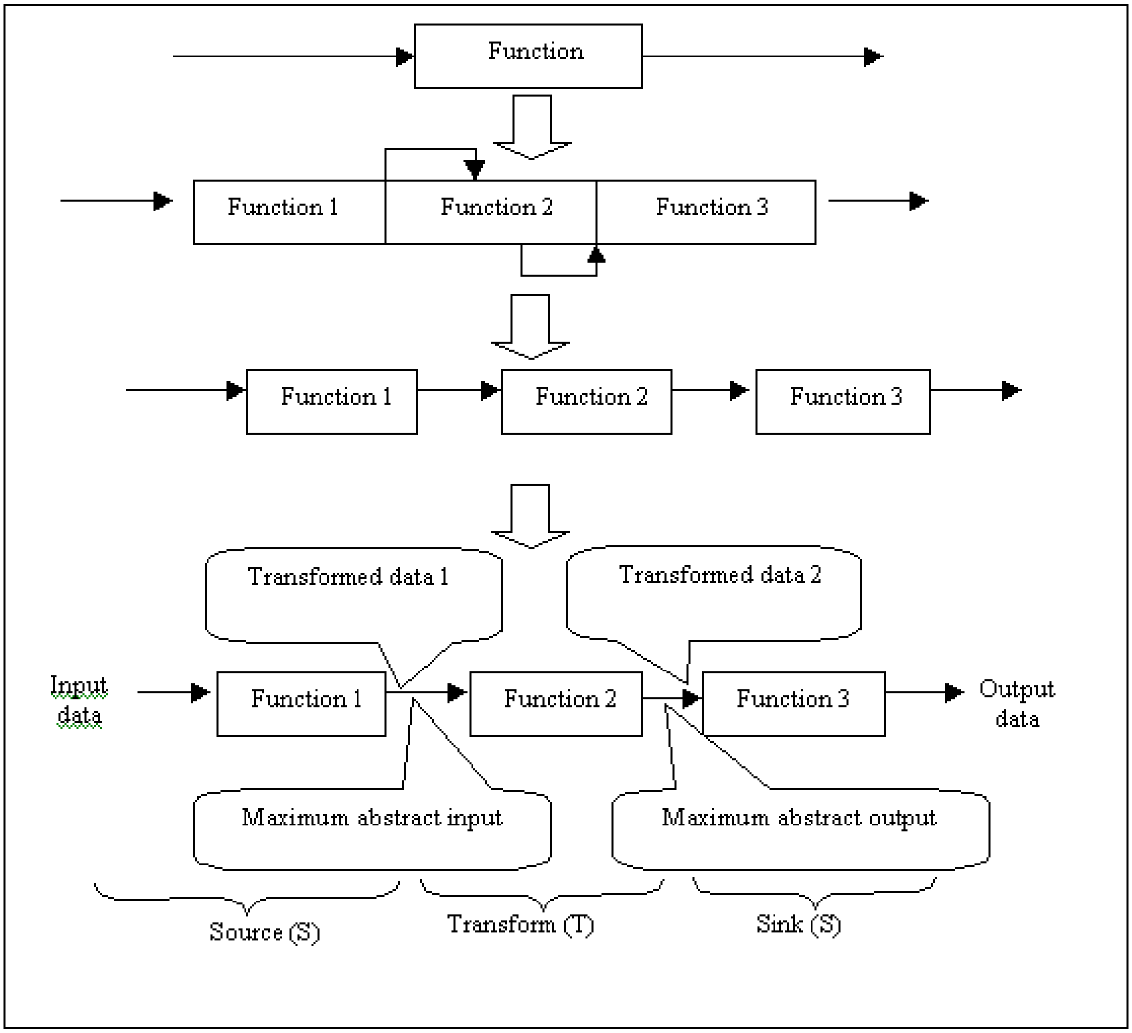
B) Data-oriented design method

The structured design method is a design method that focuses on processes. This technique was proposed by Larry Constantine, and it is used to design the structure of a module and its interface. The structured design method consists of the STS partitioning method and TR partitioning method. Each of these methods is explained in the following.

(1) STS partitioning method

The STS partitioning method applies to data flows in which no branching occurs. In this method, the target of division into each of the source (S), transform (T), and sink

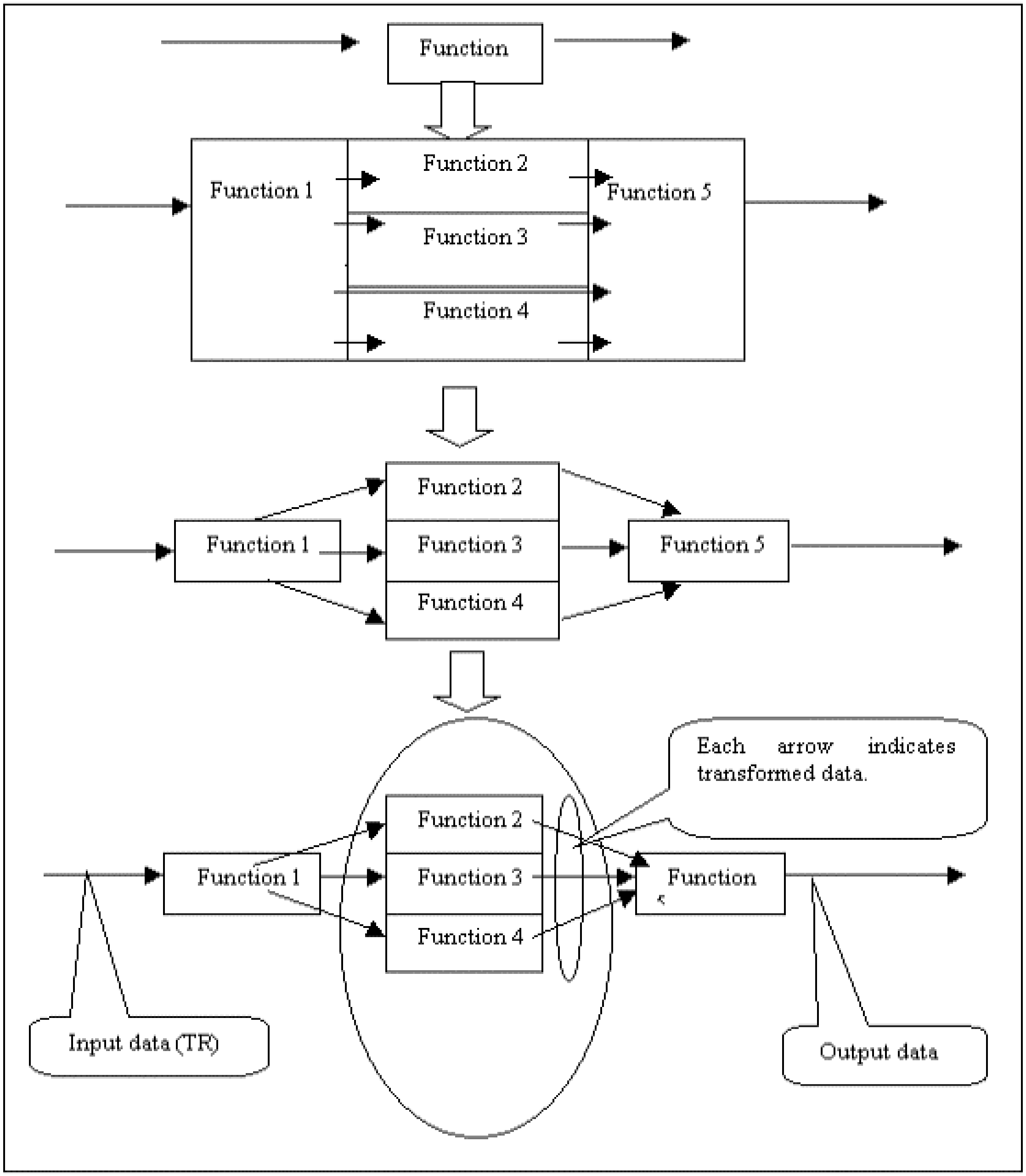
(S) is modularized. The following figure illustrates the process of the STS partitioning method:



[Software design method by the STS partitioning method]

(2) TR partitioning method

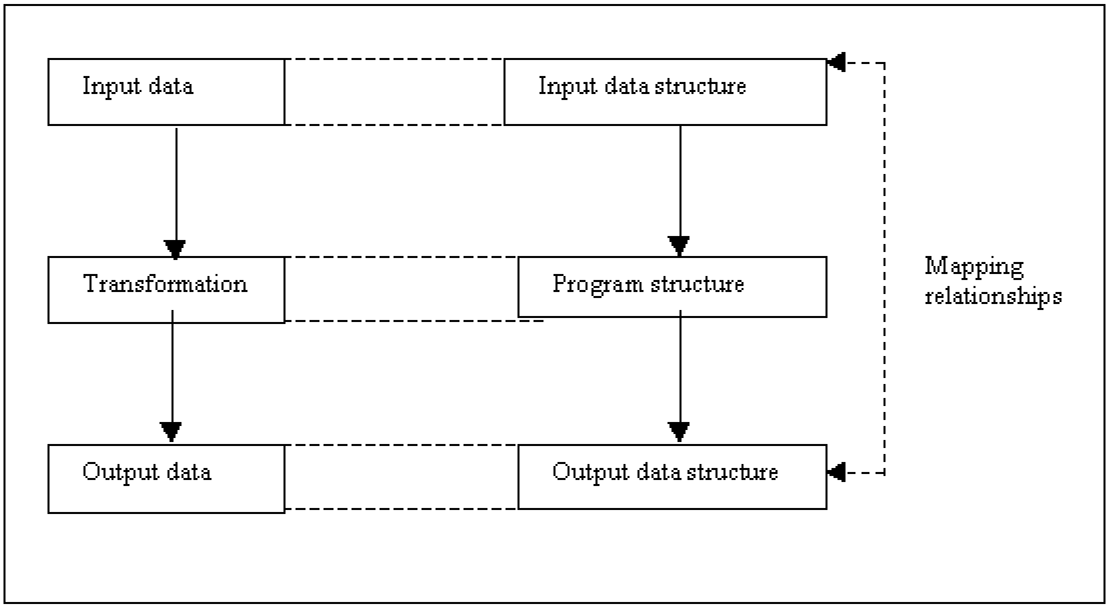
The TR partitioning method applies to flows of data in which branching occurs. In this method, the functions divided by transaction types are made modules. That is, such a flow is divided into a module for receiving transactions and assigning them to appropriate modules, transfer modules for individual transactions, and an output module. The following figure illustrates the process of the TR partitioning method.

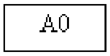
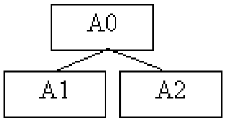
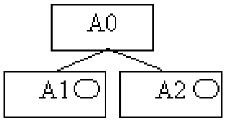
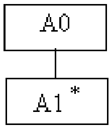

Function
Function 1
Function 2
Function 3
Function 4
Function 5
Function 1
Function 2
Function 3
Function 4
Function 5
Each arrow indicates transformed data.
Function 2
Function 1
Function 3
Function
Function 4
Input data (TR)
Output data

4.2.2 Jackson method

The Jackson method is a design method proposed by Michael Jackson, and it focuses on data used for designing software. This design method is appropriate for business computations because the processing structure is derived by clarifying the relationships between input data and output data. This method uses JSP tree structure diagrams and graphic logic. Since each JSP tree structure diagram uses the basic control structure, it is suitable for representing hierarchical structures. Graphic logic is used to define detailed program specifications. The following figure and tables illustrate the idea of the Jackson method, explain the structure of a JSP tree structure diagram, and outline a notation for representing diagrammatic logic. They are followed by an explanation of the design procedure.

[Conceptual diagram of the Jackson method]



Name	Tree structure diagram	Data structure	Program structure
Element		Shows a data item	Shows a procedure statement
Sequence		Shows records (Record A0 consists of A1 and A2)	Shows procedure blocks (Procedure A0 consists of A1 and A2)
Selection	 ([○]) is a selection condition	Shows records to be selected (Record A0 consists of A1 and A2)	Shows procedures to be selected (Procedure A0 consists of A1 or A2)
Iteration	 (* indicates repetition)	Shows repetitive records (Record A0 consists of multiple A1s.)	Shows a procedure with repetitive statements. (Procedure A0 consists of multiple A1s.)

[Basic constructs used in JSP tree structure diagram]

Name	Notation	Description
Sequence	Seq ... end	Execute procedures enclosed by Seq and end sequentially.
Selection	Sel (condition 1) … alt (condition 2) … end	Execute the corresponding procedure that matches the condition.
Iteration 1	Itr while (condition) … end	Repeat the procedure while the condition is satisfied.
Iteration 2	Itr until (end condition) … end	Repeat the procedure until the end condition is satisfied.

[Notation for diagrammatic logic]

(1) Design the data structures.
Use a JSP tree structure diagram to represent the structures of input data and output data. This makes it easier to understand the record structure, if input data and output data are considered to be files.

(2) Design the program structure.
Determine the relationships between the input data and output data represented in the JSP tree structure diagram. Also use a JSP tree structure diagram representing input data as a template to create a tree structure diagram of the program.

(3) Design the procedures.
In the tree structure diagram of the program, list the procedures to be executed for each component.

(4) Define the specifications of the program.
Refer to the tree structure diagram of the program and use a notation for representing diagrammatic logic to create detailed specifications of the program. At this point clarify conditions, such as those for selection and iteration conditions that are not considered in the tree structure diagram of the program.

4.2.3 Warnier method

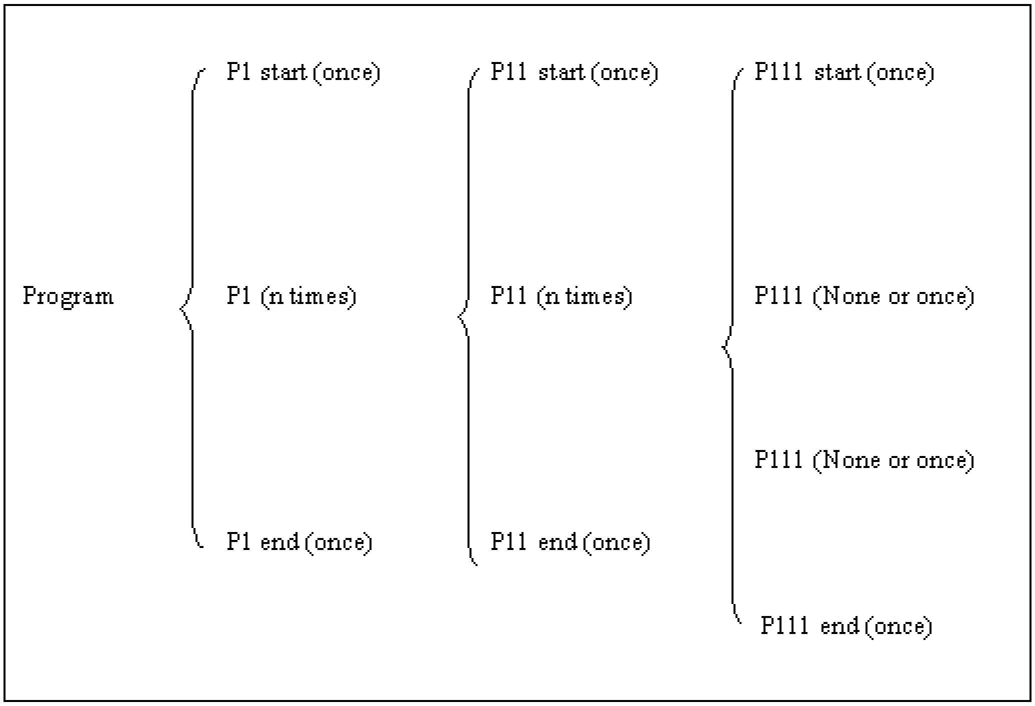
The Warnier method is a technique proposed by Jean-Dominique Warnier. Like the Jackson method, the Warnier method is a technique that derives program structures from data structures, but it differs from the Jackson method in that it derives a program structure by focusing on the structure of input data. The procedure for design by the Warnier method is given in the following.

(1) Design the structure of output data.
Clarify the logical structure of output data. Divide step by step the data construct in such a way that the sub-construct whose count of appearance is the largest is divided into smaller units any of which has the smallest appearance count.

(2) Design the structure of input data.
Clarify the logical structure of input data. The approach is the same as that in step (1).

(3) Design the program structure.
Create a program structure diagram based on the input data structure diagram. Create a flowchart based on the derived program structure diagram.

(4) Create a programming table.
A programming table is a set of grouped procedures. In this procedure, list all possible procedures based on the flowchart, and then arrange them in a programming table in units of modules.
Create detailed program specifications by following the procedure. The figure in the following shows a program structure designed using the Warnier method.



4.2.4 Object-oriented design method

The purpose of object-oriented design is to create concrete designs following the software models created in object-oriented analysis and to implement them on physical computers. Since it is possible in this method to go back to the analysis process from the design process, efficient development can be carried out by standardizing object-oriented analysis and design. The procedure of the object-oriented design method, taking the Coad & Yourdon method as an example, is given in the following.

(1) Design the system architecture.
Choose hardware and software to be used in the target system. The following points have to be reviewed regarding the software environment:

1) Object-oriented programming language
Can an object-oriented programming language be used? Is an environment supporting development provided?

2) OS and network architecture
Are the optimum OS and network architecture supported as an object-oriented environment?

3) Repository
Is there an environment that can be implemented with an object-oriented database?

4) Development environment
Is a development environment that supports programming provided?

(2) Review the classes and objects
Check for any missing objects, considering system implementation. Review the structures of classes and objects, and optimize them.

(3) Review the attributes and services

Review the attributes and services that characterize objects.

(4) Design the services.
The design of services is the most important aspect of object orientation. A service is the behavior of an object, and this behavior determines the functions implemented by the system. Contract models are used in service designs. In this procedure, the contract concluded between the server and a client is modeled.

(5) Design the dynamic structure.
The purpose of dynamic structure design is to design dynamic parts of the service protocol. The service protocol refers to the combination of messages sent and received between objects.
A dynamic structure refers to the state transitions caused by the object behavior of the corresponding messages sent and received between objects, timing control among messages, transmission sequence, and other factors.

Exercises

1. Use the best terms to fill in [***square***] in the following description about the concepts of division and integration in software design.

   The concepts of division and integration are utilized in software analysis, [***(1)***], development, and implementation. This means that software can be [***(2)***] by dividing a system into functional components, and software can be [***(3)***] by [***(4)***] the components.

2. From the group of choices listed below, select the best terms for the [***square***] in the following explanation about the concept of stepwise refinement.

   Stepwise refinement is based on the idea of [***(1)***] and in this process, each problem is solved by dividing it into [***(2)***] and subdividing them step by step to make clear [***(3)***].

Answers:
a. early solutions
b. different layers
c. details of the problem
d. minimizing complexity

3. From the following sentences, select the one that correctly explains the abstraction concept of procedure abstraction.

(1) In procedure abstraction, the internal processing of functions have to be understood before the function can be used.
(2) In procedure abstraction, a function can be used only by calling it, regardless of its external specifications.
(3) In procedure abstraction, details of internal processing of a function need not be known to use the function, if its external specifications are known.

4. List the three basic control constructs in structured programming.

5. Use the best terms to fill in the [***square***] in the following explanation about the concept of information hiding.

   Information hiding is a concept proposed by [***(1)***], and it indicates that dependencies of mutually related [***(2)***] can be [***(3)***] by hiding as much unnecessary information among the components as possible.

6. Use the best terms to fill in the [***square***] in the following explanation about the structured design method.

   The structured design method, which was proposed by [***(1)***], focuses on [***(2)***]. This technique is used to design the [***(3)***] and its [***(4)***].

7. From the group of choices listed below, select the best terms for the [***square***] in the following explanation about the TR partitioning method in the structured design method.

   The TR partitioning method is a method applies to flows of data in which branching [***(1)***]. In this method, the target of division into each [***(2)***] is [***(3)***]. That is, such a flow is divided into a module for receiving [***(4)***] and assigning them to appropriate modules, transfer modules for individual [***(4)***], and an output module.

Answers:
a. transactions

b. modularized
c. processing
d. does not occur
e. occurs
f. data item
g. group of transactions

8. From the group of choices listed below, select the correct sequence of steps for software design with the Jackson method.
(1) Procedure design
(2) Data structure design
(3) Program structure design
(4) Determination of program specifications
Answers:
a. (2) -> (1) -> (3) -> (4)
b. (1) -> (3) -> (2) -> (4)
c. (2) -> (3) -> (1) -> (4)
d. (2) -> (3) -> (4) -> (1)

9. From the following sentences, select the one that does not describe the Jackson Method.
(1) The Jackson method focuses on data when designing.
(2) The Jackson method uses JSP tree structure diagrams and graphic logic.
(3) Because the Jackson Method focuses on processes, it is not suitable for business computation.
(4) The Jackson method is a design method that focuses on the relationships between input data and output data.

10. Give a difference between the Warnier method and Jackson method.

11. From the group of choices listed below, select the correct sequence of steps for design with the Warnier method.
(1) Design the output data structure.
(2) Create a programming table.
(3) Design the input data structure.
(4) Design the program structure.
Answers:
a. (1) -> (2) -> (3) -> (4)
b. (1) -> (3) -> (4) -> (2)
c. (2) -> (1) -> (3) -> (4)
d. (3) -> (1) -> (4) -> (2)

12. From the group of choices listed below, select the best terms for the [***square***] in the following explanation about the object-oriented design method.

The purpose of object-oriented design is to create [***(1)***] following the [***(2)***] created in object-oriented analysis and to implement them on physical computers. Since it is possible in this method to go back to the [***(3)***] from the [***(4)***], efficient development can be carried out by standardizing [***(5)***] and design.

Answers:
a. software models
b. concrete designs
c. production process
d. analysis process
e. design process
f. object-oriented analysis
g. details
h. object-oriented programming

13. From the group of choices listed below, select the best terms for the [***square***] in the following description about program specifications for use with the Jackson Method.

In the Jackson Method, [***(1)***] is used and the tree structure diagram of a program is a reference for creating [***(2)***] of the program. Conditions not considered in the tree structure diagram of the program are clarified at this point. Such conditions include those for [***(3)***] and [***(4)***].

Answers:
a. selection
b. detailed specifications
c. diagrammatic logic
d. repetition

14. Enter the type of chart in the [***square***] in the following description about program structure design with the Warnier method.

Create a program structure diagram based on the input data structure diagram. Create [***(1)***] based on the derived program structure diagram.

15. List the two design methods that make up the structured design method.

16. Use the best terms to fill in the [***square***] in the following description about program specifications for use with the Warnier method.

A programming table is a set of [***(1)***] [***(2)***]. In this procedure, list all possible procedures based on the corresponding [***(3)***], and then arrange them in a programming table in units of modules.

5 Programming

Chapter Objectives

This chapter explains which programming paradigm is most suitable for various types of programming languages, rather than explaining the programming languages themselves.

5.1 Procedural Programming Paradigm
5.2 Logic Programming Paradigm
5.3 Functional Programming Paradigm
5.4 Object-oriented Programming Paradigm

5.1 Procedural Programming Paradigm

Programming paradigms are programming models that consist of a variety of programming principles organized on the basis of past experience and logical concepts.

There are the following programming paradigms:
- Procedural programming
- Logical programming
- Functional programming
- Object-oriented programming

What programming paradigm is suitable heavily depends on the programming language. The best programming paradigm varies with the programming language in use.

The explanation of this chapter begins with the procedural programming paradigm.

Procedural programming refers to a programming method that uses a sequence of procedures to be executed.

Currently used procedural programming languages include COBOL, BASIC, PL/I, C, PASCAL, FORTRAN, ADA, and, ALGOL.

Structured programming is a typical example of procedural programming techniques. Structured charts are used to visually represent the program structure in structured programming.

The concept behind the procedural programming paradigm is to build a program structure that is the most suitable for a procedure-oriented program.

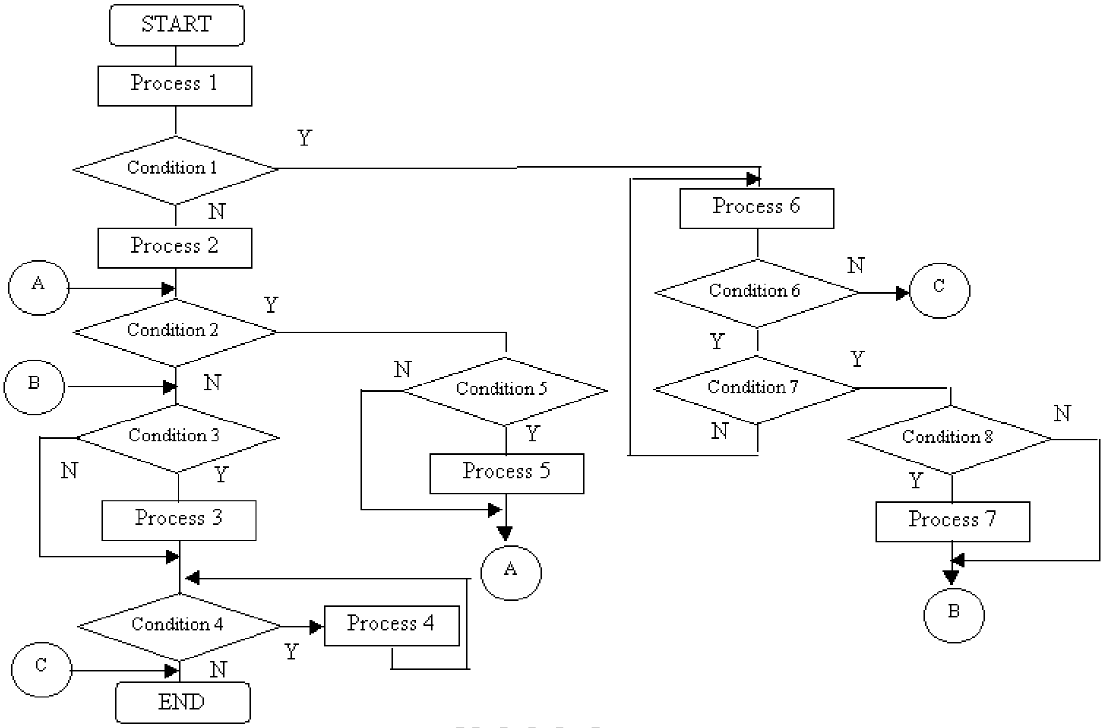
Structured programming is explained in the following sections.

5.1.1 Structured programming

Three types of control constructs are used in structured programming without using unnecessary GO TO statements: sequences, selections, and repetition.

(1) GO TO statement

A GO TO statement transfers control from a lower part of the control flow to a specified upper part, when a predefined condition is encountered, ignoring the basic flow of processing (from top to bottom).

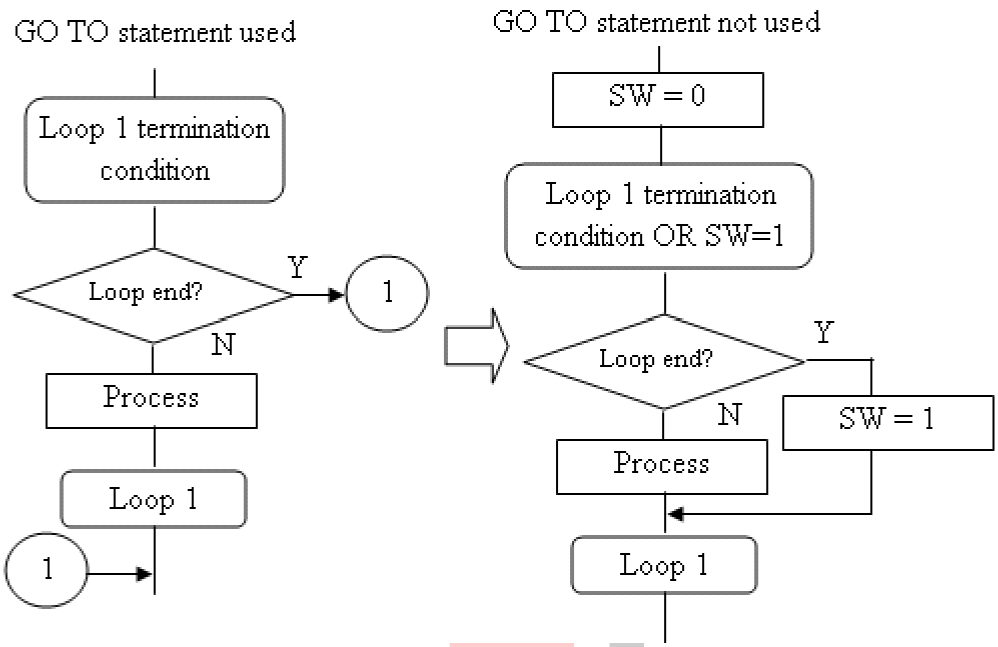


This example shows how GO TO statements make the processing control structure complicated.

As shown, use of GO TO statements makes the processing flow inconsistent. Avoid using GO TO statements whenever possible.

(2) Structure theorem

The structure theorem is that a proper program can be represented using basic control constructs. Programs written based on the structuring theorem are free of instructions that cannot be executed under any condition as well as infinite iterations. The structure theorem is intended to make programming easy to understand.



The structure theorem relies on newly defined variables (SW in the example), instead of using GO TO statements. This makes the processing structure easy to understand.

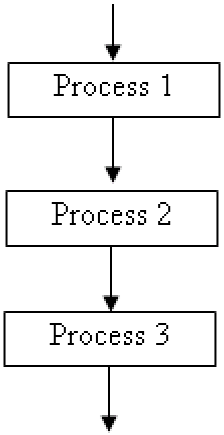
(3) Basic control constructs

To structure programs without using GO TO statements, basic control constructs are used.

These basic control constructs include sequence, selection, and iteration. They control the processing flow.

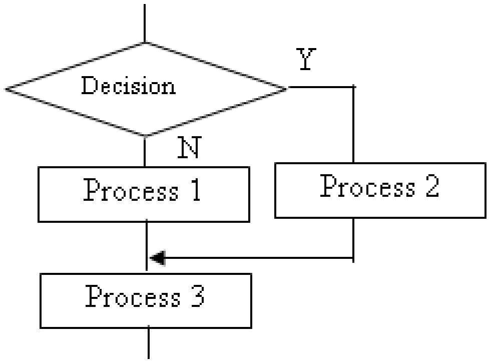
Selection control constructs can be divided into two types: One-way branches and multiple-way branches. Iteration control constructs are also divided into two types: loops with a termination condition at the top and loops with a termination condition at the bottom.

1) Sequence



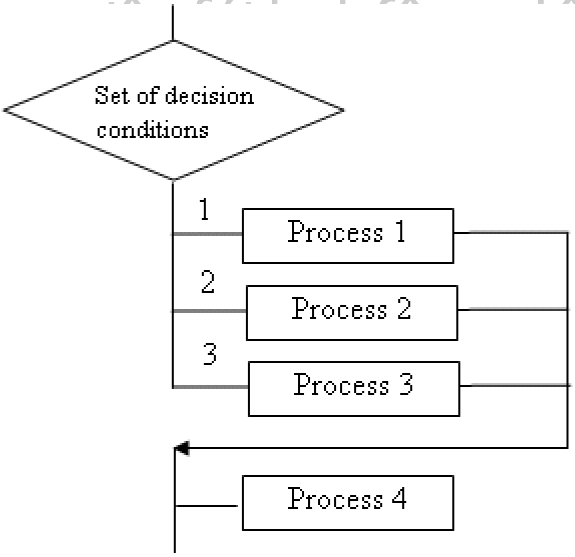
Sequences do not include selections or iterations, and processes are executed in sequence from the top down.

2) Selection based on a single condition



- If the decision is Y, process 2 and process 3 are executed in that order.
- If the decision is N, process 1 and process 3 are executed in that order.

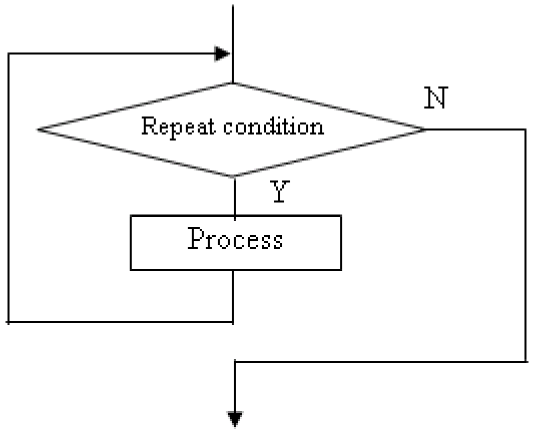
3) Selection based on multiple conditions (multiple-way branch)



If an item can assume two or more values, it will be processed in two or more stages. (Values may be expressions that can be evaluated to determine whether the condition is fulfilled.)

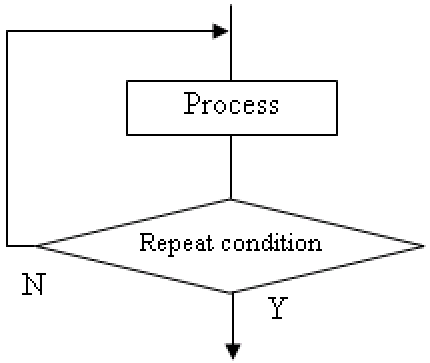
- If the value of the item is 1, process 1 and process 4 are executed in that order.
- If the value of the item is 2, process 2 and process 4 are executed in that order.
- If the value of the item is 3, process 3 and process 4 are executed in that order.

4) Loop with a termination condition at the top (DO WHILE loop)



- The process may be executed after the repeat condition is evaluated. The process is repeated as long as the condition is fulfilled (Y). Repetition ends when the condition is no longer fulfilled (N).
- Because the termination condition is at the top, it is possible that the process will never be executed.

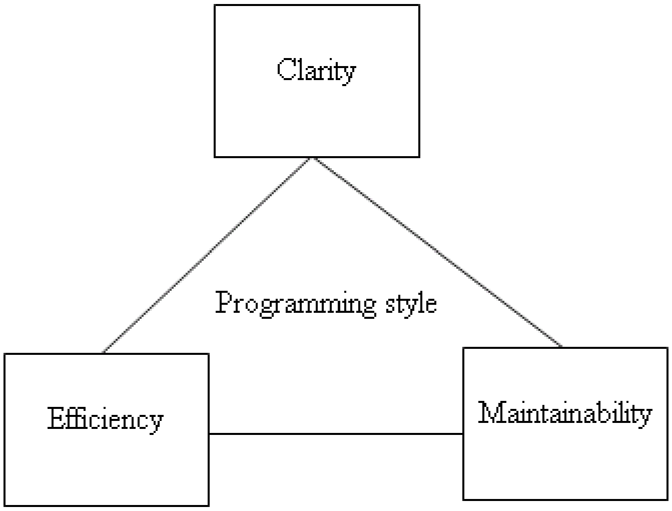
5) Loop with a termination condition at the bottom (DO UNTIL loop)



- The process may be executed before the repeat condition is evaluated. The process is repeated as long as the condition is not fulfilled (N). Repetition ends when the condition is fulfilled (Y).
- Because the termination condition is at the bottom, the process will always be executed at least once.

(4) Programming style

For procedural programming, the programmer must keep a programming style that maintains three key properties: clarity, efficiency, and maintainability.



The following explains each of these properties.

1) Clarity

- Clarity refers to a programming style that results in readable and easy-to-understand programs.

To achieve clarity, spaces and parentheses are used to show the program structure.

The start and end of control are represented by braces, "{" and "}". The contents of the process are clearly represented using structured coding techniques such as indentation and comments.

- Indentation means starting a line at a position different from that for other instructions. It is a technique that is intended to clarify the range of control and achieve visual legibility.

Example 1

```
If (conditional expression)
{
 Value_1 = value_2;
 Value_3 = value_4;
}
```

Example 2

```
for (initial_value, condition, increment) {
   value_1 = value_2;
   If (conditional expression) {
     Value_3 = value_4;
     Value_5 = value_6;
   }
   value_7 = value_8;
}
```

2) Efficiency

- Efficiency refers to a programming style that focuses on program execution time and the storage area to be used by the program. Efficiency is directly associated with the amount of processing.

- There are two types of computational complexity: area-related computational complexity, which represents the size of program storage areas and time-related computational complexity, which represents the program execution time required.

3) Maintainability

Maintainability greatly affects the length of the software life cycle. The higher the maintainability (ease of maintenance), the longer can the program be used. Ensuring that programs are well structured is a prerequisite for increasing maintainability.

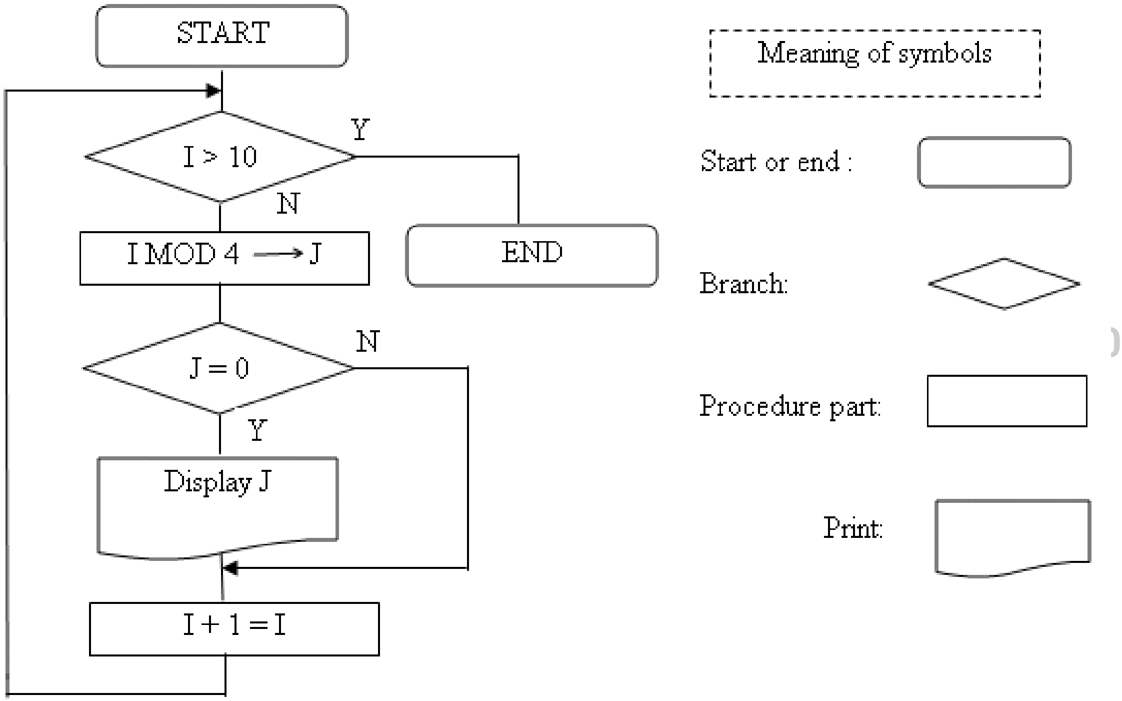
5.1.2 Structured charts

Structured charts are flow diagrams. Different notations are in use for the various structured charts.

Consider a process that displays on the screen multiples of 4 that are equal to or smaller than 10. The rest of this section shows how this process is represented in each of eight typical structured charts.

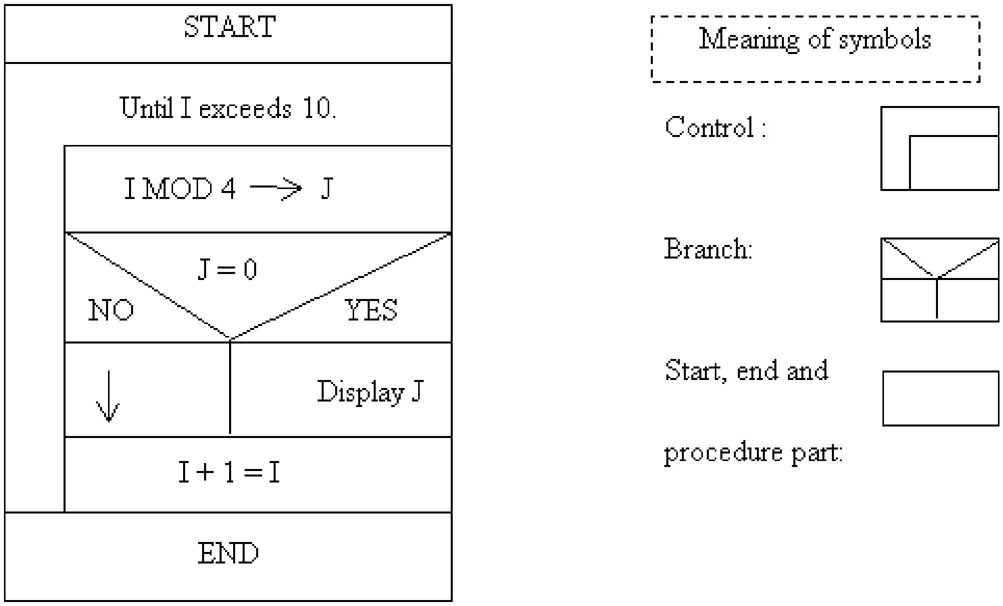
(1)Flowchart

A flowchart is a programming diagram that can represent basic control constructs.



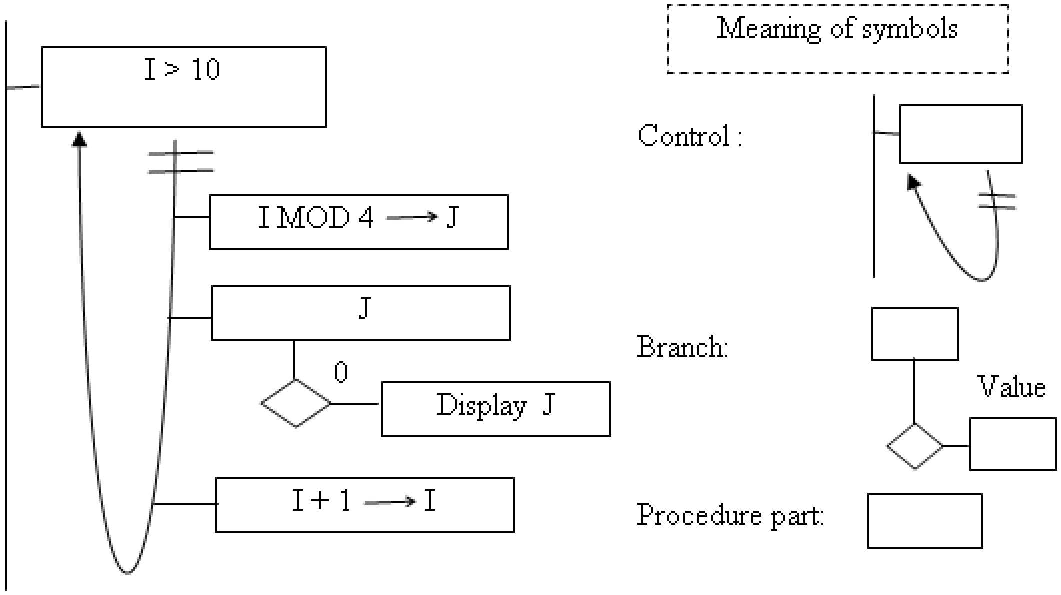
(2) NS chart (Nassi-Schneiderman Chart)

The NS chart is a structured chart that was standardized in the Netherlands and Germany.



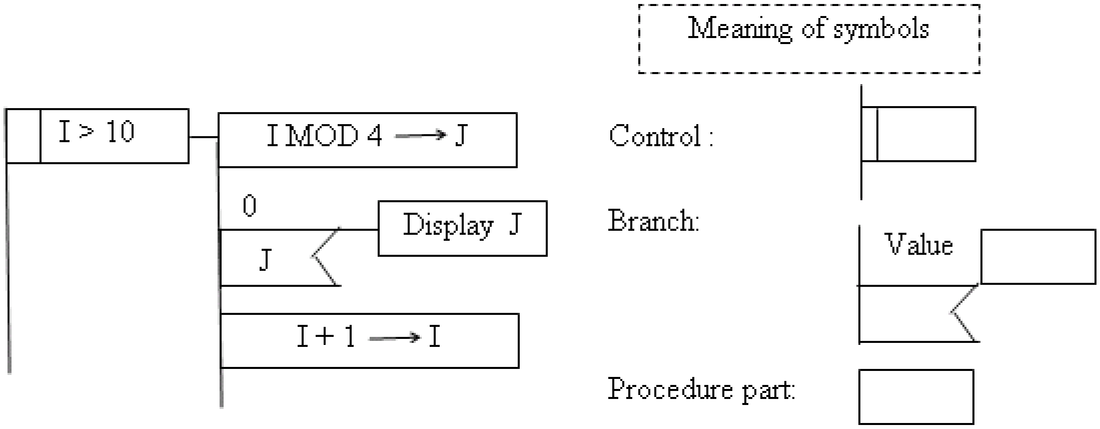
(3) DSD (Design Structure Diagrams)

The DSD is a structured chart that was standardized in the U.K.



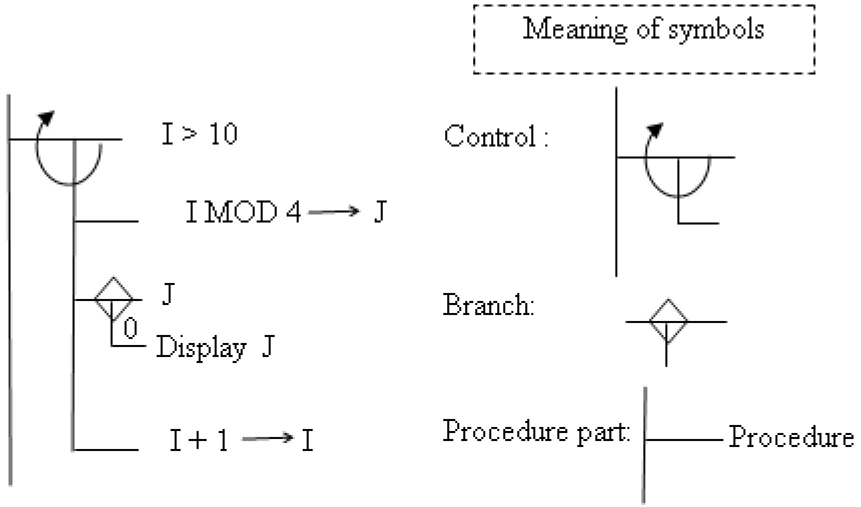
(4) PAD (Problem Analysis Diagrams)

PAD is a structured chart that was standardized by Hitachi.



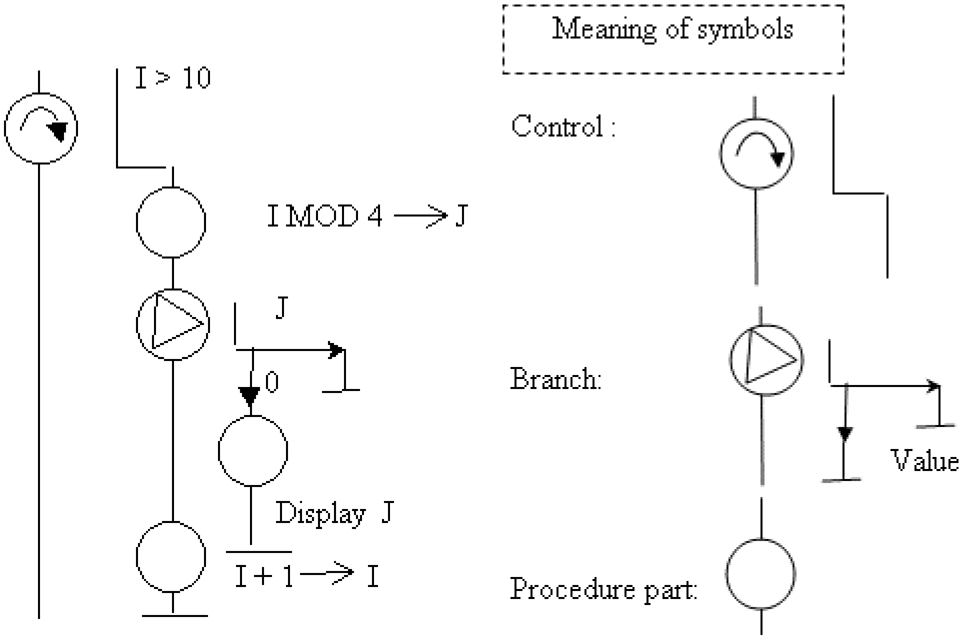
(5) SPD (Structured Programming Diagrams)

The SPD is a structured chart that was standardized by NEC.



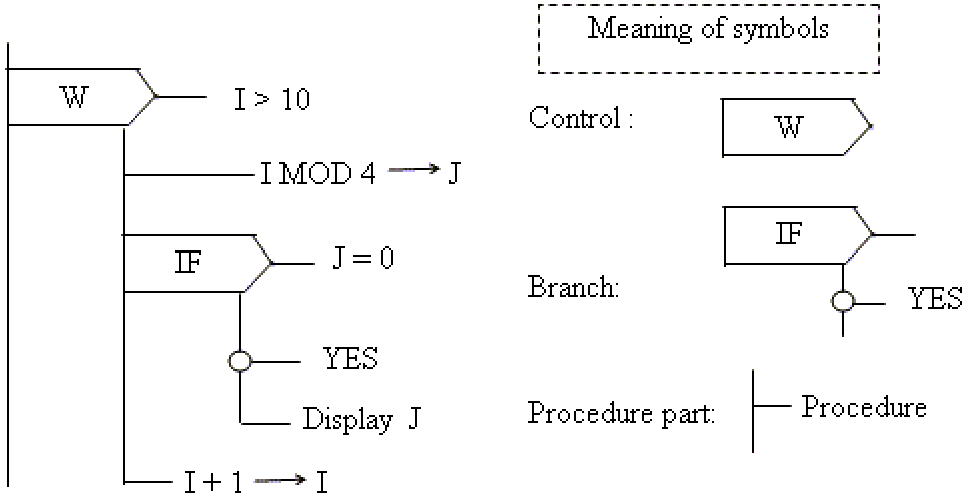
(6) HCP (Hierarchical and Compact Description Chart)

The HCP is a structured chart that was standardized by NTT.



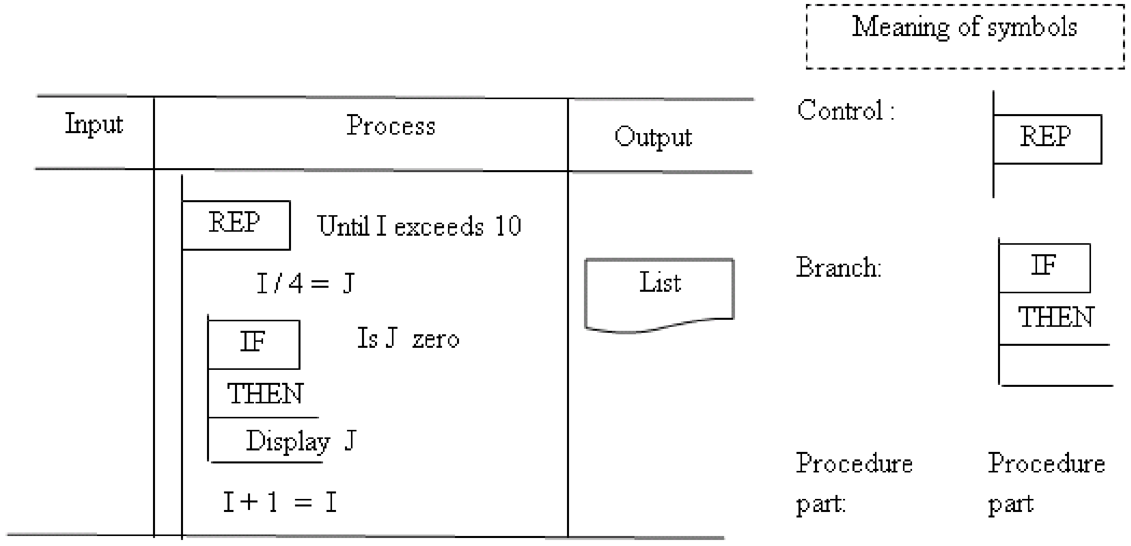
(7) YAC (Yet Another Control Chart)

The YAC is a structured chart that was standardized by Fujitsu.



(8) HIPO (Hierarchy Plus Input Process Output)

The HIPO is a structured chart that was standardized by IBM.



5.2 Logic Programming Paradigm

Logic programming is a programming method that uses a set of declarations instead of procedures.

The concept behind the logic programming paradigm is to describe input-output relationships in logical expressions.

This section explains the notation for these logical expressions.

5.2.1 Prolog (Programming in Logic)

Prolog, unlike procedural programming languages like COBOL or FORTRAN, is a logic programming language that uses logical expressions to write programs.

A logic program represents input-output relationships using logic expressions that are derived from a computational model based on the resolution principle. As a rule, logic programming consists of clauses and the resolution principle, as indicated in the following:

(1) Clause

A clause represents a logical relationship as a conclusion-condition relationship.

The syntax for a clause is as follows:

A1, A2, A3 <= B1, B2, B3

Conclusion Condition

(2) Resolution principle

- The resolution principle is a computerized automatic operation that derives a theorem from axioms. Syllogism is a method of proving the theorem by determining whether any inconsistencies occur between a set of axioms represented in logical expressions and the negation of the theorem.
- Prolog programs, written on the basis of the resolution principle, consist of rules, facts, and queries.

1) Rule (combination of a conclusion and a condition)

SLEEP(A): - ANIMAL(A). => All animals sleep.

2) Fact (conclusion)

ANIMAL(CAT). => The cat is an animal.

3) Query

? - ANIMAL(A). => What do animals do?

4) Conclusion

Let us apply the fact in 2) to the query. The query is represented as:

? - ANIMAL(CAT).

This shows that A = CAT. Next, apply this to item 1) (rule). The rule is represented as:

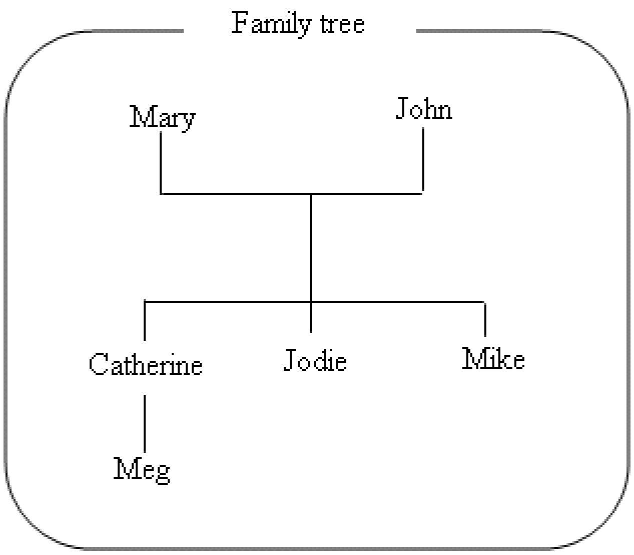
SLEEP(CAT): ANIMAL(CAT).

Thus, SLEEP(CAT). => We can thus derive "the cat sleeps."

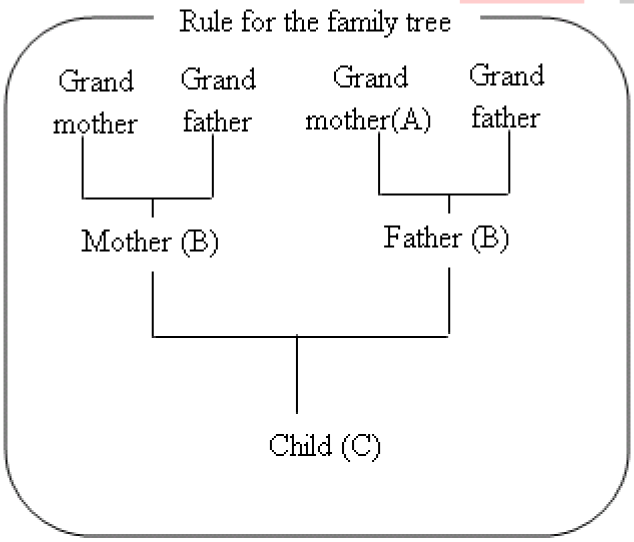
- Prolog programs consist of a set of Horn clauses.
Horn clauses are clauses that represent facts in the form listed in 2) and include only one conclusion section.

5.2.2 Unification

Unification means executing a program by matching facts and queries to rules.



Fact 1: Mother (Mary, Mike).
Fact 2: Mother (Mary, Catherine).
Fact 3: Mother (Mary, Jodie).
Fact 4: Mother (Catherine, Meg).
Fact 5: Father (John, Mike).
Fact 6: Father (John, Catherine).
Fact 7: Father (John, Jodie).



