

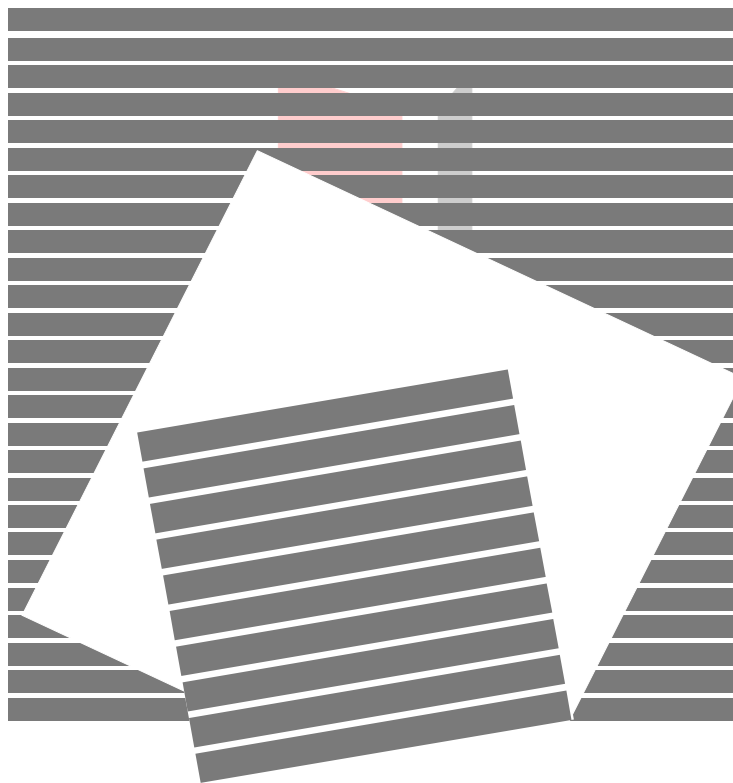


METI
Ministry of Economy,
Trade and Industry

Textbook for
Software Design & Development Engineers

NO. 3 **SYSTEM DEVELOPMENT,**
OPERATIONS
AND MAINTENANCE

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



Second Edition

REVISED AND UPDATED BY



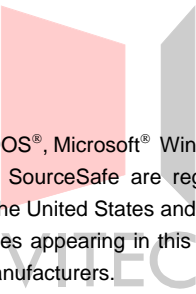
Japan Information Processing Development Corporation
Japan Information-Technology Engineers Examination Center

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

- 
- Microsoft®, MS-DOS®, Microsoft® Windows®, Microsoft® Windows NT®, Project 2002, Visio 2002 and Visual SourceSafe are registered trademarks of Microsoft Corporation of the United States in the United States and other countries.
 - The product names appearing in this textbook are trademarks or registered trademarks of the respective manufacturers.
 - “PMI” and the PMI logo are service and trademarks registered in the United States and other nations; “PMP” and the PMP logo are certification marks registered in the United States and other nations; “PMBOK”, is a trademarks registered in the United States and other nations.
 - Capability Maturity Model, Capability Maturity Modeling, and CMM are registered in the U.S. Patent and Trademark Office.

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

©Japan Information Processing Development Corporation/Japan Information-Technology Engineers Examination Center 2001,2002

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 (COConstructive 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
7.2.7 Reverse engineering	1-131
7.2.8 Re-engineering	1-131
Exercises	1-133
 <u>8 Trends in Software Engineering</u>	
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 1 Exercises	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

<http://www.vitec.org.vn>

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



<http://www.vitec.org.vn>

Part 1

SOFTWARE ENGINEERING

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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.

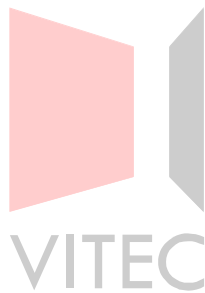
After studying this chapter, the student should be able to explain the methodologies and techniques that have been developed in this field.

1.1 Origin of Software Engineering

1.2 Definition of Software Engineering

1.3 Achievements in Software Engineering

Trung tâm Sát hạch Công nghệ Thông tin và Hỗ trợ đào tạo

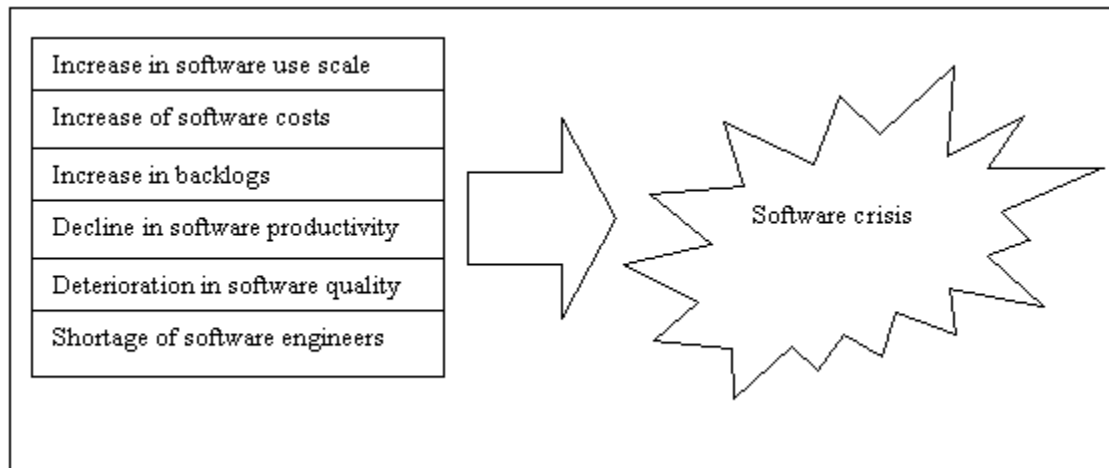


<http://www.vitec.org.vn>

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:



(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 σ (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.



<http://www.vitec.org.vn>

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

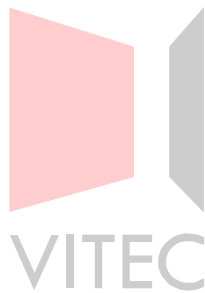
The function-oriented paradigm describes a "function" that returns an output value corresponding to an input value. It is based on functional programming languages such as Lisp.

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.



<http://www.vitec.org.vn>

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

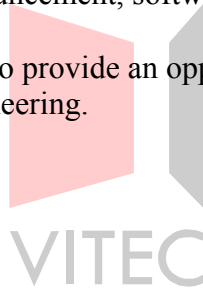
2. The following paragraph is a text about software. Fill in each blank by choosing the correct phrase from the list below.

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



<http://www.vitec.org.vn>

2 Process Models and Cost Models For Software Engineering

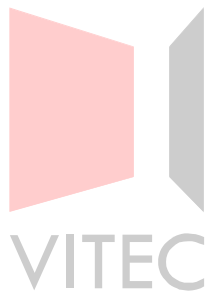
Chapter Objectives

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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



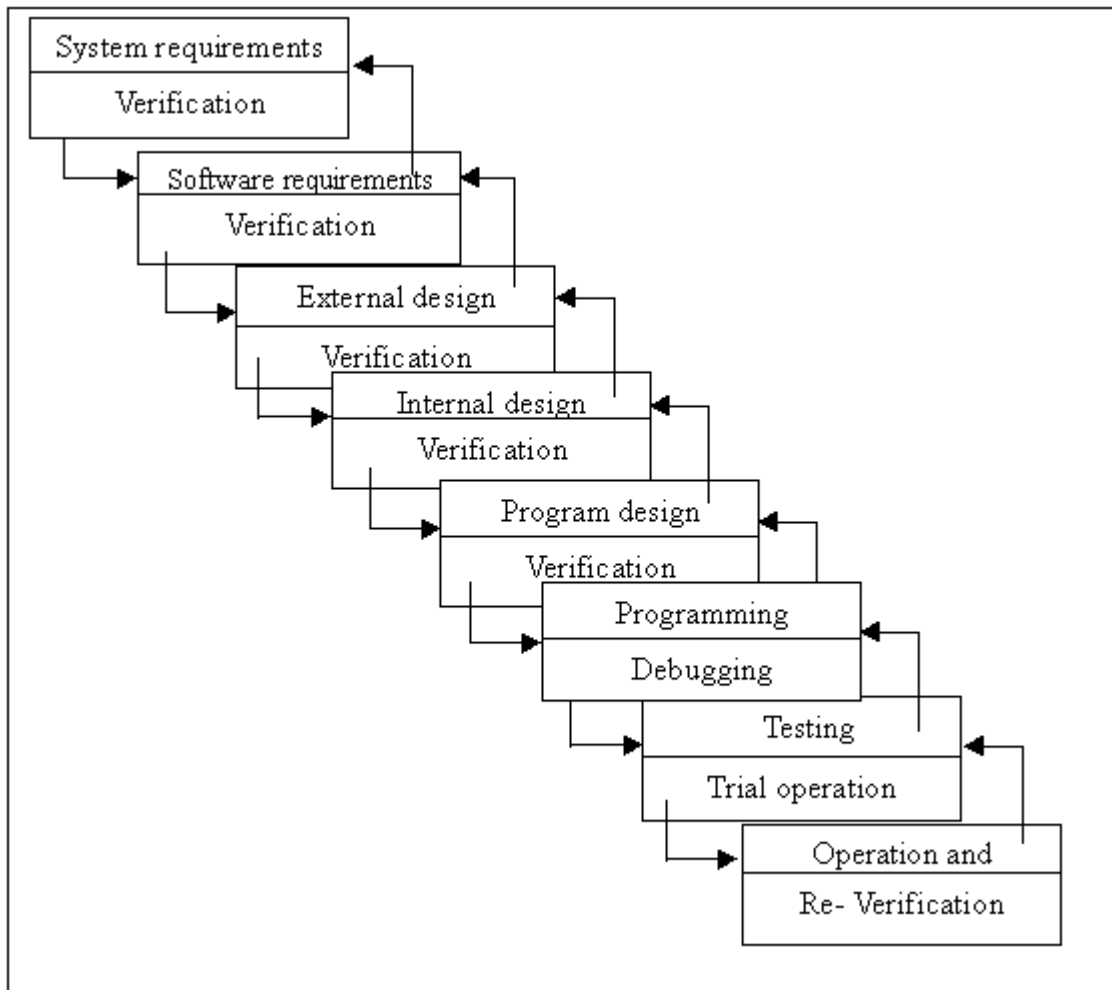
<http://www.vitec.org.vn>

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.

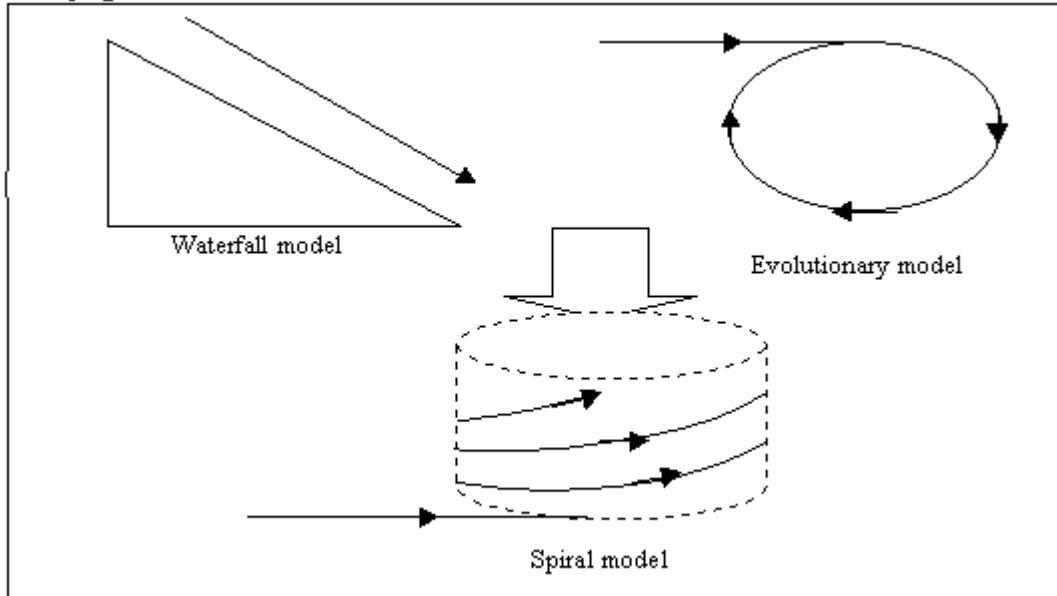
The waterfall model has been the most commonly used model since the birth of software engineering.

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]

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.

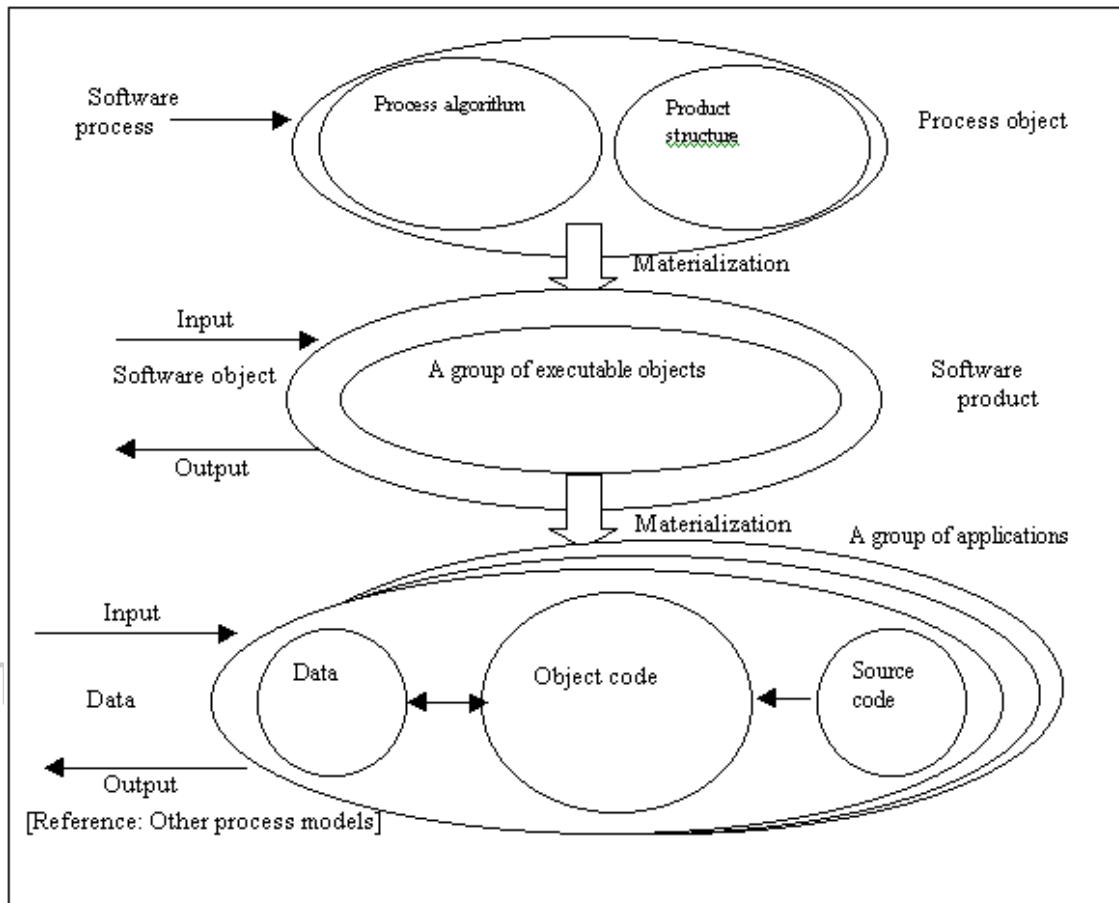
The prototyping model can have the following positive effects in system development.

- (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.

<http://www.vitec.org.vn>



[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

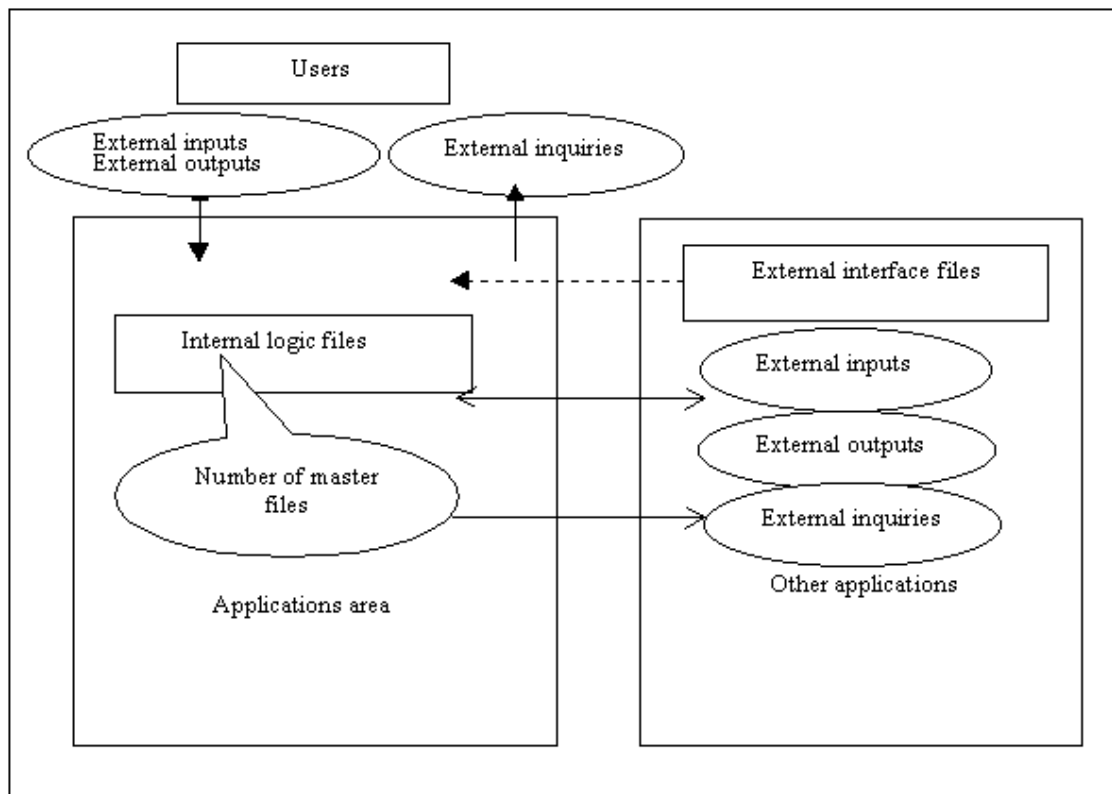
2.2.1 Halstead model

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:

$$FP = C \times (A1 + A2 + A3 + A4 + A5)$$

(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 (COConstructive 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.

$KDSI = \text{total source lines of code} - \text{number of comment lines} + \text{number of lines in JCL (job control language)}$

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

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

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.

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

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.

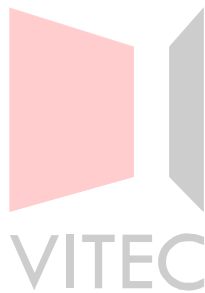
The fifteen attributes affecting development, to be used in the "intermediate" level are listed in the following.

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.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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.

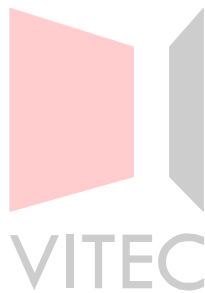
3.1 Software Requirements

3.2 Software Requirements Analyzing

3.3 Software Requirement Models

3.4 Techniques for defining Software Requirements

Trung tâm Sát hạch Công nghệ Thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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

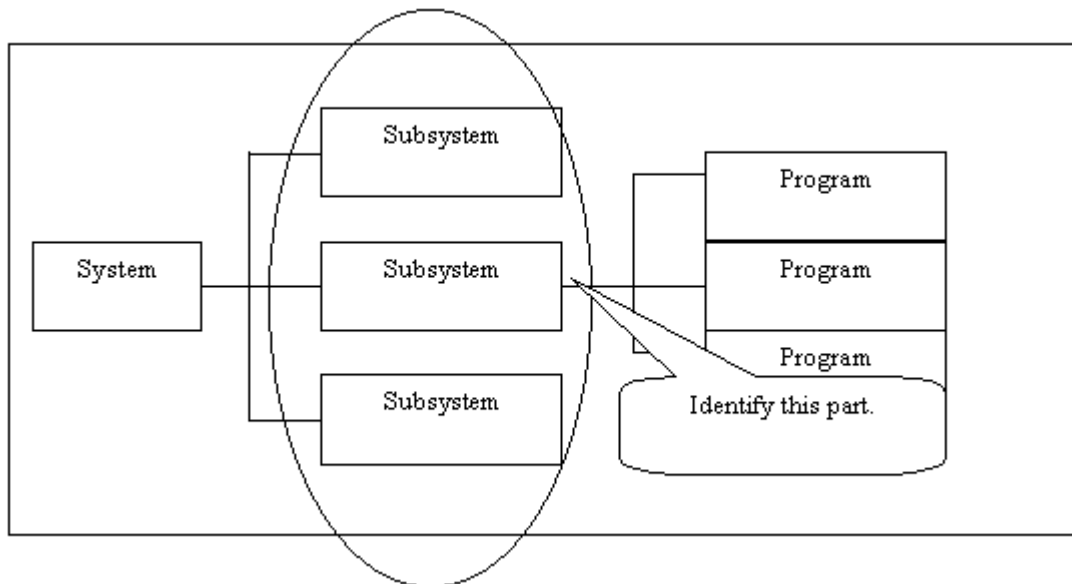
The purpose of systematization can be viewed from two standpoints: that of a customer and that of a developer. From a customer's standpoint, systematization aims to improve performance in the application in terms of time and money.

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

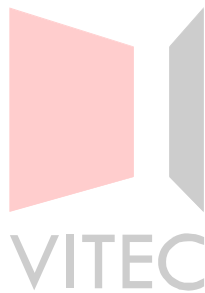
Performance is a system utility, and it refers to criteria for determining the extent of system performance. If unreasonable demands are placed on performance, functions are generally restricted. The effect of implementing a system is evaluated based on the functions and performance. The following table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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

3.2 Software Requirements Analysis

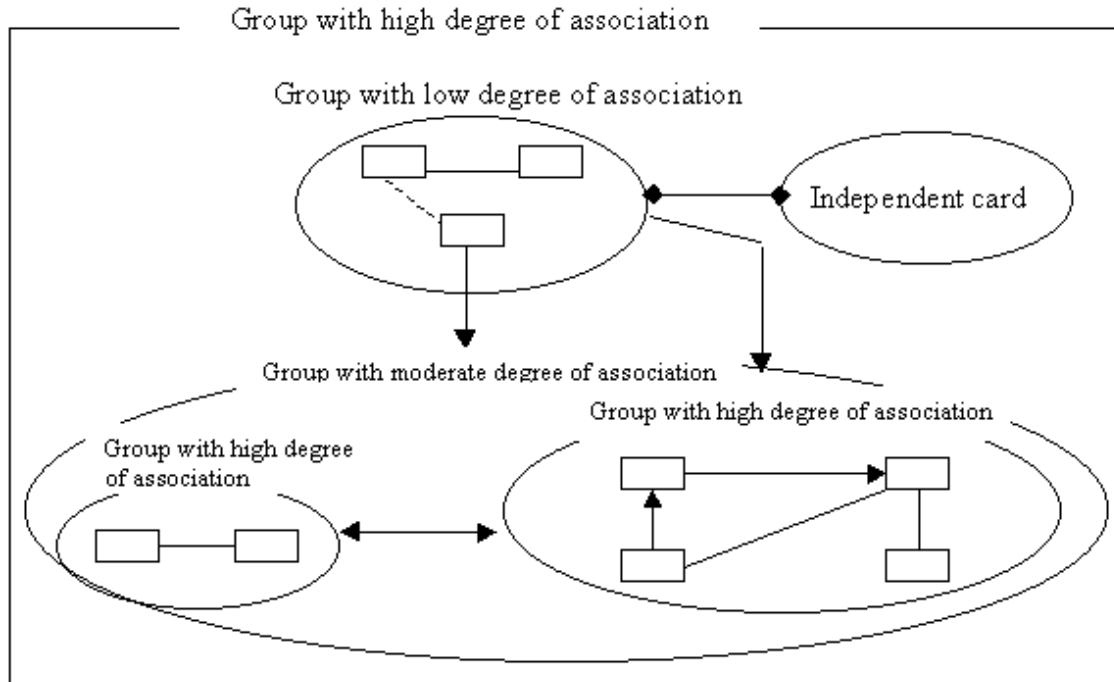
Software requirements analysis refers to a clarification on how to represent customer requirements after first identified. The following sections explain the KJ method, functional analysis, event response analysis, and structured analysis with some examples.

3.2.1 KJ (Jiro Kawakita) method

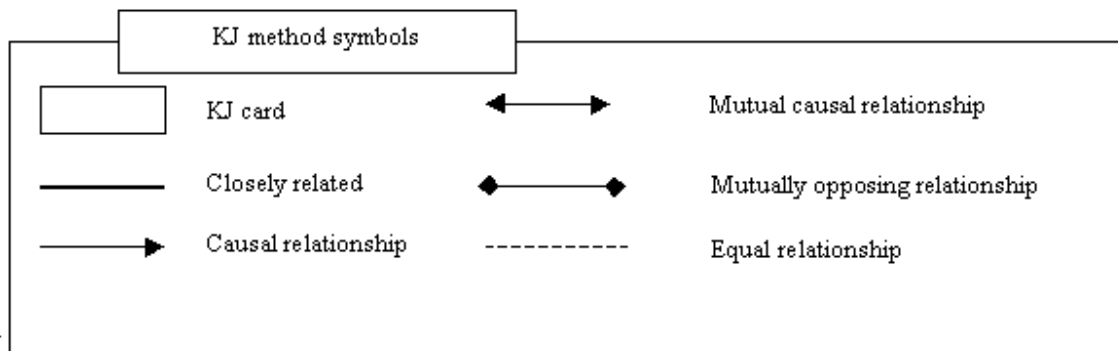
The KJ method is a software requirements analysis method proposed by Jiro Kawakita. This method supports the formulation of new ideas out of initial ideas, where the new ideas are derived from conceivable ideas for particular events. This method is suitable for determining requirements based on results 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.



- (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

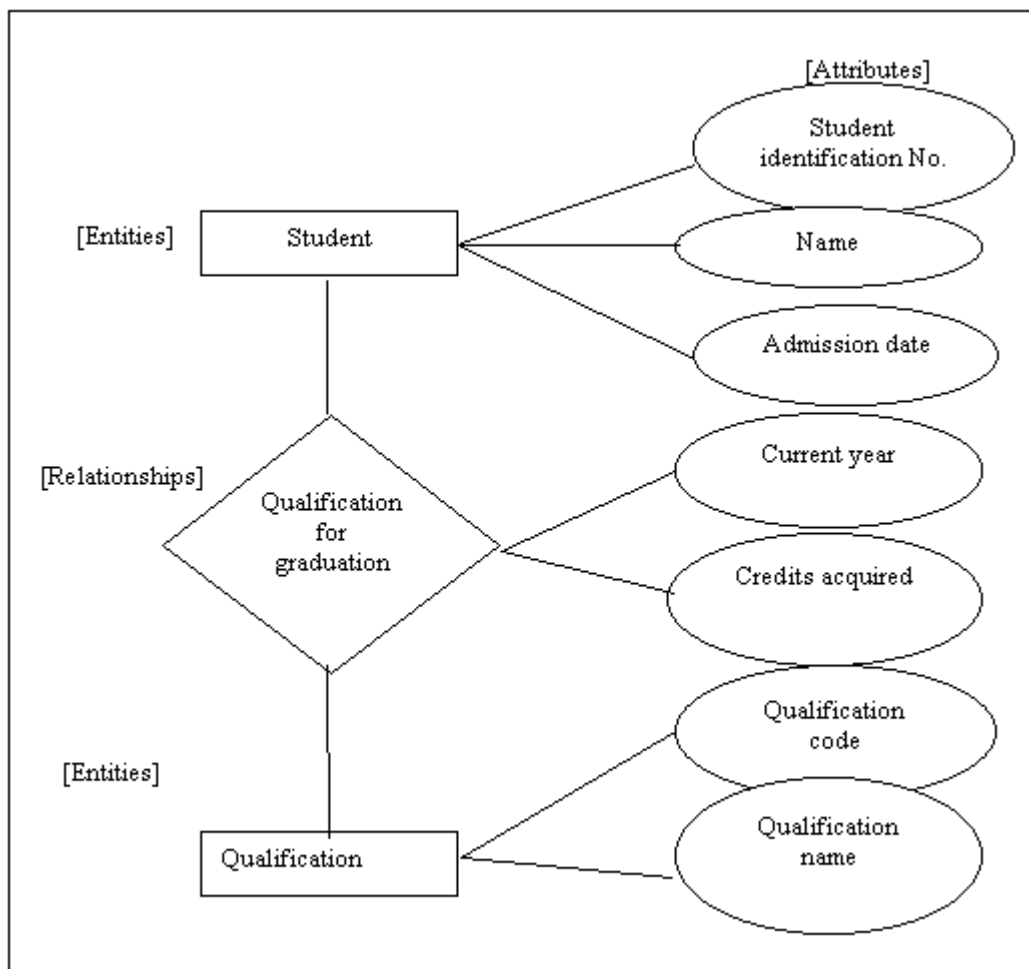
Event response analysis is a method of analyzing system responses to events from outside the system with the passage of time. Event response analysis is applied to the control flow model, finite-state machine model, and Petri-net model for software requirement models. For information about the control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



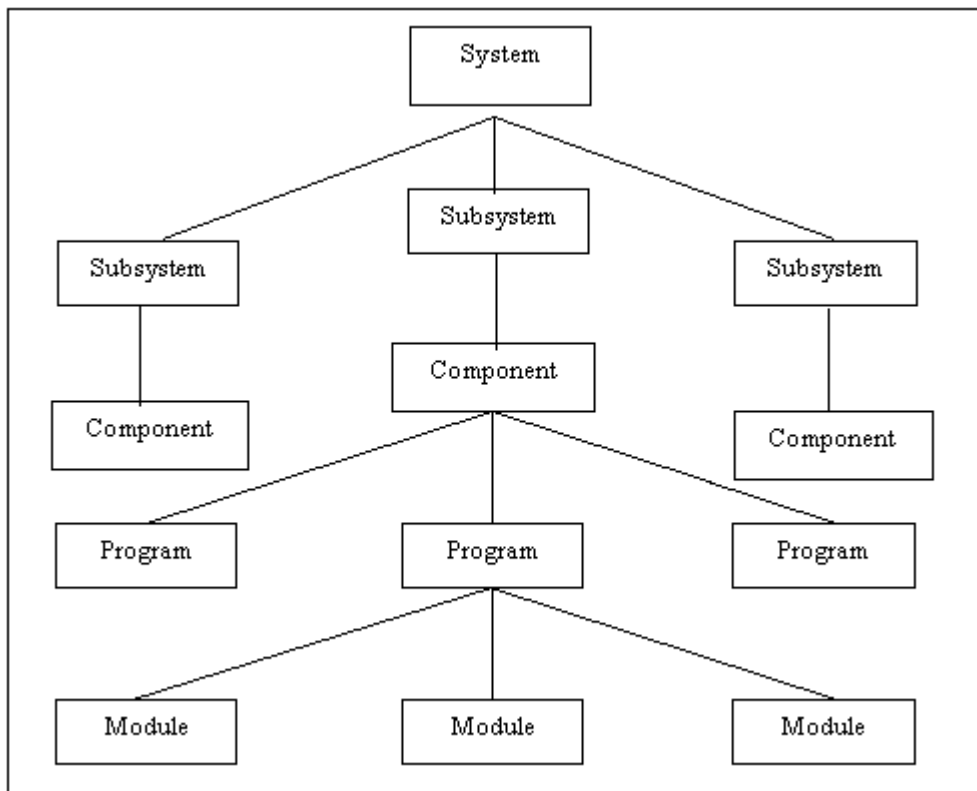
3.3 Software Requirement Models

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:

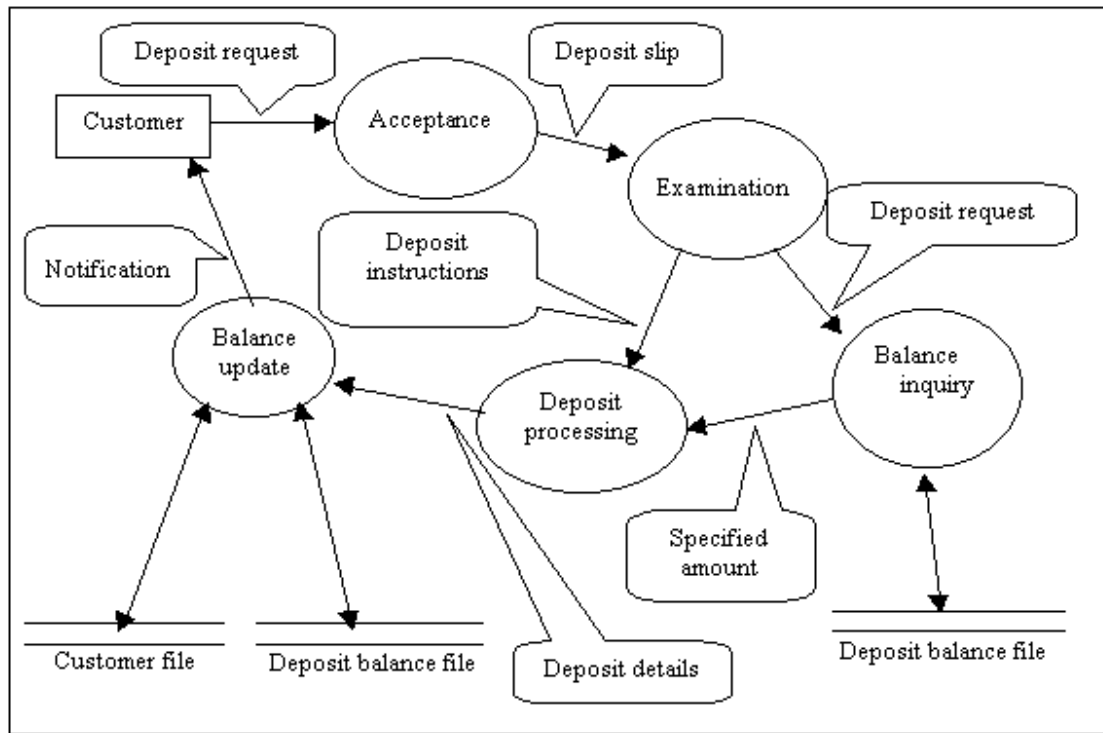
The functional hierarchical model is created by subdividing the system for each



function, as shown above. This model is the software requirement model most suitable for the waterfall life-cycle model.

3.3.2 Dataflow model

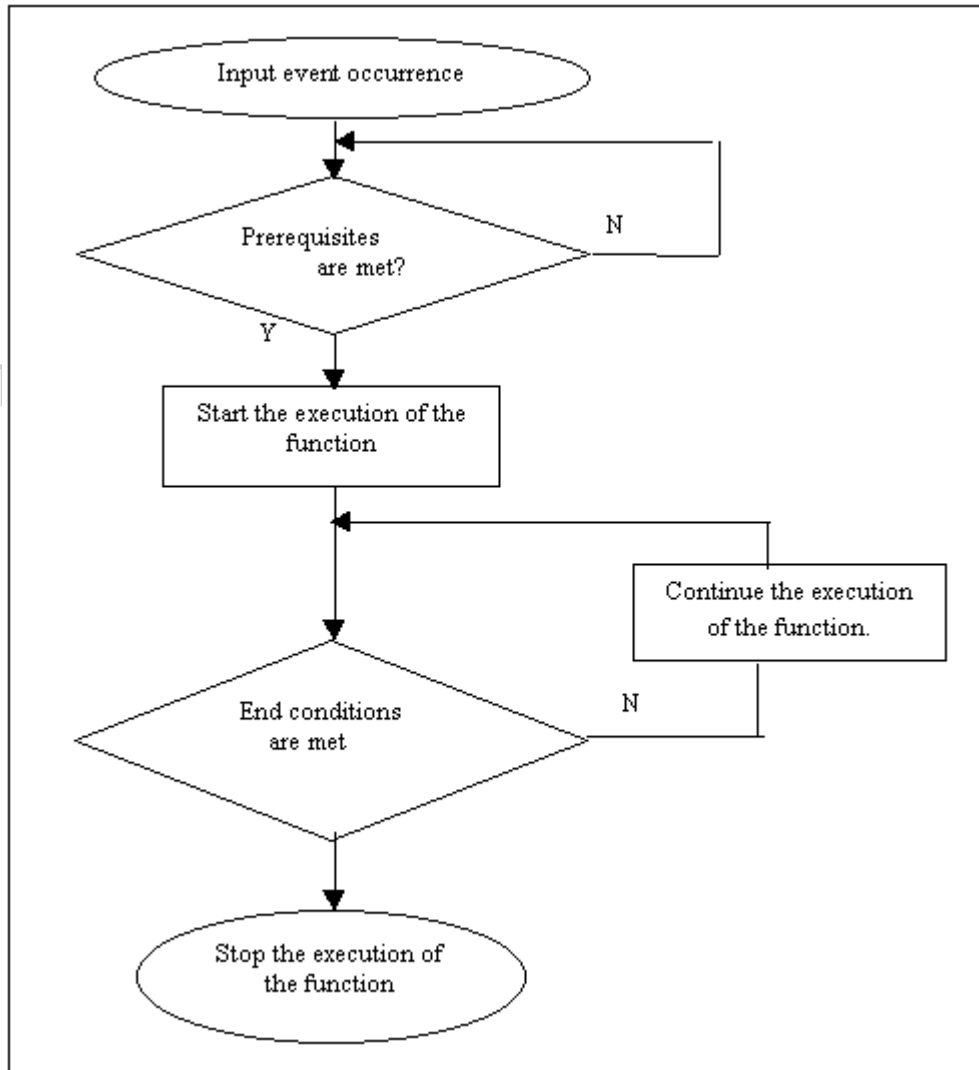
The dataflow model is a software requirement model based on functional analysis, and system functions are analyzed according to the relationships of input, functions, storage, and output. This model is suitable for analyzing requirements of application systems which are mainly business processing by 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

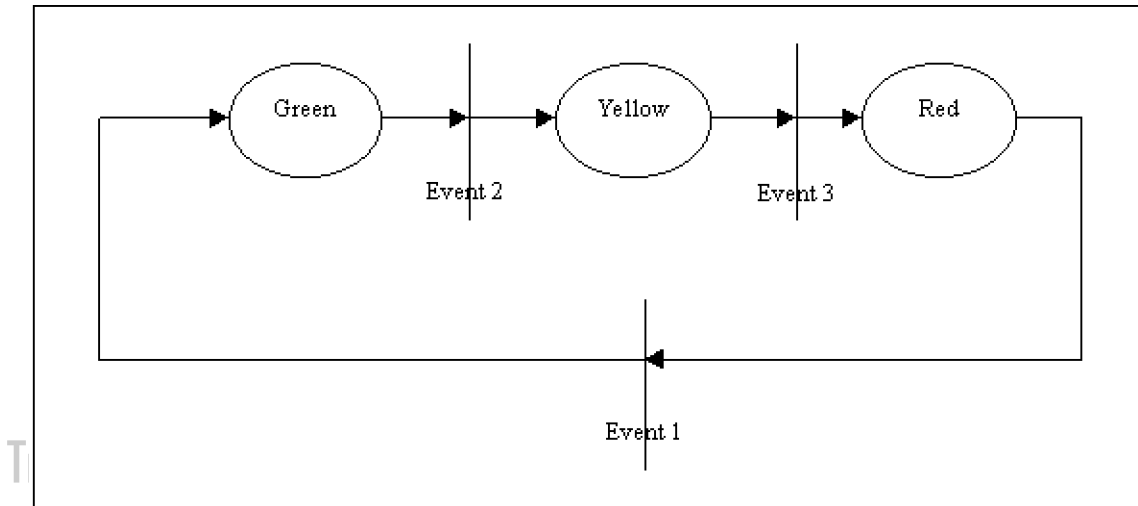
The control flow model is a software requirement model based on event response analysis. In this model, the system state and relationship between events depend on the passage of time. This model is most suitable for analyzing requirements of systems that require dynamic control. A typical example is 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.



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.

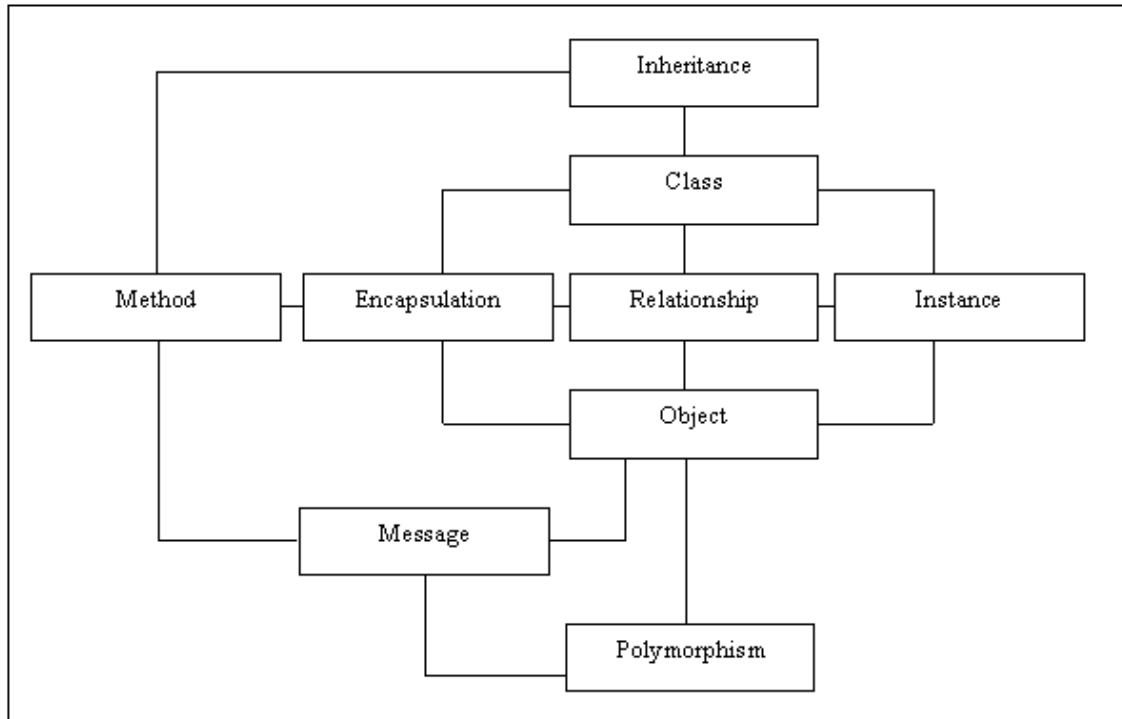
The basis of idea (1) is that the analyses of current jobs start in many cases with a close examination of input-output data. Also, existing input-output data is one of the most fundamental information.

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

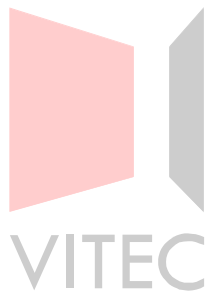
Inheriting particular properties

(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.



<http://www.vitec.org.vn>

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

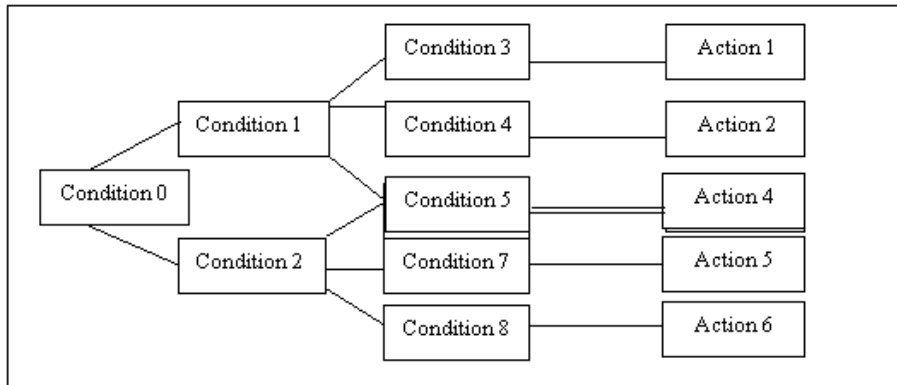
Y : The condition occurs

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.

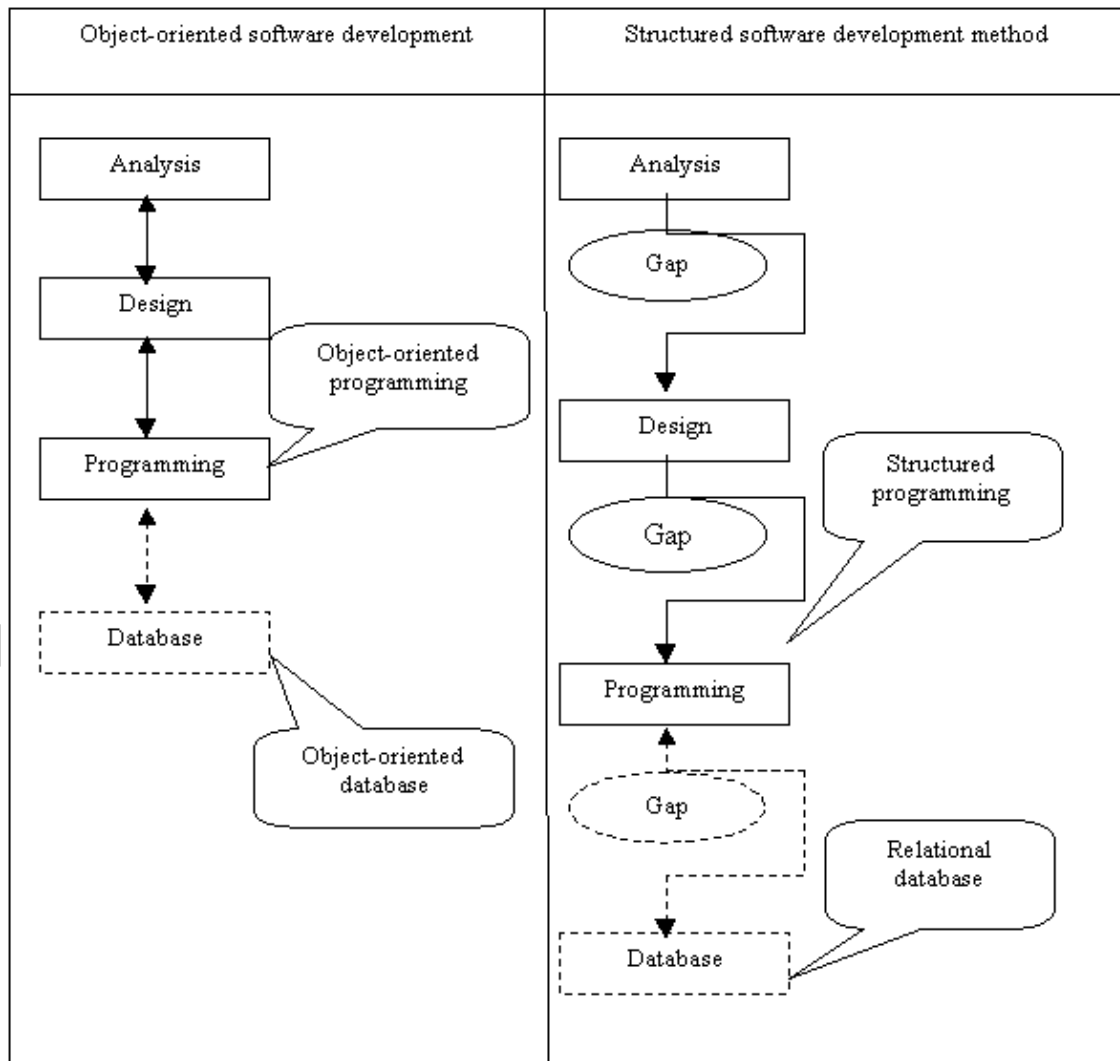


[Decision tree]

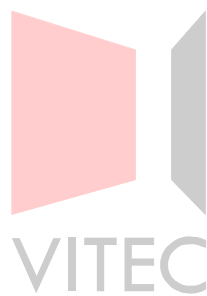
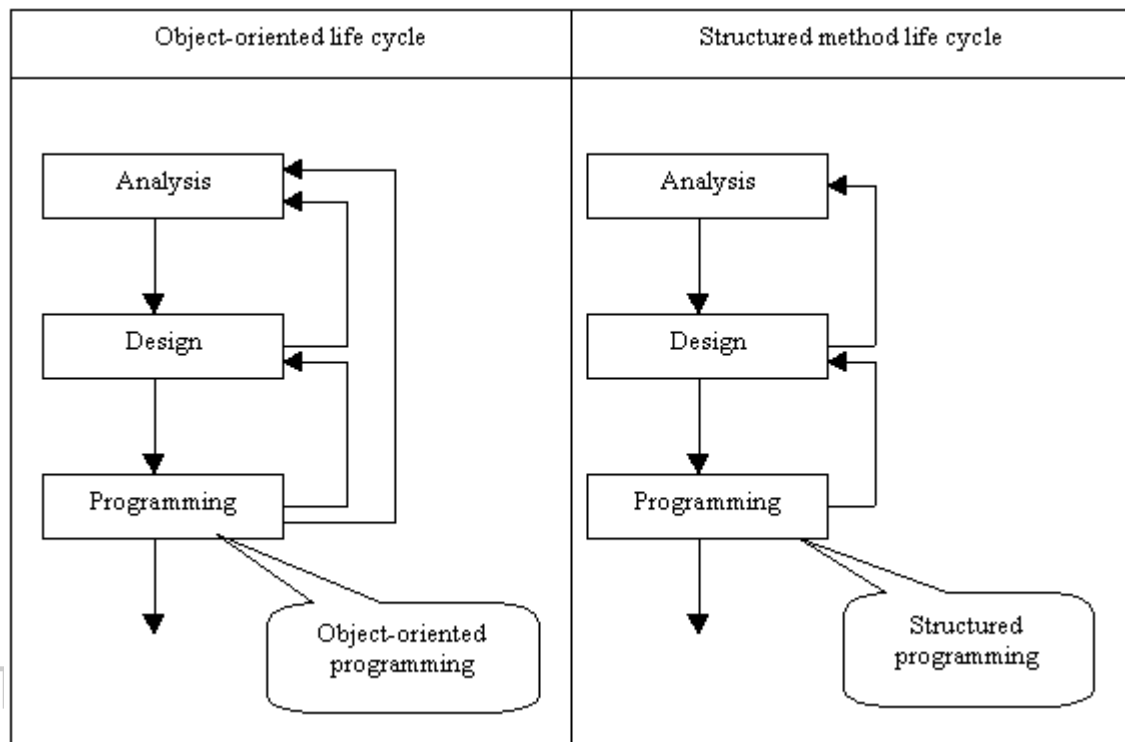
3.4.2 Object-oriented analysis method

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.

<http://www.vitec.org.vn>



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.



<http://www.vitec.org.vn>

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 external world
 - (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

<http://www.vitec.org.vn>

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.