Rule 1: Grandmother (A, C) :- Mother (A, B), Mother (B, C).
Rule 2: Grandmother (A, C) :- Mother (A, B), Father (B, C),

Here, consider the query "Who is Meg's grandmother?"

This query is represented as follows:
? - Grandmother (A, Meg). C = Meg.

Substituting this into Rule 1, we have:

Rule 1: Grandmother (A, Meg) :- Mother (A, B), Mother (B, Meg).

Such pattern matching to rules is called unification.

5.2.3 Backtracking

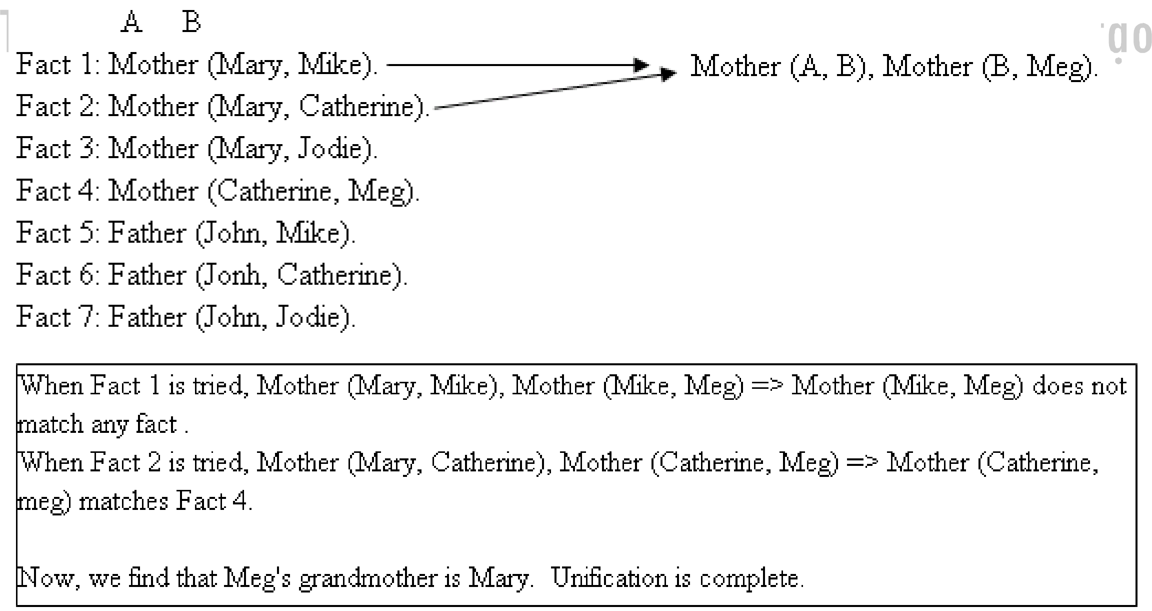
This section explains backtracking as an extension to unification.

When Fact 1 is matched to Rule 1, Rule 1 becomes as follows:
Rule 1: Grandmother (A, Meg) :- Mother (Mary, Mike), Mother (Mike, Meg).

However, Mother (Mike, Meg) is not a fact.

The program will then try to find another match.

This process is referred to as backtracking. Backtracking is repeated until a successful match with a fact is obtained. (See the figure.)



Fact 1: Mother (Mary, Mike).
Fact 2: Mother (Mary, Catherine).
Fact 3: Mother (Mary, Jodie).
Fact 4: Mother (Catherine, Meg).
Fact 5: Father (John, Mike).
Fact 6: Father (Jonh, Catherine).
Fact 7: Father (John, Jodie).

When Fact 1 is tried, Mother (Mary, Mike), Mother (Mike, Meg) => Mother (Mike, Meg) does not match any fact .
When Fact 2 is tried, Mother (Mary, Catherine), Mother (Catherine, Meg) => Mother (Catherine, meg) matches Fact 4.

Now, we find that Meg's grandmother is Mary. Unification is complete.

Conclusion
The key to writing correct programs is to correctly set facts and rules by repeating unification and backtracking.

5.3 Functional Programming Paradigm

Functional programming is a declarational programming method that is based on function descriptions.

The concept behind the functional programming paradigm is to write a program using only functions that are free of side effects, so that the semantics within the program can be appropriately represented.

The following section explains Lisp as an example of functional programming languages.

5.3.1 Lisp (List Processor)

Lisp is a functional programming language that has been widely used in the fields of artificial intelligence (AI) and mathematical processing. At present, this programming language has been standardized as "Common Lisp."

(1) Syntax

In Lisp, every program is a function that is represented in the form (F X Y), where F is the function name and X and Y are arguments.

(2) List

A list is a group of elements. The positions of elements are controlled using pointers.

The following explains atoms and lists:

- Atoms resemble pointer variables. The data type can be any of numeric, character, Boolean, and list. Atoms T and NIL represent Boolean values: T is true and NIL is false.
- A list is a collection of space-delimited atoms enclosed in parentheses.

Examples

- (I MEET BOY)
- (person (height 180) (weight 75) (age 25))

(3) Data formats

There are two data formats: numeric atoms and literal atoms.

Numeric atoms can be used as literal atoms by enclosing them in double-quotes "".

- A numeric atom is a 32-bit integer, not a floating-point number.
- A literal atom can contain any characters except double-quotes ("). Two-byte characters can also be used.

(4) Execution

A functional program is executed by invoking an interpreter that will read the program.

5.4 Object-oriented Programming Paradigm

The concept behind the object-oriented programming paradigm is to describe the behavior and states of objects. Objects are by themselves entities. This section explains typical object-oriented languages including Smalltalk, C++, and Java as well as some object-oriented concepts, such as classes.

5.4.1 Smalltalk

Smalltalk is simply structured, and it includes basic components for development, such as compiler and debugger, and class library.

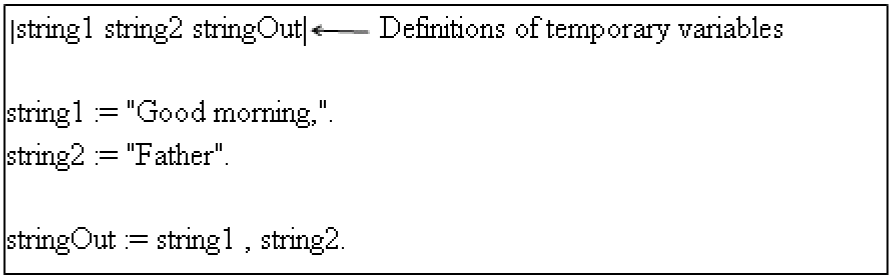
(1) Smalltalk is an object-oriented component-based development tool.

- As a programming language, Smalltalk was the first to rely on object-oriented programming concepts.
- Today, object-oriented programming concepts are also used for Lisp, Prolog, C, and other languages.

(2) Smalltalk provides a superior human-machine interface.

(3) Smalltalk provides a variety of tools that offer a wide range of functions.

Example: To concatenate two character strings "Good morning," and "Father" into a character string "Good morning, Father":



```
|string1 string2 stringOut|←— Definitions of temporary variables

string1 := "Good morning,".
string2 := "Father".

stringOut := string1 , string2.
```

In the example, temporary variable string1 and string2 are assigned the values "Good morning," and "Father", respectively.

The symbol ":=" represents assignment of a value to the variable.

Portions enclosed in double quotation marks are treated as character strings, and a comma "," concatenates the two character strings.

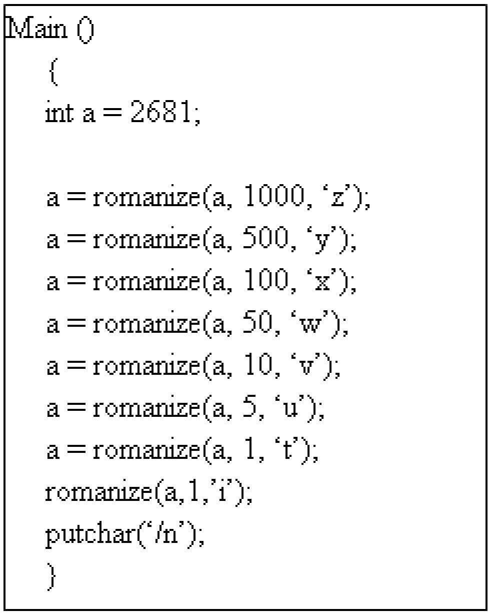
5.4.2 C++

(1) C++ is a composite programming language that was developed based on the C procedural programming language and extended to support object-oriented concepts.

(2) Because C++ is based on C, it inherits C's advantages with respect to execution efficiency and portability.

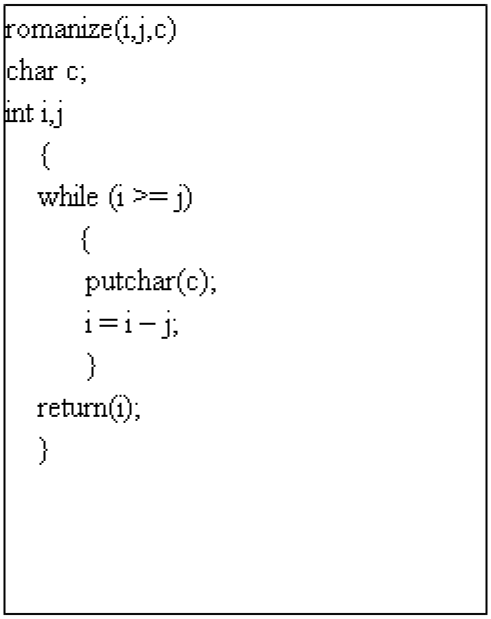
(3) Conversely, because C++ is a composite language, it is difficult for programmers to smoothly apply object-oriented concepts.

Example: The following program outputs 2681 as a string of alphabetic characters as defined within the program:



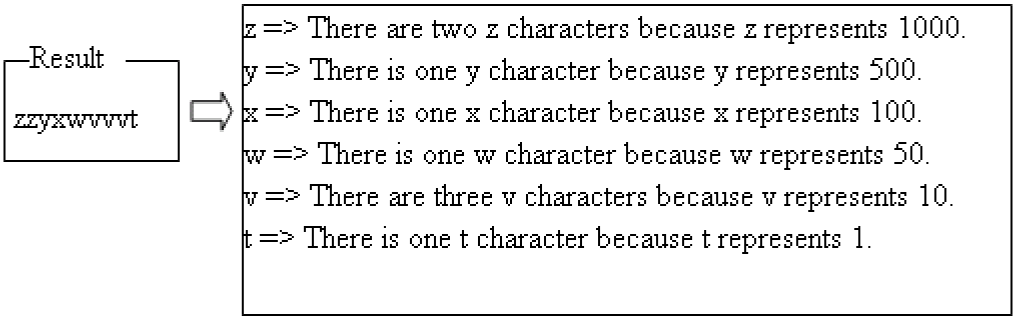
```
Main ()
    {
    int a = 2681;

    a = romanize(a, 1000, ‘z’);
    a = romanize(a, 500, ‘y’);
    a = romanize(a, 100, ‘x’);
    a = romanize(a, 50, ‘w’);
    a = romanize(a, 10, ‘v’);
    a = romanize(a, 5, ‘u’);
    a = romanize(a, 1, ‘t’);
    romanize(a,1,’i’);
    putchar(‘/n’);
    }
```



```
romanize(i,j,c)
char c;
int i,j
   {
   while (i >= j)
       {
        putchar(c);
        i = i – j;
        }
   return(i);
   }
```

The result of execution of the program is as follows:



Result

zzyxwvvvt

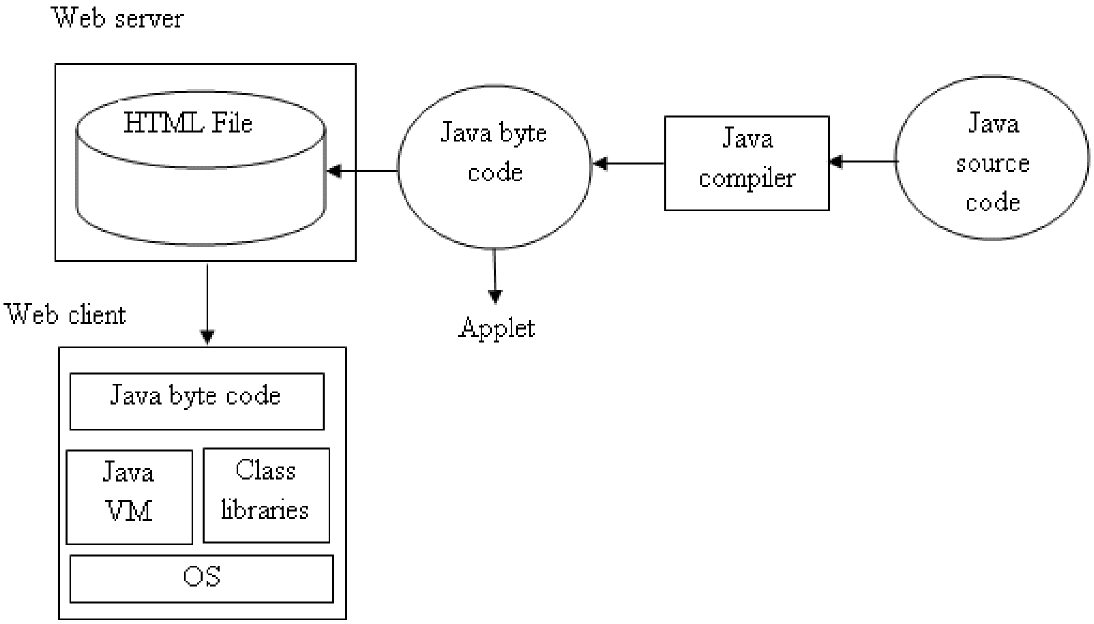
z => There are two z characters because z represents 1000.
y => There is one y character because y represents 500.
x => There is one x character because x represents 100.
w => There is one w character because w represents 50.
v => There are three v characters because v represents 10.
t => There is one t character because t represents 1.

For a number i, character c is displayed as many times as number j exists in j and i-j *(the number of existence of j) is returned.

5.4.3 Java

Java was developed by Sun Microsystems as a built-in language for information terminals. Java began to attract attention when it was extended to a cross-platform language that can operate on any platform where a Java execution environment exists, regardless of the operating environment, such as Windows or Unix.

(1) Operating environment



The procedure for executing a Java program is as follows:

1) Create Java source code (.java).

2) Compile the Java source code using a Java compiler.
Unlike compilers for other programming languages, Java compilers create byte code (.class) instead of executable code. There are three methods for compiling a Java program:

- Java interpreter
- JIT compiler
- HOTSPOT

3) Created byte code will be executed in a virtual execution environment called JavaVM (Java Virtual Machine). Because the byte code is independent of the CPU, it can be executed on any platform on which a JavaVM exists. There are two Java program types: applications and applets. Applications are directly executed on the local JavaVM, whereas applets are executed on JavaVMs residing on Web browsers.

(2) Language specifications
The Java language is based on the C++ language.

The following explains the Java language using C++ concepts:

1) Data
- Java does provide not only all basic C++ data types, but also provides additional data types that do not exist in C.
- Java uses class variables, instead of global variables used in C++. Neither pointers nor variable argument lists are available.
- Arrays are supported, but are implemented using classes instead of structures and unions.

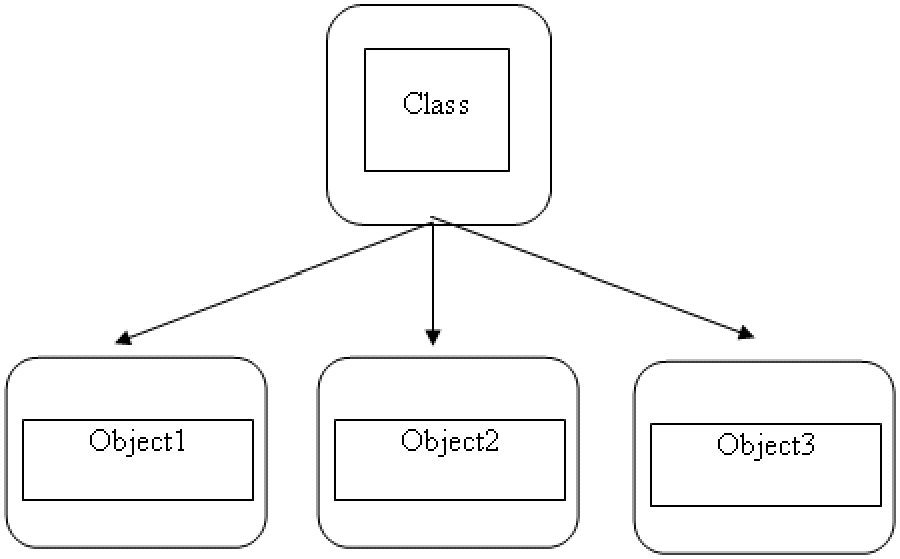
2) Procedures
- Most operators in Java are the same as those in C++, although some differences exist.
- Most control statements in Java are the same as those in C++.
- Java uses CONTINUE and BREAK statements instead of GO TO statements.

3) Object-oriented functions
As a rule, Java supports all object-oriented functions that are supported in C++.

(3) JDK (Java Development Kit)
The JDK is a Java development toolkit that consists of class library packages.
The JDK contains class libraries called Java APIs (Java Application Programming Interfaces).

5.4.4 Instantiation of classes

Instantiation is a function that efficiently creates objects from a class that has a specific value.

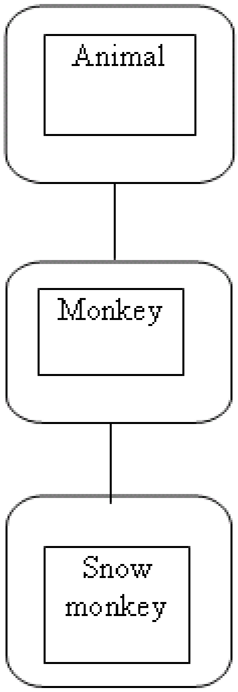


- A class is a modular unit that describes the common properties of a set of members (objects). Classes are selected under control of applications and include only important properties defined.

- In the example, Objects 1, 2, and 3 include a common method (procedure). However, if their internal states do not fulfill the applicable conditions, a different object will be created.

5.4.5 Class hierarchy and inheritance

Inheritance is a mechanism in which a child class inherits all of the properties of its parent class.



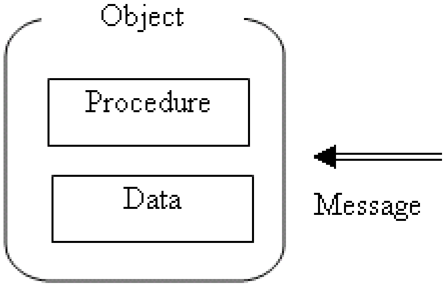
(1) Class hierarchy denotes a parent-child relationship between classes as shown in the figure. This parent-child relationship is also called an is-a relationship (structure).

(2) In the above class hierarchy, one can state the following:
- Snow monkey inherits from Monkey.
- Snow monkey is a subclass of Monkey, and Monkey is a superclass of Snow monkey.
- Monkey inherits from Animal.
- Monkey is a subclass of Animal and Animal is a superclass of Monkey.

From these facts, it can be said that Snow monkey inherits from Animal as well.

5.4.6 Encapsulation as data abstraction

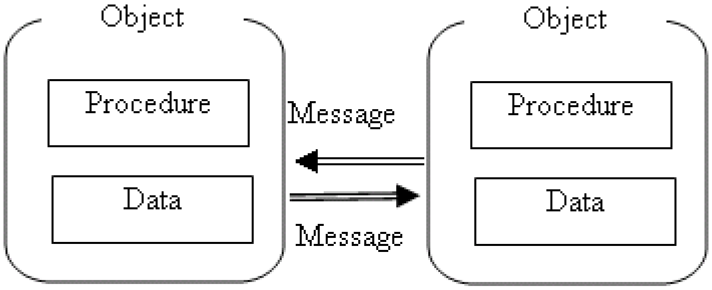
Encapsulation refers to combining data and the procedure for data handling in one package. The encapsulated data and procedure cannot be directly handled from outside.



(1) By achieving data abstraction, it is only necessary to modify the data structure when data is changed. Data abstraction thus increases reliability and maintainability of the program.

(2) Data owned by an object can be referenced only from the procedure within that object because of encapsulation. This is useful for information hiding.

5.4.7 Message passing and polymorphism



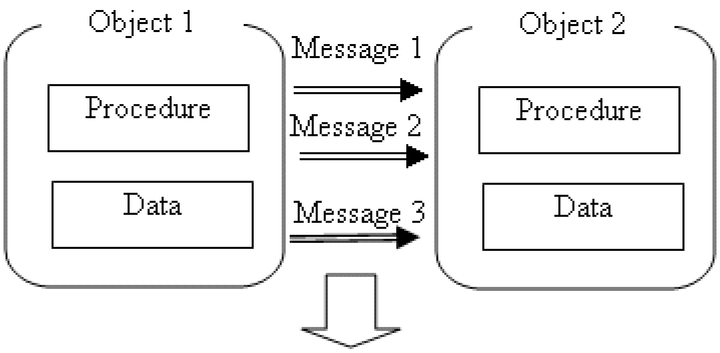
(1) Message passing

Message passing is a mechanism by which objects can send messages to, or receive them from each other for the purpose of solving problems via information exchange.

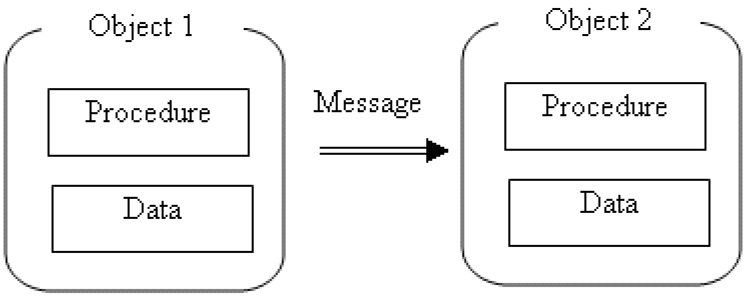
1) Message passing is a means for communication between processes. In the above example, messages are objects.

2) Sending a message means making the destination object apply the procedure.

Example 1



Example 2



(2) Polymorphism

Polymorphism is the ability of a single operation to act differently on instances (objects) derived from different classes. See the examples. In Example 1, object 1 sends specific messages to object 2. Conversely, in Example 2, object 1 only sends a simple message. This mechanism is called polymorphism. Object 2 interprets the received message. This assures that object 1 will not be affected by modifications to object 2.

Exercises

1. The following sentences explain three basic control constructs. Choose the correct phrase from the list below and put them in the blank spaces.

- (1)_____ do not include (2)_____ or (3)_____ and processes are executed in sequence from the top down.
- (2)_____ can be divided into two types: (2)_____ that use a single condition and (4)_____ that use multiple conditions.
- (3)_____ can be divided into two types: (5)_____, where the termination condition is evaluated before execution of the process and (6)_____, where the termination condition is evaluated after execution of the process.

Answers:
a. selections
b. loops with the termination condition at the bottom
c. sequences
d. multiple-way branches
e. loops with the termination condition at the top
f. GO TO statements
g. iterations

2. Choose the correct sentence from the statements below, which explain programming styles:

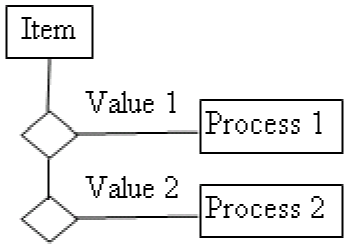
(1) Clarity refers to a programming style that results in readable and easy-to-understand programs. In structured programming, clarity is achieved by using GO TO statements.

(2) Efficiency refers to a programming style that focuses on program execution time and the storage area to be used by the program.

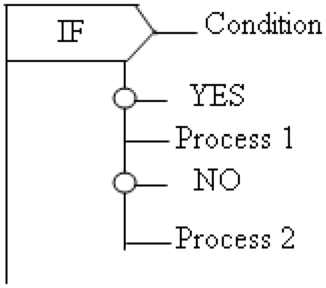
(3) Maintainability affects the length of the software lifecycle. The length of the program lifecycle varies depending on the extent to which the program is structured.

3. The following diagrams show the decision section in structured charts. Choose the correct name of each structured chart from the list.

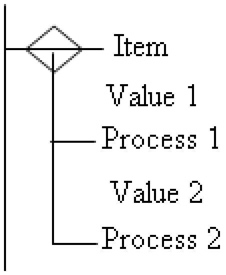
(1)



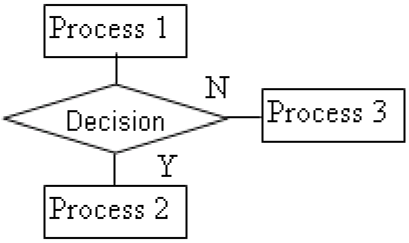
(2)



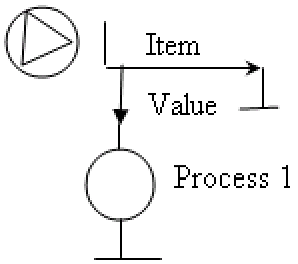
(3)



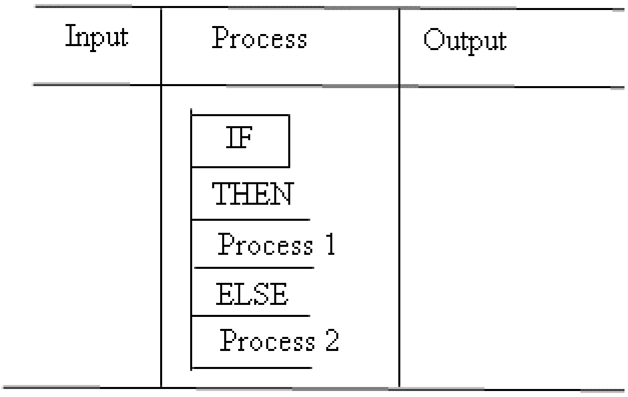
(4)



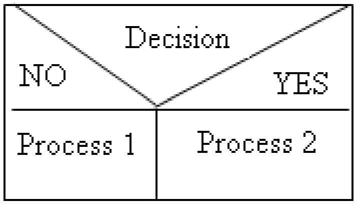
(5)



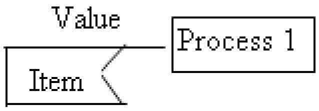
(6)



(7)



(8)



Answers:
a. NS chart
b. DFD
c. DSD
d. Flowchart
e. SPD
f. HIPO
g. YAC
h. HCP
j PAD

4. Choose the correct sentence from the following statements, which explain the logical programming paradigm:
(1) Prolog is a programming language that is called a procedural programming language.
(2) Prolog is a language for describing logical relationships and that consist of rules, facts, and queries.
(3) Horn clauses are clauses that include only one conditional section.

5. Consider the family that satisfies the following conditions. Perform unification to find who Meg's grandmother is.
Fact 1: Mother (Mary, Mike).
Fact 2: Mother (Mary, Catherine).
Fact 3: Mother (Mary, Jodie).
Fact 4: Mother (Catherine, Meg).
Fact 5: Father (John, Mike).
Fact 6: Father (John, Catherine).
Fact 7: Father (John, Jodie).

Rule: Grandmother (A, C) :- Mother (A, B), Mother (B, C).

6. How many times must backtracking be performed to find the answer to Question 5?

7 The following sentences are statements about functional programming. Choose the correct phrase from the list below and put them in the blank spaces.

- All programs are written as (1)_____.
- A functional program is executed by having an (2)_____ read the program.
- Lisp is a functional programming language that has been used in the fields of (3)_____ and (4)_____.

Answers:
a. compiler
b. functions
c. pictorial processing
d. artificial intelligence (AI)
e. interpreter
f. formula processing

8. The following sentences explain data handling in functional programming. Put the correct words in the blank spaces.

- There are two types of data structures: (1)_____ and (2)_____.
- (1)_____ are similar to pointer variables. Data types include numeric, character, and Boolean.
- (2)_____ are collections of space-delimited(1)_____ enclosed in parentheses.
- There are two data formats: (3)_____ and (4)_____.

9. Choose the incorrect statement from the following sentences that are statements about Smalltalk:

(1) Smalltalk provides a superior human-machine interface.
(2) Smalltalk is an object-oriented component-based development tool.
(3) Smalltalk provides a variety of tools that offer a wide range of functions.
(4) Smalltalk is characterized by its complex language structure and includes development environment elements as objects.

10. Choose the correct sentence from the following statements about C++:

(1) C++ is a composite programming language that was developed based on the C language, but does not support object-oriented concepts.
(2) Because C++ is a composite language, it is difficult for programmers to smoothly apply object-oriented concepts.
(3) Because C++ is based on C, it is disadvantageous with respect to execution efficiency, but advantageous with respect to portability.

11. Answer the following questions about Java compilers:

(1) What is the name of the Java compiler that can achieve reduction in loading time because it compiles only frequently used code to machine language?

(2) What is the name of the Java compiler that provides advantages with respect to execution speed because it compiles byte code while recording the results in memory (the byte code can be executed without an interpreter)?

(3) What is the name of the Java compiler that compiles source code to a machine language similar to compilers for other languages?

12. The following sentences are statements about Java. Choose the correct phrase from the list below and put them in the blank spaces.

- Java not only provides all basic C++ (1)____, but also provides additional (1)____ that do not exist in C.
- Java language uses (2)_____, instead of the global variables used in C++.
- Most (3)_____ and (4)_____ are the same as those in C++.
- Java uses (5)_____ and (6)_____ instead of GO TO statements.
- As a rule, Java supports all the (7)_____ that are supported in C++.

Answers:

a. CONTINUE statements

b. processing statements

c. operators

d. object-oriented features

e. numeric type

f. data types

g. control statements

h. class variables

i. BREAK statements

j. encapsulation

13. Answer the following questions about classes:

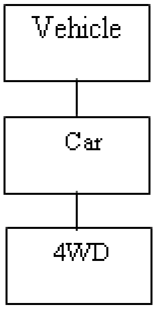
(1) What is the name of a modular unit that describes the common properties of a set of members (objects)?

(2) What is the name of the feature that creates an object with a specific value from a class?

(3) What is the name of the structure that represents a parent-child relationship between classes in a class hierarchy?

(4) When two classes are in a parent-child relationship, the child class inherits the properties of the parent class. What is the name of this feature?

14. The following sentences explain the class hierarchy and inheritance. Choose the correct phrase from the list below and put them in the blank spaces by referring to the figure below. Any word can be used as many times as required.



- 4WD (1)_____ Car.
- 4WD is a (2)_____ of Car and Car is a (3)_____ of 4WD.
- Car (4)_____ Vehicle.
- Car is a (5)_____ of Vehicle and Vehicle is a (6)_____ of Car.

Answers:
a. subclass
b. Vehicle
c. inherits from
d. superclass
e. depends on
f. 4WD

15. The following sentences are statements about explain data abstraction and encapsulation. Fill the blank spaces.
- Data abstraction increases (1)_____ and (2)_____ of the program, because it minimizes the work required when a modification occurs.
- Encapsulation refers to combining (3)_____ and the (4)_____ for (3)_____ handling in one package.
- (3)_____ owned by an object can be referenced only from the (4)_____ within that object because of (5)_____.

16. Choose the incorrect statement from the following statements about message passing and polymorphism:
(1) Message passing is a means for communication between data entities.
(2) Making an object apply a procedure means sending a message to the object.
(3) With polymorphism, the message receiver interprets the message.

6 Software Quality

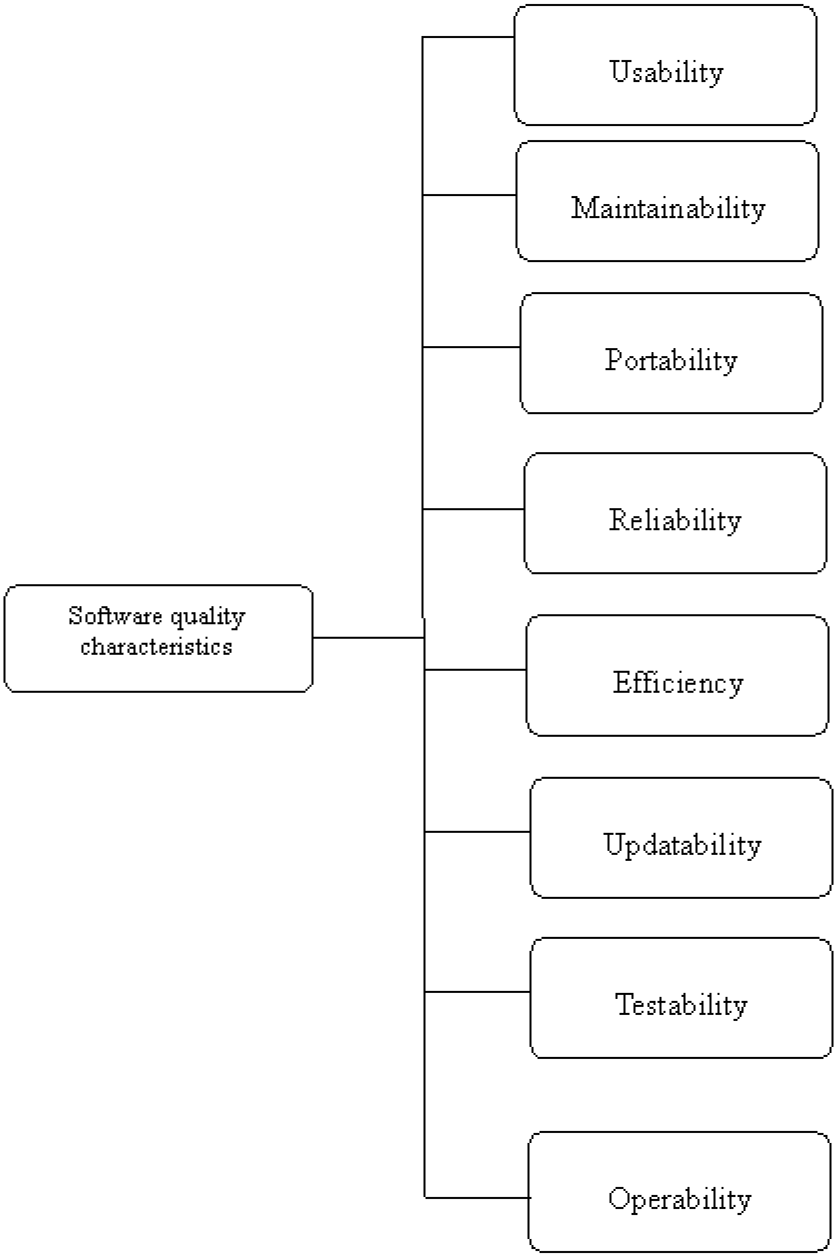
Chapter Objectives

This chapter explains software quality control and testing techniques.

6.1 Software Quality Factors
6.2 Program Testing
6.3 Quality Control of Software

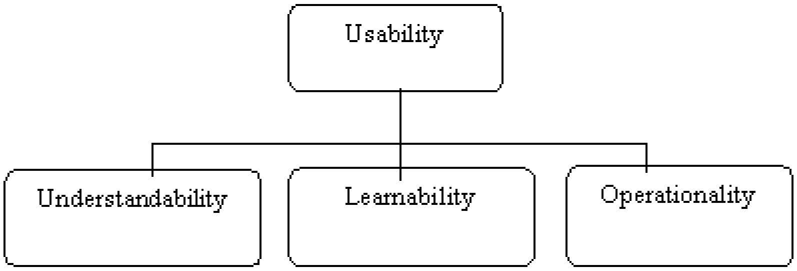
6.1 Software Quality Factors

The software quality factors are shown in the following diagram. Each of these items is explained in the following pages.



6.1.1 Usability

Usability is a factor that expresses the ease with which the software can be used by the user. Usability can be further divided into the following three sub-factors:



(1) Understandability

Understandability refers to a factor which expresses how much effort is required by the user to understand the logical concept behind the software and how to use the software.
As a measure of understandability, the degree of fulfillment in the applied model is used:

Degree of fulfillment in the applied model

$$\frac{\text{Number of functions clearly explained in the model}}{\text{Number of all functions}}$$

(2) Learnability

Learnability expresses how much effort is required to learn about the software.
As a measure of learnability, the operation familiarization time is used:

Operation familiarization

Time required for the user to become competent in using the software

(3) Operability

Operability expresses the effort required by the user to conduct operation control and operation of software. As a measure of operability, the average operation error interval and the responsive time are used.

Average operation error interval

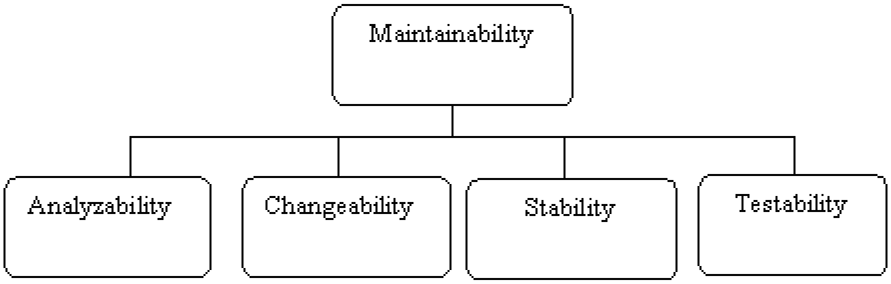
$$\frac{\text{Total operation time}}{\text{Number of error occurrences in operation}}$$

Responsive

The time when the first response was received from the system minus the time when the entry by the operator finished

6.1.2 Maintainability

Maintainability expresses how easy it is to perform changes or revisions. Maintainability can be further divided into the following four sub-factors:



(1) Analyzability

Analyzability expresses the effort required to check parts for revision and diagnosing failure causes or defects. As a measure of analyzability, the average failure analysis time and diagnostic function installation ratio are used.

Average failure analysis time

$$\frac{\text{Total time spent on failure analysis}}{\text{Failure count}}$$

Diagnostic function installation ratio

$$\frac{\text{Number of diagnostic functions already installed}}{\text{Number of diagnostic functions to be installed}}$$

(2) Changeability

Changeability expresses how much effort is required to adapt to changes in the environment, for example, due to a revision. As a measure of changeability, the average fault correction time is used.

Average fault correction time

$$\frac{\text{Total time required for fault correction}}{\text{Fault count}}$$

(3) Stability

Stability expresses to what extent the software is affected by such unexpected changes as changes due to a revision. As a measure of stability, the failure cause rate is used.

Failure cause rate

$$\frac{\text{Number of failures caused by changes}}{\text{Number of changed lines in the source code}}$$

(4) Testability

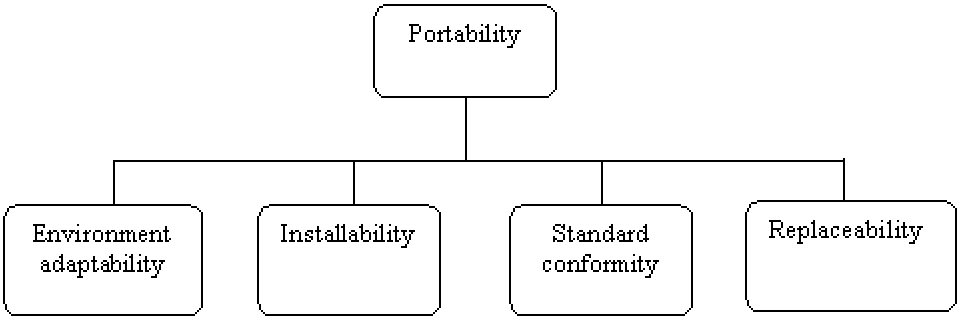
Testability expresses how easy it is to validate the revised software.
As a measure of testability, the test time per line of changed source code is used.

Test time per line of changed source code

$$\frac{\text{Test time}}{\text{Number of lines changed in source code}}$$

6.1.3 Portability

Portability expresses how easy it is to port the software from one environment to another. Portability can be further divided into four sub-characteristics.



(1) Environment adaptability

Environment adaptability expresses whether software can be applied in different environments without modification.
As a measure of environment adaptability, the adaptable machine type rate is used.

Adaptable machine type rate

$$\frac{\text{Number of adaptable machine types}}{\text{Number of specified machine types}}$$

(2) Installability

Installability expresses the effort required to install software in a certain environment.
As a measure of installability, the parameter change rate and database change rate are used.

Parameter change rate

$$\frac{\text{Number of parameters that require change}}{\text{Total number of parameters in ported software}}$$

Database change rate

$$\frac{\text{Number of records that require change}}{\text{Total number of records in ported database}}$$

(3) Standard conformity

Standard conformity expresses to what extent standards or rules related to software porting are used.

As a measure of standard conformity, the standard conformance rate is used.

Standard conformance rate

$$\frac{\text{Number of standard items the software complies with}}{\text{Number of standard items to be complied with}}$$

(4) Replaceability

Replaceability expresses how much effort is required to replace software with other software.

As a measure of replaceability, the function change rate and performance maintenance rate are used.

Function change rate

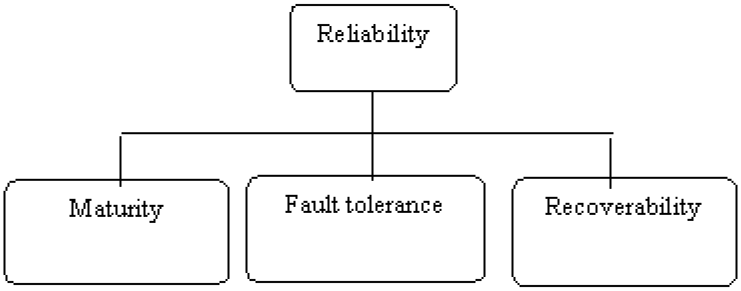
$$\frac{\text{Number of functions changed during porting}}{\text{Number of ported functions}}$$

Performance maintenance rate

$$\frac{\text{Number of performance items that can be maintained even after porting}}{\text{Number of performance items of software}}$$

6.1.4 Reliability

Reliability expresses to what degree the software can maintain its performance without failures or malfunctions for a specified period of time under explicit conditions. Reliability can be further divided into three sub-factors:



(1) Maturity

Maturity expresses the degree to which the frequency of potential defects affects the system.

As a measure of maturity, the average failure interval is used.

Average failure interval

$$\frac{\text{Total operation time of the software system}}{\text{Failure count}}$$

(2) Fault tolerance

Fault tolerance expresses to what degree the required functions of software are provided even if conditions related to the interface are not met. As a measure of fault tolerance, the system down rate and error input/error operation detection rate are used.

System down rate

$$\frac{\text{System down count}}{\text{Failure count}}$$

Error input/error operation detection rate

$$\frac{\text{Number of detected error inputs/error operations}}{\text{Number of recorded error inputs/error operations}}$$

(3) Recoverability

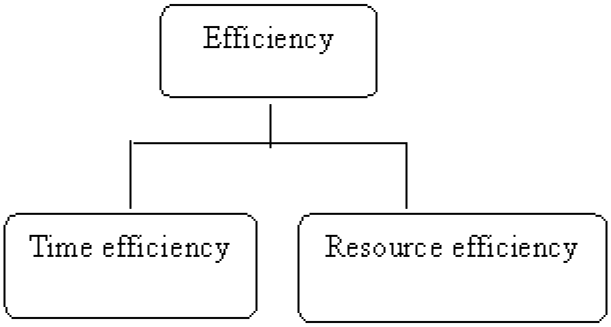
Recoverability expresses the time and effort required for recovery from software problems. As a measure of recoverability, the average recovery time is used.

Average recovery time

$$\frac{\text{Total of time periods between the times when the system went down and the times when processing was restarted}}{\text{System halt count}}$$

6.1.5 Efficiency

Efficiency expresses software performance in relation to the amount of resources used under certain conditions. Resources include other software products, hardware facilities, and services from personnel. Efficiency can be further divided into two sub-factors.



(1) Time efficiency

Time efficiency expresses the response time and the throughput within a specified time. As a measure of time efficiency, response time and throughput are used.

Response time

Time that elapses between the issuance of a request to the system and the receipt of any result

Throughput

Workload handled within a specified time.
Online system -> number of inquiries,etc.
Batch system -> number of output pages,etc.

(2) Resource efficiency

Resource efficiency expresses the amount of resources required for processing and the time required to use these resources. As a measure of resource efficiency, CPU usage, file usage, and CPU usage rate are used.

CPU usage

Time during which the CPU is used from start to finish of program execution

File usage

File size required by the operating environment when a program is executed

CPU usage rate

$$\frac{\text{Total time the CPU is used}}{\text{Total time of system operation}}$$

6.1.6 Updatability

Updatability expresses how easy it is to change the program source code and documents.

6.1.7 Testability

Testability expresses whether a test can be performed easily. To increase testability, the program needs to be structured in such a way that test cases can be easily created.

6.1.8 Operability

Operability expresses whether the user can operate the software easily.

6.2 Program Testing

The purpose of program testing is not to verify that the program is totally free of errors by performing tests based on the specifications and detecting potential errors; it is to improve program reliability by detecting as many errors as possible.

This section explains various test methods.

6.2.1 Technique for designing testcases

(1) Black box test

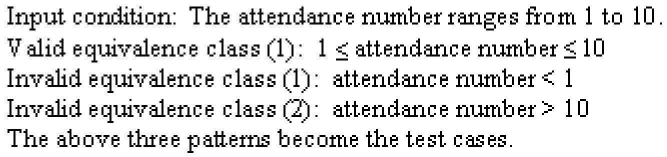
The black box test is a program test of the external specifications and mainly checks the interface functions and I/O functions. For the black box test, test techniques such as equivalence partitioning, boundary value analysis, cause-effect graphs, and error estimations are used.

1) Equivalence partitioning

During equivalence partitioning, both valid values and invalid values are entered with respect to the input specification conditions of the program.

- Valid values: These values are values for which processing is performed normally. They are so-called valid equivalence class.
- Invalid values: These values are values for which processing is not performed normally. They are so-called invalid equivalence class.

Example



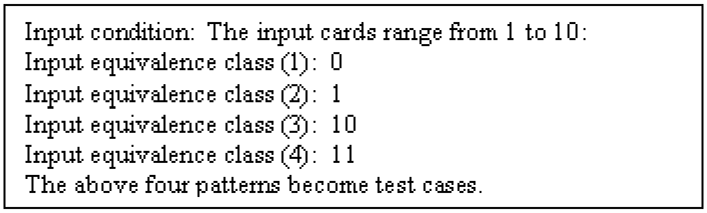
Input condition: The attendance number ranges from 1 to 10.
Valid equivalence class (1): 1 ≤ attendance number ≤ 10
Invalid equivalence class (1): attendance number < 1
Invalid equivalence class (2): attendance number > 10
The above three patterns become the test cases.

2) Boundary value analysis

The boundary value analysis is a test method where boundary values and their preceding and succeeding values are applied with respect to the I/O specification conditions.

This technique is mainly used to make sure that the initial values and end values for iterative processing are correct.

Example



Input condition: The input cards range from 1 to 10:
Input equivalence class (1): 0
Input equivalence class (2): 1
Input equivalence class (3): 10
Input equivalence class (4): 11
The above four patterns become test cases.

3) Cause-effect graph

In testing with a cause-effect graph, the input specification conditions and output results are represented in relational graphs and a decision table is created from these relational graphs.

- Cause: Input data of the program (behavior status)
- Effect: Output data of the program (execution result)

The specific procedure is as follows:

a. Assign a number for identifying the input data within the input specification conditions for the program, and then extract that data.

b. Assign a number for identifying the output data within the output specification conditions for the program, and then extract that data.

c. Create a cause-effect graph.

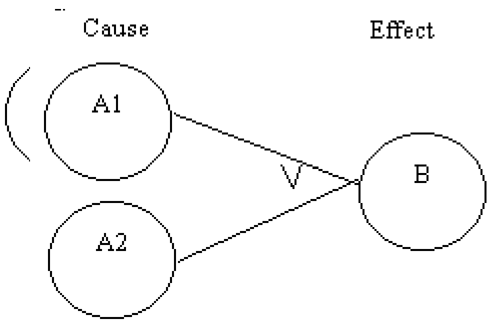
d. Assign symbols to lines connecting the causes and effects to indicate their relationship.

e. Assign symbols to causes to indicate relationships between mutual causes, and indicate constraints among causes.

f. Create a decision table based on the cause-effect graphs.

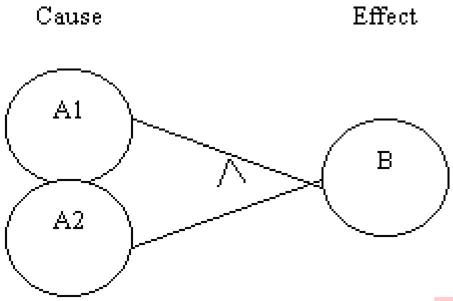
The following explains the symbols to be applied in steps d and e.

When representing the relationships in accordance with d, equivalence, negation, sum, and product are available.



- Equivalence (If A, then B)

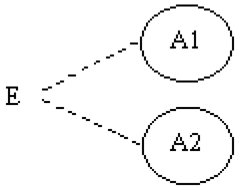
- Sum (If A1 or A2, then B)



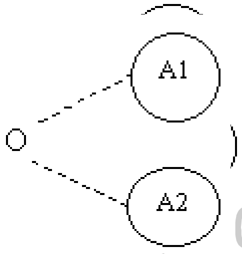
- Negation (If not A, then B)

- Product (If A1 and A2, then B)

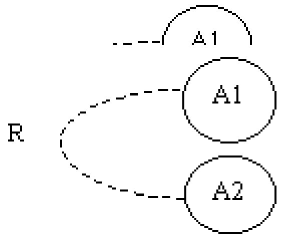
When representing the relationships in accordance with e, exclusion, inclusion, uniqueness, premise, and mask are available



- Exclusion (either of A1 or A2)



- Inclusion (at least one of A1 and A2)
- Only one(only A1 or A2)



- Premise (If A1 exists, then A2 exists also)
- Mask (If A1 exists, then A2does not)

4) Error estimation

Error estimation is a method of performing tests by preparing test cases for situations in which errors are likely to occur. This method has not yet been established as commonly used test technique.

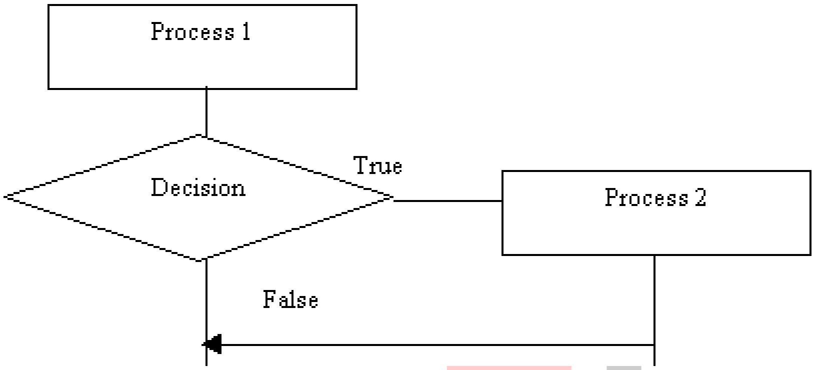
(2) White box test

The white box test is a test of the internal specification of the program and mainly uses the detailed specifications of algorithms and program sources for verification. As test cases of the white box test, such cases as instruction coverage, decision condition coverage, and condition coverage are used.

1) Instruction coverage

Instruction coverage is a case in which each instruction in the program is tested at least once.

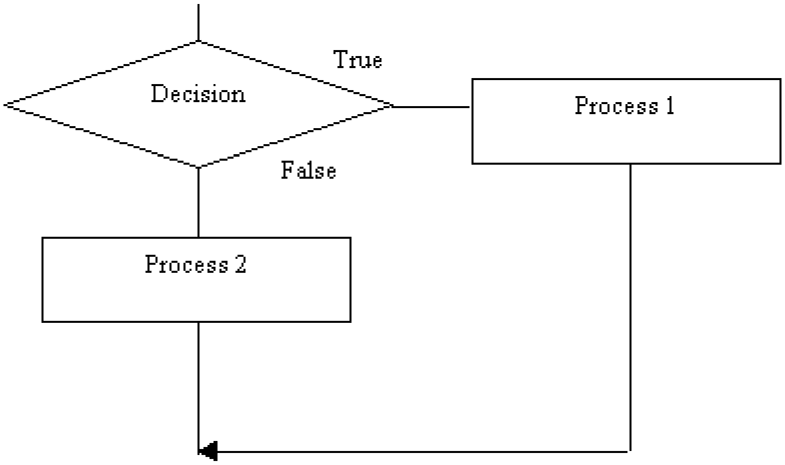
If a value that meets the decision condition is set, all the instructions in both process 1 and process 2 can be executed.



2) Decision condition coverage

Decision condition coverage refers to a test case for testing the processing for each decision condition when the decision condition is at least once true and once false.

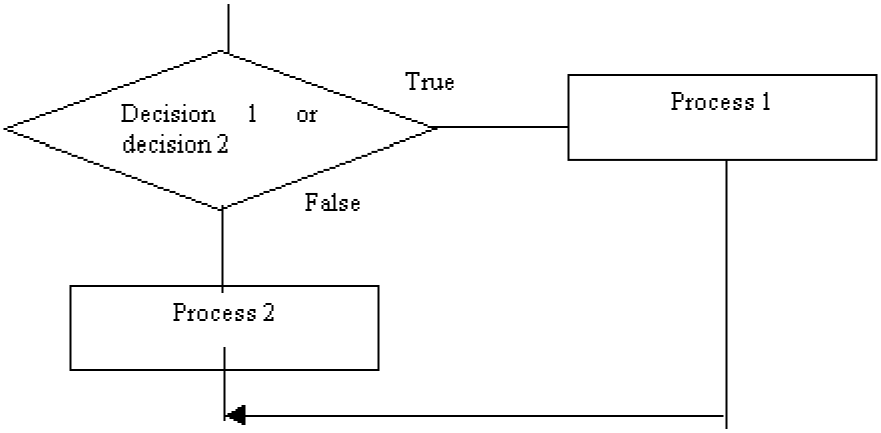
If a value that meets the decision condition is set, process 1 can be performed. If a value that does not meet the decision condition is set, process 2 can be performed



3) Condition coverage

Condition coverage refers to a test case in which the processing for all combinations of true and false in the decision conditions are tested.

If decision 1 is true and decision 2 is false or decision 1 is false and decision 2 is true, the condition becomes true, and only process 1 can be executed.



4) Decision condition and condition coverage

Decision condition and condition coverage refers to a test case in which test parts that cannot be checked by condition coverage alone are tested as a combination of decision condition coverage tests.

If decision 1 is set to false and decision 2 is also set to false, the condition becomes false, in accordance with the flow of the condition coverage, and only process 2 can be executed.

5) Multicondition coverage

Multicondition coverage refers to a test case in which testing is performed by testing the branches of the operation flow for all true or false combinations of the conditions.

If decision 1 is set to true and decision 2 is also set to true, referring to the flow of condition coverage, the condition becomes true and only process 1 can be executed.

6.2.2 Technique for testing module integration

Module integration test is a method in which the relationship between modules is tested. For this type of test, there are such tests as all-together test, big bang test, and incremental test.

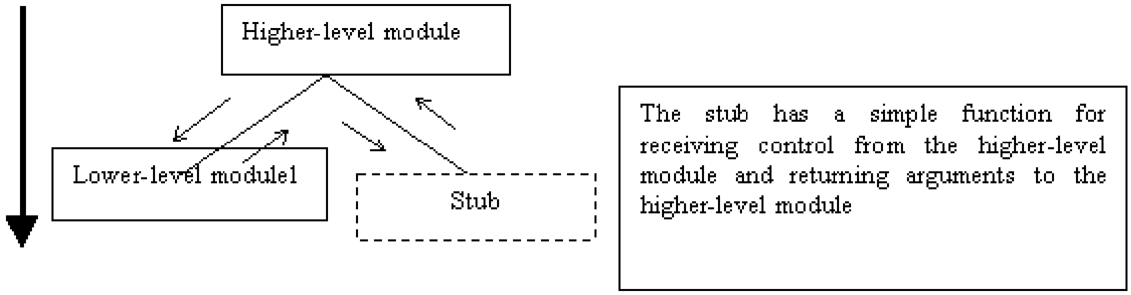
(1) All-together test
In the all-together test, modules for which the unit test has not been performed are combined into a single unit for testing. However, because multiple modules are combined, it will be difficult to identify the location of a fault, if an error occurs.

(2) Big bang test
The big bang test differs from the concurrent test only to the extent that the unit tests have been performed. The test is performed by combining all the modules for which the unit test has been successfully completed into a single unit. In other words, this test basically tests only the interface between modules.

(3) Incremental test
The addition test is similar to the big bang test in that modules are combined for which the unit tests have already been performed. However, in this test, modules are combined one after the other, whereas they are all combined in one large unit in the big bang test. As test methods, the top-down test, bottom-up test, and sandwich test are used.

1) Top-down test
The top-down test is performed by sequentially combining the lower-level modules with the higher-level modules. If a lower-level module were to have to be combined before it was actually finished, the test is performed by replacing the lower-level module with a virtual module called a stub.

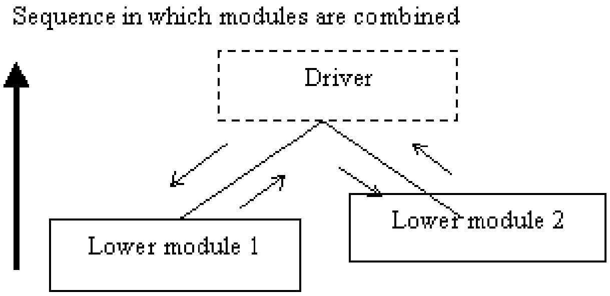
Sequence in which modules are combined



The stub has a simple function for receiving control from the higher-level module and returning arguments to the higher-level module

The top-down test is suitable to test important characteristics, such as the reliability of program control, but requires more labor-hours, because development cannot be carried out in parallel.

2) Bottom-up test
The bottom-up test is performed by sequentially combining lower-level modules with higher-level modules. If a higher-level were to have to be combined before it was actually finished, the test is performed by replacing the higher-level module with a virtual module called a driver.

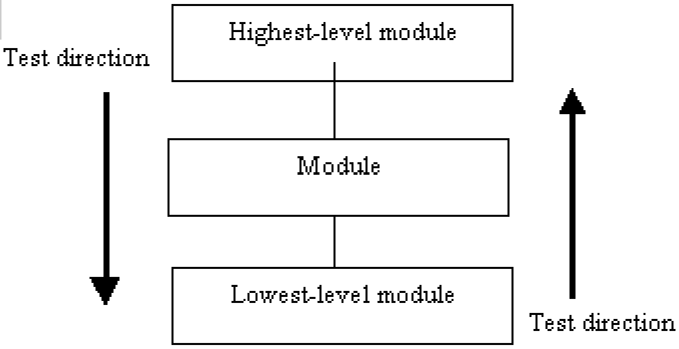


The driver passes the control and parameters to lower-level modules. The test is then performed based on the returned arguments.

Modules can be developed in parallel when this test is applied, but the test is not really extensive enough, because important parts, such as program control, are tested last with this approach.

3) Sandwich test

In the sandwich test, testing is performed in both directions, both starting from the highest-level module and from the lowest-level module. This test incorporates the advantages of both the top-down test and bottom-up test.



6.2.3 Static tests and dynamic tests

Program tests can be divided into static tests and dynamic tests, depending on whether the program to be tested is executed for the test or not.

(1) Static tests

Static tests are tests performed without executing programs. In this case, the program source code is tested as such. Various tools are used for static tests, such as source code analyzers and program structure analyzers.

1) Source code analyzer

A source code analyzer is a tool for analyzing the following points by entering the source code of the program.

- Is the source code in accordance with the coding rules?
- Are there any unnecessary instructions?
- Do the interfaces between modules match?

2) Program structure analyzer

A program structure analyzer analyzes the program hierarchies after entering the source code of the source program.

(2) Dynamic tests

Dynamic tests are tests in which results are checked after actually executing programs. For dynamic tests, such tools as test data creation tools, test coverage tools, test bed tools, and program validation tools, are used.

1) Test data creation tool

The test data creation tool is a tool that automatically creates test data based on the data structures, such as file description statements and database schema statements.

2) Test coverage tool

The test coverage tool is a tool for investigating what percentage of the program instructions or decision conditions are covered by the created test data.

3) Test bed tool

The test bed tool, a tool for providing an operating environment, is used to create a stub or driver as required for the incremental test.

4) Program validation tool

The program validation tool is a tool for validating programs. Partial validation and total validation must be carried out.

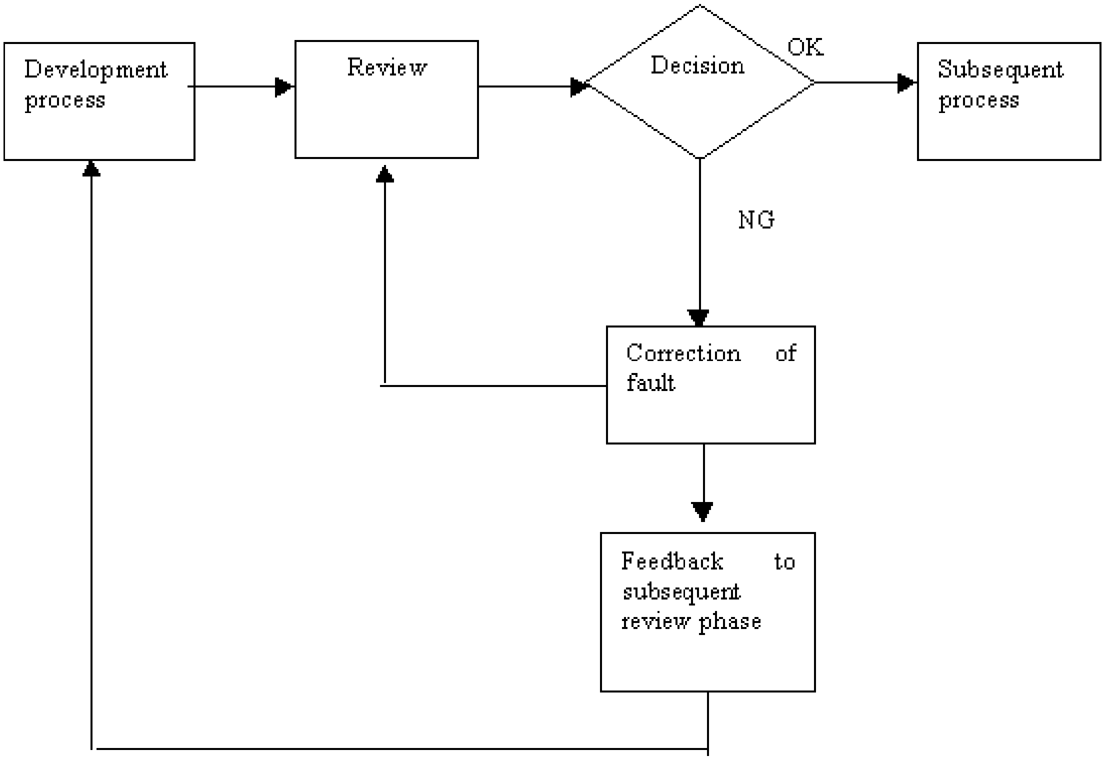
6.3 Quality Control of Software

This section explains the quality control of software.
Quality control of software refers to improving software quality by such measures as review, reliability estimation, and QC (quality control).

6.3.1 Review method

As review methods, such methods as design review for each development process, code review, etc., are used. Walkthrough and inspection are effective techniques in the internal design and program design phases.

Flow of the review



(1) Design review
The design review is useful for quality control of software.

1) Purposes of the design review
- Early detection of malfunctions and errors and ensuring that mistakes or misunderstanding do not affect subsequent design phases.
- Understanding the progress and checking the completion of the relevant design phase
- Clarifying the design of subsequent design phases
- Analyzing product quality status for feedback into subsequent review phases

2) Necessity of design review
The design review should be carried out so that mistakes or misunderstanding by the designer can be detected early in the development process to ensure that subsequent design phases are not affected.

(2) Walkthrough
The walkthrough is a short-term review that consists of a preparation phase, meeting

phase, and follow-up phase.

1) Purposes of the walkthrough
- Enhancing communication in the team
- Simulating actual operations for error detection
- Documenting errors

2) Necessity of the walkthrough
The walkthrough is a means of checking the validity of the conditions for the usage environment and procedures, as well as the role of individual functions, by preparing test data for each test and performing simulations. The important task in this phase is to check whether input/output data is adequate.

(3) Inspection
The inspection, a means of checking the review targets, has the function of an overall examination.

1) Purposes of the inspection
- Detection and correction of any error in the products at the end of each design phase.
- Collection of error information to prevent error occurrence or improve the error detection rate.

2) Necessity of the inspection
- Checking the correctness of the review targets.
- Checking the products of each design phase (plans, requirement specifications, external design specifications, internal design specifications, program specifications, and operation specifications)

(4) Code inspection
This review focuses on the source code, and the program listing is checked line by line.

6.3.2 Reliability estimation method

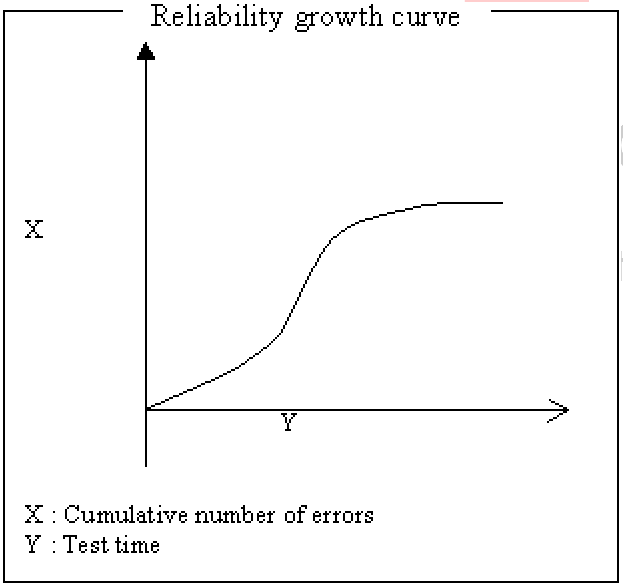
Reliability estimation refers to an estimation of reliability of the program (which was programmed based on common sense) by measuring the reliability of software with the reliability estimation model.
The following explains the error-embedded model and reliability growth model, which are both reliability estimation models:

(1) Error-embedded model
In this model, errors are estimated from the error ratio after the test, by embedding errors into the program in advance.

(2) Reliability growth model
This model estimates program reliability based on the number of errors. This model is obtained by creating a model of how the number of errors subsequently decreases as tests are performed. For the measure of reliability characteristics, the number of errors and the time for testing are used. The following two models can be used for this purpose:
- Analytic model: Model that estimates reliability based on quantitative data
- Empirical model: Model that estimates reliability based on program complexities

Reliability growth curve

The reliability growth curve is a model that shows the cumulative number of errors that occur during testing as progress in a time series, in the form of an S-shaped curve.

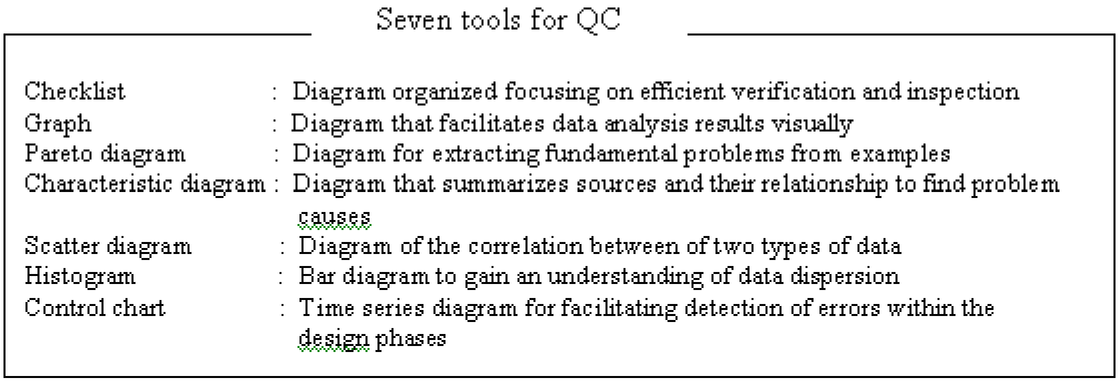


As the test time passes, the cumulative number of errors declines, which is an indication of increasing system stability.

6.3.3 QC method

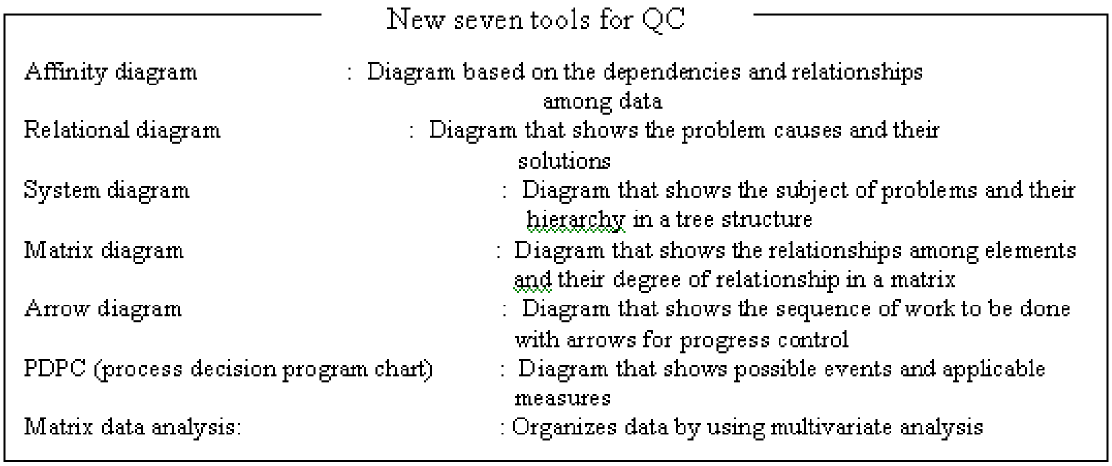
QC (quality control) refers to product quality control. The QC method is also adopted for quality control during software development.

As typical techniques used in the quality control of software, seven tools for QC and new seven tools for QC are used. The following shows the structure of the tools used in each type of technique:



Seven tools for QC

Checklist : Diagram organized focusing on efficient verification and inspection
Graph : Diagram that facilitates data analysis results visually
Pareto diagram : Diagram for extracting fundamental problems from examples
Characteristic diagram : Diagram that summarizes sources and their relationship to find problem causes
Scatter diagram : Diagram of the correlation between of two types of data
Histogram : Bar diagram to gain an understanding of data dispersion
Control chart : Time series diagram for facilitating detection of errors within the design phases



New seven tools for QC

Affinity diagram : Diagram based on the dependencies and relationships among data
Relational diagram : Diagram that shows the problem causes and their solutions
System diagram : Diagram that shows the subject of problems and their hierarchy in a tree structure
Matrix diagram : Diagram that shows the relationships among elements and their degree of relationship in a matrix
Arrow diagram : Diagram that shows the sequence of work to be done with arrows for progress control
PDPC (process decision program chart) : Diagram that shows possible events and applicable measures
Matrix data analysis: : Organizes data by using multivariate analysis

Exercises

1. Select from the solutions below the suitable terms to fill in the blank in the following statement, which explains the factors that make software quality control easy to achieve:
   The quality characteristics of software are (1), (2), (3), (4), (5), (6), (7), and (8).

Answers:
a. updatability
b. efficiency
c. operability
d. operationality
e. ease of testing
f. reliability
g. usability
h. changeability
i. portability
j. maintainability

2. Select from the solutions below the terms suitable to fill in the blank in the following statement, which explains the sub-characteristics of software quality:

- Usability is determined by such characteristics as (1), (2), and (3).
- Maintainability is determined by such characteristics as (4), (5), (6), and (7).
- Portability is determined by such characteristics as (8), (9), (10), and (11).
- Reliability is determined by such characteristics as (12), (13), and (14).
- Efficiency determined by such characteristics as (15) and (16).

Answers:
a. ease of learning
b. testability
c. changeability
d. recoverability
e. time efficiency
f. installability
g. replaceability
h. understandability
i. analyzability
j. maturity
k. environment adaptability
l. updatability
m. operationality
n. fault tolerance
o. conformance to standards
p. efficiency of resource usage
q. stability

3. From the choices below, select the correct explanation about the test cases for a block box test:
(1) During equivalence partitioning, both valid values and invalid values are entered with respect to the input specification conditions of the program.
(2) In boundary value analysis, only boundary values are applied with respect to the I/O specification conditions.
(3) In testing with a cause-effect graph, the input specification conditions and output results are represented in relational graphs, and a decision table is created from these

relational graphs.

(4) In the error estimation, test cases where errors are unlikely to occur are assumed for the purpose of testing.

4. The following description explains the procedure for creating a decision table based on the cause-effect graph. Arrange (1) to (6) in the correct order:

(1) Create a cause-effect graph.

(2) Assign a number for identifying the input data within the input specification conditions for the program and then extract that data.

(3) Create a decision table based on the cause-effect graph.

(4) Assign symbols to lines connecting the causes and effects to indicate their relationship.

(5) Assign a number for identifying the output data within the output specification conditions for the program, and then extract that data.

(6) Assign symbols to causes to indicate relationships between mutual causes and indicate restrictions among causes.

5. Select the incorrect explanation about the test technique for the white box text.

(1) Instruction coverage refers to testing each program instruction at least once.

(2) Decision condition coverage refers to a test case for testing the processing for each decision condition when the decision condition is at least once true and once false.

(3) Condition coverage refers to a test case in which the processing for all combinations of true and false in the decision conditions are tested.

(4) Decision condition and condition coverage refers to a test case in which test parts that cannot be checked by condition coverage alone are tested as a combination of decision condition coverage tests.

(5) Multi-condition coverage refers to a test case in which testing is performed by testing the branches of the operation flow for the case in which all combinations are set to true.

6. Select from the solutions below the name of the technique for testing module integration for each of the following explanations.

(1) In this test, all modules whose unit tests have been performed are combined into a single unit.

(2) In this test, all modules whose unit tests have not been performed are combined into single unit.

(3) In this test, all modules whose unit test has been performed are sequentially combined.

Answers:

a. incremental test

b. all-together test

c. dynamic test

d. big bang test

7. Select from the solutions below suitable terms to fill in the blank fields in the following statement about the addition test:

- The [***(1)***] is performed by combining low-level modules starting with the [***(2)***] sequentially.
- If a low-level module of the [***(1)***] has not been completed yet, the test is performed by replacing it with a [***(3)***].
- The [***(4)***] is performed by combining high-level modules starting with the [***(5)***] sequentially.
- If a high-level module of the [***(4)***] has not been completed yet, the test is performed by replacing it with a [***(6)***].
- The [***(7)***] is performed from both directions, starting at the same time from the [***(2)***] and [***(5)***].

Answers:

a. bottom-up test

b. highest level module

c. sandwich test

d. module

e. top-down test

f. driver

g. lowest level module

h. stub

8. Select suitable terms to fill in the blank fields in the following explanation about static tests.
- Static tests are performed without executing (1).
- Tools such as (2) and (3) are used in static tests.
- In the (2), the program hierarchy is analyzed by entering the source code of the (4).
- The [***(3)***] analyzes whether coding is as specified by the coding rules, there is any unnecessary (5), and (6) match between modules after entering the source code of the (4).

9. From the choices below, select the correct explanation about the dynamic test:
(1) A test coverage tool is a tool for checking what percentage of program instructions and decision conditions have already been executed.
(2) A test data creation tool is a tool for creating test data manually based on file description statements and data structures.
(3) A test bed tool is a tool for providing an operating environment for creating forms and drivers to perform the incremental test.
(4) The program validation tool is a tool for verifying the validity of test data.

10. Select a diagram for each of the following explanations of the Seven tools for QC from the group of solutions below:
(1) Bar diagram for gaining an understanding of data dispersion
(2) Diagram that summarizes sources and their relationship to find problem causes
(3) Diagram that facilitates visualizing analysis results of data
(4) Diagram for extracting fundamental problems from examples
(5) Diagram of the correlation between two types of data
(6) Diagram organized for efficient verification and inspection
(7) Time series diagram that facilitates detection of errors within the design phases
Answers:
a. graph
b. characteristic diagram
c. arrow diagram
d. checklist
e. Pareto diagram.
f. control chart
g. scatter diagram
h. relational diagram
i. histogram

11. Select a diagram for each of the following explanations about the new seven tools for QC:

(1) Diagram that shows the subject of problems and their hierarchy in a tree structure

(2) Diagram based on the dependencies and relationships among data

(3) Diagram that shows the relationships among elements and their degree of relationship in a matrix

(4) Diagram that organizes data by using multivariate analysis

(5) Diagram that shows problem causes and their solutions

(6) Diagram that shows the sequence of work to be done with arrows for progress control

(7) Diagram that shows possible events and applicable measures

Answers:

a. matrix diagram

b. affinity diagram

c. arrow diagram

d. relational diagram

e. system diagram

f. PDPC

g. scatter diagram

h. relational diagram

i. matrix data analysis

12. Select from the solutions below the relevant item for each explanation about a review method:

(1) Which review is a short-term review that consists of a preparation phase, meeting phase, and follow-up phase?

(2) Which review is for the purpose of detecting errors early so that errors do not affect subsequent design phases?

(3) Which review is for the purpose of validating the source code?

(4) Which review has the function of an overall examination?

Answers:

a. inspection

b. design review

c. code inspection

d. walkthrough

13. Fill the suitable terms to the blank fields in the following explanation about the reliability estimation method.

- The (1) is a model by which errors are estimated from the error ratio after the test by injecting errors into the program in advance.
- The (2) is a model by which program reliability is estimated based on numbers of errors.
- The (3) is a model by which reliability is estimated based on program complexity.
- The (4) is a model by which reliability is estimated based on quantitative data.

7 Software Development Environment

Chapter Objectives

This chapter explains the software development environment, which is considered as integration of software engineering. This chapter also explains the different tools used in software development and CASE.

7.1 Software Development Tools
7.2 CASE

7.1 Software Development Tools

Software development tools are tools that support the different kinds of work in software production.

Software development tools are developed mainly as stand-alone tools, and their purpose is to support the work in each process of software development.

Software development tools allow to switch from manual work during development and development with computers.

Software development tools can be roughly divided into those with common functions and those for individual processes.

7.1.1 Common functions

Common functions are functions that are necessary in any phase of the software life cycle.
The term software life cycle refers to any phase from the creation of software to its termination.
Examples of common functions include the following basic functions:

(1) Text editor functions
Text editors have functions of entering and editing characters, and for managing document files in text format.

For document editing, character editors and screen editors are used depending on the operation:
- Character editor

A character editor is an editor used for editing individual characters. The character editor can display an entire file as character strings, so that even special symbols, such as the line feed character, can be edited as characters.
- Screen editor

A screen editor is an editor used for editing whole screens. Because the editing status can be checked at any time by looking at the screen during use of the screen editor, this type of editor may be suitable as the user interface.

(2) Chart editor functions
Charts can show the abstract definitions of certain events by representing them schematically. Thus, charts are a suitable means of representing abstract objects, such as software. Their use of images is an excellent means for providing information in an easy and intuitive way.

A chart editor is an editor for editing individual charts. Such an editor is also called a chart editing tool. The chart editor provides functions for entering and editing text and graphics, and for formatting and checking charts. The windows management system may be incorporated in order to provide the optimum environment for these functions.

Different kinds of charts are used in the process of managing software development. Concretely, the following charts are used:

1) Analysis process
- HIPO (Hierarchy plus Input Process-Output)
- ER diagram
- Petri-net graph
- Decision table
- Decision tree
- DFD (Data Flow Diagram)

2) Design process
- JSP tree structure chart
- Module structure chart
- Flowcharts
- Warnier chart
- Structured charts (such as the NS chart, PAD chart, YAC chart, and HCP chart)

3) Development process
- Cause-effect graph

As a form of a further developed chart editing function, there is a tool that allows to automatically convert into a certain programming language after analyzing the content written in structured chart. As a result, structured charts can be directly executed as programs.

The chart editor is an essential software development tool, because definitions of specifications can be expressed with charts used in managing the different phases of software development.

(3) Functions for creating and editing documents
This function supports the creation and editing of software documents. It is also called documentation tool.

For the purpose of process management during the development phases of the software life cycle, every document is specified as the result of a single operation in a single process. That is, the type of document created in each process and its specifications are determined. The function for creating and editing documents is used to create and edit these types of documents.

The following documents are applicable:

1) Planning process
- Development plan
- Requirements definition

2) Analysis process
- Structuring specification

3) Design process
- Program design
- File design
- Database design
- Screen design

- Document design

4) Development process
- Program specifications

5) Inspection process
- Test specifications

6) Operation process
- Operation manual

7) Maintenance process
- Maintenance records

(4) Display functions
Windows is a typical example of a set of display functions. In Windows, multiple windows can be opened on the desktop as virtual work areas. Also pull-down menus and popup menus can be used, and data can be exchanged between windows.

GUI-based software development is possible by applying these functions in software development tools.

When computer systems were centered around mainframe computers, the terminals for software development were mainly CUI-based dedicated I/O terminals. Such terminals are suitable for editing source programs, but other than that, the preparation of design statements and specifications depended on work done manually on desks. With the widespread use of workstations and PCs, however, windows systems have acquired more and more capabilities for software development.

As a result, developing environments are changing from CUI-based to GUI-based, and work that used to be done manually is becoming to be done with the support of computers.

(5) Network system
In software development, cooperation among many people is often seen as necessary for the progress of work. Because there used to be no infrastructure to support such cooperation, developers used to get together in ordinary meetings for development work.

However, because of the expansion of the network environment for computers, it has become possible for people to work cooperatively from locations that are far away from one another. One kind of cooperative work in such a distributed environment is referred to as groupware.

Groupware is a system that supports cooperative work done in groups. The following are some of the functions of groupware:
- E-mail
- Electronic bulletin board
- Electronic conferencing
- Schedule management
- Workflow management

- Idea processor

The idea processor is incorporated as a function in text editor and word processor software, and it supports document creation by displaying outlines of entire documents.

- Hypermedia

Hypermedia is a function for providing or retrieving information. To do this, it manages different kinds of information, such as animation, pictures and voice data, in addition to characters.

Software development tools incorporating any of these functions can be used in software development.

7.1.2 Design process tools

(1) Design process tools

Design process tools are tools that support the processes of defining requirements and software design. Requirements definition systems are a concrete example of design process tools. These tools provide environments in which requirements definitions can be expressed as specifications that are in themselves formatted requirements definitions.

Concrete examples for design process tools are listed below:

1) SADT (structured analysis and design technique)

SADT is a tool for modeling a system to be developed, while conducting a structured analysis of that system. SADT is a product by SoftTech.

2) SREM (software requirement engineering methodology)

The requirement analysis and definition technique for real-time systems to be developed is called SREM. SREM is a series of methodologies supporting SRE. SREM is a product by TRW.

3) PSL/PSA (problem statements language/problem statement analysis)

The University of Michigan adopted this requirement definition tool in ISDOS (Information System Development and Optimization System). PSL is a format specification language that represents names and types precisely by specifying system functions as objects. The tool used to validate the completeness and integrity of contents defined in PSL is called PSA.

(2) Other support tools

There are other kinds of tools suited for individual functions and uses.

- Function analysis support tool

The function analysis support tool represents requirements definitions visually and displays them.

- Data analysis support tool

The data analysis support tool represents relationships among data.

- Event analysis tool

The event analysis tool represents relationships between the conditions under which a state changes and the resulting changed state.

- Function design support tool
The function design support tool provides descriptions of hierarchical control structure specifications when descriptions in data flow diagrams and data dictionary are not sufficient.

- Data design support tool
The data design support tool is used to organize data specifications in tables and represent data structures as diagrams.

- Prototyping tools
Prototyping tools can be divided into two categories: operational tools and functional tools. Operational tools support language description and interactive input, and they are mainly used in interactive software. Functional tools are used to create programs and for other purposes.

- Visual modeling tool
The visual modeling tool provides an environment where computers can be used to create and edit documents appended to UML diagrams displayed on a screen.

7.1.3 Development process tools

Development process tools are developed to support the software production process. These tools are related to program creation in one way or another.

Development process tools include the program editing tool and debugging tools.

(1) Program editing tool
The program editing tool supports creation and editing of programs in a specific programming language. The syntax and a simplified function of expressions in the target programming language are registered with the program editing tool in advance. Programs are created using registered items as required, with the assistance of the tool.

(2) Debugging tools
Debugging refers to the correction of program errors (bugs) in source programs. That is, a debugging tool is used to detect defects in source programs and correct them.

Debugging tools can be divided into two categories: dynamic debugging tools and static debugging tools.

1) Dynamic debugging tools
a. Tracer
- The tracer has the function of monitoring program operation over time.
b. Inspector
- The inspector creates a reference or updates variable values when execution of a program is halted.

2) Static debugging tool
The static debugging tool identifies errors in the source program structure and in the usage of names.
a. Pretty printer
Similar to the indentation function, the pretty printer formats program listings.

b. Cross reference
The cross reference tool creates mutual references between variables used in a program.

7.1.4 Test process tools

Test process tools support the software testing process. These tools are mainly used for program testing.

The program analysis tool is one example of a test process tool. By using this tool, tests can be carried out efficiently by analyzing program characteristics under specific viewpoints.

Test process tools can be divided into two categories depending on whether programs are executed during testing: dynamic analysis tools and static analysis tools.

(1) Dynamic analysis tools
Dynamic analysis tools perform tests by executing programs based on test cases and analyzing execution results.
- Test data creation tool
- Test coverage tool
- Test bed tool
- Program validation tool

(2) Static analysis tools
Static analysis tools perform tests without actually executing programs.
- Source code analysis tool
- Program structure analysis tool

7.1.5 Maintenance process tools

Maintenance process tools support the processes of managing and maintaining software. Maintenance means to modify software when a change is requested or a problem occurs after software delivery.

In the software life cycle for some programs, development was completed in one year, but maintenance lasted for seven years. In general, the required maintenance period can be expected to be five to ten times the development period. Maintenance takes up a large proportion of system development work, not only in terms of time but also in terms of costs.

The following maintenance process tools are available:

(1) Data dictionary system
The data dictionary system manages sets of information about data. If the data dictionary system is installed, information about the data handling by programs can be extracted at any time as necessary.

For example, to change the layout of a database, programs that contain the specified data can be found immediately by searching with a list.

(2) Version management tool
The version management tool manages the version number of programs as the program history. This tool can be positioned as part of configuration management.

Programs often have to be changed in the maintenance process. In such situations, multiple versions of an executable program may have been created during the development of a single program. This may lead to problems. To prevent such problems well in advance, a version management tool is necessary.

(3) Tool management tool

The tool management tool manages software development tools themselves in each process.

Each software development tool has different functions. By leaving management of such tools to this tool management tool, the functions of every software development tool can be used in the best possible way.

One of the problems of the tool management tool is that tools with different internal specifications may not be able to communicate with one another. Separate measures should be planned and taken against this problem.

7.2 CASE

In the past, most software development tools have been used as stand-alone programs. Little consideration had been given to the compatibility and relationships between software development tools. In terms of the software life cycle, the partial use of software development tools appeared to be not very cost-effective.

In recent years, however, the idea to support all phases of the software life cycle and management has been promoted. This approach would result in a software development environment that enables improvement in overall work efficiency because of the integration of different kinds of software development tools.

CASE (Computer-Aided Software Engineering) can be named would be one typical example of such a software development environment.

(1) Definition and components of CASE
CASE is an acronym for "computer-aided software engineering." This concept is also referred to as software automation.

Basic elements of CASE are as follows:

1) Workstation
A workstation is an input-output component for CASE operations. In CASE, information in different formats, such as text data and graphic data, can be entered for each process. The workstations has to provide sufficient performance so that this input-output data can be handled simply and correctly.

2) CASE workbench
The CASE workbench is CASE tool itself. It provides a variety of functions to support each process in software development.

- Graphic function
The graphic function is intended to support requirements definitions and can process requirements definitions as graphic representations.

- Analysis function
The analysis function is intended to support design and can analyze design contents.

- Generation function
The creation function is intended to support production and can generate source code automatically.
Production is supported by a combination of the generation function and test function, and support is provided from program generation to program testing.

- Test function
The test function is intended to support testing and can test programs.

- Maintenance function
The maintenance function is intended to support maintenance and can manage software system components.

- Management function

The management function is intended to support project management and can analyze process management and cost management based on data collected from the development and production processes.

7.2.1 Upper-CASE

The upper-CASE is a type of CASE that mainly supports the upstream software development processes. These processes are the analysis process, in which business processes are analyzed and requirements are defined, and the design process, in which basic design specifications are created. The upper-CASE has mainly the function of automatically creating specifications concerning the analysis and design of software.

As part of the analysis process, a data model of the current system is created, or a data model for a new system by including improvements made after the last model was created is created. Such models clarify the overall system and the positioning and relationships of data.

7.2.2 Lower-CASE

The lower-CASE is a type of CASE that mainly supports the downstream software development processes. These processes are the detailed design process, program generation process, test process, and operation and maintenance process. The lower-CASE has mainly the function of generating programs and test data.

In the detailed design process, design statements generated during the basic design process are defined in greater detail. The main features of lower-CASE can be said to be generation of the design statement itself and generation of the program source code from the design statement.

7.2.3 Component-CASE

The component-CASE is a CASE tool that supports only some specific processes in software development. The background of the creation of component-CASE seems to be that the manufacturers of CASE-based products only implemented functions for which they felt they had the most expertise.

To use component-CASE effectively and improve efficiency, the information structure to be stored in the repository (database that contains information about programs) must be standardized in advance. If the structure of the repository has been determined, component-CASE tools can be shared, which enables the user to use combinations of component-CASE tools.

7.2.4 Integrated-CASE

Unlike component-CASE, integrated-CASE supports the entire process of software development. With enhanced upper-CASE products, integrated-CASE can currently be used in most parts of system development over the software life cycle.

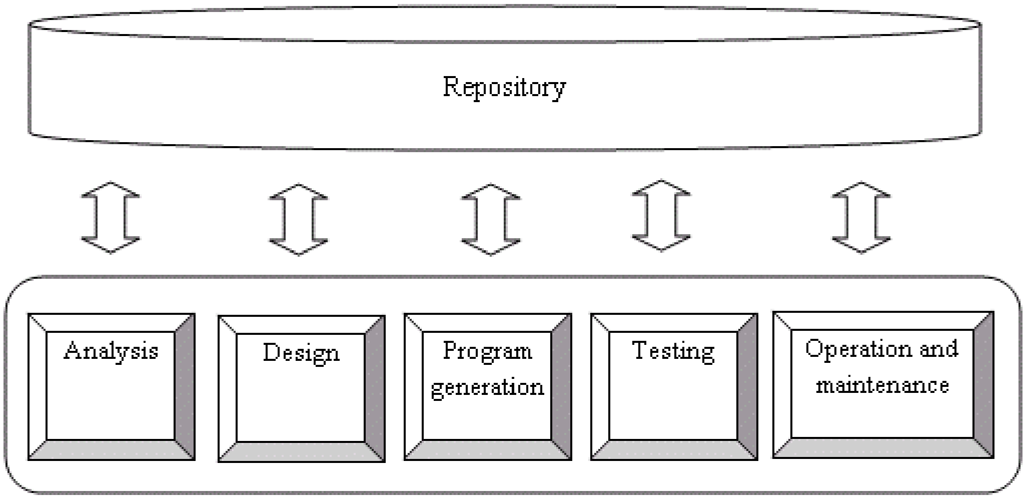
It has become possible to link upper-CASE and lower-CASE via a repository. If upper-CASE and lower-CASE have compatible repositories, they can be linked directly. A supplementary tool can be also used for linkage in some cases.

7.2.5 Repository

A repository is defined as a store or warehouse, and it provides different kinds of information generated during software development by standardizing and managing that information in a database.

A data dictionary system has been proposed for centralized management of databases. However, because different types of development support are possible by registering information about system development (not only data), repositories were invented.

Repository is generally used as a small database. However, while a “database” can be accessed directly by users, repository is “data warehouse for internal use by the system”.
Because a repository is used by multiple users and for a variety of job-related purposes, it should be of high quality and easy to operate.



7.2.6 Forward engineering

CASE supports forward engineering, reverse engineering, and re-engineering.
Forward engineering is a method where development proceeds from the upstream processes to the downstream processes in software development. That is, development proceeds according to the conventional waterfall model.

7.2.7 Reverse engineering

Reverse engineering clarifies the mechanism, specifications, purpose, components, element technologies, and other characteristics of software by disassembling and analyzing software. Analysis proceeds in the reverse direction, in other words, from the downstream processes to the upstream processes of software development.

The concrete procedure for reverse engineering is as follows:

(1) Derive the program specifications by conducting a static analysis based on the text of the source program.

(2) Based on the derived program specifications, conduct a backward-analysis to create design specifications and requirements definitions.

(3) With this approach, it becomes possible to check specifications during maintenance becomes feasible.

Every step of this procedure can be accomplished effectively by using information that is stored in the repository.

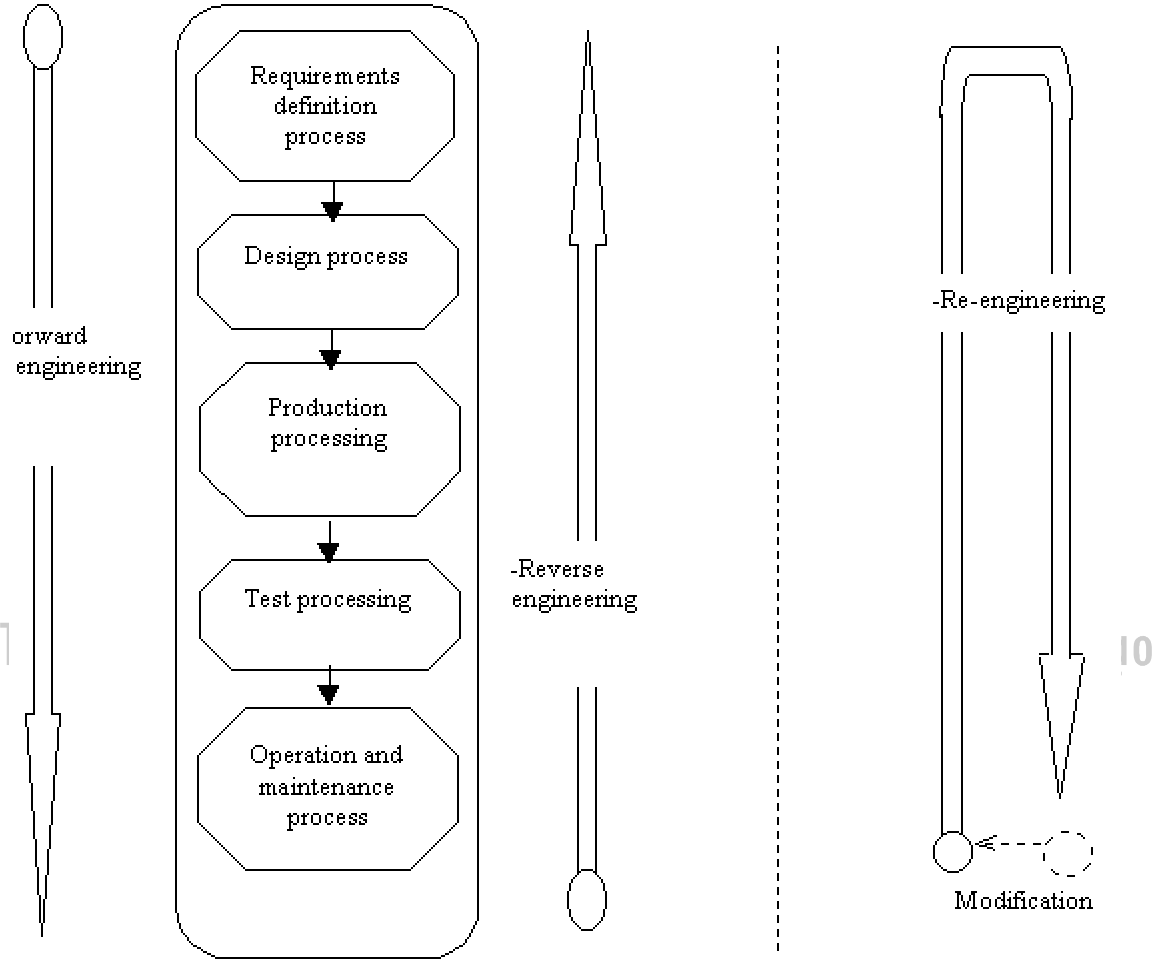
7.2.8 Re-engineering

Re-engineering is a method to construct a system that is similar to an existing one by partially modifying it, based on information obtained by reverse engineering.

Both forward engineering and reverse engineering are used in re-engineering.

The concrete procedure for re-engineering is as follows:

(1) Check the specifications by reverse engineering.
(2) Carry out forward engineering for partial changes of requirements definitions.
(3) Develop a software system based on other requirements specifications.



The reusability of software can be advanced rapidly by applying re-engineering.

In the past, reuse of software was limited to individual programs and modules, which resulted in low productivity and difficulties related to automating software development. However, with the progress in software development using re-engineering, automation of software modifications can be implemented as well.

Exercises

1. From the following group of choices, select the statement that correctly describes use of a tracer as a debugging tool.

Answers:

a. Outputs contents of magnetic tape files and magnetic disk files

b. Outputs the contents of memory when an error occurs during program execution

c. Function for monitoring program operation over time

d. Outputs the contents of a specified location in memory each time a specific program instruction is executed.

2. From the following group of choices, select the function provided by upper-CASE.

Answers:

a. Automatic generation of programs

b. Support of generation of test data

c. Management of sets of information about data

d. Requirement analysis of systems

3. From the following group of choices, select the CASE function that is used commonly in each process of software development.

Answers:

a. Repository

b. Matrix diagram

c. Reverse engineering

d. Inspector

e. Code creation

4. From the following group of choices, select the best description of the purpose of reverse engineering.

Answers:

a. Standardization of data names and data definitions throughout a system

b. Transformation of source programs into structured programs

c. Improvements of tasks by analyzing and reconstructing existing tasks

d. Clarification of mechanisms and specifications by disassembling and analyzing software

5. CASE tools can be classified according to the development process and scope in which they are applied. From the following group of choices, select the classification to which the automatic program creation function belongs.

Answers:
a. design process
b. downstream process
c. operation process
d. upstream process

6. The following statement is a description about CASE tools. From the group of choices listed below, select the best terms for (1) to (4) and complete the sentences.

(1) are provided as software for automating and assisting system development and maintenance work. (1) are classified for each process in system development.(2) supports the work of defining requirements and work of the analysis and design of applied tasks. (3) supports work in the production process and testing process. (4) supports work such as design and testing of whole processes in system development.

Answers:
a. The integrated-CASE
b. The upper-CASE
c. CASE tools
d. The lower-CASE

7. The following statement is about text editors among software development tools. From the group of choices listed below, select the text editor function that is described in the statement.

An entire file is displayed as a string so that special symbols, such as the line feed character, can also be edited as characters.

Answers:
a. character editor
b. line editor
c. screen editor
d. chart editor

8. From the following sentences, which are related to the common functions of software development tools. Select the one that describes the display function.

a. With the expansion of the network environment with computers, this function enables users in a distributed environment to work cooperatively.

b. This function is for entering and editing characters, and for managing document files in text format.

c. A typical example for this function is window systems. In window systems, multiple windows can be opened on the desktop as virtual work areas

d. This function supports the creation and editing of software documents, and is also called the documentation tool.

9. This question concerns groupware in a network system. From the following group of choices, select two items that cannot be realized by groupware functions.

Answers:

a. E-mail

b. Electronic bulletin board

c. Electronic conferencing

d. Workstation

e. SREM

f. Idea processor

g. Hypermedia

8 Trends in Software Engineering

Chapter Objectives

This chapter explains developments in software engineering up to the present, and future trends.

8.1 Software Tools

8.1 Software Tools

Since the 1990s, software engineering has undergone relatively rapid development for such reasons as proposal of a variety of new development techniques and introduction of software tools. This section explains the state of software engineering up to date and future trends.

8.1.1 Component-orientation

(1) Web computing

The rapid changes in computing have currently a significant affect on the operating environment for software. In particular, remarkable progress has been made in the field of Web computing. The term Web computing covers two types of computing: ubiquitous computing and pervasive computing. Both of these terms have the connotation "information can be acquired regardless of location".

1) Ubiquitous computing

The word "ubiquitous," which is of Latin origin, means "existing everywhere." Ubiquitous computing refers to an environment in which computers exist everywhere in the daily environment of people and work autonomously, but in cooperation with one another. For example, a computer can perform processing by automatically cooperating with other computers while referencing resources via the Internet.

2) Pervasive computing

The word "pervasive" means to penetrate. In a pervasive computing environment, people can acquire any kind of information at home, in offices, and elsewhere by using computers, televisions, and other end-user devices like personal digital assistants (PDAs), car-mounted terminals, and cellular phones.

Examples of pervasive computing include computing that is based on PalmOS and WindowsCE, operating systems for PDAs that are currently the focus of widespread attention throughout the world.

In particular, PalmOS accounts for a market share of more than 70% in the world market for PDAs.

3) Development of application service providers

Application service providers provide a system of conducting network-based business, thus eliminating the need for enterprises to invest in equipment.

One major reason why application service providers are drawing attention is the development of Internet technologies. The development of Internet technologies occurred mainly because of an increase in the communication speeds, an improvement in the stability of communication lines, and enhanced security.

The application service provider market, which is already growing at a significant pace, is expected to grow further, as high-speed connection lines become more widely available.

(2) Software engineering

In 1968, the Science Committee of the North Atlantic Treaty Organization (NATO) proposed using the term "software engineering." Since then, software engineering has evolved together with the development of software. The related categories are as follows:

1) Software requirements analysis engineering

Software requirements analysis engineering is an approach of developing software in which customer requirements are analyzed in accordance with certain specifications.

2) Object-oriented software engineering

Object-oriented software engineering is an approach of developing software in which a system is analyzed and designed based on an object-oriented paradigm.

3) Agent-oriented software engineering

An agent is defined as an "entity that recognizes a situation based on interaction with an external environment and autonomously proceeds problem solving for achieving a certain goal in cooperation with other entities." Agent-oriented software engineering is an approach of developing software based on this definition.

4) Component-based software engineering

Component-based software engineering is an approach of developing software by reusing software components to enhance development productivity and maintainability. This technique is aimed at achieving higher productivity and maintainability by accumulating more and more reusable components.

5) Software tools

Software tools are used to support a variety of tasks related to software production. Software tools are mainly developed as stand-alone tools. A software tool is aimed at providing the most appropriate tool for supporting individual processes in software development.

6) Software architecture

Software architecture refers to the structure of software. Software architecture covers such topics as software components and their relationships, the roles among these components, and applicable control structures.

7) Software estimation methods

Typical software estimation methods include the Function Point (FP) and the Constructive Cost Model (COCOM). FP, a method for estimating effort in development work, was developed in 1979 by A. Albrecht of IBM. FP focuses on the functionality of software and measures the amount of functionality based only on external software attributes.
In 1981, B. W. Boehm of TRW released COCOMO. COCOMO is a statistical model that arithmetically analyzes the relationship between programmer workloads and costs.

8) Pattern usage
Pattern usage appears to be one of the most effective uses of the results of software development in the future. Collecting common patterns can be expected to improve the quality and productivity of software development.
In the area of object-oriented software development, what is known as "design patterns" has recently become the focus of attention.

9) Software quality management
ISO/IEC 9126, an international guideline, reflects an international trend aimed at defining software quality management.
ISO/IEC 9126 is a guideline for evaluating software products with respect to quality characteristics and their applications. This guideline has an important role in current efforts to standardize software quality management.

10) Software process
Typical software process models include the Capability Maturity Model (CMM) and ISO 9000.CMM is a software process maturity model initially used by the U.S. government for procurement and now widely used throughout the world. ISO 9000 is a quality management system in which a series of processes in an organization is applied in a form of a system.

11) Software configuration management
Software configuration management is required to clearly manage the complex program configuration of a product. The use of software configuration management results in such advantages as providing information about products in the configuration definition phase and the creation of information required during program development.

12) Software metrics
Software metrics is an evaluation scale that represents the characteristics of software. Software metrics, which are used to clarify problems in the source code, include the number of lines, operators, and loops.

(3) Software project management
With software systems rapidly becoming larger and more complex in recent years, it is becoming extremely important to use software project management to manage processes, information, and software configuration. The essential topics of software project management are as follows:

1) Software quality
Software quality expresses the degree to which user requirements are met. In other words, the quality of software is evaluated by checking whether the software meets user requirements.

2) Cost estimation
In project management, parameters affecting costs are considered from the viewpoint of a cost model for software development. Currently, such techniques as FP and COCOMO are used for cost estimation.

3) Software risk management
Perform software risk management to identify and decrease risks that a project team faces. Recognition, evaluation, alleviation, and reporting are procedures used for software risk management.

4) Software reliability
Software reliability is generally defined as the "ability of software to maintain the targeted quality for a specified length of time under specified conditions." Insufficient software reliability leads to a deterioration of software quality.

(4) Real-time system
In real-time systems, any processing request is immediately processed. Real-time systems are used to manage responses under severe time constraints. Example for such systems include aircraft control systems, automobile engine control systems, and online banking systems.

(5) Programming and test
This section explains changes in programming languages and test techniques.

1) Programming style
To date, a variety of programming languages have been developed and used. Looking back at the historical background behind the development of programming languages, one can observe that the targets for which languages have been developed has shifted away from the intrinsic mechanism of computer operation and come closer and closer to the concept of human thought. This transition is reflected in the shift from such low-level languages as machine language and assembler language to such high-level languages as COBOL, FORTRAN, PL/I, ALGOL, and Pascal.

COBOL, FORTRAN, and PL/I are also called procedural languages. Procedural languages, which are the “orthodox” languages used to describe the processing sequence in a program, are still widely used in the business field.

Typical non-procedural languages include Smalltalk, C++, and Java. Non-procedural languages are used to describe what should be done, rather than describing the processing sequence in a program. Additionally, these three languages, which have obtained widespread attention, are classified as object-oriented languages, or languages with functions that allow implementing the concepts of object-orientation.

2) Software test techniques
In the current software development process, the test phase is said to require the greatest amount of effort. Thus, the productivity of software development as a whole largely depends on whether testing can be performed efficiently.
Generally used software test techniques include the white box test and the black box test.

- White box test
White box test refers to a technique of conducting a test, while keeping in mind the internal structure of a system. Test cases are designed based on the detailed logic specifications and the source program listing before the test is executed.

- Black box test
Black box test, contrary to the white box test, refers to a technique of conducting a test while considering only the relationship between input and output, instead of keeping in mind the internal structure of a system.

(6) Prospects of software engineering

It can be expected that software engineering will still be subject to a variety of technological innovations. For example, process programming will be put to practical use or a new software development environment, with leading-edge computer technologies fully adopted, will be provided. Process programming refers to the concept of considering the software development procedure as a process in itself and describing the process as a specification.

As is the case with object-orientation, agent-orientation is assumed to be a promising topic. Examination is currently underway on how to manage on computers an agent that can behave more intelligently than an object.

Also, in the field of support tools for software development, more technological innovations for standardization can be expected. Currently, consideration is being given to the development of integrated-CASE, which would include the functions of both upper-CASE and lower-CASE as well as the construction of an integrated environment based on the standardization of repositories and interfaces between tools.

As software development becomes more diversified, CASE itself is expected to become differentiated and diversified depending on the application field, the characteristics of the target system, etc. Engineers who use CASE will need to understand CASE better than before and have to be able to select a tool that matches the objective best.

8.1.2 JavaBeans

This section first explains the relationship between the World Wide Web (WWW) and Java, as well as the current situation related to these topics. An explanation of JavaBeans follows.

(1) Environment of the World Wide Web
The Web was developed by the Conseil European pour la Recherche Nucleaire (CERN) in Switzerland in 1989. This system was first developed to share information between research institutions within the same organization. Use of the Web gradually spread on the Internet, and then rapidly expanded throughout the world when Mosaic, a Web browser,

was released by the National Center of Supercomputing Applications (NCSA) of the University of Illinois. However, people pointed out the limitations of the Web as this system began to be used more widely and by more people.

Basically, servers perform all of the processing on the Web. A browser only presents information by, for example, displaying HTML documents and images, and merely conveys the input from a user to a server. In other words, the server performs all of the processing in response to the user's input. The related processing takes a significant amount of time. This poses no problems if the server is located close to the user within the network. In the case of the Internet, however, servers are distributed throughout the world.

One possible cause of the limitations of the Web is that "a browser can process only certain things." Thus, people began to realize that, if the processing capabilities of the browser could be enhanced, various problems related to using the Web could be solved.

One idea for overcoming the limitations of the Web is to execute a user-defined program on a browser as required. This small program to be run on a browser is called an applet. Under this arrangement, downloading an applet as required allows a browser to carry out different types of processes without the need of providing the browser with complex functions.

This idea of executing an applet on a browser solves therefore one of the problems related to using the Web. However, a number of obstacles and difficulties had to be overcome before it becomes possible to implement such a system that could be executed on a browser. Under the impression that no existing programming language was sufficient for applet programming, Sun Microsystems developed a new programming language called Java.

(2) History of Java
In 1990, a team led by James Gosling developed Java. The initial objective of development was to develop a simple and bug-free electronic product for consumers. The Java language first received attention as a language for creating applets on the Web in combination with HTML documents. However, not only did Java achieve this goal, it also enabled the development of applets that would function independent of which central processing unit (CPU) or operating system (OS) was used.

At first, many people saw Java as only "an applet development language." As it developed further, however, it became a more popular language, because it came to offer improved performance, better support for internationalization, and enhanced security. JavaBeans, which was subsequently introduced, provides the structure for a database interface, the RMI (Remote Method Invocation) mechanism, which can be used to realize distributed objects, and the standards for development tools.

(3) JavaBeans
A Bean is defined as a reusable software component that can be visually handled with a builder tool.

Conventional builders supported only specific languages and software groups. Conversely, the specifications of JavaBeans were designed to enable JavaBeans to support different builders. JavaBeans was based on the assumption that a program can be developed on a Graphical User Interface (GUI) builder. It may be said that its origin itself reflects a strong sense of priority given to responding to builders.

(4) Components of JavaBeans
JavaBeans substantially consist of a set of Java classes and required resources in which the interfaces between Beans and builders are standardized. In JavaBeans, three elements are conceptually required apart from the Java classes that directly comprise JavaBeans.

1) Method
A method describes a procedure to be executed.

2) Event
An event is a transfer of information between Beans.

3) Property
A property defines the nature and behavior of a specific Bean.

(5) Functions using Java
1) JavaEE
JavaEE, one of the functional sets written in Java, is a collection of functions required for an enterprise server that is to used as an job processor, for electronic commerce, or comparable purposes.

2) JavaME
JavaME, one of the functional sets written in Java, is a collection of functions required for embedded equipment such as home appliances, PDAs, and cellular phones.

3) Jini
Jini is a distributed system based on Java. A Jini system is intended to construct a general-purpose network that detects changes in connected hardware and software and enables connection to any type of computers and home appliances.

4) JTRON
JTRON is a specification of the next-generation real-time operating system (OS) that merges the execution environments of ITRON, a real-time OS, and Java. Thus, ITRON

OS, which has proved its suitability as real-time OS for information appliances, is

combined with Java, which provides superior portability. The introduction of JTRON allows using Java, which is widely used for user interface design and hardware control, on equipment for which the conventional ITRON OS was used.

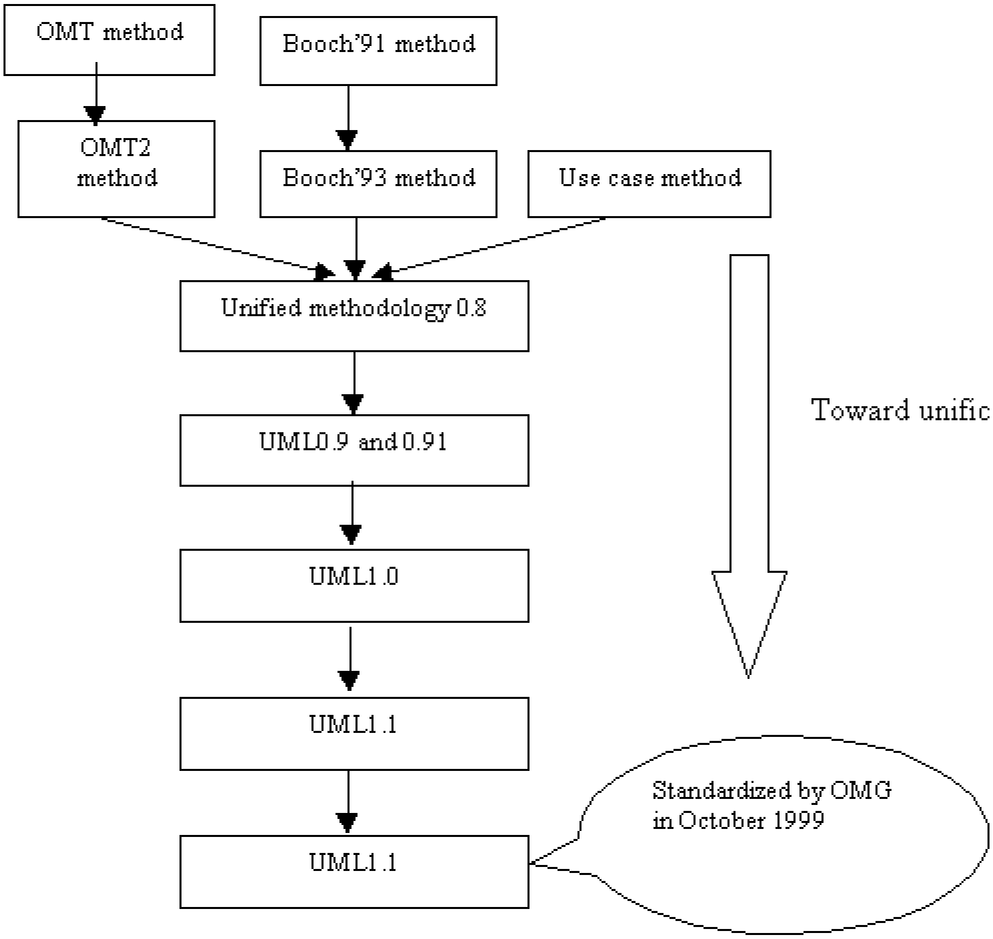
8.1.3 UML

(1) How UML was established

A software development methodology was proposed by G. Booch for use on Ada. Subsequently, the software development methodology was put together as the Booch method, a general design technique not restricted to Ada. Since the late 1980s, various methodologies related to software development have been proposed one after another.