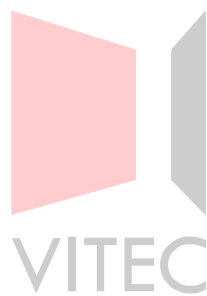
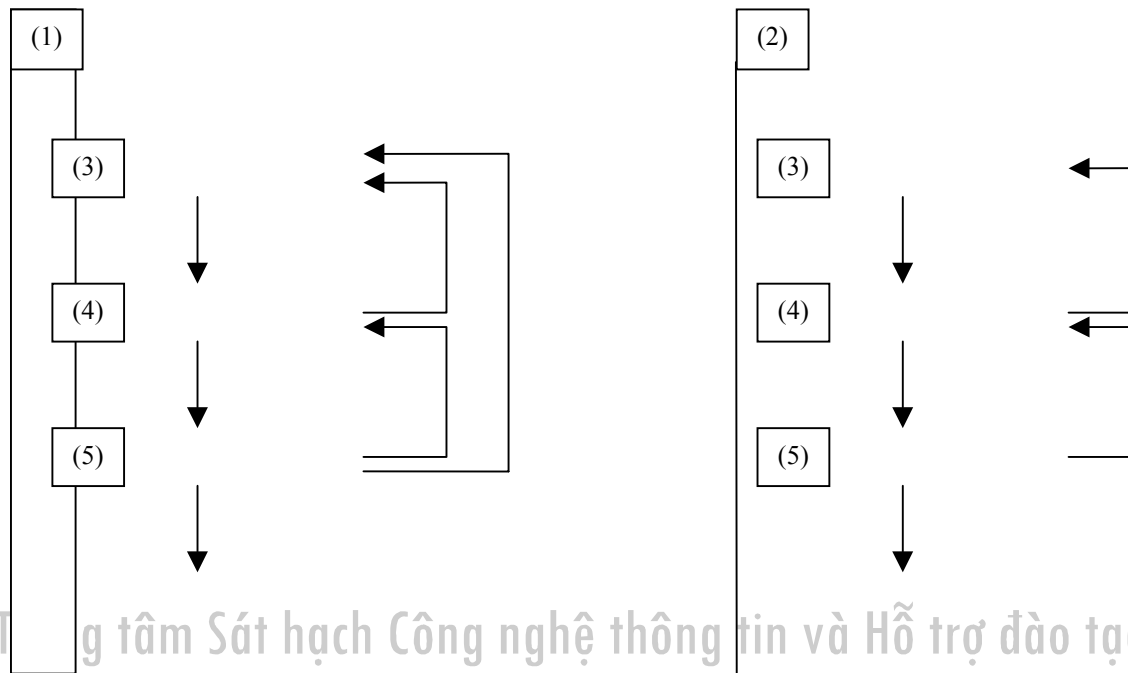
19. Use the best terms to fill in the [***square***] in the following description about system effectiveness.

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.



<http://www.vitec.org.vn>

4 Software Design

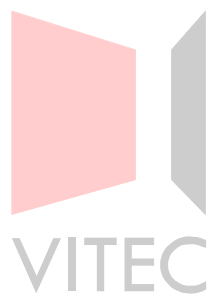
Chapter Objectives

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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

4.1 Concepts of Software Design

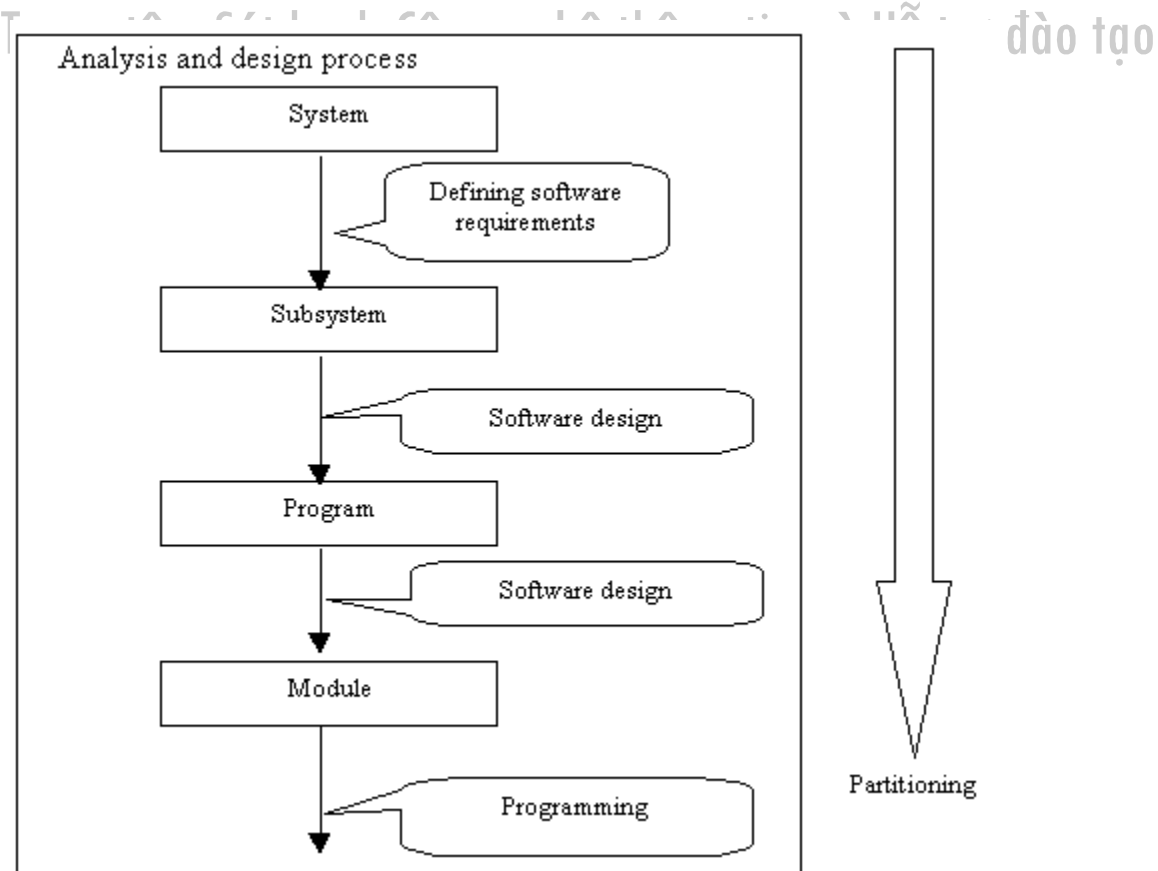
The functions required in software can be identified by partitioning a system during the design process, and the system can then be implemented by combining the identified functions. The concepts of division and integration are thus important in software design. This section explains software division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

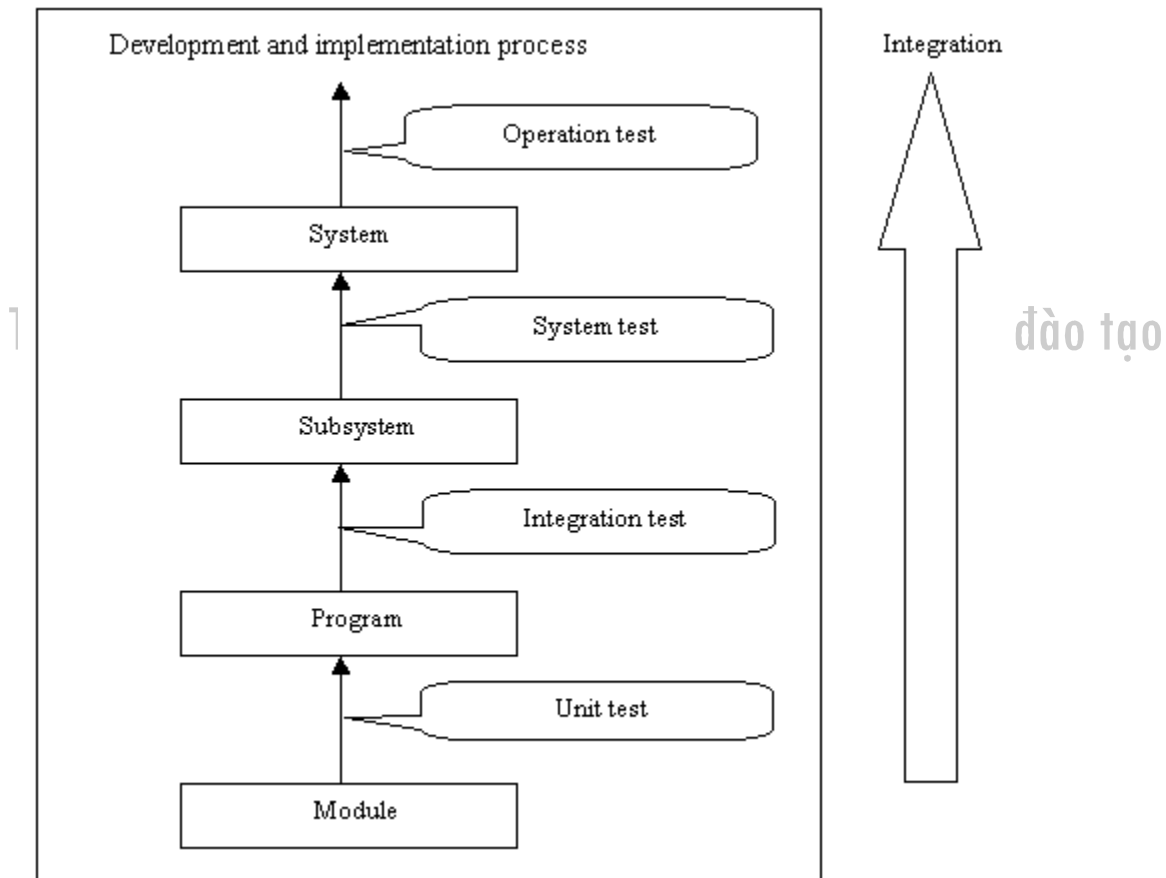
The concept of division is a principle used in the software analysis and design process. More precisely, the structure of a system can be divided step by step to organize the system into subdivisions. The following figure shows the analysis and design process:



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

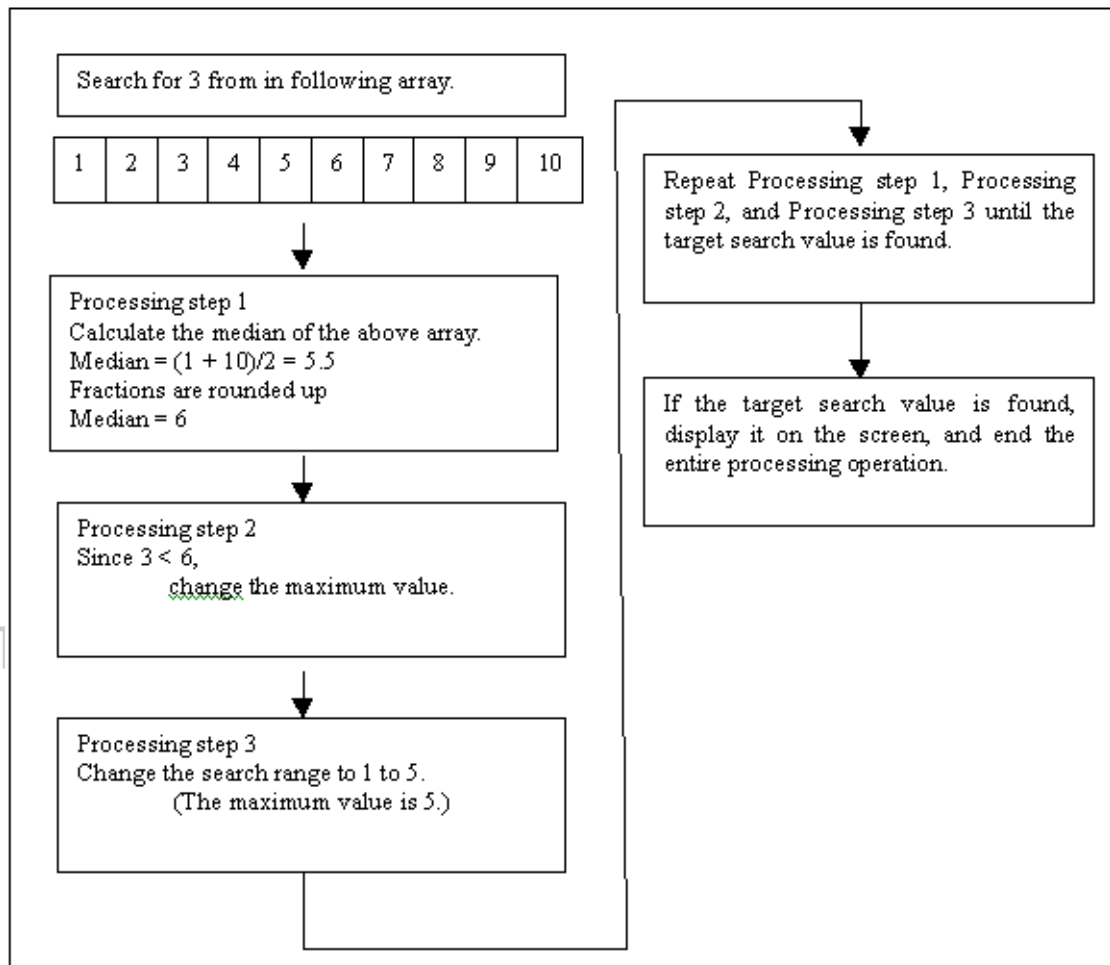
The concept of integration is a principle used in a series of development and implementation processes ranging from programming to verification. More precisely, a system can be created by integrating divided modules, programs, and subsystems. The following figure shows the development and implementation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



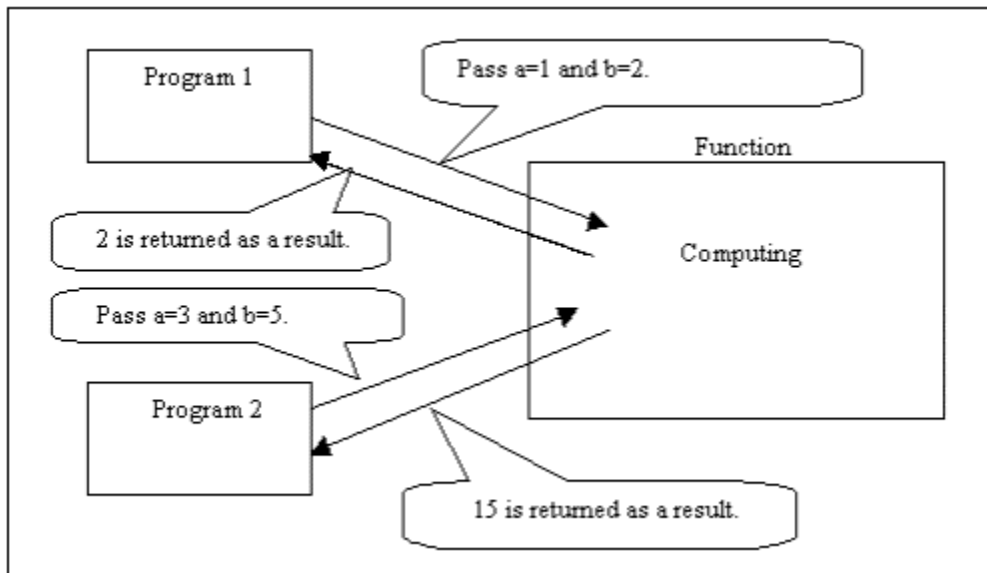
As mentioned in the diagram, subdividing a target section step-by-step is called stepwise refinement.

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

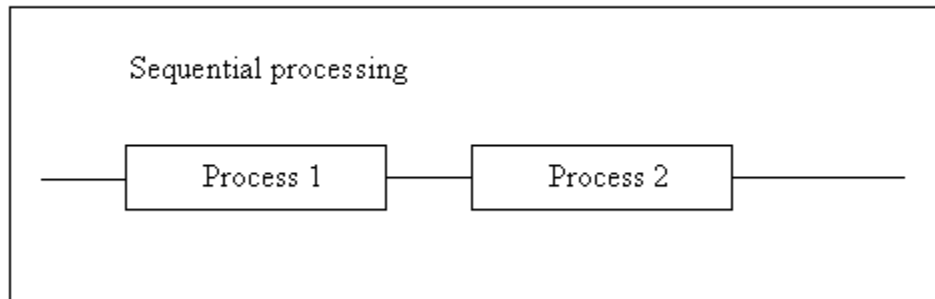
Control abstraction is imposing restrictions on control patterns by structuring algorithms described in a procedural programming language. The following table lists more concrete descriptions:

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

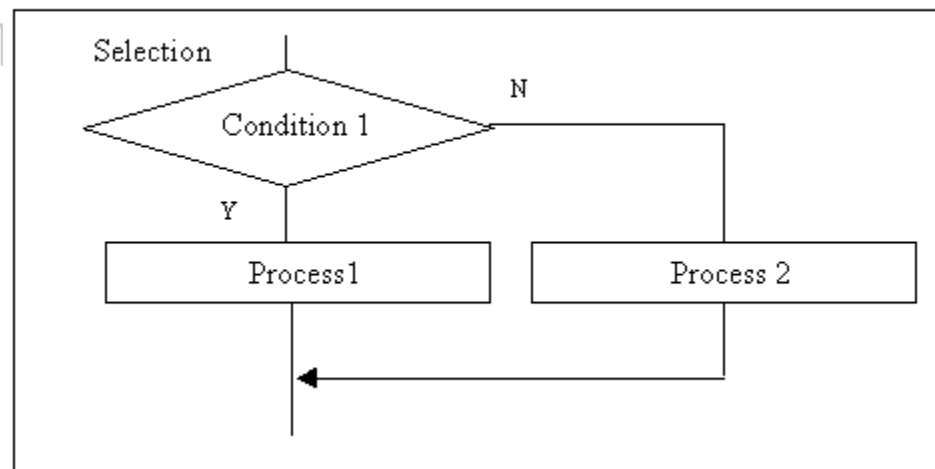
The flows of control and text described in the table are related to each other for the ease of understanding. That is, it is easy to understand an algorithm described in a program, if the flow of control matches that of text. Other principles for facilitating creation of easily understandable programs 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]

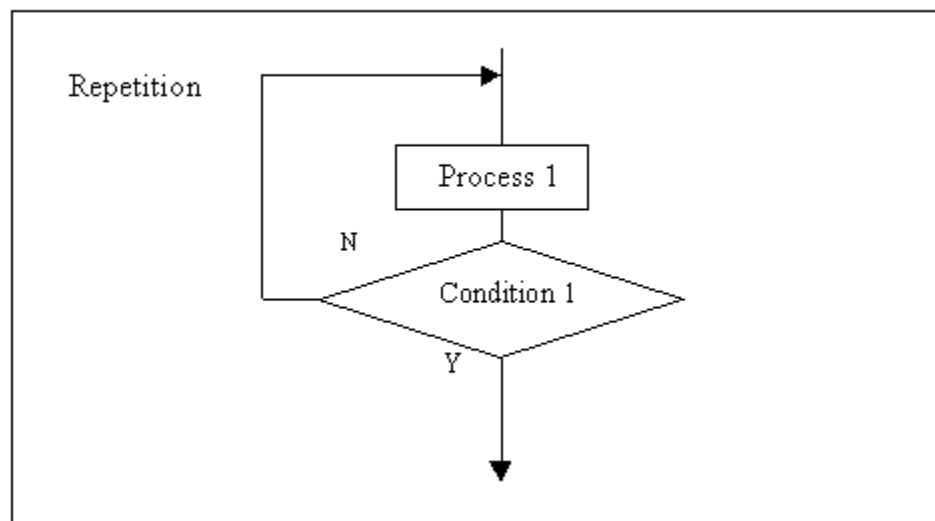


[Basic control structure of structured programming 2]



ợ đào tạo

[Basic control structure of structured programming 3]

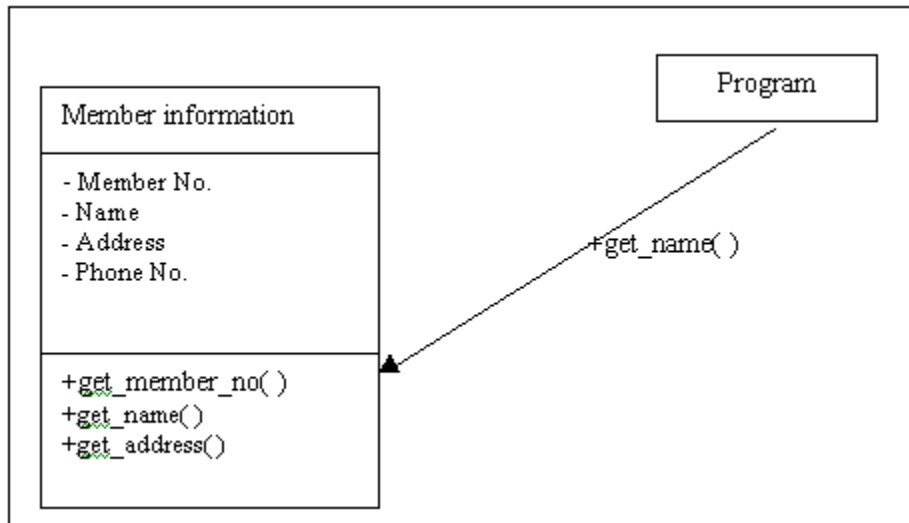


(3) Data abstraction

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()".

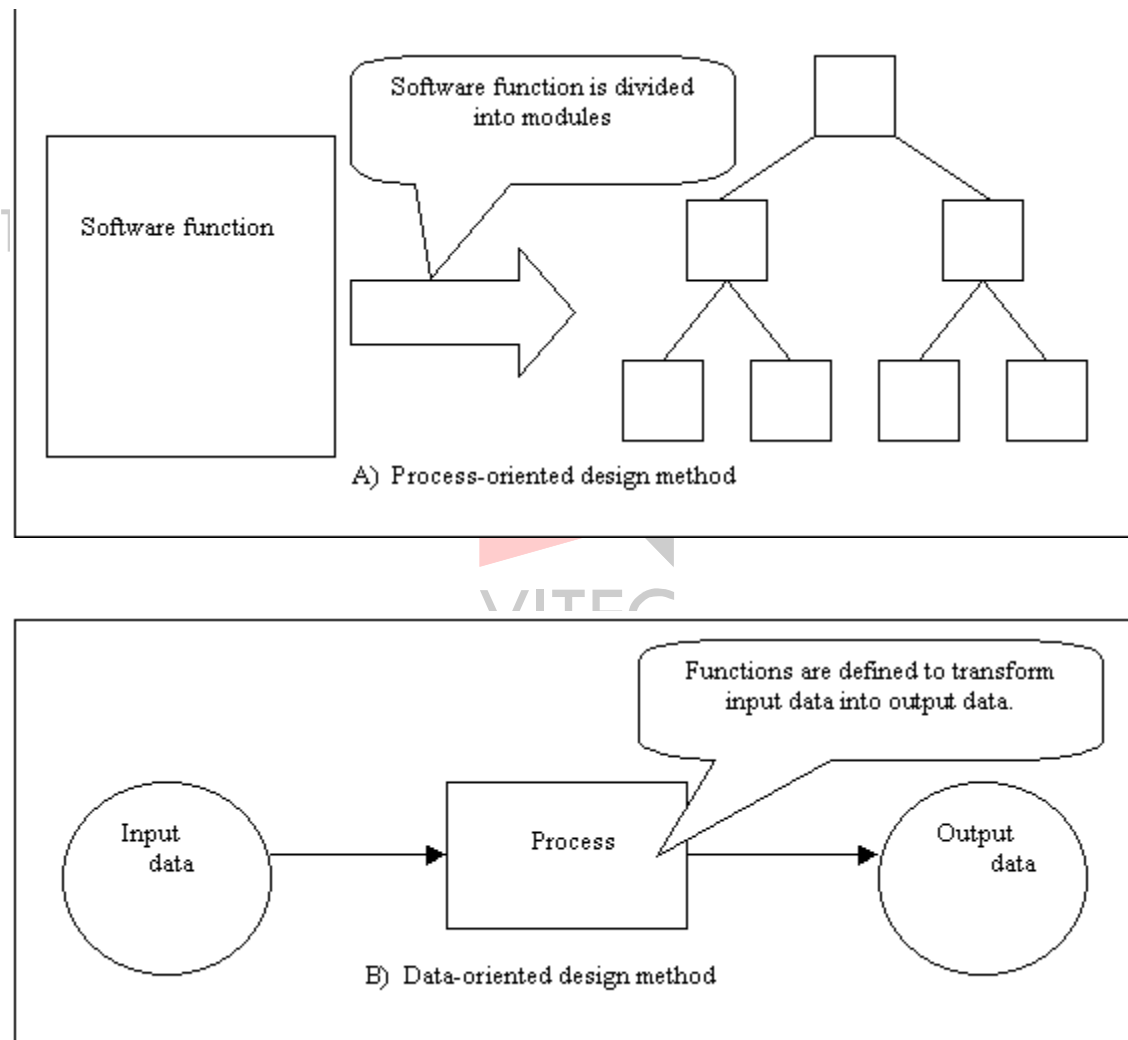
<http://www.vitec.org.vn>

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:

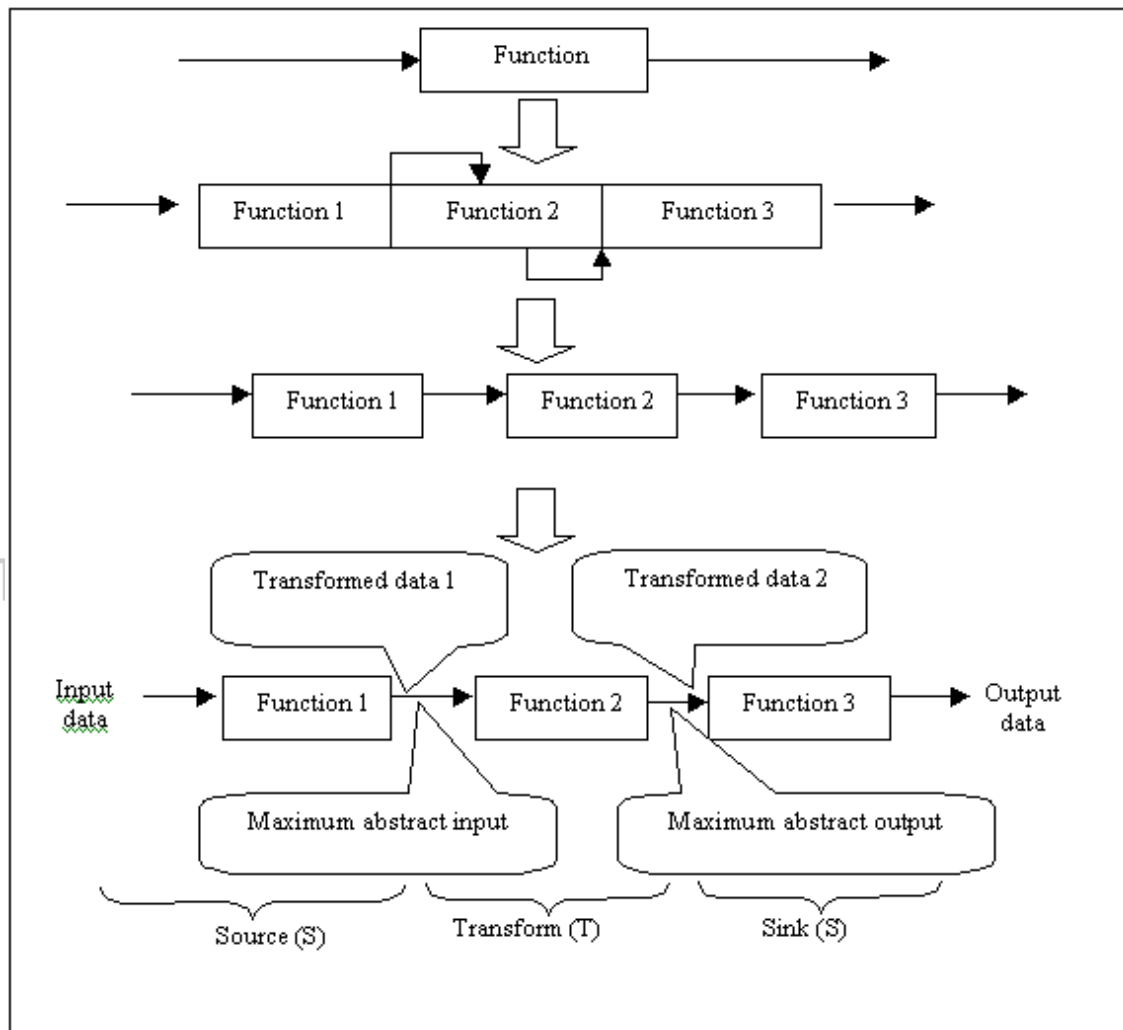


The structured design method is a design method that focuses on processes. This technique was proposed by Larry Constantine, and it is used to design the structure of a module and its interface. The structured design method consists of the STS partitioning method and TR partitioning method. Each of these 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 is each of the source (S), transform (T), and sink

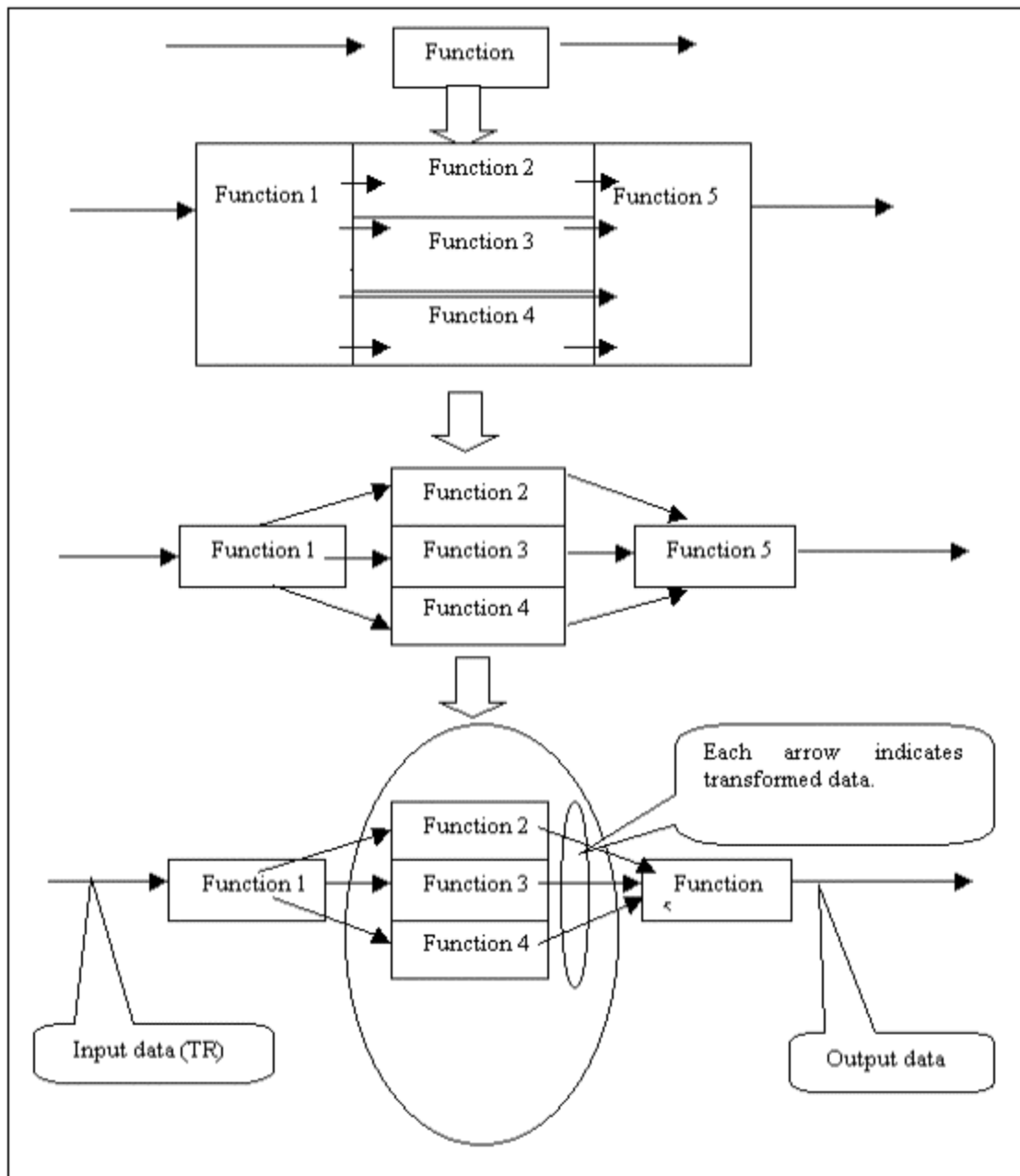
(S) is modularized. The following figure illustrates the process of the STS partitioning method:



[Software design method by the STS partitioning method]

(2) TR partitioning method

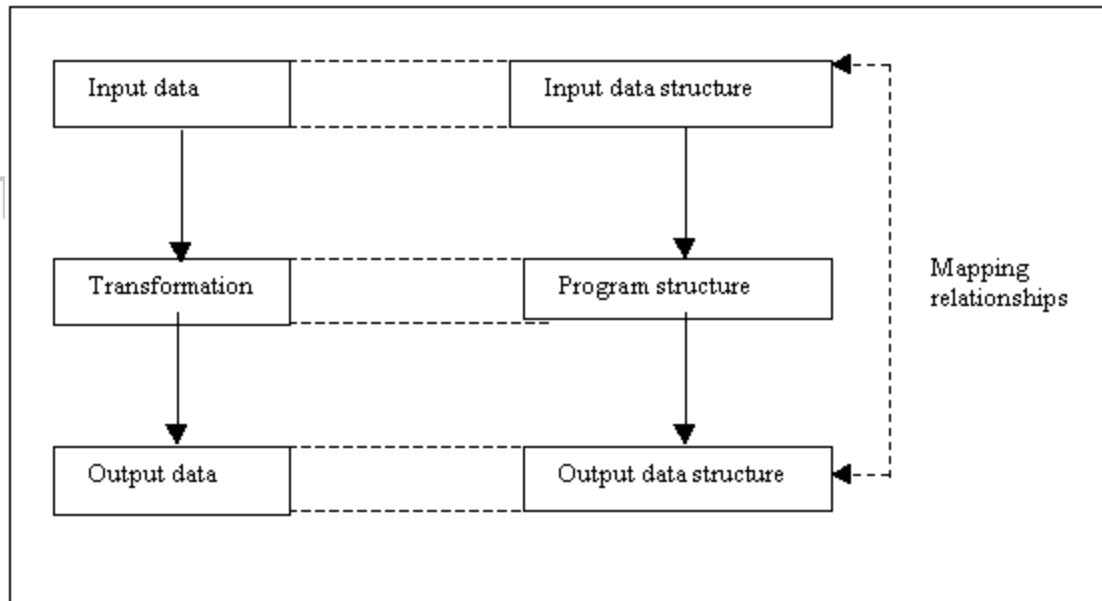
The TR partitioning method applies to flows of data in which branching occurs. In this method, the functions divided by transaction types are made modules. That is, such a flow is divided into a module for receiving transactions and assigning them to appropriate modules, transfer modules for individual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



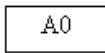
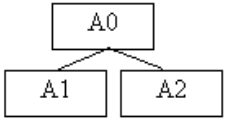
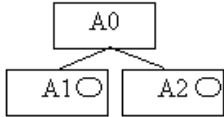
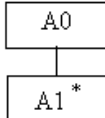
4.2.2 Jackson method

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]



<http://www.vitec.org.vn>

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).

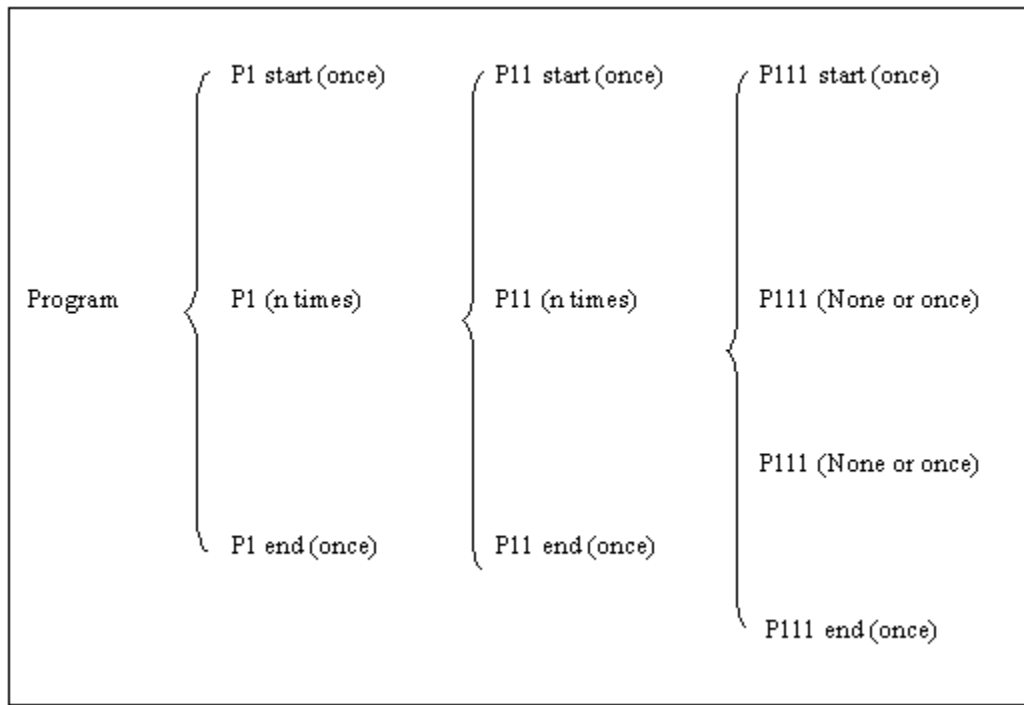
(3) Design the program structure.

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.

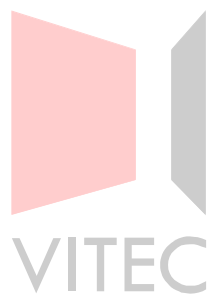
The design of services is the most important aspect of object orientation. A service is the behavior of an object, and this behavior determines the functions implemented by the system. Contract models are used in service designs. In this procedure, the contract concluded between the server and a client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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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.

VITEC

<http://www.vitec.org.vn>

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



<http://www.vitec.org.vn>

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.

VITEC

<http://www.vitec.org.vn>

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.

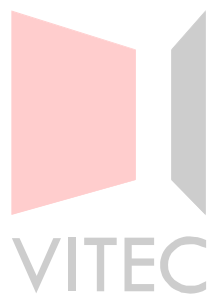
5.1 Procedural Programming Paradigm

5.2 Logic Programming Paradigm

5.3 Functional Programming Paradigm

5.4 Object-oriented Programming Paradigm

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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.

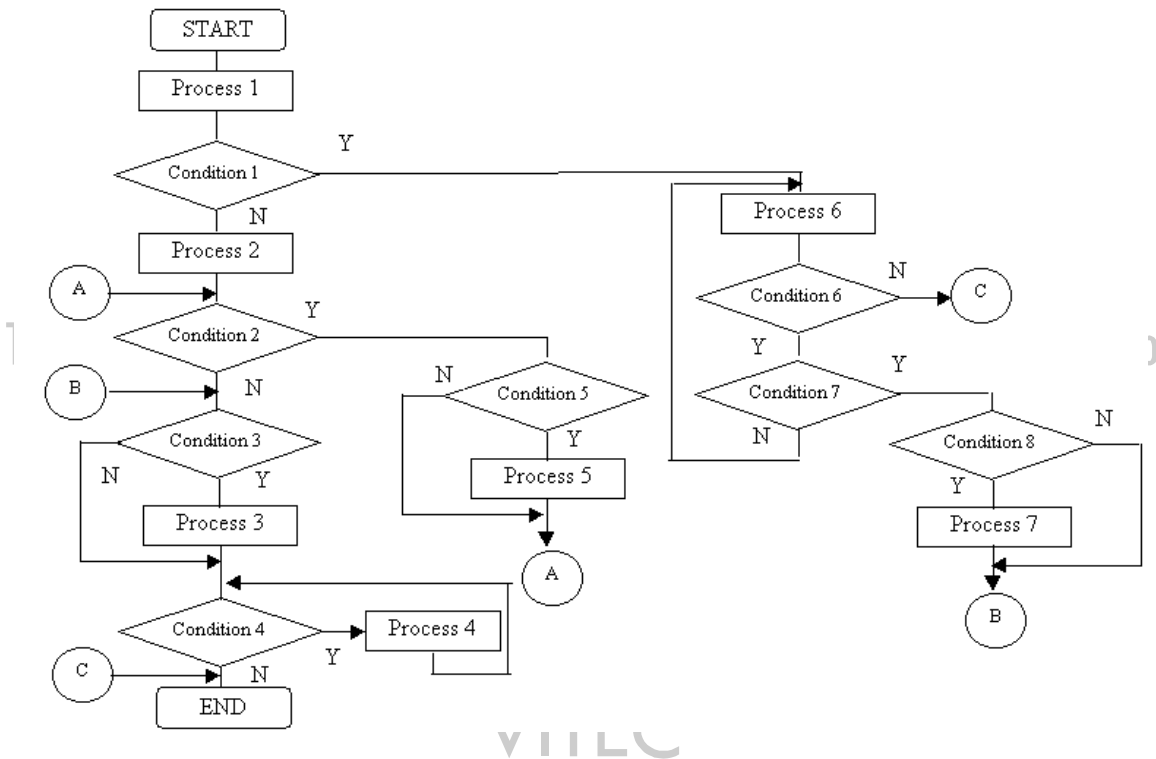
<http://www.vitec.org.vn>

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).

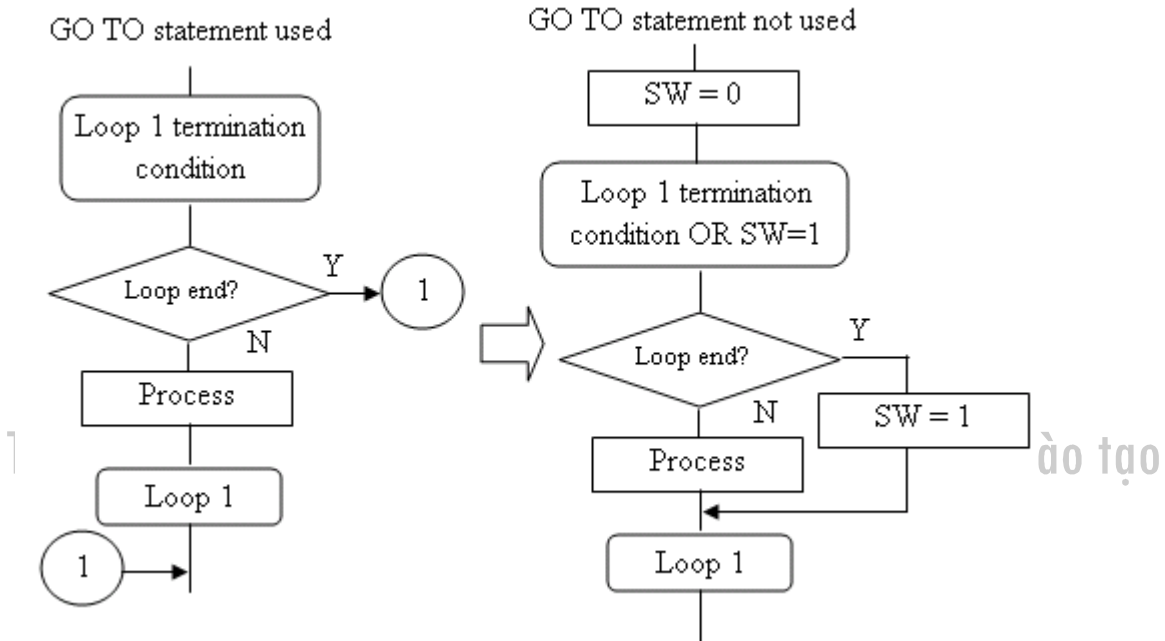


This example shows how GO TO statements make the processing control structure complicated.

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.

VITEC

<http://www.vitec.org.vn>

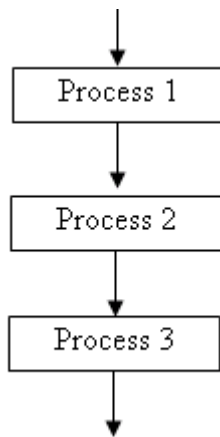
(3) Basic control constructs

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

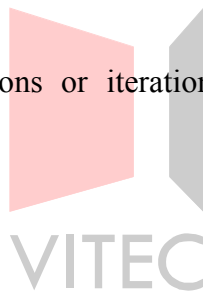
These basic control constructs include sequence, selection, and iteration. They control the processing flow.

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

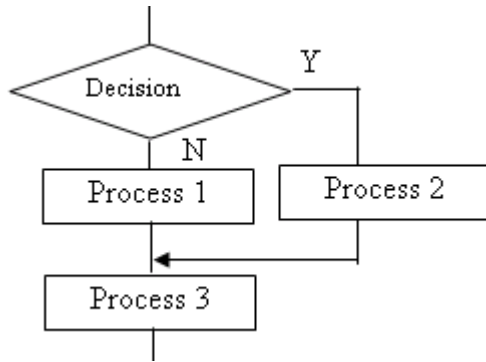


Sequences do not include selections or iterations, and processes are executed in sequence from the top down.



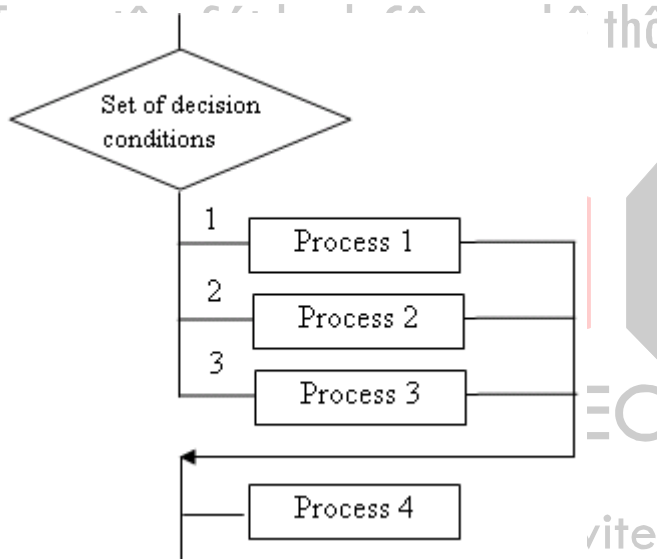
<http://www.vitec.org.vn>

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.

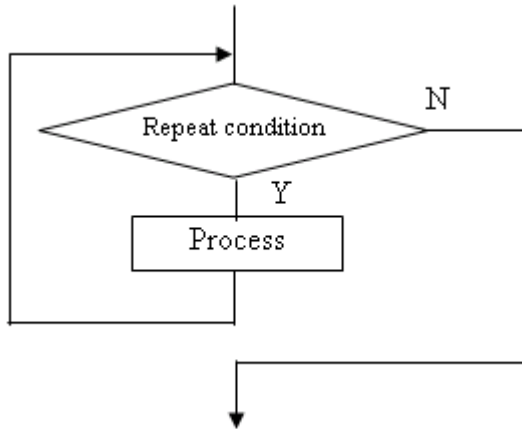
3) Selection based on multiple conditions (multiple-way branch)



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.)

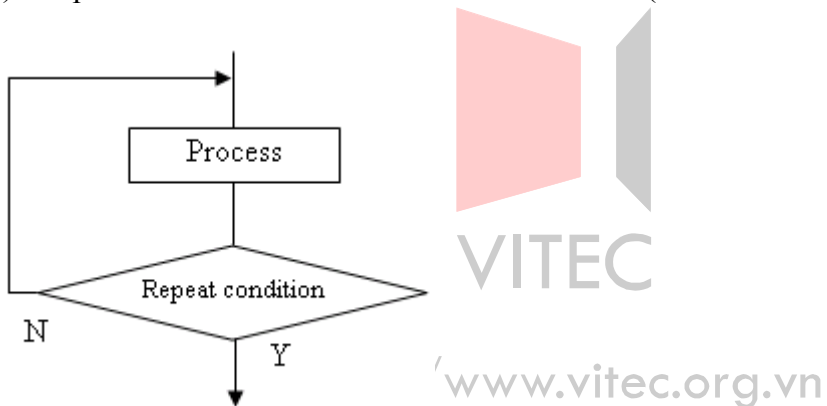
- If the value of the item is 1, process 1 and process 4 are executed in that order.
- If the value of the item is 2, process 2 and process 4 are executed in that order.
- If the value of the item is 3, process 3 and process 4 are executed in that order.

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.