In 1993, G. Booch promoted the establishment of standards for object-oriented analysis and design methods. In October 1994, G. Booch and J. Rumbaugh, an advocate of the Object Modeling Technique (OMT), began to work together toward a unified methodology that is a forerunner of the Unified Modeling Language (UML).

In November 1997, UML 1.1 was formally approved by the Object-oriented Management Group (OMG) as an object-oriented modeling language standard. Eventually in October 1999, UML 1.3 was formally approved.



(2) Future trends of UML

The future trends of UML include customization of modeling in specific fields, refinement of meta model definitions, and refinement of state chart definitions that represent system states. Currently, UML comes in two types: One for business modeling and another for development process definitions. UML is expected to support in the future various fields such as modeling of real-time systems.

Additionally, it is important to standardize development processes using UML, although it is not directly related to UML. A UML-based development process is not included in the UML specification. Therefore, even if it was decided that a development product is to be written in UML, differences can occur between the various phases of development. The most suitable development process depends on the scale of the development organization, field of development, and characteristics of products to be developed.

This is why there are already many general-purpose development processes that adopt UML as a language for describing a model.

8.1.4 COTS

Commercial Off The Shelf (COTS) refers to a commercial software part that can be used without changes.

This method of developing new software by combining existing software parts is a way to realize efficient software development. Because this method uses software parts whose reliability have been proven in actual use, it draws attention as a way to enhance reliability of software.

However, currently used software parts depend on the operating environment. If software parts are developed with a focus on a specific operating environment, some work must be done to port the software part to other operating environments. In such a case, a software part is no longer a part in the strict sense.

Exercise extra caution in designing software using COTS, because, while the use of COTS software is quite advantageous in terms of costs, expandability, and flexibility, it can be technically difficult.

Some notes on using COTS software are listed in the following.

<Notes on using COTS software>

- When crammed with too many functions, COTS software may lead to performance deterioration.
- Sufficient examination is required before deciding to adopt COTS software, because ascertaining the performance and reliability of COTS software is difficult and costly.
- In a real-time system that requires high reliability, such as the Automatic Train Control (ATC) system, a way must be found to retrieve monitoring and system analysis information from each of the system parts including COTS software.
- COTS software, having many features and being constantly expanded, cannot be easily evaluated or compared with other products.

From the above points of view, software development processes must be standardized and development techniques must be further improved before an efficient widespread use of COTS software becomes possible.

Exercises

1. The following paragraph is a description about pervasive computing. Fill in each blank by choosing a correct word from the list below.

The word "pervasive" has the meaning of (1). In the pervasive computing environment, people can acquire any kind of information at home, in offices, and elsewhere by using PCs, (2), and other (3) such as personal digital assistants (PDAs), car-mounted terminals, and (4) terminals.

Answers:
a. workstations
b. penetrating
c. devices
d. appropriated
e. network

2. Select from the following list a programming language that is not an object-oriented language.

Answers:
a. COBOL
b. C++
c. Java
d. FORTRAN
e. SmallTalk

3 The following paragraph is a description about a Java-based system. Which system is the paragraph about? Choose one from among the choices.

This is a distributed system based on Java. This system is intended to construct a general-purpose network that detects changes in connected hardware and software and enables connection to any type of computers and home appliances.

Answers:
a. JavaEE
b. JavaME
c. Jini
d. JTRON

Answers for No.3 Part 1 Exercises

Answers for Part 1 Chapter 1 (Overview of Software Engineering)

1. (1): c (2): b

2. (1): c (2): f

Answers for Part 1 Chapter 2 (Process Models and Cost Models for Software Development)

1: (1) b, (2) c, (3) d, and (4) a

2: (1) a, (2) c, (3) b

Answers for Part 1 Chapter 3 (Defining Software Requirements)

1. (3)

2. (2)

3. (2)

4. (1) system, (2) with the passage of time

5. (1) entities, (2) clustering

6. (1) d, (2) c, (3) b, (4) a

7. (1) system operation, (2) subsystems, (3) waterfall

8. (1) functional analysis, (2) function, (3) data flow diagram (DFD)

9. (2)

10. (4)

11. (1) b, (2) c, (3) f

12. (1) d, (2) a

13. (1) abstract data, (2) class

14. (1) process, (2) parallel

15. (1) b, (2) c, (3) a, (4) d

16. (1) d, (2) c, (3) a

17. (1) a, (2) c, (3) d

18. (1) X, (2) O, (3) X

19. (1) function, (2) performance

20. - financial constraints, - time constraints, - legal constraints, - technical constraints

21. (1)object-oriented life cycle, (2)structured method life cycle, (3)analysis, (4)design, (5)programming

Answers for Part 1 Chapter4 (Software Design)

1. (1) design, (2) designed, (3) developed, (4) integrating

2. (1) d, (2) b, (3) c

3. (3)

4. - Sequential (continuation), - Selection, - Repetition

5. (1) David Parnas, (2) components, (3) eliminated

6. (1) Larry Constantine, (2) processes, (3) structure of a module, (4) interface

7. (1) e, (2) g, (3) b, (4) a

8. c

9. (3)

10. The difference is that the Warnier method derives the program structure, focusing on the structure of input data.

11. b

12. (1) b, (2) a, (3) d, (4) e, (5) f

13. (1) c, (2) b, (3) a, (4) d

14. flowchart

15. - Process-oriented design method, - Data-oriented design method

16. (1) grouped, (2) procedures, (3) flowchart

Answers for Part 1 Chapter5 (Programming)

1. (1) c (2) a (3) g (4) d (5) e (6) b

2. (3)

3. (1) c (2) g (3) e (4) d (5) h (6) f (7) a (8) i

4. (2)

5 Mary

6. Two times

7. (1) b (2) e (3) d (4) f
Answers to (3) and (4) are interchangeable.

8. (1) atoms (2) lists (3) numeric atoms (4) literal atoms

9. (4)

10. (2)

11. (1) HOTSPOT (2) JIT compiler (3) native compiler

12. (1) f (2) h (3) c (4) g (5) a (6) i (7) d
Answers to (3) and (4) are interchangeable. Similarly, answers to (5) and (6) are interchangeable.

13. (1) class (2) instantiation (3) is-a (4) inheritance

14. (1) c (2) a (3) d (4) c (5) a (6) d

15. (1) reliability (2) maintainability (3) data (4) procedure (5) encapsulation

16. (1)

Answers for Part 1 Chapter6 (Software Quality)

1. (1) a, (2) b, (3) c, (4) e, (5) f, (6) g, (7) i, (8) j (The order of answers are not relevant)

2. (1) h, (2) a, (3) m, (4) i, (5) c, (6) q, (7) b, (8) k, (9) f, (10) o, (11) g, (12) j, (13) n, (14) d, (15) e, (16) p
(Answers of (1) to (3), (4) to (7), (8) to (11), (12) to (14), and (15) and (16) are interchangeable)

3. (3)

4. (2) - (5) - (1) - (4) - (6) - (3)

5. (5)

6. (1) d, (2) b, (3) a

7. (1) e, (2) b, (3) h, (4) a, (5) g, (6) f, (7) c

8. (1) programs, (2) program structure analyzer, (3) source code analyzer, (4) source program,

(5) statements, (6) interfaces

9. (1)

10. (1) i, (2) b, (3) a, (4) e, (5) g, (6) d, (7) f

11. (1) e, (2) b, (3) a, (4) i, (5) h, (6) c, (7) f

12. (1) d, (2) b, (3) c, (4) a

13. (1) error-embedded model, (2) reliability growth model, (3) empirical model,
(4) analytic model

Answers for Part 1 Chapter7 (Software Development Environment)

1. c

2. d

3. a

4. d

5. b

6. (1) c, (2) b, (3) d, (4) a

7. a

8. c

9. d, e

Answers for Part 1 Chapter8 (Trends in Software Engineering)

1. (1): b, (2): a, (3): c, (4): e

2. a, d

3. c

Part 2

EXTERNAL DESIGN

1 External Design Procedures

Chapter Objectives

This chapter explains the necessary preparations and procedure for external design.

1.1 Preparations for External Design Work
1.2 External Design Procedures

1.1 Preparations for External Design Work

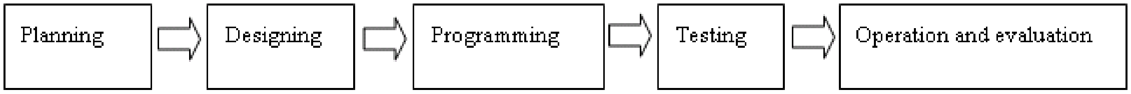
1.1.1 Policies for system development

As the enterprise environment constantly changes, the information systems have to change accordingly. The following situations may emerge due to the changes:

– As an enterprise becomes larger, the amount of data to be processed increases.
– A enterprise has to process data faster to improve its customer service.
– A enterprise requires a variety of information to make more appropriate decisions.

Although information systems have to satisfy such demands as above, the existing systems may not be adequate. For example, a system may not finish processing data within a prescribed amount of time or data output may be too slow, when the quantity of data exceeds its processing capacity. Such systems can not be considered as functioning properly. When an enterprise's existing system cannot keep up with the new business environment, the development of a new system should be considered.

When developing a new system, how to accommodate changes should be considered. Sometimes simply changing, adding, or deleting work procedures may be sufficient to deal with the changes of business environment; or using computers to automate business activities may be an effective measure. However, computerization is not always the perfect solution. Therefore appropriate measure should be chosen after careful examination of business activities. When developing a new system, consider how it can accommodate not only the current changes but also anticipated changes in the future. If enough attention is not paid to system development with regard to future operation, the life of the developed system may be very short service life.
For the development of a new system, use a suitable software life cycle model that is appropriate for the scope and scale of the project.
New information systems are generally developed in the five steps shown below. A specific plan must be made to determine the scope of the development project and to define the development process.



Planning: Find out what are the actual business activities in a enterprise , and analyze possible future requirements for the new information system. Then, made a plan for the new information system that can accommodate the changes of the enterprise 's environment.

Designing: Design the information system based on the above.

Programming: Design and create the programs according to the design.

Testing: Test the programs to check whether they work as intended.

Operation and evaluation: Migrate business information from the existing information system to the new system, and begin actual operations of the new system.

To prepare a development environment, select the applicable standards, methods, tools, and other items to use in the development process.

Adequate knowledge of development standards, development methods, software (i.e., tools, basic software, middleware, programming languages), and hardware is essential.

1.1.2 Systematization requirements definition

The systematization requirements definition is a document that specifies the purpose of systematization, the system functions, information requirements (e.g., flow of information), system configuration, cost-effectiveness of systemization, and systematization schedule, based on the system concept.

Because the external design is done according to the systematization requirements definition, the definition must be thoroughly understood before proceeding to external design.

Since the person in charge of systematization requirements definition and the designer of external design may not always be the same, systematization requirements definition must be clearly understood.

It is also important that the team members working on the external design reach the same level of understanding before the first step of their work.

Checkpoints:

– When you read the systematization requirements definition, can you imagine a concrete system based on the definition?

If any points are unclear, ask questions about them.

Try to find out the points for further consideration, if any, in the requirement specifications. Also check for possible omissions in the specifications and whether or not the system can be actually built.

– Can you represent the algorithm in good charts and/or tables?

Check important algorithms in the requirements to see if the information is clear by drawing flowcharts, data flow diagrams, decision tables, and other charts and/or tables representing such algorithms.

– Image the operation of the system, and make sure that there is nothing unrealistic.

Try to find other possible problems and points to keep in mind for the actual operation.

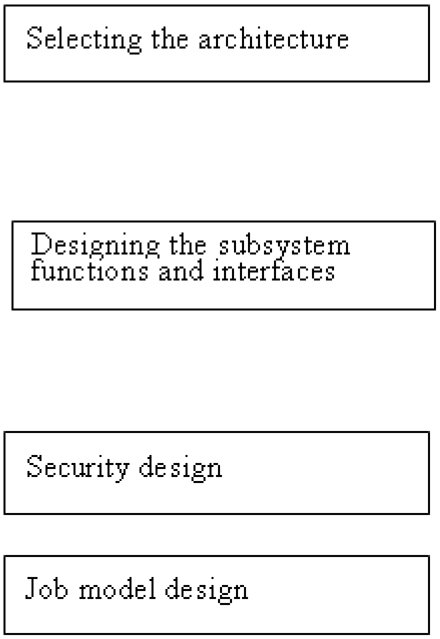
– What are the objectives of the system?

The functions described in the systematization requirements definition are generally the basis of the system design. It is important that the intended purposes of the functions are thoroughly understood, as well as their individual tasks and the specific operations for accomplishing the tasks. This is because the systematization requirements definition is merely an external specification, and the measures described in the specifications may not be necessarily matched with the specifications for the computers actually used. Consequently, the method to achieve those functions may have to be changed from those written in the specifications to something more suited for the applicable situation after further consideration of such factors as operating conditions, limitations of hardware and software, cost, and development time. If the objectives of the system are kept at the forefront of considerations while it is being designed, the purpose of any changes in the process of system development can be quickly understood, and the essential elements of the system are thus not lost.

1.2 External Design Procedures

1.2.1 Designing the system functions

Follow the steps shown in the figure to design the system functions. The principle is to divide the system into subsystems by functions and to develop interfaces to be used between the subsystems.


Selecting the architecture
Designing the subsystem functions and interfaces
Security design
Job model design

Hardware configuration,
software configuration, application packages, scope of systematization, candidate architectures

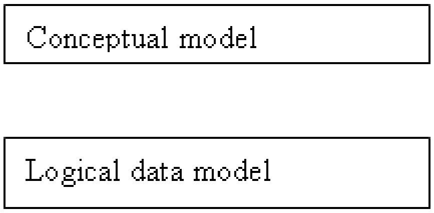
Division into subsystems,
defining the subsystem functional specifications, defining subsystem interfaces

Security policies, security requirements, system implementing security

Refining job flows and data flows

1.2.2 Designing the data models

The design of the data models includes the data used in the system.



ER model

Hierarchical model, network model, relational model

1.2.3 Producing the external design

The external design is documented as the basis of the internal design.

Creating a user manual (outline version)	Participants Review method Format of user manuals
Designing the system test specifications	Determining policies for system tests System tests environment and documentation technique
External design documents	System configuration diagram, subsystem relationship diagram, system job flows, system functional specifications, input-output specifications (screen specifications, report specifications), data specifications

1.2.4 Design review

The user manual, specifications for system tests, and the external design documents which are the result from the external design work are reviewed. By the reviews the products of this phase are evaluated in order to eliminate problems which may be carried forward to the next process.

2 System Function Design

Chapter Objectives

In this chapter, you learn how to design a job model based on the understanding of the relationships between job flow and subsystems, after choosing a system configuration and a hierarchical structure of the system. You also learn how to do the security design required for the system.

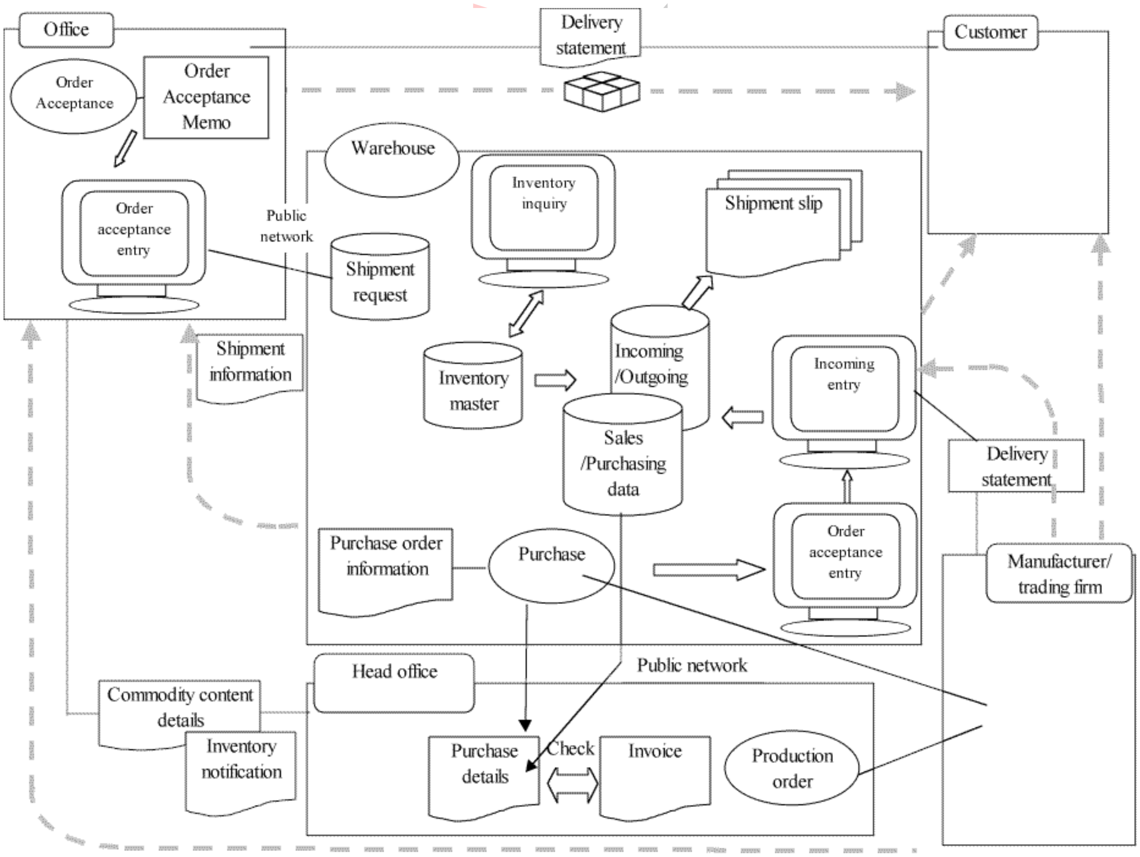
2.1 Selecting a System Architecture
2.2 Subsystem Function Specifications and Interface Design
2.3 Security Design
2.4 Job Model Design

2.1 Selecting a System Architecture

Determine what mode of processing is most suitable for realizing systematization requirements. To select a mode of processing, examine the following items conforming to the performance requirements, reliability requirements, and operating requirements explained later, and then create a diagram so that an overall image of the system can be understood.

Hardware	- Server equipment configuration (e.g., CPU and memory capacity) - Client equipment configuration - Network configuration
Software	- OS selection - DBMS selection - Communication middleware selection - Development tool selection - Use of packages
Security	- Study of security level
Use of existing assets	- Study of use of existing assets (e.g., hardware and databases)

[Overview of a new system]



2.1.1 Hardware configuration

Select a hardware configuration according to the requirements specifications. Select the optimum hardware after carefully considering whether all hardware should be new or some existing hardware equipment can be used.

It is also necessary to determine whether the hardware system should have a multiplexing configuration to enhance system reliability, or whether a fault-tolerant hardware configuration is required, so that the system operates as a whole correctly even if a failure occurs somewhere in the system.

2.1.2 Software configuration

Select a software configuration as specified in the requirements specification. Similar to the process for hardware, select the optimum software after carefully considering whether all software should be new or existing software can be used.

(1)Basic software (such as an operating system)

Basic software, such as an operating system, can either be acquired from a different vendor, or, as for mainframes, both the operating system and the hardware may come from the same vendor.

Be careful when selecting the operating system, because the selected operating system may not be suitable for a specific system development/operation due to the functions provided by the operating system.

(2) Middleware

Middleware is software that runs on an operating system and is positioned between the operating system and applications to provide more concrete, advanced functions to individual applications.

Since many of the functions provided by the operating system are only basic functions necessary for all applications, functions necessary for implementing advanced, concrete functions are usually provided by middleware.

Since middleware accommodates differences between the operating system and hardware, it has the advantage of facilitating the development of multi-platform applications.

The examples of middleware are DBMS, the TP monitor which provides transaction processing functions, and ORB which provides a distributed object environment. Since the selection of middleware has a large effect on system development and operation, it is important to select suitable middleware conforming to system requirements.

2.1.3 Application packages

If an application package can be used to implement a system that satisfies the requirements specifications, decide whether to use the package or to develop a system from a scratch, depending on the budget, technical level, and development period.

(1)Advantages of using an application package
1) Systems can be developed efficiently (shortening the development period).
2) Systems can be implemented at a lower cost.

(2) Disadvantages of using an application package
1) Systems may not be flexible enough.
2) Some functions may be restricted.
3) The total construction of the system to meet the requirements is not possible, which may lead to complexity in the operation management.

2.1.4 Scope of systematization

It is essential to clarify the scope of systematization and the parts to be left for manual work. Insufficient systematization may lead to more complex operation. The work steps described below can make the systematization scope clearer.
Note that each work step is not to be done independently but to be done mutually related, and these work steps done together lead to a step-by-step refinement of the system.

(1) Function clarification
While keeping cost-effectiveness and development period in mind, clarify which system functions are needed based on the system requirements definitions.

(2) Design of the human interface
Since this part of the system is used by users, examine its operability, frequency of use, and proficiency of the users.

(3) Code design
Code should be designed considering its scope of use and period of its use in order to minimize changes to the code.

(4) Logical design of files
Check whether any items are missing, groups of items are appropriately specified, and make sure that item attributes and number of digits match requirements.

(5) Migration design
Check the migration schedule, management of resources involved in migration, migration procedures, and impact on currently running systems when migrating.

(6) Operation design
Design the system from the standpoints of operation organization, jobs, processing mode, scope of use, and the purpose of the information system.

(7) Creation of an external design document
Since an external design document is an input document for successive internal design processes, create this document in such a way that it can be easily understood in the successive processes.

(8) Review and approval
Review the external design to find any problems at an earlier stage, and then make improvements as necessary. List the functions that are required or can be implemented,

and obtain the customer's approval.

2.1.5 Candidate architectures

(1) Hardware
Examine the hardware equipment configuration from the standpoint of processing capacity and reliability to ensure that it is robust enough for actual operation.

1) Examination of CPU capacity
Estimate the processing time for batch jobs and online responses to check whether system requirements are satisfied.

2) Estimation of required memory
Re-examine potential problems with system memory after determining whether a multiplexed configuration is necessary and estimating needed memory size. Examine and review whether the communication control devices have enough memory even when online traffic increases or data length changes.

3) Examination of the number of channels
For large-scale online databases, check the number of jobs that simultaneously use the magnetic disk devices and magnetic tape devices, and estimate the entire load on the system by taking the number of accesses and data volume into account. Then, determine the number of channels and total capacity to be prepared.

4) Examination of the type and number of magnetic disk devices
Calculate the required number of magnetic disk devices according to the capacity of the resident files required for the whole system and the work area capacity for the jobs running simultaneously. Then, choose the type of magnetic disk devices by considering cost, installation space, and capacity.

5) Examination of the type and number of external storage media
To ensure a stable processing time for backup and recovery, devices of high density and high operating speed must be selected. It is also necessary to note the tape density, if data is to be delivered to other departments or companies.

6) Examination of the type and number of printers
In conjunction with increases in the volume of input/output data, examine whether it is necessary to increase the number of printers or change to high-speed I/O devices.

7) Check of the specifications and control methods for special devices
Clarify the specifications and control methods for special devices. Adequate checks of the connection method and transmission control architecture must be carried out.

8) Examination of I/O devices of other manufacturers
If the equipment configuration includes I/O devices of other manufacturers, the procedure for isolating faults and the scope of responsibility must be well defined.

(2) Software configuration
Examine the software functions required for job operation and machine operation. Make sure to obtain software at the appropriate time.

1) Basic software
Examine the essential components, such as those for automation (of the operator's tasks)and methods of accessing terminals and data sets.

2) Software related to online networks
Examine the software required for cluster functions, file transfers to terminals, and operations on unattended terminals.

3) Language processing software
Select a suitable language for system development.

4) Migration software
Decide whether to incorporate programs, packages, and utilities of other operating systems, manufacturers, and software providers.

(3) Network configuration
Examine the network configuration based on behavioral analysis of terminal users.

1) Examination of placement of terminals
Examine the placement with regard to the sites where data is entered, sections that require output reports, the number of operators, and the installation environment.

2) Examination of the model and number of terminals
Examine the model and number of terminals with regard to line traffic intensity and whether the operator is full-time assigned or not.

3) Examination of the cluster model and the number
Select a cluster model by considering a unification of lists of each cluster and an increase in the number of terminals. Also consider the unattended and automated operation of terminals and clusters as well as disaster-prevention facilities.

4) Examination of the line type
Select an appropriate line type from lease lines, telephone lines, packet switched networks, and high-speed digital networks based on data length, line traffic, and terminal locations.

5) Examination of connection speed
Review the connection speed by calculating the processing capacity based on data length and traffic intensity.

6) Examination of the communication method
Considering the operation mode,etc., decide whether to use a full duplex or half duplex system and whether to use polling/selecting system or contention mode.

7) Determination of data transmission senders and receivers
Decide on the initiators of data transmission between the host, clusters and terminals, the scope of their responsibilities, and the method for data protection.

8) Examination of network
Analyze the line type and the placement of the terminals, clusters, subhosts, and host in order to build a suitable network.

2.2 Subsystem Functional Specifications and Interface Design

2.2.1 Subsystem division

(1)Information from an analysis process

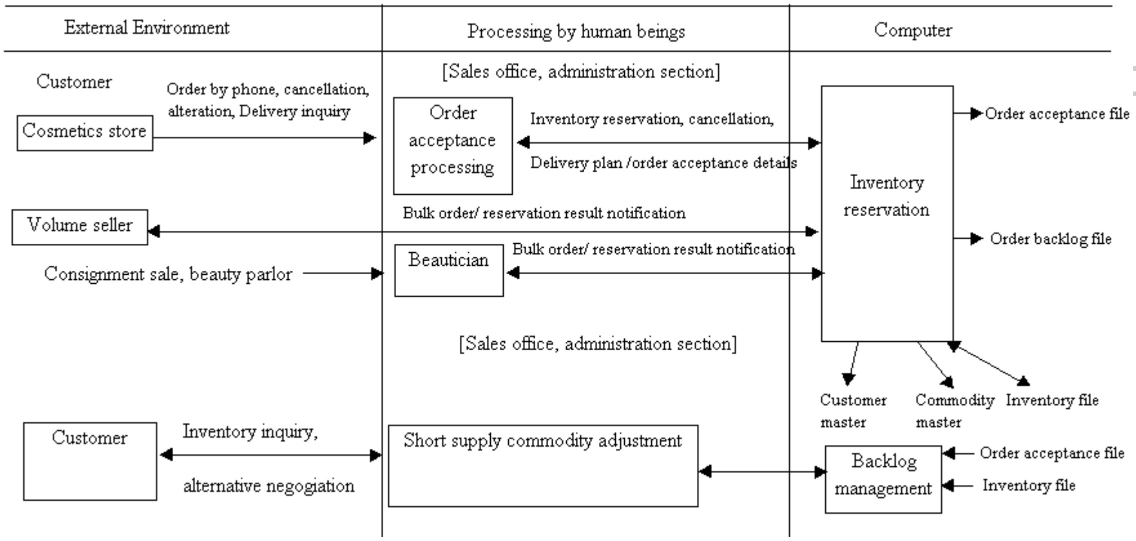
In an analysis process, examine how to create a new job system by analyzing the job requirements for achieving objectives. Summarize the results in a job flow diagram and systematization functional diagram.

As shown below, the job flow diagram shows the relationship between the job flow and computer processing. The systematization functional diagram provides an overview of each processing function, showing what kinds of computer processing are implemented for every job process.

These two categories of information constitute the basic information for specifying system functions.

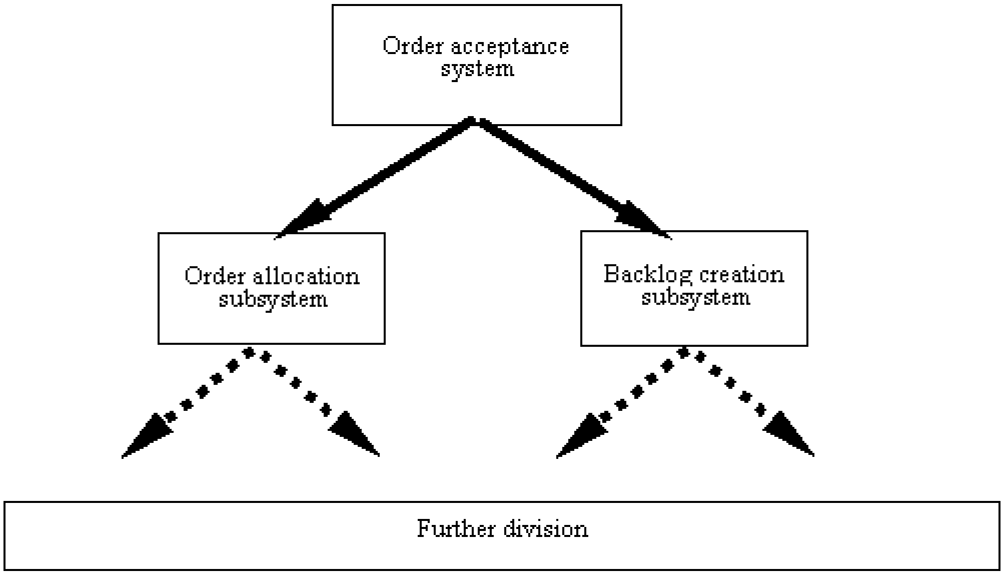
[Example of job flow]

Order acceptance and returned goods control


External Environment
Processing by human beings
Computer
[Sales office, administration section]
Customer
Order by phone, cancellation, alteration, Delivery inquiry
Cosmetics store
Order acceptance processing
Inventory reservation, cancellation,
Delivery plan /order acceptance details
Inventory reservation
Order acceptance file
Order backlog file
Volume seller
Bulk order/ reservation result notification
Beautician
Bulk order/ reservation result notification
Consignment sale, beauty parlor
[Sales office, administration section]
Customer master
Commodity master
Inventory file
Customer
Inventory inquiry,
alternative negogiation
Short supply commodity adjustment
Backlog management
Order acceptance file
Inventory file

(2) Function of division into subsystems by analyzing the job flow
The main task in system design is to divide a system into subsystems, which are blocks of processing that can be combined.
Following the requirements specification, divide a system function unit into basic function units of subsystems by stepwise layering.
Depending on the system scale, a subsystem may be divided even further into lower subsystems in a hierarchy.

[Example of subsystem partitioning]



2.2.2 Defining subsystem functional specifications

In observations of the relationship between job processing and computer processing, each computer processing function can be found to correspond to a part of job processing by people. In the previous example, the task of order acceptance processing done by people corresponds to the order allocation in the computer system. Accordingly, divide job processing into individual processing functions to be executed on the computer system.

The purpose of partitioning into subsystems is to determine the outlines of functions to be provided by each subsystem and the input/output functions. The outline of the exception handling function must also be determined.

(1) Examination of the exceptional functions

Screen out the processing functions that rarely occur and decide whether to incorporate them into the computer system or to handle them manually.

(2) Design of system functions

After further dividing the functions to be computerized, screen out the corresponding units of subsystems in the systematized job flow diagram. Then, by sharing, dividing, and/or integrating common functions, sort out and relate the corresponding subsystem units in the system.

(3) Decision on a common processing method

Some technical issues may arise for the system functions designed in (2), and they must be resolved in order for the functions to work in actual operation. These issues include satisfaction of performance requirements, standardization of operation and operability, and error check methods to ensure reliability. A common standard processing method should be defined beforehand for such technical problems. This method is called common processing method design.

(4) Subsystem design

Based on (1) through (3) above, clarify the interface between subsystems, revise the subsystem units, and summarize the overall flow in a system flow diagram.

(5) Decisions on subsystem functions

Define the processing function in sufficient detail in each program, so that the purpose of each subsystem in the system flow can be understood from the corresponding function in the program.

(6) Extraction of parts creation targets

To reduce the development scale, extract as parts from subsystems those portions that can be shared during development, those portions that are likely to be changed, and those portions which can be performed by reusing an existing application(s).

(7) Re-examination of feasibility

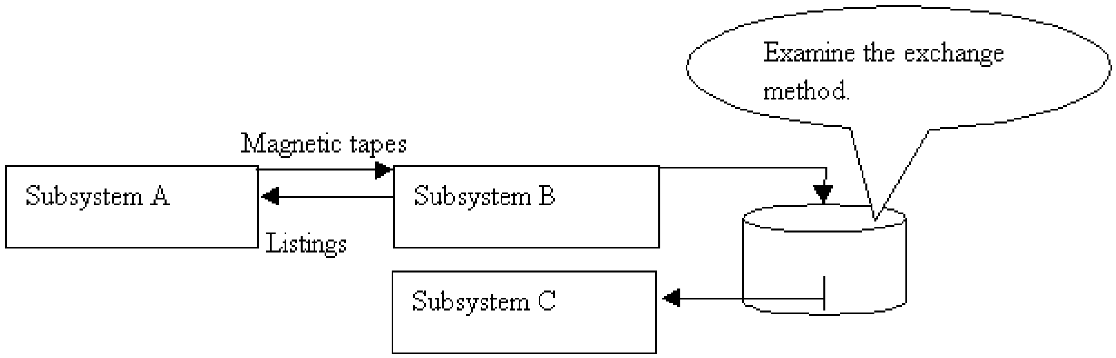
Performing the work steps mentioned above can provide greater detail for the implementation method of the system than that which is provided by system analysis. This allows the examination the feasibility of the system again.

2.2.3 Subsystem interface definition

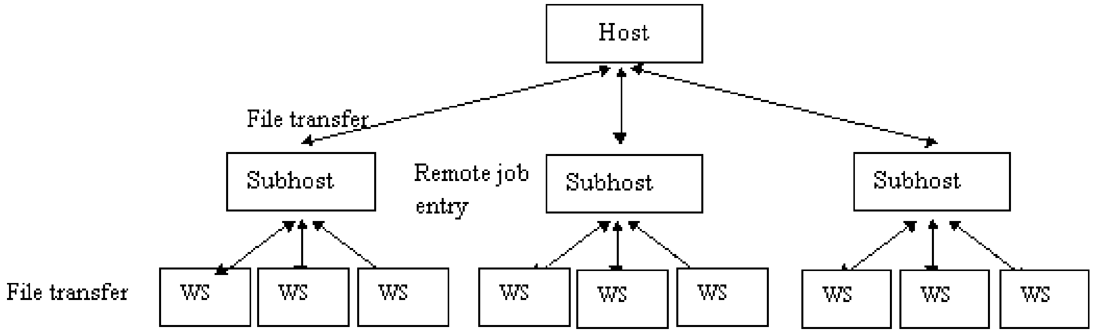
To exchange data between subsystems, the optimum interface configuration must be

found by examining the data.
The resulting configuration should be reflected in the file design and database design.

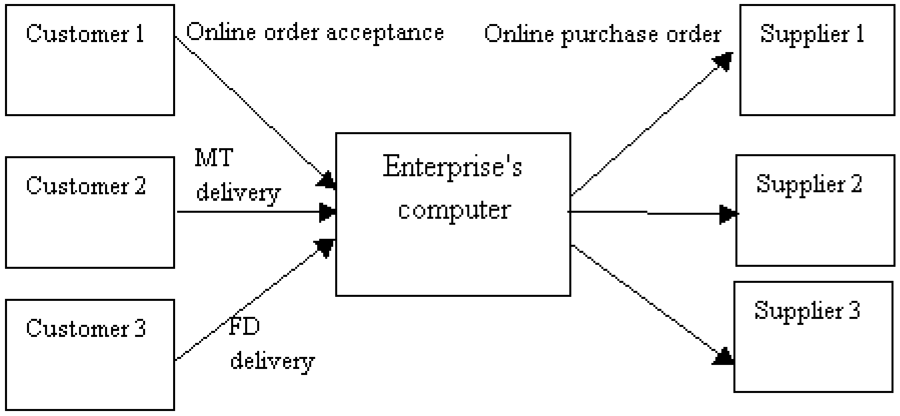
(1)Linkage to other job systems and subsystems
- Examine the method of exchanging data between systems.



(2) Linkage between a client and a server in a distributed system
- Clarify the type of data to be transferred and the data flow.
- Examine the transfer mode (e.g., batch transfer/real time and file transfer/remote batch).
- Examine the operation mode (e.g., operating time and actions to be taken when an error occurs).



(3) Data exchange between companies
- Clarify the type of data to be exchanged and the data flow.
- Clarify the storage media used (e.g., MT, FPD, and via lines) and protocol.
- Examine the transfer mode (e.g., batch transfer/real time and file transfer/remote batch)
- Examine the operation mode (such as operating time and actions to be taken when an error occurs).



Check such interfaces as shown, and summarize them in a subsystem relationship diagram.

2.3 Designing Security

System security means assuring system security by taking necessary measures. Generally, the security measures for an information system cover computers, communications equipment, and communications networks. They also cover the information stored, processed, searched, and transferred with the equipment, their operations, method of use, maintenance programs, specifications, and procedures. Losses of information system resources can be caused by a variety of events. These include natural disasters, such as earthquakes and fire, system failures, such as those caused by computer failures and communication line failures, operating errors, such as mishandled operation and data input errors, and damage by computer crimes, such as data deletion or falsification by unauthorized access to information systems.

To prevent such damage, companies should define security policy and design a system to implement security functions by following the defined security policy.

2.3.1 Security policy

The security policy is a basic policy regarding a enterprise 's information security. It includes the standard for security measures and individual concrete implementation procedures. It specifies who is allowed to access certain information, who is allowed to perform certain operations, and which data is to be encrypted.

Define policies to prevent the use of information for other than its intended purposes, unauthorized access, and disclosure of confidential information. In some cases, the policy includes the measures to be taken in the event that the system stops because of a problem, data loss, and data or system damage because of infection of a computer virus.

(1) Formulating the security policy

Formulate the security policy by considering the following points:

1) Balance of network security, server security, and application security
2) Balance among job processing systems
3) Balance between risks and measures
4) Balance between measures and non-operability

(2) Main points of security policy

The following items must be decided when defining the security policy:

1)What information is to be offered and to whom?
2) What information and contents to be provided are authorized?
3) How can the system determine the users and sites that are allowed to access the pages and data provided?
4) Has the person responsible for overall security been determined?
5) How can organizational measures be implemented in the event that unauthorized access takes place?
6) How can security tests and evaluations be carried out and checked?

2.3.2 Security requirements

Examine the measures for protection of information systems based on the security requirements in the requirements definitions. Security is generally not a serious concern in a closed environment, such as enterprise LAN, but they can become very serious in a network environment, such as the Internet.
In particular, the following requirements must be examined:

(1) Safety measures
1) Method of checking users and function of the check
When a user ID is entered, the system compares the entered user ID with the user ID stored in the system to authenticate the user ID.

2) Access control and its function
Carry out access control in three stages: identification -> authentication -> permission.

3) Encryption function
A. Secret encryption method
In this method, the encryption key and decryption key are the same. Information is kept confidential as both senders and receivers have the same key.
B. Public encryption method
In this method, the encryption key is made public, but the decryption key is kept secret. A sender sends information after encrypting it with the public encryption key. The receiver decrypts the text by using the decryption key.

4) Monitoring function
This function monitors the system usage and collects execution data by using software and hardware.

5) Function for checking the communication party
This function is used to make sure that the sender is the correct party. Telephone callback is a typical example.

6) Unauthorized modification prevention function
This function identifies whether a person is authorized for modification.

(2) Reliability measures
1) Measures for hardware failure
2) Function of monitoring and controlling load status
3) Function for detection and isolation of failure locations when a failure occurs
4) Function for degradation and reconstruction when a failure occurs
5) Checkpoint/restart function
6) Alternate function, backup/recovery function, and journal use function when an error occurs to an important file
7) Fault detection function of multiple systems and alternate function when a failure occurs

(3) System maintainability
1) Consistency
There is no contradiction among specifications, documents, and programs; and symbols, terms, and notations are standardized.

2) Traceability
The procedures of development processes, which covers from requirements specification to design, programming, and testing, work records, test data, and review minutes are created and stored for reference.

3) Self-explanatory descriptions
The purpose, functions, conditions, input/output, and algorithms of coding are explained in comments in the source code.

4) Structure
There is high independency among the program modules, which are strongly coupled with specific data, so that modifications and corrections can be localized.

5) Simplicity
Program descriptions are simple and easy to read. The source code itself works as a reference document containing useful and necessary information.

6) Extensibility
The system and program have adaptability for increases of memory size and data volume and against additions and extensions of processing functions.

2.3.3 Security implementation method

To manage information, it is necessary to identify information to be protected and to specify how to maintain, manage, and use the important information identified. An important task is to distinguish confidential information from other information that can be made public.
To ensure the security of an information processing system, access to the system must be restricted to only the persons who are allowed to use the system.

(1) Password setting
One of the most widely-used methods for identifying individual users is to require a password. Two types of password are available: one for identification and one for information access.

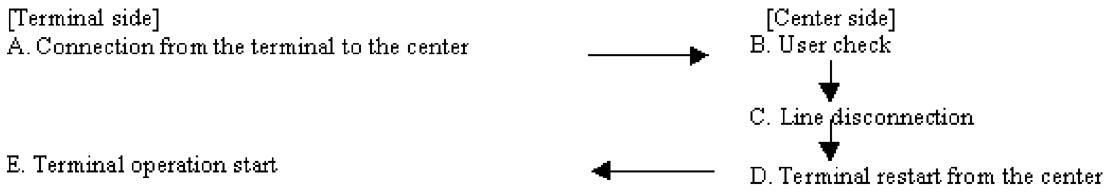
1)Password for identification
This kind of password is entered together with a user ID by a user attempting to log on to a system.
The purpose of this password identification is to confirm that the person who entered the user ID is actually the person to whom access right has been granted.

2) Generating a password for identification
A password for identification may be generated randomly by the system or specified by the user.
If the system generates the password randomly, the password is a combination of completely meaningless alphanumeric characters and is thus hard to remember. If the user specifies the password, the password is often easier to remember but also easier to guess.

3) Password for information access
This kind of password is individually defined to protect a specific file or specific data set. This password must be entered into the system when accessing a specific file or

specific data set after logon with the password for identification.

(2) Setting of the remote party check function
The cost of conducting strict checks of remote parties is high. To avoid such high costs, a method such as callback, in which a connection to a terminal is temporarily cut and then re-established by the center, can be used instead.


[Terminal side]
A. Connection from the terminal to the center
[Center side]
B. User check
C. Line disconnection
D. Terminal restart from the center
E. Terminal operation start

(3) Monitoring function for illegal access
The target data includes "business programs/business data" and "system data (e.g., user data and password data)".
Unauthorized use, falsification, and damage can be assumed for each occurrence of unauthorized access.
As technical countermeasures, the use of system standard functions such as password checks and a method of checking together with ID cards can be considered.
In regard to operation, the establishment of a management operation standard, selection of a password administrator, and accumulation and analysis of system operation records are necessary along with these technical countermeasures.
In addition, installation of a line encryption device to encrypt and decrypt data efficiently is desirable for online systems in order to prevent wiretapping and tampering with line data.

(4) Detection function for invalid data
Messages and response sequences may not be delivered or may be incorrectly delivered because of transmission line noise, attenuation distortion, phase distortion, and instantaneous power supply interruptions.
A variety of techniques for detecting invalid data are available, depending on the type of line used and the transmission procedure (e.g., BSC procedure, FTS procedure, and HDLC procedure), and are generally provided as standard system functions.
During network design, it is more important to design the processing after detection of invalid data than its detection. General notes related to this topic are given below.

1) If lines are used, choose the transmission line and transmission procedure after checking the contents and operation method for the job. Also balance the cost of using the lines.

2) System processing after detecting for invalid data must be determined during system design.

[Example]

- Retry count --- If the re-execution count exceeds this retry count, the system disconnects the line.
- Setting of non-response time --- If the non-response time from a remote party exceeds a predetermined value, the system disconnects the line.

- Re-execution method --- If a line error is detected during batch transfer, decide whether to retransmit data from the beginning or continue transmission from the portion in which the error occurred.

2.4 Job Model Design

2.4.1 Details of job flow

(1) Concept of job flow

After understanding the current job data contents and the related input/output operations, draw the job flow in a job flow diagram for a review and to check for problems.

The job flow is important supplementary material for the written descriptions in documents. The job flow diagram can help to determine the current state of a job, which is important for an analyzer.

Descriptive materials	Graphical materials
1. Not appropriate for an overview 2. Appropriate for displaying details about a specific part 3. Mutual relationship between parts of the work cannot be seen easily 4. The overall image of the job flow cannot be easily understood	1. Appropriate for an overview 2. When viewing details about a specific part, the level of understanding may vary depending on the experience of the users. 3. Mutual relationship between parts of the work can be easily identified. 4. Appropriate for understanding the overall image of the job flow

In addition to movements of physical objects, such as reports, in the job flow diagram, summarize the whole range of job-related activities, including work by people.

The purpose of creating a job flow diagram is to summarize all activities that are related to the business and the relationships among these activities. Thus, evaluate business-related activities by dividing them into the following four levels:

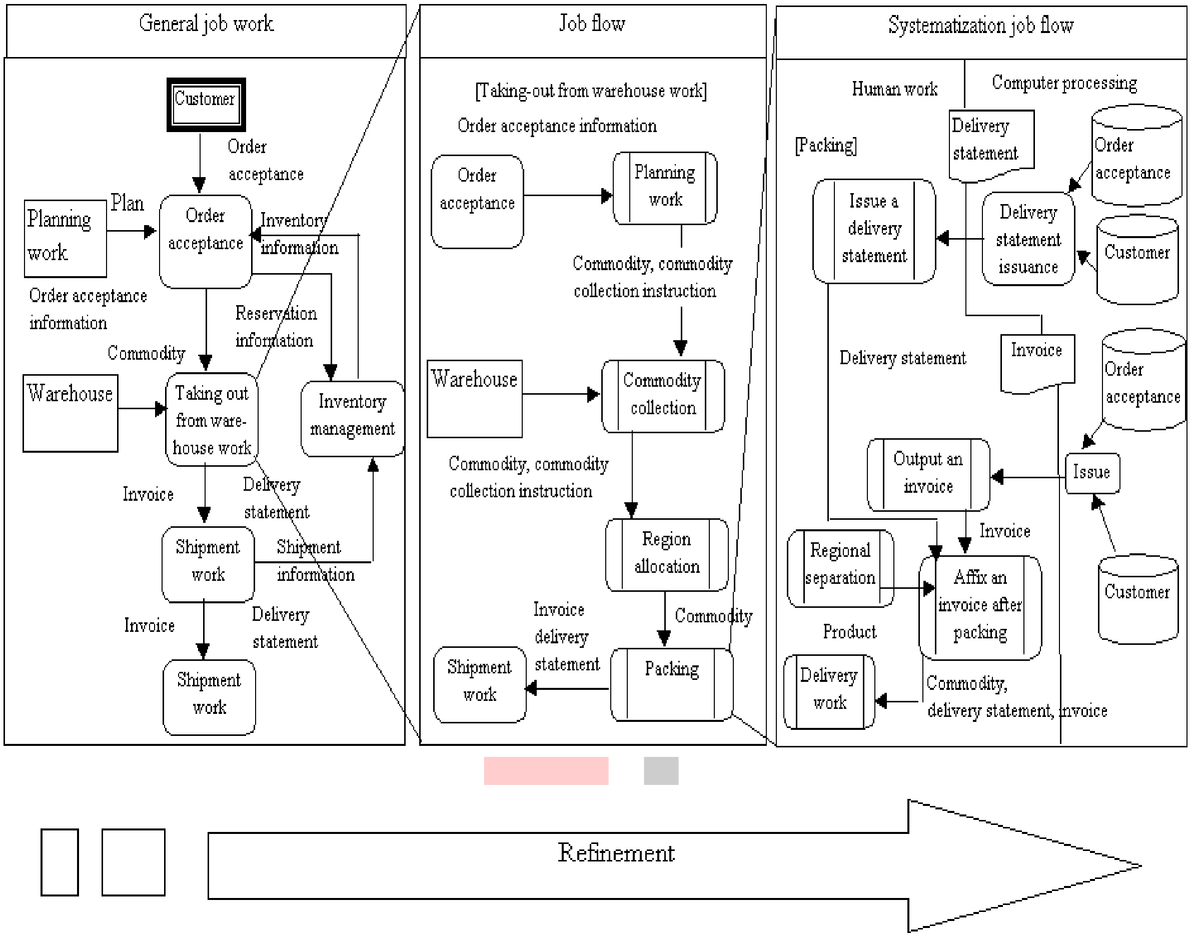
- Job domain: Basic functional unit as viewed from the standpoint of the whole enterprise (Examples) Finance, production, sales

- Job function: Functional unit which belongs to a job domain and has a single objective (Examples) Estimation, order acceptance, inventory control

- Job process: Functions according to accomplishing an objective in the job process (Examples) Production arrangement, shipment

- Work: Unit of a job process in which a person follows a specific procedure or handles a specific resource
  (Examples) Entry in an inventory book, extraction of customer information from a terminal

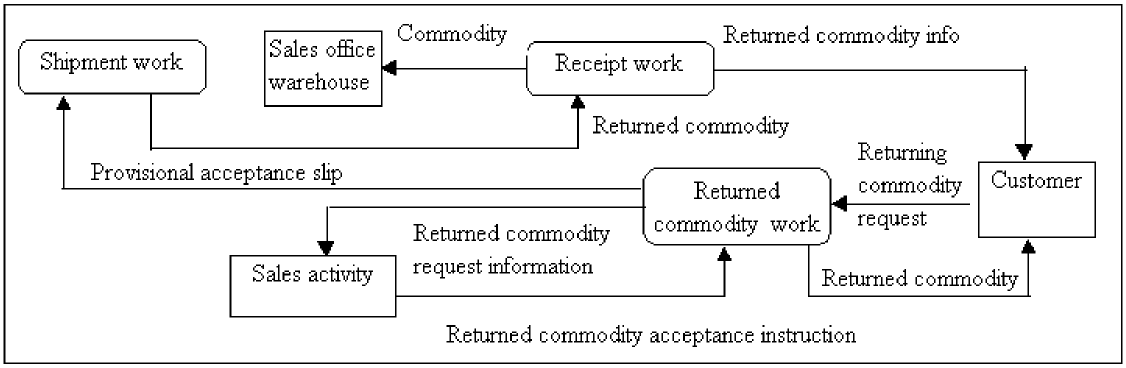
Based on the division of business activities into the four levels, draw a job flow diagram in three layers.



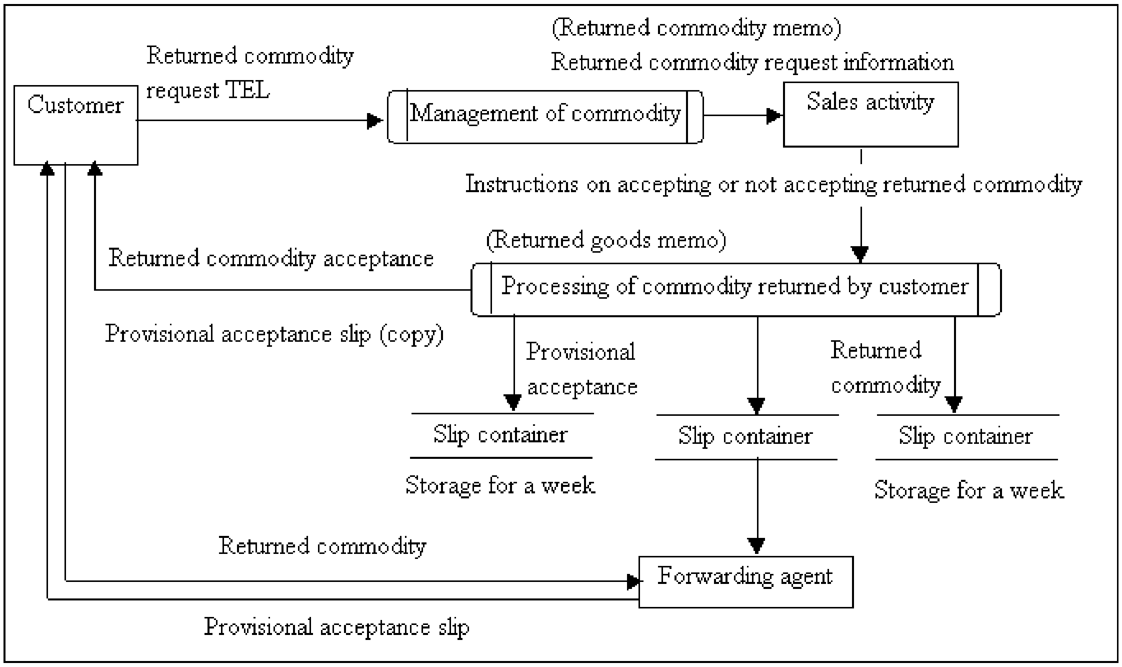
The first layer (general job flow diagram) of the job flow diagram represents the relationships among business areas to be analyzed at the level of business functions. The job flow represents details about each business function in the general job flow diagram. These documents clarify the scope of analysis, target business functions and units.

The next page shows examples of the general job flow and job flow.

[General job flow] (example)



[Job flow] (example)



(Returned commodity memo)
Returned commodity request information
Returned commodity request TEL
Customer
Management of commodity
Sales activity
Instructions on accepting or not accepting returned commodity
(Returned goods memo)
Returned commodity acceptance
Processing of commodity returned by customer
Provisional acceptance slip (copy)
Provisional acceptance
Returned commodity
Slip container
Slip container
Slip container
Storage for a week
Storage for a week
Returned commodity
Forwarding agent
Provisional acceptance slip

[Checkpoint]

- For what purpose do you do this?

  The roles and purposes of slips, reports, and works must be sufficiently obtained from the flow creator or persons in charge of the job by having interviews.

  Ask them, "For what purpose do you do this?" and "What are the results of doing this?" If this question-and-answer process is neglected, you may not be able to distinguish what is really necessary from what is useless, resulting in an incomplete system.
- Are current practices reflected in the job flow? If the job flow diagram is not current, check whether the current business practices are reflected, because the business contents data may have changed.
- Identify the main jobs and exceptional jobs in the job flow.

  The procedures for the major works, such as inventory updates, order acceptance checks, sales bookings, returned goods processing, money receipt, and purchase order processing, are the core of the job. The main flow of job is the part to which the most attention must be paid when carrying out system design. In addition, exceptional tasks must be performed sometimes, and it must therefore be examined whether to computerize those exceptional jobs. However, neglecting the system design of the main jobs while focusing on the exceptional jobs could result in serious consequences

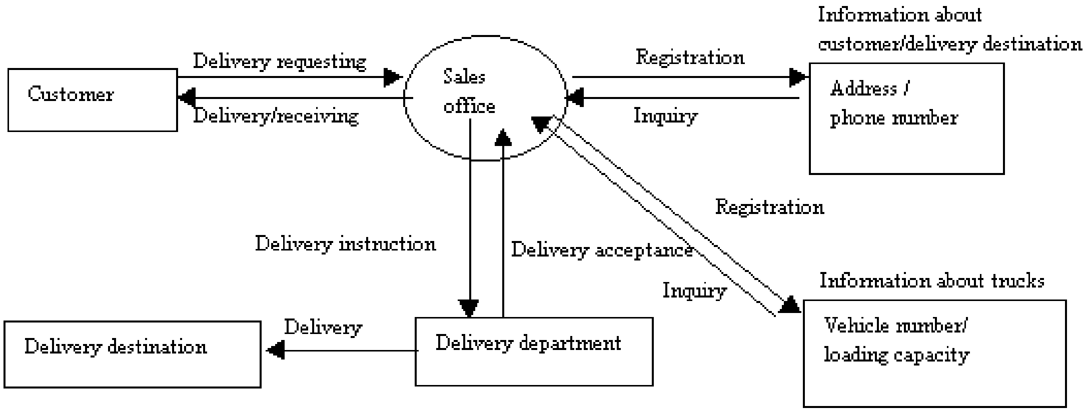
2.4.2 Data flow refinement

After understanding the details of the current jobs, conduct a data analysis of current conditions. In the previous section, we extracted things and information from jobs and analyzed their flow in the jobs within the target scope of the analysis.