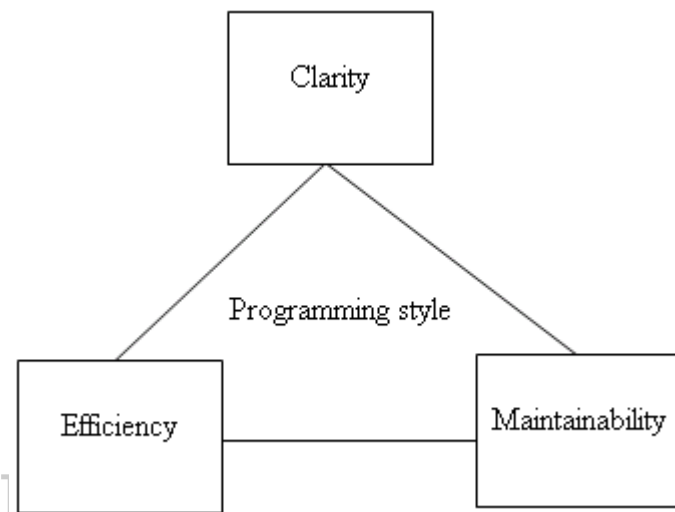
5) Loop with a termination condition at the bottom (DO UNTIL loop)



- 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;
}
```

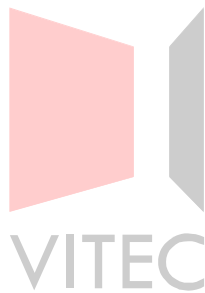
2) Efficiency

- 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.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

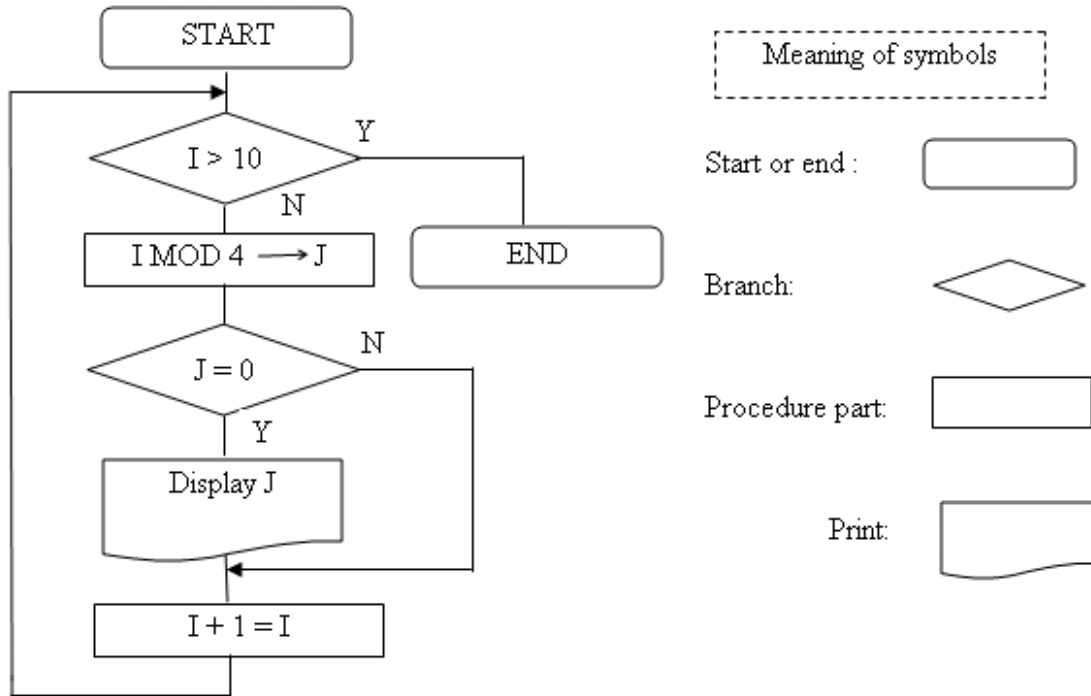
5.1.2 Structured charts

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.

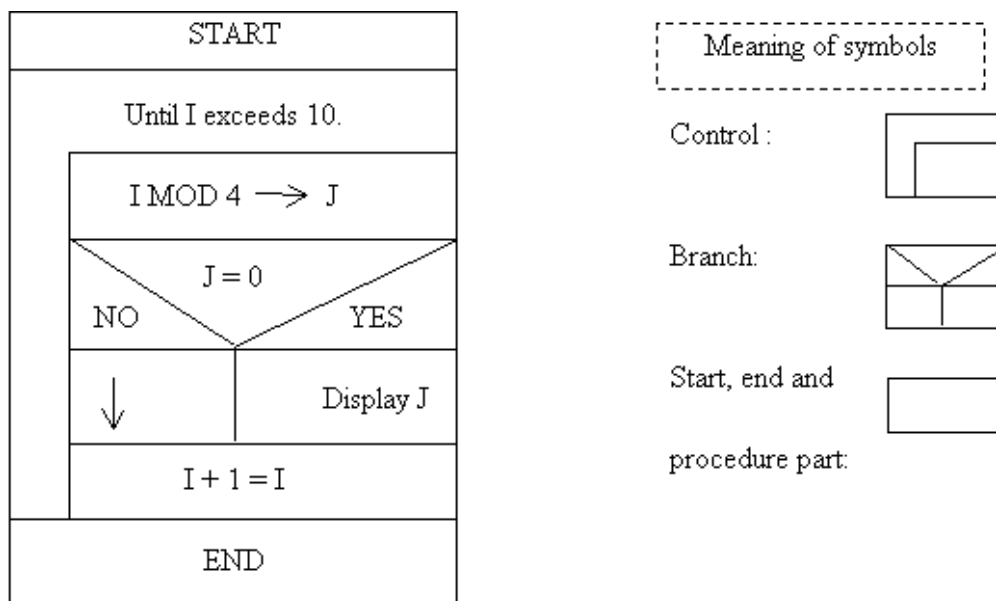
(1) Flowchart

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



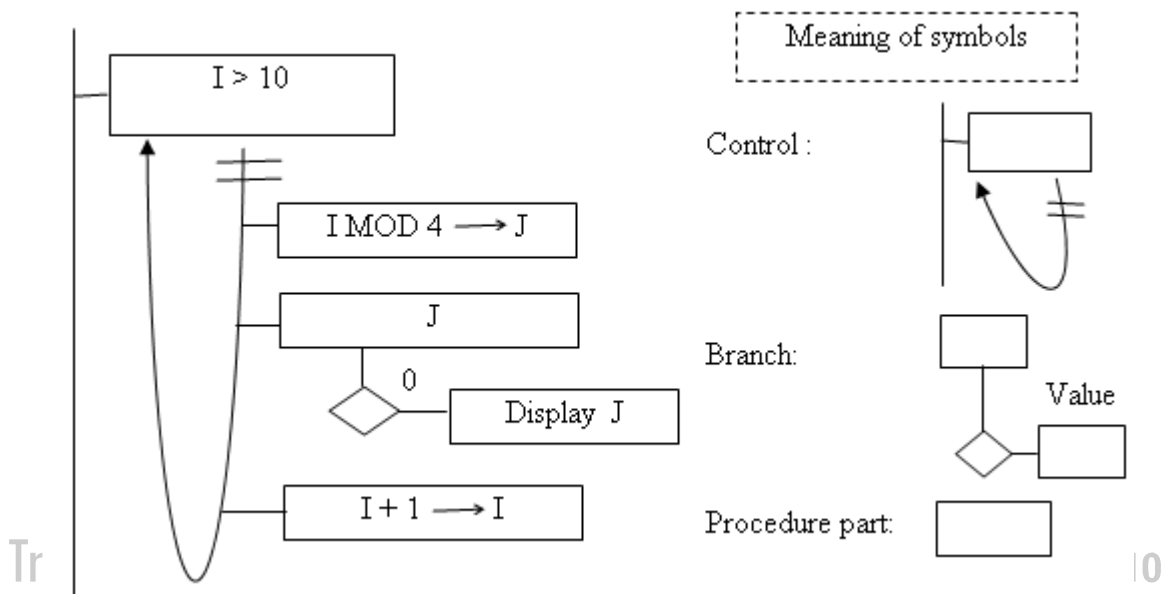
(2) NS chart (Nassi-Schneiderman Chart)

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



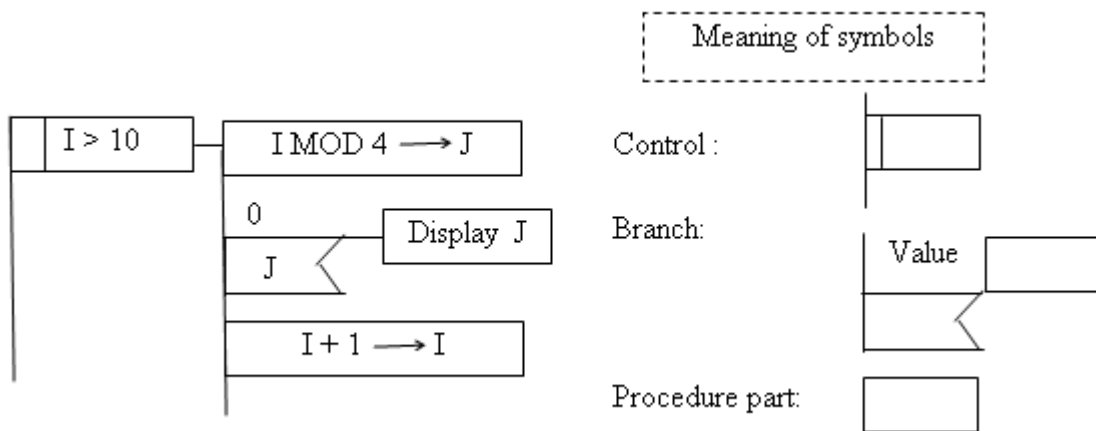
(3) DSD (Design Structure Diagrams)

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



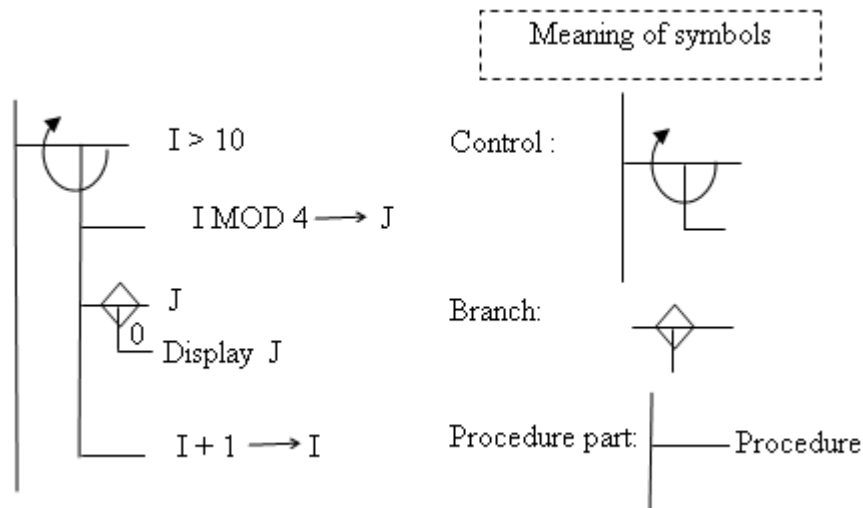
(4) PAD (Problem Analysis Diagrams)

PAD is a structured chart that was standardized by Hitachi.



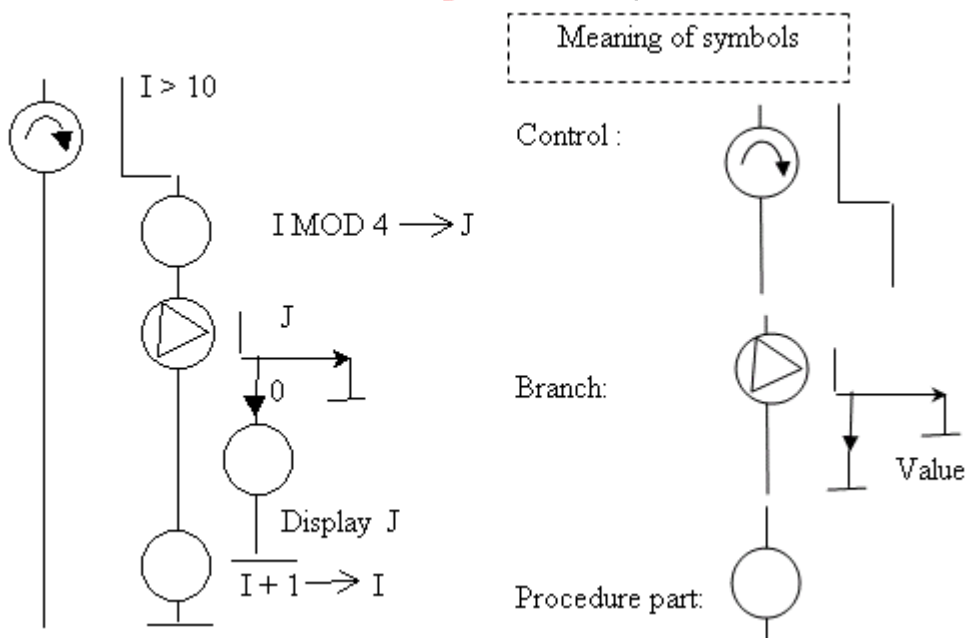
(5) SPD (Structured Programming Diagrams)

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



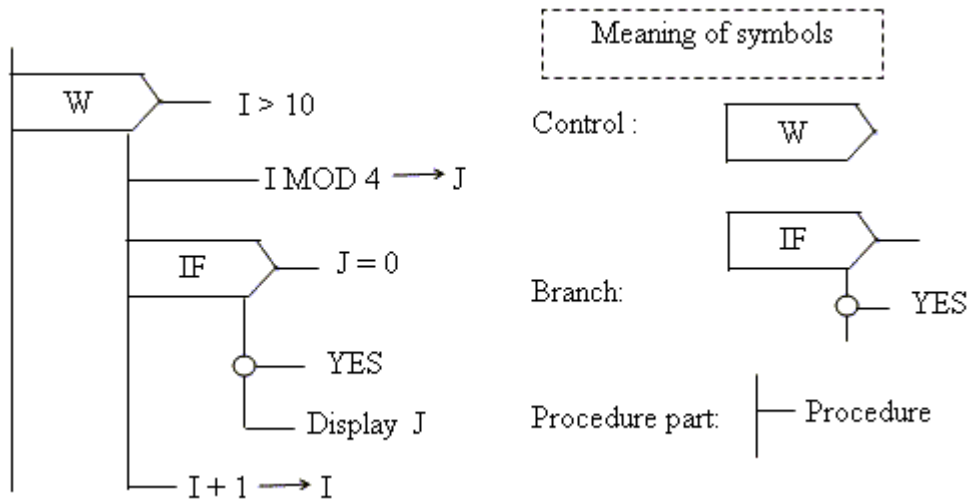
(6) HCP (Hierarchical and Compact Description Chart)

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



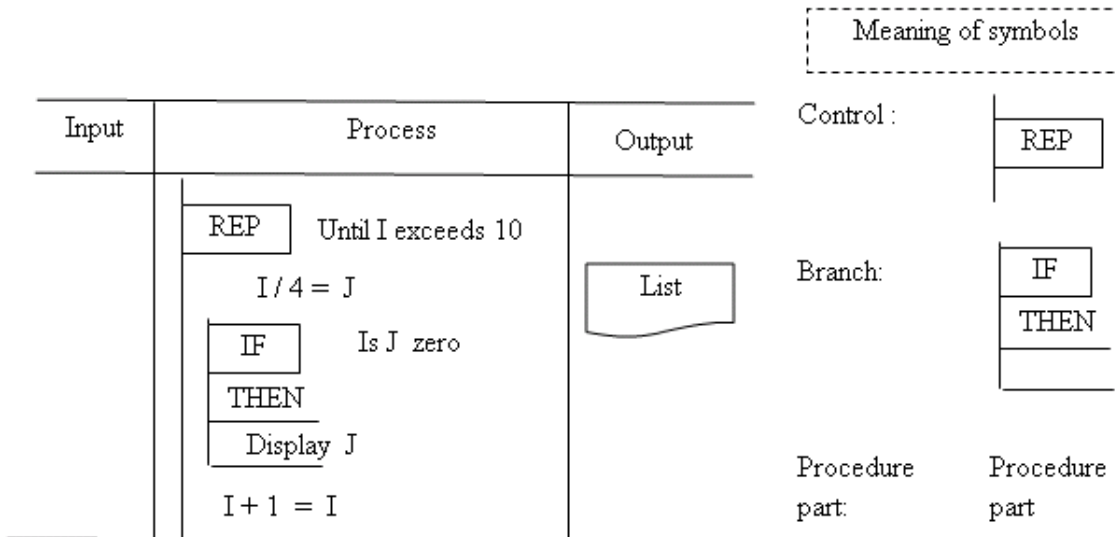
(7) YAC (Yet Another Control Chart)

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 \Leftarrow 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). \Rightarrow All animals sleep.

2) Fact (conclusion)

ANIMAL(CAT). \Rightarrow The cat is an animal.

3) Query

? - ANIMAL(A). \Rightarrow 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).

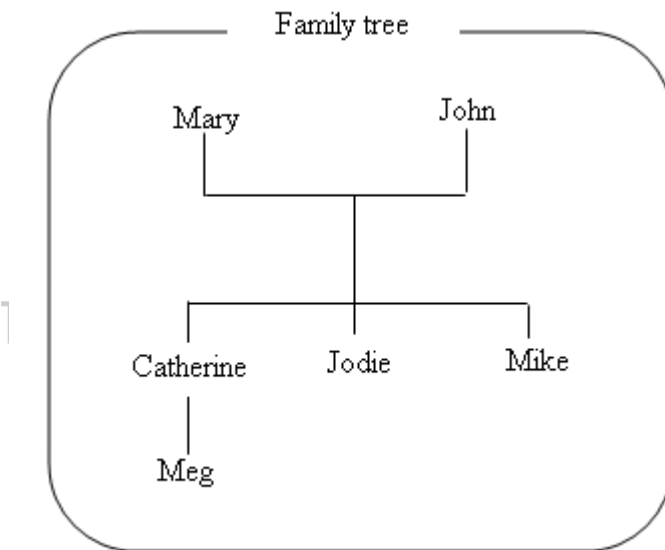
Thus, SLEEP(CAT). => We can thus derive "the cat sleeps."

- 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.

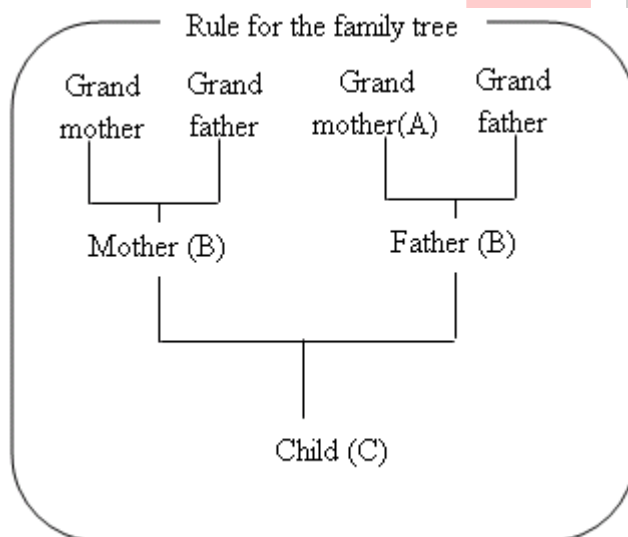
5.2.2 Unification

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).

ng tin và Hỗ trợ đào tạo



c.org.vn

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.

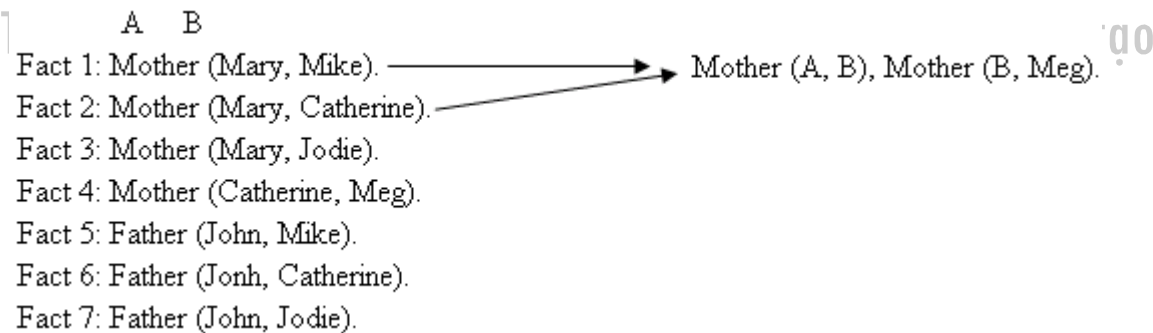
When Fact 1 is matched to Rule 1, Rule 1 becomes as follows:

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.)



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:

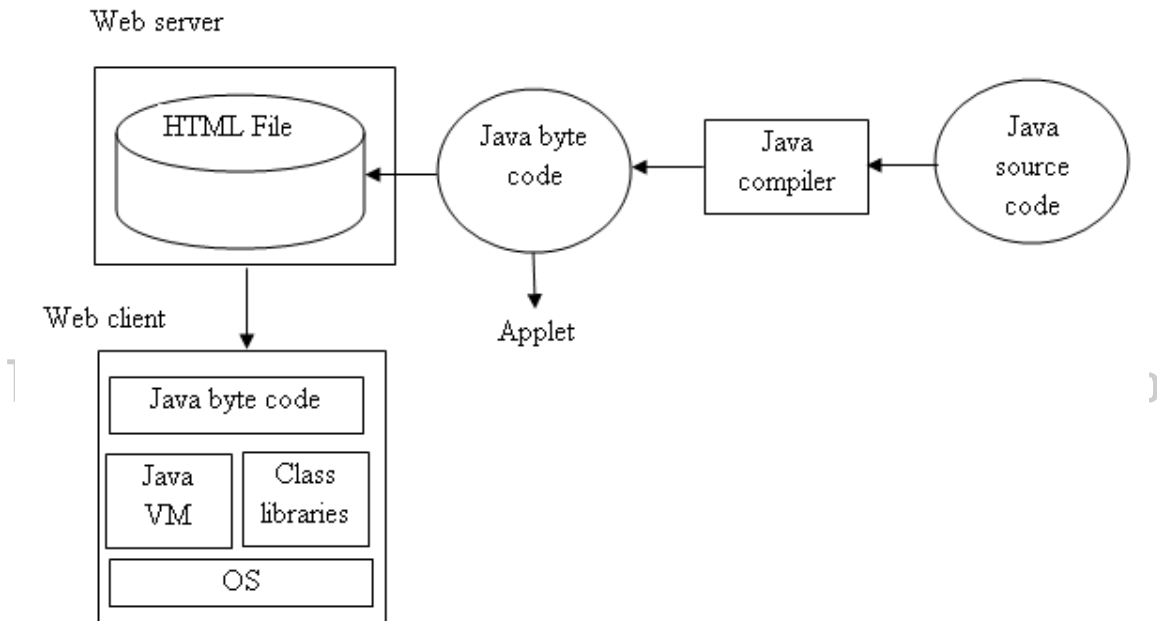
Result		z => There are two z characters because z represents 1000.
zyxwvvvt		y => There is one y character because y represents 500.
		x => There is one x character because x represents 100.
		w => There is one w character because w represents 50.
		v => There are three v characters because v represents 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++.

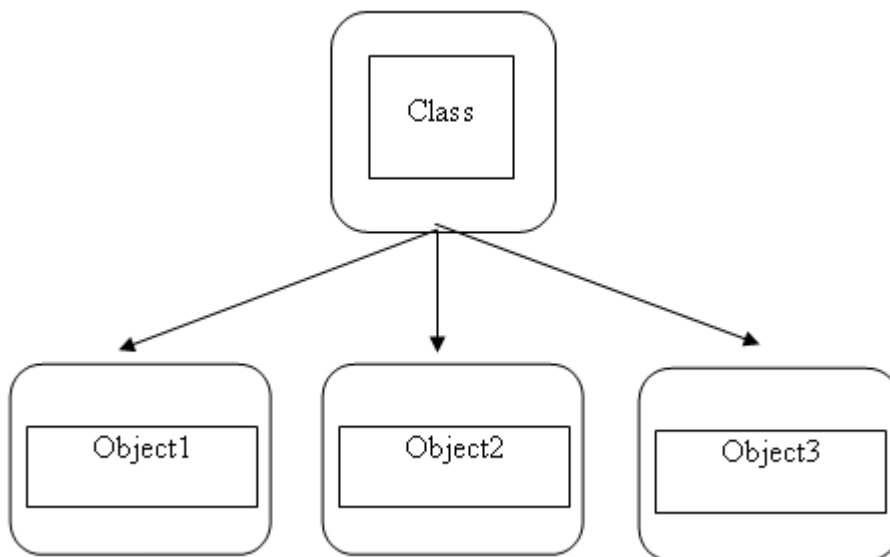
(3) JDK (Java Development Kit)

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.

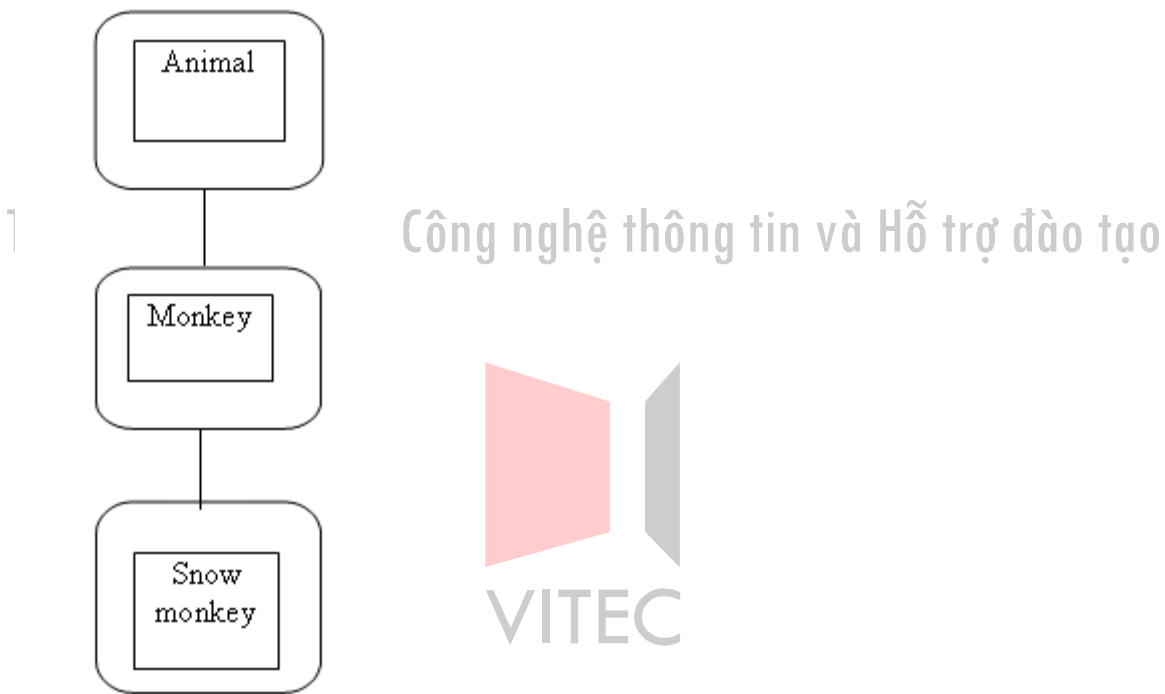


- A class is a modular unit that describes the common properties of a set of members (objects). Classes are selected under control of applications and include only important properties defined.

- 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).

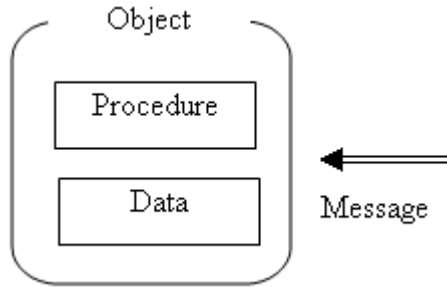
(2) In the above class hierarchy, one can state the following:

- 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

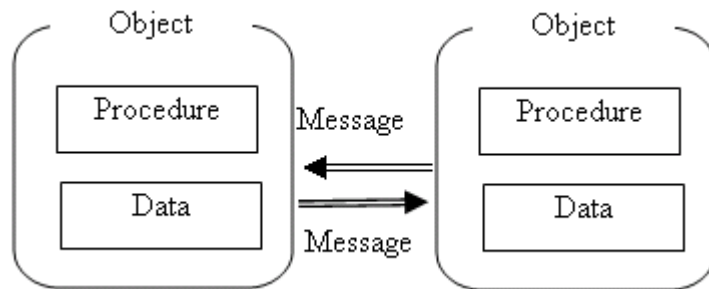
Encapsulation refers to combining data and the procedure for data handling in one package. The encapsulated data and procedure cannot be directly handled from outside.



(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



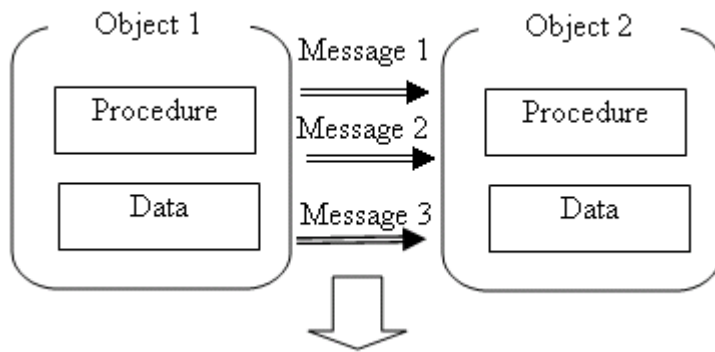
(1) Message passing

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.

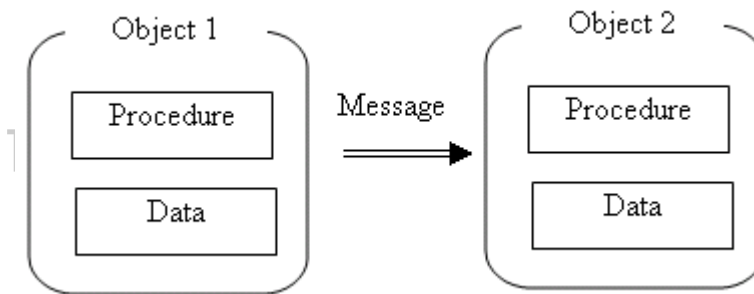
1) Message passing is a means for communication between processes. In the above example, messages are objects.

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.

<http://www.vitec.org.vn>

Exercises

- 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.

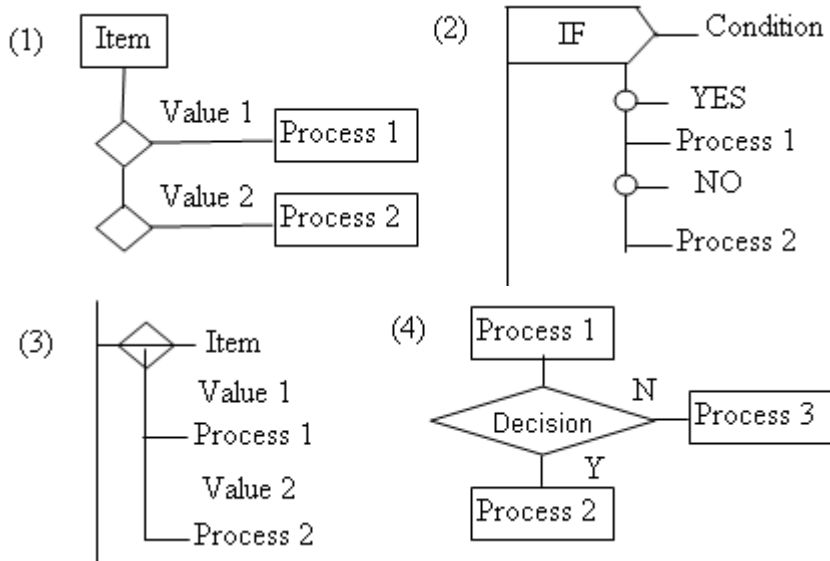
Answers:

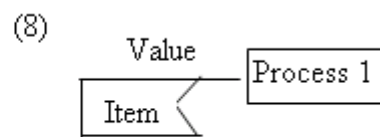
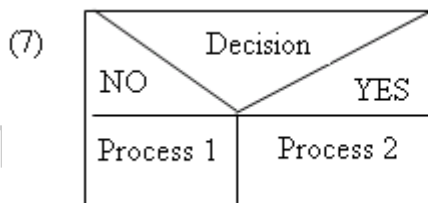
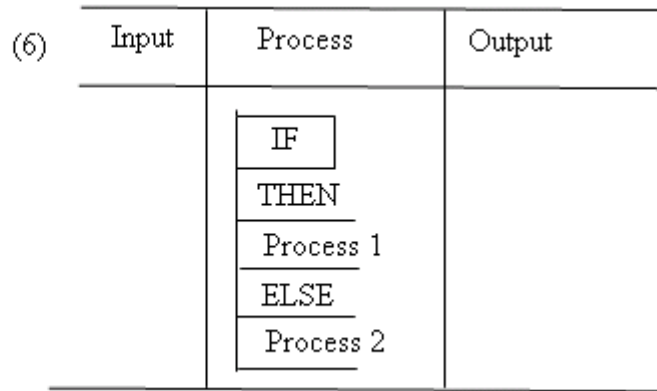
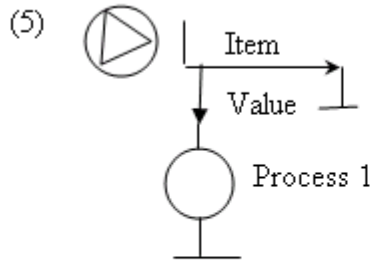
- selections
- loops with the termination condition at the bottom
- sequences
- multiple-way branches
- loops with the termination condition at the top
- GO TO statements
- iterations

- 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.

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





Answers:

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



<http://www.vitec.org.vn>

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:

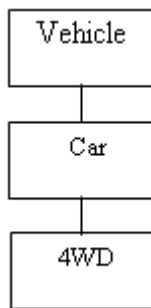
- a. CONTINUE statements
- b. processing statements
- c. operators
- d. object-oriented features
- e. numeric type
- f. data types
- g. control statements
- h. class variables
- i. BREAK statements
- j. encapsulation



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

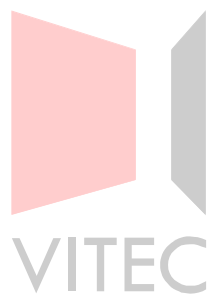
This chapter explains software quality control and testing techniques.

6.1 Software Quality Factors

6.2 Program Testing

6.3 Quality Control of Software

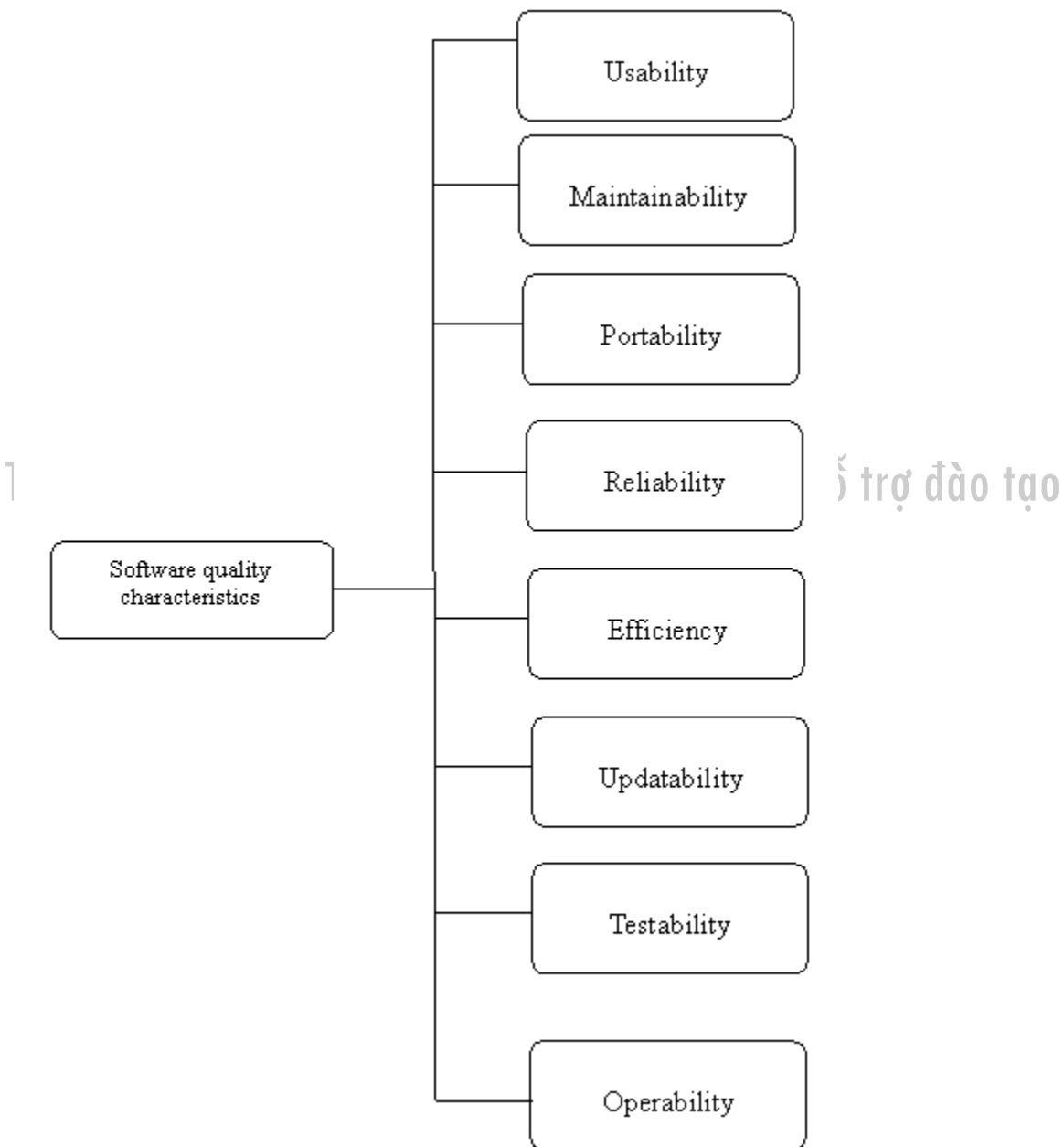
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

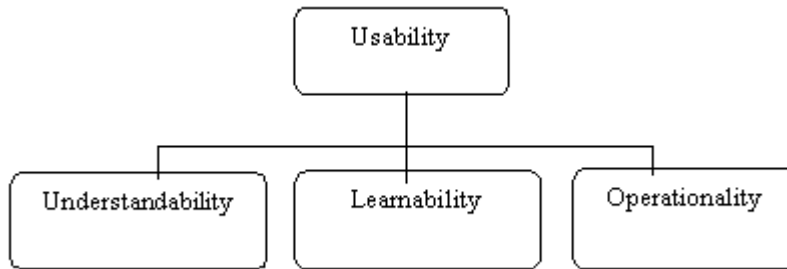
6.1 Software Quality Factors

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:

$$\text{Degree of fulfillment in the applied model} = \frac{\text{Number of functions clearly explained in the model}}{\text{Number of all functions}}$$

(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:

$$\text{Operation familiarization} = \frac{\text{Time required for the user to become competent in using the software}}{\text{Number of all functions}}$$

(3) Operability

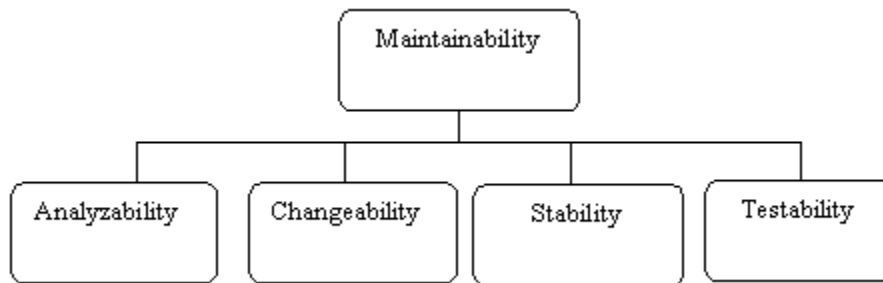
Operability expresses the effort required by the user to conduct operation control and operation of software. As a measure of operability, the average operation error interval and the responsive time are used.

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

$$\text{Responsive} = \frac{\text{The time when the first response was received from the system minus the time when the entry by the operator finished}}{\text{Number of error occurrences in operation}}$$

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.

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

$$\text{Diagnostic function installation ratio} = \frac{\text{Number of diagnostic functions already installed}}{\text{Number of diagnostic functions to be installed}}$$

(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.

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

<http://www.vitec.org.vn>

(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.

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

(4) Testability

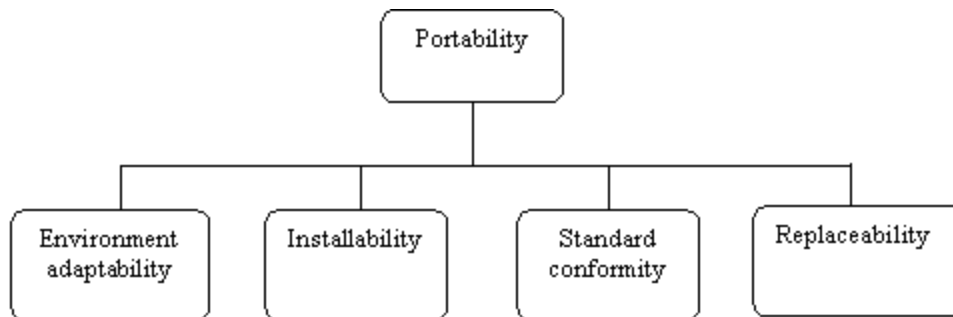
Testability expresses how easy it is to validate the revised software.

As a measure of testability, the test time per line of changed source code is used.

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

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.

$$\text{Adaptable machine type rate} = \frac{\text{Number of adaptable machine types}}{\text{Number of specified machine types}}$$

(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.

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

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

(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.

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

ig tin và Hỗ trợ đào tạo

(4) Replaceability

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

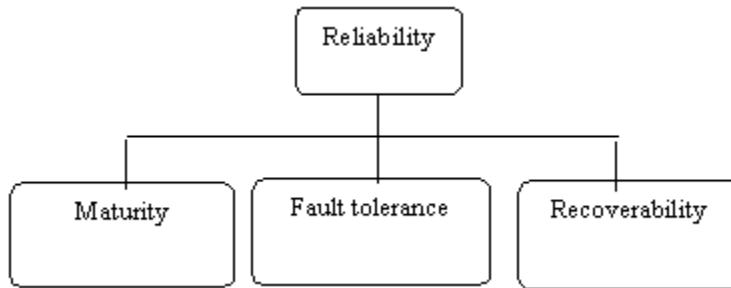
As a measure of replaceability, the function change rate and performance maintenance rate are used.

$$\text{Function change rate} = \frac{\text{Number of functions changed during porting}}{\text{Number of ported functions}}$$

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

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.

$$\text{Average failure interval} = \frac{\text{Total operation time of the software system}}{\text{Failure count}}$$

(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.

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

$$\text{Error input/error operation detection rate} = \frac{\text{Number of detected error inputs/error operations}}{\text{Number of recorded error inputs/error operations}}$$

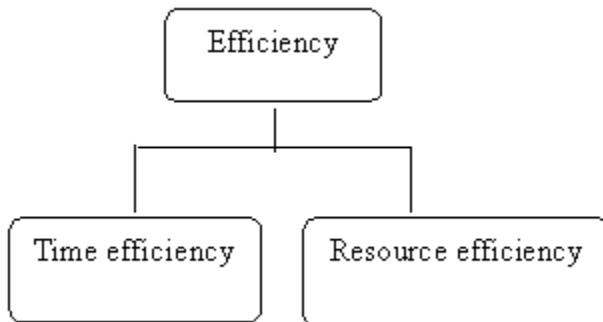
(3) Recoverability

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

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

6.1.5 Efficiency

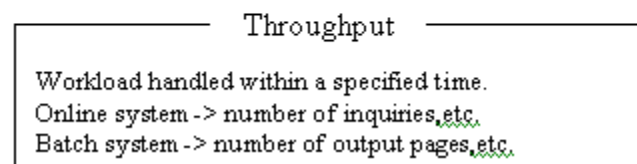
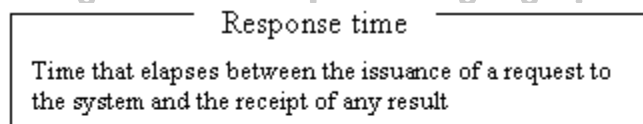
Efficiency expresses software performance in relation to the amount of resources used under certain conditions. Resources include other software products, hardware facilities, and services from personnel. Efficiency can be further divided into two sub-factors.



(1) Time efficiency

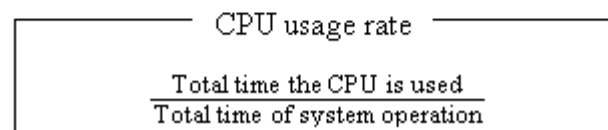
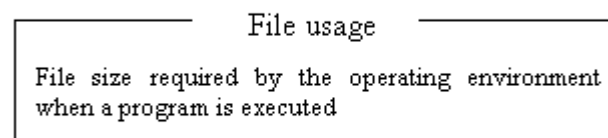
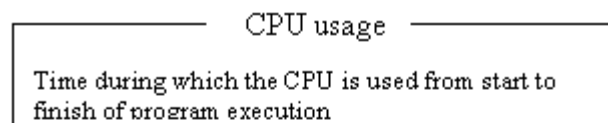
Time efficiency expresses the response time and the throughput within a specified time.

As a measure of time efficiency, response time and throughput are used.



(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.



6.1.6 Updatability

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

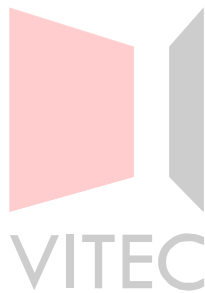
6.1.7 Testability

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.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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): $1 \leq \text{attendance number} \leq 10$

Invalid equivalence class (1): $\text{attendance number} < 1$

Invalid equivalence class (2): $\text{attendance number} > 10$

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.

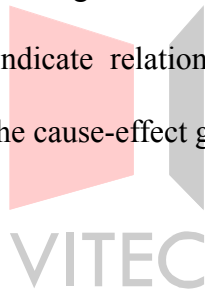
3) Cause-effect graph

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:

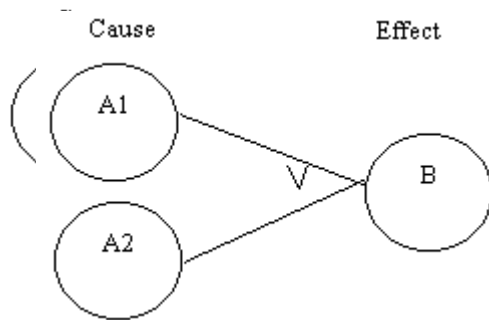
- Assign a number for identifying the input data within the input specification conditions for the program, and then extract that data.
- Assign a number for identifying the output data within the output specification conditions for the program, and then extract that data.
- Create a cause-effect graph.
- Assign symbols to lines connecting the causes and effects to indicate their relationship.
- Assign symbols to causes to indicate relationships between mutual causes, and indicate constraints among causes.
- Create a decision table based on the cause-effect graphs.



<http://www.vitec.org.vn>

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

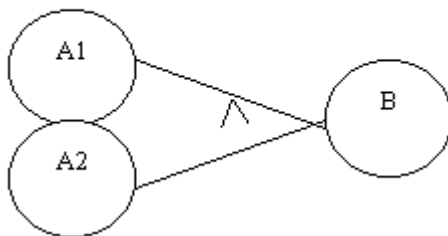
When representing the relationships in accordance with d, equivalence, negation, sum, and product are available.



- Equivalence (If A, then B)

- Sum (If A1 or A2, then B)

T Cause Effect nghệ thông tin và Hỗ trợ đào tạo



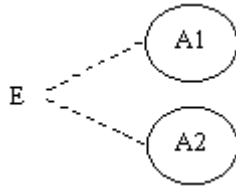
- Negation (If not A, then B)

- Product (If A1 and A2, then B)

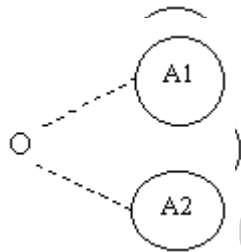
VITEC

<http://www.vitec.org.vn>

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



- Exclusion (either of A1 or A2)

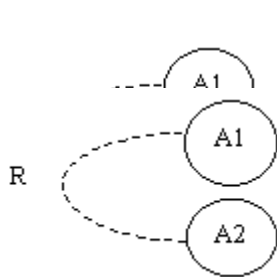


Trung 1

Công nghệ thông tin và Hỗ trợ đào tạo

- 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 A2 does not)

<http://www.vitec.org.vn>

4) Error estimation

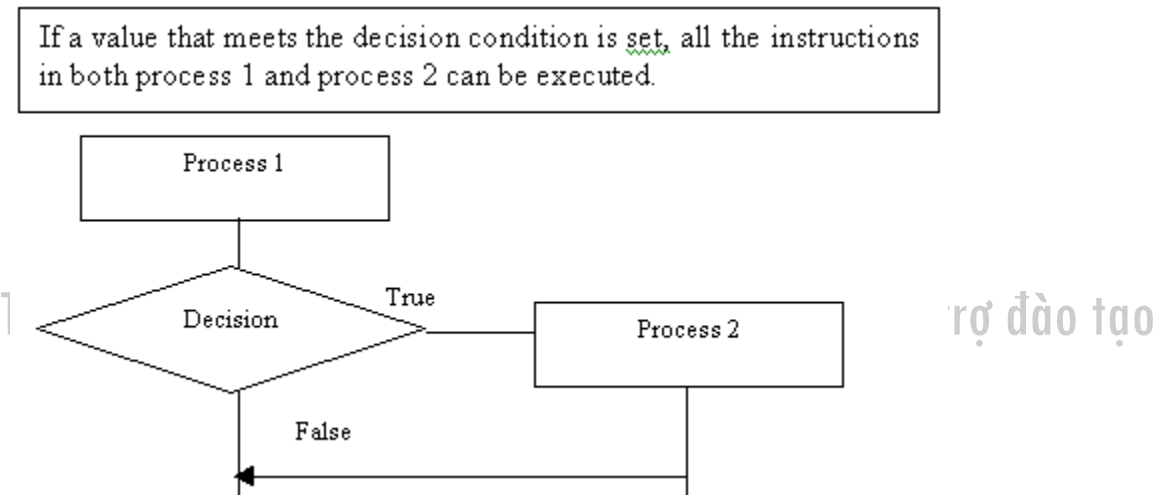
Error estimation is a method of performing tests by preparing test cases for situations in which errors are likely to occur. This method has not yet been established as commonly used test technique.

(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.

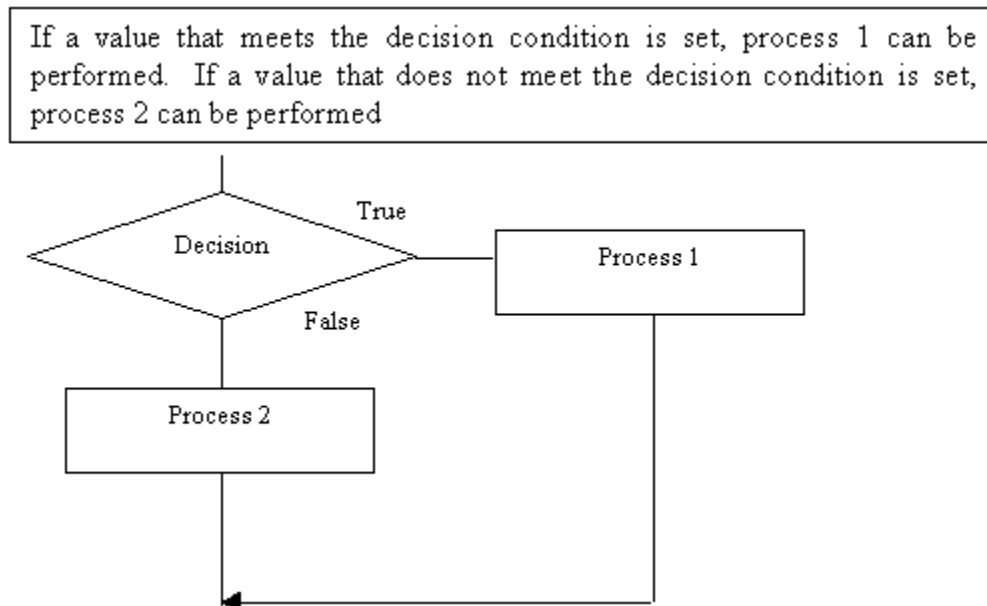
1) Instruction coverage

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



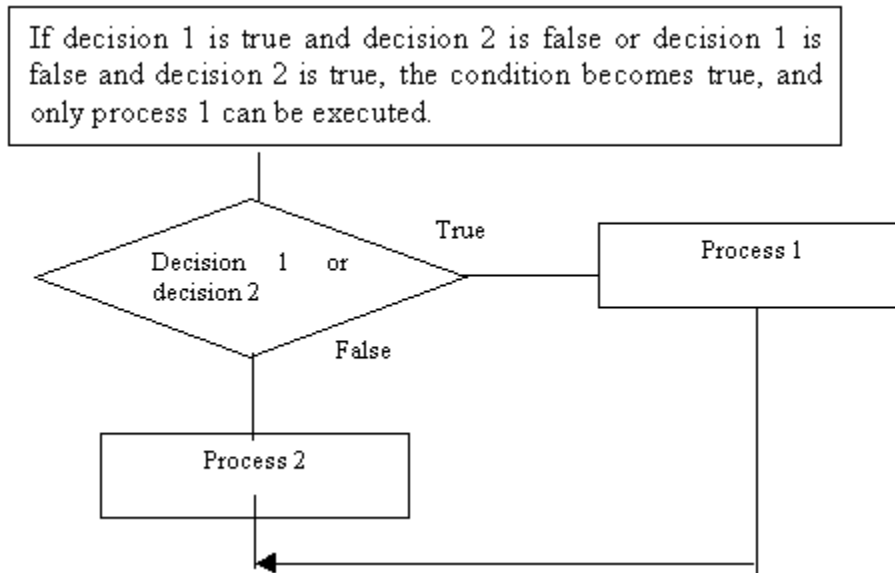
2) Decision condition coverage

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.



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.



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.

<http://www.vitec.org.vn>

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.

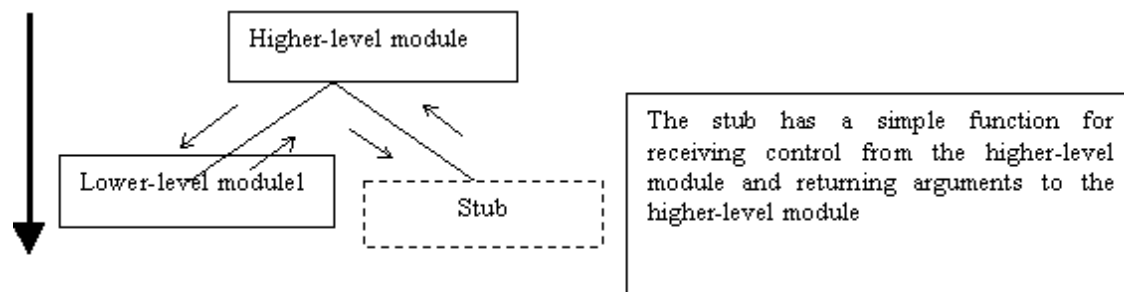
(3) Incremental test

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.

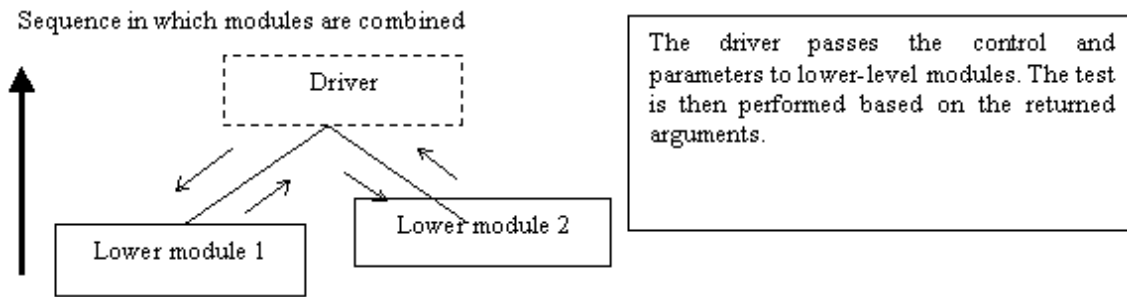
Sequence in which modules are combined



The 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

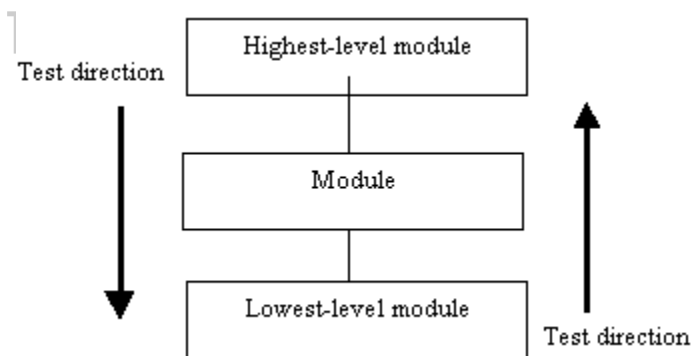
The bottom-up test is performed by sequentially combining lower-level modules with higher-level modules. If a higher-level module were to have to be combined before it was actually finished, the test is performed by replacing the higher-level module with a virtual module called a driver.



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.



ng tin và Hỗ trợ đào tạo

VITEC

<http://www.vitec.org.vn>

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.

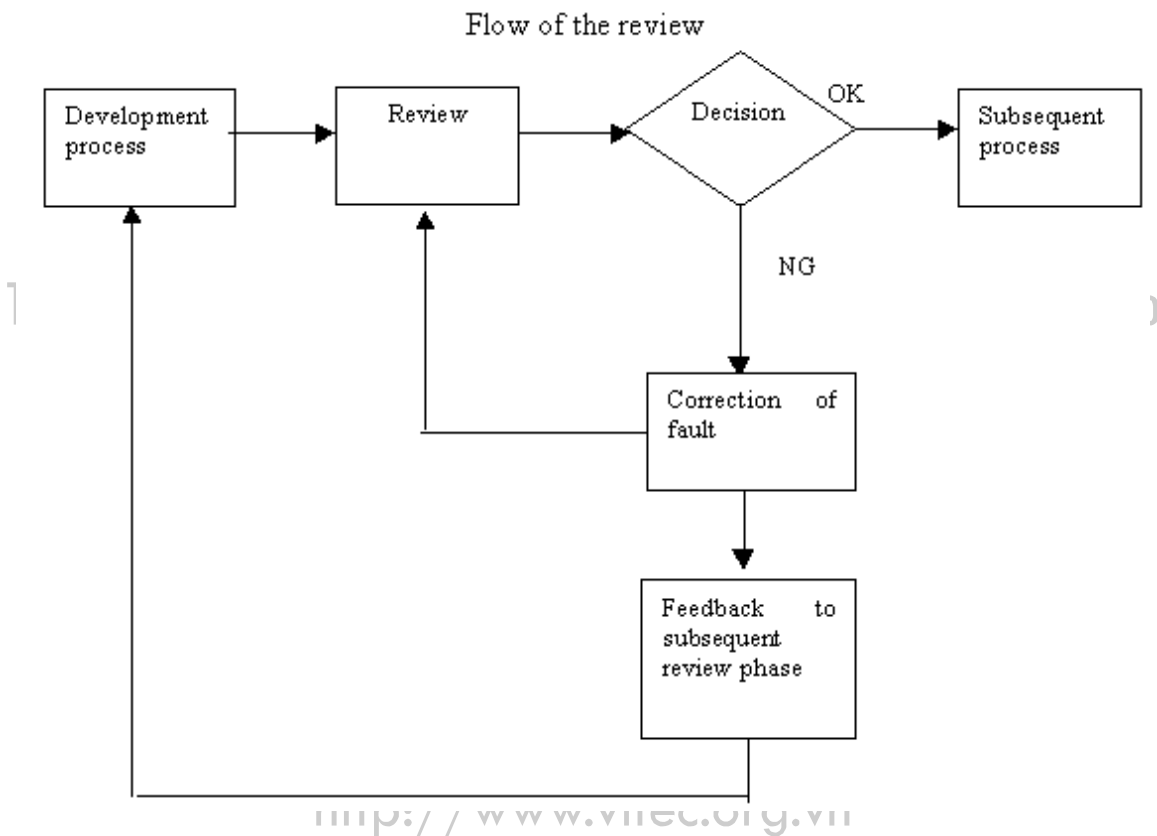
6.3 Quality Control of Software

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.



(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.

<http://www.vitec.org.vn>

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.

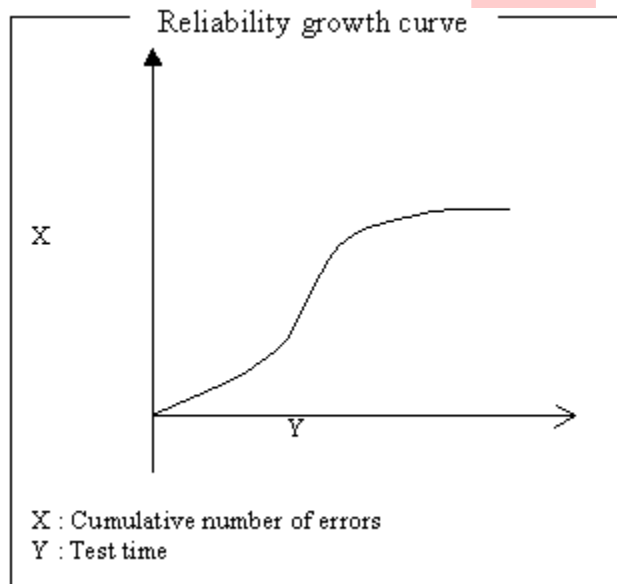
(2) Reliability growth model

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

VITEC

<http://www.vitec.org.vn>

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 test.

- (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.

VITEC

<http://www.vitec.org.vn>

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

VITEC

<http://www.vitec.org.vn>

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

VITEC

<http://www.vitec.org.vn>

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?

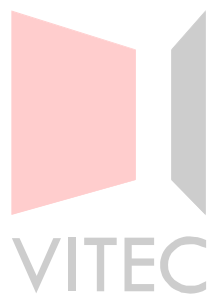
Answers:

- 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.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

7 Software Development Environment

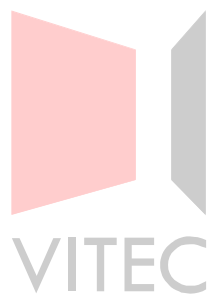
Chapter Objectives

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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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.

<http://www.vitec.org.vn>

(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.

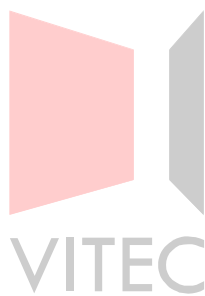
(3) Tool management tool

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.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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

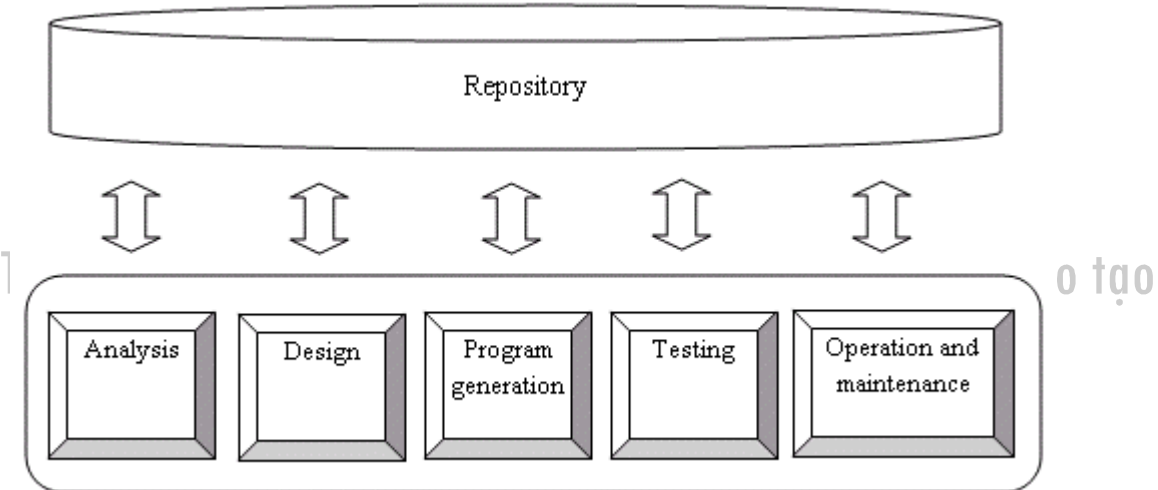
A repository is defined as a store or warehouse, and it provides different kinds of information generated during software development by standardizing and managing that

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

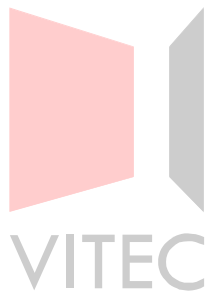
Re-engineering is a method to construct a system that is similar to an existing one by partially modifying it, based on information obtained by reverse engineering.

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

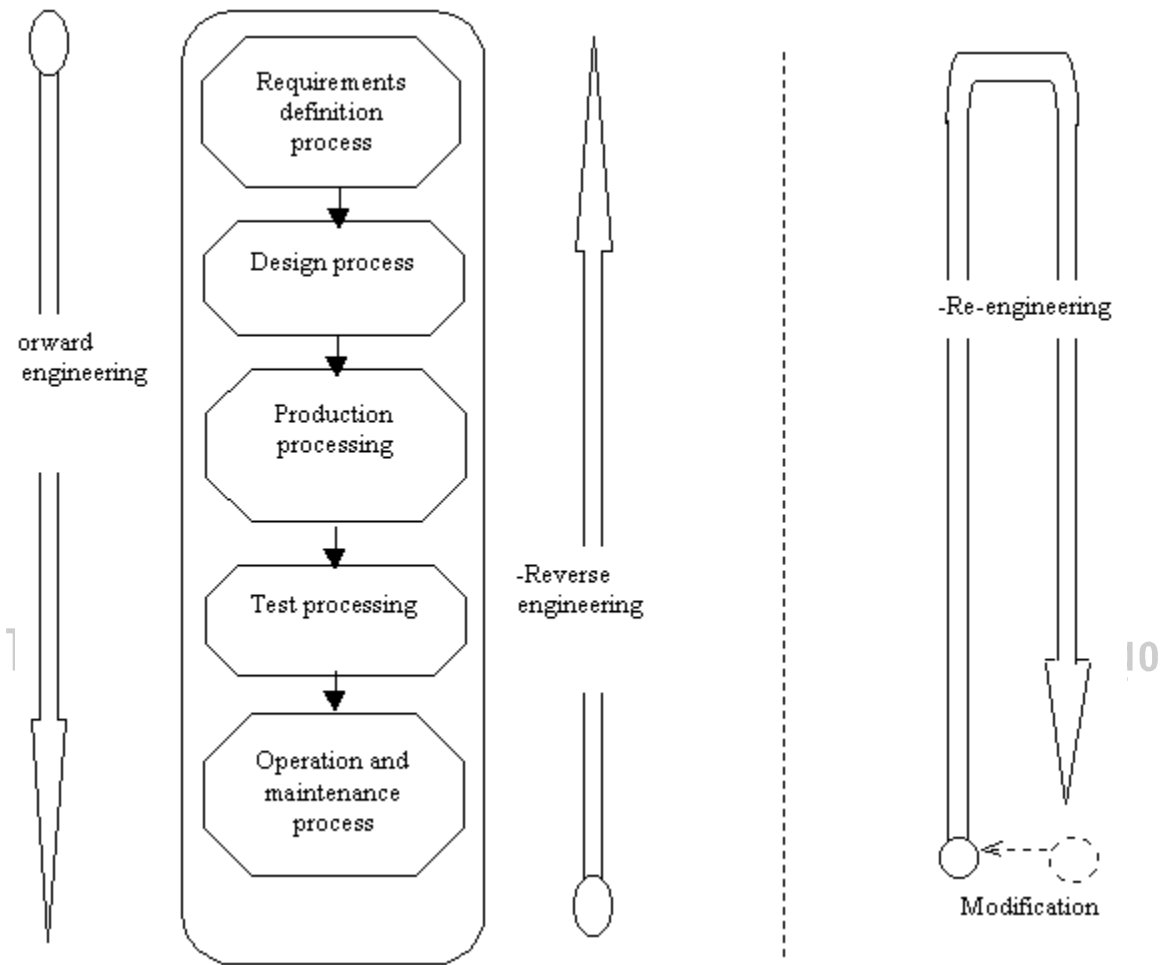
The concrete procedure for re-engineering is as follows:

- (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.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>



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:

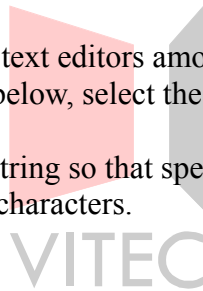
- a. The integrated-CASE
- b. The upper-CASE
- c. CASE tools
- d. The lower-CASE

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

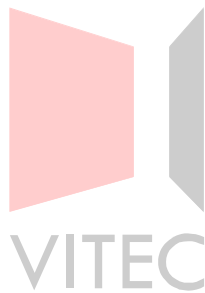


<http://www.vitec.org.vn>

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



<http://www.vitec.org.vn>

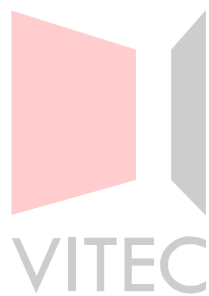
8 Trends in Software Engineering

Chapter Objectives

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

8.1 Software Tools

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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 Européen pour la Recherche Nucléaire (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 be used as a 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

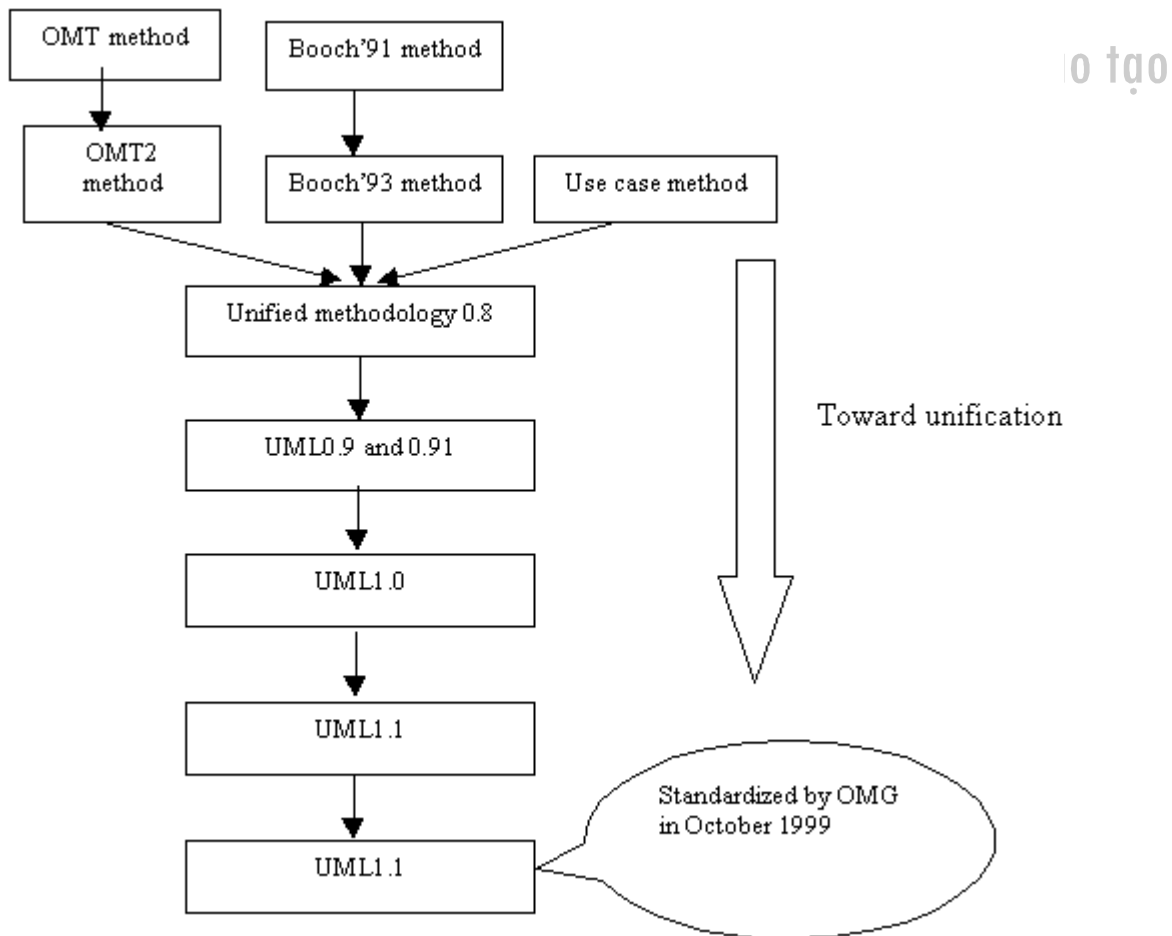
OS, which has proved its suitability as real-time OS for information appliances, is

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

<http://www.vitec.org.vn>

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

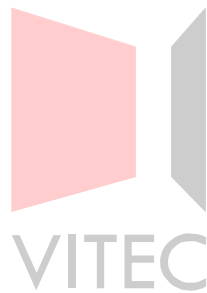
<http://www.vitec.org.vn>

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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

1 External Design Procedures

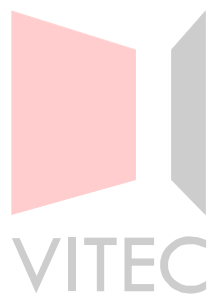
Chapter Objectives

This chapter explains the necessary preparations and procedure for external design.

1.1 Preparations for External Design Work

1.2 External Design Procedures

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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:

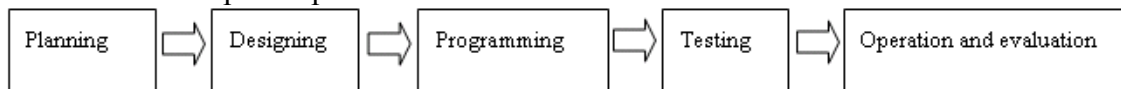
- As an enterprise becomes larger, the amount of data to be processed increases.
- A enterprise has to process data faster to improve its customer service.
- A enterprise requires a variety of information to make more appropriate decisions.

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.

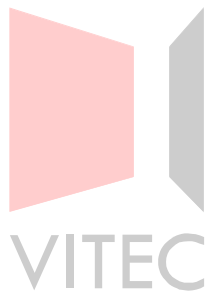
1.1.2 Systematization requirements definition

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.



<http://www.vitec.org.vn>

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.

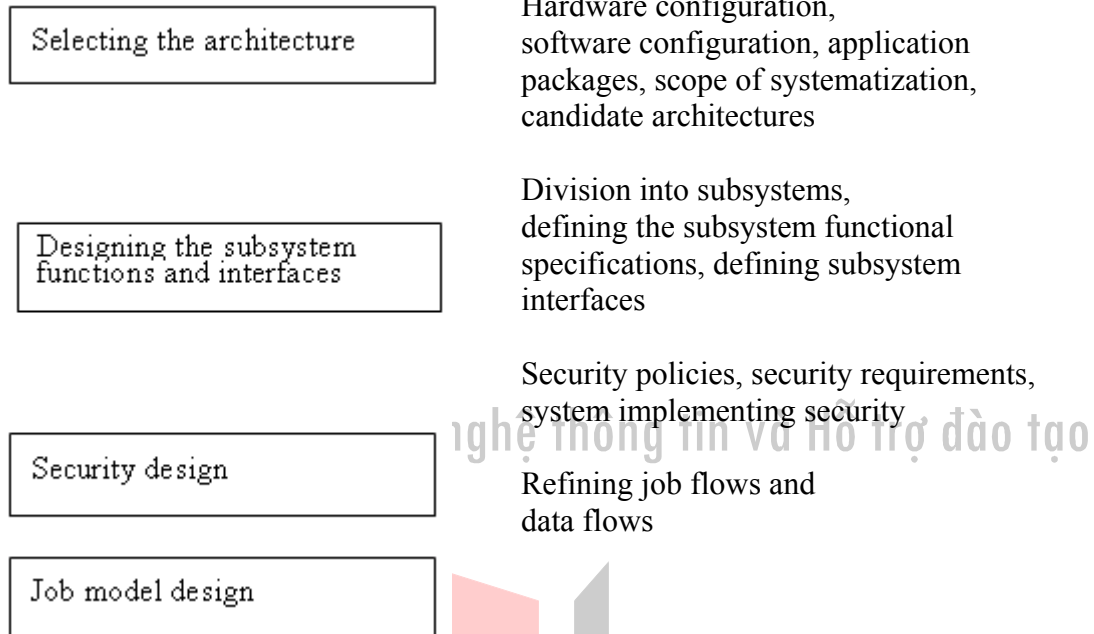
- **What are the objectives of the system?**

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

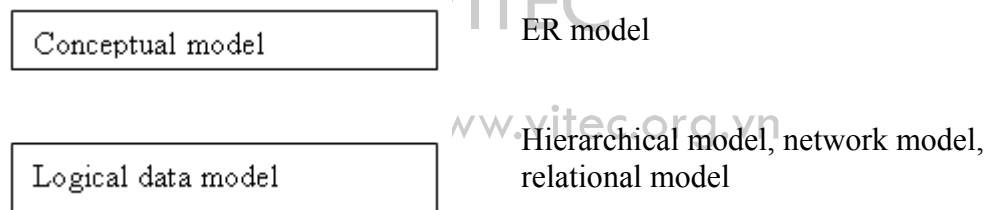
1.2.1 Designing the system functions

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.



1.2.2 Designing the data models

The design of the data models includes the data used in the system.



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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

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.



<http://www.vitec.org.vn>

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.

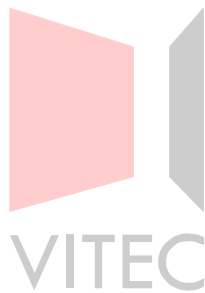
2.1 Selecting a System Architecture

2.2 Subsystem Function Specifications and Interface Design

2.3 Security Design

2.4 Job Model Design

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



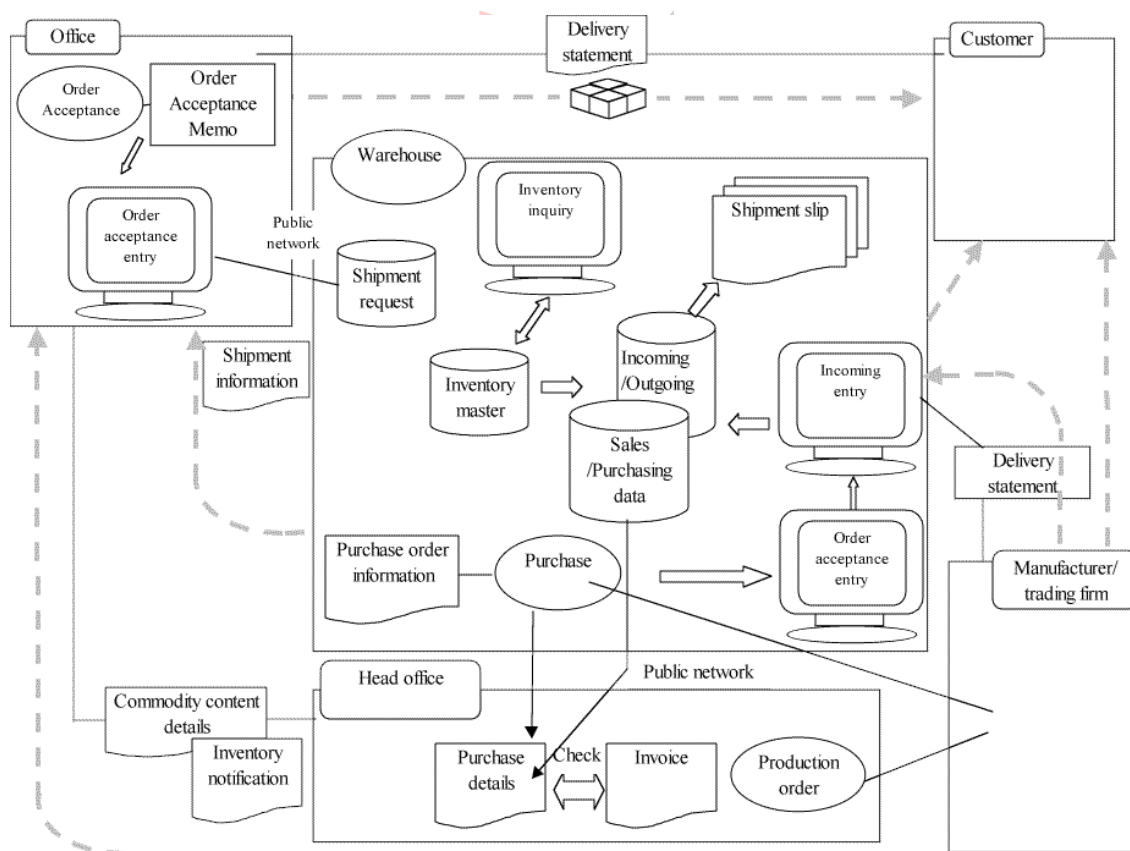
<http://www.vitec.org.vn>

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	<ul style="list-style-type: none"> - Server equipment configuration (e.g., CPU and memory capacity) - Client equipment configuration - Network configuration
Software	<ul style="list-style-type: none"> - OS selection - DBMS selection - Communication middleware selection - Development tool selection - Use of packages
Security	<ul style="list-style-type: none"> - Study of security level
Use of existing assets	<ul style="list-style-type: none"> - 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

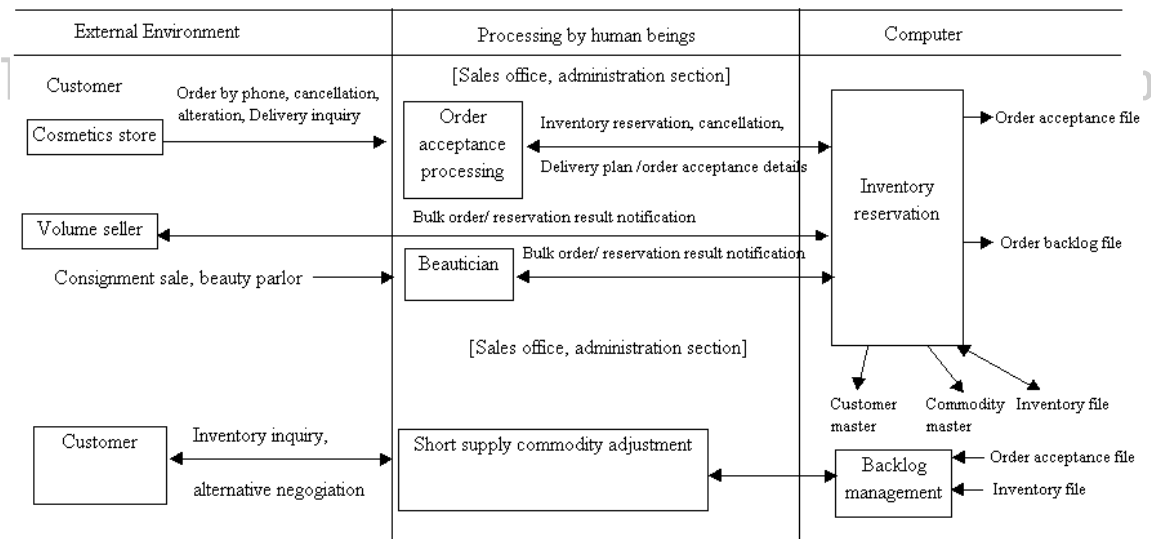
In an analysis process, examine how to create a new job system by analyzing the job requirements for achieving objectives. Summarize the results in a job flow diagram and systematization functional diagram.

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



<http://www.vitec.org.vn>

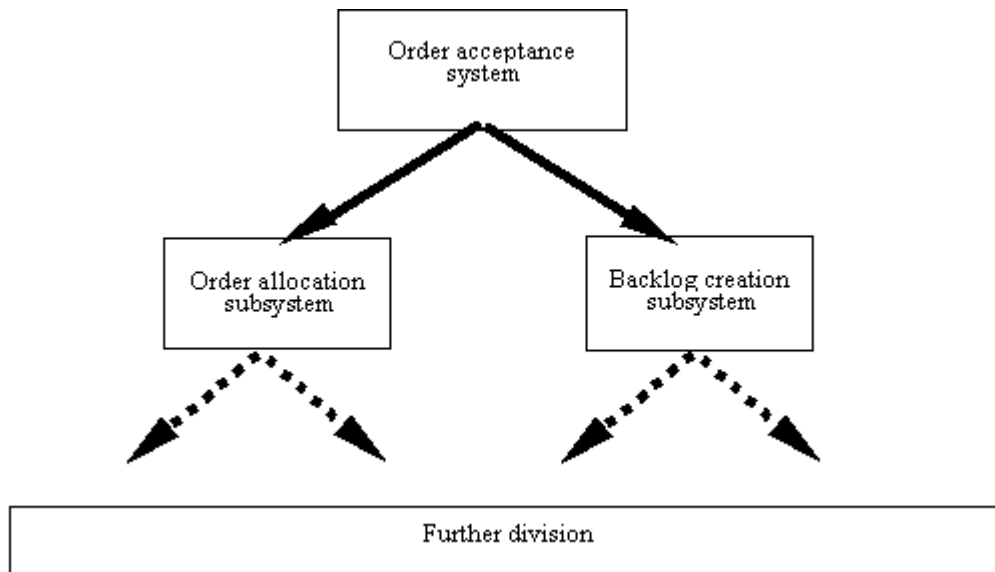
(2) Function of division into subsystems by analyzing the job flow

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]



<http://www.vitec.org.vn>

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

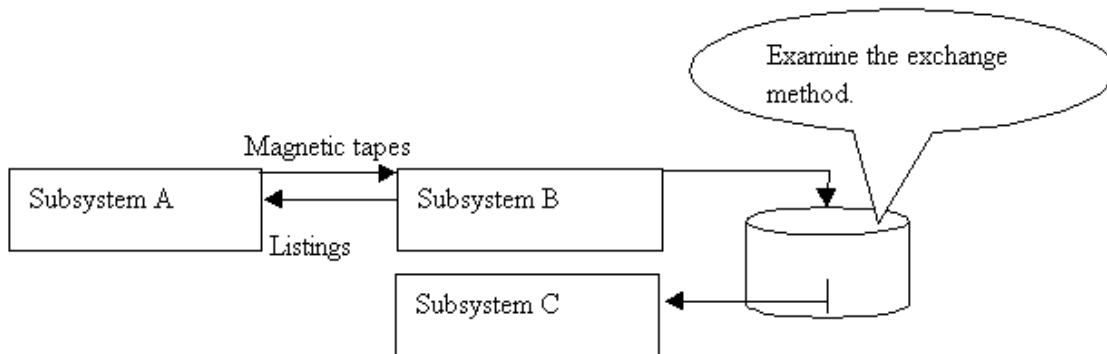
To exchange data between subsystems, the optimum interface configuration must be

found by examining the data.

The resulting configuration should be reflected in the file design and database design.

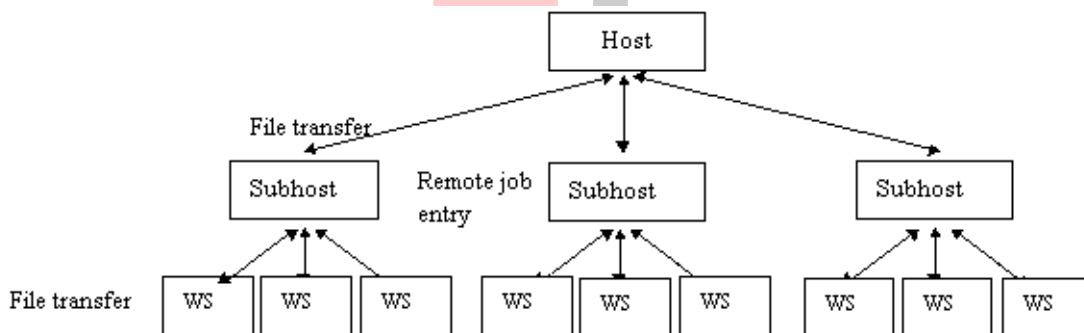
(1) Linkage to other job systems and subsystems

- Examine the method of exchanging data between systems.



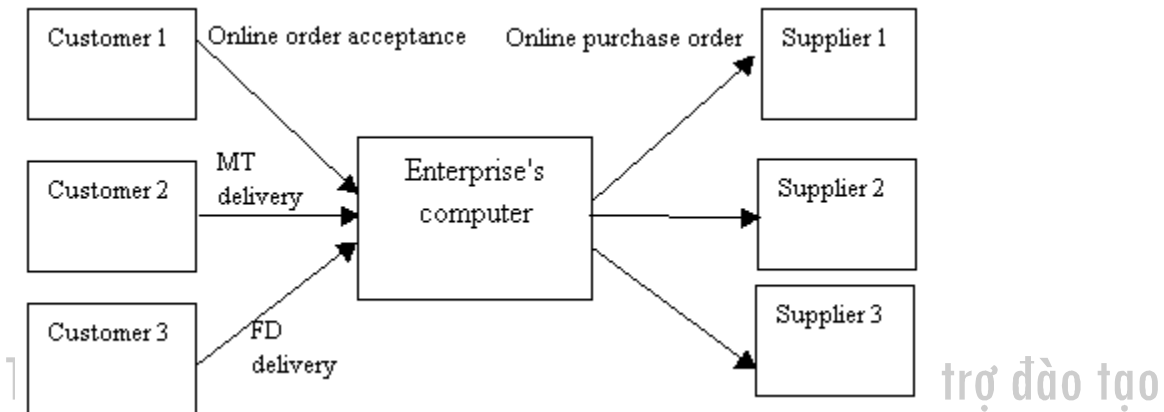
(2) Linkage between a client and a server in a distributed system

- 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.



<http://www.vitec.org.vn>

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.

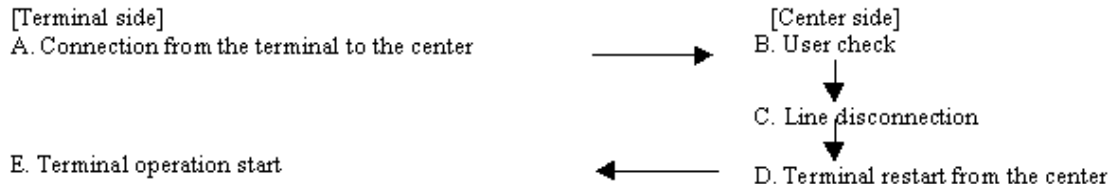
3) Password for information access

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.



(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.

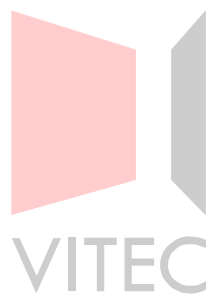
[Example]

- 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.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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.

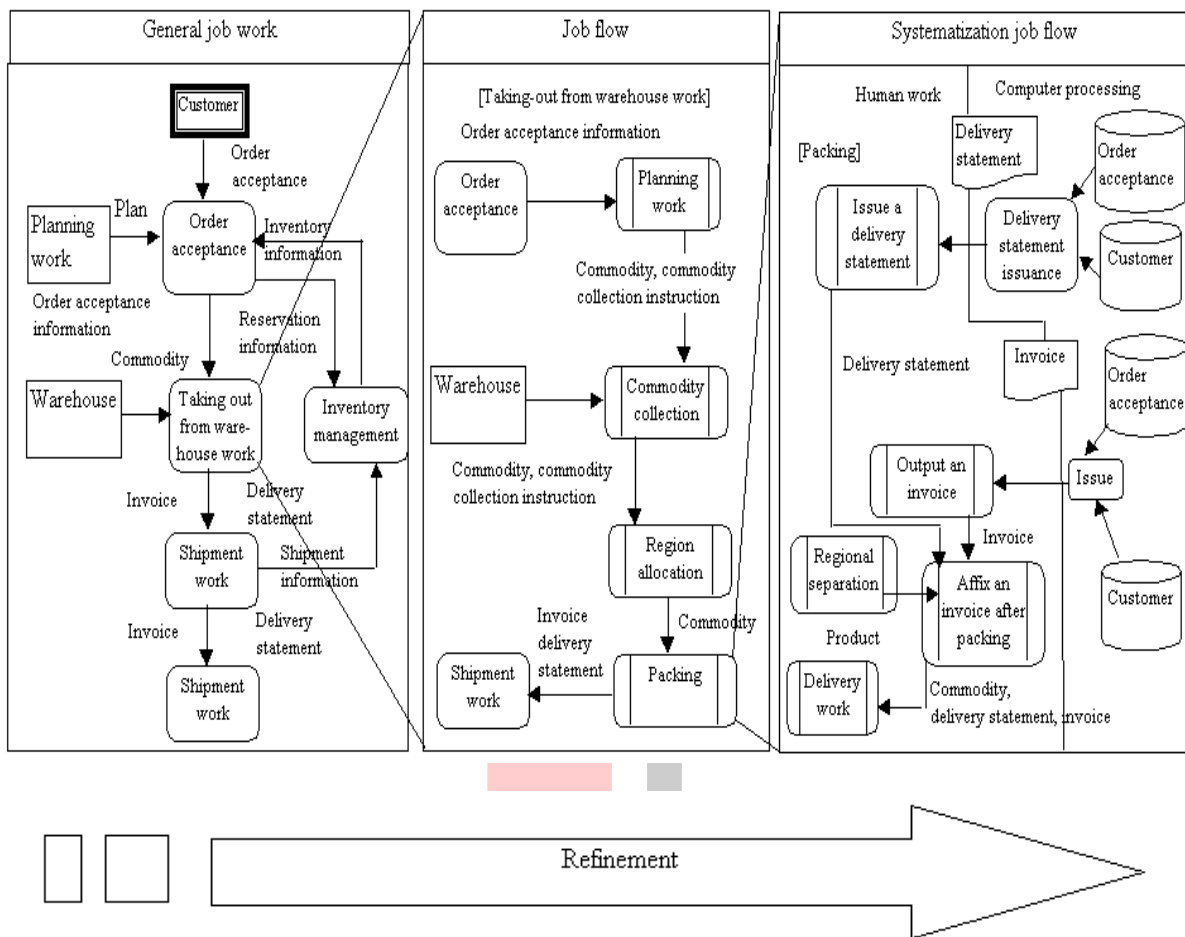
Descriptive materials	Graphical materials