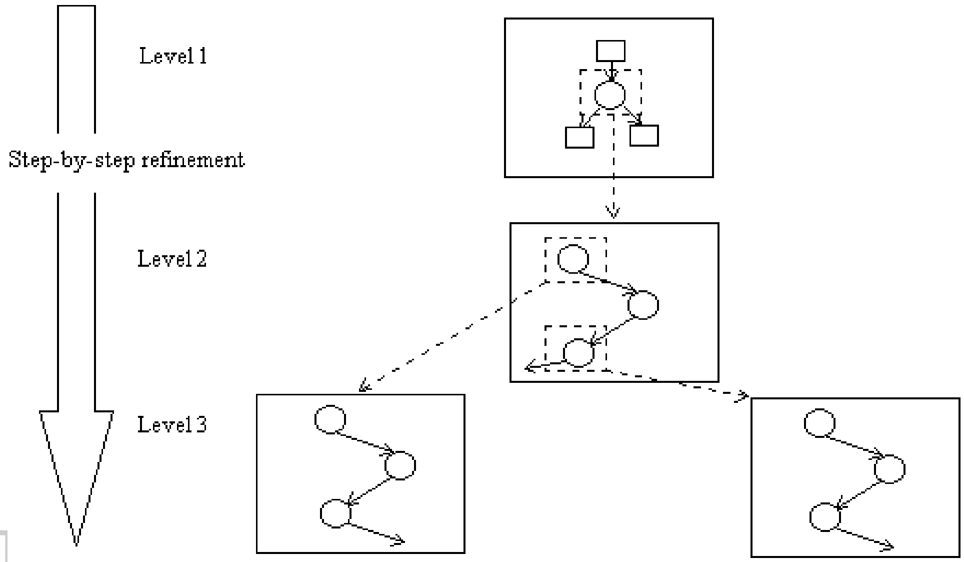
In this section, we analyze the "data" that concretely represents these "things and information." When you analyze and clarify the business contents to make them clear, the units for information processing become clear as well.

For example, there is a job called "delivery work" in the figure. The sales office carries out the delivery work using trucks. This work requires information about the trucks (e.g., the vehicle number and loading capacity). It is more reasonable to handle all information about a truck, such as its vehicle number and loading capacity, in a set "information about trucks" rather than as individual, independent items of information. Information about the customer is also required for the delivery work, and in this example, it is more reasonable to handle all information about a customer, such as the customer's name, address, and phone number, as a set of information.

Data analysis thereby clarifies things and information in the job flow and the units for processing these things and information.



A Data Flow Diagram (DFD) is a typical technique that is currently used for this purpose. First draw a rough DFD, and then repeat division of the data flow into finer parts. The DFD is complete when further division is not possible.



Exercises

1. The following sentences are descriptions of documents created in system analysis process. From the answers listed below, select the correct document for each description.

(1) Summary of code types and the code system used in the job

(2) A summary of job functions to be improved and effect of such improvements, required information, and problems that may arise because of the improvements. This summary is based on the results of requirements analysis.

(3) Job flow represented in a diagram. It is used to create a new job model based on job improvement proposals.

(4) The functions that a computer performs according to document (3) are detailed in 3 to 10.

(5) Relationships of data items represented in diagrams, for use in a data-oriented approach

Answers

a. Job code investigation table
b. Systematization job flow diagram
c. Conceptual data model
d. Outline of systematization functions
e. Job improvement proposal
f. Component relationship diagram
g. Input/output investigation table

2. Select a good approach to creating a current logical model based on the current physical model in the system analysis phase.

(1) Integrate duplicated functions.

(2) Improve the shortcomings of the current system to reflect the requirements specifications.

(3) Describe concrete slop names.

(4) Describe intermediate files and backup files.

3.The following sentences are descriptions of system design. Complete them by selecting appropriate terms for (1) to (4) from the answers listed below.

Items of system design can be roughly divided into (1) and (2).

(1) is a consideration from the logical aspect of how to implement the required functions in the computer system. This consists of (3) and data structure design. For efficient and stable operation of a computer system in job operations, (2) of the computer system is necessary. (2) includes the design in terms of (4), performance, and security.

Answers:
a. System function design
b. Operational design
c. Functional design
d. Measures against failures

4. Place the following processes in the correct sequence within the procedure for system functions specification.
a. Subsystem design
b. Examination of exception functions
c. Extraction of potential objects for parts creation
d. Re-examination of feasibility
e. System function design

5.From the following answers, select the appropriate description of monitoring, which is one of the performance evaluation methods for computer systems:
Answers
a. Obtaining data for improving the system configuration and response performance by measuring the execution status of each program and usage of resources.

b. Collecting performance data of all the components of the system shown in their catalogs and calculating the overall system performance based on it.
c. Measuring the overall processing performance of the system including input/output devices and the control program by running a typical program.
d. Calculating the average execution speed of all instructions by calculating a weighted mean value using the frequency of use as a weight for each classified instructions group.

6. Select the correct definition of a fault-tolerant system from the following answers:
Answers

a. System that maintains operation of the required functions for the system as a whole even if part of the system fails
b. System that has a standby system at a remote location in preparation for occurrence of regional disasters

c. System in which resources are shared by multiple processors connected in a network
d. System in which a single transaction is carried out in parallel by multiple processors configured and processing results are checked against one another

7. For each of the following sentences, write O if the sentence describes an item that must be considered when a user specifies a password. Otherwise, write X.

(1) Specifying the user's name, phone number, or birthday as the password may lead to problems, but it is almost impossible to guess the password if such items are combined.

(2) Words and phrases found in an English dictionary should not be used as a password.

(3) Once a password is specified, it should not be changed, because it is risky to change the password periodically.

(4) Users should not reveal their password to anyone, even to their supervisor or system administrator.

8. For each of the following sentences, write O if the sentence is correct regarding the review of a network configuration. Otherwise, write X.

(1) Review the model and number of terminals with regard to source of input data, installation environment, and line costs.

(2) Review the line type and connection speed based on the processing capacity to handle data length and line traffic.

9. For each of the following sentences, write O if it describes an advantage of using application packages is correct. Otherwise, write X.

(1) The system development period can be shortened.

(2) Implementable functions can be constructed as desired.

(3) Operation management is easy.

(4) Systems can be implemented at lower costs.

10. The following sentences explain the review of a hardware configuration. Select a corresponding item for each description from the following answers.

(1) Review of the resident file capacity of the entire system and the work area capacity for running jobs simultaneously

(2) Review based on usage for backup and recovery, and on delivery of data to other departments or companies.

(3) Review based on response requirements for batch jobs and online jobs.

(4) Review of the necessity of additional devices and high-speed processing devices based on variations of the amount of input/output data.

Answers

a. CPU performance

b. Model and number of external media

c. Model and number of magnetic disk devices

d. Model and number of printers

11. You want to access an in-house network from outside the enterprise .From the following descriptions of methods to enhance security, select the incorrect description.

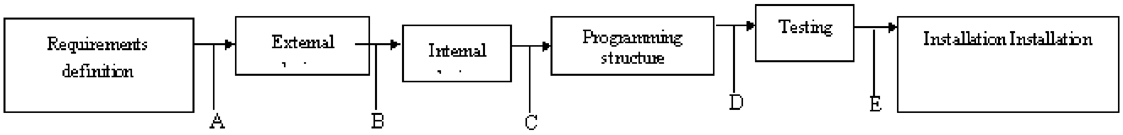
(1) Create a password by combining two or more English words that it is hard to guess but less likely to forget.

(2) When using a network from outside, use a password that can be used only once.

(3) To prevent unauthorized access, do not reveal to the public the telephone number used for network connections.

(4) When using a network from your home, use function to callback the phone number specified in advance for your user ID.

12. The following figure shows processes of system development. Select the process in which a system plan is created for the purpose of clarifying the system scale and schedules.



13. From the following sentences, select the best description of data analysis for constructing a database that can be fully utilized for job processing by positioning data as information resources of a enterprise .

(1) The amount of data to be analyzed should be limited at an earlier stage so that the range of review does not diverge.

(2) A data model created after analysis should not be modified.

(3) In regard to efficiency, the method of physical data storage should have the greatest emphasis.

(4) The manager of a department and persons in charge of a job should be involved in reviews at an early stage of analysis.

14. From the following sentences, select the best description of consistency in system maintainability.

(1) The procedures of development processes, starting with the requirements specification to design, programming, and testing, procedures, work records, test data, and review minutes are created and stored for reference.

(2) Each program module is relatively independent and strongly coupled with specific data, so that modifications and corrections due to changes can be localized.

(3) There is no contradiction among specifications, documents, and programs; and symbols, terms, and notations are standardized.

(4) Program descriptions are simple and easy to read. The source code itself works as a reference document containing useful and necessary information.

3 Data Model Design

Chapter Objectives

This chapter explains how to design data models that conform to specifications for interfaces between subsystems and for subsystem I/O data.

3.1 Conceptual Models
3.2 Logical Data Models

3.1 Conceptual Models

3.1.1 ER model

A conceptual model represents applications in a data model based on data and the relationships among data irrespective of file and database characteristics.

A typical example is the ER model, where E means Entity and R means Relationship.

The ER model consists of the following elements:

- Entities (management objects, such as customers and commodities)
- Relationships (relationships with another entity)
- Attributes (characteristics of an entity, such as the commodity code, commodity name, and unit price of a commodity)

During system development, the jobs for which information systems are to be developed are analyzed to find problems and their causes. This information is then used to construct a new system for solving the problems.

Two approaches can be followed for this work. One is a component-oriented approach, and the other is a data-oriented approach.

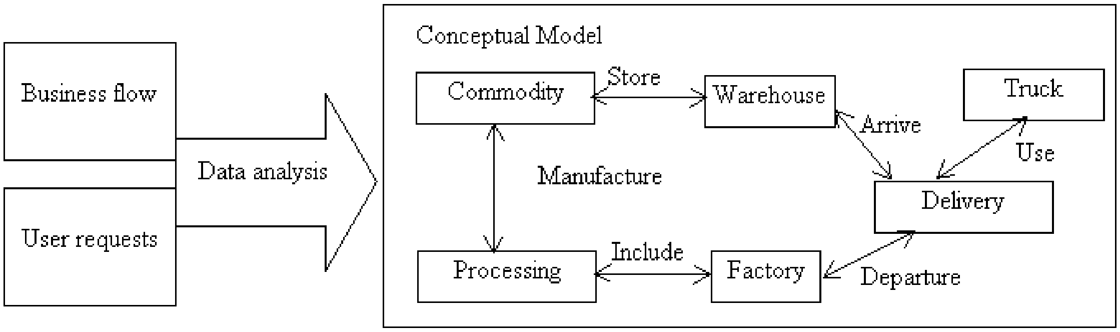
In the component-oriented approach, the work to be done by computer systems is determined first by examining what can be achieved by computers and what tasks must be performed by people.

In conventional system development, databases and files are usually designed to accomplish the processing objectives of specific business applications. Therefore, data duplication can occur without noticing. Data duplication makes it hard to ensure data integrity and means that any addition or modification of data affects other parts in the system. All of these factors degrade system maintainability.

An increasing number of companies are planning to reconstruct the existing information system so as to meet demands from changes in the industry. One problem that occurs related to such activities is how to integrate files in the current system with the new system. System development used to be carried out separately in the form of separate projects for individual jobs. As a result, data duplication, data inconsistency, and different data format caused a lack of system flexibility, integrity, and compatibility, and it was hard to acquire information across jobs.

In this circumstance, the concept behind the data-oriented approach is solving these problems by sharing data for linkage between jobs. According to this concept, the relationships among jobs are treated as relationships among data, and systems are divided into subsystems, and system functions are designed based on data sets structures.

In system development using the data-oriented approach, the inter-relationships among data used in the system are analyzed in a system analysis process (data analysis). After that, a model (conceptual model) is created in which data items are arranged without duplication according to job conditions and user requests.
Databases, files, and system functions are designed in the system design process based on this conceptual model.



Of the information shown in the system job flowchart, input and output data are checked in detail with job investigations and requirements analyses, but files are created for the convenience of processing by considering the relationship between input and output. If files were designed individually, data items may be duplicated in different files, resulting in unnecessary redundancy of files.
In the system analysis process, files are grouped to eliminate duplication and redundancy of data items. Moreover, the data items essential for the logic of job processing are examined and determined. The tasks called data analysis and conceptual model creation are performed for this purpose.
In the design process, the results of system analysis are used to examine file units, record data item definitions, and access methods, while considering resources and restrictions on actual operation.

1) Data analysis
In the first step, data is analyzed to determine the structure of individual data items. An example of a sales slip is shown below.

Document_code	Customer_code	Customer_name	Document_No	Transaction_date	Total_sales

Commodity_name	Commodity_code	Quantity	Unit_price	Sales_amount

□ Represents iterative data items

In the above layout, “commodity name” and “unit price” duplicate data items in the commodity master. Generally, “commodity name” and “unit price” are data items that should be stored only in the commodity master. One reason for including these items in other files is for convenience of processing. However, this means that any changes in “commodity name” and “unit price” must be included in not only the records in the commodity master but also the corresponding records in other files.

The problem caused by the duplication of such data items in a very big number of records in multiple files can easily be imagined.

During data analysis, the logical file structure is defined in the following procedure without duplication of data items.

First, find the data item (which may be a combination of multiple items) that can be the key for identifying individual records, and underline it.

For a record having iterative items, find an identification key for the iterative items, and underline it (do the same for nested records).

Document_code	Customer_code	Customer_name	Document_No	Transaction_date	Total_sales

Identification key for non-iterative items

Commodity_name	Commodity_code	Quantity	Unit_price	Sales_amount

Identification key for the iterative portion

A record having iterative portion is called a non-normal form record. The next step is to separate iterative portion and break the record into subrecords that have no iterative portion.

a. Creating a record without iterative portion

Document_code	Customer_code	Customer_name	Document_No	Transaction_date	Total_sales

.... 1 Record

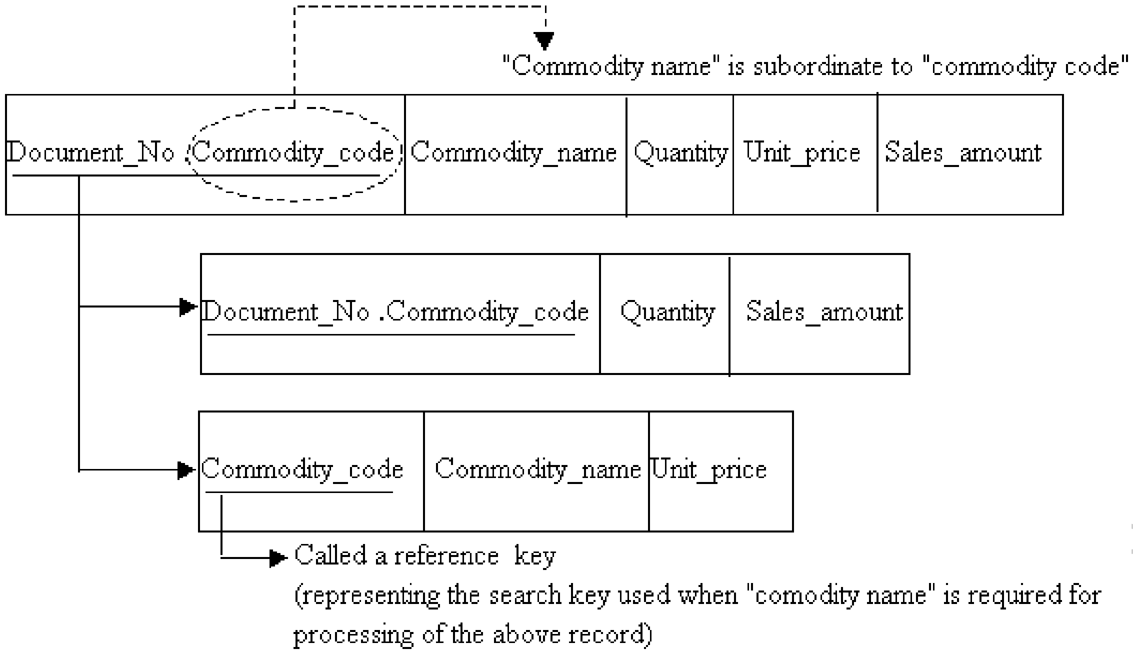
b. Creating a record by concatenating the identification key for the non-iterative portion and the identification key for iterative portion

Document_No .Commodity_code	Commodity_name	Quantity	Unit_price	Sales_amount

.... n records

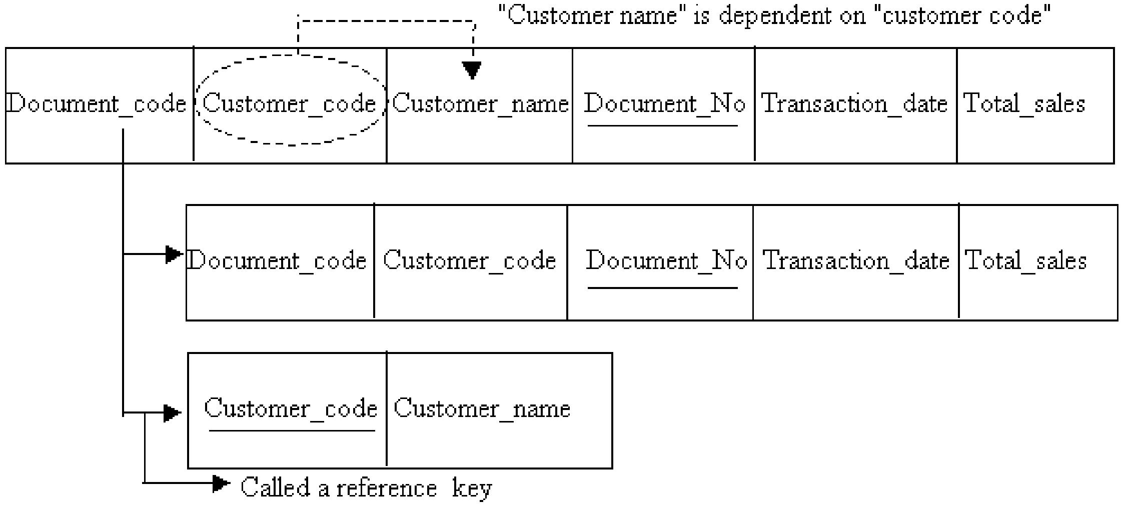
Identification key created by concatenation (also called concatenation key)

If the concatenated key includes a dependent data item, separate it into a subrecord. In the above example, "commodity name" in the record is a dependent data item.


"Commodity name" is subordinate to "commodity code"
Document_No
Commodity_code
Commodity_name
Quantity
Unit_price
Sales_amount
Document_No .Commodity_code
Quantity
Sales_amount
Commodity_code
Commodity_name
Unit_price
Called a reference key
(representing the search key used when "comodity name" is required for processing of the above record)

"Depend on" means a relationship which uniquely determines a data item (e.g. commodity name) by setting a value of the identification key (e.g. commodity code). Although "commodity name" is also determined when a value is set to the "document No. / commodity code" is made, consider on which item "commodity name" is strongly dependent.

Further, if a data item is dependent on a data item other than the concatenated key, separate it into a subrecord. In the example for step this would be applicable to a "customer name".



The results of the above record division operation are arranged as follows:

a. A record having duplicated data items

Document_code	Customer_code	Customer_name	Document_No	Transaction_date	Total_sales

Commodity_name	Commodity_code	Quantity	Unit_price	Sales_amount

b. This record is divided into subrecords that have no duplicated data items.
However, the identification and reference keys are not to be considered as duplication.

Document_code	Customer_code	Document_No	Transaction_date	Total_sales

Customer_code	Customer_name

Document_No .Commodity_code	Quantity	Sales_amount

Commodity_code	Commodity_name	Unit_price

c. The relationships among divided records can be maintained with identification and reference keys.

Perform the above operation for all virtual files shown in the systematization job flow charts, and combine the records having the same identification key into one record.

In the above example, one could imagine that the items "commodity name and unit price" and "customer name" in records that are created by dividing based on each assigning a reference key could be integrated into the commodity master and customer master, respectively.

A set of groups of record specified like this is called conceptual data, and the diagram that shows the relationship of conceptual data by using concatenated and reference keys is called a conceptual model. The conceptual model is utilized as one type of source information used to examine physical file units.

2) Creating an ER model

The following procedure is for creating an ER model by assuming the divided records described above as integrated records:

a. First, assign each record with name that clearly indicates the types of information in it. Job terms used for operation are preferable.

Sales:

Document_code	Customer_code	Document_No	Transaction_date	Total_sales

Sales detail:

Document_No .Commodity_code	Quantity	Sales_amount

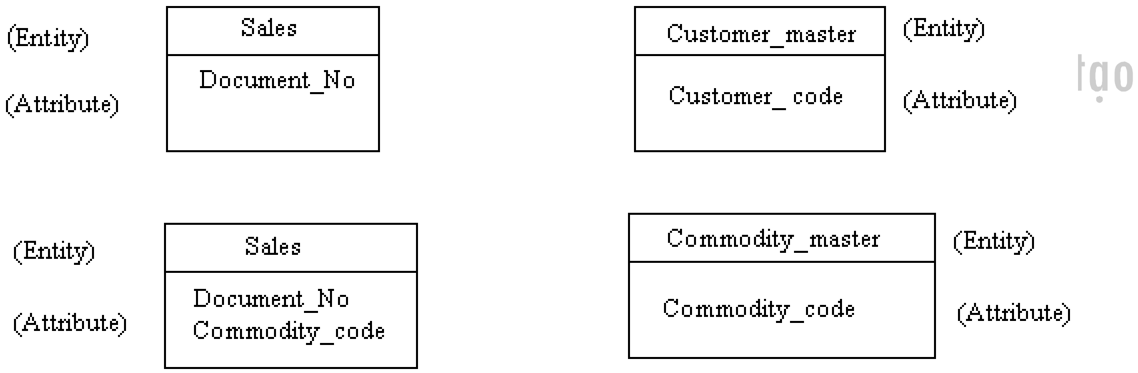
Commodity master:

Commodity_code	Commodity_name	Unit_price

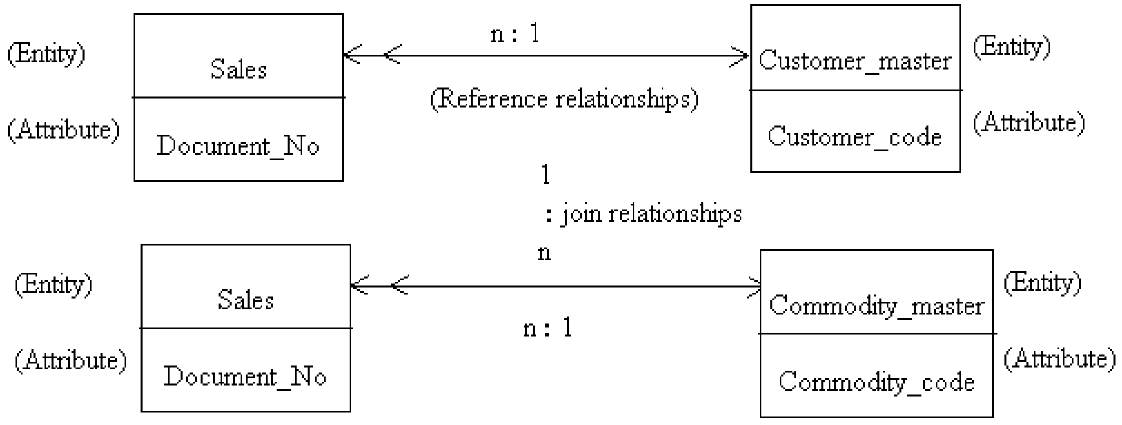
Customer master:

Customer_code	Customer_name

b. Next, specify record names and identification key names as shown below:



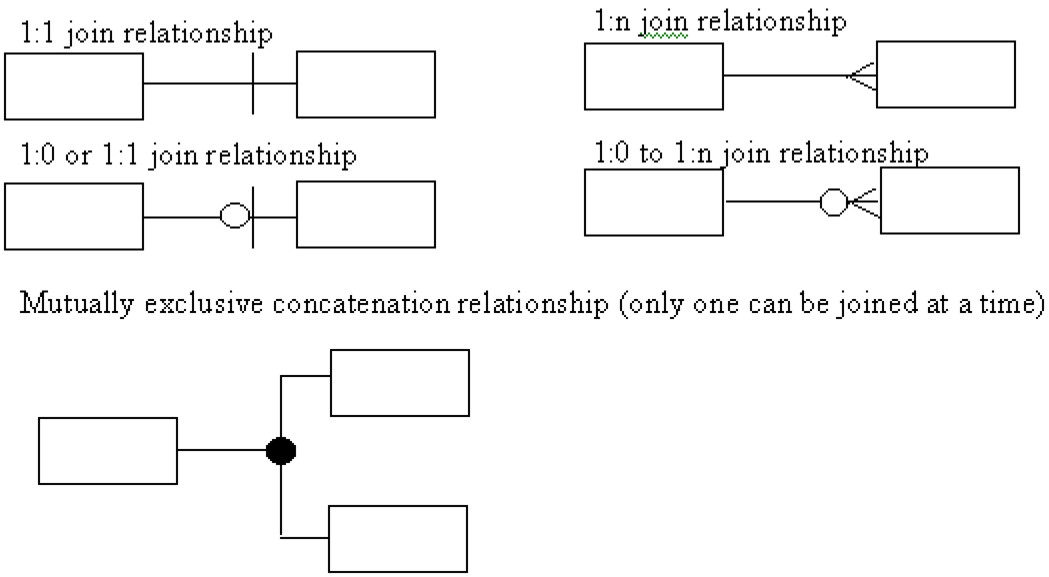
c. Connect the records with relationship lines based on relationships of concatenation and reference keys and the number of corresponding records.



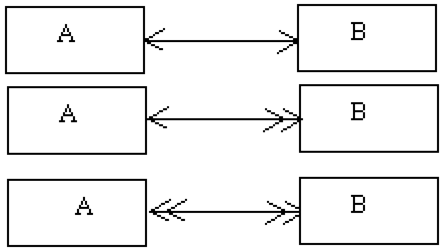
Examine the operating conditions to determine whether to define each ER model as one file (physical file unit).

[Reference: ER notation]

(1) Bird leg notation


1:1 join relationship
1:n join relationship
1:0 or 1:1 join relationship
1:0 to 1:n join relationship
Mutually exclusive concatenation relationship (only one can be joined at a time)

(2) Simple notation of relationships

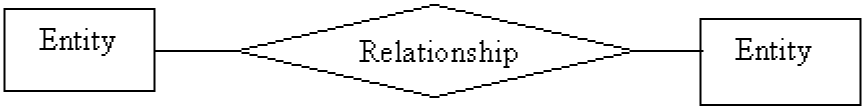


A and B have 1:1 relationship.

A and B have 1:n relationship.

A and B have m:n relationship.

(3) Other simple notations

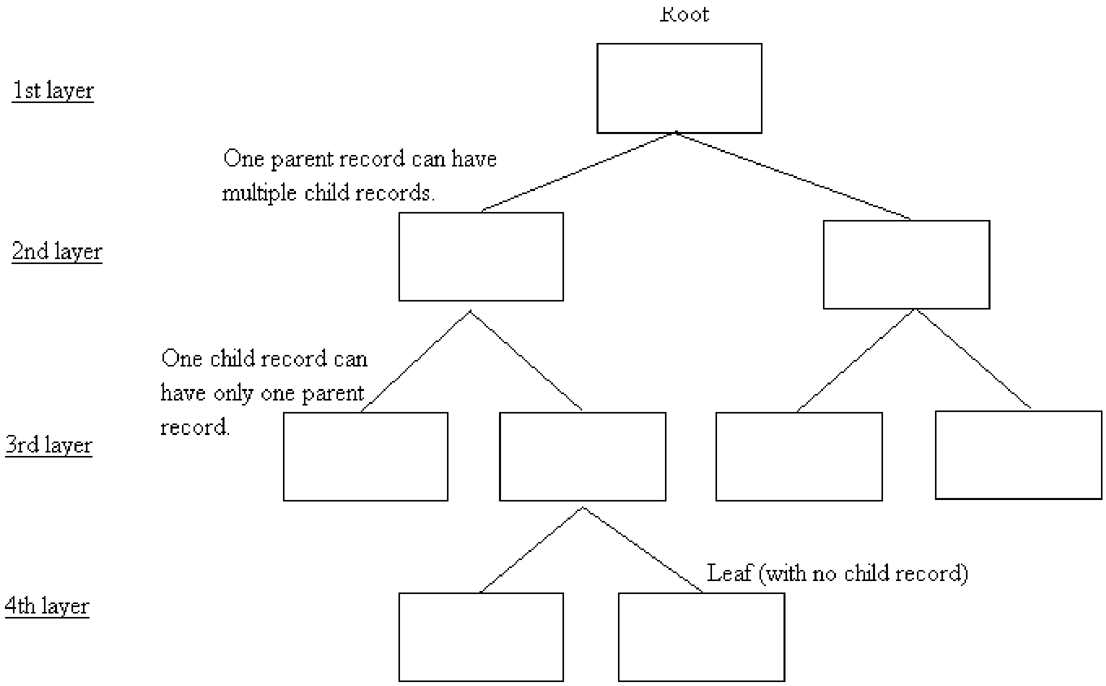


3.2 Logical Data Model

3.2.1 Hierarchical model

The hierarchical model is also called tree structure model because of its shape.
Multiple child records correspond to one parent record. Only one parent record can be viewed from a child record. For example, suppose a banking application where an account record is defined as a parent record and the transaction records of accounts are defined as child records. By storing their relationships in a single block on a disk, one can fetch information of the account record and the past transaction history in one access. With ordinary files, the account master and transaction master are accessed separately, and this takes time. However, if they are stored in one file, the processing algorithm becomes complicated. Using a hierarchical database avoids these problems.

[Hierarchical model]

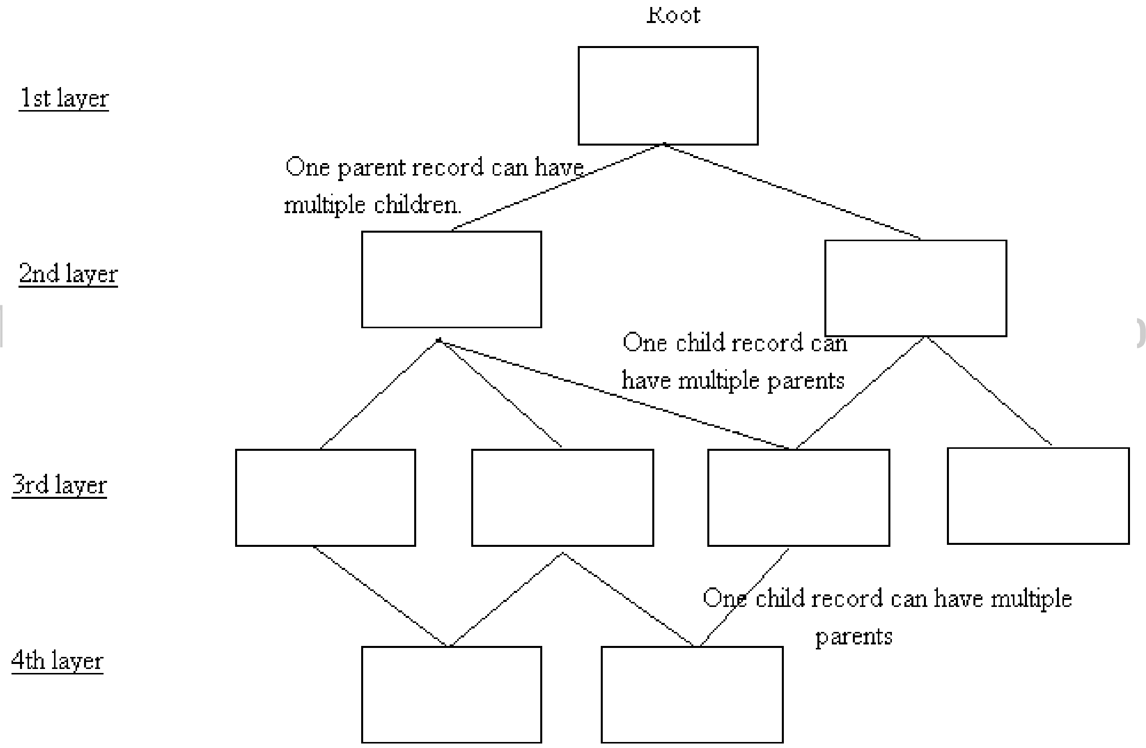

Root
1st layer
One parent record can have multiple child records.
2nd layer
One child record can have only one parent record.
3rd layer
Leaf (with no child record)
4th layer

3.2.2 Network model

The basic structure of the network model is the same as the hierarchical model.

In the network model, not only can multiple child records correspond to a single parent record, but a single child record can also have multiple parent records. Flexible databases can be implemented by extending the hierarchical structure described in the preceding section so that a single child record can have multiple parent records. With regard to improving access efficiency, the same as for a hierarchical structure applies.

[Network model]


Root
1st layer
One parent record can have multiple children.
2nd layer
One child record can have multiple parents
3rd layer
One child record can have multiple parents
4th layer

3.2.3 Relational model

The relational model is a model which was proposed by Edgar F. Codd of IBM. It is also called the tabular model.

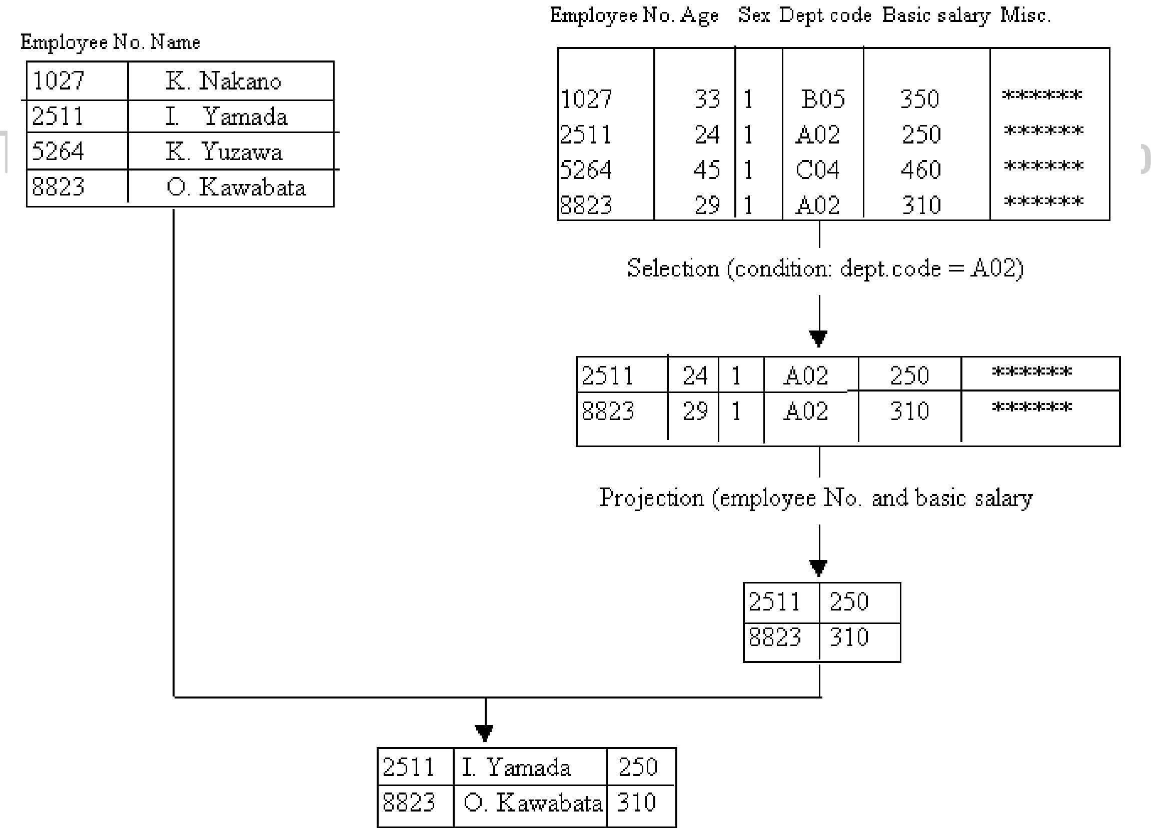
The relational model does not have a fixed structure. It is based on individual tables and logically connects elements in a table to represent the relationships among data.

There is no restriction on the sequence of rows and columns, but each column name and information in each row must be unique within one table.

Because its data structure is in the form of a two-dimensional table, the data in relational model can be handled or combined easily. The relational model can thus be utilized in various ways.

The basic operations data in this model are selection, projection, and join.

[Example of table manipulation of relational model]



Employee No.	Name
1027	K. Nakano
2511	I. Yamada
5264	K. Yuzawa
8823	O. Kawabata

Employee No.	Age	Sex	Dept code	Basic salary	Misc.
1027	33	1	B05	350	*******
2511	24	1	A02	250	******
5264	45	1	C04	460	******
8823	29	1	A02	310	******

Selection (condition: dept.code = A02)

2511	24	1	A02	250	******
8823	29	1	A02	310	******

Projection (employee No. and basic salary

2511	250
8823	310

2511	I. Yamada	250
8823	O. Kawabata	310

Exercises

1. Select the statement that correctly describes the data-oriented approach.

(1) Concept of a process based on data characteristics and the relationships among data items

(2) Developed version of a program structured design technique

(3) Opposite concept of object orientation

(4) Analysis of job, mainly by refining functions

(5) Assumption of use of a relational database management system in an implementation environment

2. Select the best description of characteristics of data-oriented design and component-oriented design.

(1) Encapsulation in data-oriented design means modularization of data and processes separately.

(2) Compared with process-oriented design, data-oriented design enables to use computer resources more effectively to gain as much performance as possible.

(3) For the first quick systemization of a specific job, data-oriented design is more effective than process-oriented design.

(4) The structured analysis method is used mainly for data-oriented design.

(5) In data-oriented design, job modeling is performed first followed by data modeling.

3. Select the sentence that correctly describes the data-oriented approach.

(1) The data structure is likely to change, but process specifications are relatively stable unless the target job changes.

(2) In a system developed with the data-oriented approach, multiple jobs use databases that are constructed separately.

(3) Data-oriented design is based on the idea that, considering data as shared resource, software specifications are defined from the resource side, placing importance on consistency and integrity of the data.

(4) The greatest advantage of data-oriented design can be found in the development of machine tool control software.

(5) From the initial design stage, data-oriented design treats logical design and physical design as one thing.

4. Select the best description of the ER model.

(1) The relationships among different types of entities are mainly represented as status transitions.

(2) An entity has either a characteristic value or instance value.

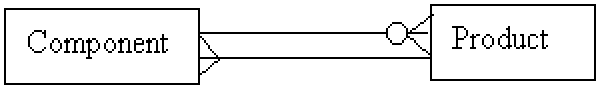
(3) The ER model is also called job model.

(4) An entity can only be any item that physically exists, such as a commodity.

(5) An entity has attributes that represent its characteristics.

5. Select the best description of the ER model.
(1) The logical flow of a program is represented in a nested structure of rectangles.
(2) Events in an application system are represented in a diagram showing the relationships among entities.
(3) Dependency relationships of functions and modules are represented in a tree structure.
(4) Program functions are represented in a diagram with a focus on the data flow.
(5) Program logic is represented in a flowchart.

6. Select the best interpretation of the following ER model.



(1) A component consists of products, and a product consists of components.
(2) A component is part of a product, and every product consists of components.
(3) Every component is a product, and a part of a product is a component.
(4) A component uses either no product or one or more products, and a product uses one or more components.
(5) There is a component existing as a product, and every product consists of components.

7. Select the best description of the ER model.
(1) Data and control flows can be written.
(2) Relationships that contain other relationships can be represented.
(3) The same entity can be on both ends of a relationship.
(4) A hierarchical relationship between attributes can be represented.

8. There are three data models: hierarchical model, network model, and relational model. Select the best description of the hierarchical model.
(1) Representations of objects in a network structure become redundant.
(2) The association between records does not have to be defined in advance, and records can be associated dynamically during data manipulation.
(3) The hierarchical model represents data in multiple two-dimensional tables.
(4) A certain record can have multiple parent records.

9. Select the sentence that does not describe the ER model.
(1) Only one type of relationship between two entities can be represented.
(2) Entity information and inter-entity information are represented separately.
(3) A relationship can have an attribute.
(4) Every entity must have an identifier, and an identifier consists of one or more attributes.
(5) Not only can entities be concrete objects such as a person, location, and building, but they can also be abstract concepts such as a technique and delivery process.

10. Select the sentence that correctly describes the ER model.

(1) After the relationships between every job process and data are defined, the resultant relationships among entities represent every process in a job.

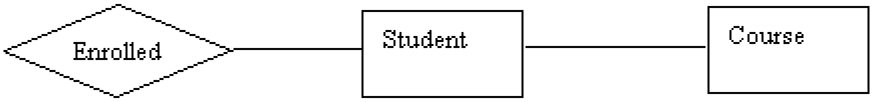
(2) The life cycle of data can be represented.

(3) Information used in the real world (job) is abstracted and represented as the relationships among entities.

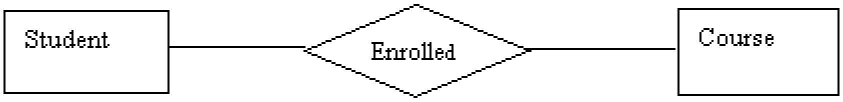
(4) ER models are created with the assumption that they are to be implemented in relational databases.

11. Select the ER model that correctly represents a student enrolled in a course.

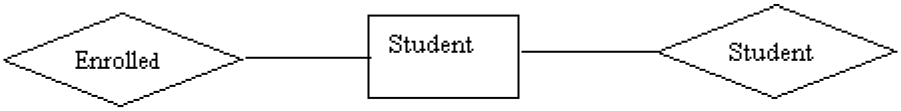
(1)



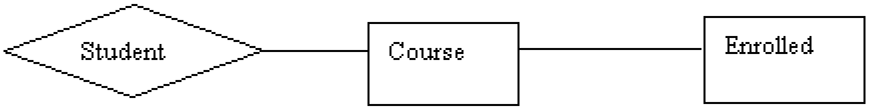
(2)



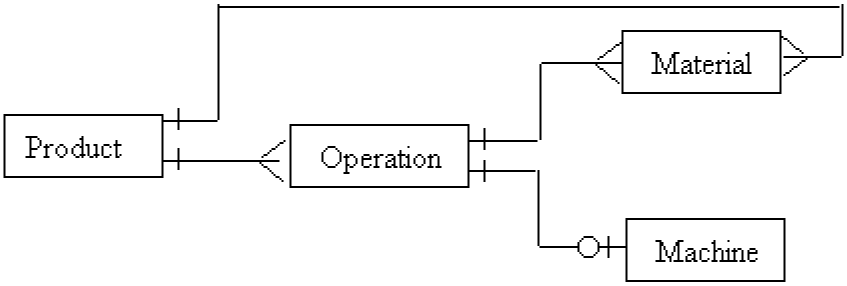
(3)



(4)



12. Select the best interpretation of the following ER model:



(1) One material is used for multiple products.

(2) One material is used for multiple operations.

(3) One operation manufactures multiple products.

(4) There are operations that do not require a machine.

4 Preparation of External Design Documents

Chapter Objectives

This chapter explains how to design an job model based on the relationship between job flows and subsystems after determining the system configuration and system hierarchical structure, and how to design the security measures required for each subsystem.

4.1 Preparation of User Manual (Outline)
4.2 Design of System Test Specifications
4.3 External Design Documents
4.4 Design Review

4.1 Preparation of User Manual (Outline)

4.1.1 Participants

The user manual includes a detailed explanation of the human interface and explains how to operate devices by following the corresponding operating manual, operation testing plan, item-code design, and input/output design.

In order to create an image of system operation, participants must collaborate with the persons in charge of design and the operation manager. Depending on the circumstances, end users might also participate.

4.1.2 Review method

Because process and data flows are made concrete in this design phase, a review generally takes the form of a walkthrough.

As a result of the review, the design specifications will be better understood. Another purpose of the review is to find problems in specifications and operations early in the development process.

4.1.3 Format of user manual

The user manual can be a paper, electronic, or WEB document. Which format to select depends on the preferred access method of the user and the frequency of the use.

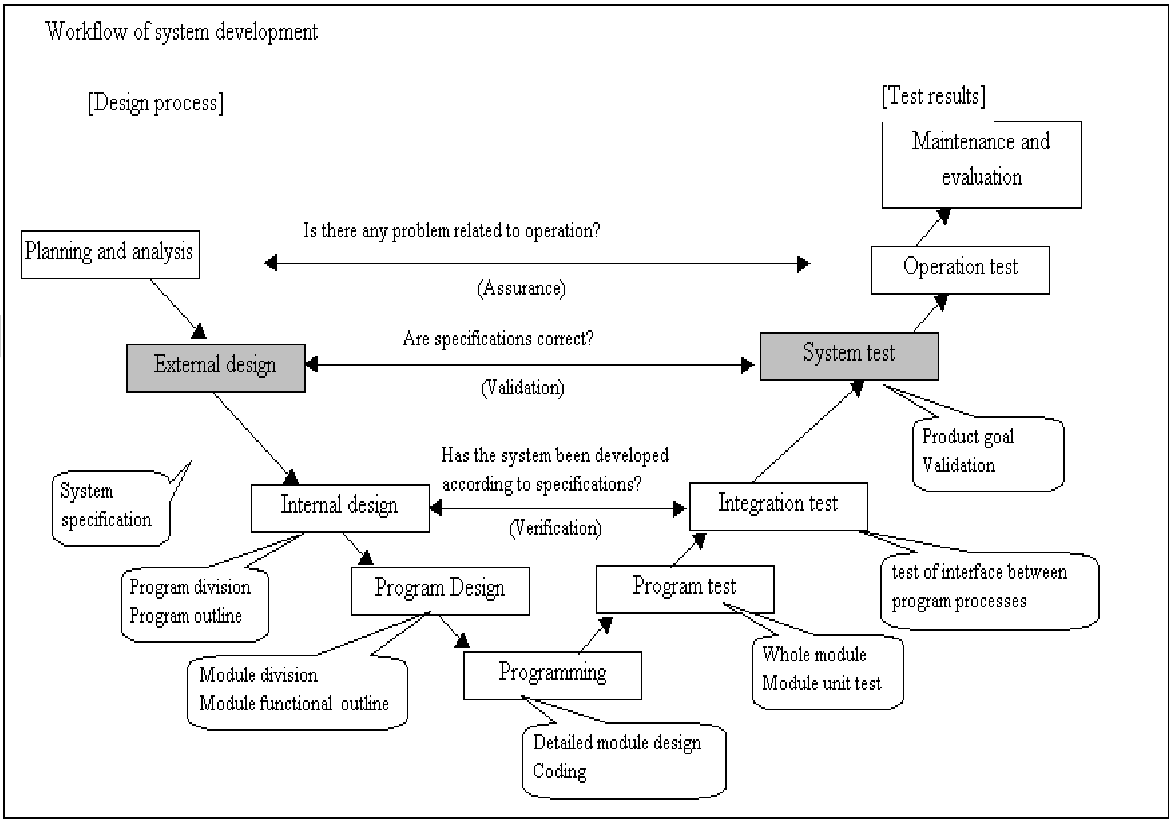
In the preparation of user manuals, it is important to ensure that the manual is easy to read, no procedures or tasks are missing, explanations are easy to understand, and that troubleshooting methods are easy to find.

4.2 Design of System Test Specifications

4.2.1 Policy planning for system testing

The system development processes are roughly classified into design and test processes. The design process does stepwise refinement from the external design, to internal design, and to program design.

In the test process, a system is checked to verify that it has been developed according to the specifications defined in the design process.



As shown in the table, the program test, integration test, and system test are conducted from the standpoint of the system developers, and the operation test is conducted from the standpoint of the users.

For test design, it is important to select and specify test items considering the standpoint of the persons who carries out the testing.

Test	Considerations for test item selection
Program test	- Do program execution results satisfy the functions described in program specifications? - Does all internal routes in the program work properly?
Integration test	- Can all programs in a component be executed? - Is data transferred smoothly between programs? - Is data transferred correctly between components?
System test	- Have product objectives defined in the external design been completely achieved? - Is consistency maintained between individual programs and the software as a whole?
Operation test	- Have all items required for actual job operation been tested sufficiently? - Is migration to a new system easy? - How difficult is it for the person in charge of operation to become familiar with the new system in actual operation?

The purpose of the system test is to determine the extent to which the objectives for the system as a whole have been achieved. To obtain an evaluation of the entire system, however, the test must be conducted from different standpoints, such as from the hardware or the operator.

The system test includes the following test items:

(1) Function test

The objective of this test is to check whether system functions are consistent with the target product functions.

(2) Performance test

The objective of this test is to check system performance (response, throughput, and processing time) against performance targets.

(3) Exception processing test (reliability test)

The objective of this test is to check processing routes other than those of usual system operation, such as error processing and recovery functions.

(4) Limit test

The objective of this test is to determine maximum system loads, such as related to simultaneous access to a resource.

(5) Configuration test

The objective of this test is to check all hardware devices supported by the system. In the case of a system whose functions can be used selectively (such as a mathematical planning system), this test is conducted for each software configuration.

(6) Compatibility test

The objective of this test is to check compatibility items (such as compatibility of function alone, compatibility with respect to input/output formats, and compatibility with respect to calculation precision) that are specified for a part or all of other systems.

(7) Operability test

The objective of this test is to check how easy for the end user to the system is to use (such as whether system messages are easy to understand).

(8) Durability test

The objective of this test is to check whether the system can withstand long periods of use.

(9) Security test

The objective of this test is to check the protection of sensitive data, including information privacy and confidentiality of in-house information.

(10) Document test

The objective of this test is to confirm that the user manual is correct before it is distributed to users. That is, the test checks whether users can perform all tasks by following the descriptions in the user manual.

(11) Memory test

The objective of this test is to confirm that sufficient memory capacity for system operation has been provided, such as in main and secondary memory devices.

(12) Ease-of-installation test
The objective of this test is to confirm that the operations and tasks required during system installation are not too complicated. For installation of software via a terminal, it is especially important that end users can install the software on their own.

(13) Regression test
The objective of this test is to check whether use of an existing program, modified by an addition or change of a function, affects other programs or causes unexpected results.

4.2.2 System test environment

The system test should be conducted under conditions that are as close to the actual operating conditions as possible.
Basically, the system test should be conducted for daily, monthly, annual, and occasional processing.

(1) Prepare operation schedules for batch and online processing.
(2) Assume actual operating conditions for online processing for reference purposes.
(3) Perform all tasks required for system operation, such as file initialization and file copying.
If a terminal cannot be connected, use a simulation tool for providing a virtual terminal.

4.2.3 Documentation procedure

Create documentation by using the following procedure:
(1) Filtering out test items
Determine and confirm with the external design documents which items have to be checked in tests.

(2) Designing test cases
To deal with applicable test items efficiently, analyze which test items can be combined and design appropriate test cases.

(3) Designing test data
Design the data that is to be used in the test. At this stage, it is not necessary to define specific values; it is sufficient to decide conditions for test data.

(4) Predicting test results
Project execution results for each test item specified during the examination of test items. Be sure to perform this task. Also examine methods of confirming test results.

(5) Preparing test specifications
Summarize the contents and results of steps (1) to (4), and include them in documents.

4.3 External Design Documents

4.3.1 System Configuration Diagram

The System Configuration Diagram shows a overview of the relationships between the human interface and individual jobs based on system requirements specification.

Prepare hardware, software, and network configuration diagrams as supplementary documents.

4.3.2 Subsystem Relationship Diagram

The Subsystem Relationship Diagram shows a overview of the relationships among those subsystems by dividing each system function into subsystems until it reaches basic level functions.

4.3.3 System Job Flow

(1) Definition of system functions

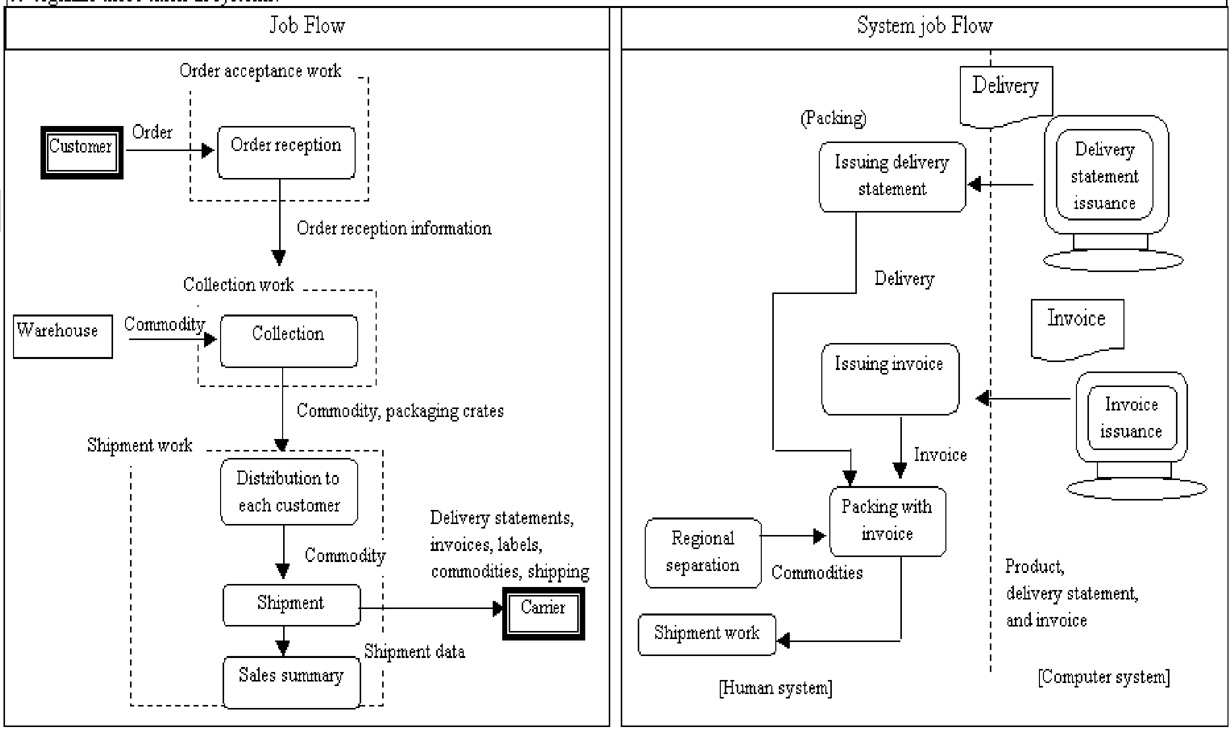
The System Job Flow is a document that shows an overview of system function definitions.

The System Job Flow clarifies the organization of work by defining which tasks are to be performed by people and which tasks are to be performed by the computer.

Definition of system functions

Clarification of which tasks are to be performed by people and which are to be performed by the computer

Define job function analysis results more precisely, clarify the tasks to be performed by people and those to be performed by computer, and check how to organize these tasks in systems.



4.3.4 System Function of Specifications

Divide the functions targeted for computerization according to the system job flowchart into smaller parts to determine the subsystems. Then, by organizing the functions with respect to function sharing, function distribution, and function integration, define the subsystems within the system and their relationships to one another, and collect this information in the System Function of Specifications.

4.3.5 Input-output Specifications (screen and report specifications)

In input/output design, consider not only consider data items and layout for conceptual files and input/output information, but also handling rules such as conditions and processing.

In input/output design, it is important to ensure that screens and forms are easy to understand and use.

Because it is difficult for the end users to decide whether the setup of screens and forms are acceptable without actually seeing and using the created system, specification changes may be requested later. However, because specification changes may significantly affect the system development, finalizing the specifications by conducting an early review by including users is important.

4.3.6 Data Specification

Various types of data are used in computer systems, such as for input/output forms, screens, data storage files, and codes for input/output. The data specifications specify the data structures used in the computer system.