1. Not appropriate for an overview 2. Appropriate for displaying details about a specific part 3. Mutual relationship between parts of the work cannot be seen easily 4. The overall image of the job flow cannot be easily understood	1. Appropriate for an overview 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

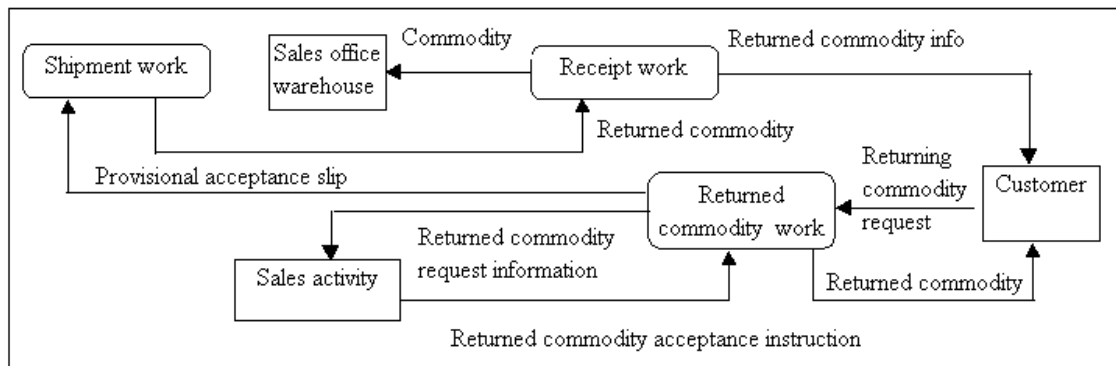
Based on the division of business activities into the four levels, draw a job flow diagram in three layers.



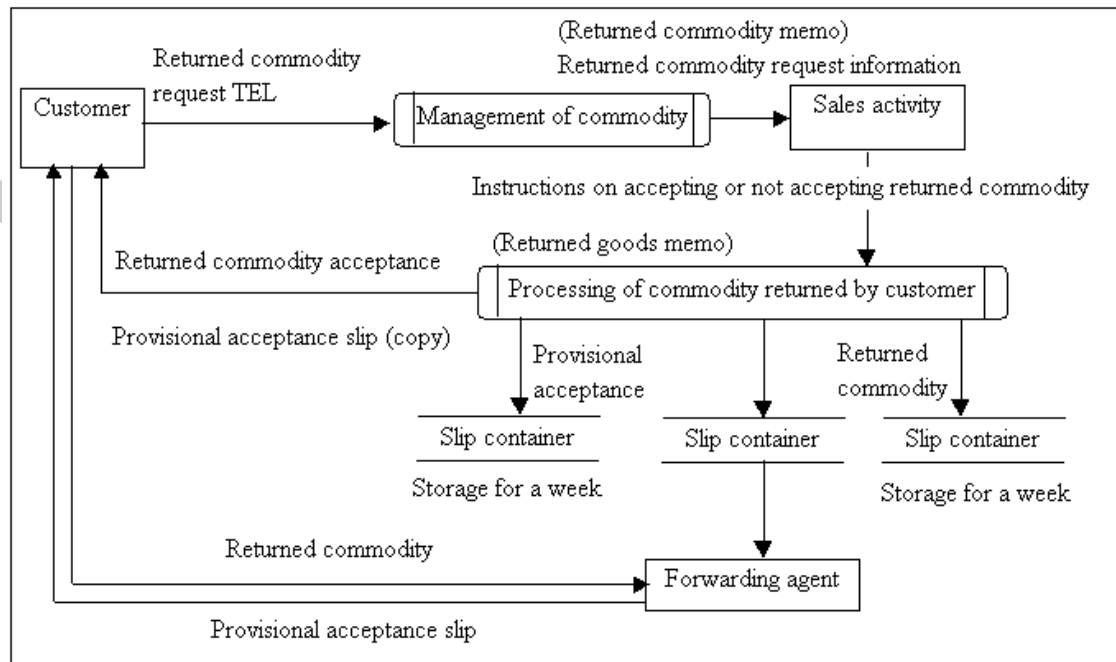
The first layer (general job flow diagram) of the job flow diagram represents the relationships among business areas to be analyzed at the level of business functions. The job flow represents details about each business function in the general job flow diagram. These documents clarify the scope of analysis, 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)



<http://www.vitec.org.vn>

[Checkpoint]

- For what purpose do you do this?

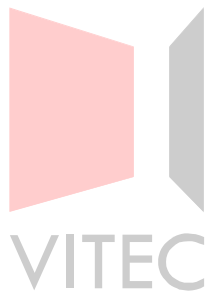
The roles and purposes of slips, reports, and works must be sufficiently obtained from the flow creator or persons in charge of the job by having interviews.

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



<http://www.vitec.org.vn>

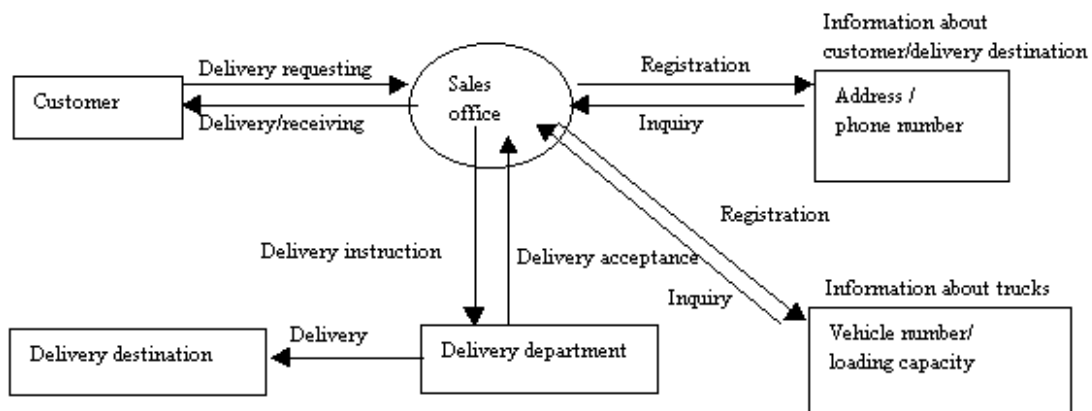
2.4.2 Data flow refinement

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.

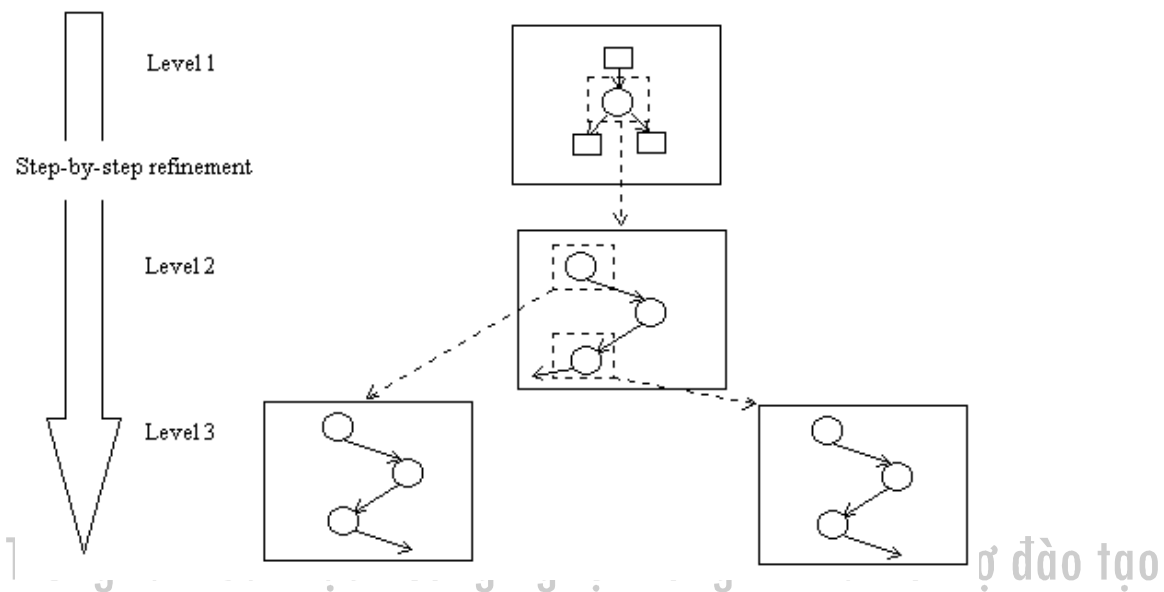
For example, there is a job called "delivery work" in the figure. The sales office carries out the delivery work using trucks. This work requires information about the trucks (e.g., the vehicle number and loading capacity). It is more reasonable to handle all information about a truck, such as its vehicle 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.



<http://www.vitec.org.vn>

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.



<http://www.vitec.org.vn>

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

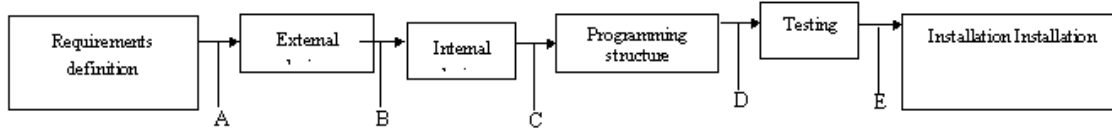
7. For each of the following sentences, write O if the sentence describes an item that must be considered when a user specifies a password. Otherwise, write X.
- (1) Specifying the user's name, phone number, or birthday as the password may lead to problems, but it is almost impossible to guess the password 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.

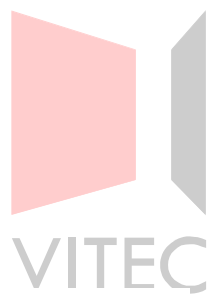
3 Data Model Design

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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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.

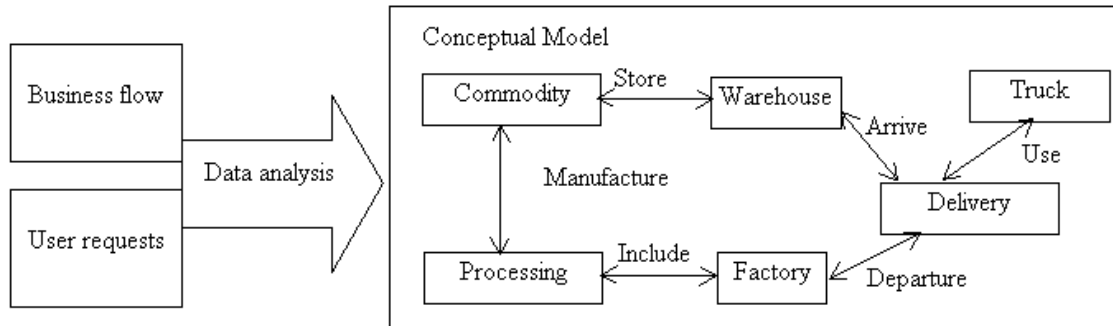
In the component-oriented approach, the work to be done by computer systems is determined first by examining what can be achieved by computers and what tasks must be performed by people.

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

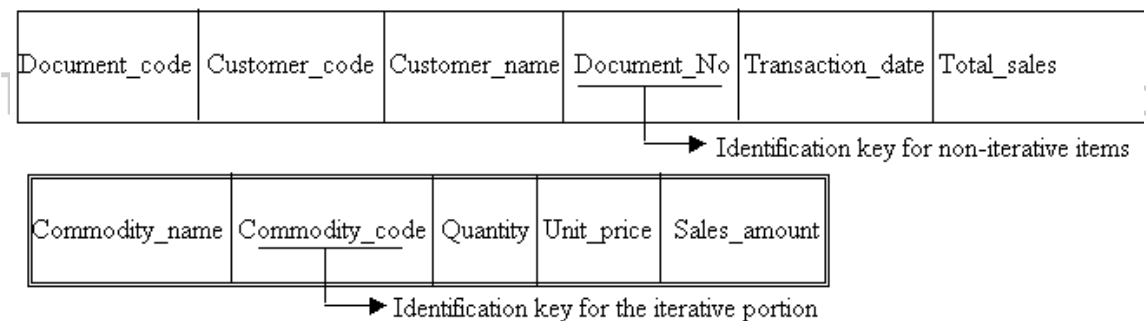
In the above layout, “commodity name” and “unit price” duplicate data items in the commodity master. Generally, “commodity name” and “unit price” are data items that should be stored only in the commodity master. One reason for including these items in other files is for convenience of processing. However, 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.

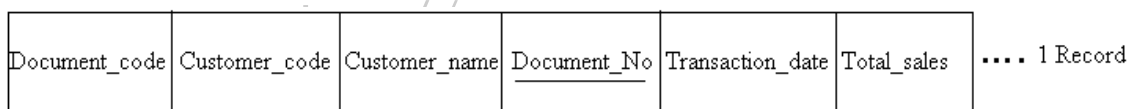
First, find the data item (which may be a combination of multiple items) that can be the key for identifying individual records, and underline it.

For a record having iterative items, find an identification key for the iterative items, and underline it (do the same for nested records).

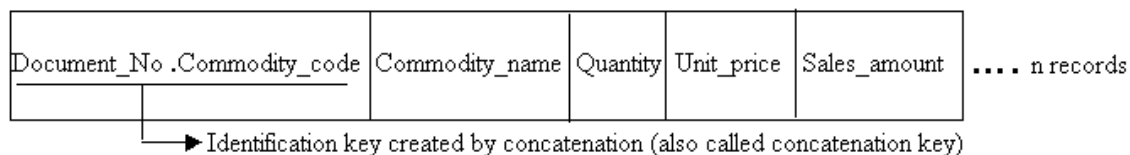


A record having iterative portion is called a non-normal form record. The next step is to separate iterative portion and break the record into subrecords that have no iterative portion.

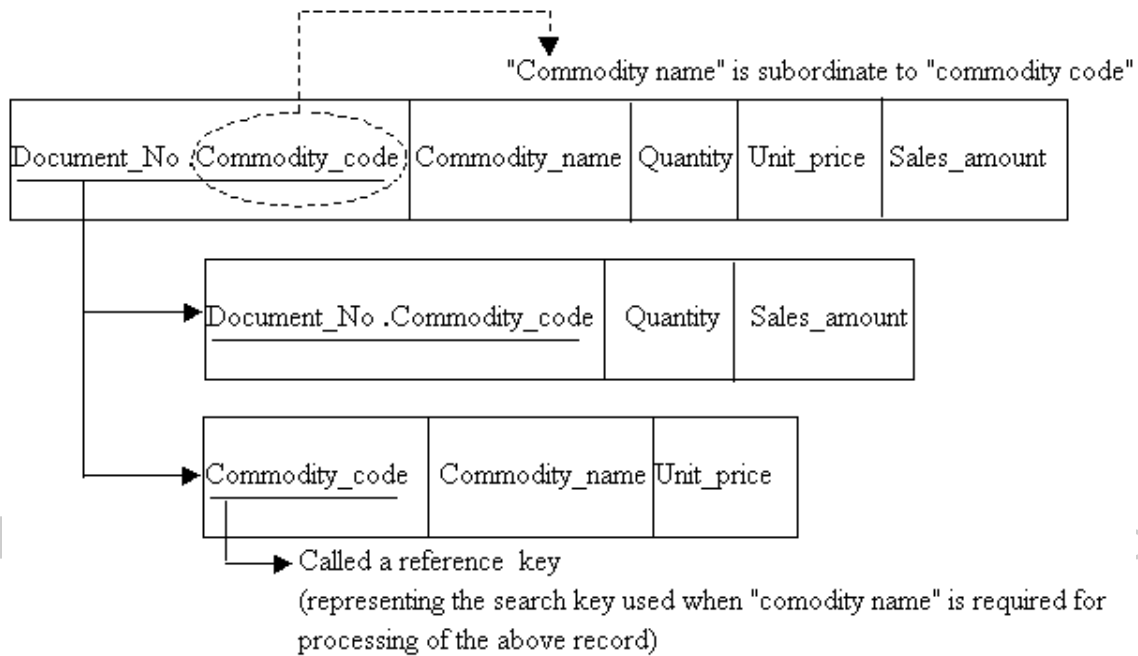
a. Creating a record without iterative portion



b. Creating a record by concatenating the identification key for the non-iterative portion and the identification key for iterative portion

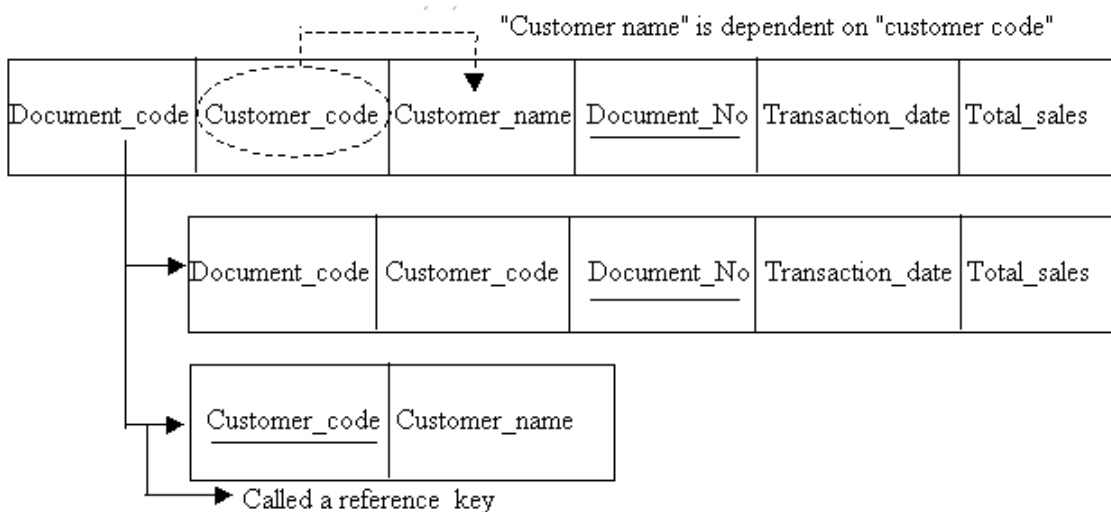


If the concatenated key includes a dependent data item, separate it into a subrecord. In the above example, “commodity name” in the record is a dependent data item.



“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	<u>Document_No</u>	Transaction_date	Total_sales	<u>Customer_code</u>	Customer_name
---------------	---------------	--------------------	------------------	-------------	----------------------	---------------

<u>Document_No.Commodity_code</u>	Quantity	Sales_amount	<u>Commodity_code</u>	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	<u>Document_No</u>	Transaction_date	Total_sales
---------------	---------------	--------------------	------------------	-------------

Sales detail:

<u>Document_No .Commodity_code</u>	Quantity	Sales_amount
------------------------------------	----------	--------------

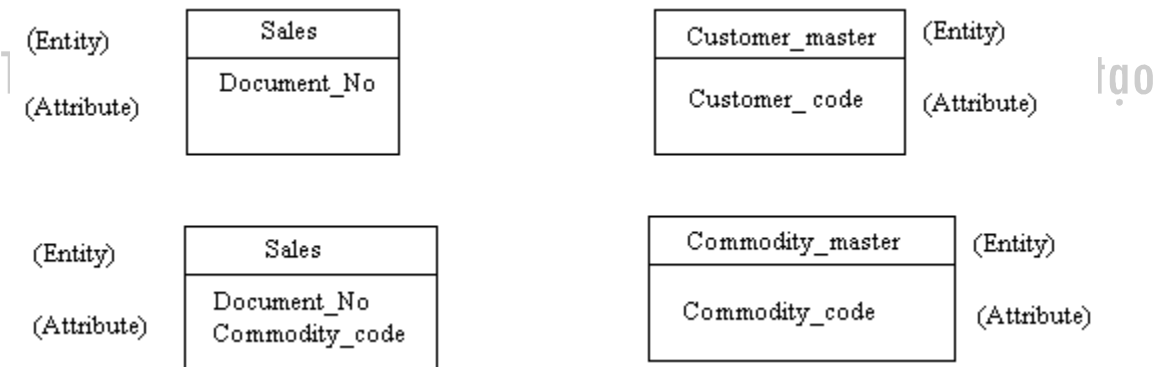
Commodity master:

<u>Commodity_code</u>	Commodity_name	Unit_price
-----------------------	----------------	------------

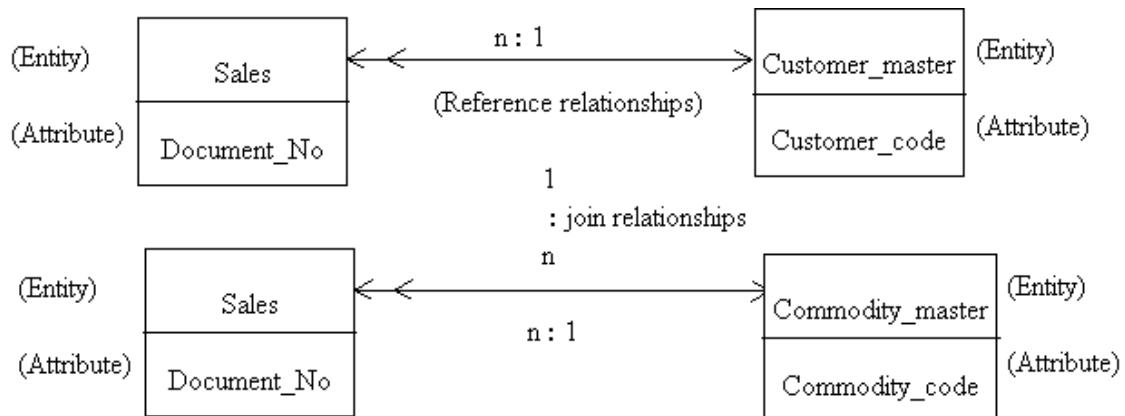
Customer master:

<u>Customer_code</u>	Customer_name
----------------------	---------------

b. Next, specify record names and identification key names as shown below:



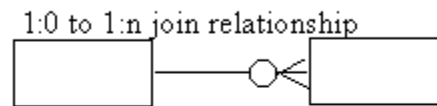
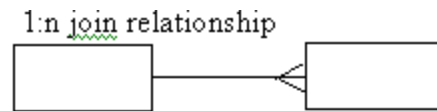
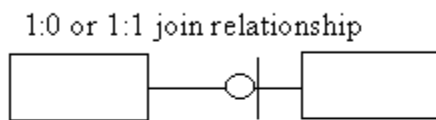
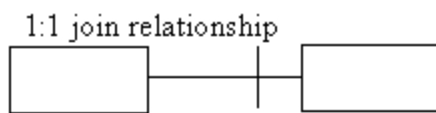
c. Connect the records with relationship lines based on relationships of concatenation and reference keys and the number of corresponding records.



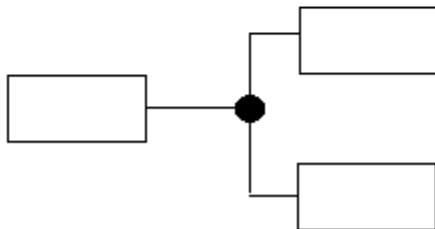
Examine the operating conditions to determine whether to define each ER model as one file (physical file unit).

[Reference: ER notation]

(1) Bird leg notation



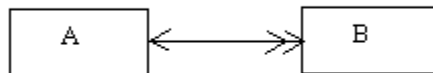
Mutually exclusive concatenation relationship (only one can be joined at a time)



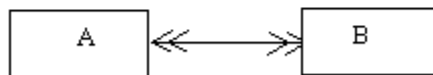
(2) Simple notation of relationships



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



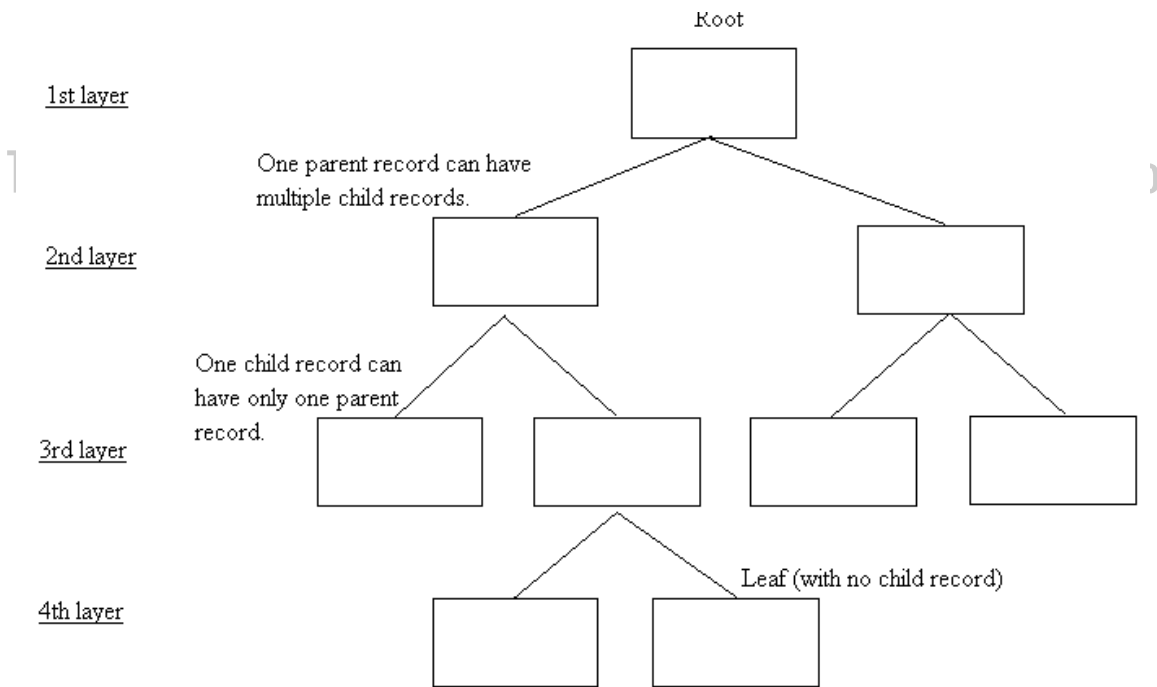
3.2 Logical Data Model

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]

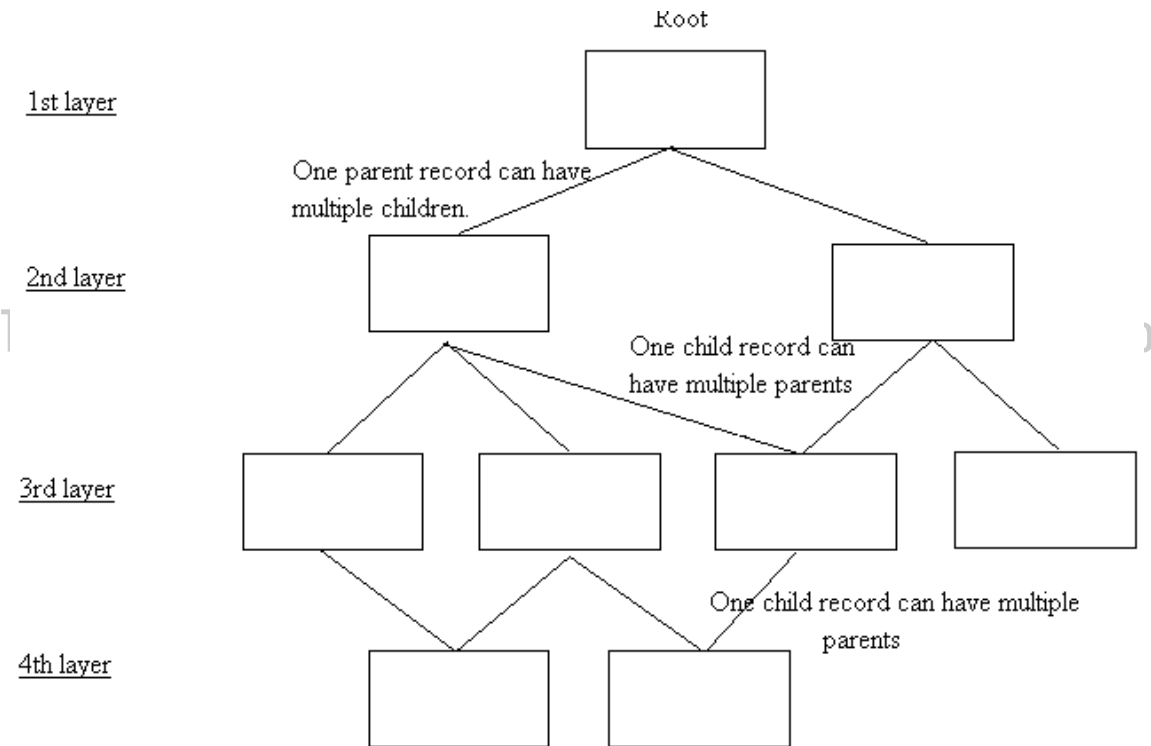


3.2.2 Network model

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]



<http://www.vitec.org.vn>

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.

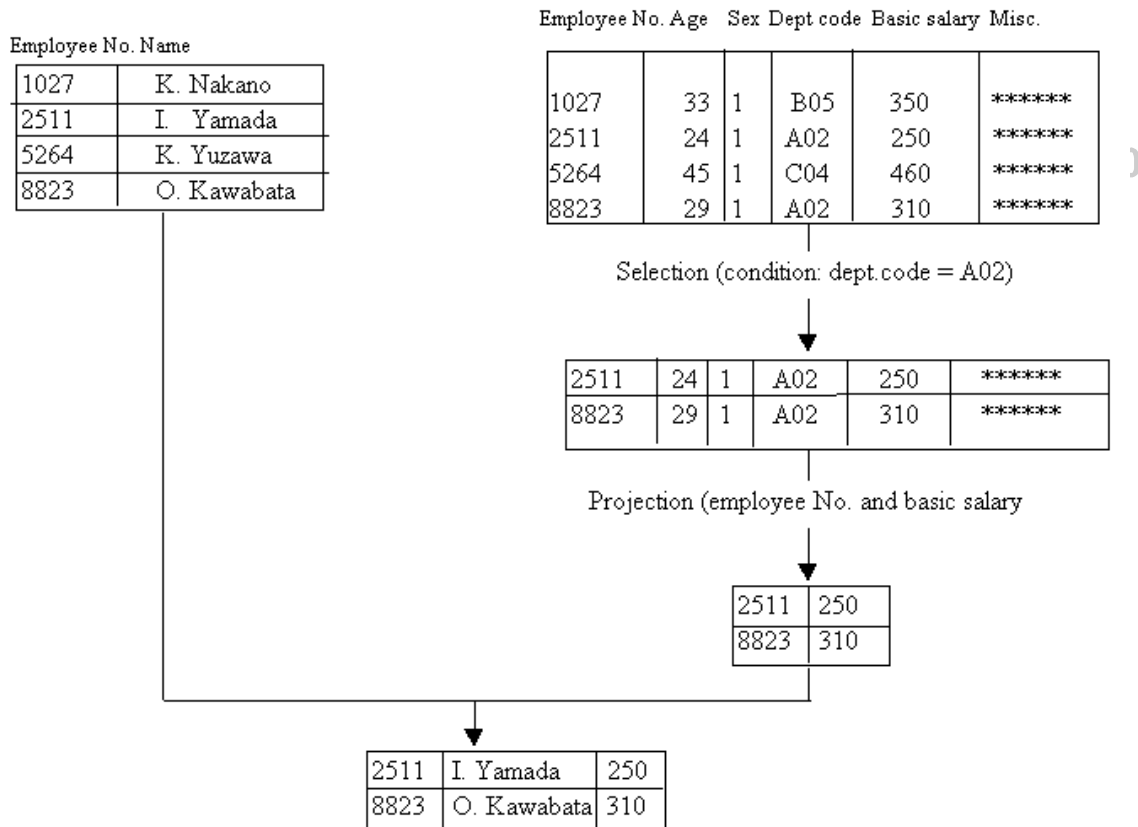
The relational model does not have a fixed structure. It is based on individual tables and logically connects elements in a table to represent the relationships among data.

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]



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.

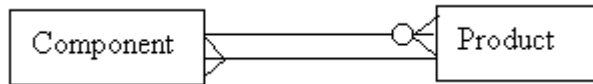
3. Select the sentence that correctly describes the data-oriented approach.
 - (1) The data structure is likely to change, but process specifications are relatively stable unless the target job changes.
 - (2) In a system developed with the data-oriented approach, multiple jobs use databases that are 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.

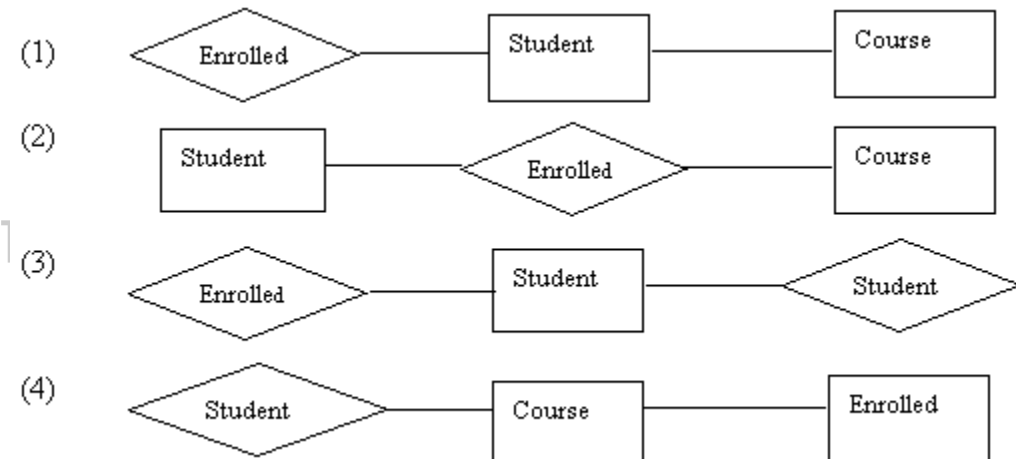
9. Select the sentence that does not describe the ER model.

- (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.

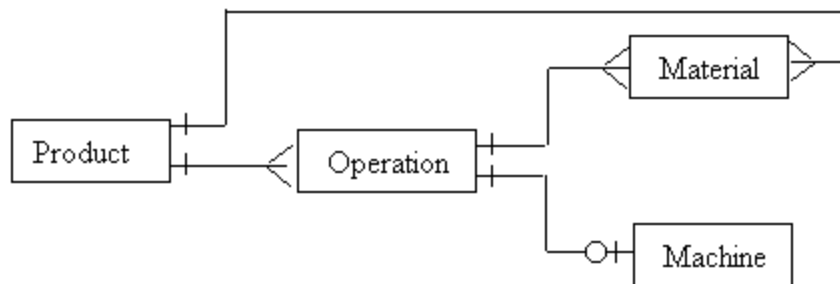
10. Select the sentence that correctly describes the ER model.

- (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.



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.

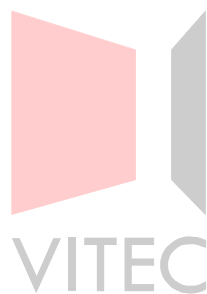
4.1 Preparation of User Manual (Outline)

4.2 Design of System Test Specifications

4.3 External Design Documents

4.4 Design Review

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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

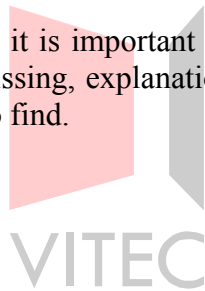
Because process and data flows are made concrete in this design phase, a review generally takes the form of a walkthrough.

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.



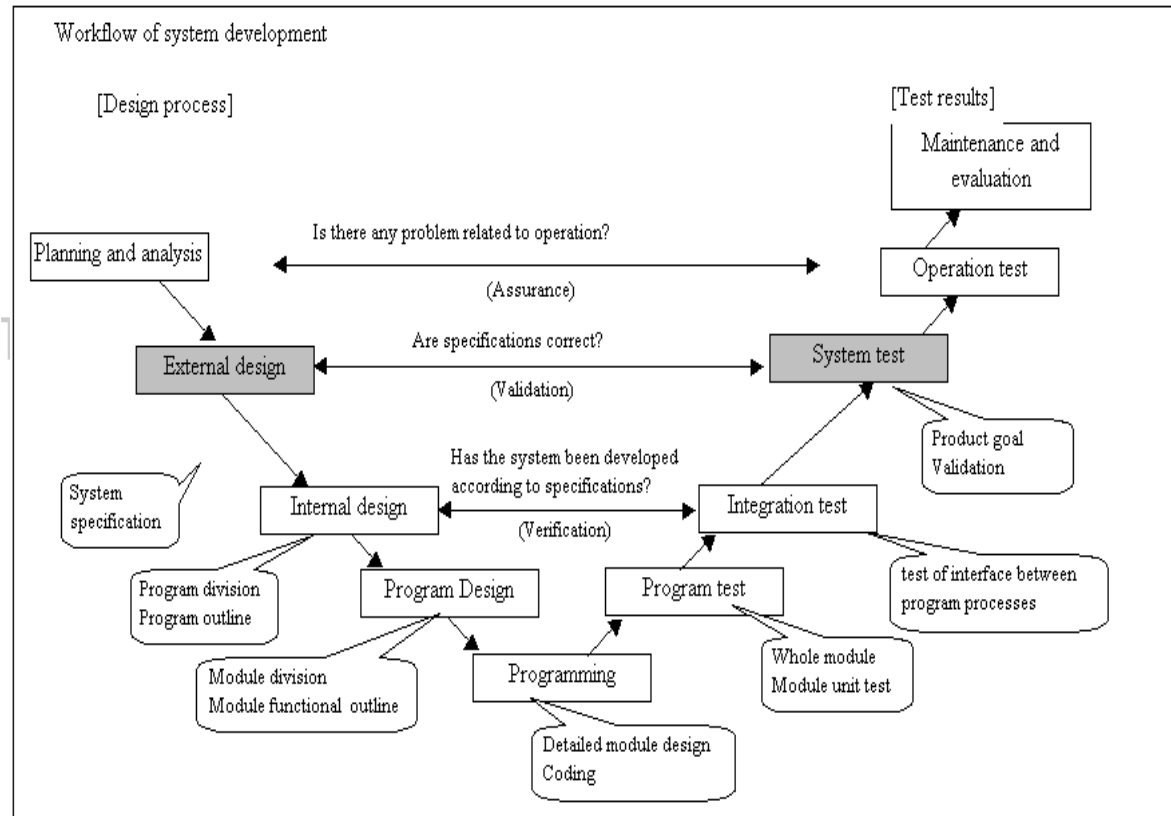
<http://www.vitec.org.vn>

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.



<http://www.vitec.org.vn>

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	<ul style="list-style-type: none"> - Do program execution results satisfy the functions described in program specifications? - Does all internal routes in the program work properly?
Integration test	<ul style="list-style-type: none"> - Can all programs in a component be executed? - Is data transferred smoothly between programs? - Is data transferred correctly between components?
System test	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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?

<http://www.vitec.org.vn>

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

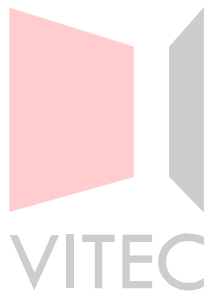
The System Configuration Diagram shows a overview of the relationships between the human interface and individual jobs based on system requirements specification.

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.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



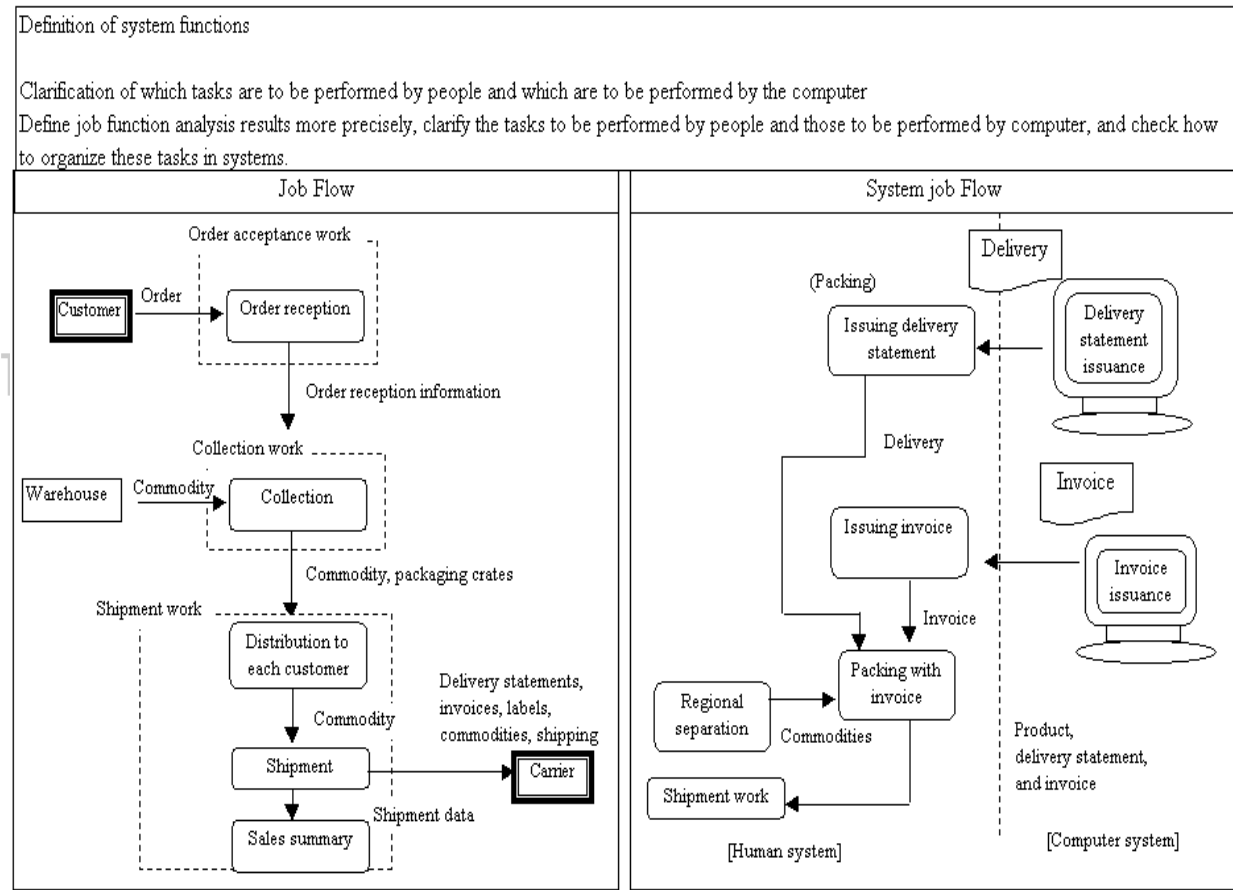
<http://www.vitec.org.vn>

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.



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 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

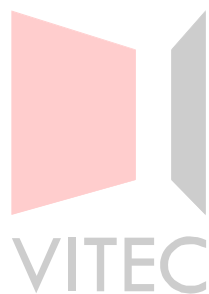
what would happen if implemented. The participants confirm the whether or not each requirements is met. (Walkthrough)

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.

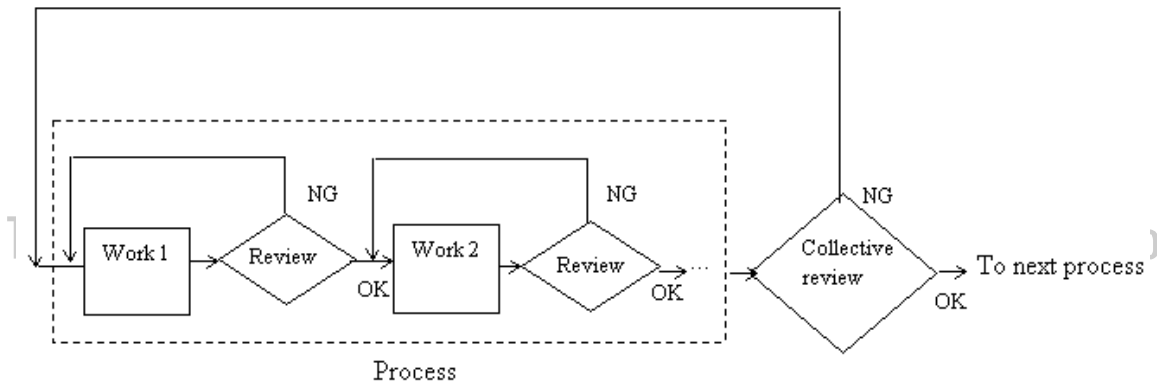
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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.



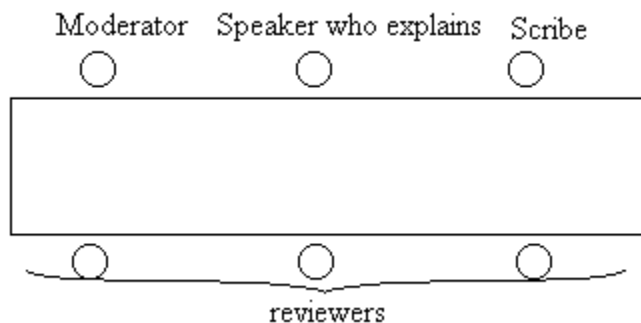
There are the following forms of review:

(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.



<http://www.vitec.org.vn>

[Reference: Review considerations]

Review considerations	Check item
Checks related to perfection	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - Are notations, terms, and symbols consistent among specifications? - Do descriptions in the specifications correspond to plans and update specifications?
Checks related to feasibility	<ul style="list-style-type: none"> - Is the system designed for easier use? - Is the system satisfactory in performance, reliability, expandability, maintainability, and portability?
Testability check	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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?

VITEC

<http://www.vitec.org.vn>

Review considerations	Check item
Operability	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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.

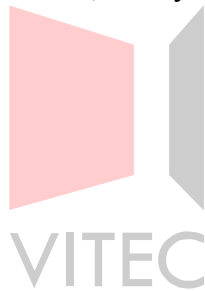
2. From the following descriptions of objectives and results regarding carrying out walkthroughs during the course of a project, select the one that is not correct.
 - (1) Finding a mistake in the design process saves more time and money than if the mistake is found in a subsequent process.
 - (2) Because 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 inappropriate 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



<http://www.vitec.org.vn>

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)

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



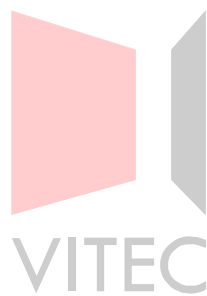
<http://www.vitec.org.vn>

- 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)

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Part 3

INTERNAL DESIGN

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

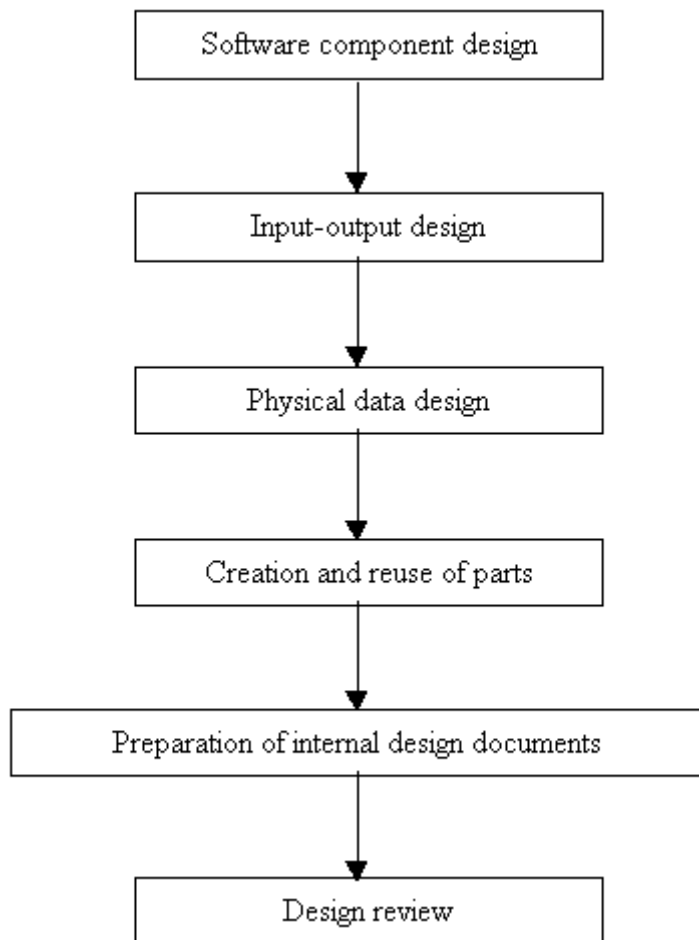


<http://www.vitec.org.vn>

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 be 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.



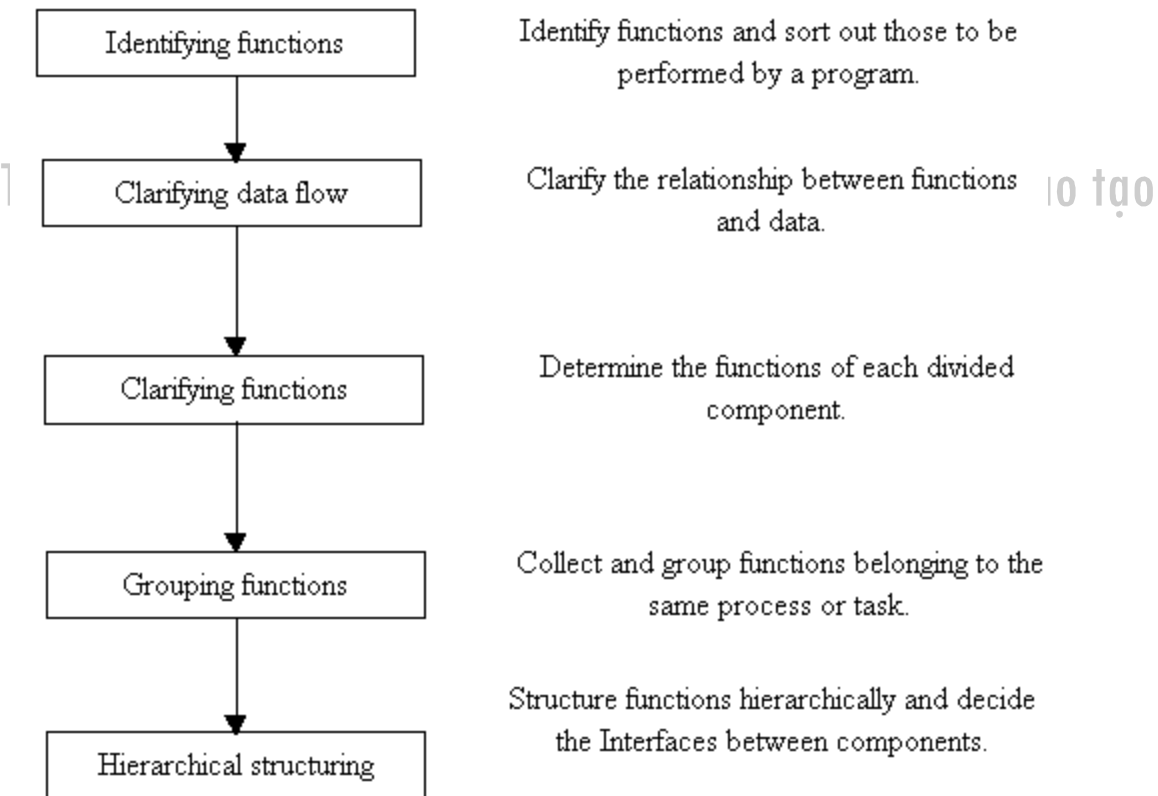
1.1.1 Software component design

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:



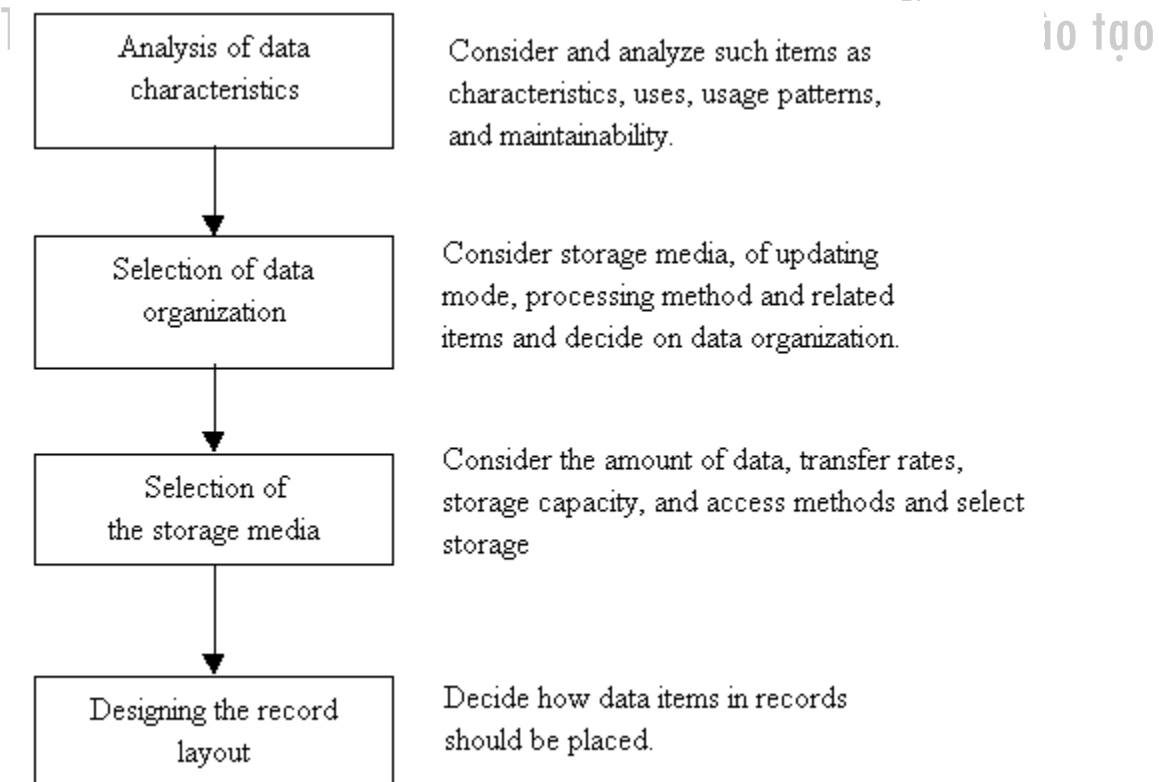
1.1.2 Input/output design

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

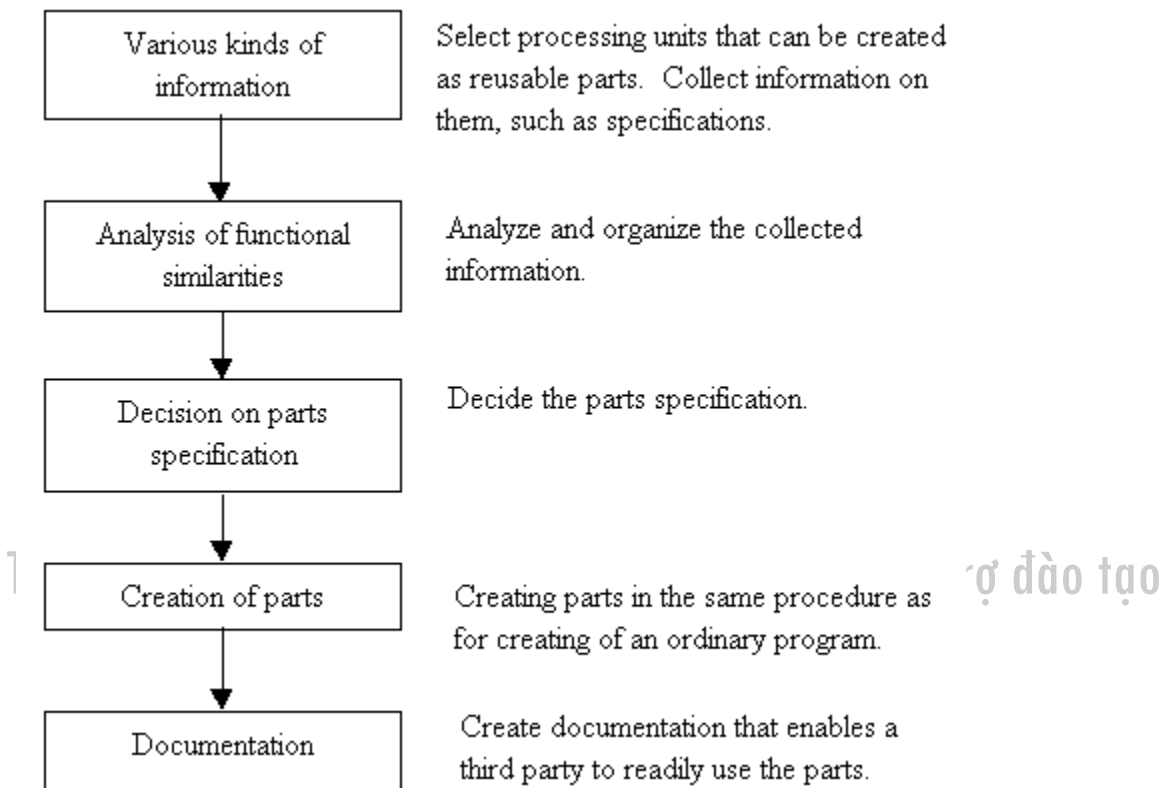
1.1.3 Physical data design

The physical data design (file design) is performed by the following procedures:

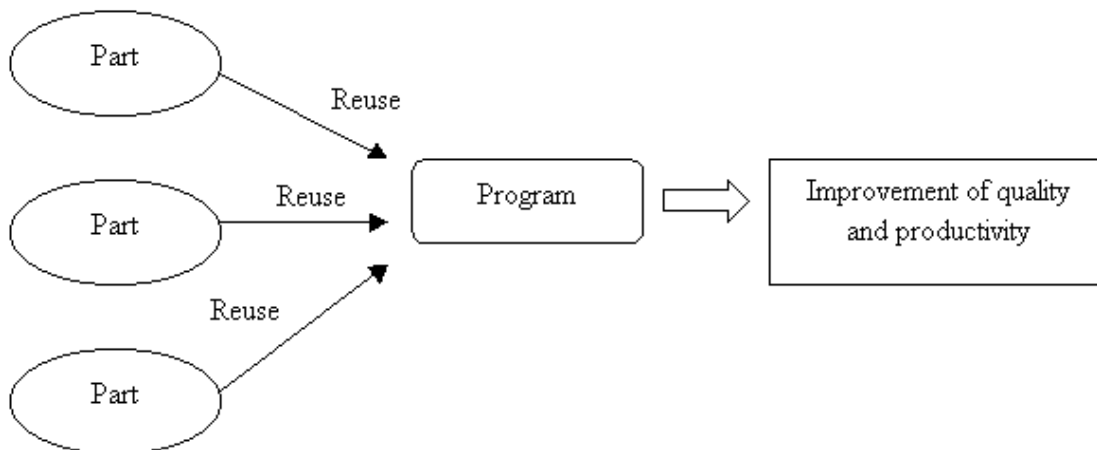


1.1.4 Creation and reuse of parts

Creation of parts is performed by the following procedure:



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

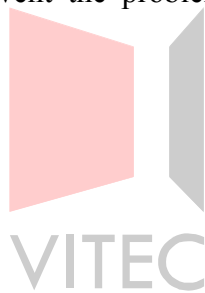
Review to detect defects and omissions.

<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.



<http://www.vitec.org.vn>

2 Software Component Design

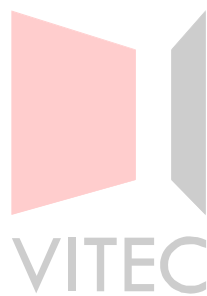
Chapter Objectives

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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

2.1 Designing Software Structure

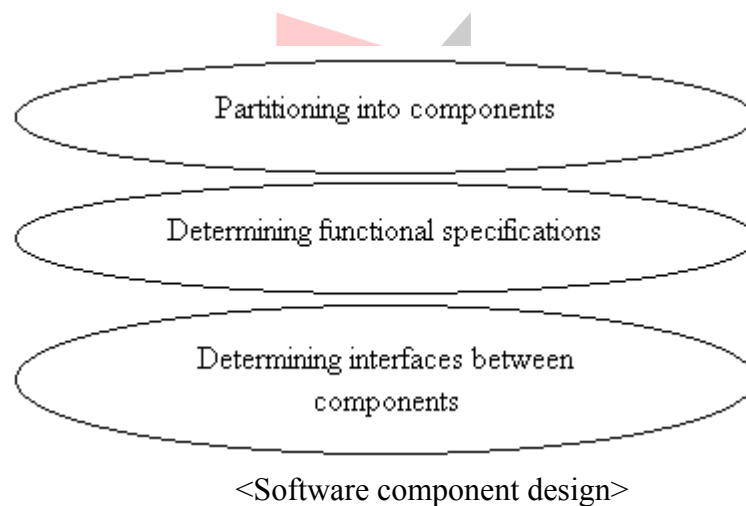
As software becomes larger and more complex, the overall structure (software architecture) of the system greatly affects software design. Software architecture refers to the structure of software, that is, the elements of the software, their functions, and the relationships among the elements. The software 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.



< 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

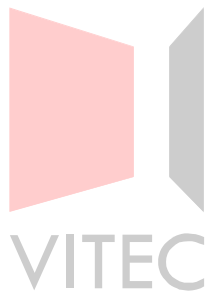
The processing functions of partitioned components are determined.

(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.



<http://www.vitec.org.vn>

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

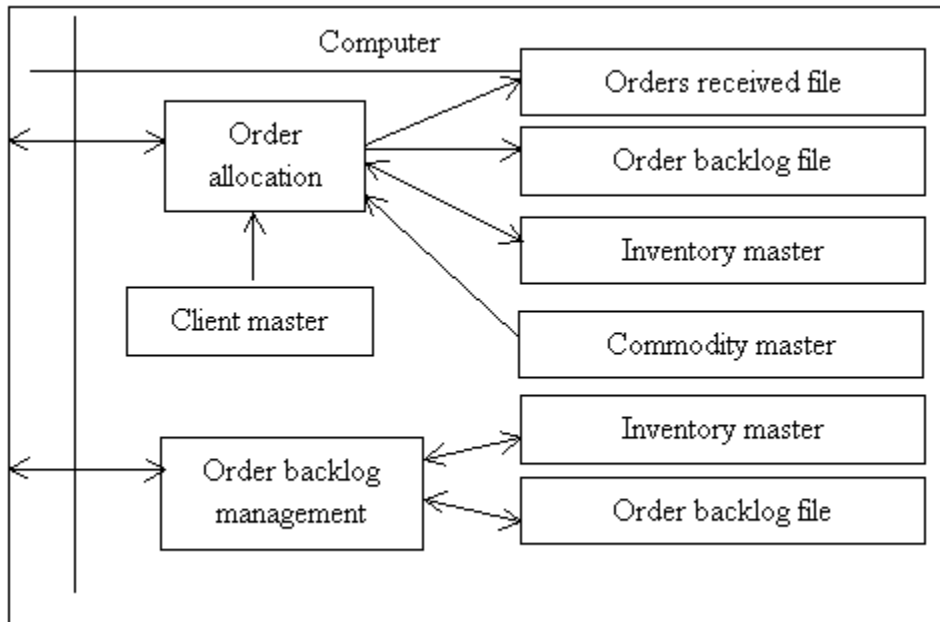
The components specified in the systematization job flow are extracted first, and then the files used as interfaces between the components are clarified because data transfer between components is accomplished with files.

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.

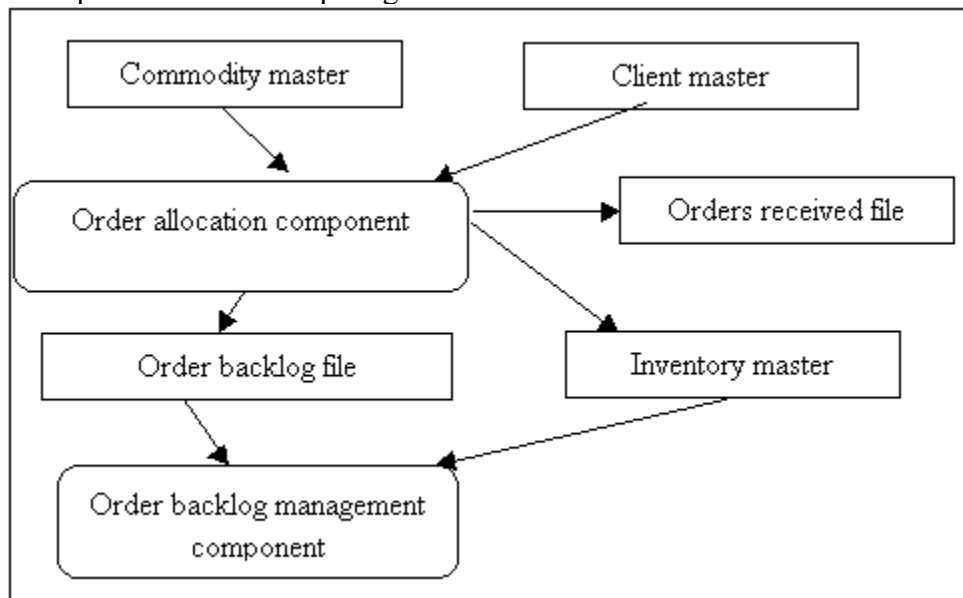
VITEC

<http://www.vitec.org.vn>

Systematization job flow



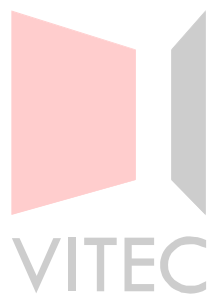
Components relationship diagram



(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).

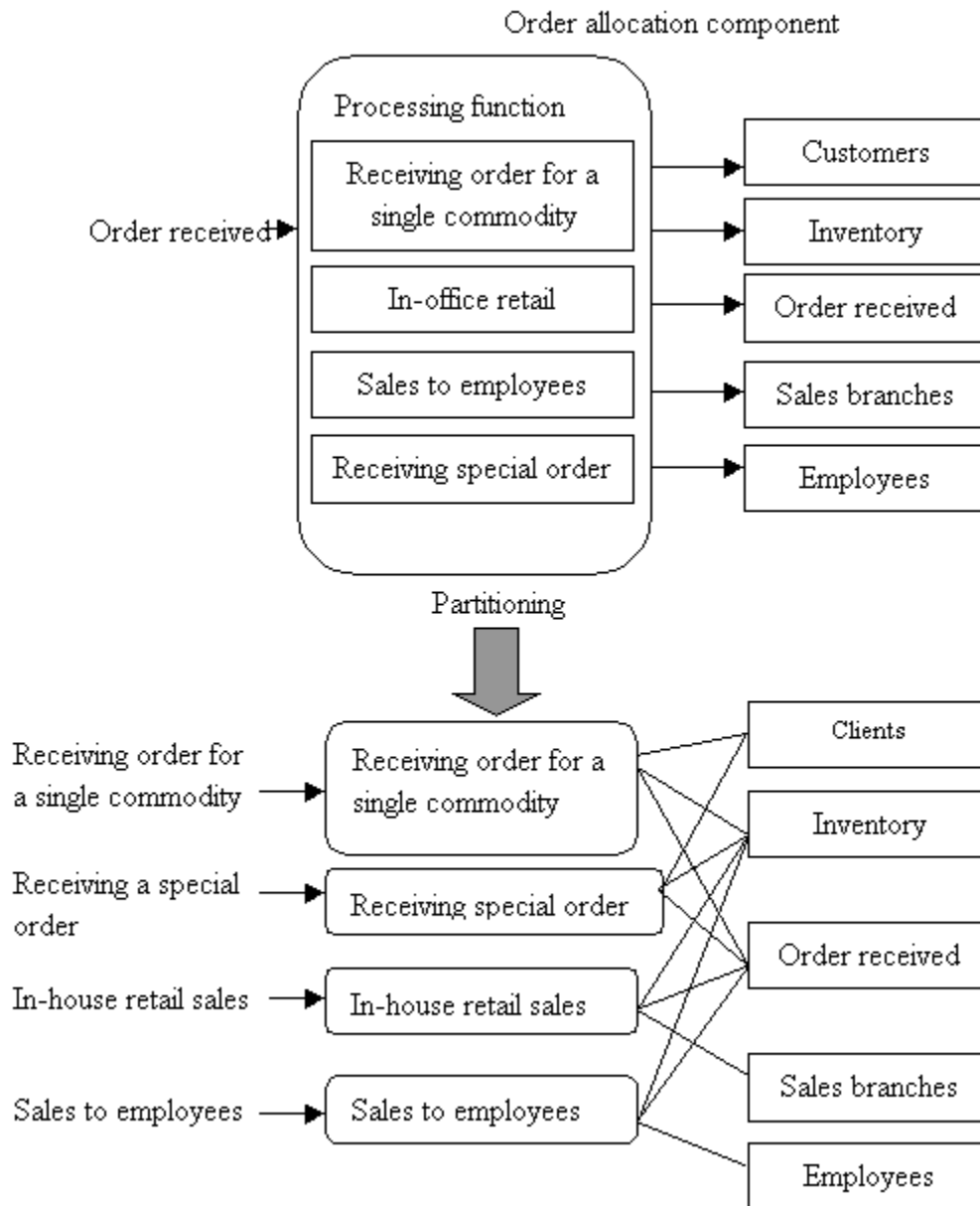
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

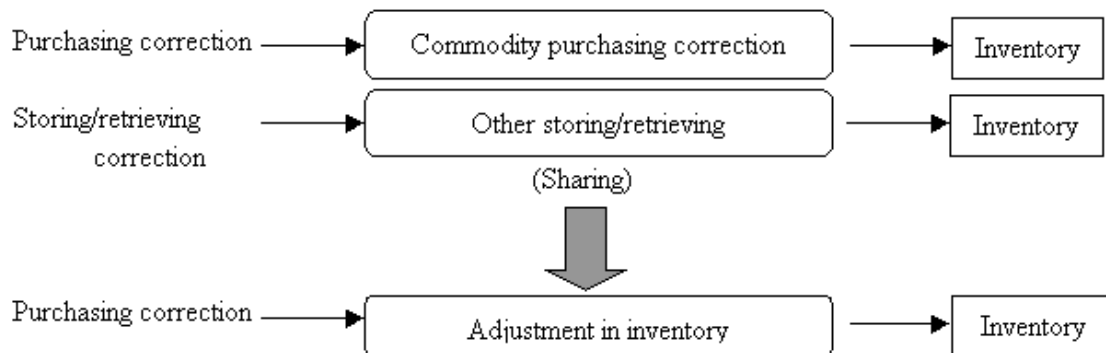
- Partitioning components

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

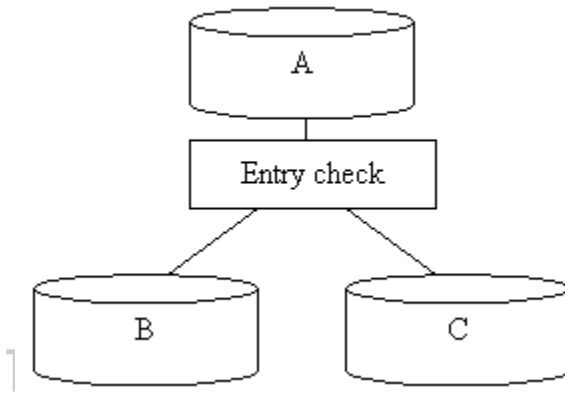
Creation of statistical tables

a. Partitioning programs for batch processing

This section first explains the partitioning of programs for batch processing.

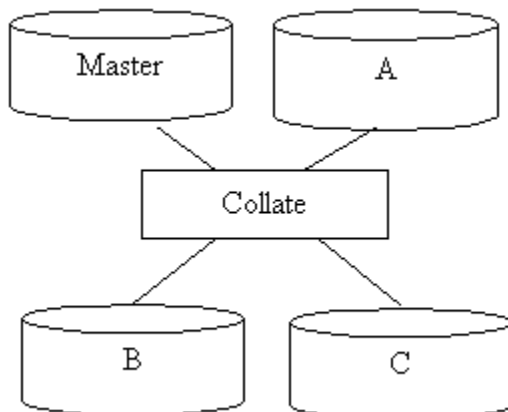
For partitioning by processing patterns, functions of programs that require batch processing can be classified in the following processing patterns, which are frequently used.

< Input check system >



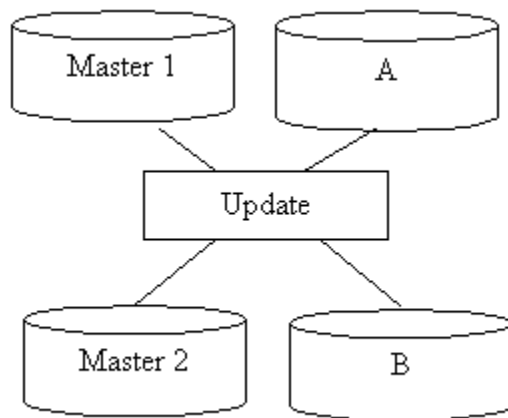
Contents of input data file A are checked. If they are correct, they are written to file B. If they contain an error, they are written to file C. This pattern also applies to extraction process of records.

< Collation system >



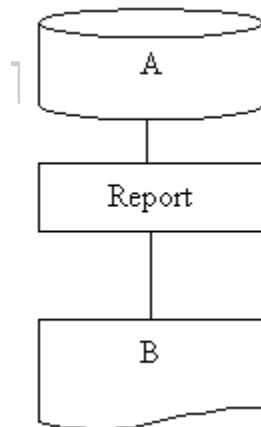
Input data file A is collated with the master file. If the input data matches, it is written to file B. If the input data does not match, it is written to file C.

< 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 >

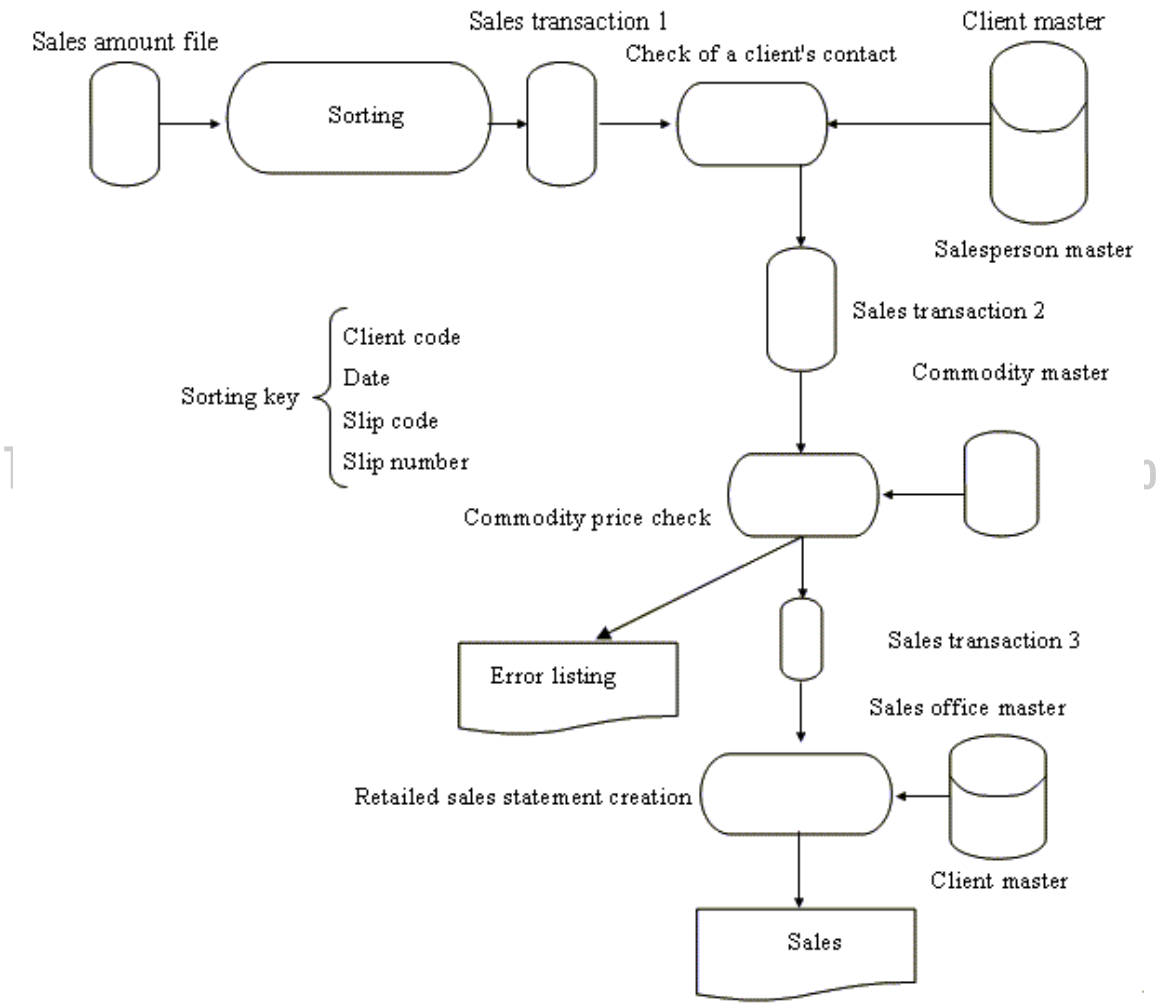


Create listing B by using entry file A

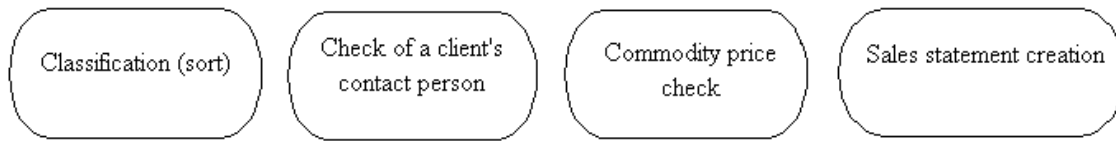
VITEC

<http://www.vitec.org.vn>

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.



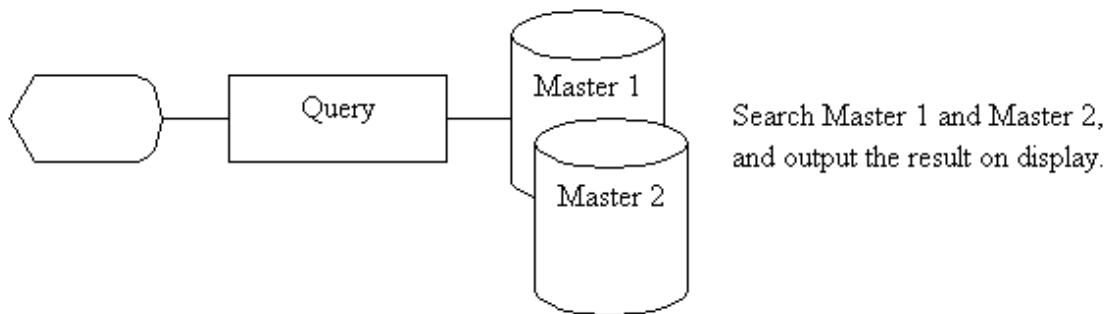
Also try to reduce the program size as much as possible to make development and maintenance easier. For example, a guideline may be established to limit the module size of COBOL programs for compilation to 500 steps or less. Because no program in this example has an especially large number of steps, each program remains as is.

b. Partitioning programs for online processing

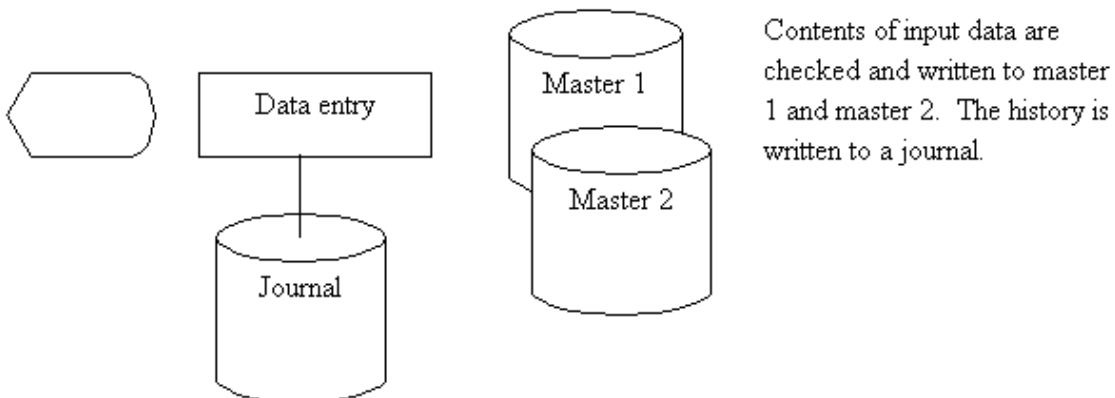
Partitioning programs for online processing is explained below.

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:

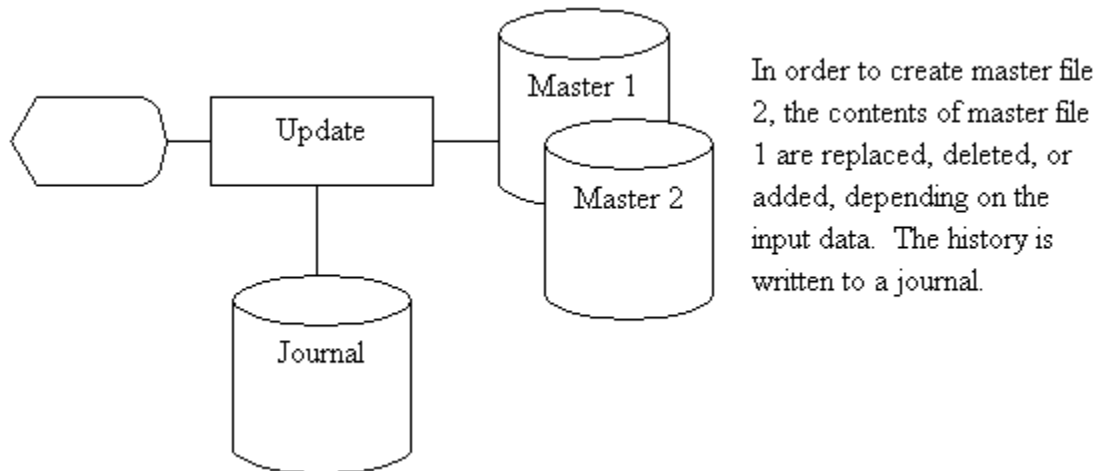
< Query processing >



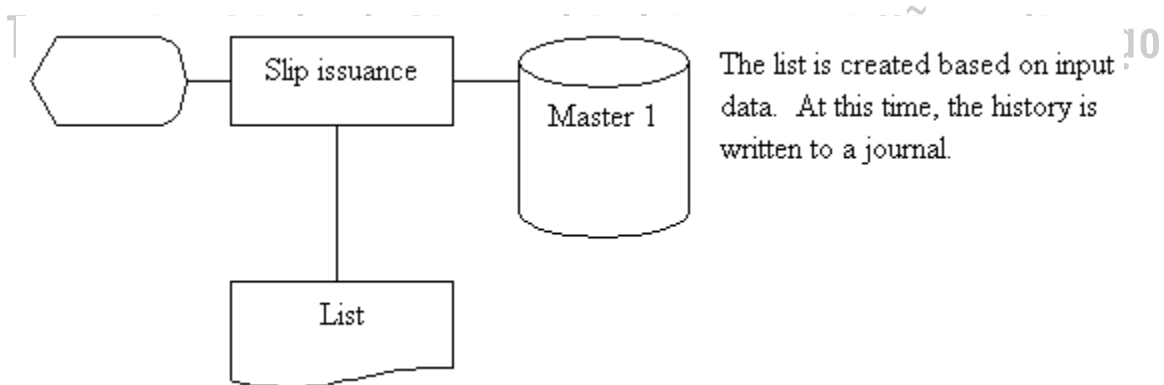
< Data entry processing >



< Update processing >



< Slip issuance processing >



- 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.

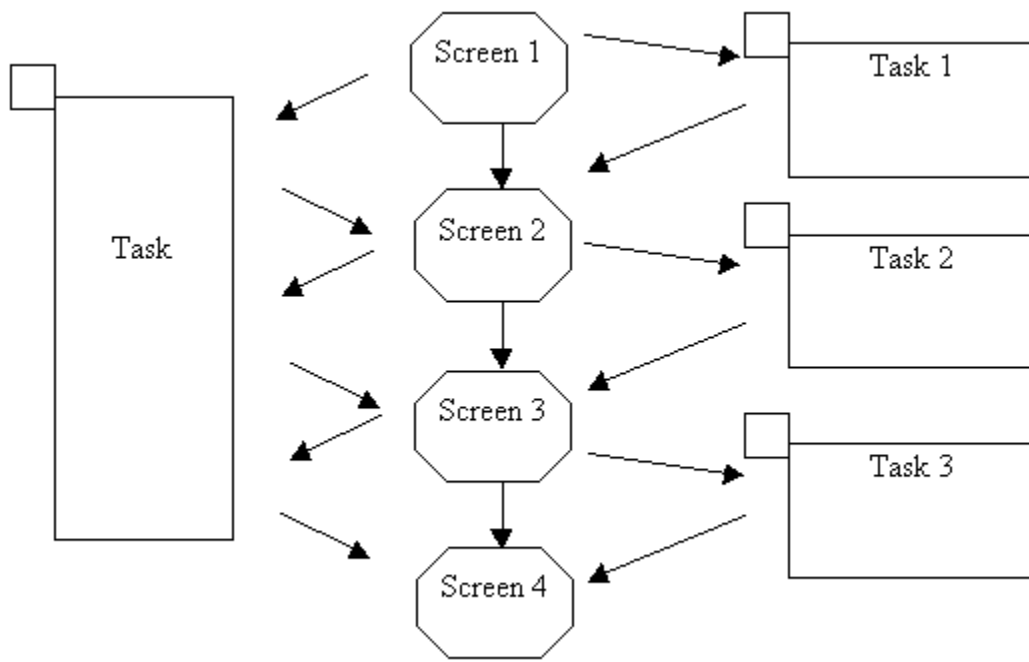
Implementing these functions in a single task, the program generally becomes complicated and development productivity declines, but the response of the program becomes better than a program with multiple screens correspond to multiple tasks. Multitasking should be selected when the traffic intensity 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



Trong tâm sự học công nghệ thông tin và tin học ứng dụng

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.

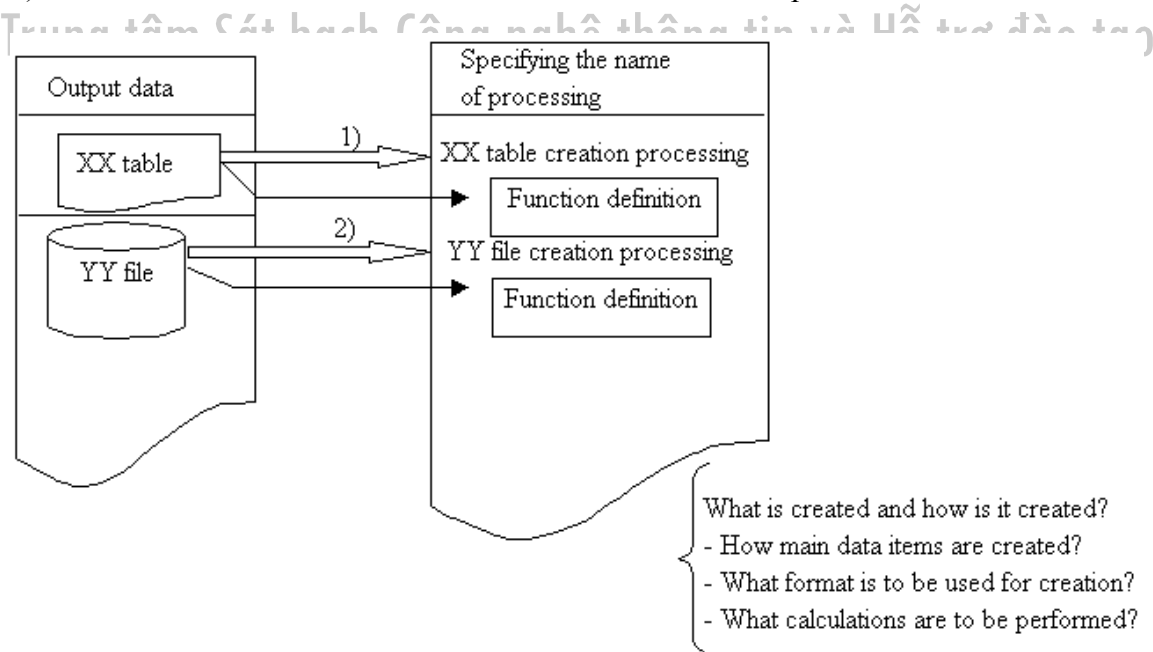
The component outline definition can be created by following the procedure below:

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.



(2) Definition of input-output data

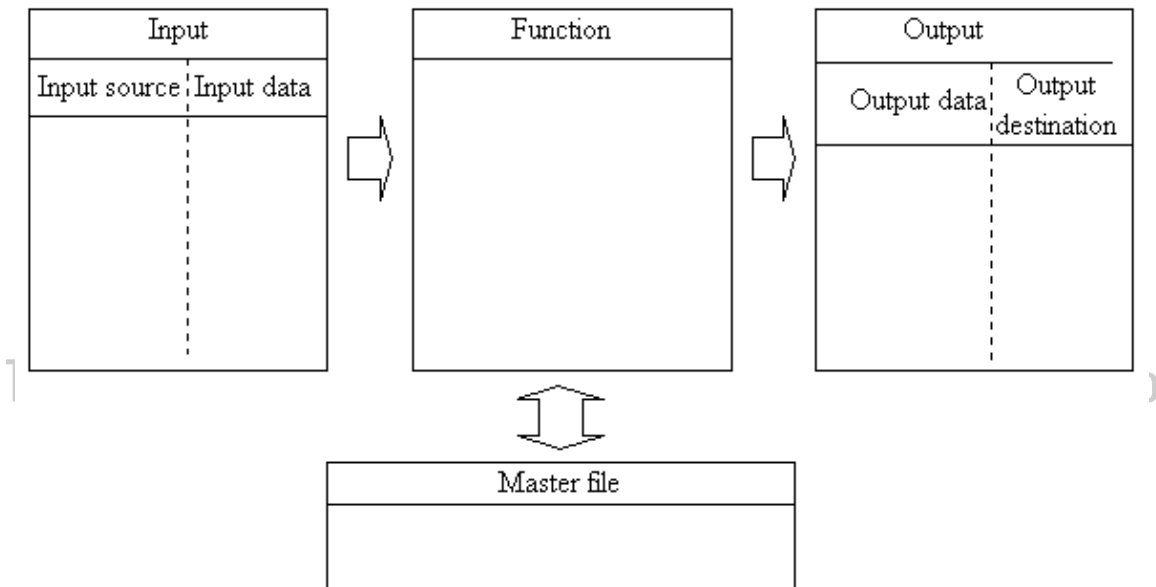
The necessary input-output data and master file are created for every processing function.

(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:

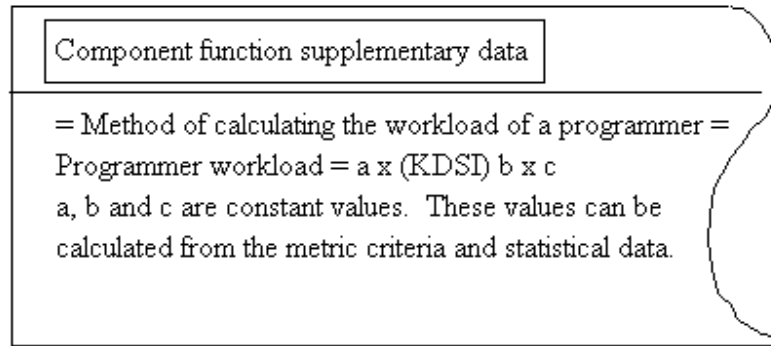
< Component outline definition >



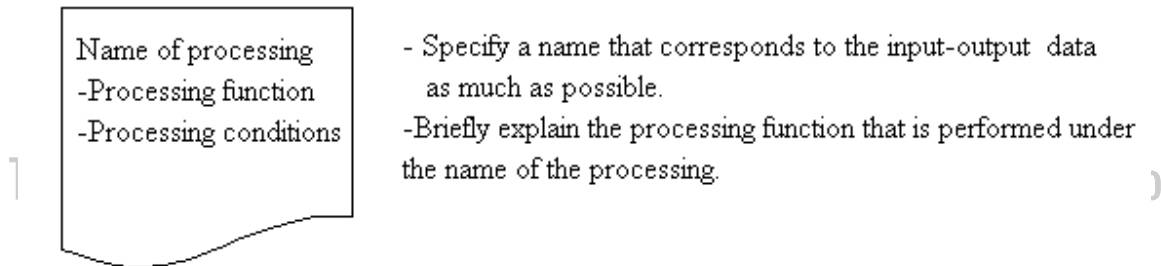
Note the following points:

- 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)



3) Function section



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 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

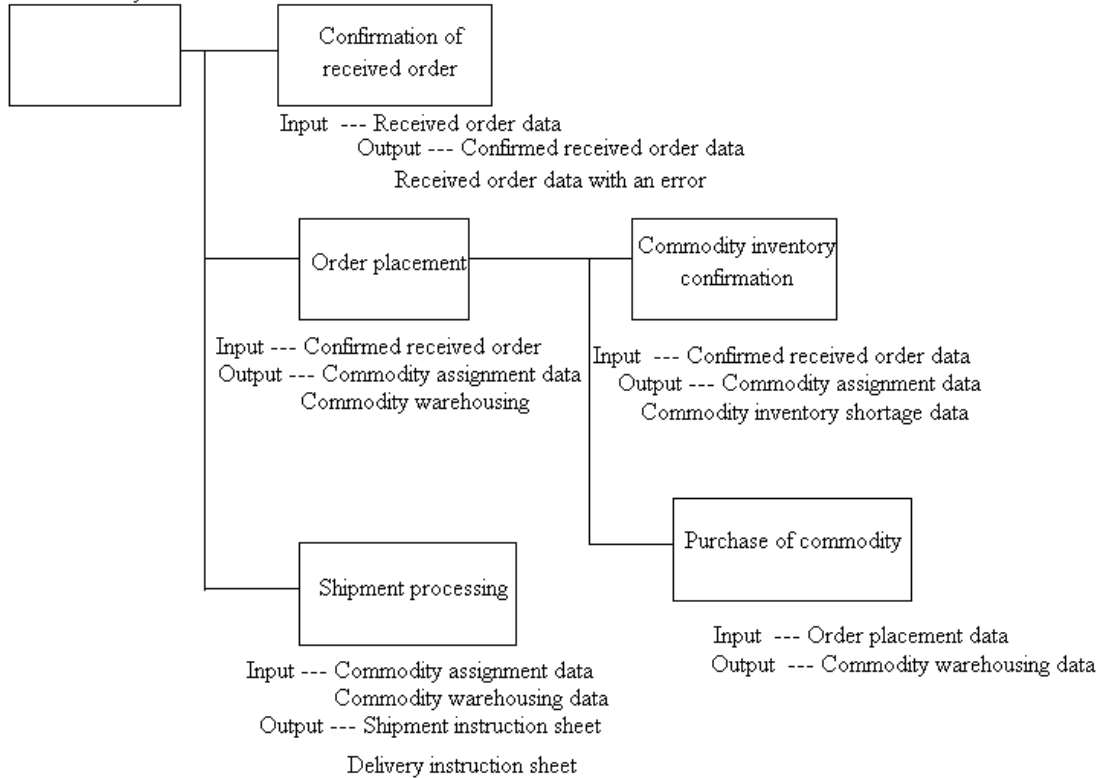
Reuse of software is a very effective method, because it leads to not only reduced development and maintenance costs, but also to improvements in quality. Therefore, specifications and partitioning must be determined with due consideration to reusability.

<http://www.vitec.org.vn>

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



VITEC

<http://www.vitec.org.vn>

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)

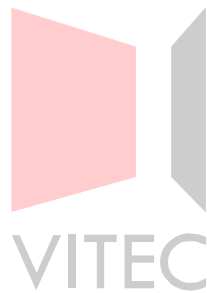
This tool is for structural system analysis and modeling. The features of this tool are stepwise partitioning of functions and modeling them in a hierarchical structure.

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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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



<http://www.vitec.org.vn>

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



<http://www.vitec.org.vn>

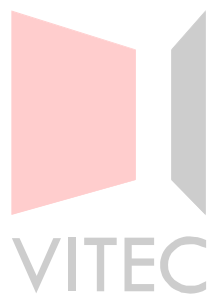
5. During partitioning of batch processing, it is sometimes classified by the processing pattern. From the group of answers listed below, select the correct definition for each of the following descriptions.

- (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

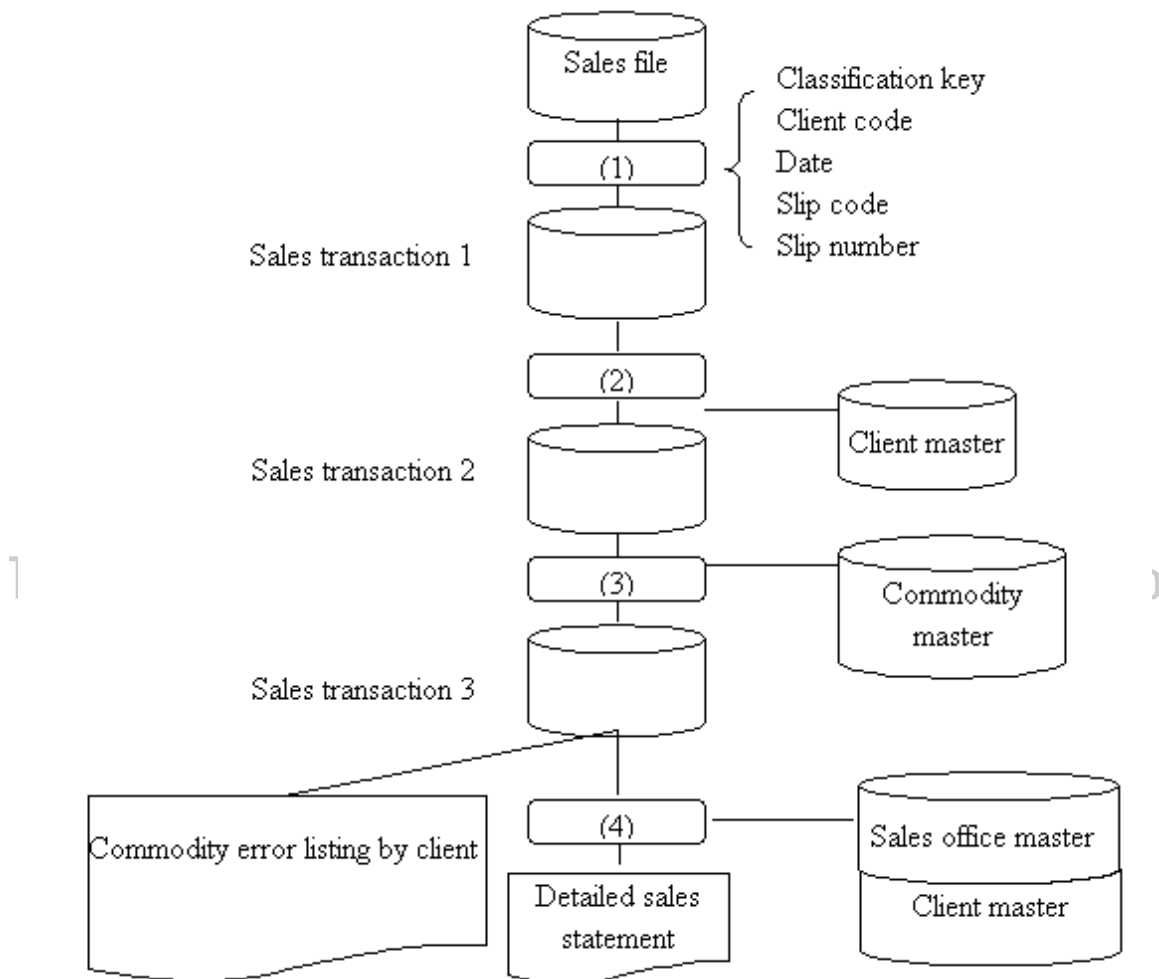
- a. Input check system
- b. Collation system
- c. Update system
- d. Report creation system

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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

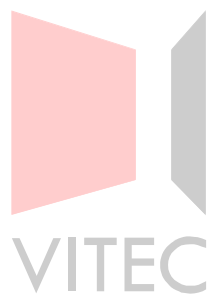
Chapter Objectives

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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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.



<http://www.vitec.org.vn>

(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)

- Are any data items missing? Are they really necessary?
- Are there any errors concerning the attributes of data items?
(e.g., data type, number of characters per line, editing format, decimal point)
- Is there any misinterpretation of data items?
(e.g., Is a price the selling price or the buying price?)
- Does hardware support data items? (e.g., barcode)
- 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.

- Selection of input media, input devices, and input methods
- Design of input items

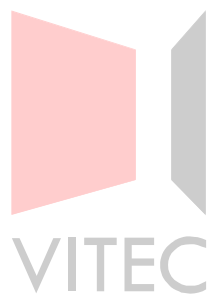
1) Selection of input media, input devices, and input methods

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:

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

<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.

<http://www.vitec.org.vn>

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:

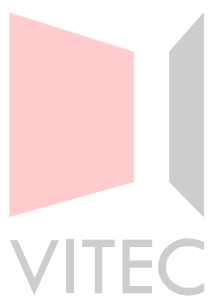
1. ☐ Design of the report outline
2. ☐ Design of report details
3. ☐ Review

1) Design of the report outline

a. Standardize the output format

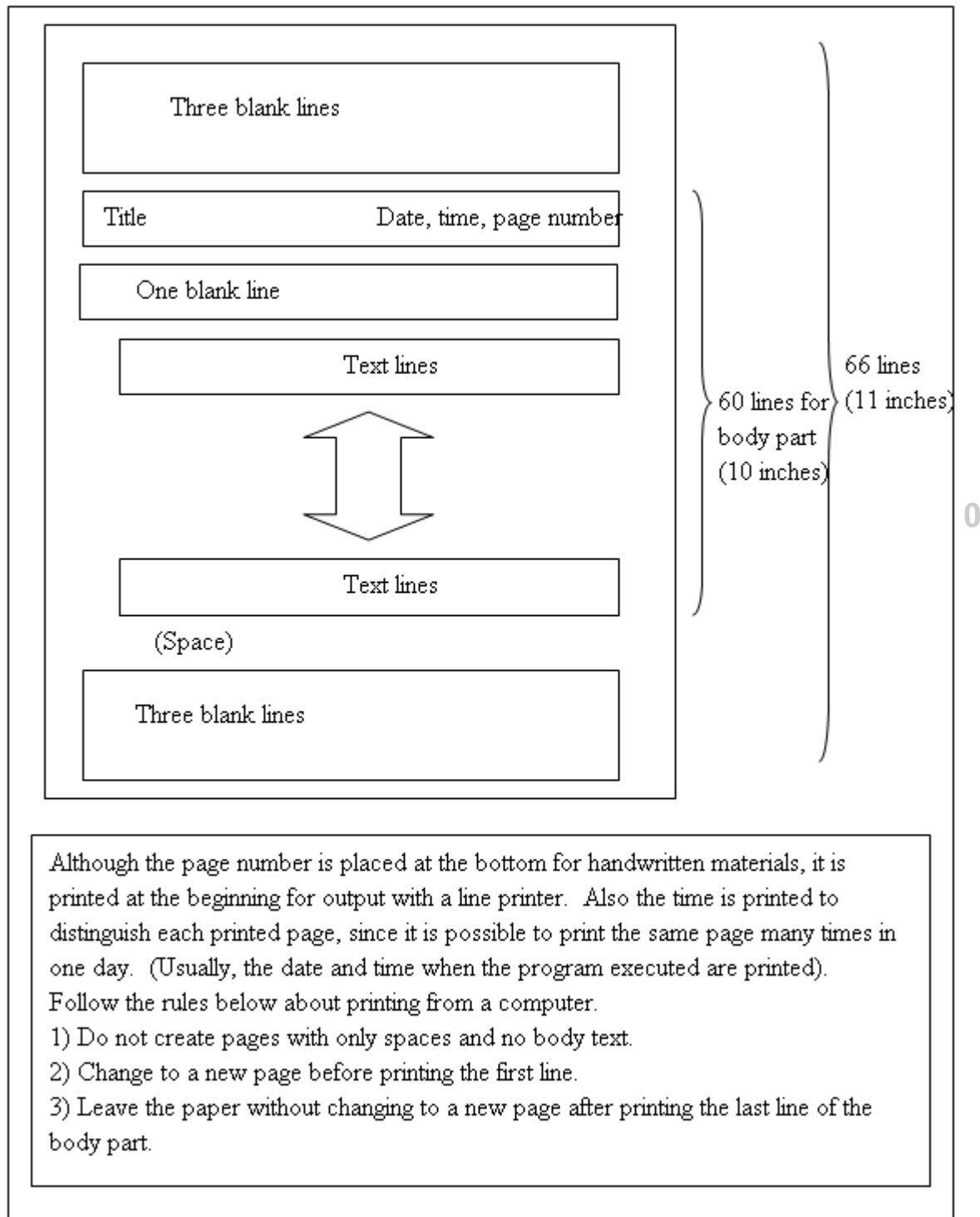
The most important factor for reports is that they are easy to read and understand. If standardization is not taken into consideration in the design, the total system will be disorganized and difficult to use. For that reason, standardize the output formats of the reports (e.g., titles, headers, organizations of details). An example report layout for a sheet of paper from a line printer is given below:

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

<Example page layout for sheet of paper from a line printer sheet>



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

Only for the last Per page

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.

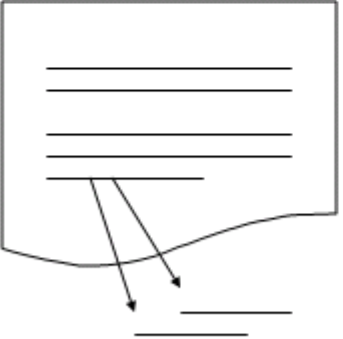
d. Create a specification of the output report outline.

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.

VITEC

<http://www.vitec.org.vn>

<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.

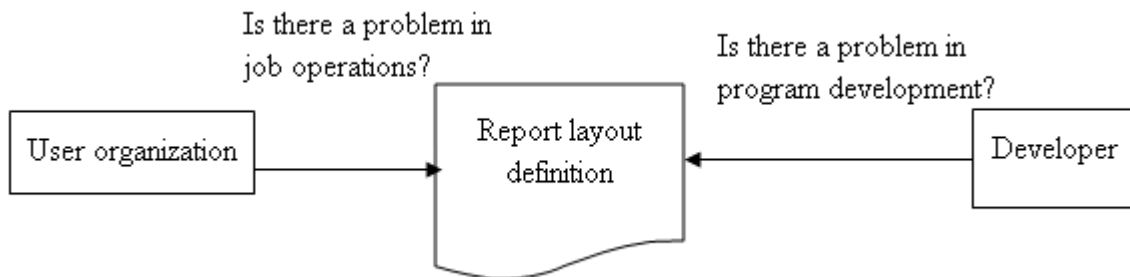
Considering the output processing requirements and sequence discussed, define the report layout in detail. For users, such a definition of the report layout is designed to confirm the specific appearance of the report, and for programmers, it serves as a specification for program design. Therefore, this 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.



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.

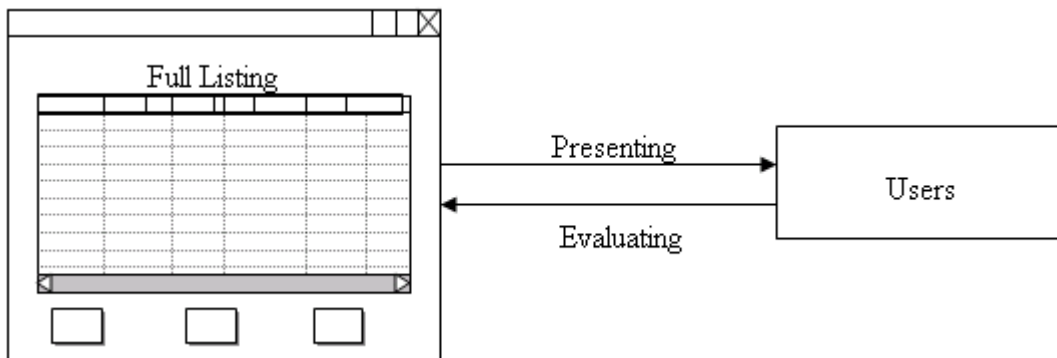
The following steps are taken in layout design:

1. Creating a screen image
2. Standardizing the screen configuration
3. Creating a screen transition diagram
4. Creating a screen specification
5. Review

g tin và Hỗ trợ đào tạo

(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.

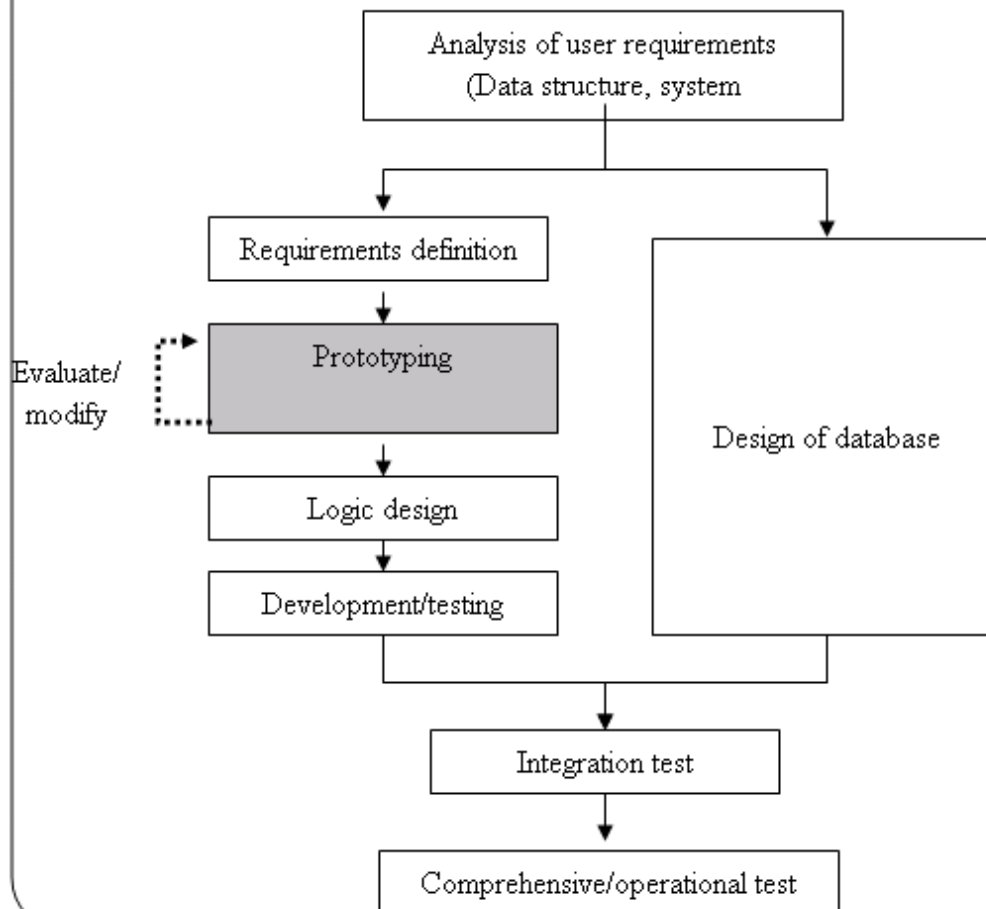


The operability and ease of viewing are difficult to check using only a specification on paper in developing a new job system, and users may submit requests for specification changes during the testing stage after the system is created.

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.

<http://www.vitec.org.vn>

(3) Procedures for creating a screen transition diagram

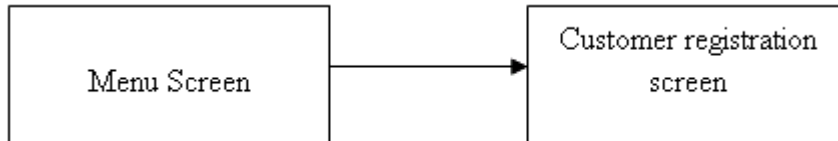
Once an outline of the screen is completed, summarize the correlation of screens in the screen transition diagram. Particularly with GUI-based applications, the screen transition diagram becomes more complicated than that for CUI-based applications. This is because a number of event processes are required, 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.

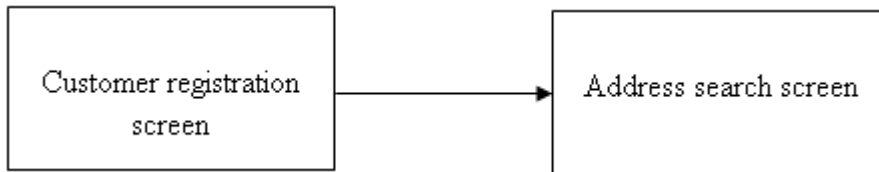
- Simple screen transition

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.



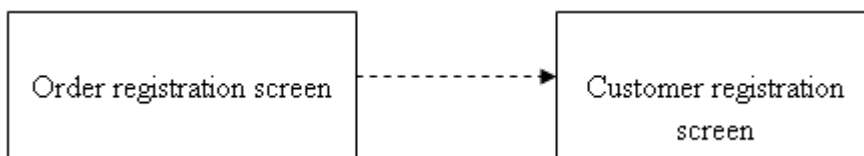
- Transition to an independent child screen

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.



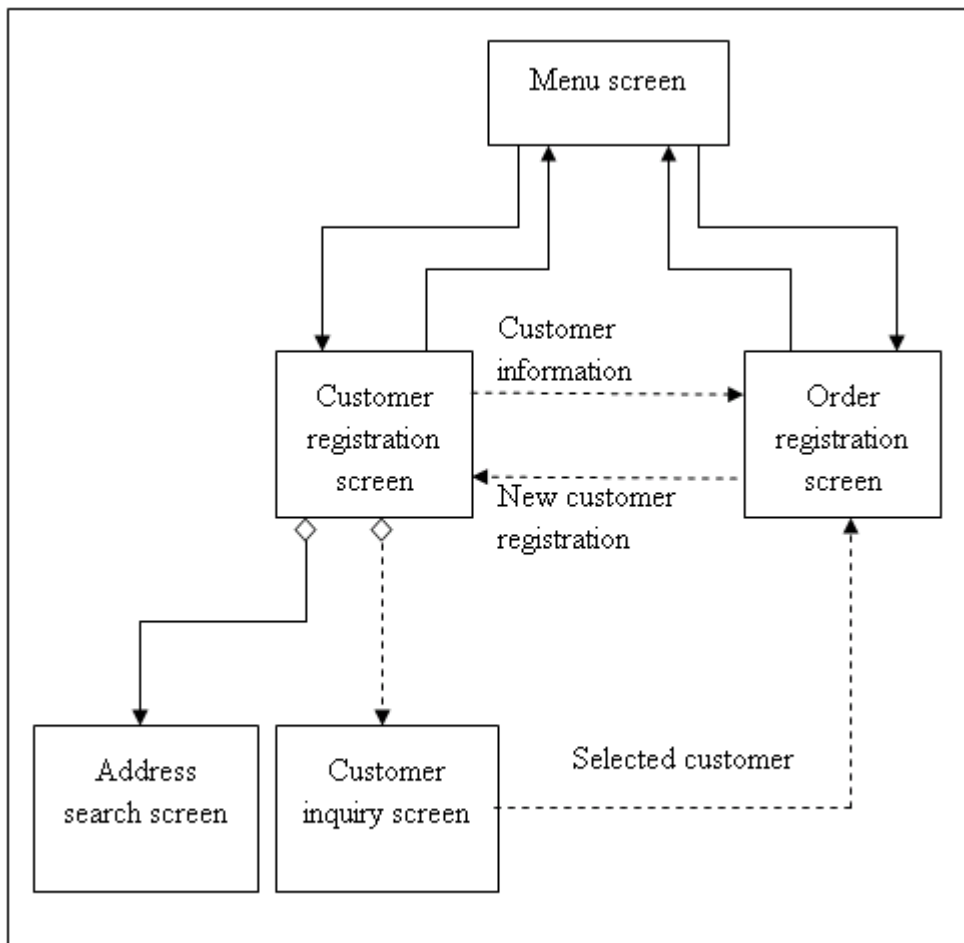
- Transition to an independent screen

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.



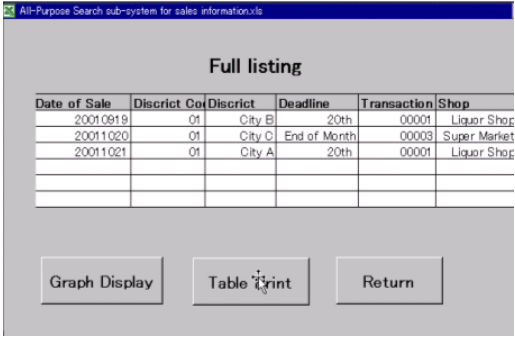
An example of a screen transition diagram

<http://www.vitec.org.vn>

(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:

Liquor sales basic system (general-purpose search subsystem for sales information)		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Displaying detail table				
 <p>Area for displaying detail table</p>		Control	Operation	Function	
			Initial	-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.	
		Graph display button	Click	Displays the graph display screen	
		Table print button	Click	Displays the print preview screen	
		Return button	Click	Displays the search specification screen	

<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.

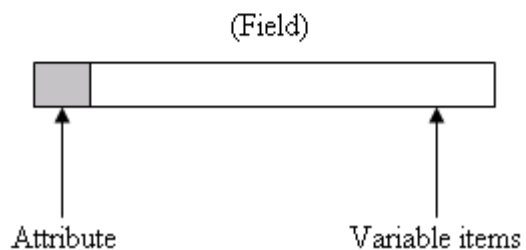
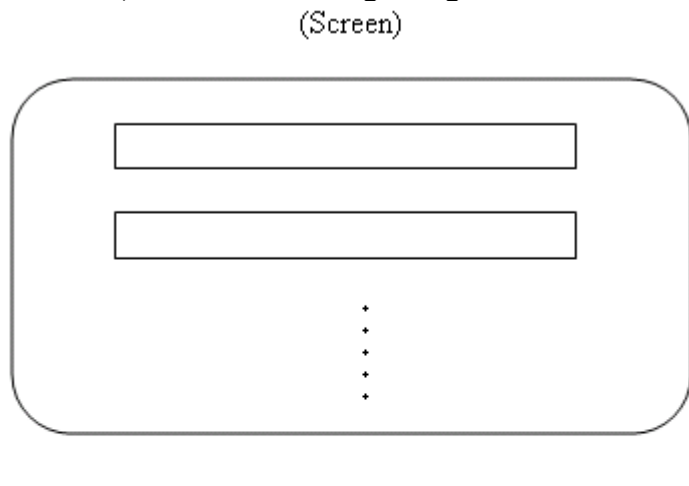
b. List of functions

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 <i>ZZZ, ZZZ, ZZ9</i>

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

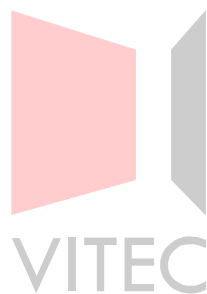
Displayed in the upper part of a window or screen, it can be used to select an item in the menu.

- 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.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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.

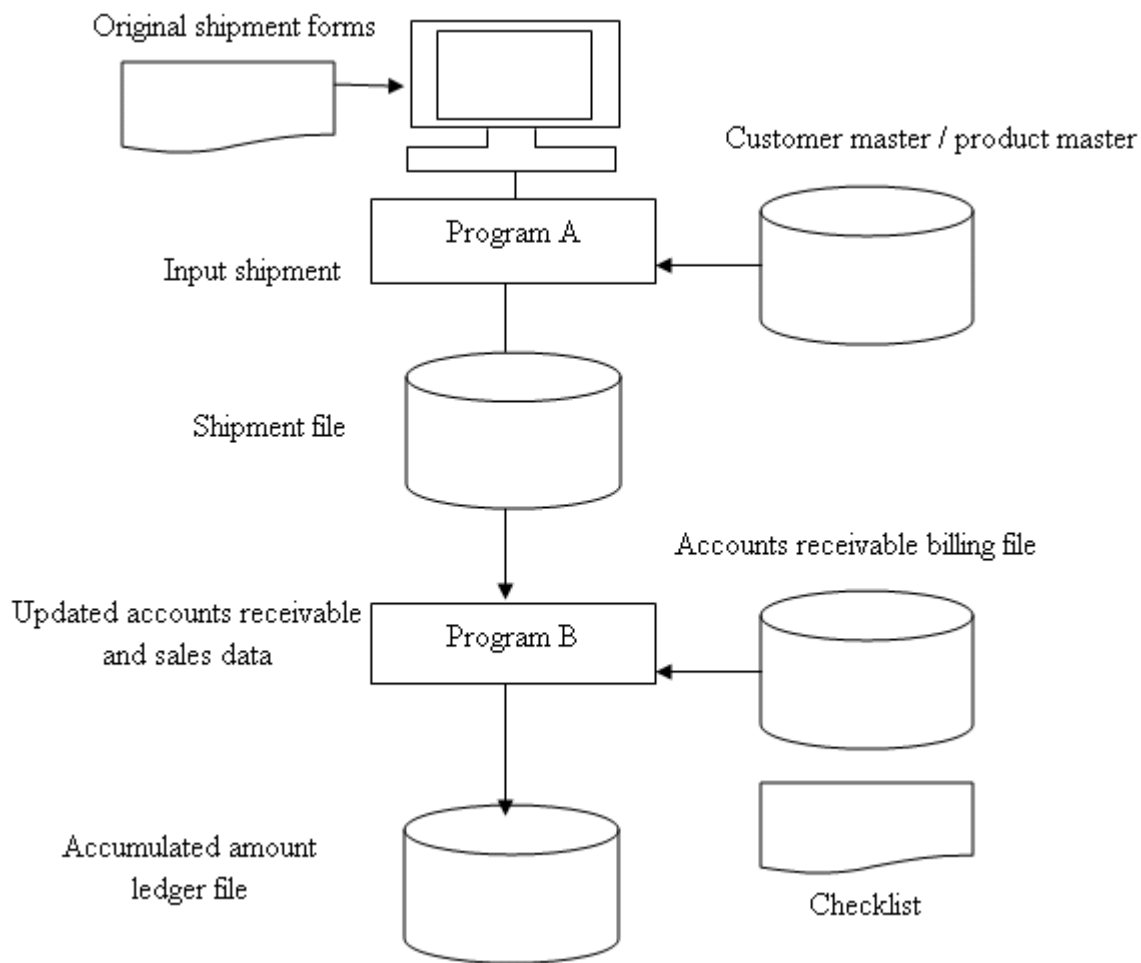
To meet these control objectives, data processing procedures by business applications (e.g., check) must be combined appropriately with a series of manual steps for input and output. In other words, it is essential to establish and operate a business system that not only performs computer processing, 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.



<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

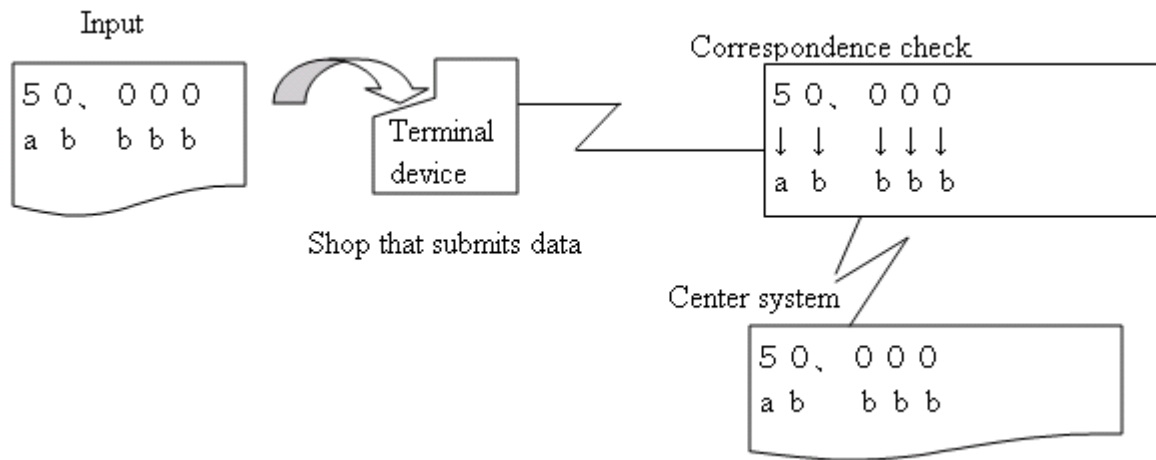
Calculate the total of a specific data item manually before computer processing. Then, calculate the total for the same item with a computer, and compare the two totals. By using this approach, it is possible to check if any data has been omitted in the batch.

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.



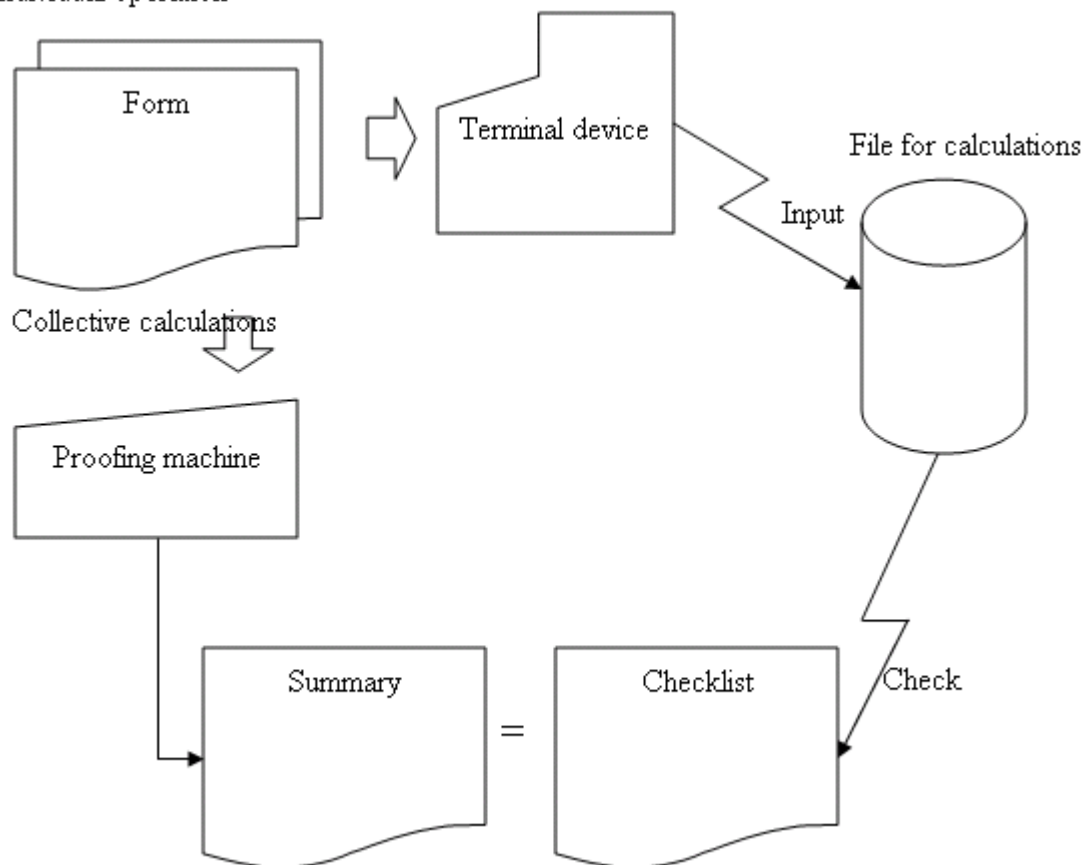
b. ID check

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.

Individual operation



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	$ \begin{array}{r} 1\ 2\ 3\ 4\ 5\ 6 \\ \times \text{XXXXX X} \\ \hline 1\ 2\ 1\ 2\ 1\ 2 \\ 1\ 4\ 3\ 8\ 5\ 12 \end{array} $
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.	$24 \div 10 = 2 \dots 4$ 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

VITEC

<http://www.vitec.org.vn>

	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

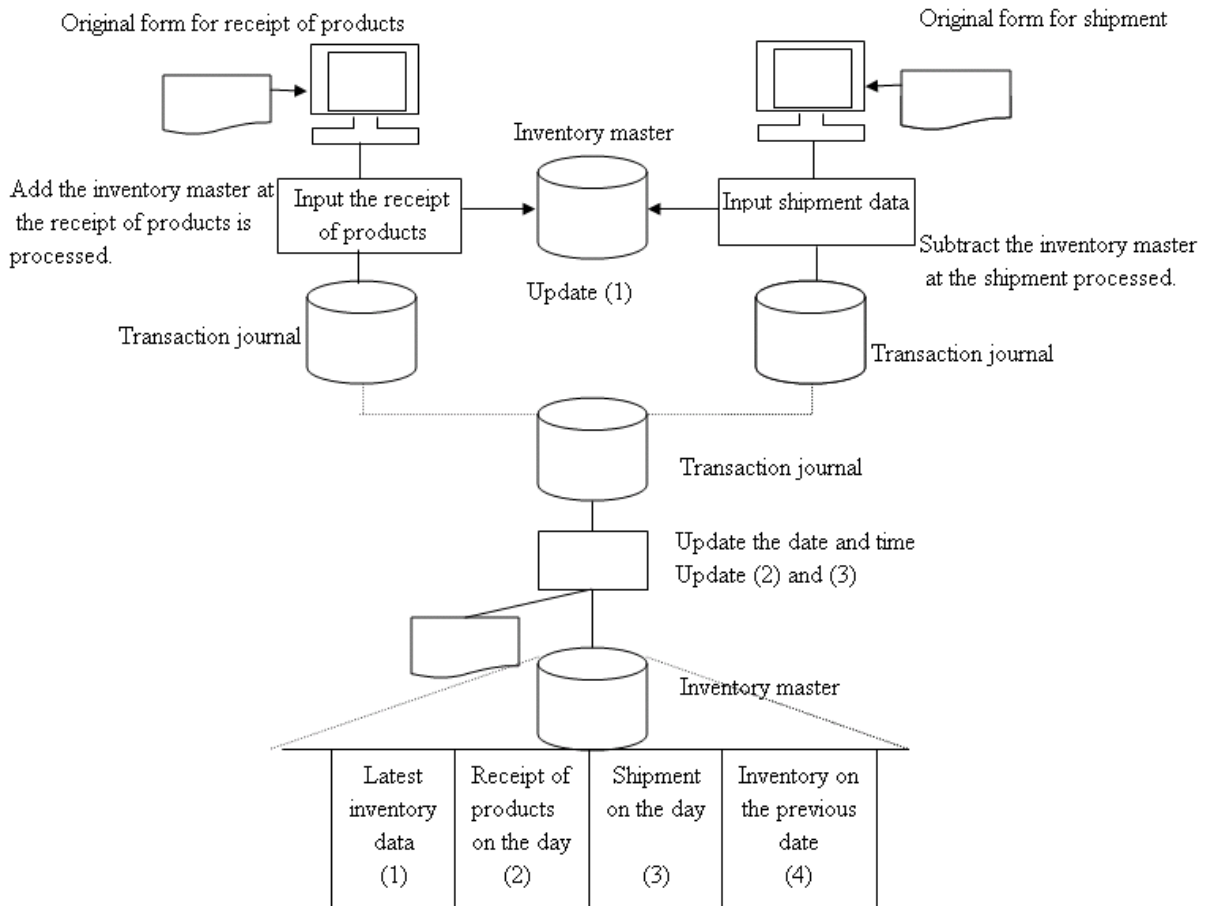
Controlling the completeness of update targets not only at the time of updates, but also in a series of processes beginning from identifying the transaction file received until processing immediately after the update is finished.

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.



If $(1) = (2) - (3) + (4)$ is not valid, the conclusion is that there is 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.

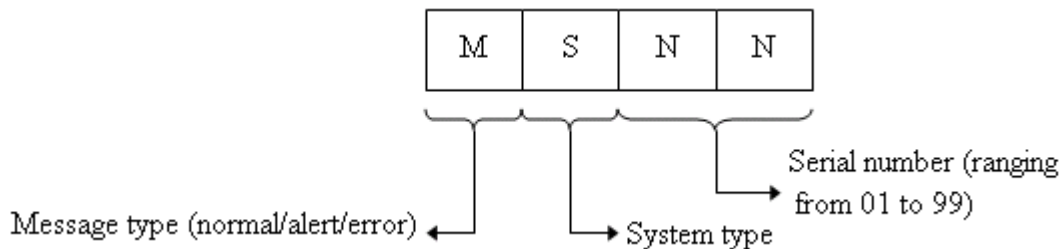
c. Standardizing log messages

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.



2) User messages

User messages are outputted from the client terminal in the following events:

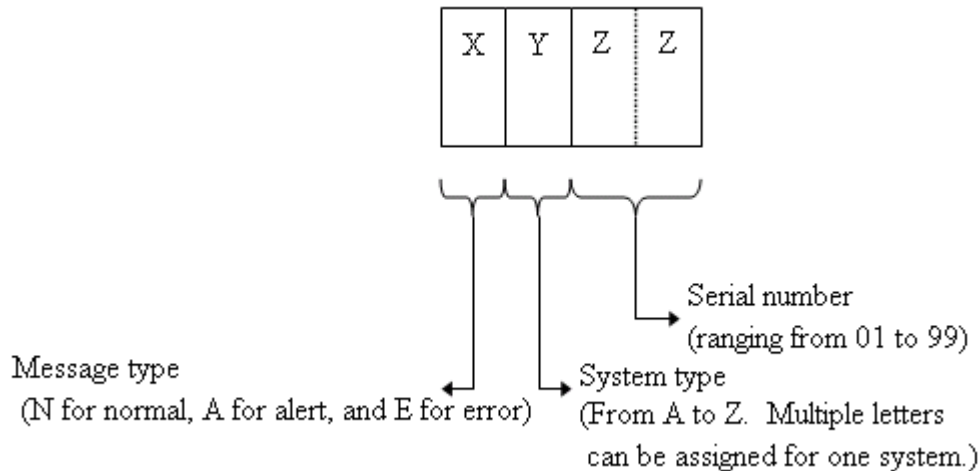
- When it is reported that transaction processing has completed normally or abnormally
- When it is reported that an erroneous value is found during the input check

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 >



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 XXX.XXX.XX is wrong. Please correct it.					
(Time when message is issued) When invalid data is entered	(Embedded information) xxxxxxx 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.

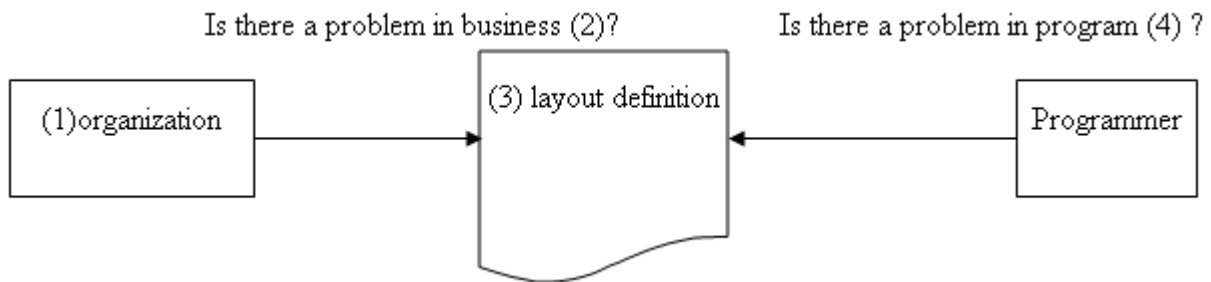
It is important to choose input methods that is less (1) for users and is less likely to cause (2) in daily and routine jobs.

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 elements of the report design process. From group of choices listed below, select the best terms to fill in the blanks and complete the diagram.



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.

<http://www.vitec.org.vn>

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.

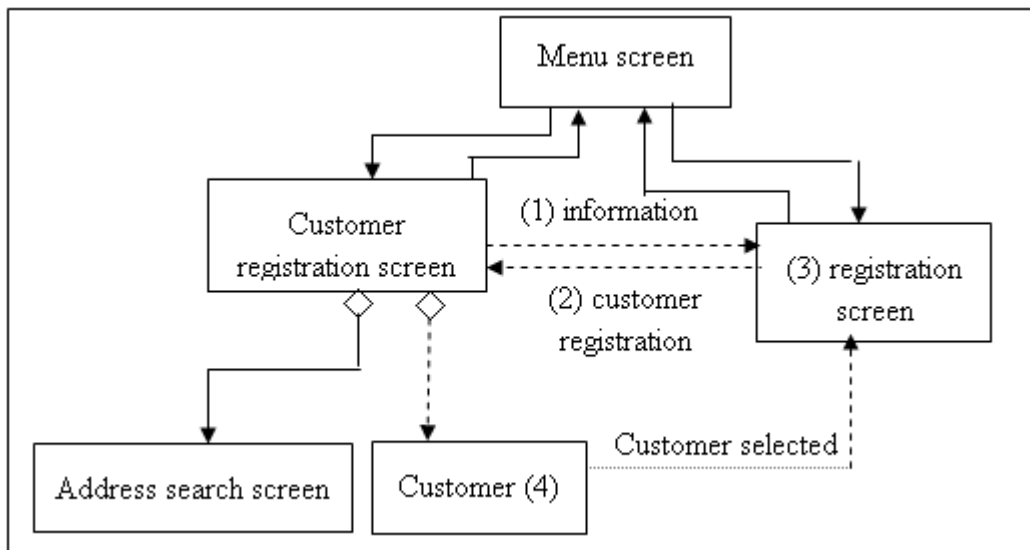
Used to ask for a user's response or give instructions, this is also displayed when instructions are required from a user or a message has to be issued.

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

17. What type of check is indicated by the following sentence? From the group of choices listed below, select the best term.

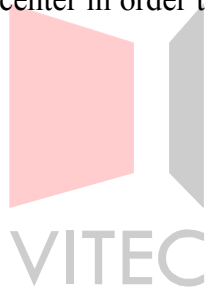
Check whether the input value is valid data by creating a table defining the allowable range of input values.

Answers:

- a. Combination check
- b. Batch total check
- c. Range check
- d. Balance check
- e. Limit check

18. From the following sentences, select the one that explains check digit.

- a. Input a numeral string and character string (double input) for a variable data item, such as an amount of money, to check the correspondence between the two strings on the receiving side.
- b. 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. 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.
- d. 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.



<http://www.vitec.org.vn>

4 Physical Data Design

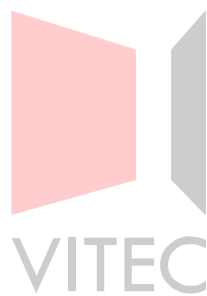
Chapter Objectives

This chapter explains the physical design for files and databases. The explanations related to files focus on file organizations and physical structures. The explanations related to databases focus on the concept of and method for mapping a logical data structure to a physical data structure.

4.1 File Design

4.2 Physical Data Design

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

4.1 File Design

Files are used to exchange data between components in the system, to temporarily store data entered from the terminal, to search for common information stored in master files in various processes, and for other purposes. To design these files beforehand is important for the system development.

File design can roughly be divided into logical file design and physical data design. In the logical file design, the specification of the logical aspects of file structures is examined from the point of view of job processing functions. In physical data design, the physical storage formats of records are examined.

Work category	Design item
Logical file design	<ul style="list-style-type: none">- Select data items- Define the record layout- Determine the file organization- Determine the key and alternate key items- Define the file structure (for a database)
Physical data design	<ul style="list-style-type: none">- Determine block length, CI length, logical page length- Estimate the empty space- Estimate the file capacity

The purpose of logical file design is to design data items for each file after the file structure has been clarified in a system flowchart or by using the conceptual data model in a data-oriented approach.

At this stage, files are only considered to the extent that they are required in accordance with the logic of job processing. Intermediate files used for the convenience of computer processing or program management are not considered yet.

At this stage, files are sorted and files are also defined as interfaces to the extent that required for dividing programs to keep programs from becoming too large. These files are prepared at the stages of detailed design such as system structure design.

4.1.1 Analysis of data characteristics

(1) Confirmation of file usage and use mode

Files are used in the system in a variety of ways. The table below lists the most commonly used file types. Note, however, that the type of items stored in the file, the storage period, the required file capacity and other characteristics may differ depending on the use.

File type	Description	Example
Master file	Basic processing file for a specific purpose. This file stores information permanently or semi-permanently. The cycle from record creation to deletion is long.	- Customer master - Commodity master - Inventory master
Transaction file	File for temporarily storing data generated at a specific timing. For system auditing, it is preferable to save and store this file as source data. This file is mainly used to store master update data.	- Sales journal - Order backlog file - Money receipt file
Accumulation file	File for editing transaction data for a long period (month or year) depending on the purpose	- Sales accumulation file - Billing accumulation file
Aggregation file	File used to accumulate and reorganize the contents of other files in order to improve a specific aspect of processing efficiency	- Sales file classified by customer - Shipping file classified by area and by commodity
Management file	File for recording control information for executing application programs and improve operational efficiency	- Automatic numbering file - Parameter file
Succession file	Interface file for transferring data between subsystems	- Sales achievement data
Library	File for storing and managing software resources required for system development and operations	- Source library - Load module library and definitions
Backup	Master maintenance file (for managing the number of generations)	- Inventory master backup
Log file	File for recording update data for recovery from a file failure	- History log file - Backout file
System file	File used by a control program for system execution management	- Page file - Spool file
Work file	File temporarily used for a specific type of processing	- Sort work

(2) Extracting and arranging required data items

Data items are extracted and distributed to each file by two kinds of processes:

1) To extract data items, track the workflow in a component relationship diagram, consider the data type and the timings of creation, update, and deletion, and check documents and existing files.

Let us first consider the inventory master. Suppose that the inventory master is used in a processing flow from order reception to shipping and purchase. For each process, extract the required items sequentially as shown below.

Analyze the processing flow and the linkage of files and data items

Order reception item	Shipping item	Purchase item	
Merchandise code	Merchandise code	Merchandise code	
Merchandise name	Merchandise name	Merchandise name	⇒ To merchandise master, because this item does not change
Selling price	Selling price	Selling price	⇒ To merchandise master, because this item changes very rarely due to the characteristics of merchandise.
Stock quantity	Stock quantity	Stock quantity	
	Shelf number	Shelf number	
		Stock date	
		Purchase unit price	
		Vendor code	⇒ Managed by using merchandise master
		Vendor name	⇒ Managed by using merchandise master

[Inventory master]

2) Study to which files the extracted data items should belong as attributes. Determine the files to which the data items belong, and gather related items. (Delete duplicate data items)

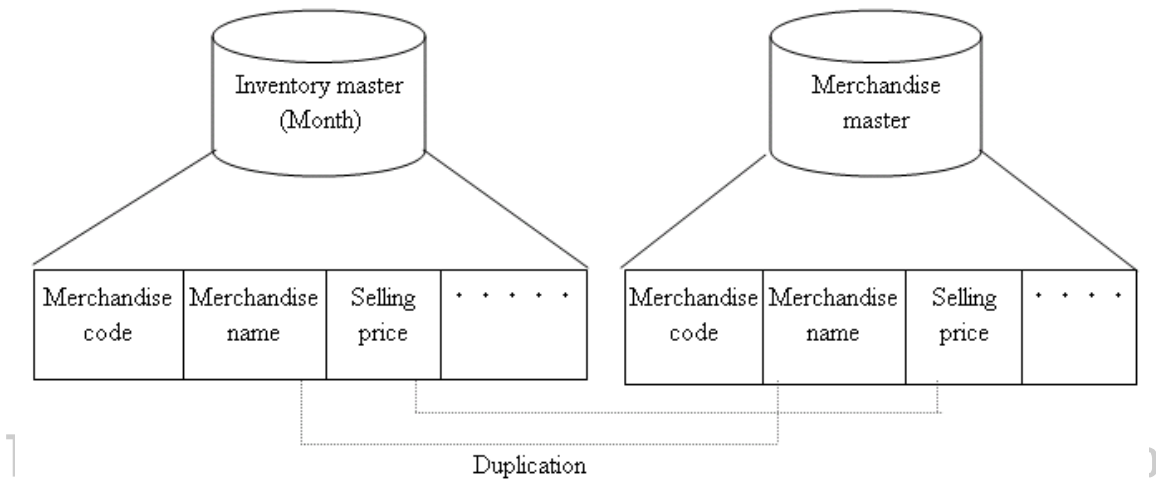
Let us consider the inventory master and merchandise master. The inventory master is a monthly file for managing the merchandise stock status. Suppose that a merchandise name is entered into the merchandise master at the stage shown in the previous figure and also contained as a data item in the merchandise master storing basic information.

Although the merchandise name is also contained in the inventory master, only one item of merchandise exists physically. For this reason, it would be more natural to have also only one unique instance merchandise name data.

When comparing the inventory master and the merchandise master, the merchandise master seems the more appropriate (natural) choice to store merchandise name data.

Duplicate data items may place a great burden on the maintenance and management of software resources. During file design, duplication of data items should therefore be minimized and file consistency should be maintained.

For logical file design, there is no need to consider computer processing and physical characteristics. Priority should be given to the logic of job processing. Considering computer processing and physical characteristics during physical data design improves the quality of design.



After extraction of data items for each file, the contents of design are detailed in the subsequent steps of system design. Addition and correction of data items might become necessary at this stage. Review the data items from the following points of view:

- Review the job data items for items to be added (cumulative and aggregation items) or to be deleted.
- Extract necessary data items (code category and decision flags) to be added for program creation.
- Analyze whether to add items that are likely to become necessary for future system extensions.
- By considering the processing efficiency and the efficiency of using hard disk space, analyze whether to move or delete data items.

4.1.2 Determination of the logical data organization

(1) Determining the file organization

Control programs support several kinds of file organizations.

Select a file organization and database management system (DBMS) from the points of view of program processing conditions (sequential, random, or combined processing), performance, measures against failures, and ease of creation or reorganization.

The file should be designed by considering not only the file organization method and configuration but also access performance, measures against failures, and compatibility with other files.

<Check items>

- Main access method (random or sequential)
- Sharing of write file or database (subject to exclusive control)
- Frequency of data additions (every day, month, or year)
- Necessity of assuring data correctness when a system failure occurs (Necessity of recovery design)
- Amount of created access requests (access load)
- Flexibility with respect to job addition or system changes
- Reference used for access (key items)

The following file and database types exist:

1) General files

General files can be classified into sequential, direct, and indexed sequential files. The table below summarizes the characteristics of these file types.

Characteristics of file organizations

Organization	Characteristics	Access method	Method for creation or reorganization	Measures against failures
Sequential file	Records on storage media are accessed in order in which they are physically stored. Direct access is not possible.	Sequential	Operating system utility or user program	User's responsibility
Direct file	A record is accessed by obtaining its location using a calculation formula	Random	User program	User's responsibility
Indexed sequential file	Records are arranged in logical sequence by key. Indexes to these keys permit direct access to individual records	Sequential and random	Operating system utility or user program	User's responsibility
VSAM file	Sequential, direct, and indexed sequential organizations are integrated for sequential and random access.	Sequential and random	Creation utility (The VSAM environment should be set.)	User's responsibility

2) Other files

Moreover, the following file organizations are used:

- Partitioned file

Partitioned files are used mainly for program libraries and only seldom as data files. This type of file consists of multiple small files, which are called members or segments. The segment files are managed in directories; these directories can be organized in a hierarchical fashion.

- Inverted file

An inverted file is designed for direct access to a record of a specified file data item. Disk addresses and other information are stored in separate file and that file is used as an index for efficient input and output operations.

3) Measures against failures

For protection from hardware failures, a journal or backup file is created or the file is duplicated.

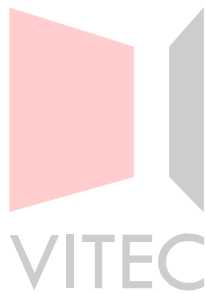
For protection from a software failure, a form of transaction control called a rollback or commitment is applied. Suitable post-processing after failure occurrence can then be designed using the exceptional event notification function of the corresponding access method.

4) Access performance

To evaluate file access performance, the average access time is calculated from the disk access time and the operating system overhead. The number of file accesses in each process is then calculated and the access time is estimated considering the average access count and average access time.

If a large number of accesses is anticipated at a certain time period, multithreading the access routine should be considered. (Thread: unit of processing when a component is further divided)

Using compression and decompression is also helpful for reducing the overall size of files.



<http://www.vitec.org.vn>

4.1.3 Selection of data storage media

When selecting storage media, the required processing speed, storage area, ease of handling, and similar job characteristics should be considered. Moreover, consideration should be given regarding whether the media is to be for general use or for backup purposes.

File storage media can be classified into three major types - card, tape, and disk.

Card type	Punched card	
	Magnetic card	
	Barcode tag	
Tape type	Paper tape	
	Magnetic tape	Cassette
		Cartridge
		Open reel
Disk type	Magnetic disk	Hard disk
		Floppy disk
	Magneto-optical disk	
	Optical disk	

Tape-type media is suitable for sequential access because data is arranged sequentially from the start point to the end point. Since tape is suitable for storing a large volume of data, it is suitable for the backup or physical transport of data.