(1) Code design

Various codes, such as for payment slips and for customers, are used in a computer system. For the code design, find out what kind of codes are required before designing the code system to be used in the computer.

(2) Input/output design

Computer users exchange data with computers using forms and screens. The input/output design defines the specifications according to which the user will interact directly with the computer for input/output. Input/output design can be roughly classified into two levels: item design and layout (format) design.

1) Item design

Specify the data format, length, and name required in input or output operations for every form and screen.

2) Layout design

Create a space chart that shows the precise layout of input/output forms and screens enough for program development.

3) Message design

Prepare a list of general messages to display on the screen.

(3) Logical data design

For the system function design, determine the kinds of files required for the system, and prepare file and database specifications.

4.4 Design Review

4.4.1 Review system

Because programs are created according to the contents of system design, all tasks performed during and after programming must start over, if the system design contains a mistake. If the program must be rewritten from the very start, this results in an enormous waste of time and money. Therefore, it is important that the system design contains no mistakes. On the other hand, the contents that must be handled during system design are huge, and errors may occur for such reasons as lack of communication and insufficient knowledge on jobs.

It is therefore important to find mistakes earlier in the development process by conducting a review. By finding mistakes, the amount of repeated work in subsequent processes is reduced, and the review also helps to improve system quality. The following three are the benefit of conducting a review:

(1) Early error detection reduces repeated work, minimizing work delays.
(2) System quality is improved.
(3) Productivity is improved, as a result of the synergistic effect of (1) and (2).

In other words, conducting a review can result in important advantages. It is important to select the appropriate mode of review according to the individual circumstances and carry it out effectively.

A review consists of the following kinds of elements:

- Frequency :Detailed review after completion of each work item, or review when a certain process is complete
- Participants :Review by users, or peer review by developers (designers)
- Method :Review by reading and discussing, or review by simulation
- Mode :Review by holding meetings, or review by circulation

(1) Conducting a review

1) Review by reading and discussing

In this method, the author explains the documents to be reviewed, and other persons confirm the details by asking questions. The designer clarifies the basis of the design by explaining the purpose and other details of the design. The participants carefully check the entire design to determine whether the design is consistent and complete, whether interfaces with other parts are provided, and standardization has been considered sufficiently.

2) Review by simulation

The purpose of the simulation review is to find errors. Solutions to errors are usually not investigated during the review. If solutions were discussed, it would take too much time.

[Review procedures]

- Determine in advance the time and place for holding a review meeting and its participants.
- Distribute review documents in advance, because participants have to read through the documents. The participants have to examine the design in regard to requirements for functions, performance, productivity, and maintainability.
- On the day of the review, the author explains the contents one-by-one by simulating

what would happen if implemented. The participants confirm the whether or not each requirements is met. (Walkthrough)

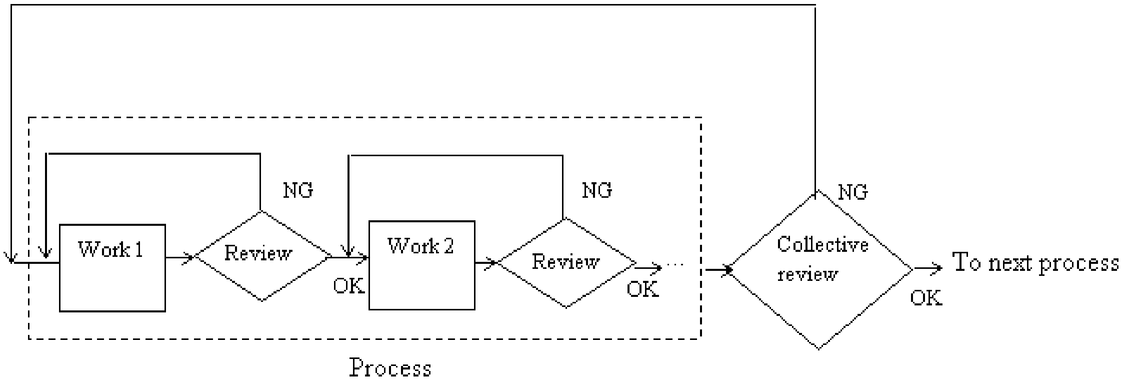
During the review, the scribe writes down comments. (Minutes)

If the discussion becomes sidetracked or the subject is off the topic, the leader intervenes and returns the discussion to the subject of the review.

- After the review, the author makes corrections based on the minutes, and the leader confirms the corrections.

4.4.2 Organization

There are many tasks in system development. The figure shows the relationship between a review conducted after each task in one process and a collective review conducted after the end of the process. In this concept, contents for individual tasks are clarified in a process and details are reviewed, and after all contents are found to be acceptable, the collective review for the process is conducted. Because this method has frequent reviews, the design contents assigned for each designer can be confirmed soon, and this results in less repeated work and a less effect on other designers if a problem is found. By conducting reviews frequently, the skill of each person in charge can be recognized, and if a person's skills are inadequate, an action to remedy the situation can be taken.



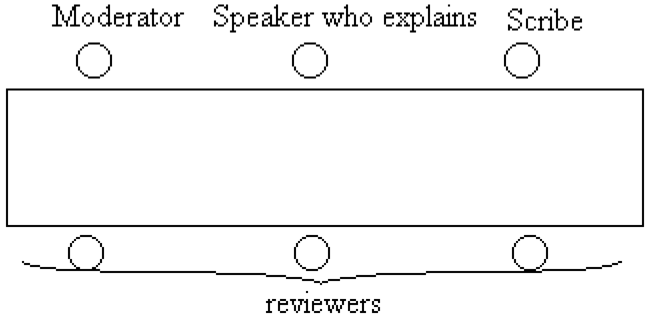
There are the following forms of review:

(1) Review by meeting
In a review conducted by meetings, persons in charge of design and development gather together for a meeting and carry out the review in the mode of a conference.

The person responsible for administering the meeting assumes the role of the moderator, and the person in charge of design explains the design to participants. This type of review is used when it is necessary to find out the participants' opinions and obtain their consent as well as having design contents confirmed. It has the following advantages:

- Because related personnel are gathered together, the review can be conducted from different viewpoints.

- Confirmation can be accomplished by Q&A.
- Examined results and confirmations are conveyed to all participants simultaneously,



which is also effective from educational perspective.

(2) Review by circulation

In case of a review by circulation, the objects for review are passed around among reviewers.

The document originator hands over relevant documents to the reviewers. The reviewers check the accuracy of the documents, whether any requirement is missing, and the consistency of terms. Then return the findings, and this process is repeated.

By conducting reviews every time a few documents have been prepared, document quality can be improved.

Generally speaking, the types of people who are suited for review are the following.

1) A persons who is not affected by preconception and can express opinion relatively freely.

2) A person who has technical skills and related experience

3) A person who can understand the importance and meaning of the document based on the objective of the system

If review results indicate a difference from the requirements specification, the external design document may be modified after obtaining the users' approval in case the modification is related to functional constraints in the implementation.

If any function is missing or has a problem in the requirements specification documents, the document should be modified and reviewed again.

[Reference: Review considerations]

Review considerations	Check item
Checks related to perfection	- Does any item still have to be determined? - Is there a reference to non-existing functions, input, or output? - Is any function missing? - Are any design contents missing?
Checks related to consistency	- Are notations, terms, and symbols consistent among specifications? - Do descriptions in the specifications correspond to plans and update specifications?
Checks related to feasibility	- Is the system designed for easier use? - Is the system satisfactory in performance, reliability, expandability, maintainability, and portability?
Testability check	- Are documents summarized in decision tables,etc. so that test cases can be easily analyzed? - Is there any ambiguous description? - Are tests handled in a quantitative manner?
Reliability	- Is the input error detection function sufficient? - Is the input combination check function sufficient? - Is an input value range check function provided? - Is a check for incorrect operation tasks provided? - Is a media setup error detection function provided? - Have preventive measures (such as providing a backup system) for hardware problems been studied?
Security	- Are access rights checked before confidential data can be accessed? - Are access rights checked before confidential function can be used? - Is there a function to detect and report unauthorized use to the system administrator?

Review considerations	Check item
Operability	- Are all operation tasks clarified? Are commands and messages corresponding to these tasks clarified? - Do messages have a standardized format? - Is a message outputted for every abnormal event ? - Is the method of replying to response requests standardized? - Is the input format standardized? - Are input operand keywords and parameters easy to understand? - Is the output format standardized? - Is necessary information, such as a title, page number, and output date, contained in header information? - Are output types clarified, and can the user select only necessary items? - Are output units and the floating point precision of calculation results appropriate for the user? - Are error messages from the system and output data clearly distinguished? - Is the response time appropriate for an interactive or online system?
Checks related to perfection	- Are performance target values clearly specified? - Has the customer confirmed the performance target values? - Are performance targets higher than the performance of the existing system? - Do performance values taken into consideration the system life cycle? - Has the validity of the requirements for setting performance targets been examined?
Expandability	- Has hardware expansion been considered? - Has the rate of increases in workload on the hardware been clarified? Is there a clear hardware expansion plan to match these projected increases?
Portability	- Is a standardized high-level programming language used to minimize dependencies on the computer type or OS? - Is the input/output interface a logical interface and free of effects caused by hardware differences? - Is a standard file format used? - Is the human interface logical and unaffected by differences in hardware and OS?

Exercises

1. From the following descriptions, select the two that apply to software design reviews.

(1) An advantage of design reviews is that a defect created in the software life cycle has a high probability of being found earlier in the upstream process.

(2) If system users participate in a design review, they tend to point out problems from different view points than that of designer's. Because this may delay the design process, it is better to keep them from participating.

(3) Although the design review is useful for setting an end point to design work, it may be simplified, because it is more effective to conduct a detailed program review in the next process.

(4) Because the purpose of the design review is to evaluate the design itself, it is better that the designer does not participate in the review, so that the evaluation becomes more objective.

(5) Various persons involved in development and use of the system should participate in design reviews, and conduct reviews from their different viewpoints.

2. From the following descriptions of objectives and results regarding carrying out walkthroughs during the course of a project, select the one that is not correct.

(1) Finding a mistake in the design process saves more time and money than if the mistake is found in a subsequent process.

(2) Because the purpose of a walkthrough is to find errors, programs are not modified at this stage.

(3) One of the purposes of walkthroughs is participation of managers and an evaluation of developers.

(4) Participants gain increased knowledge about the system by learning what the other people involved in the project are doing.

(5) Verifying programs by peers results in improved system quality.

3. From the following descriptions, select the one that applies to design reviews by walkthrough.

(1) A moderator who is neither a person in charge of development nor a manager is assigned to be the person responsible for planning and holding reviews.

(2) In order to find problems, development results are reviewed in detail under the leadership of a person in charge of development.

(3) Reviews by managers focus on the progress status of the project.

(4) The appropriateness of a development period and cost is reviewed by the person responsible for development and by user organizations.

(5) Countermeasures for problems are examined by discussions held in a free and open environment without restrictions placed on participants' speaking.

4. Select the sentence below that best describes how to proceed with a design review by walkthrough.
(1) A manager prepares a review results report and obtains approval from the quality assurance department.
(2) A document outlining the entire system is prepared for examination.
(3) A problem pointed out in a meeting is fully discussed at the time, and its countermeasures are determined.
(4) A manager plans a meeting and selects participants.
(5) A document for review is distributed in advance, so that participants can acquire an understanding of the related problems before the meeting.

5. From the following descriptions, select the one which is inappropriate to document review during software development.
(1) In each walkthrough, a period ranging from half a day to one day is necessary to find and correct all problems.
(2) Inspections are intended for reviews that are conducted only to find errors.
(3) The development schedule of a project must include a review period as part of the development work.
(4) By conducting a review, project members' sense of participation is raised, and new members can be expected to acquire increased system knowledge as a result.
(5) Walkthroughs and inspections are quality improvement measures that can be applied throughout the entire software development process.

6. From the following methods of review by a designer and multiple persons concerned, select the one conducted at the end of every design process to find design errors early in the development process in order to improve software quality and productivity.
(1) Top-down test
(2) System test
(3) Walkthrough
(4) Code inspection

7. From the following descriptions, select a correct one regarding system tests.
(1) A black box test is effective for checking how much extent of requirements specifications have been implemented.
(2) Avoid checking whether recovery from file destruction or a hardware problems is easily accomplished; otherwise, the completed system may be damaged.
(3) To conduct a system test, test cases should be prepared only in such a way that no higher load than assumed in the requirements specifications is applied.
(4) Because the human interface and ease of use during operation depend on the operating environment of the system, testing these items is meaningless.

8. When an existing program is modified by adding or changing a function, a check is performed to determine whether the modification affects other programs and/or produces unexpected results. Select a test that performs such a check.

(1) Field test

(2) System test

(3) Regression test

(4) Bottom-up test

9. Select the term below that means the work done in the external design process of system development.

(1) Logical data design

(2) Structured design of programs

(3) Physical data design

(4) Requirements definition

10. Select the sentence below that is the best description of a design review.

(1) In a round-robin system, the workloads of every participant are made to be the same, so that the system is not affected by differences in participants' experience, skills, and level of knowledge.

(2) Because selected problems and items can be analyzed from different aspects during inspection, applicable countermeasures can be developed easily.

(3) In a walkthrough, the leader explains to participants the review documents and products, both of which are prepared by members.

(4) Because participants do not review material before the inspection, the person in charge explains each item to the participants during a review.

(5) In development using a prototype, reviews can be replaced with direct verifications of prototype operations by users.

11. From the following system development processes, select the one that is <u>inappropriate</u> for external design work.

(1) Code assignment target objects are selected, and a code table is prepared for each code assignment target.

(2) Report output media and the report layout design are selected.

(3) Job flows are sorted out and checked to clarify users' systematization requirements.

(4) The program structure is determined, and detailed program functions are examined.

(5) Screen transitions and layouts for implementing interactive processing are designed.

12. Select the term below that means the work done in the external design during the system development process.
(1) Report design
(2) Physical data design
(3) Job analysis
(4) Test case design

13. Select the sentence below that describes an inappropriate means for making technical documents easy to understand.
(1) State facts and opinions separately.
(2) Do not use overly long modification phrases and double-modification phrases.
(3) Arrange modification words precisely in sequence.
(4) Use the passive voice as much as possible.
(5) Standardize terms and characters.

14. What is the correct sequence of test processes in system development?
(1) Operation test, unit test, integration test, and system test
(2) Unit test, integration test, system test, and operation test
(3) Integration test, system test, unit test, and operation test
(4) Operation test, system test, unit test, and integration test
(5) Unit test, integration test, operation test, and system test

Answers for No.3 Part 2 Exercises

Answers for Part 2 Chapter 1 (External Design Procedures)

(No exercises for this chapter)

Answers for Part 2 Chapter 2 (System Function Design)

1. (1) a, (2) e, (3) b, (4) f, (5) g

2. (1)

3. (1) a, (2) b, (3) c, (4) d

4. b, e, a, c, d

5. a

6. a

7. (1) X, (2) X, (3) X, (4) O

8. (1) X, (2) O

9. (1) O, (2) X, (3) X, (4) O

10. (1) c, (2) b, (3) a, (4) d

11. (1)

12. A

13. (1)

14. (3)

Answers for Part 2 Chapter 3 (Data Model Design)

1. (1)

2. (2)

3. (3)

4. (5)

5. (2)

6. (4)

7. (3)

8. (1)

9. (1)

10. (3)

11. (2)

12. (4)

Answers for Part 2 Chapter 4 (Preparation of External Design Documents)

1. (1) and (5)

2. (3)

3. (2)

4. (5)

5. (1)

6. (3)

7. (1)

8. (3)

9. (1)

10. (2)

11. (4)

12. (1)

13. (4)

14. (2)

Part 3

INTERNAL DESIGN

1 Procedure for Internal Design

Chapter Objectives

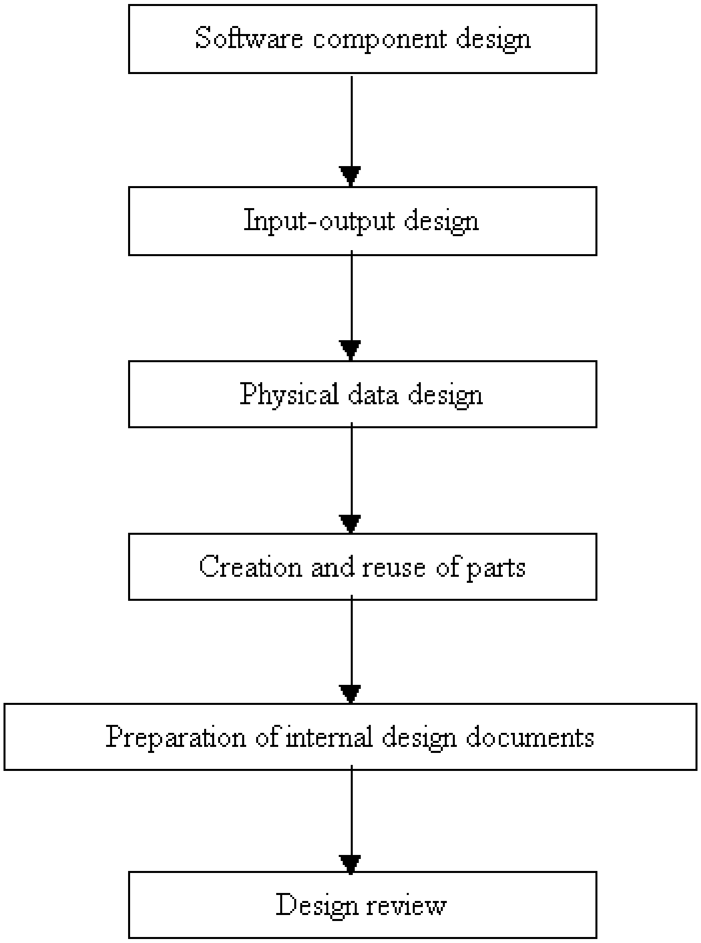
This chapter explains the necessary preparations and procedures that must be carried out before internal design.

1.1 Internal Design Procedures

1.1 Internal Design Procedures

Internal design is the work of designing the functions to be performed on a computer. They have been defined in the phases up to the external design. The characteristics of the computer equipment to be used must the taken into consideration, and designing work is conducted from the inside viewpoint of "how data is processed".

Software component design is the first step in internal design. Software components mean here the subsystems corresponding to functions that were defined during external design in units of components (programs). Software component design is followed by input/output design, physical data design, creation and reuse of parts, the preparation of internal design documents, and a design review.


Software component design
Input-output design
Physical data design
Creation and reuse of parts
Preparation of internal design documents
Design review

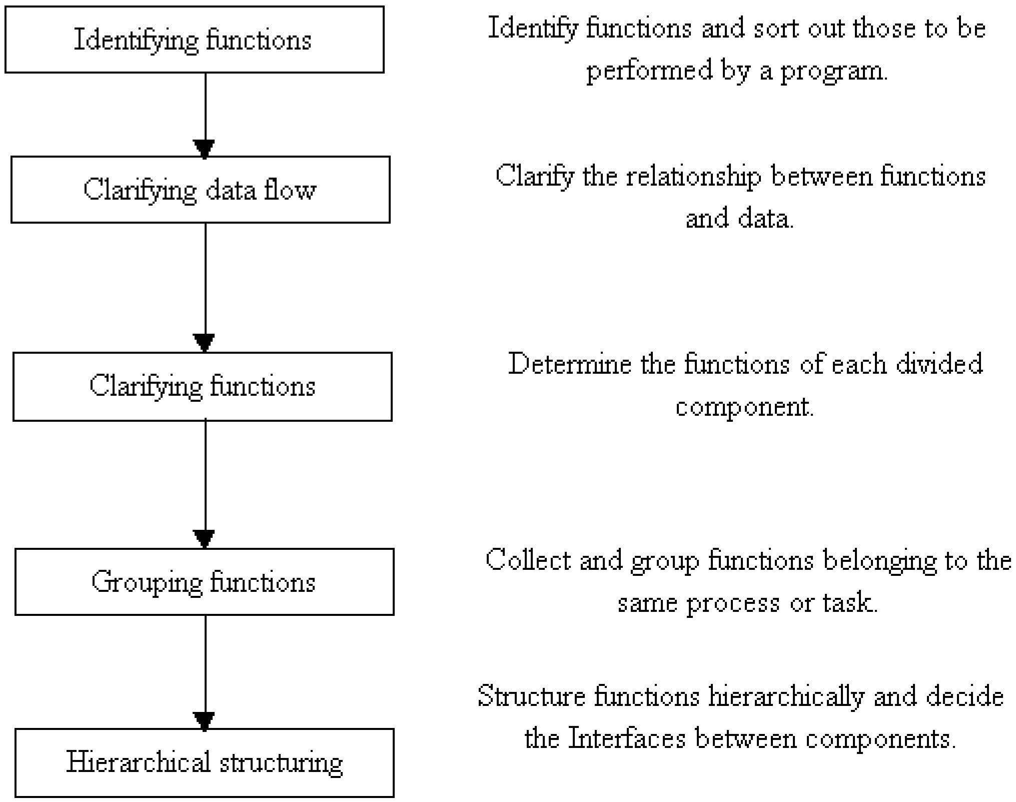
1.1.1 Software component design

The development of a system is stepwise refinement, proceeding in the sequence of overall system design => subsystem design => component design. The resulting system will therefore have a hierarchical structure.

Software component design uses a method called software structured design, in which subsystems defined during external design are divided into units of programs. This clarifies the program structure of the system and also shows what sort of program is to be created.

- Procedures for software component design

Software component design is performed by the following procedures:


Identifying functions
Identify functions and sort out those to be performed by a program.
Clarifying data flow
Clarify the relationship between functions and data.
Clarifying functions
Determine the functions of each divided component.
Grouping functions
Collect and group functions belonging to the same process or task.
Structure functions hierarchically and decide the Interfaces between components.
Hierarchical structuring

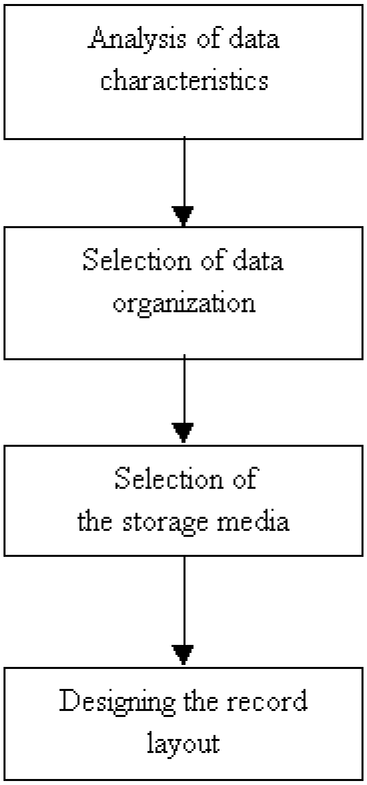
1.1.2 Input/output design

Input/output design is the design of forms and screens for the human-machine interfaces, which are the interfaces that connect a system with a user. Input/output design is classified into the following kinds.

Report design	Design of input forms Design of output reports
GUI design	Screen design
Data checks	Method for checking data Method for correcting errors

1.1.3 Physical data design

The physical data design (file design) is performed by the following procedures:


Analysis of data characteristics
Selection of data organization
Selection of the storage media
Designing the record layout

Consider and analyze such items as characteristics, uses, usage patterns, and maintainability.

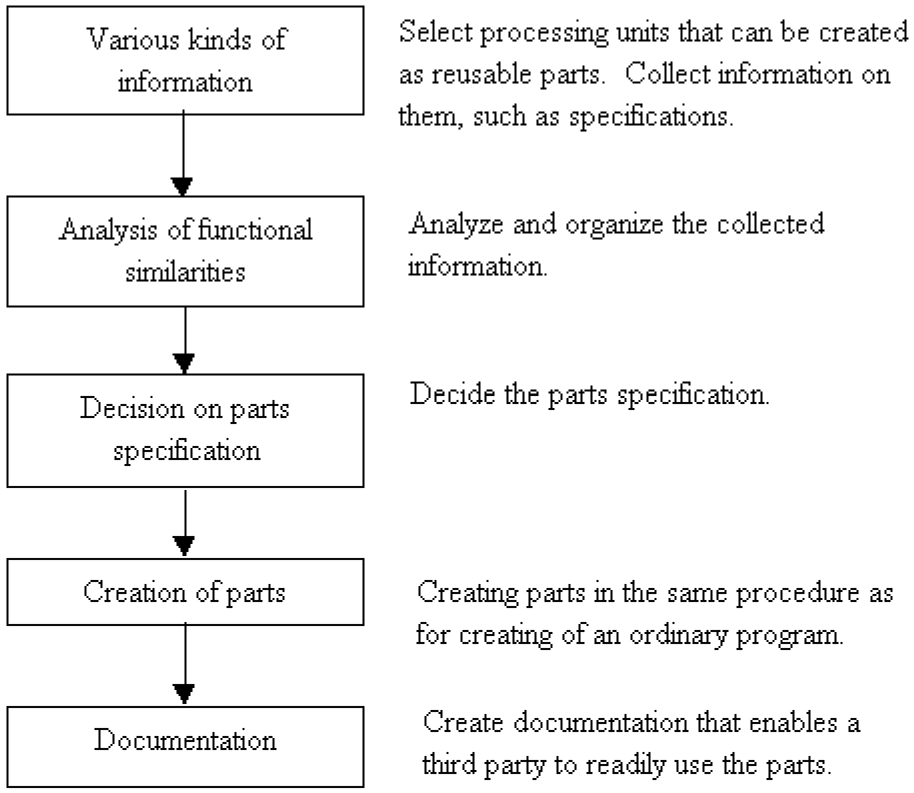
Consider storage media, of updating mode, processing method and related items and decide on data organization.

Consider the amount of data, transfer rates, storage capacity, and access methods and select storage

Decide how data items in records should be placed.

1.1.4 Creation and reuse of parts

Creation of parts is performed by the following procedure:


Various kinds of information
Analysis of functional similarities
Decision on parts specification
Creation of parts
Documentation

Select processing units that can be created as reusable parts. Collect information on them, such as specifications.

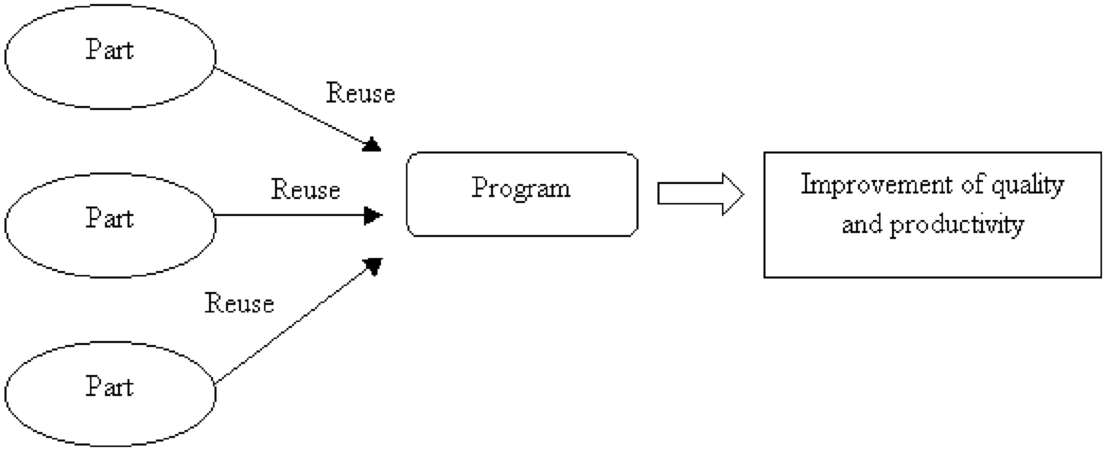
Analyze and organize the collected information.

Decide the parts specification.

Creating parts in the same procedure as for creating of an ordinary program.

Create documentation that enables a third party to readily use the parts.

Effectiveness of the Creation and Reuse of Parts



1.1.5 Preparation of internal design documents

Writing down the work performed so far in internal design documents.

<Outline of internal design documents>
- Diagrams for partitioning into components
- Interfaces between components
- Component processing specifications
- Screen design documents
- Report design documents
- File design documents
- Database design documents
- etc.

1.1.6 Design review

Review to detect defects and omissions.
<Methodology>
Walkthrough: Peer review by developers including the author.
Inspection : Carried out in an organized fashion as administered by a person in charge (called a "moderator").

Review the documents that were produced during the internal design. Evaluation of the results of internal design will prevent the problems from being carried over to the subsequent work phases.

2 Software Component Design

Chapter Objectives

In this chapter, you learn how to partition a system into components based on the output of the system design from the viewpoint of internal processing. This chapter also explains design component functional specifications and interfaces between partitioned components.

2.1 Designing Software Structure
2.2 Use of Middleware and Tools

2.1 Designing Software Structure

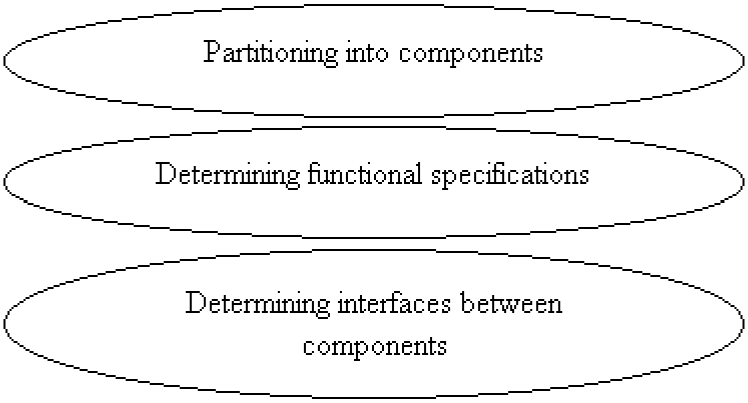
As software becomes larger and more complex, the overall structure (software architecture) of the system greatly affects software design. Software architecture refers to the structure of software, that is, the elements of the software, their functions, and the relationships among the elements. The software architecture expresses the structure and the characteristics of software in the implemented environment when the system has been implemented.

The software component design during the internal design is performed as software structured design. The software structured design partitions the subsystems that are determined according to the external design into functions (components), based on the internal specifications.

By partitioning the subsystems, the structures of programs in a system are clarified, and the specific processing that the program has to perform becomes clear.

There are two categories of component design. One is the design of structures where a single component corresponds to multiple programs, and the other is the design of structures where a single component corresponds to a single program.

Software structure design consists of partitioning into components, determining the functional specifications for each component, and determining interfaces between components. This results in software component design.


Partitioning into components
Determining functional specifications
Determining interfaces between components

<Software component design>

< Procedures of software component design >

(1) Identifying functions
In a subsystem partitioned according to the external design, the program functions are identified in detail in order to clarify the processing functions that are required for the program.

(2) Clarifying the data flow
Relationships between the functions and data are clarified by understanding the flow of data, as expressed in the data flow diagram.

(3) Clarifying processing functions
The processing functions of partitioned components are determined.

(4) Grouping processing functions
The functions belonging to the same process or task are grouped.

(5) Arranging the hierarchy of the functions
The functions are arranged in a clear hierarchical structure according to their relationships among each other, and the interfaces between the components are determined.

2.1.1 Partitioning into components

Component decomposition during software component design means to partition a subsystem into units of programs.

The systemization job flow chart is created by dividing the structural functions of the target jobs into roughly two categories: jobs done by people and jobs done by computers. The functions shown in the systemization job flow chart are only roughly specified. For actual computer processing, the subsystems must be partitioned into components where jobs are physically executable units of processing. Also, the systematization job flow chart does not always clearly show how files connect one programs to another. In the work of designing system functions, the component functions that are extracted in an analytical process must therefore be divided to clarify relationships among components, with regard to computer processing.

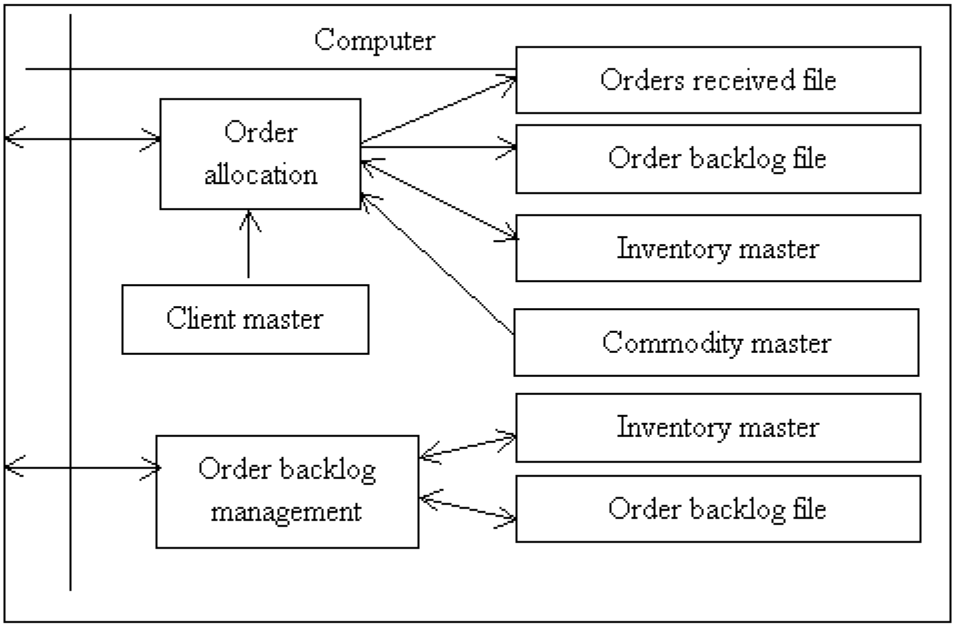
< Procedures >
- Structuring the data flow for components
- Partitioning of components (Components relationship diagram)

(1) Structuring the data flow for components

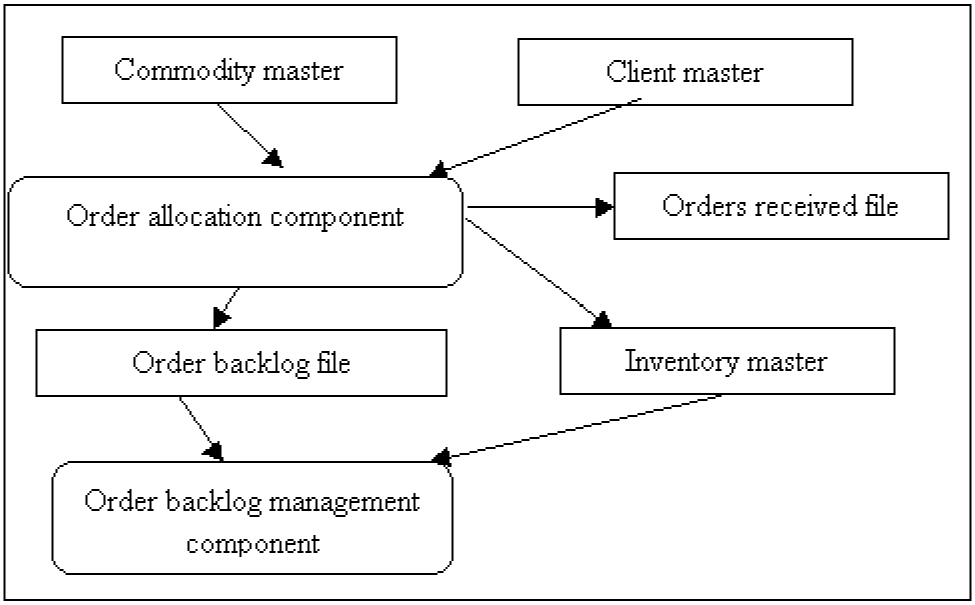
The components specified in the systematization job flow are extracted first, and then the files used as interfaces between the components are clarified because data transfer between components is accomplished with files.

For example, the following systemization job flow chart includes the inventory allowance and the management of order backlog. The order allocation component and the management of order backlog component are related using the input/output files of the respective components. In this example, both of these components access the inventory master file and the file of order backlog. Data transfer between these components is accomplished by using the inventory master file, and those files function as intermediary files.

Systematization job flow



Components relationship diagram

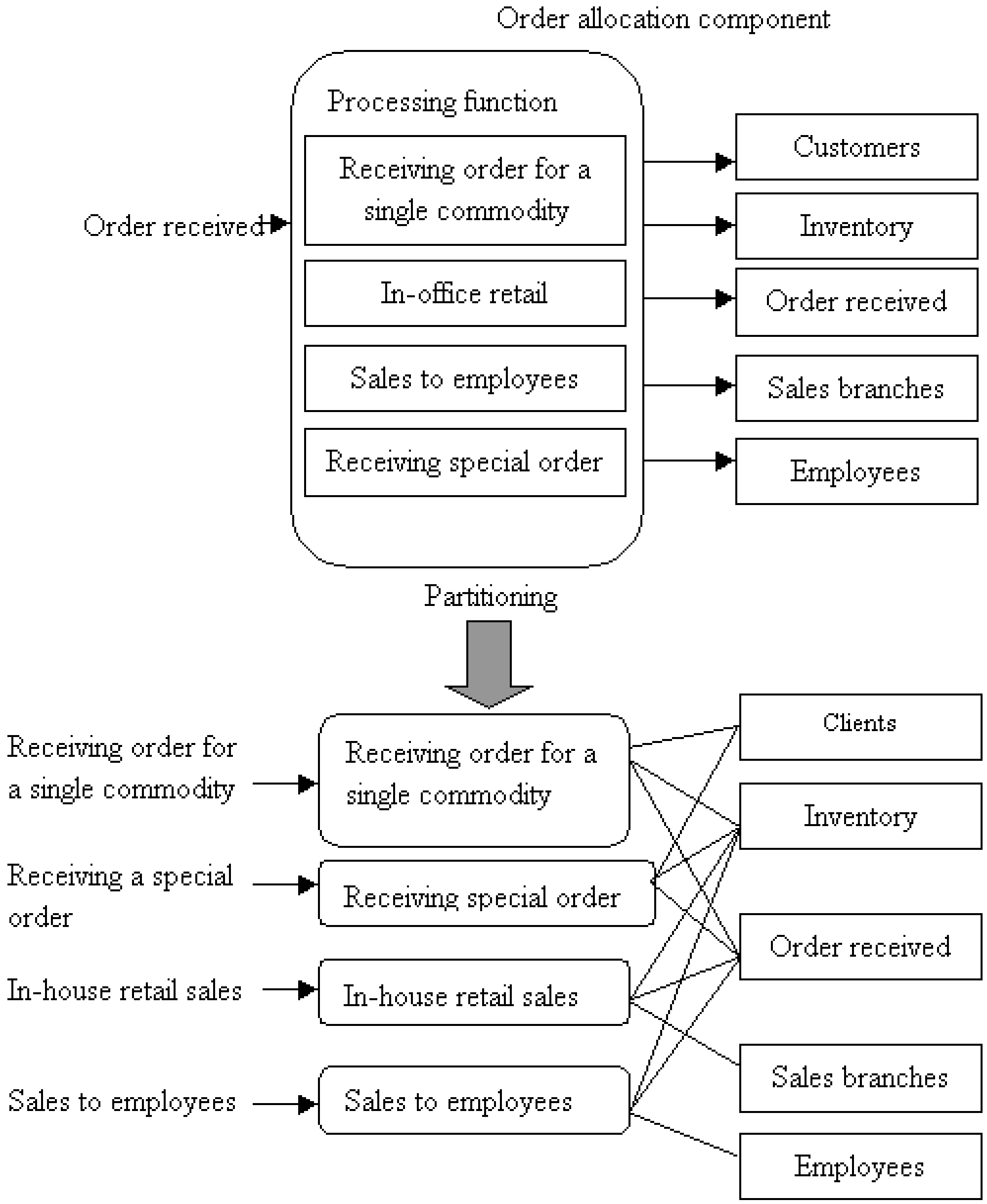


(2) Partitioning and integration of components

Relationships between functions of components and the file units for connecting the components are specified, when the components relationship diagram is created. At this stage, however, the components differ to a greater or smaller degree in the level of their functions, and similar processing functions may overlap. The processing functions need to be further partitioned for actual computer processing, because the components relationship diagram defines the components according to their functions for job processing. Therefore, component functions are partitioned or integrated considering both the operational aspect and the development aspect (ease of development and prevention of redundant development).

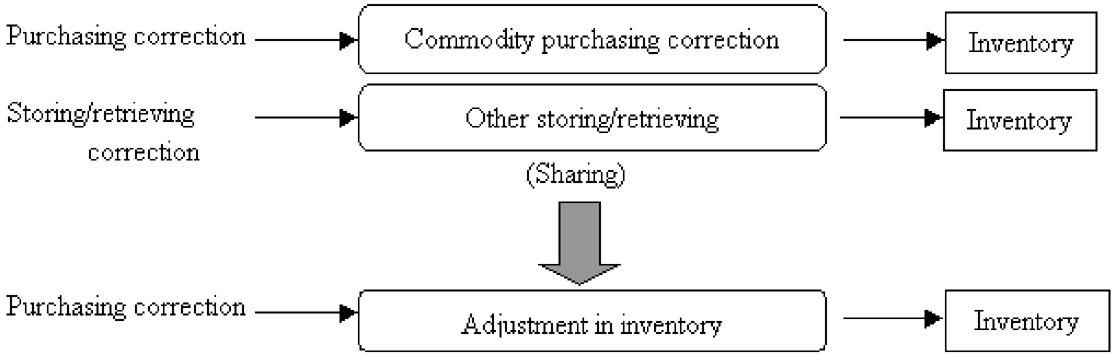
- Partitioning components

The illustration shows an example for this task. When the order allocation component is specified and processing functions, such as receiving orders for a single commodity, in-office retail, sales to employees, and receiving special orders, are required, these functions are examined to determine whether they should remain in a single component or be partitioned into separate ones.



- Integration of components

It must be reviewed whether components that have similar job processing functions can be shared. The amount of software development work can be reduced if similar components are shared.



(3) Method of partitioning a component

A component can be in the following ways

1) Partitioning by job function

If components have similar functions but cannot be handled as the same component in operations, determine whether they must be partitioned into separate components according to their individual functions.

2) Partitioning for efficiency of development

If a component has multiple functions, partitioning the component can reduce the complexity inside the component. This enables developers to concentrate on the development and testing of core job functions with high traffic volumes.

3) Partitioning by processing pattern

Typical processing patterns such as integrating files, partitioning files, and creation of statistical tables are prepared in advance. Analyze the combinations of processing patterns with which the basic function to be partitioned can be processed. A component can then be partitioned into program units by processing patterns.

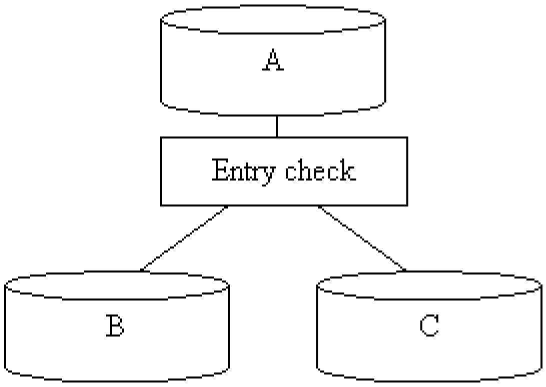
(Typical processing patterns)
Integration of files
Partitioning of file
Collation of records
Retrieval of records
Addition of record
Classification of records
Creation of statistical tables

a. Partitioning programs for batch processing

This section first explains the partitioning of programs for batch processing.

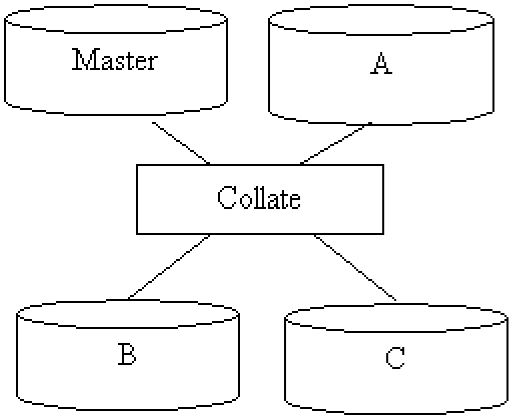
For partitioning by processing patterns, functions of programs that require batch processing can be classified in the following processing patterns, which are frequently used.

< Input check system >



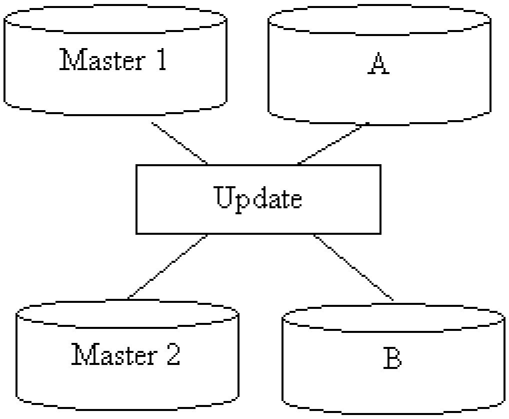
Contents of input data file A are checked. If they are correct, they are written to file B. If they contain an error, they are written to file C. This pattern also applies to extraction process of records.

<Collation system>



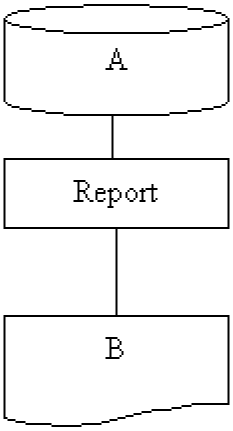
Input data file A is collated with the master file. If the input data matches, it is written to file B. If the input data does not match, it is written to file C.

< Update system >



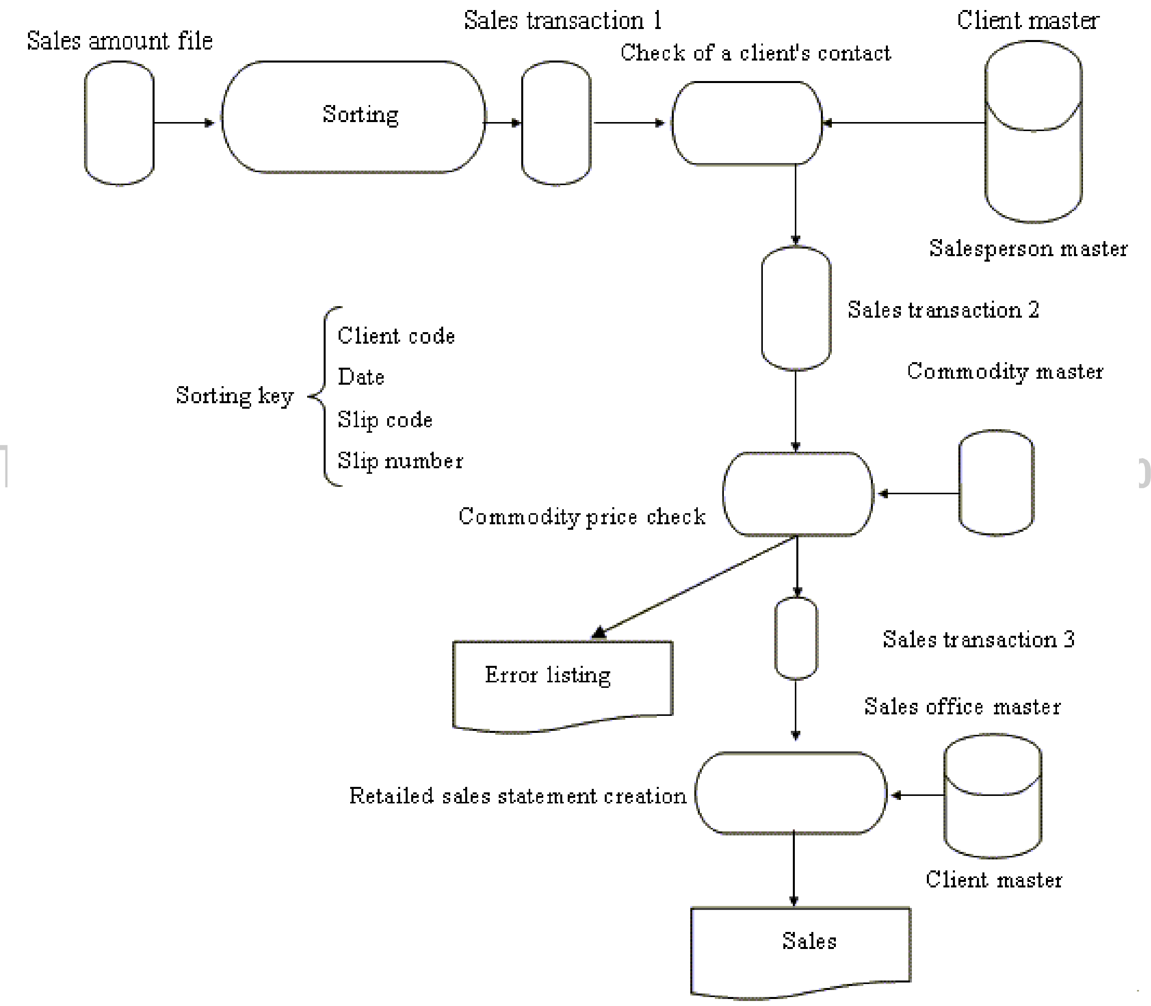
Contents of the old master file 1 are replaced, deleted, or added, depending on the contents of input data file A, and new master file 2 is created.

< Report creation system >

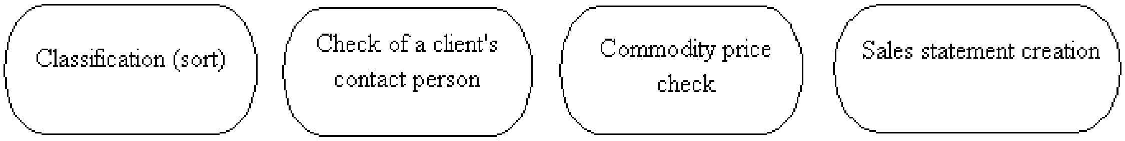


Create listing B by using entry file A

The work of partitioning a component into programs starts by linking the data flow and function within a component in greater detail, according to the system flow diagram and the system outline definition created in the external design process. The data flow that is created in the same process for the daily sales report creation component within the subsystem for the sales amount is shown below.



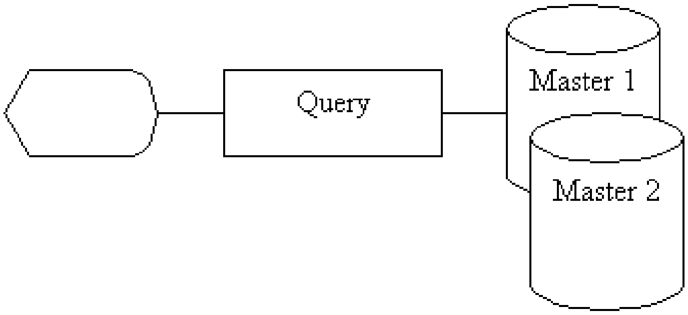
Next, partition components to make the processing patterns simpler. In this example, partitioning into the following four programs is possible.



Also try to reduce the program size as much as possible to make development and maintenance easier. For example, a guideline may be established to limit the module size of COBOL programs for compilation to 500 steps or less. Because no program in this example has an especially large number of steps, each program remains as is.

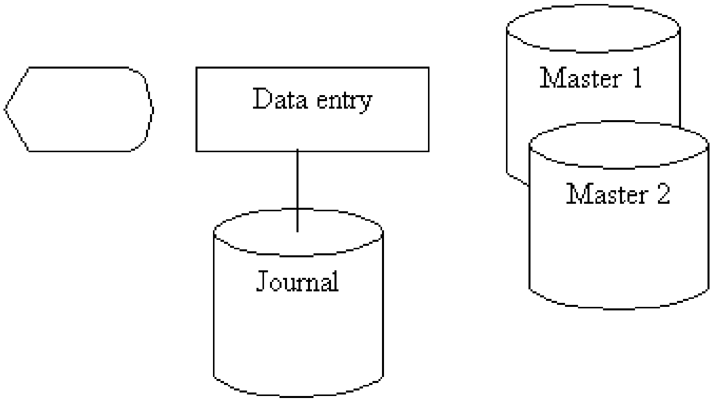
b. Partitioning programs for online processing
Partitioning programs for online processing is explained below.
For partitioning by processing patterns, functions of programs that perform on-line processing can be classified in the following processing patterns, which are frequently used:

< Query processing >



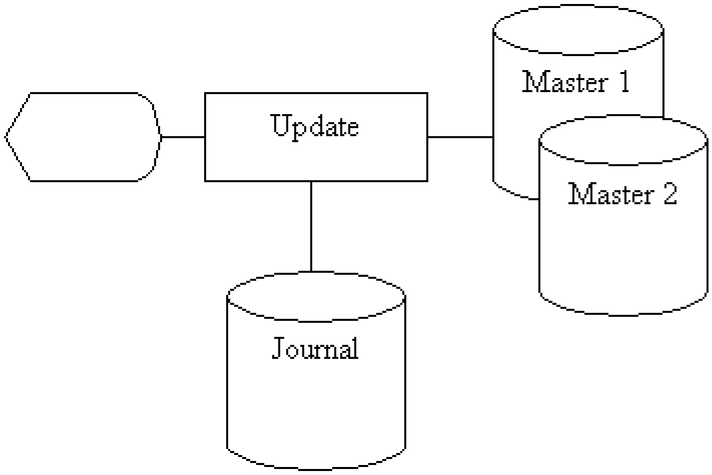
Search Master 1 and Master 2, and output the result on display.

< Data entry processing >



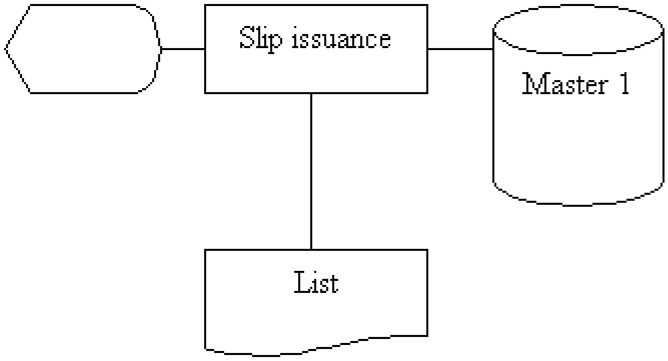
Contents of input data are checked and written to master 1 and master 2. The history is written to a journal.

< Update processing >



In order to create master file 2, the contents of master file 1 are replaced, deleted, or added, depending on the input data. The history is written to a journal.

< Slip issuance processing >



The list is created based on input data. At this time, the history is written to a journal.

- Functions in the component must be understood, and partition a program to match with these patterns as much as possible.

The most important feature of online programs is performance. During program partitioning, the program is partitioned in tasks, according to the system outline definition created in the external design process. (In online processing, it can be assumed that programs correspond to tasks.) Partitioning of a task is described as follows by using interactive processing (process with multiple questions and answers) that is the most frequently used in outline processing.

In interactive processing, the input/output screens and tasks are closely related. The following two types of processing are provided as a guideline, even for tasks that are to be partitioned. In the discussion below, the program is assumed to have a transaction management function (connections between program and terminal can be disconnected for each transaction) and a multiple dialog function (function for carrying out dialog processing with multiple terminals concurrently)

- Multiple screens and a single task

A series of interactive processing functions that execute a job is carried out in a single task.

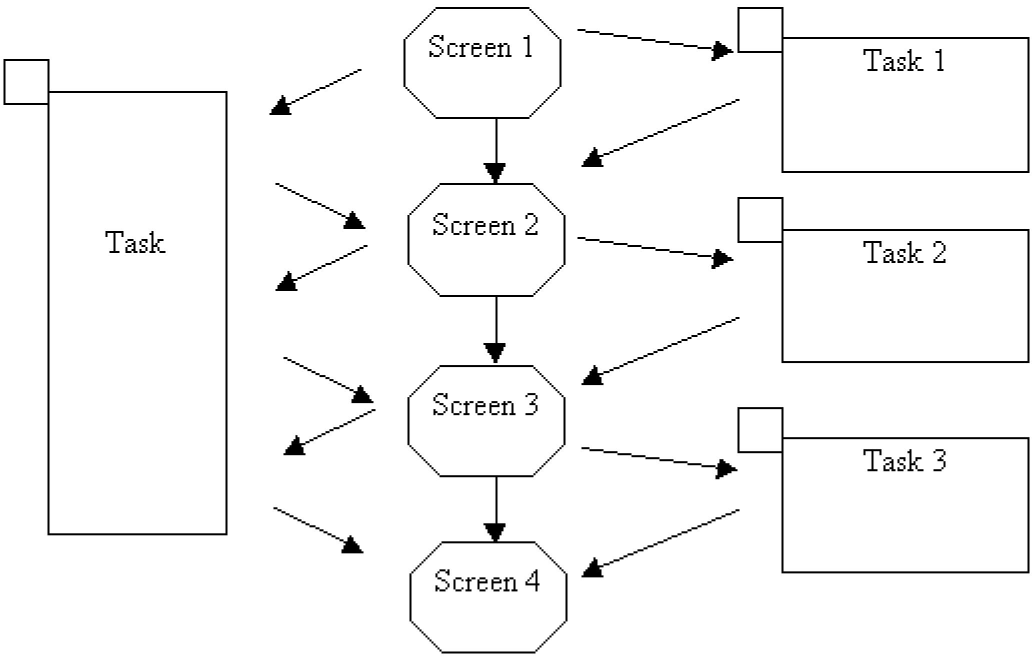
Implementing these functions in a single task, the program generally becomes complicated and development productivity declines, but the response of the program becomes better than a program with multiple screens correspond to multiple tasks. Multitasking should be selected when the traffic intensity becomes high. If the program is designed with a re-entrant structure (a structure where re-entry is possible), memory requirements can be lowered. When a multitasking operation is selected, exclusive control must be used because multiple tasks could access the same single file at the same time. If exclusive control is not used in this case, the contents of the file can be corrupted. At the same time, preventative measures must be taken against system failures and abnormal terminations, because it would be possible that updating has been completed for some files but not others, which would result in a loss of consistency between the files. Algorithms for recovery from such system failures and abnormal terminations would be sophisticated. Therefore, the algorithm of the program should be designed in such a way that all file updates are performed at the end of processing.