Disk-type media is suitable for direct access because it allows data to be fetched in random access. Since disk-type media allows data to be updated with ease, this type of media is suitable for time-critical operations.

4.1.4 Record layout design

Before designing the layout of records, data item duplications should be minimized. This section explains how to normalize data and to eliminate duplicate data items.

(1) Data normalization

Data normalization is a procedure for arranging data in a manner that eliminates duplications.

In this procedure, a record to be analyzed is divided in three steps, from the first normalization to the third normalization. After the third normalization, the record has no data duplications and can be used as a basic unit for file or database design.

The normalization process is explained using the example of a sales record having the repetitive structure shown below. The sales slip number is used as an identifying key to distinguish a record.

Sales record

Sales slip No. ■	Date	Customer code	Customer name	Address	Telephone	Merchandise code	Merchandise name	Unit	Quantity	Unit price	Amount

■ : Identifying key

Repeated five times

1) First normalization

The detail section appears five times in the sales record that is to be analyzed. The first normalization separates this repetitive section from the record and divides the record into flat records separated.

The separated records require a new key for identification. In the case below, the separated detail record cannot be identified by using the sales slip number alone, but requires the use of a merchandise code to distinguish each detail. In other words, a key having the form <Slip number + Merchandise code> must be used.

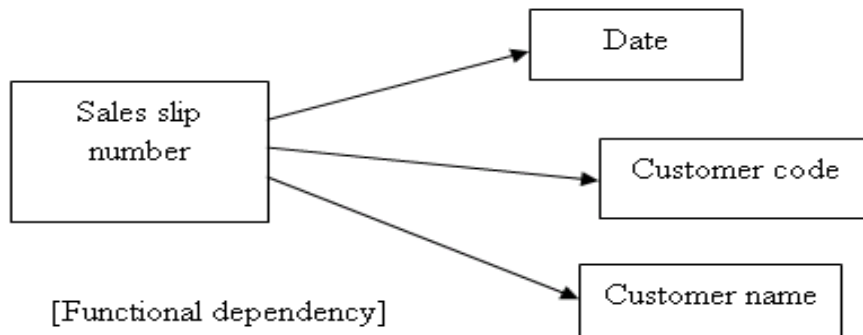
A key that is a concatenation of several ID keys that are produced by the first normalization is called a concatenated key.

Sales slip No. ■	Date	Customer code	Customer name	Address	Telephone
------------------	------	---------------	---------------	---------	-----------

Details	Sales slip No. ■	Merchandise code	Merchandise name	Unit	Quantity	Unit price	Amount
---------	------------------	------------------	------------------	------	----------	------------	--------

Concatenated key

Once the sales slip number has been determined for the above record, the date and customer code are automatically determined. In other words, determining one data value determines the value of another data item. This relationship is called functional dependency.



2) Second normalization

The second normalization is performed using the concatenated key of the separated record. Data items that are functionally dependent on a part of the concatenated key are extracted and separated. For a sales record, the merchandise name, unit, and unit price are determined once the merchandise code has been determined. Therefore, the record can be divided as follows:

Sales slip No. ■	Date	Customer code	Customer name	Address	Telephone

Sales slip No. ■	Merchandise code ■	Quantity	Amount

Merchandise code. ■	Merchandise name	Unit	Unit price

3) Third normalization

After the second normalization, functionally dependent data items other than the ID key are extracted and separated. Since the customer name, address, and telephone number are functionally dependent on the customer code in the above example, only the customer code is left in the original record. This remaining key is called the reference key.

Sales slip No. ■	Date	Customer code
---------------------	------	---------------

Customer code. ■	Customer name	Address	Telephone number
------------------	---------------	---------	------------------

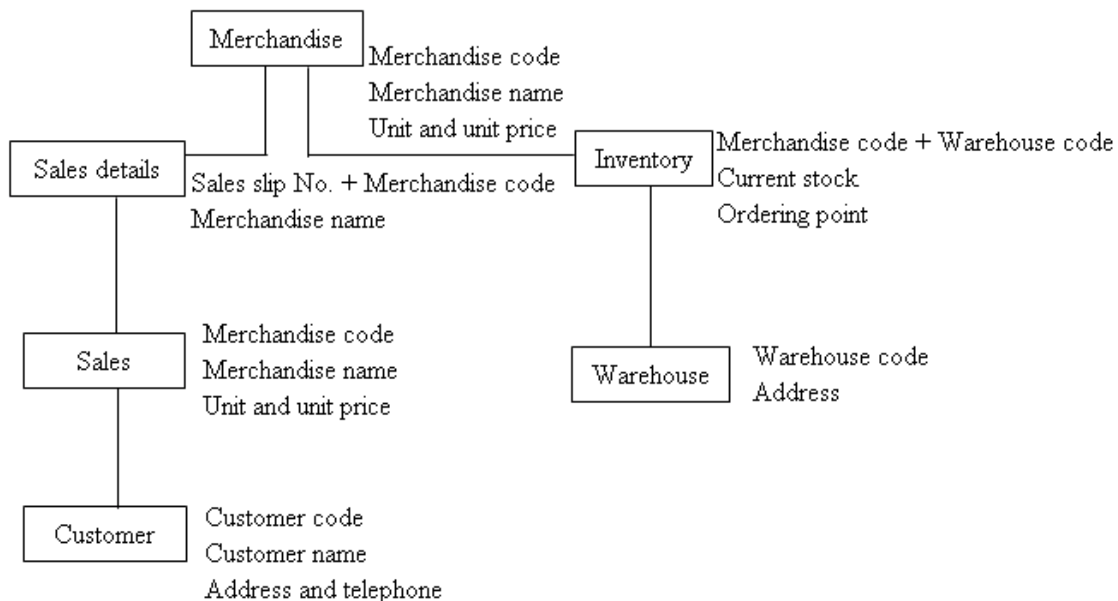
Sales slip No. ■	Merchandise code ■	Quantity	Amount
------------------	--------------------	----------	--------

Merchandise code. ■	Merchandise name	Unit	Unit price
---------------------	------------------	------	------------

4) Arrangement of reference relationships between records

The records produced by the above mentioned third normalization are small but contain data items of clear related meanings. If a data item is changed, it is sufficient to replace only a specific record without affecting other records.

After records other than sales records are normalized and their reference relationships are organized, the overall organization of data within the system will become clear. A diagram representing relationships between records, called a conceptual model, can then be produced.



In the next step, logical file is designed based on the conceptual model obtained as explained above. First, the conceptual model is divided into units which can be handled with the DBMS, such as a network database or relational database, which runs on the operating system. Files and databases are designed after this division takes place.

(2) Determination of the number of digits and the data format

Once a data item in a record has been determined, the number of digits and the data format are determined for it with the following in mind:

- Determine the number of digits by considering future extensions

If the data volume increases or it becomes necessary to add new information to the existing code, an inadequate number of digits will make it difficult to take appropriate measures. In preparation for these cases, a spare area should be reserved for following the item for which the number of columns is likely to become insufficient.

If it is a numeric item, change it into a fixed point format.

- Determine a data format for high processing efficiency by considering the language to be used

When designing the data format, the language to be used should be checked and the use of numeric values should be considered well in advance. (For example, when handling quantitative items in FORTRAN, the packed decimal format cannot be used).



<http://www.vitec.org.vn>

Quantitative item	Advantages/ typical use	Disadvantages
Zone decimal	<ul style="list-style-type: none"> - Used for input-output (for display of data without conversion) - Check of file contents by using a dump listing becomes easy 	<ul style="list-style-type: none"> - Items are long, which is not economical - Not suitable for internal processing, because it takes long time to convert packed data
Packed decimal	<ul style="list-style-type: none"> - Used for internal processing - Check of file contents by using a dump listing becomes easy 	<ul style="list-style-type: none"> - An area greater than that for fixed floating-point format is required - Low computing speed
Fixed point format	<ul style="list-style-type: none"> - Large numeric values can be expressed with a few bytes, which makes it possible to keep the storage area small - High computing speed 	<ul style="list-style-type: none"> - Requiring boundary adjustment - Check of file contents by using a dump listing becomes difficult
Floating-point format	<ul style="list-style-type: none"> - Allows handling of wide-range high-precision data - Used for scientific and engineering calculations 	<ul style="list-style-type: none"> - Requiring boundary adjustment - Check of file contents by using a dump listing becomes difficult - Is not useful for business data processing

(3) Determination of item layout

Once the format of each data item has been determined, data items must be arranged within records. Data items are generally arranged in the following order:

- Key items (data used for record identification)
- 2) Record deletion symbol, exclusive control flag, and the like
- 3) Code category, creation date, and other common items
- 4) Display items
- 5) Calculation items
- 6) Spare areas

Note also the following points:

- Mutually related items should be placed closely.
- Spare areas should be distributed to several places.

(4) Determination of the record type

Record can be classified into fixed-length records, variable-length, and undefined-length records. Variable-length records contain information about the record length, while undefined-length records do not.

(5) Creation of file record specifications

Create the file record specifications with the following points in mind.

- Specify the following elements in the file specifications
- Media: Media type
- File organization: File organization type
(Sequential, partitioned, direct, relative, indexed, sequential, VSAM, etc.)
- Record organization: Fixed or variable
- Blocking factor: Number of records in a block
- Record length: Number of bytes in a record

2) Specify the format and contents of each data item.

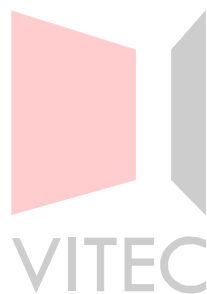
The contents, meaning, and use of each item must be described.

In particular, the range of probable values for a data item and the decision codes and their meanings should be clearly specified.

3) Enter data sizes and other information into the remarks.

This information is useful for estimating performance and capacity.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

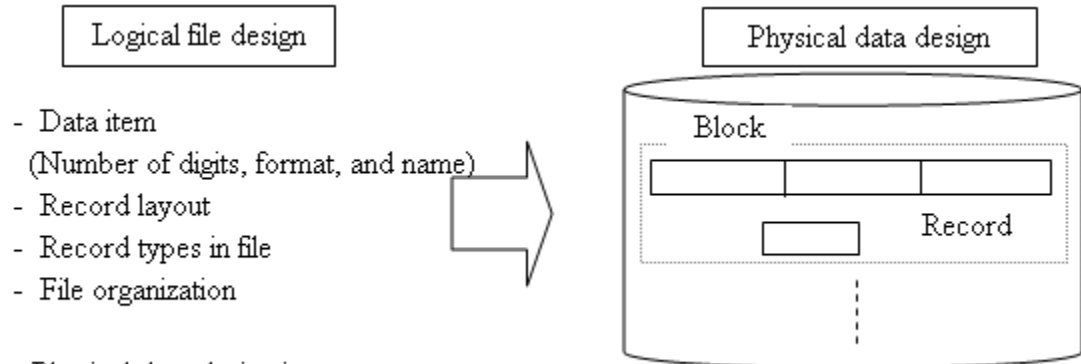


<http://www.vitec.org.vn>

4.2 Physical data design

4.2.1 Mapping of the logical data structure

During logical file design, the file specifications necessary from the viewpoint of processing functions were examined for specifying a file type necessary for job processing, extracting data items for each file, and determining the file organization. In the physical design, records' physical storage structures, such as setting of the block length and blank space, are examined for each file.



- Physical data design items

The items of physical data design depend on the file organization. The table lists typical file design items.

<Design items classified by file organization>

File organization	Design item	Consideration
Sequential organization file	<ul style="list-style-type: none"> - Block length (blocking factor) - File capacity estimate 	If the block length is inadequate, the storage efficiency decreases or the time required for I/O operations increases.
VSAM	<ul style="list-style-type: none"> - CI length and CA length - Index CI length - Ratio of blank space in CI - Ratio of blank space in CA - File capacity estimate <p>(Notes☺ CI: Control interval CA: Control area</p>	Decrease frequency of CI/CA partitioning.
Database	<ul style="list-style-type: none"> - Storage design (Record arrangement in the storage space) - Record storage characterization (Division into sub-ranges) - Record size estimate - Logical page size estimate - Capacity estimate for each storage area - Physical page size estimate 	If a logical record is not stored in a logical page, the processing efficiency will decrease. Moreover, overflows can occur.

(2) Considerations for physical data design

The design of the record storage structures greatly affects the processing performance of the system (response time, batch processing time, and other performance-related characteristics), file capacity, and file reliability. Depending on the design, the processing efficiency might decrease or hard disk space might be wasted. Especially the physical data design for frequently accessed files or large-capacity files requires meticulous care.

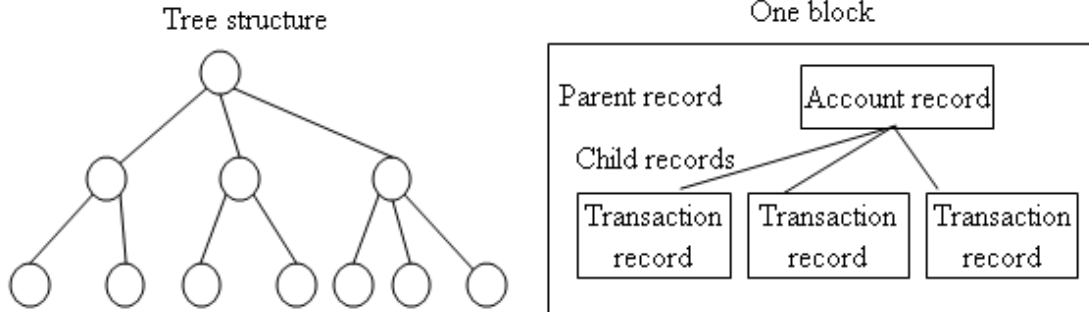
The items for consideration depend to a large extent on the file organization. Some considerations that would apply commonly are as follows:

1) Database

There are three typical database models -- hierarchical database, network database, and relational database.

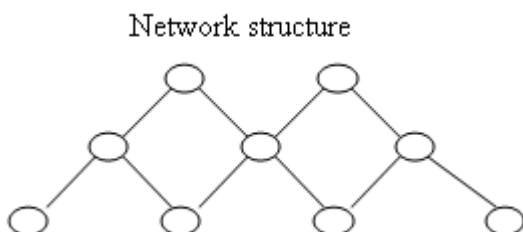
- Hierarchical database (Tree structure)

A hierarchical database has a tree structure. Several child records correspond to one parent record. A child record has only one parent record. In case of a banking application, a parent record may be an account record and its child records may be account transaction records. If the relationship is stored in a block on the disk device, the contents of transaction records and the past transaction history data can be fetched by a single access. Conversely, general files require separate accesses to the account master and transaction master. This increases the access time and results in a complicated processing algorithm if the contents are stored in a single file. Using a hierarchical database eliminates these problems.



- Network structure (Network database)

A network database has a network structure. Multiple child records correspond to one parent record but each child also has multiple parent records. This type of database, which is an extension of the hierarchical database model explained above, allows a child record to have multiple parent records, which increases the flexibility of the database. With respect to improvements in access efficiency, the same comments apply as those for the hierarchical database model.



- Relational model (Relational database)

In a relational database, the structure of the database is not fixed, but relationships between data items are represented by the logical linkage of elements listed in the table. E. F. Codd proposed this model in 1970 and commercial versions of this model were announced successively in the 1980s. The basic data operations of this model are Selection, Projection, and Join.

Example of table manipulation

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

Table join

<http://www.vitec.org.vn>

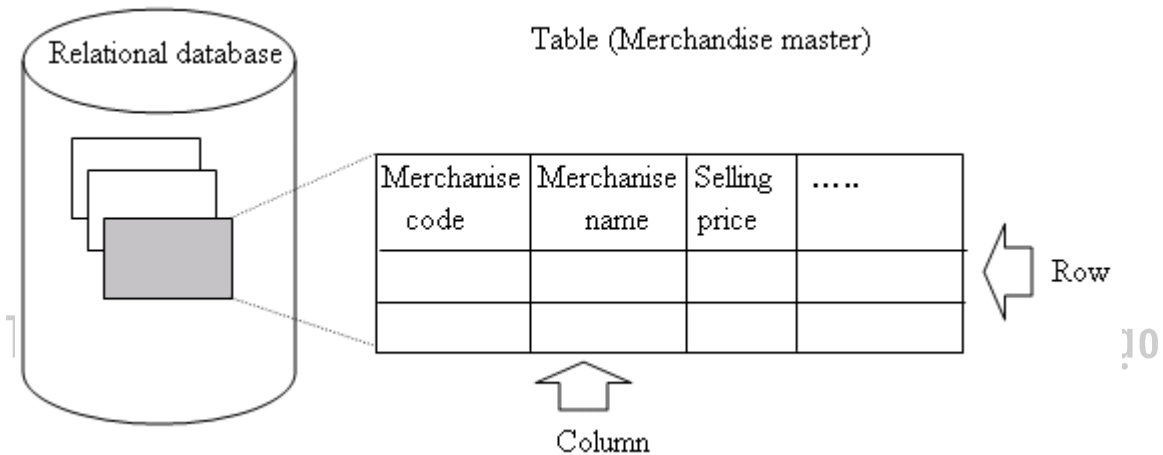
A hierarchical database or network database is not suitable for non-routine processing, because any design changes will make it necessary to start the design process again from the beginning. On the contrary, a relational database allows dynamic creation of tables and is suitable for non-routine processing. Since the access efficiency of the relational databases is comparably low, this type of database should only be used with care.

(3) Design of a relational database

A relational database is created on the basis of a conceptual data model created during the system analysis stage. The database is designed from the aspects of logical structure, storage structure, and physical structure. Among these three aspects, the logical structure is designed during logical file design by the following procedures:

1) Table design

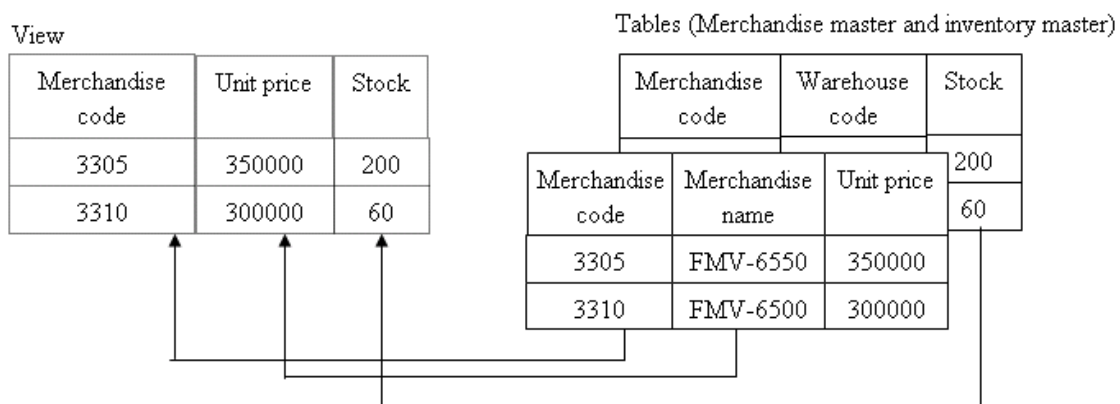
Based on analyzed and normalized data, one record is designed as one table. The data items necessary for job processing become table components.



2) View design

A view is a virtual table created by extracting from tables items necessary for job processing for business, to simplify data operations. The characteristics of view creation are as follows:

- Specific columns can be extracted from several tables and edited into a new virtual table.
- This virtual table makes it possible to rearrange columns and to change the order(ascending or descending order)of the row.
- Views make it possible to make specific rows and columns public while hiding unnecessary data.



(4) SQL (Structured Query Language)

SQL is used to handle the contents of a relational database.

This language consists of a Data Definition Language (DDL) and a Data Manipulation Language (DML) and defines access privileges and directive statements. The directive statements are for search, update, addition, deletion, and other operations.

The following shows an example of the SELECT statement:

SELECT (name of items to be displayed) : Delimited with a comma.

FROM (name of tables to be used) : Delimited with a comma.

WHERE (search and join conditions): Specified by AND or OR.

The SELECT statement can be used for the following purposes:

- Aggregation

Records the i-th items of which have the same value are grouped using the GROUP-BY statement.

- Relocation

The display order is specified using the ORDER-BY statement.

- Sub-inquiry

A further search based on search results is called a sub-inquiry.

(5) Object-oriented database

The object-oriented database is a further extension of the relational database. The purpose of object orientation is to abstract the actual facts and reflect only the abstraction on the computer.

An object-oriented database can have the following component types:

- 1) Compound object

- A compound object has other objects, such as a list or class instance.

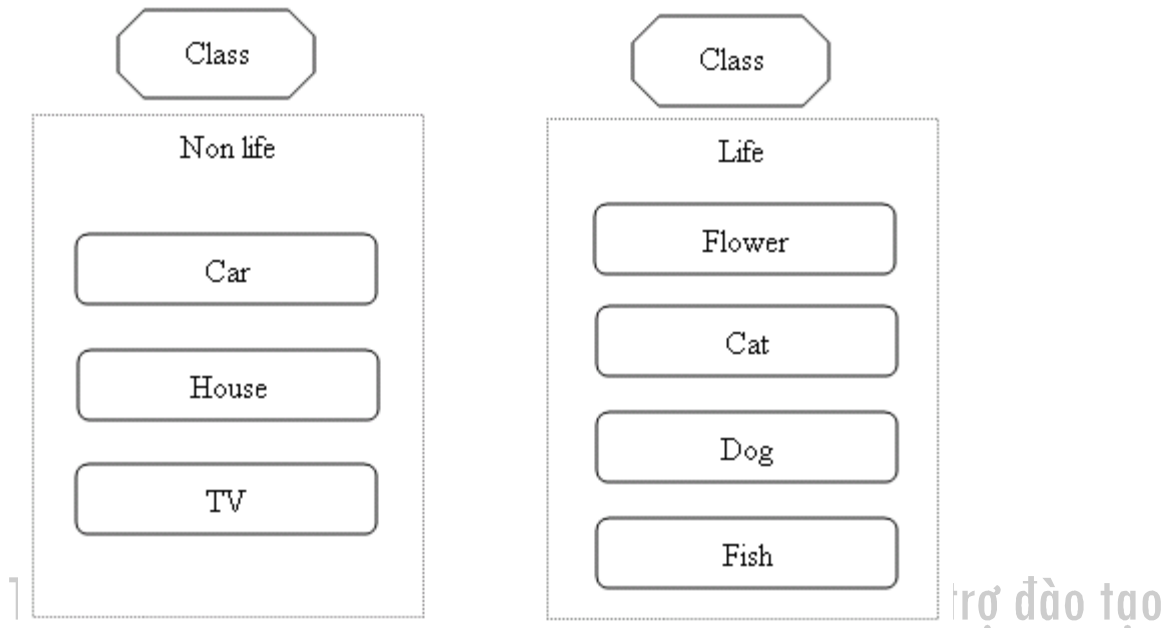
- 2) Object identification

- Even if attributes change over time, the identifier remains unchanged. This object characteristic allows an individual identification that does not depend on attribute values.

- 3) Class (Type)

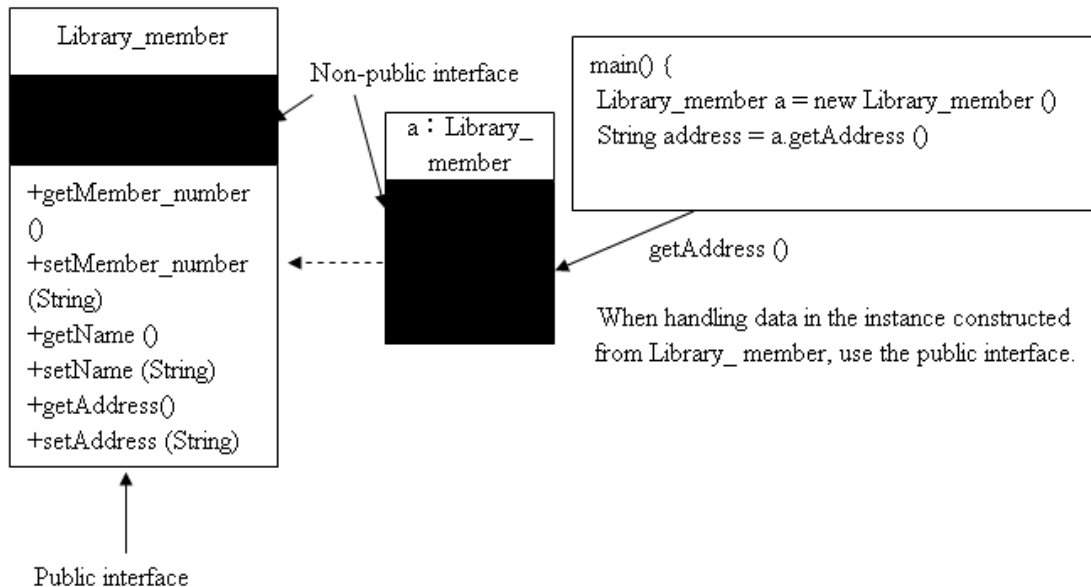
- A class is a common attribute. Grouping objects of the same data structure (attribute) and behavior (operation) is called classing.

A class may also be called a type.



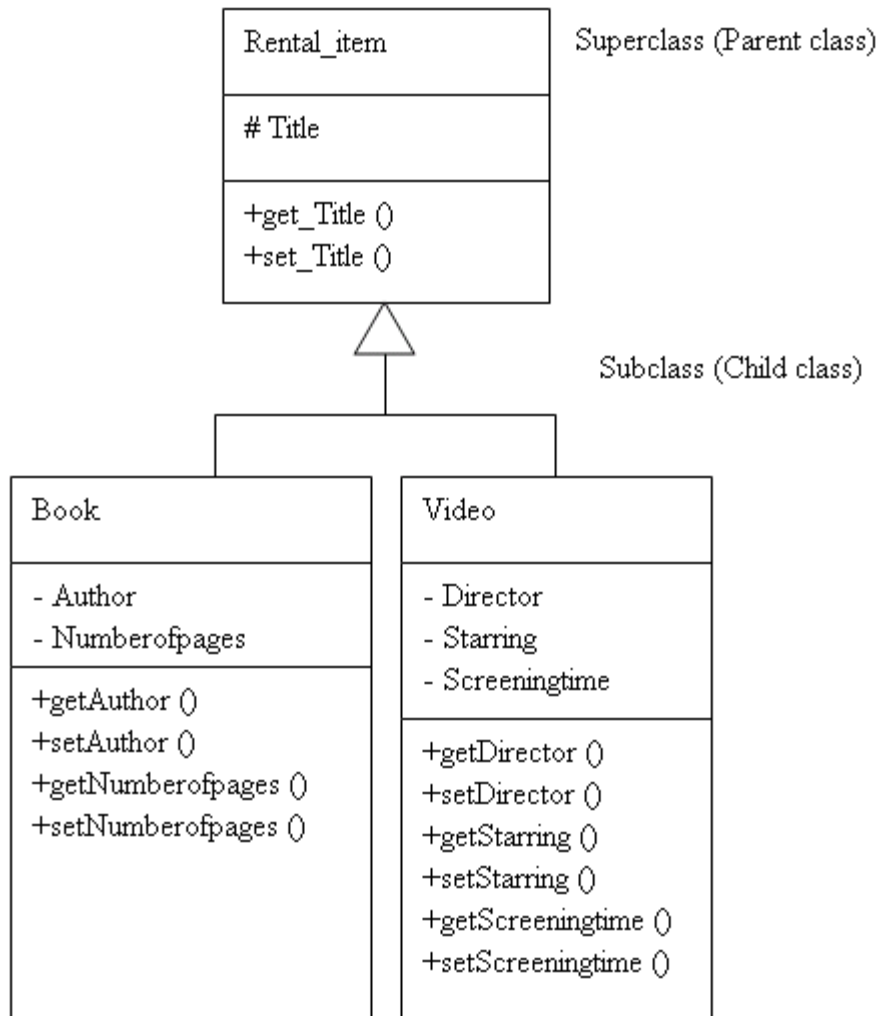
4) Encapsulation

- Encapsulation (also called information hiding) refers to separating the external characteristics of an object from the internal details of the object. All information about an object-oriented system is accumulated in an object and handled only by object operations. Behavior and information are encapsulated in an object. The diagram below illustrates how the concept of information hiding is realized:



5) Inheritance

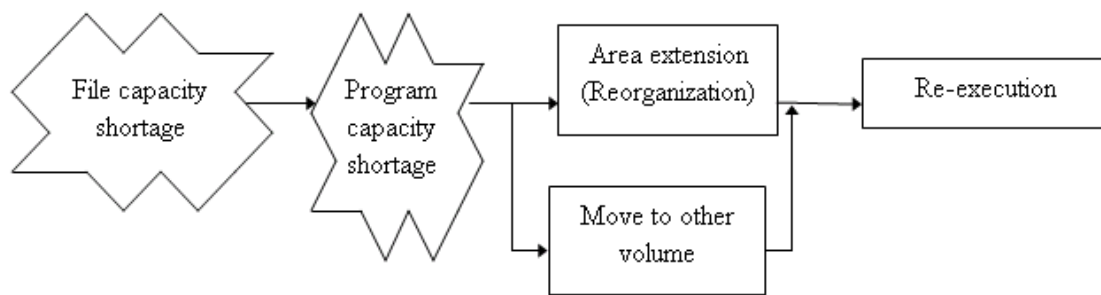
Inheritance means that subclasses inherit attributes and methods from a superclass.



4.2.2 Capacity calculation

(1) File capacity estimate

A file capacity shortage during system operation can result in various problems. One especially serious problem is that an online program in a client-server system may not be able to restart processing until the area is extended. Therefore, an appropriate estimate for the required storage capacity should be made during physical data design.



The number of records is the basis of estimating the storage capacity and should be estimated considering file use. The following table explains how to estimate the number of records for a transaction file, a master file, and an accumulation file.

<Estimating the number of records>

	Transaction file	Master file	Accumulation file
Update frequency.	Updated every day (However, there is a difference between processing peak and other days)	Not updated or added frequently. Fixed, or may be updated or added together with transaction data. (Bank ledger file and other)	Updated or added in proportion to daily transaction data
Data accumulation period	Mainly in days	Until the record is deleted (Including a certain margin of safety, if the record is not deleted physically but left with a deletion flag)	May be stored for several years as transaction data accumulated until reorganization
Required capacity	With some margin for peak days	Number of records to be registered when the master is used (If deleted records are included, may be several times the number of records in use.)	Sufficient capacity to store an average volume of data for the storage period In units of months or years if the storage period is several years.

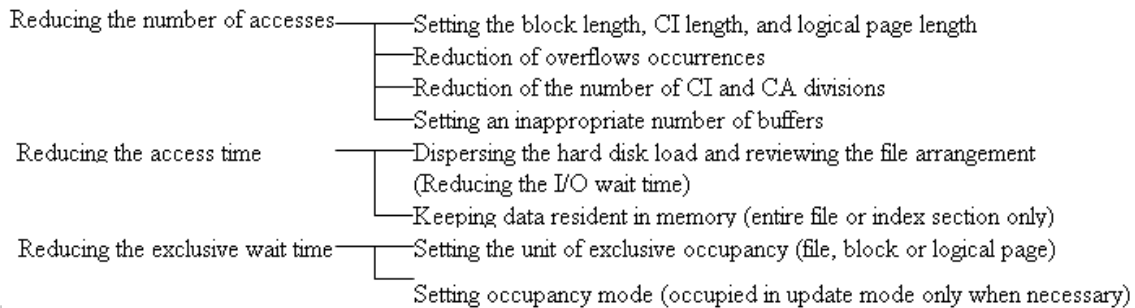
- If a storage area is reserved for centralized file accumulation, storage capacity will be wasted. Files used regularly for job processing should be stored on the hard disk, and other files should be saved onto magnetic disks or other storage media.
- If a significant increase in the volume of job processing is anticipated, a certain increment or margin rate (safety factor) should be applied.
- A network database is composed of several files and requires that consideration be given to parent-child record relationships (for transaction-, master-, and accumulation-type data) as well a consideration of the ratio between parent and child records. Therefore, the basic design information must be more precise than that for general files. The composition of the individual records for each case is more important than for general files.

4.2.3 Performance improvement and optimization

(1) Access efficiency

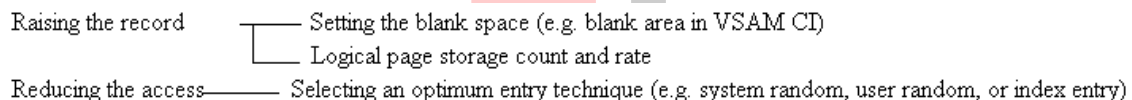
For business data processing or similar process of including frequent file I/O processing, the time for I/O processing accounts for a large percentage of the total processing time, because the I/O processing speed is much lower than the CPU processing speed. Especially in online processing, the I/O time greatly affects the response time.

For physical data design, note the following to improve the access efficiency:



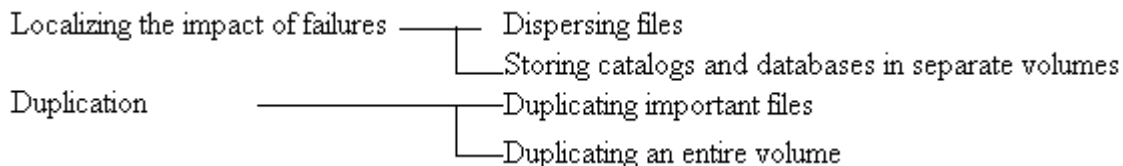
(2) Space efficiency

In a reserved disk area, records should be stored efficiently to minimize empty space (unused area). The efficiency of using a reserved area is called space efficiency. From the viewpoint of disk capacity and reorganization, the space efficiency is especially important for a system handling large files, because the hard disks account for a significant percentage in the total hardware expenses. The following items should be considered to improve space efficiency.



(3) Reliability

Measures against failures must be examined to protect important data from destruction in case of a malfunction. In terms of operation, files should generally be backed up periodically for recovery in case of a malfunction. The following measures against file failures should also be examined to improve reliability and facilitate failure recovery.



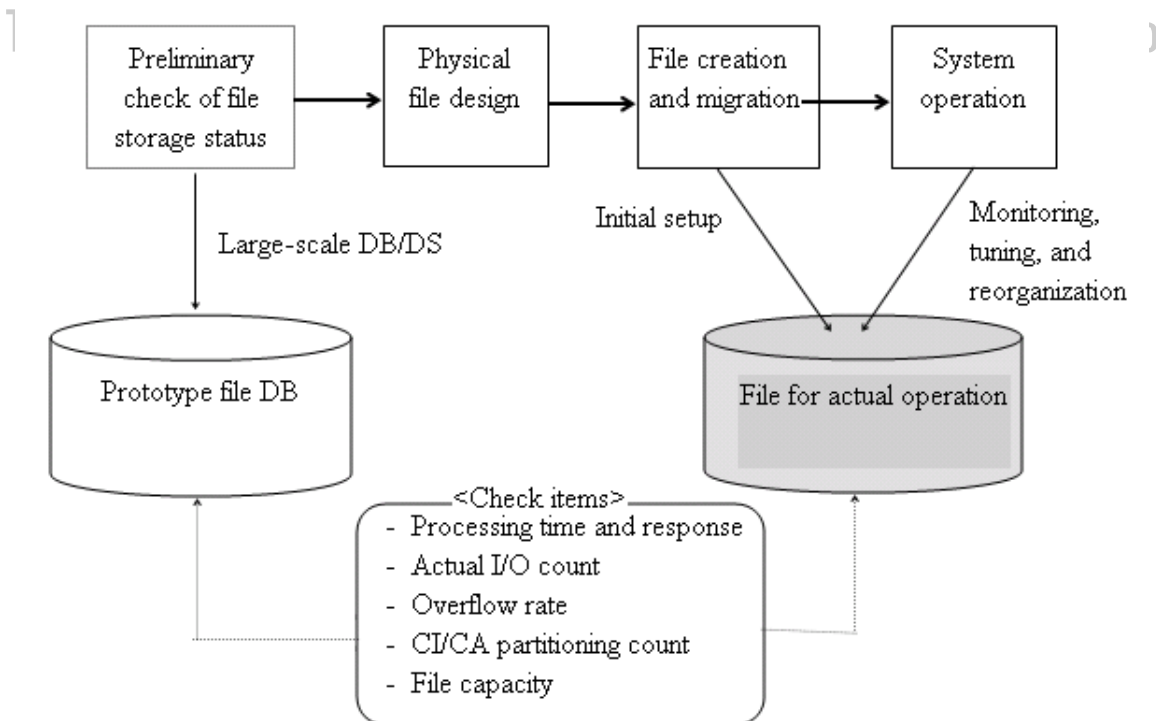
(4) Tuning of physical data design

To optimize the access efficiency and space efficiency explained, the distribution of storage record creation (key dispersion), number of records, the record creation/deletion cycle, and other basic file information listed should be obtained accurately. For example, if information about data dispersion is not considered sufficiently during design, overflows may occur. In case of VSAM, this would result in CI/CA partitioning and lower the processing efficiency. However, it is very difficult to obtain such basic file information accurately. Especially for a new file, the corresponding values must be estimated based on limited information, such as the number of payment slips and the number of forms.

Basic file information

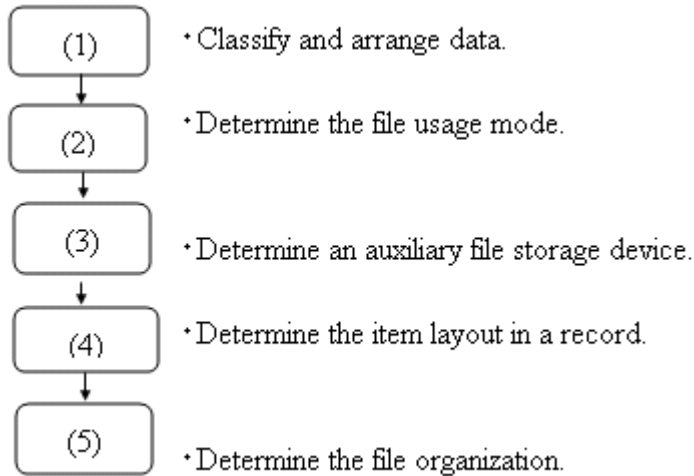
- Number of records (average and peak)
- Record creation/deletion cycle (frequency of addition and deletion)
- Parent-child record ratio and dispersion in a hierarchical structure
- Key distribution (even, uneven, or unknown)

Therefore, physical data design must be reviewed during testing process and at the operation stage for improvement (tuning). The record storage status does not remain unchanged after design, but changes during system operation. This may cause significant discrepancies with the estimation that was made during design. At the operation stage, the processing efficiency and the hard disk access load and capacity must be monitored periodically to determine the necessity of tuning. For a system whose file design is important (severe performance requirements and large files), it is necessary to create a pilot database (prototype database) or file from actual data in advance to verify the access performance and capacity.



Exercises

1. The following chart illustrates physical data design. Fill in the boxes by selecting from the answers below.



Answers

- a. Determination of logical organization
- b. Record layout design
- c. Selection of storage media
- d. Determination of physical organization
- e. Analysis of data characteristics

2. The following statements are about record types. Select the type of record each statement is about from the answers below.

- (1) This is the most widely used record type with a fixed record length.
- (2) The record length is not fixed, but determined by the program.
- (3) The record length is not fixed, but recorded at the beginning of the file.

Answers

- a. Variable-length record
- b. Fixed-length record
- c. undefined-length record

3. Database languages can be classified into two types. One type defines the database structure and consistency constraints, while the other handles the actual database contents. Select these two language types from the answers below.

Answers

- a. NDL
- b. SQL
- c. 4GL
- d. DDL
- e. QBE
- f. DML
- g. ISO

4. Select the correct sentence from the following statements, which are about relational database operations.

- a. Projection means to fetch a table from rows matching specific conditions.
- b. Selection means to fetch specific columns from a table.
- c. Join means to fetch rows matching specific conditions from two or more tables and creating a derivation table.
- d. Insertion means to insert a specific column into a table.

5. In this problem about relational database operations, select a combination of necessary operations from the answers below to obtain Table Y from Table X.

Table X

Name	Age	Section	Birthplace
Okamoto	22	4	Kumamoto
Yamada	31	2	Tokyo
Tanoue	26	1	Hokkaido
Suzuki	45	3	Tochigi
Sato	29	1	Tokyo
Yoshida	38	2	Tokyo

(Data manipulation) Table Y

Name	Age	Birthplace
Yamada	31	Tokyo
Sato	29	Tokyo
Yoshida	38	Tokyo

Answers

- a. Projection and selection
- b. Selection and join
- c. Sum and selection
- d. Projection and join

6. Select the correct sentence from these statements about relational databases.
- A parent-child record relationship is expressed by a data structure using a link.
 - Records are linked to each other not by pointers, but by item values in each record.
 - Each kind of data items in a record has a key that is stored as an index.
 - A parent-child record relationship is expressed by a data structure using a pointer.

7. In the following text about the characteristics of a logical data model, fill in the blanks with appropriate phrases selected from the table below.

A hierarchical database model and a network database model implement a relationship between entities by a combination of parent-child relationships. One child has (1) in a hierarchical database model and (2) in a network database model. Meanwhile, a relational database is based on the concept of mathematical sets. Rows in different tables are related to each other by (3).

	(1)	(2)	(3)
a.	Multiple parents	Always one parent	Value matching
b.	Always one parent	Several parents	Value matching
c.	Several parents	Several parents	Link
d.	Several parents	Always two parents	Pointer
e.	Always two parents	Several parents	Pointer

8. Select the correct sentence from these statements about the purpose of normalization.
- Dependencies between attributes are viewed as functional dependencies, and the dependencies are reduced to minimize the physical number of I/O operations.
 - Data duplication is eliminated from various data models to improve database access efficiency.
 - By normalization up to the third level, everybody will obtain the same table of relationships. This eliminates merely subjective decisions during database design.
 - Data duplication is eliminated to duplicate updates or inconsistencies during database operations.

<http://www.vitec.org.vn>

9. Select the most appropriate explanation about the procedure for the first normalization of data.
- A data item that is not a key, but which can be a key is found out, and that data item and those which depend on it are then separated.
 - Key items are separated from other items to separate the data items themselves and their relationships.
 - A repeated collection of data items is separated to form an independent collection of data items.
 - Data items are classified into completely and partially dependent items.
 - Dependencies between data items are analyzed in accordance with the purpose of the job, and data item duplication is minimized.
10. In this explanation of the normalization of a relational database, select a sequence from the answers below that puts the following statements into the correct order.
- Attributes dependent on non-key attributes are separated as another record.
 - Attributes dependent on some keys are separated as another record.
 - Repetitive data is separated as an another record.

Answers

- (a)-(b)-(c)
- (a)-(c)-(b)
- (b)-(c)-(a)
- (c)-(a)-(b)
- (b)-(a)-c)
- (c)-(b)-(a)

11. How far is the following table normalized? Select the correct answer.

Employee No.	Name	Hire year	Position	Position allowance
1234	Taro Tanaka	1967	Department manager	12,000
1235	Ichiro Suzuki	1975	Section chief	7,000
1236	Jiro Yamada	1981	Section chief	7,000

Answers

- First normalization
- Second normalization
- Third normalization
- Fourth normalization

12. The following descriptions compare the characteristics of a relational database and an object-oriented database. Select the correct description.
- a. Data and procedures are encapsulated and treated as one thing, and therefore objects which are complex in structures and contains behaviors can be handled.
 - b. Data manipulations and relationships are mathematically defined to keep independency between a programming language and a data operation language.
 - c. Because relationships are defined as logical data structures data manipulations can be performed in a non-procedural manipulation language.
 - d. A class hierarchy modeled after information on the real world can be implemented to handle data and operations separately.
13. Select the correct description from the following statements about the access efficiency for relational databases.
- a. Efficiency is improved if all read access paths accesses always are done through access paths using an index.
 - b. An increase in data volume will not lower access efficiency, if an optimum access path is prepared in advance.
 - c. Use of an exclusive control function prevents simultaneous processing of numerous transactions from affecting efficiency.
 - d. For frequently updated columns, indexes should be minimized to improve access efficiency.
14. Select the correct description for a master file from the following statements related to the analysis of data characteristics.
- a. Temporary file storing collectively the data generated at all time point
 - b. File containing long-term transaction data organized by purposes
 - c. File containing control information for control of program execution and improvement of operation efficiency
 - d. File storing basic information permanently or semi-permanently for specific processes
15. Select from the answers below two examples of using a master file.
- Answers
- a. Sales journal
 - b. Merchandise master
 - c. Bill accumulation file
 - d. Parameter file
 - e. History log file
 - f. Customer master
 - g. Source library

16. In the following statement about logical file design, fill in the blanks with answers selected from below.

For (1) file design, it is not necessary to consider (2) processing and (3) characteristics. Priority should be given to the (1) of (4) processing. Considering (2) processing and (3) characteristics for improving the (5) of design should be performed during (3) data design.

Answers

- a. merchandise
- b. job
- c. data
- d. physical
- e. computer
- f. quality
- g. business
- h. information
- i. logical

17. The following statements explain the characteristics of file organizations. Select the file organization corresponding to each description from the answers below.

- (1) Records are located and accessed based on calculation.
- (2) Records on storage media are accessed in their physical order. This file organization does not permit direct access.
- (3) This file organization contains an index section and a data section, and a record is accessed using a key.
- (4) The sequential, direct, and indexed sequential file organizations are integrated for sequential and direct accesses.

Answers

- a. VSAM file
- b. Sequential file
- c. Indexed sequential file
- d. Direct file

<http://www.vitec.org.vn>

18. The table lists the types of storage media. Fill in the blanks by selecting from the answers below.

(1)	Punched card	
	Magnetic card	
	Barcode tag	
Tape type	Paper tape	
	Magnetic tape	(2)
		Cartridge
(4)	Open reel	
	Magnetic disk	(3)
		Floppy disk
	(5) disk	
	Optical disk	CD-ROM

Answers

- a. Cassette
- b. Hard disk
- c. Card type
- d. Magneto-optical
- e. Disk type

19. From the following statements about data formats, select the statement that refers to the fixed point format.

- a. This format is used for input-output and facilitates checking using a dump listing.
- b. Since it allows a wide range of data to be handled with high precision, this format is used for technical calculations.
- c. Large numeric values can be expressed in fewer bytes, which saves storage space.
- d. This format, used for internal processing, facilitates checking using a dump listing.

<http://www.vitec.org.vn>

5 Creation and Reuse of Parts

Chapter Objectives

Concerning parts and their reuse, this chapter explains the importance of and necessity for standardization and creation of parts.

5.1 Creating and Reusing Parts

5.2 Using Software Packages

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

5.1 Creating and Reusing Parts

5.1.1 Concept of creating and reusing parts

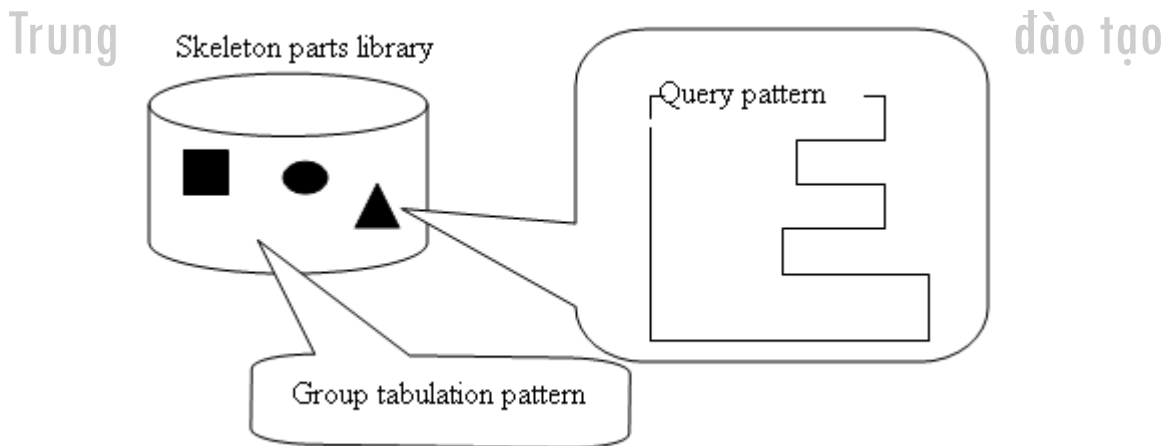
In the system development, multiple programs are created for a variety of purposes. However, some of these programs perform very similar tasks; developing multiple programs for performing the same tasks would mean an unnecessary increase in the size of development of a new system. Extracting portions that are shared by multiple programs and reusing them as parts avoids this problem, because this approach reduces the duplicate works. Such parts can be classified into the following two major categories:

- Skeleton parts

Processes performed by programs are classified into basic patterns.

The main process in each pattern is made a skeleton part. This aims at the standardization of the rough processing logic in a program. Report printings, group summation, query processing, etc. are typical examples.

- Standardize basic processing patterns
- Coding is only for adding specialized processing

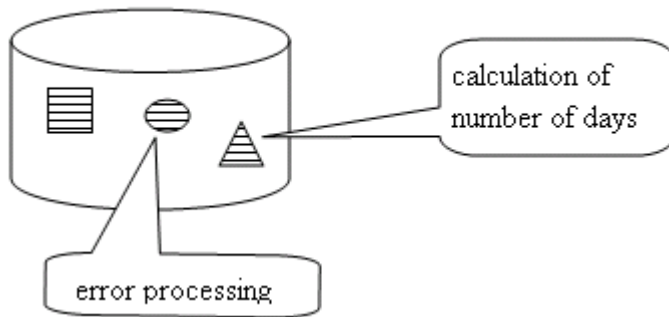


- Functional parts

Parts that are to perform a set of processing functions in programs. Their examples are code checking, code conversion, interest calculation, tax calculation, calculation of number of days, error processing, and message processing, etc.

- Uses a single function as a black box for processing
- Coding and testing are almost unnecessary

Functional parts library



The following advantages are obtained by program parts creation:

- Productivity is improved by avoiding duplicate in the development process.
- The number of test runs can be reduced by using parts which have actually run.
- Maintainability is improved as a result of localizing functions.
- Creation of program patterns promotes standardization.
- Using existing applications reduces the new development work.

(1) Kinds of parts and their characteristics

1) Type parts and form parts

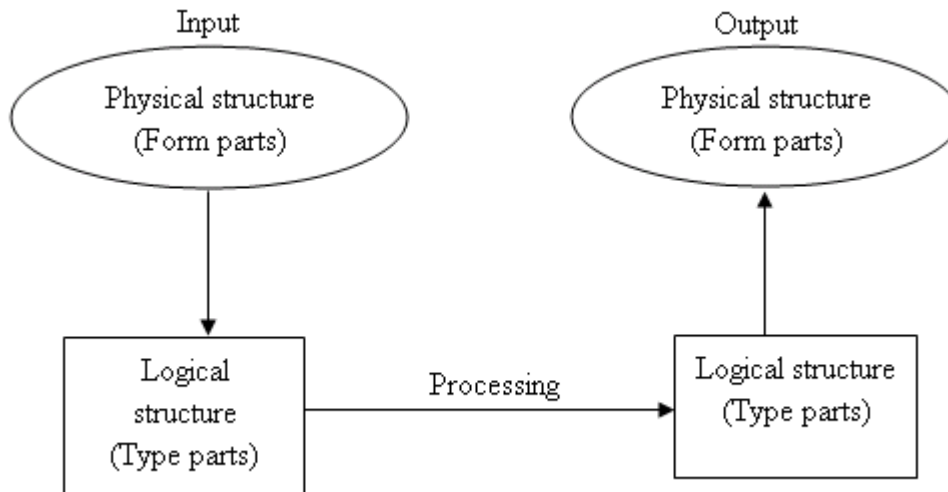
a. Type parts

Type parts are parts whose essential meanings and function are defined independent of the format and appearance. Examples of type parts are sets of functions to operate on these character string type, record type, table type, array type, and tree type.

b. Form parts

Form parts are parts for which the physical format and appearance of information is pre-defined.

An example for form parts would be sets of functions for representing information in the form of commands, as display on video data terminals (VDTs), in reports, business graphs, files, and databases.



[Separation of logical structure and physical structure]

2) Function parts

Function parts convert input data to output data without having the internal states.

Examples would be built-in FORTRAN functions.

3) Data parts and process parts

a. Data parts

Data parts consist of data themselves.

Examples are table information and dictionary information.

b. Process parts

Process parts consist only of code for processing.

Examples would be built-in FORTRAN functions.

(Function parts may be regarded as process parts depending on the classification.)

4) Object parts

Each object part can be executed as an independent process.

Examples are buttons, pull-down menus, and drop-down list boxes in windows.

5) Resident parts and embedded parts

a. Resident parts

Resident parts exist independently in the system, and can be called dynamically from programs and terminals.

An example would be an application programming interface (API) for calling individual operating system functions from applications.

b. Embedded parts

These parts are statically embedded in programs in advance. (These embedded parts cannot be shared dynamically by multiple programs.)

An example is a common module that is created when a program is divided into modules.

6) Black box parts and white box parts

a. Black box parts

The internal structure of these parts is hidden from users. Users can handle programs in high-level functional units regardless of the contents of the parts.

An example is a date check part that transfers date data for a validity check.

b. White box parts

The internal structure of these parts is disclosed to users. Users can change the parts as they want, but if the parts are changed, they no longer function as standard parts.

An example is a template for a report output program having a basic structure composed of input, editing, and printing section.

7) Parametric parts and non-parametric parts

a. Parametric parts

Parts that have parameters for specification of a parts creation method and operating conditions.

If a part is applied to a wide range of processing, a single type of part having the same function may not be sufficient. In these cases, it may be necessary to prepare a parts family containing multiple kinds of parts.

Parametric parts enable users to select the required parts from a parts family by using parameters.

b. Non-parametric parts

Parts that have no parameters.

8) Closed system parts and open system parts

a. Closed system parts

Parts that are prepared in advance and enable the programmer to completely create the software to be developed by using only these parts (the interfaces of these parts follow

unique specifications).

Closed system parts cannot be used to implement any functions not included in the parts themselves, but using closed system parts enables the programmer to create organized and easy-to-understand systems.

b. Open system parts

Parts that enable the programmer to mix parts with programs written in a generally used programming language (the interfaces of these parts are based on industry standards).

Using open system parts may not help in creating organized and easy-to-understand systems, but maintains compatibility with existing systems.

(2) Points in parts design

1) Collection of various documents and other information

As preparation for designing parts, select the processing units that can be used as parts according to the processing functions to be created. Then, collect the specifications and other information about the selected processing units.

2) Analysis and identification of similar functions

After collection is completed, analyze the collected documents and other information, and then identify similar functions close to one another. In this operation, consider the quality requirements of parts and constraints on parts maintenance in order to determine the scope of part application, such as whether the part is to be used throughout the enterprise, only in a specific department, or by external users as well. Also examine whether each kind of part has a unique interface or all parts share an industry standard interface, whether it is easy to use the part for general-purpose applications and whether it has a high degree of portability.

Assigning codes to the parts facilitates identification and classification, and this type of information can become useful when creating a parts search tool later on.

3) Determining parts specifications

After selecting the units of the parts to be created, specify the parameters and variables required for the parts, and determine the parts specifications.

4) Creating parts

With the same method as for normally creating a program, follow the specifications to create parts, and then test the parts to complete the parts creation process.

5) Documenting method of using parts

After the parts creation process is complete, document functional explanations, parameters, and methods of use for the parts.

5.2 Using Software Packages

If a suitable software package is available, using such a package results in improved quality and productivity during software development.

5.2.1 Subprogram libraries

For reusing programs, it is useful to collect programs in a library that is organized in units of reuse based on a specific standard. In the procedure-oriented languages used in a variety of fields, procedure functions are created as library parts. One can develop a program by combining these functions.

Many subprogram libraries are commercially available, and they can be reused as independent libraries by packaging them together with existing components.

(1) Package selection criteria

1) Range of supported processes