- Multiple screens and multiple tasks

With this approach, the interactive processing function for each input/output screen is executed as a single task.

Because responses to each input/output screen is handled by a separate task, files and tables must be provided for sending data from one program to the next. In other words, more file accesses and table retrievals are required than for multiple screen and single task processing, and response is generally not as good. However, the programs are simpler, and such processing has therefore such advantages as ensuring good productivity and easy recovery in case of a system failure or abnormal termination by using the take-over files.

	Responsiveness	Productivity	Reliability	Memory
Multiple screen and single task	Excellent	Poor	Recovery processing is complicated.	Less than that for multiple screen and multiple task processing
Multiple screen and multiple task	Poor	Excellent	Recovery processing is easy.	Greater than that for multiple screen and single task processing



4) Partitioning by difference in processing timing

The differences in processing timing include the following.

Processing cycles --- Cycles such as daily, weekly, monthly, or term-end

Time point of processing --- Processing at the beginning or end of the term or processing and online start or end processing

5) Partitioning by processing efficiency

Array of records

To update a master file in a sequential organization, the records to be processed must be arranged in the same sequence as that of the master file keys. If efficiency can be increased by rearranging the sequence of records, the program should be partitioned by master files that are to be updated.

a. Update processing of a master file

If only "correct" records are to be updated during a master file update, input data check processing and update processing of the master file should be partitioned as separate programs.

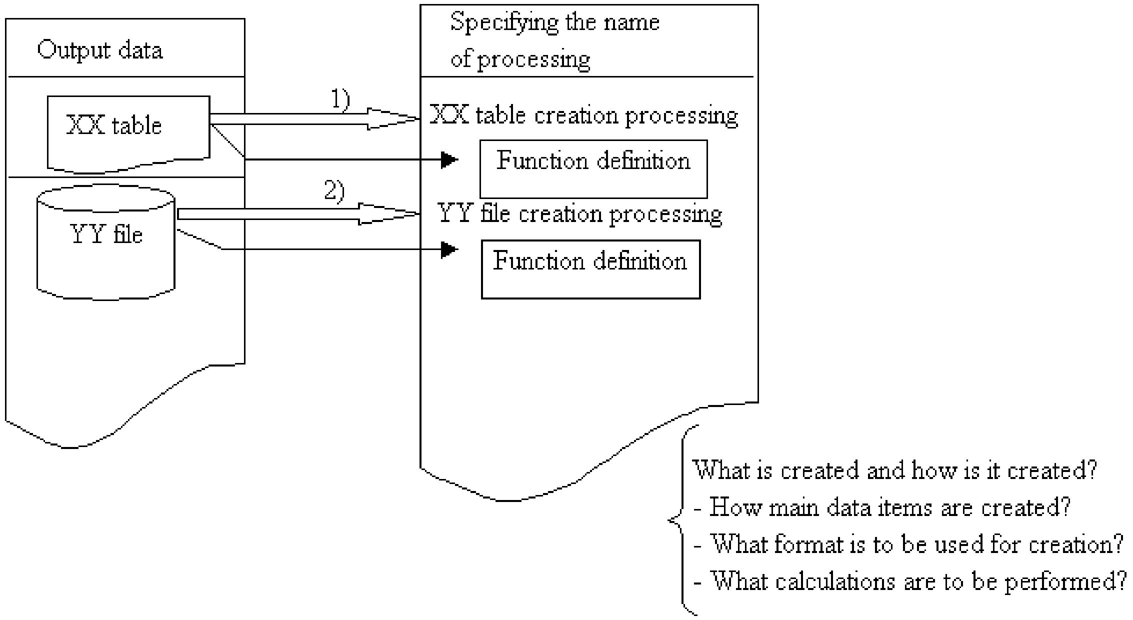
2.1.2 Determining functional specifications

The functions, input-output data, and the master file are clarified in this step, and the results are compiled into the "functional component definition" for every component that has been partitioned.

The component outline definition can be created by following the procedure below:

1. Definition of component functions
2. Definition of input-output data
3. Definition of input source and output destination

(1) Definition of component functions
Component functions must be defined to clarify the purpose of the component. Each function must be described in the format of "noun + verb".
1) A name for processing to output data must be given.
2) An outline must be defined in what condition and how output data is created.


Output data
XX table
YY file
1)
2)
Specifying the name of processing
XX table creation processing
Function definition
YY file creation processing
Function definition
What is created and how is it created?
- How main data items are created?
- What format is to be used for creation?
- What calculations are to be performed?

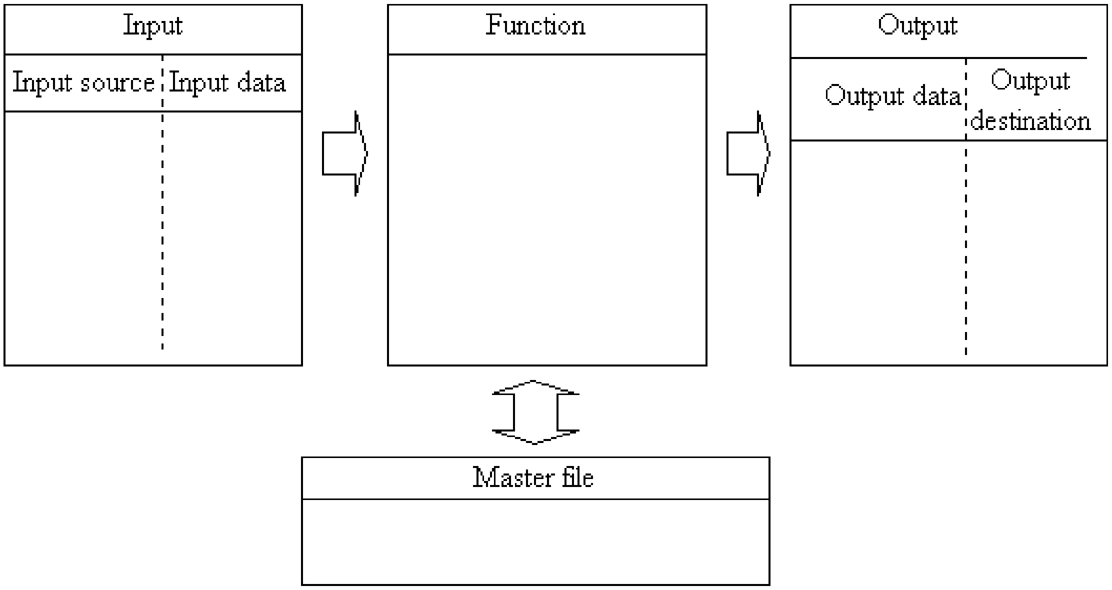
(2) Definition of input-output data
The necessary input-output data and master file are created for every processing function.

(3) Definition of input source and output destination

When department becomes an input source and output destination, the name of the department is to be written. When other components become the input source and output destination, the names of the components are to be written.

An example of the component outline definition is given below:

< Component outline definition >



Note the following points:

1) During the definition of a function, the function must be described with a focus on “what” is to be created. Specify “how to” if necessary, but make sure not to describe it in detail.

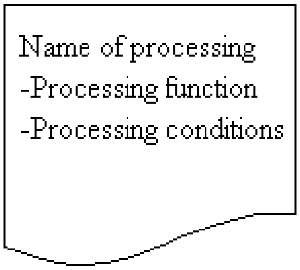
2) If the component function cannot be described on a sheet of paper, use only general-purpose paper for a supplementary description. If there are specific conditions, if any, specify them.

(Example of supplementary description on general-purpose paper)

Component function supplementary data

= Method of calculating the workload of a programmer =
Programmer workload = a x (KDSI) b x c
a, b and c are constant values. These values can be calculated from the metric criteria and statistical data.

3) Function section

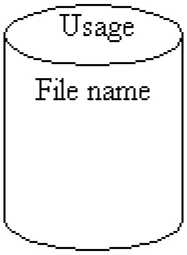


- Specify a name that corresponds to the input-output data as much as possible.

-Briefly explain the processing function that is performed under the name of the processing.

4) Input-output section
Write down the input-output data names that are defined in the components relationship diagram.

5) Master file section
Write down the file name and usage in the media drawing symbol.



-Usage
Description of how to use input, output, update, etc.
-File name
Match the master file name with the file name in the system flow diagram

(4) Consideration on determining component function specifications

1) Understandability

When the flows of data and control in a program can be easily understood, productivity during development, operability, and maintainability will increase. Therefore, programs should be partitioned to be easily understood. For this purpose, it is important that the interfaces between components are simple and the flow of data control is easy to understand.

2) Completeness

The processing contents of partitioned components must be sufficient to provide the contents defined for the original basic function. To achieve completeness, functional omissions during external design are often compensated during internal design.

3) Productivity of development

Productivity of development can be improved by partitioning complicated processing into separate programs. The specifications should be such that they do not to include too complicated processing into a single component.

4) Operability

Ease of operation after implementation must be considered for the program to be created.

5) Processing capacity

The specifications on requirements definitions and processing capacity that were agreed on during external design must be satisfied.

6) Maintainability

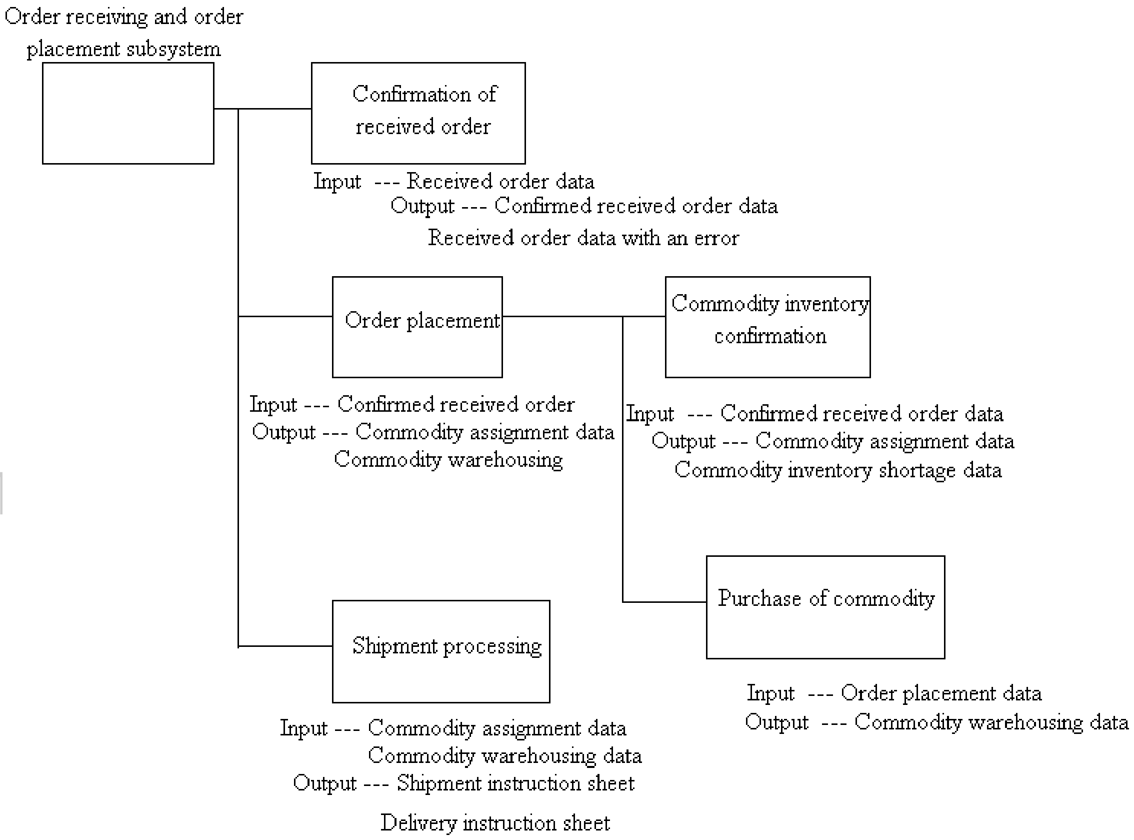
The time required for operation and maintenance may become longer than that required for development, and maintenance costs become greater than development costs. Therefore the specifications must be determined considering the ease of maintenance.

7) Reusability

Reuse of software is a very effective method, because it leads to not only reduced development and maintenance costs, but also to improvements in quality. Therefore, specifications and partitioning must be determined with due consideration to reusability.

2.1.3 Interfaces between components

Specifications of the interfaces between components are an important factor for partitioning into components and determining functional specifications. The interface for input and output must be defined for each component.


Order receiving and order placement subsystem
Confirmation of received order
Input --- Received order data
Output --- Confirmed received order data
Received order data with an error
Order placement
Input --- Confirmed received order
Output --- Commodity assignment data
Commodity warehousing
Commodity inventory confirmation
Input --- Confirmed received order data
Output --- Commodity assignment data
Commodity inventory shortage data
Purchase of commodity
Input --- Order placement data
Output --- Commodity warehousing data
Shipment processing
Input --- Commodity assignment data
Commodity warehousing data
Output --- Shipment instruction sheet
Delivery instruction sheet

2.2 Use of Middleware and Tools

-Middleware

Middleware is software that is positioned between the OS and application software. It provides sophisticated and practical services (functions). Examples of middleware are database software, development software, and communication software.

(1) Typical middleware

1) Data Base Management System (DBMS)

DBMS is software that manages databases as common data and handles access requests to the data. DBMS data formats and procedures of use are standardized, so that the DBMS can be made independent from any specific application software. Productivity and performance of application software can be improved by assigning data management to the dedicated software.

Data that can be managed with a DBMS can be classified into several types. Typical types are the card type, relational type, and object type. At present, relational type is the most common one. The greatest market shares are held by Oracle of Oracle Corporation for large-scale systems and by Access of Microsoft Corporation for small-scale systems.

2) Transaction Processing (TP)

Transaction Processing refers to a processing method in which multiple related processing operations are grouped into a single processing unit.

All of the multiple processing operations that are managed as a transaction are assured to be either a “complete success” or a “complete failure.” For example, in systems such as a fund transfer system, processing such as money deposits and money payments must be a “complete success” or “complete failure.” Transaction Processing (TP) thus means that both of the above processing operations will be grouped into a single transaction and that the all processing is determined only after the state of “complete success” or “complete failure” is reached. A system for implementing TP is called TP monitor.

3) Common Object Request Broker Architecture (CORBA)

CORBA is a specification of the distributed object technology established by OMG (Object Management Group: standardization organization for distributed object technology). Standards for software (called ORB) are specified, so that messages can be exchanged among objects (program parts) in distributing environments containing different types of computers. The CORBA specifications include the basic structure of ORB, the ORB procedures to be called from programming languages, and the rules for message exchange between different ORBs.

- Tools

There are different kinds of tools for software development: tools for use in the design stage, and tools for use in the development stage. This section describes tools to be used in the design stage.

Typical design tools

1) Structured Analysis and Design Technique (SADT)

This tool is for structural system analysis and modeling. The features of this tool are stepwise partitioning of functions and modeling them in a hierarchical structure.

2) Documentation tools

These tools supports creation and editing of the software documents that are generally created in the process of software development. Examples of documents that are created in the design stage are listed below:

- Screen design documents
- Form design documents
- Program design documents

Exercises

1. From the following descriptions, select the one that describes the purpose of software component design in the internal design process.

(1) To clarify the program structure of a system and what kind of program is to be created

(2) To provide an understanding of data linkage with other job systems and to clearly specify data exchange with other systems and processing timing

(3) To review operating requirements and the hardware and software structures determined during the planning stage for a system, and organize the operating conditions of the system.

2. The following description is about the relationship between component partitioning and the systematization job flow chart. Complete the description by selecting the correct words from the group of words/phrases listed below.

The systemization job flow chart is created by dividing the structural functions included in the target jobs into roughly two categories: jobs done by (1), and jobs done by (2). The functions specified on the systematization job flow chart are only roughly specified. Consequently, for actual computer processing, the subsystems must be (3) where jobs are (4) or executable units of processing. However, the systematization job flow chart does not always show how (5) files are connected to one another (6). The (7) that are extracted in (8) must therefore be divided to clear relationships among components, with regard to computer processing in the work of designing system functions.

a. physical
b. partitioned into components
c. people
d. analytical process
e. clearly
f. component functions
g. computers
h. program
i. clarify

3. From the group of answers listed below, select the correct definition for each of the following descriptions of partitioning into components.

(1) Consideration of whether complexity can be reduced and development can be made easier by partitioning into components

(2) Consideration of whether the occurrence of failures can be distributed by partitioning into components

(3) Consideration of whether similar functions can be handled in the same way in the processing of jobs

(4) Consideration of processing modes such as batch, online, transaction, and file transfer

Answers

a. Partitioning by job functions

b. Partitioning by processing time and reliability

c. Partitioning by development efficiency

d. Partitioning by processing modes

4. During partitioning for online processing, some are classified by processing patterns. From the group of answers listed below, select the correct definition for each of the following descriptions.

(1) Checks input information and outputs its contents as data into a file

(2) Replaces, deletes, or adds to the contents of the master file, depending on the contents of input information

(3) Collates input information with the contents of the master file, and the results are displayed on the screen

(4) Creates a list based on input information

Answers

a. Collation processing

b. Data entry processing

c. Update processing

d. Slip issuance processing

5. During partitioning of batch processing, it is sometimes classified by the processing pattern. From the group of answers listed below, select the correct definition for each of the following descriptions.

(1) Creates a listing from the input file

(2) Replace, delete and add operations to the contents of the old master file, depending on the contents of the input data file, creates a new master file. The update history is written to the file at the same time.

(3) Input data file A is collated with the master file. If the input data matches, it is written to file B. If the input data does not match, it is written to file C.

(4) Checks the contents of input data file A. If they are correct, they are written to file B. If they contain an error, they are written to file C.

Answers

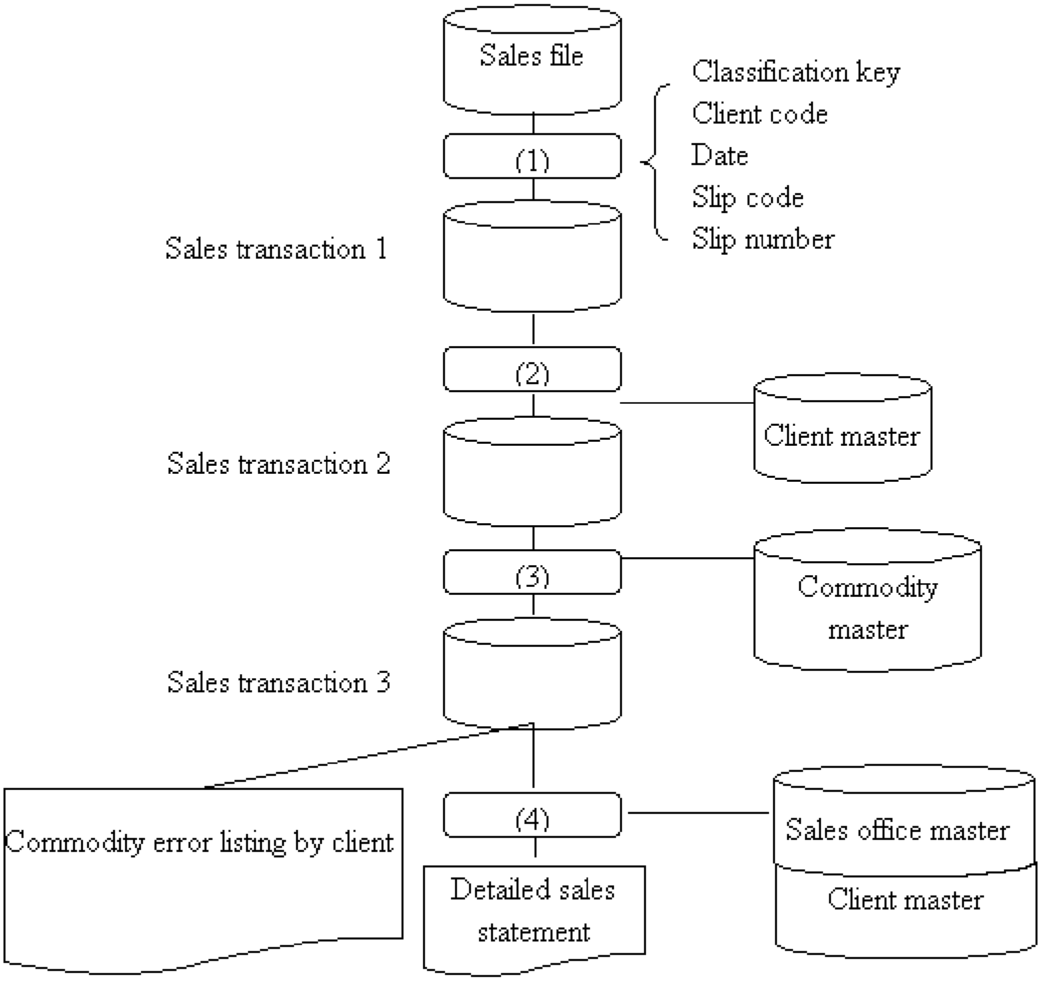
a. Input check system

b. Collation system

c. Update system

d. Report creation system

6. From the group of answers listed below, select the correct words to fill in the blanks in the following chart on program partitioning for batch processing.



Answers

a. Check of a client's contact person
b. Detailed sales statement creation
c. Sorting
d. Commodity check

7. The following chart summarizes the relationship between the input/output screens and tasks for interactive online processing. From the group of answers listed below, select the correct words to fill in the blanks in the following chart.

	Responsiveness	Productivity	Reliability	Memory
Multiple screens and single task	(1)	(2)	Recovery processing is (3)	(4) than that for multiple screen and multiple task processing
Multiple screens and multiple tasks	(5)	(6)	Recovery processing is (7)	(8) than that for multiple screen and single task processing

Answers

a. Complicated

b. Poor

c. Less

d. Greater

e. Easy

f. Excellent

8. When the process function is determined, the process function, input-output data, input source, and output destination are handled as the "Process function definition." The following are descriptions of the process definition. Mark a circle (O) next to a correct description. Mark an x (X) next to an incorrect description.

(1) In the process definition, emphasis is placed on "How to" create rather than "What" to create.

(2) Subsequent system structure designs are performed based on this process function definition.

(3) Functions of the process are described in the format of "noun + verb".

(4) The section and department in charge of input and output are not specifically required in the process definition.

9. The following sentences describe the “Completeness” of , one of considerations on determining component functional specifications. From the group of answers listed below, select the correct words/phrases to fill in the blanks.

The processing contents of partitioned (1) must be sufficient to provide the contents defined for (2). To achieve (3), (4) during external design are often (5) during internal design.

Answers

a. the basic function

b. compensated

c. functional omissions

d. components

e. completeness

10. The following descriptions are considerations on determining the component functional specifications. From the group of answers listed below, select the correct definitions for each of them.

(1) Ease of operation in the practical use after implementation must be considered for the program to be created.

(2) In case the flows of data and control in a program can be easily understood, development productivity, operability, and maintainability will increase. For this purpose, it is important that the interfaces between components are simple and the flows of data and control are easy to understand.

(3) Reuse of software is a very useful, because it leads to not only reduced development and maintenance costs, but also to improvements in quality. Therefore, specifications and partitioning must be determined with due consideration to reusability.

(4) The time required for operation and maintenance may be longer than that required for development, and maintenance costs become greater than development costs. Therefore, the specifications must be determined considering the ease of maintenance.

(5) Productivity of development can be improved by partitioning complicated processing into separate programs. The specifications should be specified not to include too complicated processing into a single component.

(6) The specifications on requirements definitions and processing capacity that were agreed on during external design must be satisfied.

Answers

a. Understandability

b. Productivity of development

c. Operability

d. Processing capacity

e. Maintainability

f. Reusability

3 Input-output Design

Chapter Objectives

This chapter explains the report design procedures and details in the input-output design process. It also explains how to include data check in GUI design as well as message design. This chapter is intended to provide a grasp of input-output design with consideration to report and screen characteristics.

3.1 Report design
3.2 GUI design

3.1 Report design

Input-output design has two parts: design at the item level and design at the layout level. Design at the item level consists of designing detailed attributes of and the correlation between data items for reports and screen objects. All data items for input from the screen and report output must be listed.

The following steps are taken for input-output design (at the item level):

1. Design of output data items
2. Design of input data items

(1) Design of output data items

First, the procedures for designing an output report are examined. There are three steps in the procedure for the design of output data items:

1. Confirming the purpose of the report.
2. Confirming the output device
3. Determining the output data items

The details of each step are explained as followings:

1) Confirming the purpose of the report and considering the selection of printing paper

Different paper types must be selected depending on the kind of jobs, how reports are used, and whether reports are provided for customers or for internal use. For reports to customers, special paper forms with pre-printed frames and titles are usually used, since readability is considered most important. In contrast, the cost factor is important for reports intended only for internal use, so general-purpose paper that is not pre-printed is used. The other type of paper is form overlay output report paper. The features of these paper types are as follows:

Item	Cost	Paper replacement	Readability
Special paper	High	Required	High
General-purpose paper	Low	Not required	Poor
Form overlay output report paper	Low	Not required	High

2) Confirming the output device

While the type of output device and output media are selected during the planning process, selection requirements are reviewed in this step.

Review them with performance requirements in mind since selection of an output device is closely related to the purpose of reports.

In selecting an output device, note the following points:

- Are reports to be printed with the device?
  To temporarily refer to information, which is a situation that does not require printing of the information, consider introducing display devices, so that users can quickly refer to the information.
- Is it necessary to print a large volume of data?
  If it is necessary to print a large volume of data, consider introducing high-speed printers. Estimate the amount of output time according to the volume of output data, and confirm whether printing can be completed within the time specified in the operating requirements. Note that the data for peak times and the peak month is used for estimating the output time.
- Does only a small volume of data have to be printed?
  If only a small volume of data has to be printed, a high-speed printer is not required.
- Is cut-sheet paper to be used?
  If cut-sheet paper is used, consider introducing printers with inserters capable of automatic feeding, or cut sheet printers.
- Is tracing (copying) to be done?
  If data is printed and traced on multi-part forms, such as payment slips, a line printer is required. Without an impact system such as a wire dot system, traces on multi-part forms cannot be done.
- Is special paper to be used?
  To print reports without using expensive special paper, consider using the form overlay function for printing the frames of the reports together with data.
- Is special data to be printed?
  Printing of image data (e.g., maps), figures (e.g., lines, circles, arcs, painted figures), barcodes, and OCR characters requires a line printer for cut-sheet paper or a multi-purpose printer. There are several types of barcodes, including NW7, JAN, Code 39, and Interleaved 2 out of 5. Check whether the printer can print the type of barcodes to be used.
- Is printing to be in color?
  A color printer is required for printing in color.

After confirming that the correct type of printer is selected, review the hardware functions.

- Are the maximum number of printed characters per line, character size, and character spacing settings appropriate?
  The character size, font face, and character spacing vary depending on the type of printer, and their values determine the maximum number of printed characters per line. Therefore, it is necessary to consider whether the printer is capable of printing the desired reports.
- Are the printable areas of satisfactory sizes?
  If existing reports require printing in non-printing areas, check whether such printing is also supported with the printer and printing paper in the new system. For example, some types of line printers do not start printing unless paper is pressed against their photoconductive drums. Therefore, the perforated parts of fanfold paper are regarded as non-printing areas.

- Is the printing system appropriate?
  If a printer is noisy during printing, it may not be appropriate for use in an office. For example, to use a dot impact printer, consider using a soundproof hood, or reconsider the printer model or the installation location.

- Can the desired type of cut-sheet paper be used?
  a. Size of paper (A3, A4, A5, B4, B5, letter size)
  b. Type of paper (regular paper, mail stickers, labels on backing sheets)
  c. Number of traceable sheets when multi-part forms are used

Confirm that printing is supported by software
If image, figures, or graphs are printed, confirm that such printing is supported by software (control program and application software). It is necessary to check the software, since the size of printed characters and fonts vary depending on the type of software.

3) Determining the output data items
- Determine the output items
  Review the output items and details based on the " Input-Output Study Sheet" created during the planning process. Confirm the new data items required and data items to be changed for use with the new system, and list the data items for printed reports.
- Determine the attributes (basic attributes) and the number of characters per line of printed items, and add a description for each item.
  Examples for the design of output items are listed below.

(Examples)

Data item	Attribute	Size	Description
		Integer	
Sales code	Numeral	2	Sales office codes are outputted in ascending order.
Sale office name	Character	5	Each sales office name is outputted in five characters.
Customer code	Alphanumeric character	12	Customer codes are outputted in ascending order.
Customer name	Character	12	Each customer name is outputted in twelve characters.
Slip number	Alphanumeric character	6	Slip numbers are outputted in ascending order.
Date	Date	8	Dates are outputted in four digits for the year and four digits in the month and day, per slip.
Product code	Numeral	6	Product codes are outputted in ascending order.
Product name	Character	12	Each product name is outputted with twelve characters.
Quantity	Quantity	3	Quantity of sales by product is outputted.
Unit price	Amount	7	Unit prices by product are outputted.
Amount	Amount	10	Unit price x quantity is outputted.
Person in charge	Alphanumeric character	2	Name of person in charge, depending on the slip, is outputted.
Slip total	Amount	10	Totals per slip are outputted.
Total for a customer	Amount	11	Totals by customer are outputted.
Total for a sales office	Amount	11	Totals by sales office are outputted.
Grand total	Amount	11	Grand total of a sales statement is outputted.

- Checkpoints for output items (review with end users)

a. Are any data items missing? Are they really necessary?
b. Are there any errors concerning the attributes of data items?
(e.g., data type, number of characters per line, editing format, decimal point)
c. Is there any misinterpretation of data items?
(e.g., Is a price the selling price or the buying price?)
d. Does hardware support data items? (e.g., barcode)
e. Can barcode printing be read?
(Create a test print of a barcode, and read it with the input device to be actually used.)

(2) Design of input data items

This section reviews details of how to select input devices and input media in the planning process. And then input items are defined.

1. Selection of input media, input devices, and input methods
2. Design of input items

1) Selection of input media, input devices, and input methods

It is important to choose input methods that are easy for users and are less likely to cause input errors in daily jobs.
If few errors remain in input data and relatively clean data is obtained, the subsequent processes are quite different. Therefore, select input media, input devices, and input methods with consideration of those who create input data (i.e., Are they dedicated operators, part-time employees, general staff, or average consumers?), their skill level, input data volumes, and the data correction method in the event of errors.

The following table summarizes what considerations should be made for such selections:

<Criteria for selecting media for input and output >

Media	Criteria for selection	Features
Keyboard Ten-key board Mouse	Suitable for real-time processing systems (when processing results are required immediately at data source)	- Data can be directly inputted on the spot
Optical mark reader	Suitable for input of a few items	- Not all characters, including certain alphanumeric characters, can be expressed.
Optical character recognition unit		- Coding work on data sheets is not required. - Keypunchers and key punching devices are not required. - The time from data creation to input is significantly reduced. - The work of creating data can be performed in different places, because the cards can be easily filled out on the spot.
		-There are restrictions with regard to character patterns. -Recognition errors and rejected characters must be corrected.
Handy terminal	Suitable for real-time processing systems whose use is not restricted by location	- Such devices are portable because they are small. - Accumulated data can subsequently be processed on a PC.
Touch panel	Suitable for real-time processing systems Suitable for operation by beginners	- Keys are arranged alphabetically. - Different data items can be inputted for each key by replacing the master sheet.
Image scanner	Suitable for image input (scanning)	- Can read a book which has thickness. - Can read halftone materials such as photographs
Barcode reader	Suitable for instantaneous input (reading) of many kinds of data items	- Data can be read simply by touching the barcode with the scanner part. - There are restrictions on the width and kinds of readable barcodes.

There are other input methods, such as input with pen and voice input.

Several input methods have recently been becoming popular, and they are explained below.

- Input with the graphics functions and mouse of a PC

  By using the graphics functions of a PC, precise and multicolor images can be displayed, and processing speeds can increase. This can significantly simplify PC operations.

  For example, if icons are created with images that are easily associated with specific instructions for a PC, processing can easily be selected by clicking the icons. It is not necessary to input characters directly or look at the keyboard to select a function key. Processing can be selected using the mouse, while looking only at the icon on the screen.

- Input with a barcode reader

  Recently, many products are being handled with coded information, which has spread to many areas, including food and clothing, books, home appliances, and other daily-use products. Codes are printed as barcodes on such products.

  There are several kinds of barcode readers: a light pen type, a handy terminal type, and a type built into registers, such as those seen in supermarkets. Barcodes can be read using different types of barcode readers designed for different purposes. With barcode readers, even less experienced operators can read long codes correctly and instantaneously.

  Product code input at supermarkets is a good example of this system in operation. The barcode system in POS register systems is commonly used in the sales management field. An example of such use is inventory control with data read using barcode readers.

- Input with a handy terminal

  Handy terminals are suitable for inputting data immediately at individual sales transactions that take place in more than one location.

  A handy terminal is portable and can be carried anywhere, since it is the size of a pocket book. Data can be inputted immediately after products are sold, and the data is accumulated in the device. By connecting the handy terminal to an optical adapter, the accumulated data can be quickly transferred to a PC. A large volume of data can be processed collectively on the PC.

  This system is effective in a large restaurant, for example, where orders can be received from many tables. Another example is order management, where salespeople located throughout a large area receive product orders.

2) Design of input items

- Determining input items

  Select the minimum data set for input. Data does not have to be input if it can be retrieved from master files. For this data, input only for key items in order to minimize the input work.

- Determining attributes and number of characters per line of input items

  Use the same method as that for determining output data items.

Examples for the design of input items are listed here.

Data item	Attribute	Size	Description
		Integer	
Sales code	Numerals	2	Sales office codes are inputted.
Sale office name	Characters	5	Each sales office name is outputted in five characters.
Customer code	Alphanumeric characters	12	Customer codes are inputted.
Customer name	Characters	12	Each customer name is inputted in twelve characters.
Payment slip number	Alphanumeric characters	6	Input the slip number.
Product code	Numerals	6	Product codes are inputted.
Product name	Characters	12	Each product name is inputted in twelve characters.
Quantity	Quantity	3	Quantity of sales by product is inputted.
Unit price	Amount	7	Unit prices by product are inputted.
Person in charge	Alphanumeric characters	2	Name of person in charge, depending on the slip, is inputted.

The date, amount, slip total, customer total, sales office total and grand total are automatically calculated.

(3) Design of report layout

This section explains how to design detailed reports for input and output.

Take design activities from a user's perspective and work with staff members who are in charge of jobs when the input and output formats are defined. This is important to avoid the use of different formats resulting from misinterpretations and so that design work can be performed efficiently.

Taking into consideration readability, usability, and standardization of formats, define the input and output formats and determine the editing requirements for reports in detail.

The following steps are taken for layout design:

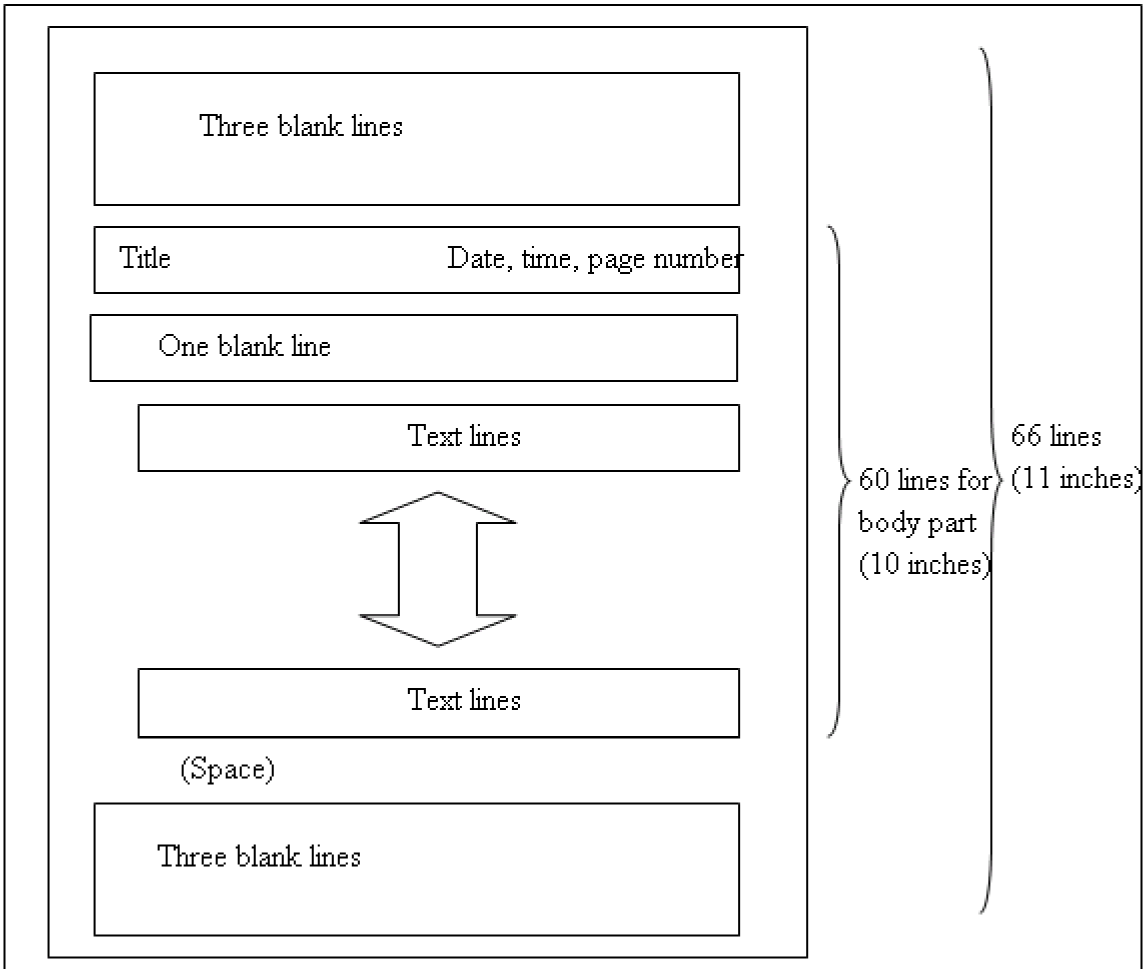
1.□ Design of the report outline
2.□ Design of report details
3.□ Review

1) Design of the report outline

a. Standardize the output format

The most important factor for reports is that they are easy to read and understand. If standardization is not taken into consideration in the design, the total system will be disorganized and difficult to use. For that reason, standardize the output formats of the reports (e.g., titles, headers, organizations of details). An example report layout for a sheet of paper from a line printer is given below:

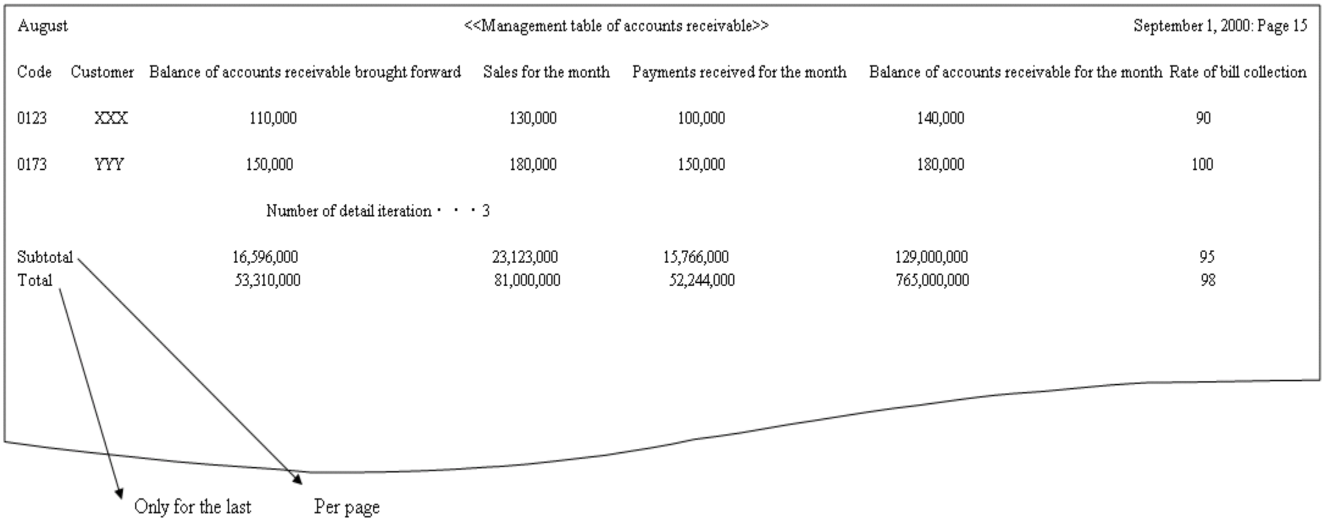
<Example page layout for sheet of paper from a line printer sheet>



Although the page number is placed at the bottom for handwritten materials, it is printed at the beginning for output with a line printer. Also the time is printed to distinguish each printed page, since it is possible to print the same page many times in one day. (Usually, the date and time when the program executed are printed).
Follow the rules below about printing from a computer.
1) Do not create pages with only spaces and no body text.
2) Change to a new page before printing the first line.
3) Leave the paper without changing to a new page after printing the last line of the body part.

b. Design the outline of the output report layout.

Taking the standardization items in previous section into consideration, create an outline of the layout based on the output data items.



August <<Management table of accounts receivable>> September 1, 2000: Page 15

Code	Customer	Balance of accounts receivable brought forward	Sales for the month	Payments received for the month	Balance of accounts receivable for the month	Rate of bill collection
0123	XXX	110,000	130,000	100,000	140,000	90
0173	YYY	150,000	180,000	150,000	180,000	100
		Number of detail iteration・・・3				
Subtotal		16,596,000	23,123,000	15,766,000	129,000,000	95
Total		53,310,000	81,000,000	52,244,000	765,000,000	98

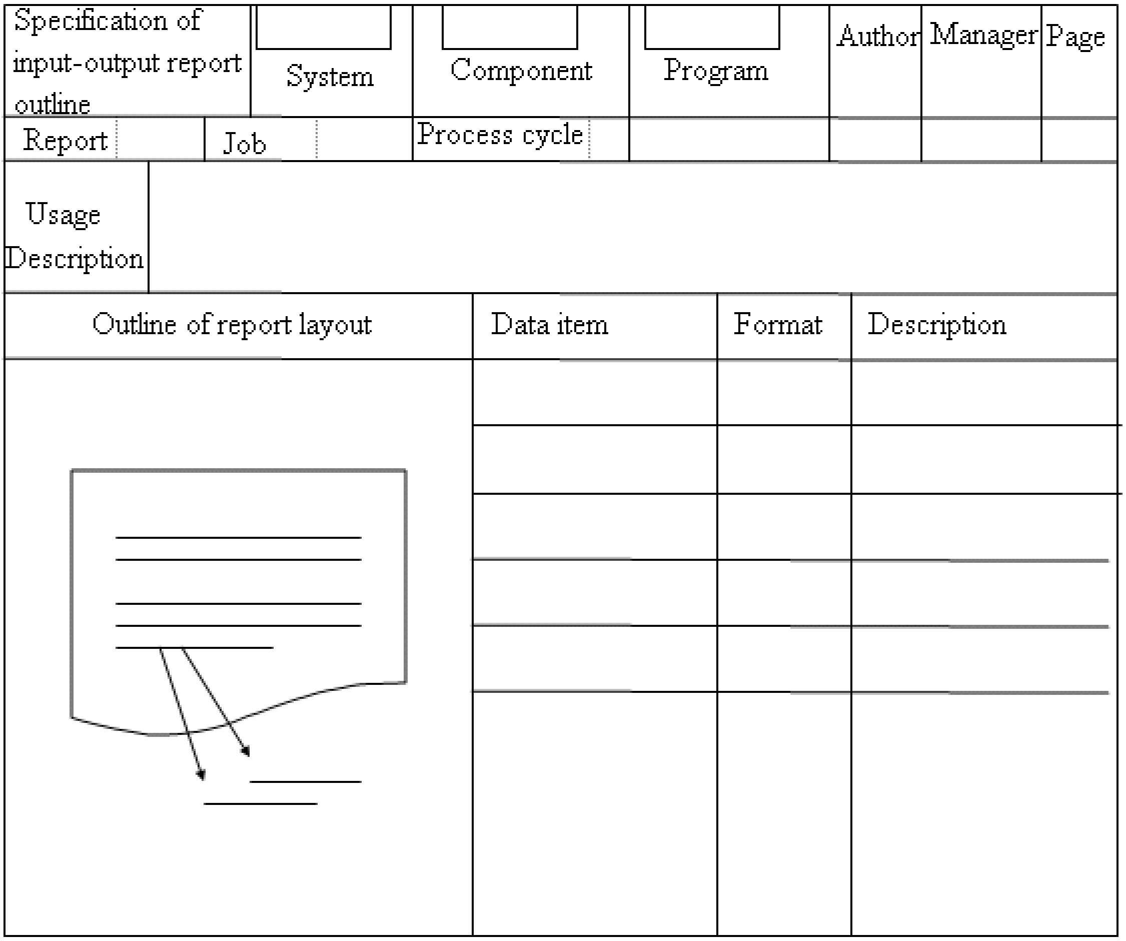
c. Confirm the estimated volume of output data and the processing cycle

The estimated volume of output data (pages, sheets, cases) and the processing cycle are closely related to the operating schedule of the system. Since it is basic material for determining required hardware performance, review it in the user interface design process as well as in the planning process.

d. Create a specification of the output report outline.

Summarize all of the items determined in a specification of the output report outline. Normally, specification details are entered onto the following sheet of paper at each step during the design of the output report outline.

<Example specification of output report outline>



Specification of input-output report outline	System	Component	Program	Author	Manager	Page
Report	Job	Process cycle				
Usage Description						
Outline of report layout	Data item	Format	Description			

2) User review

After finishing the specification of the output report outline, work in cooperation with user organizations to confirm that there is no problem in the layout.

3) Design of report details

Using the document, create a detailed output layout down to the level of program specifications. Note the following two points:

a. Identifying the requirements for output processing.

The following requirements must be determined for output processing particular to this stage:

- Number of output lines per page.
- Processing requirements for page feed.
- Processing requirements when the sort key is changed
- Processing requirements when output data is continued to the following page
- Processing requirements for the total by sort key, grand total, etc.
- Processing requirements for data with zero occurrence

b. Matching the processing sequence of input data with that of output data

A basic principle is to match the processing sequence of data for output reports (output sequence) with the processing sequence of input data (input sequence).

Generally, edit the data so that output data items are horizontally placed, one by one, next to the corresponding input data item. If such data is vertically edited and outputted, it is necessary to decide whether data is outputted after one page of data is accumulated in memory or whether it is outputted consecutively after the final line has been reached. From the viewpoint of the program development efficiency, it is best to discuss the editing format of horizontal output with persons in charge of the respective jobs.

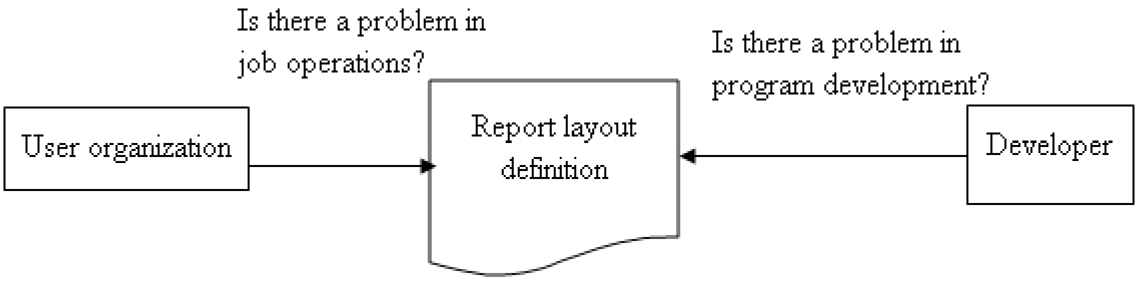
Considering the output processing requirements and sequence discussed, define the report layout in detail. For users, such a definition of the report layout is designed to confirm the specific appearance of the report, and for programmers, it serves as a specification for program design. Therefore, this report layout definition must include details, including the output position, number of characters per line, and editing format.

Take care concerning the following points, when creating the report layout definition.

- Explain data items and details of output items, if they are difficult to understand without an explanation.
- Make sure that output processing requirements are specified.
- Be sure to place the listing codes in the code specification definitions, since the listing codes is occasionally printed on the report to identify the type of listing.

4) Review

The report layout definition serves as an external specification. Work to achieve a consensus about the report specifications between user organization and the development side.


Is there a problem in job operations?
Is there a problem in program development?
User organization
Report layout definition
Developer

3.2 GUI design

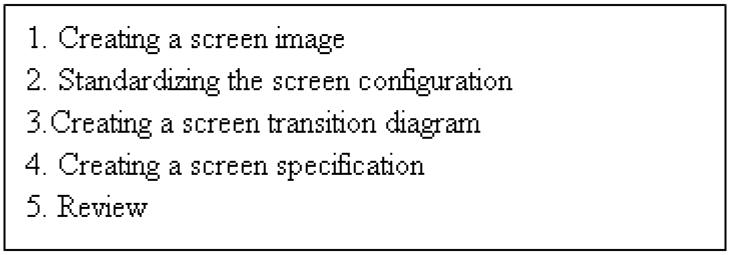
3.2.1 Screen design

The screen display is very important, as it serves as a human machine interface (HMI) for linking persons in charge of individual jobs with computers in the transaction process.

Therefore, most screen images created with the screen development tool are presented to system users. The prototyping method for creating screen images while they are being evaluated by users is also used for screen design.

In case of designing applications to run on an operating system such as Windows which uses a graphical user interface (GUI), a number of processing aspects should be taken into consideration for many screens (e.g., menu, input-output, error display, help). At the same time consideration should be given to such aspects as the data input sequence, checking of input data, and selection and execution of the processing functions.

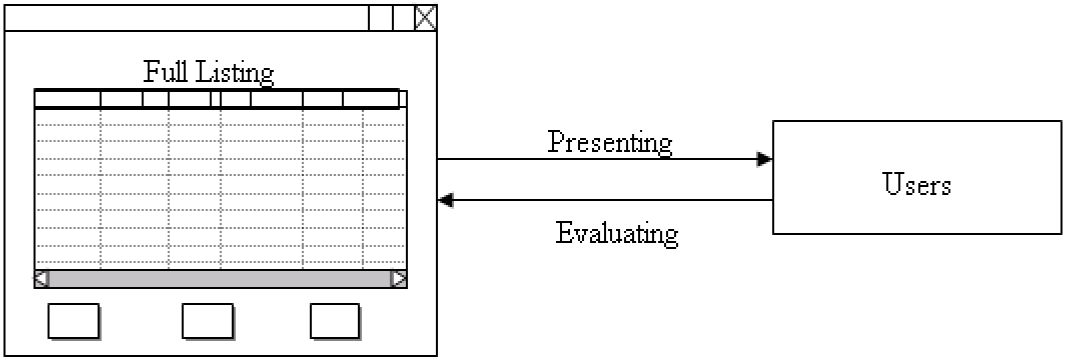
The following steps are taken in layout design:



1. Creating a screen image
2. Standardizing the screen configuration
3. Creating a screen transition diagram
4. Creating a screen specification
5. Review

(1) Creating a screen image

Create a screen image based on the input and output items identified in input-output design (at the item level), and ask users to evaluate it. Without specifying detailed settings at this point, present rough designs and actual screen images created with screen design tools.



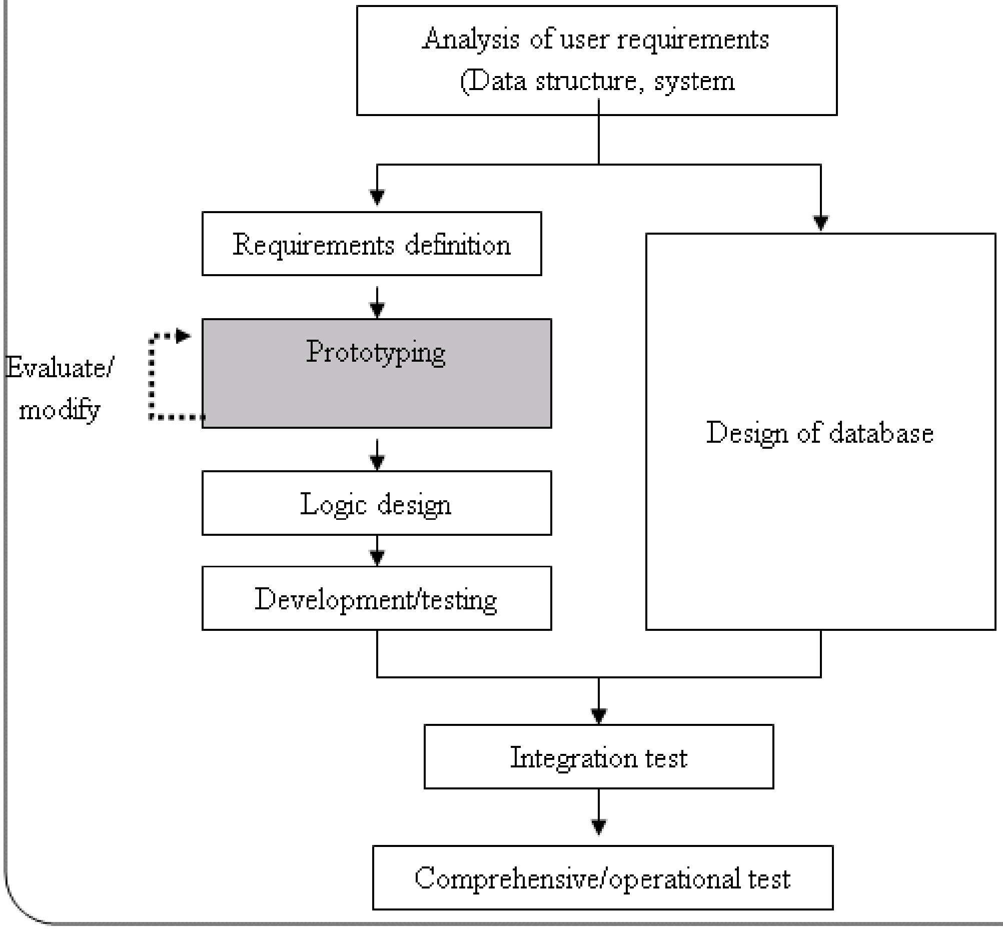
GUI design using the prototyping

The operability and ease of viewing are difficult to check using only a specification on paper in developing a new job system, and users may submit requests for specification changes during the testing stage after the system is created.

If a number of changes are made in specifications at this stage, the result may be a degradation of quality.

To prevent this, a prototype should be created as early in the process as possible. When looking at the prototype, users in the related divisions can readily describe points for improvement, and they can accurately convey to programmers what kind of information is lacking in the current design.

Since the prototype is created as only an experimental model, it is convenient to use support tools to create the prototype easily and quickly. Generally, the screen design tools provided with the program development environment such as Visual Basic, as well as support tools available on the market, are used for creating prototypes. After prototypes are evaluated by users, they are used as templates for the finished products.



(2) Standardizing the screen configuration

Compared to the previous character user interface (CUI), a GUI environment, such as Windows, offers a graphical and easy-to-use interface. This is one of the features of GUI applications. However, because of too much freedom provided, operating screens tend to be disorganized and difficult to use, unless a number of points are taken into consideration for standardization.

Pay attention to the following points in standardizing the screen configuration.

1) Display

- Physical size, resolution, and number of colors supported by displays

2) Screen (divided into displayed objects called windows)

- Location of standard buttons (e.g., OK, Cancel, Register, Search)
- Display location of messages, etc.
- Display of screen title and menus
- Consistency in expression of alphanumeric characters
- Expression of sentences and detailed items
- Color coordination

3) Control

- Style, size, color, and characters displayed
- Input check process
- Sequence of moving the focus (e.g., defining the tab sequence)

4) Menu

- Design menus with consideration of the standard specification (common client area) of the screen

5) Direct input from a keyboard

- Maintain consistency in the assignment of shortcut keys

6) Messages

- Determine how messages are displayed when a time-consuming process is executed (busy).

7) Error

- Execute standardized processing if an error occurs

8) Help

- Develop detailed Help information in accordance with the manual, and maintain consistency in terminology, descriptions, and explanations of methods.

.

(3) Procedures for creating a screen transition diagram

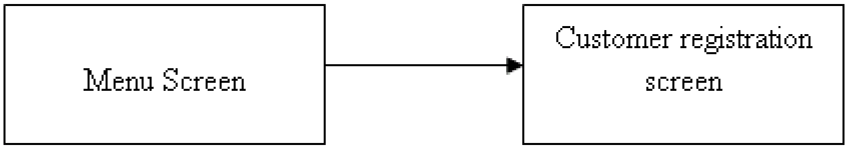
Once an outline of the screen is completed, summarize the correlation of screens in the screen transition diagram. Particularly with GUI-based applications, the screen transition diagram becomes more complicated than that for CUI-based applications. This is because a number of event processes are required, such as for "the button was clicked" and "the window size was changed."

To create the screen transition diagram, take the following steps:

1) Classify the screens into the following four patterns by focusing on the transition pattern.

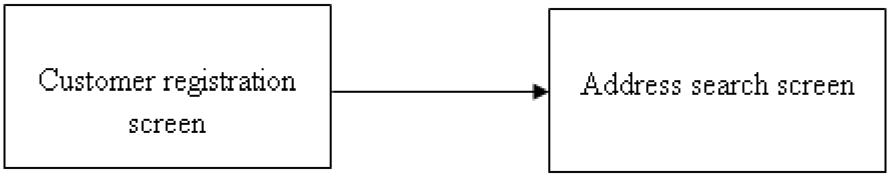
- Simple screen transition

A conventional simple transition.



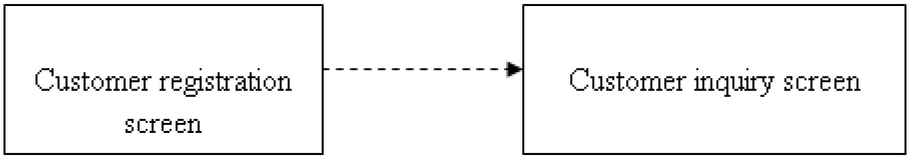
- Transition to a child screen

Move to a pop-up screen. When a child screen is displayed on the parent screen, the underlying parent screen cannot be operated.



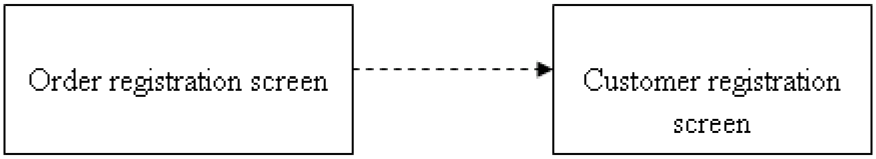
- Transition to an independent child screen

Transition to an independent child screen is to move to a pop-up screen in the same way as for the transition to a child screen explained on the previous page. Unlike a transition to a child screen, the parent screen and other screens can be operated while the child screen is displayed.

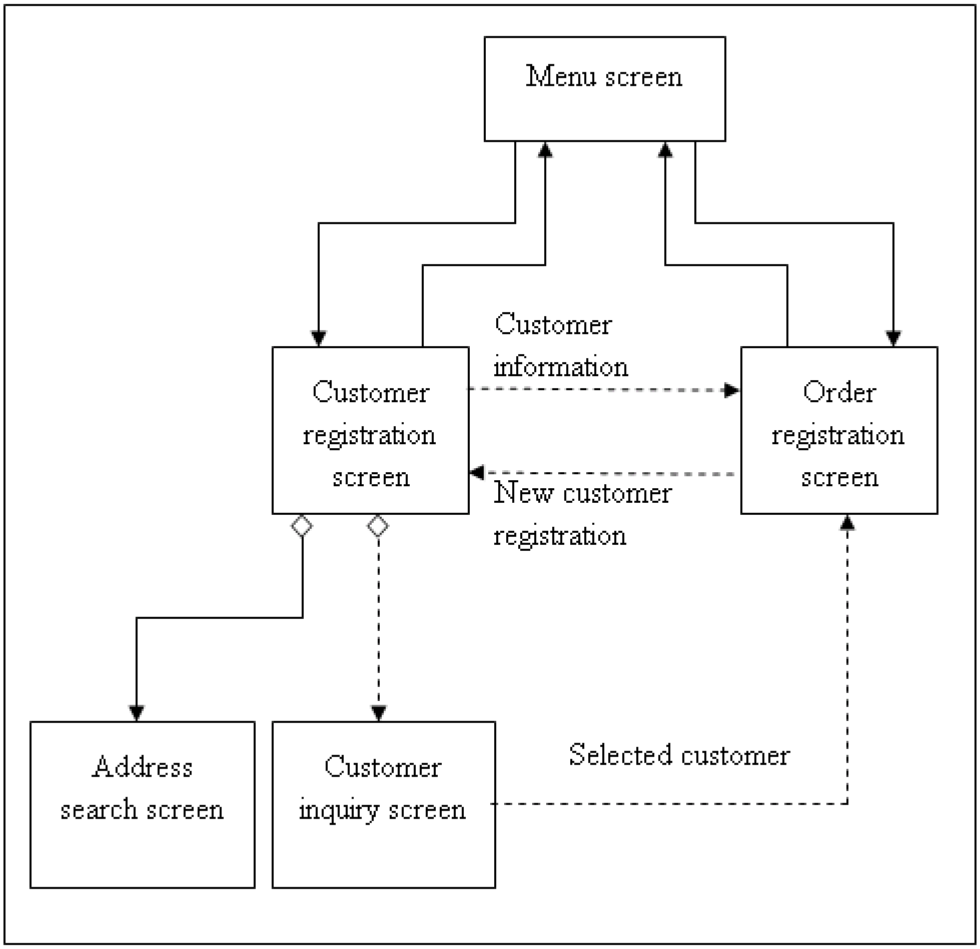


- Transition to an independent screen

Start an independent new screen.



2) Link the screens in accordance with the classifications of step 1)
Specify the requirements (events) for transitions on the line between screens in the diagram. An example of a screen transition diagram is given here.



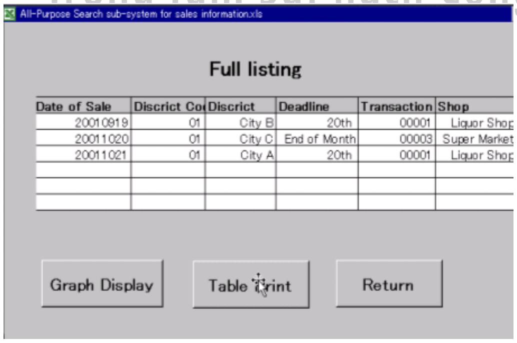
An example of a screen transition diagram

(4) Creating a screen specification

Decide on a detailed format for a screen specification, and define field attributes based on the new screen information identified while deciding on screen images and the screen transition diagram.

An example screen specification is as follows:

<table>
<tr><td colspan="2">Liquor sales basic system
(general-purpose search subsystem for sales information)</td><td>Date of creation</td><td>Approved by</td><td>Reviewed by</td><td>Person in charge</td></tr>
<tr><td>Screen specification</td><td>Displaying detail table</td><td></td><td></td><td></td><td></td></tr>
</table>

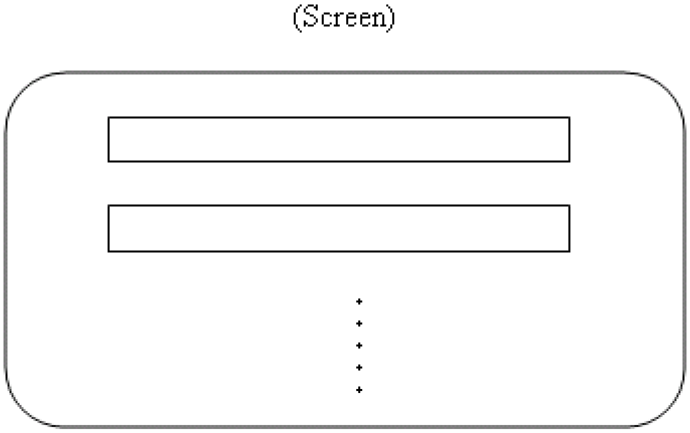
<table>
<tr><td rowspan="5">
All-Purpose Search sub-system for sales information.xls
Full listing
<table>
<tr><th>Date of Sale</th><th>Discrict Co</th><th>Discrict</th><th>Deadline</th><th>Transaction</th><th>Shop</th></tr>
<tr><td>20010919</td><td>01</td><td>City B</td><td>20th</td><td>00001</td><td>Liquor Shop</td></tr>
<tr><td>20011020</td><td>01</td><td>City C</td><td>End of Month</td><td>00003</td><td>Super Market</td></tr>
<tr><td>20011021</td><td>01</td><td>City A</td><td>20th</td><td>00001</td><td>Liquor Shop</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>Control</td><td>Operation</td><td>Function</td></tr>
<tr><td>Area for displaying detail table</td><td>Initial</td><td>-Displays in a table information meeting the conditions defined in the search specification screen.
-This follows the setting specified in the display settings screen for display items and sequence of display.</td></tr>
<tr><td>Graph display button</td><td>Click</td><td>Displays the graph display screen</td></tr>
<tr><td>Table print button</td><td>Click</td><td>Displays the print preview screen</td></tr>
<tr><td>Return button</td><td>Click</td><td>Displays the search specification screen</td></tr>
</table>

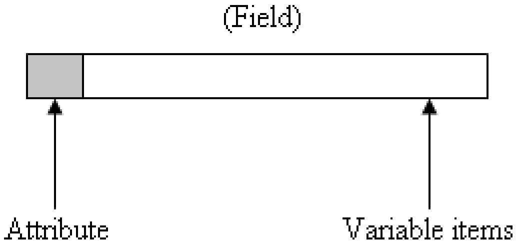
<Example of Screen Specification Design)

a. Screen image
This is the screen image to be displayed. If screen images are created in advance with the screen design tool, attach a hardcopy.

b. List of functions
Defines the names of parts such as the buttons on the screen, and summarizes their functions. Provide descriptions of events for individual screens, attributes of parts, input check specifications and output specifications, etc.

c. Defining the field attributes
Decide on the field attributes of input and output items, and summarize them in descriptions of items for screen display.
The screen consists of multiple fields. Each field consists of a one-byte (equivalent to a single character) attribute at the beginning and a variable item.





The criteria used for defining field attributes are as follows:

- Color and brightness

Seven colors can be set for displayed color: green, white, red, blue, purple, light blue, and yellow.

Arrange the color in accordance with the degree of importance of the item, such as "red" for error and "yellow" for alert.

For better readability, make sure there is a large difference between the brightness of characters and that of the background.

- Display suppression

This can be specified to suppress display of input content on the screen. It is used to enter passwords, which should not be seen by third parties.

- Blink

This is used to call attention to an item, such as error items.

- Reverse

This is used to emphasize a field and identify an area.

- Ruled lines

This is used to frame items.

A vertical ruled line is displayed in one column. Since an attribute before a variable item (one-character wide) shares the same column with a vertical ruled line, only one column is sufficient to display the vertical ruled line with an attribute.- Enlarged character

This is used to enlarge characters on the screen to make them easier to see, or it is used for emphasis.

- Protection setting

This is specified for items that operators are not allowed to input.

- Transfer setting

This is specified for a field to be transferred to the host

Usually, transfer is not specified for fields set for protection. Also to reduce transfer volumes, transfer is not specified for data held by the host.

- Print suppression

This is set for a field that is not to be printed because the field information is not required on the hardcopy. For example, if the hardcopy is used as an official report, it is desirable to not print the normal end message.

The following table lists examples of the summary of items for screen display.

Example of items on display screen

Screen name	Order entry			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Transaction category	3	Numeral	Green (blink)	Error items blink.
Customer code	5	Numeral	Green (blink)	Error items blink.
Customer name	30	Character	White	15 characters, left-justified
Product code	8	Numeral	Green (blink)	Error items blink.
Product name	22	Character	White	11 characters, left-justified
Quantity	6	Numeral	Green (blink)	Error items blink.
Unit price	7	Numeral	White	
Amount	9	Numeral	White	
Quantity in stock	10	Numeral, special character	White	Displayed in the format of ZZZ, ZZZ, ZZ9

Define the common rules at the beginning of screen item descriptions.

The following is an example:

- Set "Reverse" to the title.
- Set "White" to the ruled lines.
- Set "Red" to error messages.

(5) Input from the screen

There are two ways to input data from the screen: menu input method and fill-in method.

1) Menu input method

Select one of the multiple options displayed on the screen by using a light pen or by typing in the relevant number.

2) Fill-in method

Use the keyboard to input data for each entry item on the screen.

(6) Standardized GUI components

With current GUIs for screen displays, interfaces and layouts are standardized, so that similar operations can be done for different.

The standard components are as follows:

1) Menu

- Menu bar

Displayed in the upper part of a window or screen, it can be used to select an item in the menu.

- Pull-down menu

When a certain item is selected from the menu bar, a menu whose content depends on the selected item is displayed.

- Pop-up menu

  This menu is displayed at the position of the pointer. The content of the menu depends on the display function of each button of the pointing device like a mouse.

2) Dialog box

- Dialog box

  It is used to prompt the user to respond or give instructions. This is also displayed when instructions are required from a user or a message has to be issued.

- Selection dialog

  Items are displayed for selection.

- Working dialog

If a noticeable length of time is required to complete a process, this displays the progress status.

- Message dialog

  When important information, such as an alert, is reported, this displays the information.

- Prompt dialog

  This is used for direct input from the keyboard.

3) Boxes for selecting items

- Checkbox

  Multiple items can be selected from multiple options.

- List box

  Options are listed so that an input item can be selected.

- Radio button

  One item can be chosen from multiple options provided by this type of box.

4) Buttons

- Command button

  Processing starts for an item when the corresponding button being displayed is clicked. Generally, it is displayed as a rectangular button.

- Toggle button

  When clicked, it switches between On and Off.

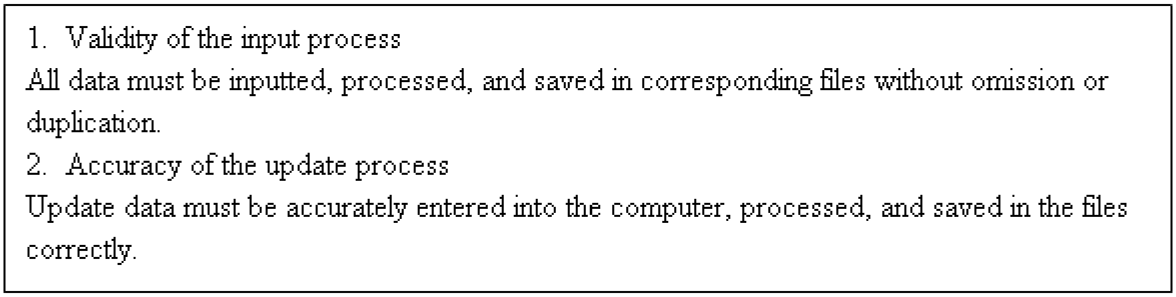
3.2.2 Input and output check methods and message design

This section explains how to check input and output data and design messages.

(1) Ensuring the validity of data

Ensuring that data is complete and consistent is an important factor in ensuring the reliability of the system. Specifically, it is necessary to establish the mechanism that prevents a loss or duplication of data in all processes, from approval in the data creation stage to input CPU processing, data in files, and output. Functions for preventing input errors, ease in verifying input results, and a prevention function against unauthorized acts are mandatory for the system. Traceability from file data and output to the data creation stage is also required.

The following two points must be taken into consideration to ensure the reliability of data:



1. Validity of the input process

All data must be inputted, processed, and saved in corresponding files without omission or duplication.

2. Accuracy of the update process

Update data must be accurately entered into the computer, processed, and saved in the files correctly.

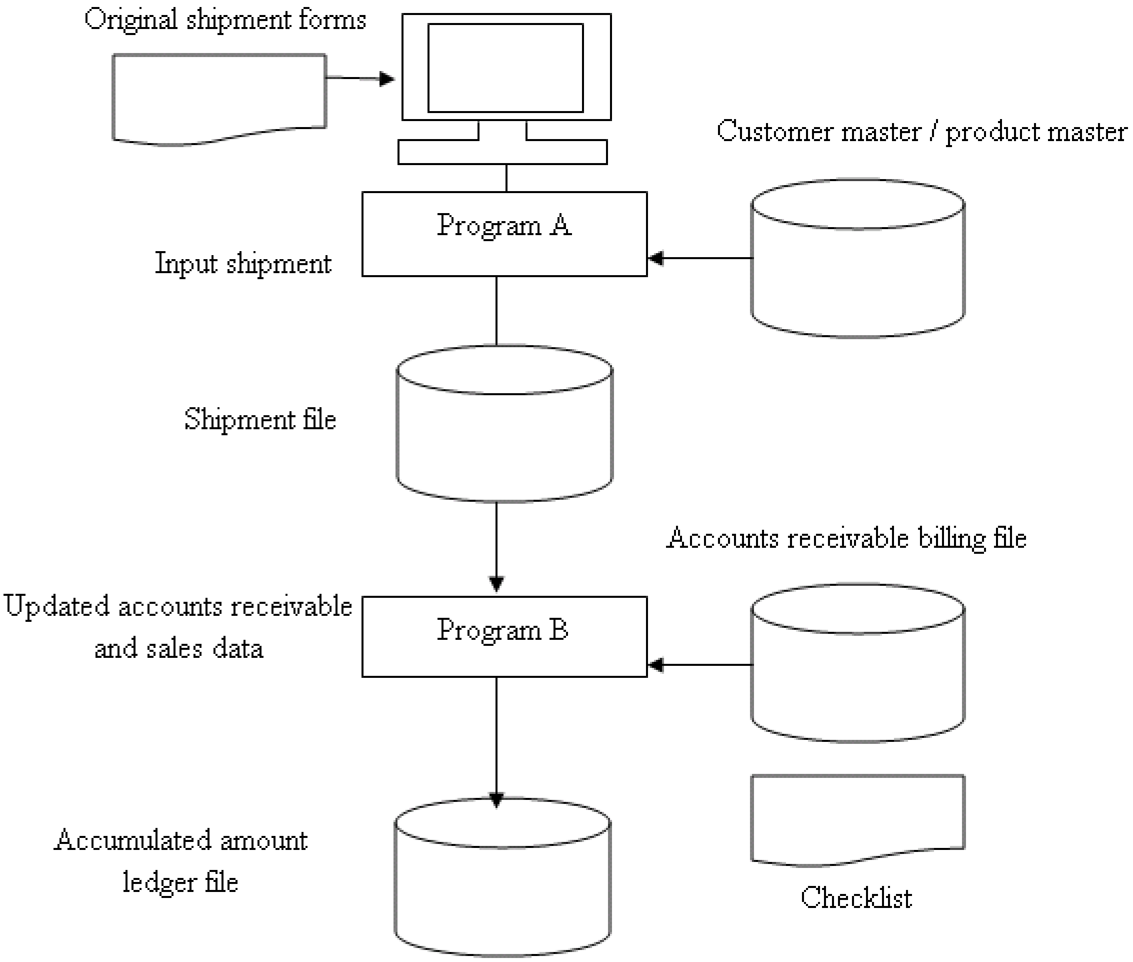
To meet these control objectives, data processing procedures by business applications (e.g., check) must be combined appropriately with a series of manual steps for input and output. In other words, it is essential to establish and operate a business system that not only performs computer processing, but also ensures the reliability of data.

An example of sales operation is given below. To operate a reliable sales business system, a system must be designed to capture the data in all original shipment forms correctly and reflect them in bills, ledgers, and accounts receivable.

The following paragraphs summarize the control objective and control method used in each stage.

1) Input process

An example of the input and output process flow is below.


Original shipment forms
Customer master / product master
Program A
Input shipment
Shipment file
Accounts receivable billing file
Updated accounts receivable and sales data
Program B
Accumulated amount ledger file
Checklist

<Example of the input and output process flow>

In a sales operation, the original forms are completed in the shipment process. In this system, terminals are used to input data into a computer. Program A checks sales data while managing the input process.

The following two points should be taken into consideration:

a. Have all sales transactions been inputted? (Completeness of input)

If data on the original forms is omitted during input or the same form data is inputted more than once, correct results cannot be obtained.

b. Has the sales transaction data been inputted correctly? (Accuracy of input)

Even if all original forms are inputted without omission, data from each original form must be checked to make sure data is inputted correctly.

For example, check if input data for an amount includes a character other than numerals.

It is necessary to ensure the completeness and accuracy of input at the input stage. To ensure the completeness of input, consider taking measures such as using batch totals to check input or checking every output report. To ensure the accuracy of input, take measures such as comparing different codes (e.g., customer code, product code) to those in the master table.

2) Update processing

In program B, sales amounts calculated in Program A is sorted by customer, and accounts receivable amounts are obtained. The program then writes the amounts of the accounts receivable and the amounts billed and update the ledger at the same time. For update processing, the completeness and accuracy of updates should be taken into consideration in the way as for input data.

One of the methods for ensuring the completeness of updates is to use a program to control the quantity (amount) of input data and the quantity (sum) of output data, and then check for any discrepancy between those values before and after the processing by programs.

With regard to the accuracy of update, there is a way to compare the total of the amounts of the data items (e.g., amounts of sales) to the totals in the updated listing.

(2) Accuracy of input data

The accuracy of input data means whether each input data item is correct or not. Typical checking methods are as explained below. Check the data using the program for data input processing.

1) Checking the data items

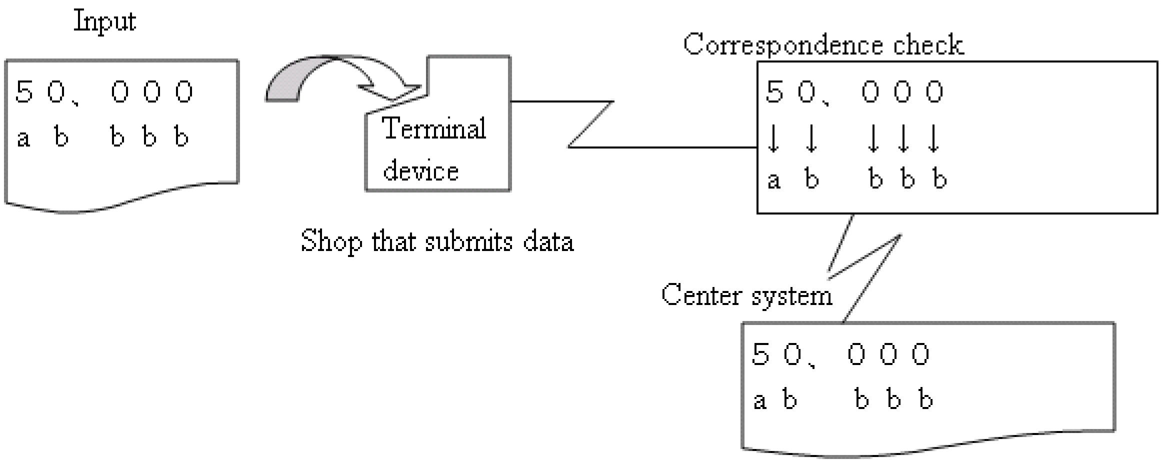
- Numeric check
  Check whether input data includes characters other than numerals in numeric items.
- Range check
  Check whether the input value is valid data by creating a table defining the allowable range of input values.
- Balance check
  Compare the results of calculations in case the value of multiple resultant data items are equal for the input values.
- Limit check
  Check whether the value of the numeric data is within the specified range.
- Combination check
  Check whether a combination is possible in case input data has correlations with other data input items.
- Format check
  Check whether the number of the digits of the input data item is consistent with the one specified for that item.
- Batch total check
  Calculate the total of a specific data item manually before computer processing. Then, calculate the total for the same item with a computer, and compare the two totals. By using this approach, it is possible to check if any data has been omitted in the batch.

2) Checking the value of an important item

If the value of an item requires particular accuracy (such as a amount of money), redundancy such as with a check digit and double input is provided to enhance accuracy.

a. Check by double input of an amount of money

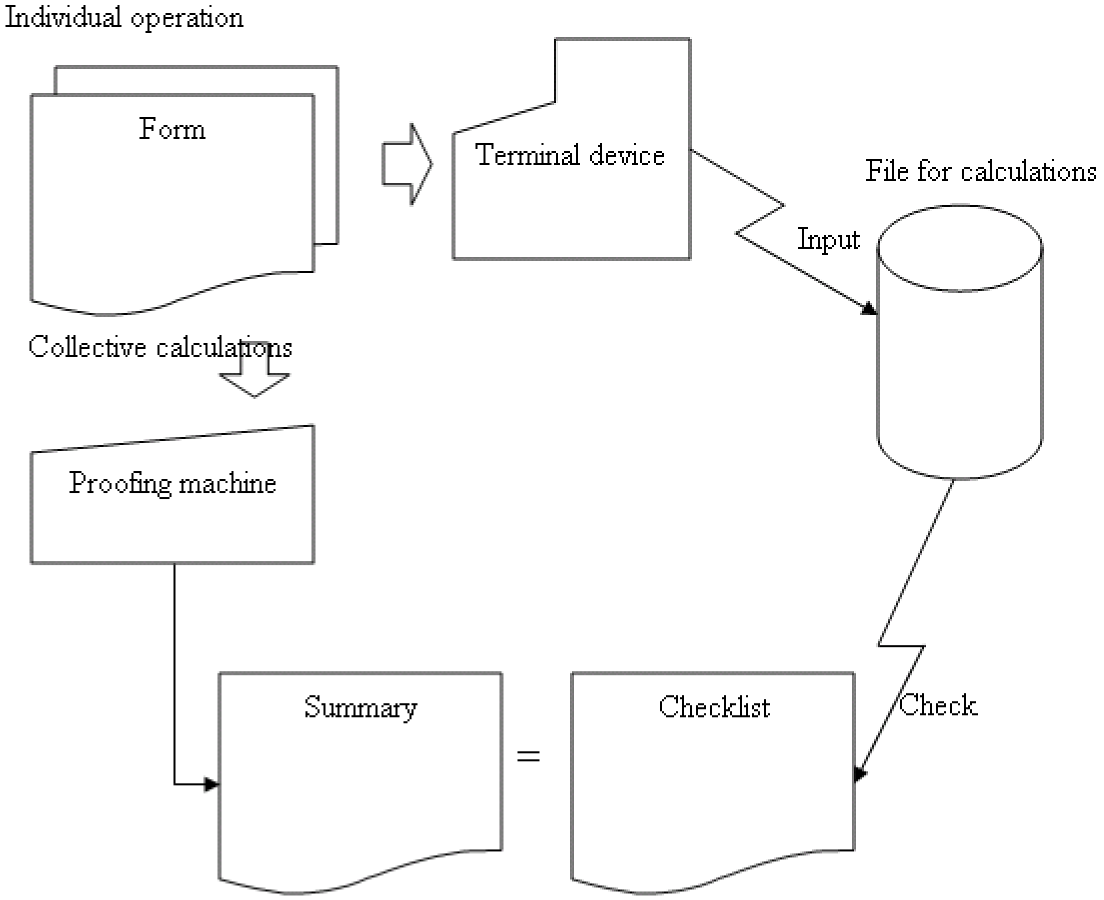
Input a numeral string and character string (as double input) for a variable, such as an amount of money, to check the correspondence between the two strings on the receiving side of both parties.



b. ID check
Check the validity of data codes by inputting sub-keys in the master record as well as a master file search key, and compare the input data to the master record.

c. Thorough check

Calculate the amount of transactions by transaction type and the sum based on data on the terminal, when online processing has completed for the day. Compare the totaled value to the value counted per transaction at the center in order to find omitted data input, duplicate input, and erroneous input for the sum.



If the amounts in both the forms are equal, the conclusion is that there is no error.

d. Check digit

Place weight on each digit of the input number, and append the lowest digit of the result of addition to the input number. The same process is performed in the computer program to check, if the data is correct. With this method, careless mistakes can be prevented.

<Example of check digit>

(In case of System 1 Modulus 10)

Calculation procedures	Example
1.□Decide the multiplier (weight) and place weight on each digit of the basic code.	(Basic code)123456 (Weight) 121212
2. Multiply using the digits	1 2 3 4 5 6 XXXXX X 1 2 1 2 1 2 1 4 3 8 5 12
3. Add the numbers from left to right. To add a two-digit number, separate the digits.	1+4+3+8+5+1+2=24
4. Divide the result by the constant "modulus," and take the remainder.	24÷10=2…… Remainder 4
5. Subtract the remainder from "modulus," and use it as a check digit.	10-4=6
6. Place the check digit at the end of the basic code.	1234566

	System1	2	3	4	5	6	7
Weight	121212	131313	765432	987432	137137	765432	10,1,10,1,10,1
Modulus	10	10	10	10	10	11	11
Transcription error prevention rate	100	100	87	94.5	100	100	100
Transposition error prevention rate	97.8	88.9	100	100	88.9	100	100
Double transposition error Prevention rate	0	0	88.9	88.9	88.9	100	0
Random error prevention rate	90	90	90	90	90	91	91

- Transcription error: Error in a single digit
- Transposition error: Input error from the number reversal of neighboring two digits
- Double transposition error: Input errors from the number reversal of two digits with one digit in between them
- Random error: Irregular error including more than one of the errors above.

Type of errors	Description	Example
Erroneous character (transcription error)	Erroneous display of a specific digit	Correct : 123456 Incorrect : 128456
Reverse (transposition error)	Error by reversing the numerals on the left and right	Correct : 123456 Incorrect : 123546

3) Correcting errors

Erroneous input data must be corrected, and data must be inputted again.

a. Batch processing

- Feedback system

  An error listing is outputted and sent back to the data source so that they can modify it.
- Automatic correction system

  If an error is found, it is corrected according to the correction rules at the organization in charge or by the program.
- Erroneous data removal system

  Only correct data is processed, and erroneous data is carried over to the next process cycle.

b. Online processing

- All data is transmitted.

  Transmit all input data, and correct it by displaying the erroneous part. After correcting the data, transmit all input data again.
- Transmitting the partial data

  Transmit and display only the portion related to the erroneous data to correct. Transmit only the corrected data again.
- Screen composition

  Transmit and display only the erroneous data, while keeping input data in multiple screens. This is the processing method for replacing only the error items.

(3) Completeness of update

Controlling the completeness of update targets not only at the time of updates, but also in a series of processes beginning from identifying the transaction file received until processing immediately after the update is finished.

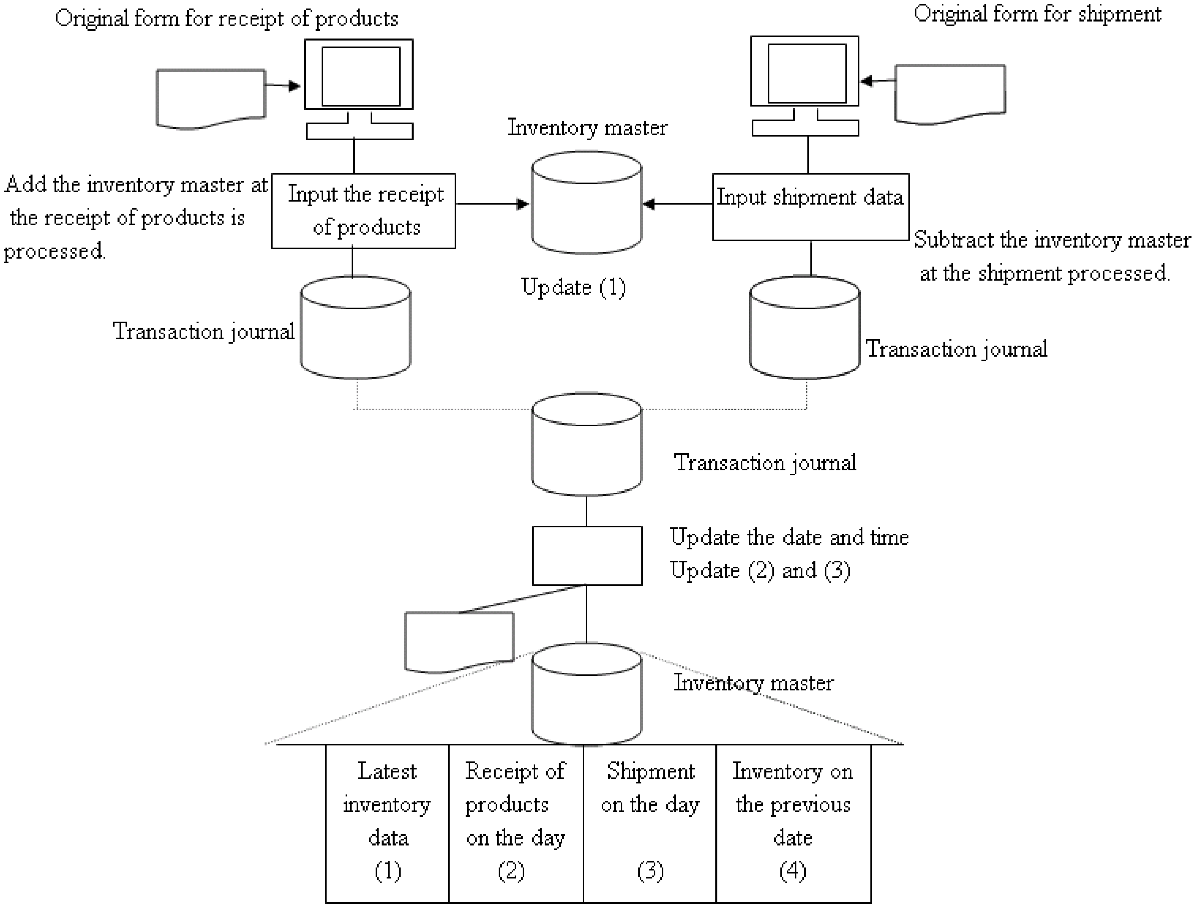
1) Duplicate processing method

A method of ensuring completeness by different procedures for processing the same process in

(Example)

a. Transaction data is entered online on a real-time basis, and the latest inventory data in the inventory master is updated immediately. Transactions are saved at the same time in the transaction file for batch processing.

b. Receipt of products and shipments for the day is updated in the inventory master based on the transaction records, and the results are compared to the latest inventory at the specified time interval (date and time). If there is a discrepancy between the latest inventory totals and the total products and shipments for the day plus the total inventory for the previous date, a checklist is outputted and the inventory data is updated to correct it after an investigation is conducted.


Original form for receipt of products
Original form for shipment
Inventory master
Add the inventory master at the receipt of products is processed.
Input the receipt of products
Input shipment data
Subtract the inventory master at the shipment processed.
Update (1)
Transaction journal
Transaction journal
Transaction journal
Update the date and time
Update (2) and (3)
Inventory master
Latest inventory data (1)
Receipt of products on the day (2)
Shipment on the day (3)
Inventory on the previous date (4)

If (1)=(2)-(3)+(4) is not valid, the conclusion is that there an error.

2) Online serial number check system
In online processing, a serial number may be added to a transaction during the process of receiving the transaction from the terminal. If the online system has failed and processing restarts, it is possible to check the extent of transaction processing, that is, find out the last transaction that has been completed. By using serial numbers, the completeness of an update can be ensured.

(4) Standardization of messages and design of message codes
Many kinds of messages for different people are outputted from computers, including user messages, messages for system operators and messages for programmers and maintenance staff. It is necessary to set criteria to define the standardized messages and to determine the kind of messages displayed for specific events, the meaning of individual messages, and the kind of expressions to be used. In other words, a message structure (input-output design) must be developed. Once the message structure is created, the process for message coding begins. Operability is enhanced if codes are added to messages in advance for control, and the codes are used instead of sentences in messages to handle the message.

1) Log messages for system administrators
Log messages for system administrators may be outputted in the following events:
Batch processing starts or ends. A failure occurs during off-line processing

The following points should be taken into consideration in the design of log messages.
a. Displaying start and end messages

Batch processing depends on log messages to obtain information about the start and end of programs. This information is especially important in the event of error occurrence, so the start and end messages must always be outputted.

b. Displaying the counts of input and output data items after work is completed

This is useful information for checking the consistency between input data and output data. It is especially important information for one-by-one processes for input and output to determine how much data is processed when processing is completed.

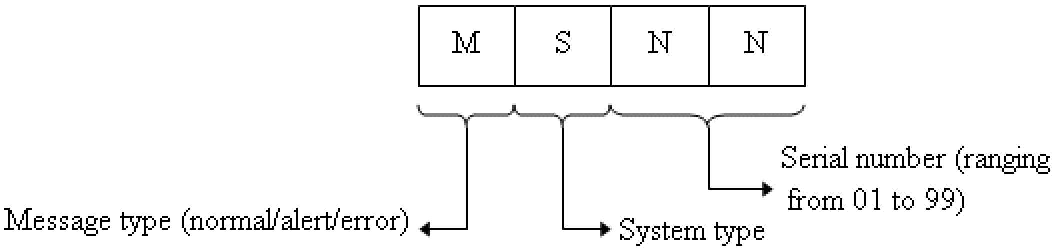
c. Standardizing log messages

A message can be categorized as normal, alert, and error. It is necessary to standardize the codes of log messages and the body of messages.

(Example of log output message format and output message)

* *	Code	Processing name	Body of message	Message type	* *
* *	NU12	SALCON	The XX process is completed correctly	NORM	* *

- Use the pre-defined program ID for the processing name
- Consider the type of characters, the maximum number of output characters, and combinations of characters for the body of messages. Avoid using multiple message codes by adopting the embedding system.
- Make the message type (normal, alert, error) easy to distinguish. One method is to print a serious error in red in order to provide notification.
- A code is provided uniquely to the body of a message (use of the embedding system is acceptable). It is necessary to set the criteria for providing the codes. A sample format is given below.


M
S
N
N
Serial number (ranging from 01 to 99)
Message type (normal/alert/error)
System type

2) User messages

User messages are outputted from the client terminal in the following events:

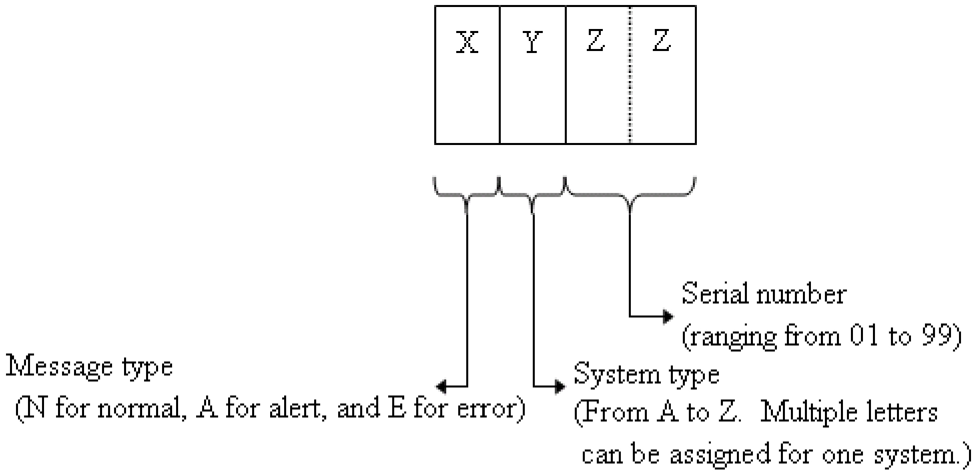
- When it is reported that transaction processing has completed normally or abnormally
- When it is reported that an erroneous value is found during the input check

User messages are designed with consideration of these factors.

3) Standardizing user messages

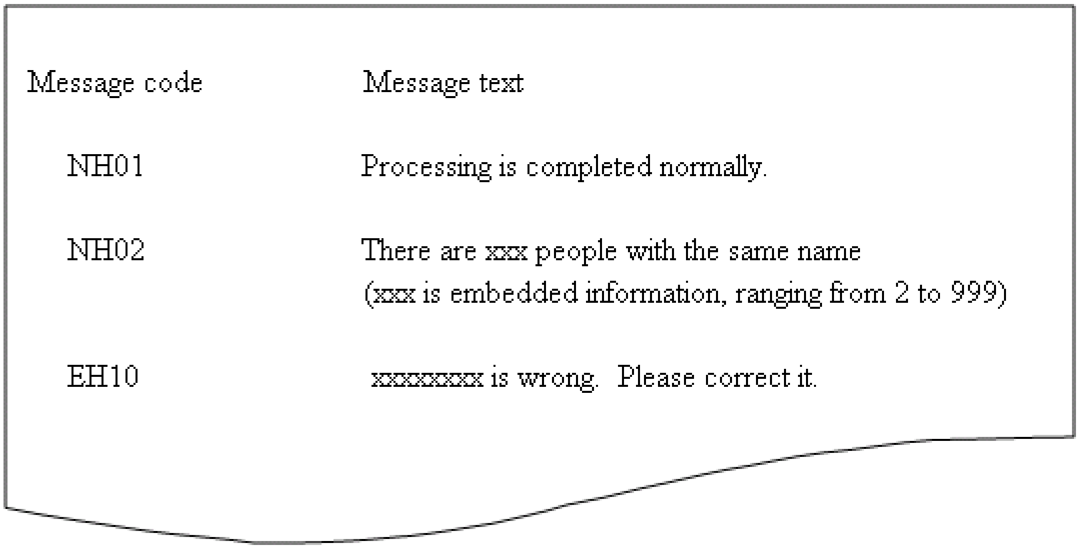
Decide on the criteria for issuing the message IDs and output method (e.g., display in a dialog box). Standardize the characters and sentence style to be used in messages. It is important to create appropriate messages providing details about the relevant topics, so that users can recognize the state of the system by reading the message.

<Example of criteria for issuing message >


X
Y
Z
Z
Serial number
(ranging from 01 to 99)
Message type
(N for normal, A for alert, and E for error)
System type
(From A to Z. Multiple letters
can be assigned for one system.)

4) Creating message specifications

Taking discussed points into consideration, create a message table.

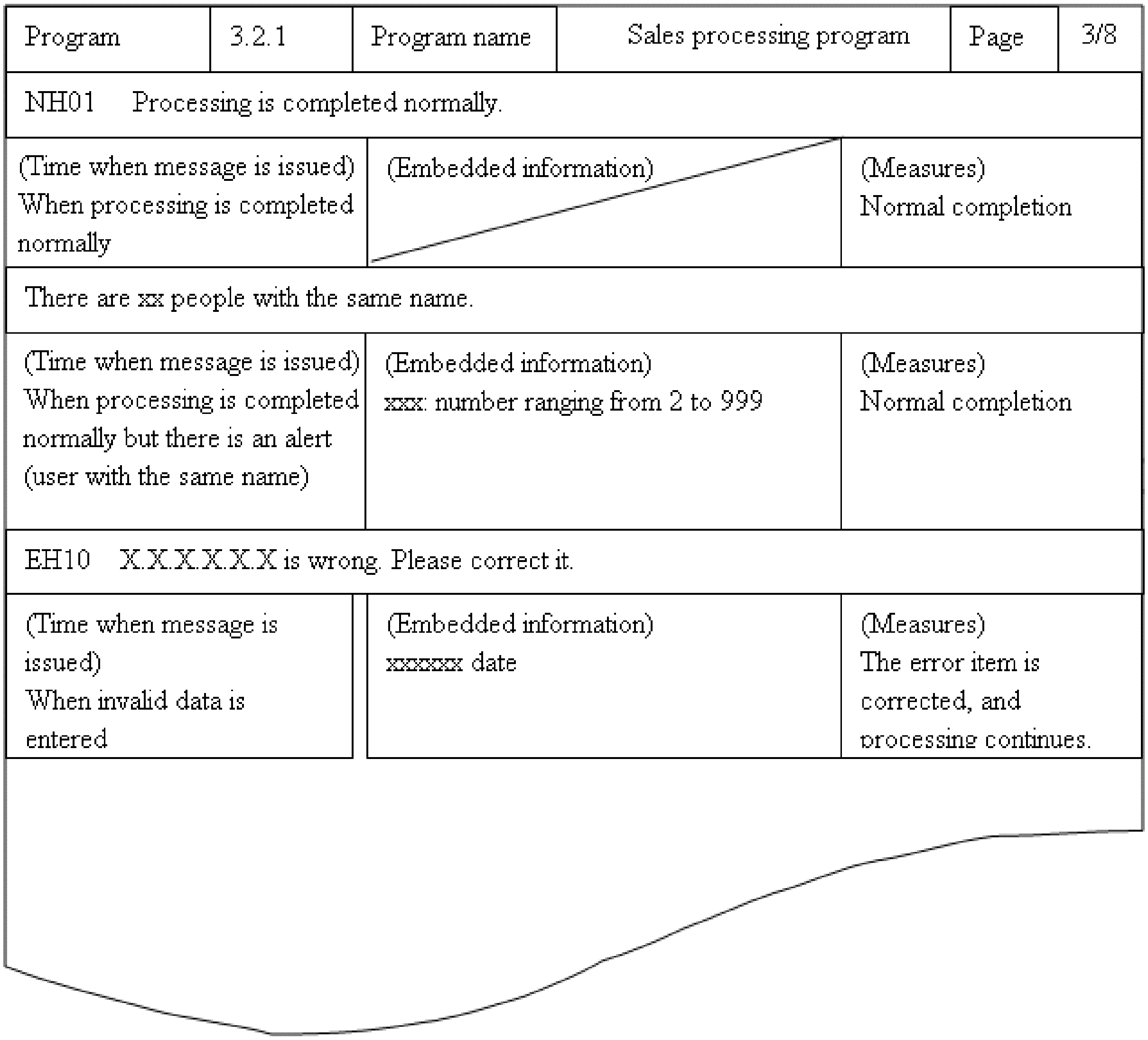


Message code	Message text
NH01	Processing is completed normally.
NH02	There are xxx people with the same name (xxx is embedded information, ranging from 2 to 999)
EH10	xxxxxxxx is wrong. Please correct it.

5) Creating message tables by processes

Create message tables by processes from the entire message listing. Include new messages on the message tables, if appropriate. It is necessary to identify the administrators who manage messages.

(Example)



Program	3.2.1	Program name	Sales processing program	Page	3/8
NH01 Processing is completed normally.					
(Time when message is issued) When processing is completed normally	(Embedded information)		(Measures) Normal completion		
There are xx people with the same name.					
(Time when message is issued) When processing is completed normally but there is an alert (user with the same name)	(Embedded information) xxx: number ranging from 2 to 999		(Measures) Normal completion		
EH10 X.X.X.X.X.X is wrong. Please correct it.					
(Time when message is issued) When invalid data is entered	(Embedded information) xxxxxx date		(Measures) The error item is corrected, and processing continues.		

Exercises

1. The following passage is about an input method for computer systems. From the group of choices listed below, complete the passage by selecting the correct terms.

With (1), even less experienced operators can read long codes (2) and instantaneously, and a system that causes few (3) can be realized. A (4) is the size of a pocket book, and it is portable. It is suitable for inputting data from multiple and (5) locations.

Answers:
a. handy terminal
b. input errors
c. unspecified
d. barcode readers
e. correctly

2. The following sentences are about GUI.
From group of choices listed below, select the terms that best match the description in each sentence.
(1) This is a graphic representation of detailed processing and a program.
(2) An item equivalent to the title of a menu is displayed in the upper part of a screen, and when it is elected the items in the menu are displayed a detailed menu is displayed.
(3) Users can use this to choose only one item from multiple options.
(4) Users can use this to choose multiple items from multiple options
Answers:
a. Checkbox
b. Pull-down menu
c. Pop-up menu
d. Icon
e. Radio box

3. The following sentences are points about standardizing the screen format in a GUI environment. Mark a circle next to correct statements, and mark an x next to incorrect ones.
(1) Do not use shortcut keys, and use many function keys.
(2) Standardize the locations of buttons, messages, etc.
(3) Create menus in accordance with standard specifications.
(4) Users can define the sequence of moving the focus (defining the tab sequence).
(5) Maintain consistency in the terminology, descriptions, and explanation methods in Help information.

4. What is the method of comparing specific data items in system processing results with corresponding results of manual calculations done in advance in order to check for omissions and extra additions in the records and to make sure that processing results are correct? Select one of the following choices.
Answers:
a. Validity check
b. Check digit check
c. Batch total check
d. Count check

5. Of all GUI components, what type of menu can be displayed at any place by right-clicking the mouse? The menu is displayed at the location of the mouse pointer when the right mouse button is clicked. Select one of the following choices.
Answers:
a. Drop-down menu
b. Pop-up menu
c. Feedback menu
d. Message menu
e. Cascading menu

6. The following table lists features of printing paper for reports. From the group of choices listed below, select the best terms to fill in the blanks and complete the table.

Item	(1)	Paper replacement	Readability
Special paper	High	(2)	High
General-purpose paper	Low	(3)	(4)
Form overlay output report paper	(5)	Not required	(6)

Answers :
a. High
b. Cost
c. Expensive
d. Not required
e. Low
f. Inexpensive
g. Required

7. What are the features of handy terminals as related to the criteria for selecting input and output media? Select one of the following sentences as the correct answer.
a. Recognition errors and rejected characters must be corrected.
b. Multiple items can be input for each key by sheet replacement.
c. Images with halftone such as photos can be read.
d. Accumulated data on a terminal can subsequently be processed collectively with a PC.

8. The following statements are about criteria for selecting the input and output media. Which device correspond to each statement? From group of choices listed below, select the appropriate device for each statement.
(1) Suitable for instantaneous input (reading) of many kinds of data items
(2) Suitable for input of a few items
(3) Suitable for real-time processing systems
(4) Suitable for image input (scanning)
Answers:
a. Optical mark reader
b. Barcode reader
c. Image scanner
d. Keyboard

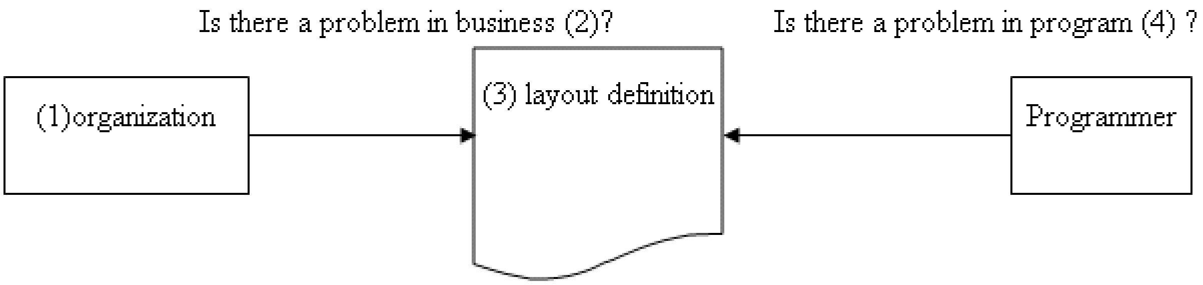
9. The following passage explains how to select input media, input devices, and input methods. From group of choices listed below, select the best terms to fill in the blanks and complete the sentences.

It is important to choose input methods that is less (1) for users and is less likely to cause (2) in daily and routine jobs.

If few errors remain in input data and clean data is obtained, the subsequent processes are quite different. Therefore, select input media, input devices, and input methods with consideration of those who create (3) (i.e., Are they dedicated operators, part-time employees, general staff, or average consumers?), their (4), input data volumes, and the data (5) in the event of errors

Answers:
a. input data
b. cost
c. difficult
d. age
e. output data
f. input errors
g. skill level
h. correction method

10. The following diagram illustrates the review celements of the report design process. From group of choices listed below, select the best terms to fill in the blanks and complete the diagram.


Is there a problem in business (2)?
Is there a problem in program (4) ?
(1)organization
(3) layout definition
Programmer

Answers:
a. development
b. user
c. screen
d. internally
e. operations
f. function
g. report

11. The following sentences are about screen transitions. Of the following sentences, which one is about a transition to an independent child screen?
a. Start an independent new screen.
b. Move to a pop-up screen. The parent screen and other screens can be operated, while the child screen is displayed.
c. It is a simple screen transition.
d. Move to a pop-up screen. When a child screen is displayed on the parent screen, the underlying parent screen cannot be operated.

12. The following sentences are about criteria for specifying field attributes. What kind of field attributes do they explain? From group of choices listed below, select the appropriate term for each sentence.

(1) This can be specified to suppress display of input content on the screen. It is used to input passwords, which should not be seen by third parties.

(2) This is specified for a field to be transferred to the host. Usually, transfer is not specified for fields set for protection.

(3) This is set for a field that is not to be printed, because the field information is not required on the hardcopy. For example, if the hardcopy is used as an official report, it is desirable to not print the normal end message.

(4) This is specified for items that operators are not allowed to input.

(5) This is used to emphasize a field and identify an area.

Answers :

a. Ruled lines

b. Reverse

c. Transfer setting

d. Suppression of display

e. Protection setting

f. Blink

g. Suppression of printing

13. Choose the sentence that does not describe GUI design with the prototyping method.

a. After prototypes are evaluated by users, they are used as templates for the finished products.

b. When looking at the prototype, users in the related departments can readily describe points for improvement, and they can accurately convey such information to programmers.

c. The operability and ease of viewing are difficult to check using only a specification on paper in developing a new job system, and users may submit requests for specification changes during the testing stage after the system is created.

d. Although prototypes are created only for experimental use, one should not depend too much on the support tools to create prototypes easily until users confirm their effectiveness.

14. From the following phrases, select the one that describes work in the GUI design process.

a. Design of the report outline

b. Creating the screen image

c. Determining output data items

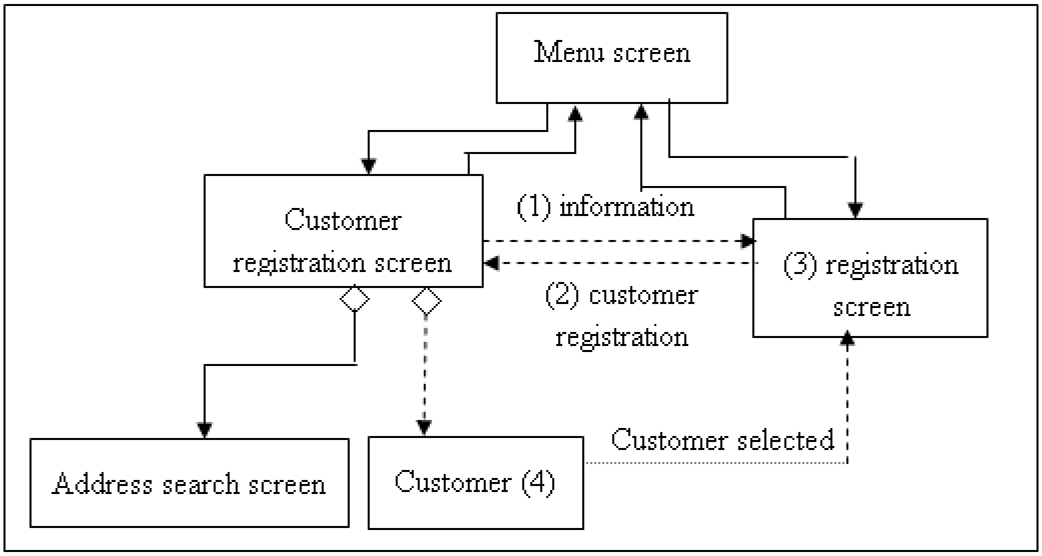
d. Design of database

15. The following passage is about dialog boxes. From group of choices listed below, select the best terms to fill in the blanks and complete the sentence.
Used to ask for a user's response or give instructions, this is also displayed when instructions are required from a user or a message has to be issued.

Dialog boxes are used to (1)(2)(3) or instruction. Also it is displayed when (2) instructions are (4) or a (5) has to be issued.

Answers :
a. required
b. prompt
c. user's
d. message
e. response

16. The following diagram shows an example of creating a screen transition diagram. From the group of choices listed below, select the best terms to fill in the blanks and complete the chart.



Answers:
a. inquiry
b. new
c. customer
d. processing
e. order
f. exceptional