Lower-stream process, GUI construction process, reuse process, and other processes

2) Range of supported

a. Specific application domains (e.g., job processing, scientific and engineering, process control)

b. Specific methodologies and techniques. (e.g., hierarchical structuring, data-oriented approach, object-oriented).

(2) Development platform

Which software packages can be selected depends on the development platform, such as whether the software is to be developed for mainframes or personal computers, workstations, a distributed network environment, for an intranet or extranet.

(3) Embedding tools by third parties

To supplement the required functions, examine whether compatible and embeddable tools are available from third-party vendors and whether the tools can be actually embedded into the system.

(4) Execution environment platform

Selectable software packages vary depending on the platform to be developed, such as whether the software is to be developed for mainframes, personal computers, workstations, a distributed network environment, and for an intranet or extranet.

(5) Packaging technology

1) Industrial standards for componentware implementation

These standards include COM of Microsoft; the Common Object Request Broker Architecture (CORBA) supported by the distributed object standardization organization, OMG; and JavaBeans and Enterprise JavaBeans of Sun Microsystems.

2) Design pattern

Design pattern is a convenient method for reusing object-oriented designs. This method enables systems to easily use designs by cataloging common interactions between useful objects.

3) Object-oriented re-engineering

For example, use of CASE tools, which automatically create source programs, enables efficient re-engineering.

4) Legacy wrapping

Legacy wrapping enables effective use of large-scale software resources used by enterprises over many years without structural changes. It does so by enabling the use of software with the same interface as that of new applications from desktop PCs to the core system on mainframes.

With this method, existing system resources are linked to new applications using distributed object techniques, such as CORBA.

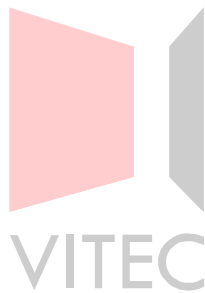
5.2.2 Class libraries (combination with object-oriented languages)

In object-oriented program languages, a group of objects having common attributes and methods (procedures) is called a “class.”

A class library is a collection of related classes. In this method, a part is created by treating a program having specific functions as one “class” in an object-oriented programming language.

Such a “class” can be used as a part of a program. Productivity can therefore be improved by collecting general-purpose classes in a class library in advance.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Exercises

1 Select the statement(s) below that are not correct as representation of advantages of parts creation for programs.

- (1) Productivity is improved by avoiding duplication in the development process.
- (2) The number of test runs can be reduced by using parts which have been actually run.
- (3) Productivity is improved as a result of distributing functions.
- (4) Creation of program patterns makes standardization easier.
- (5) Using existing applications reduces the development work for developing new systems.

2 Select the statement(s) below that describe the characteristics of function parts.

- (1) Parts that can be executed individually as independent processes
- (2) Parts that can be called dynamically from programs and terminals
- (3) Parts whose internal structure is hidden from the user
- (4) Parts that convert input data to output data without having processing states
- (5) Parts that have parameters that specify parts creation methods or operating conditions

3 Explain the differences between type parts and form parts.

4 In the classification of parts in parts design, select the kind of parts each of which is a set of values, such as commonly used constants.

- (1) Process parts
- (2) Object parts
- (3) Data parts
- (4) Parametric parts
- (5) Black box parts

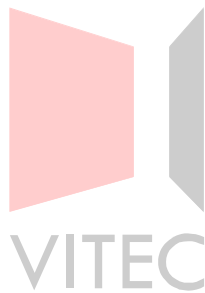
5. Explain the purpose(s) of using software packages.

<http://www.vitec.org.vn>

6 Select the statement(s) below describing the legacy wrapping included in the packaging technology.

- (1) This method enables effective use of large-scale software resources used by enterprises over many years without structural changes. It does so by enabling the use of software with the same interface as that of new applications over the whole range of systems, from desktop PCs up to the core system on mainframes.
- (2) These include COM of Microsoft; the Common Object Request Broker Architecture (CORBA) supported by the distributed object standardization organization, OMG; and JavaBeans and Enterprise JavaBeans of Sun Microsystems.
- (3) This is a convenient method for reusing object-oriented designs.
This method enables systems to easily use designs by cataloging common interactions between useful objects. Use of CASE tools, which automatically create source programs, enables efficient re-engineering.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

6 Creation of Internal Design Documents

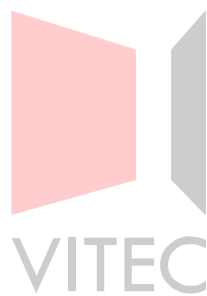
Chapter Objectives

This chapter explains how to create internal design documents and how to conduct a design review.

6.1 Organization of Internal Design Documents

6.2 Design Review

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

6.1 Organization of Internal Design Documents

The internal design documents consist of the following set of design documents:

Diagram for partitioning into components
Interface between components
Component processing specifications
Screen design document
Form design document
File design document
Database design document

When creating internal design documents, consider the following:

Concepts of processes
Parallel processes and synchronization
Exclusive processing and sharing
Communication function between processes
Deadlock
Time control
State transition processing
Interrupt processing

6.1.1 Diagram for partitioning into components

The purpose of creating diagrams for partitioning into components is to further partition subsystems into components.

The following are examples of items required on these diagrams:

(1) Header

- Specification name
- System name
- Subsystem name
- Component name (process name)
- Date of creation
- Author's name

(2) Contents

- Illustrate the components in each subsystem.
- Provide brief descriptions of functions as necessary.

(3) Footer

- Version
- Date of revision record
- Record reviser
- Description of revision

An example of a diagram for partitioning into components is shown next.

< Example of diagram for partitioning into components >

Specification name	System name	Subsystem name	Component name	Date of creation	Author
Diagram for partitioning into components	<Applicable system name>	<Applicable subsystem name>	<Applicable component name>	2001/2/11	Ueda
<div><div>ABCF355Z Creation of MT for bank</div><div><div>ABCB2000 Transfer DB extraction</div><div>Transfer DB extraction F SORT</div><div>ABCB2010 Creation of MTF for bank</div><div>File partition</div></div></div>					
Version	Date	Reviser	Description of revision		
-----	-----	-----	-----		
-----	-----	-----	-----		
-----	-----	-----	-----		

6.1.2 Diagram for interface between components

The purpose of designing diagrams showing interfaces between components is to define the relationships among partitioned components.

The following are examples of items required on these diagrams:

(1) Header

- Specification name
- System name
- Subsystem name
- Component name (process name)
- Date of creation
- Author's name

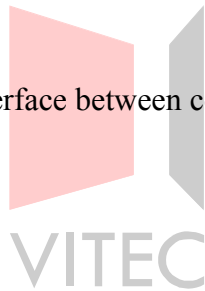
(2) Contents

- Define data input to and output from components precisely.
- Explain the flow of data precisely, such as where data is input from and to which component or subsystem the data is transferred.

(3) Footer

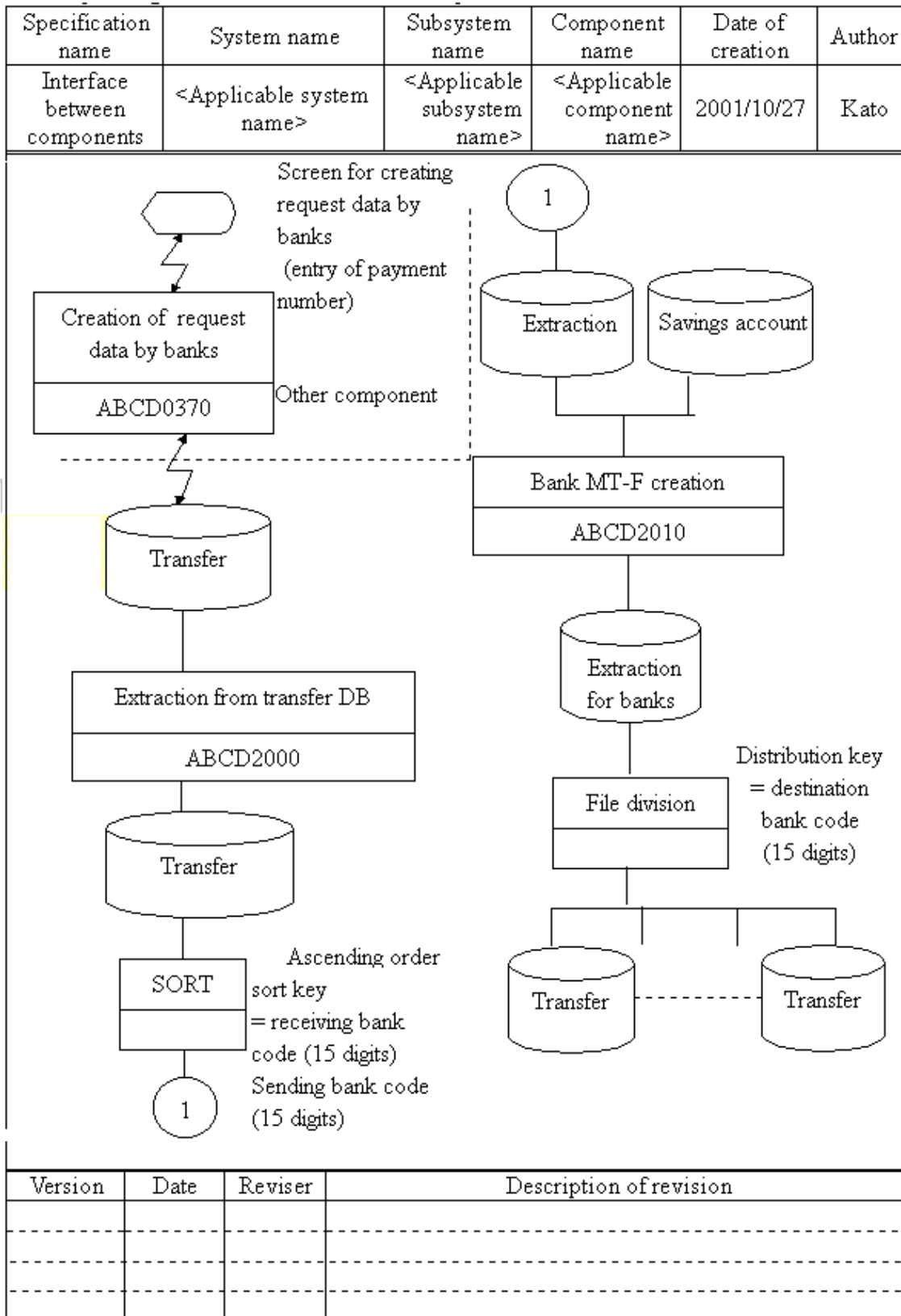
- Version
- Date of revision
- Record reviser
- Description of revision

An example of a diagram of the interface between components is shown next.



<http://www.vitec.org.vn>

< Example diagram of interface between components >



6.1.3 Component processing specifications

The purpose of creating component processing specifications is to define the functions of partitioned components.

The following are examples of items required in these specifications:

(1) Header

- Specification name
- System name
- Subsystem name
- Component name (process name)
- Date of creation
- Author's name

(2) Contents

1) Input

- Write the name of the input source and the name of input data.

2) Function

- Write the processing names of every component.
- Outline the processing of each component.

3) Output

- Write the name of the output destination (output destination component) and the name of output data.

4) Master file

- Describe the files and databases used in such a way that their purposes, such as input, output, and input/output, are clearly indicated.

(3) Footer

- Version
- Date of revision
- Record reviser
- Description of revision

<http://www.vitec.org.vn>

An example of component processing specifications is shown next.

< Example of component processing specifications >

Specification name	System name	Subsystem name	Component name	Date of creation	Author
Component processing specifications	<Applicable system name>	<Applicable subsystem name>	<Applicable component name>	2001/6/4	Okano
General flowchart					
<pre>graph TD AE{{Acceptance entry}} --> OALC[Order acceptance input transaction with LC] subgraph Reference_file [Reference file] CIF[(Credit information file)] CUF[(Customer information file)] D1[-] D2[-] D3[-] end CUF --> OALC OALC --> HLF[(Handling ledger file)] OALC --> LIC[(LC information file)] OALC --> TIF[(Transaction information file)] OALC --> S[Statement] OALC --> OAT[Order acceptance table] OALC --> U[] style U fill:none,stroke:none</pre>					
Description of tasks			Dept. or branch	Input terminal	
For negotiation requests based on LCs issued by banks, the component performs processing for acceptance, checks the validity of handling, and determines handling conditions 1. Automatic numbering of acceptance number (REF No.) 2. Automatic recording in handling ledger			Sales office	UBT	
			Remarks		
Version	Date	Reviser	Description of revision		

6.1.4 Screen design document

The purpose of the screen design document is to clarify the functions of the graphical user interface (GUI). This design requires special consideration about the human interface, because this is the part that system users interact with.

The following are examples of items required in the document:

(1) Header

- Specification name
- System name
- Subsystem name
- Component name (process name)
- Screen name
- Date of creation
- Author's name

(2) Screen layout

1) Title, menu, and toolbar areas

- Specify the screen name, screen number, date, and time, etc.

2) Input-output areas

- Decide the standard formats for such items as display locations and display formats, and write them.
- For the input screen, use the same kind of picture as for the input form.
- For the output screen, indicate only items absolutely necessary.
- Arrange the screen layout so that the flow of operation moves from top to bottom and from left to right.
- Write which corrective action is to be taken in the event that the screen freezes.

3) Message area

- Write the maximum number of characters in messages and the display formats.

4) Guidance area

- Write the input mode and communication status.

(3) Footer

- Version
- Date of revision
- Record reviser
- Description of revision

An example of the screen design document is shown next.

< Example of screen design document >

Specification name	System name	Subsystem name	Component name	Date of creation	Author
Screen design document	<Applicable system name>	<Applicable subsystem name>	<Applicable component name>	2001/9/30	Kimoto

Screen format

Title, menu, and toolbar areas

Date _____
Form No. _____

Input-output area

Display Graph

Print List

Return

Details (body)

Message area

Guidance area

Control	Operation	Function
Display Graph button	Click	Display the Graph Display screen
Print List button	Click	Opens the Print Preview screen.
Return button	Click	Opens the Search Condition Specifications screen

Version	Date	Reviser	Description of revision

6.1.5 Form design document

Forms are classified into two types: input forms for the case that data is generated during operations, and output forms that the system uses for printing out reports and as materials for confirmation.

Design input forms so that users can easily enter data and are not likely to enter invalid data by mistake. When designing output forms, consider whether they are easy to read, easy to understand, and clear enough to convey their intended meanings.

The following are examples of items required on these forms:

(1) Header

- Specification name
- System name
- Subsystem name
- Component name (process name)
- Form name
- Date of creation
- Author's name

(2) Form layout

1) Title, date of creation, and page count

- Standardize the display locations and formats as much as possible.

2) Header

- Specify the position of header item names with corresponding data contents so that they can most easily be read.

3) Details

- Clearly differentiate the lines for totals and the lines for details.
- If multiple totals such as subtotals, intermediate totals, and grand totals are to be printed, assign titles, so that their hierarchical levels can be easily recognized.

4) Page feed and line feed conditions

- Write down the page feed and line feed conditions individually.

(3) Footer

- Version
- Date of revision
- Record reviser
- Contents of revision

An example of the form design document is shown next.

6.1.6 File design document

The purpose of the file design document is to define the characteristics of data stored in files and as data items.

The following are examples of items required in this document

(1) Header

- Specification name
- System name
- File name
- Record name
- Date of creation
- Author's name

(2) File layout

1) File organization

- Write the type of file organization used, such as sequential, indexed, direct, partitioned, or VSAM organization.

2) Storage media

- Write the type of media used, such as hard disks, magnetic tapes, and floppy disks.

3) File information

- Write file names, record names, record lengths, block lengths, file sizes, and distinctions among fixed-length, variable-length, and undefined-length records.

4) Update mode

- Write whether records have been added or deleted.

5) Processing cycle

- Write daily processing, monthly processing, annual processing, or non-periodical processing.

6) Processing mode

- Write master processing, transaction processing, or historical processing.

7) Record layout

- Write group item names, elementary item names, item lengths, item attributes, iteration counts, and changed definitions.

8) Index keys

- Write down primary keys and secondary keys precisely.

(3) Footer

- Version
- Date of revision
- Record reviser
- Contents of revision

An example of the file design document is shown below.

< Example of file design document >

Specification name	System name	Subsystem name	Component name	Date of creation	Author								
File design document	<Applicable system name>	<Applicable subsystem name>	<Applicable record name>	20/9/00	Nishi								
Medium	DASD		Processing cycle	Daily processing									
File organization	ISAM		Processing mode	Transaction processing									
Record format	Fixed length		Prime key	FORM-CD									
Record length	70 bytes		Secondary key	PROD-CD									
Update mode	Add, change, delete												
Item names													
<table border="1"> <tr> <td>Form code</td><td>Status code</td><td>Customer code</td><td>Commodity code</td><td>Quantity</td><td>Unit price</td><td>Total amount</td><td>Reserved</td></tr> </table>						Form code	Status code	Customer code	Commodity code	Quantity	Unit price	Total amount	Reserved
Form code	Status code	Customer code	Commodity code	Quantity	Unit price	Total amount	Reserved						
Level	Item name	Coding name	PICTURE	Description									
01	Sales record	SALESAMT											
02	Form code	FORM-CD	9(5)	Serial number of a form that is always used for accounting									
02	Status code	STATUS	9	00: Sold 01: Must be returned									
02	Customer code	CUST-CD	9(5)										
02	Commodity code	PROD-CD	9(5)										
02	Quantity	QTY	9(3)	A positive value is entered in this field for returned goods.									
02	Unit price	UPRICE	9(6)	This field is blank for returned goods.									
02	Total amount	AMT	S9(9)	A negative value is entered in this field for returned goods.									
02	Reserved	FILLER	X(36)										
Edition	Date	Reviser	Description of revision										

6.1.7 Database design document

The layout of the database design document is designed and created according to the physical organization of a database based on normalized record specifications. Examine whether file design document explained in the previous section can be used for the database design document.

The following are examples of items required in such documents;

(1) Header

- Specification name
- System name
- Schema name
- File name
- Record name
- Date of creation
- Author's name

(2) File layout

1) Database format

- Write the database formats used, such as relational database, hierarchical database, or network database.

2) File information

- Write record lengths, logical page lengths, and database capacity.
- Write the parent and child relationships for records in a hierarchical database and network database.

3) Update mode

- Write an indication of whether records are added, changed, or deleted.

4) Processing cycle

- Write daily processing, monthly processing, annual processing, or non-periodical processing.

5) Processing mode

- Write master processing, transaction processing, or historical processing.

6) Record layout

- Write elementary item names, item lengths, and item attributes.

7) Index keys

- Write prime keys and secondary keys precisely.

(3) Footer

- Version
- Date of revision
- Record reviser
- Contents of revision

An example of the database design document is shown next.

< Example of database design document >

Specification name	System name	Subsystem name	Component name	Date of creation	Author																																																																																										
Database design document	<Applicable system name>	<Applicable schema name>	<Applicable record name>	20/9/00	Yosizumi																																																																																										
Format	Network database		Processing cycle	Daily processing																																																																																											
Medium	DASD		Processing mode	Transaction processing																																																																																											
Record length	70 bytes		Prime key	FORM-CD																																																																																											
Logical page length	4096 bytes		Secondary key	PROD-CD																																																																																											
Update mode	Add, change, delete																																																																																														
Item names																																																																																															
<table><tr><td>Form code</td><td>Status code</td><td>Customer code</td><td>Commodity code</td><td>Quantity</td><td>Unit price</td><td>Total amount</td><td>Reserved</td></tr></table>						Form code	Status code	Customer code	Commodity code	Quantity	Unit price	Total amount	Reserved																																																																																		
Form code	Status code	Customer code	Commodity code	Quantity	Unit price	Total amount	Reserved																																																																																								
<table><tr><td>Level</td><td>Item name</td><td>Coding name</td><td>Picture</td><td colspan="2">Description</td></tr><tr><td>01</td><td>Sales record</td><td>SALESAMT</td><td></td><td colspan="2"></td></tr><tr><td>02</td><td>Form code</td><td>FORM-CD</td><td>9(5)</td><td colspan="2">Serial number of a form that is always used for accounting</td></tr><tr><td>02</td><td>Status code</td><td>STATUS</td><td>9</td><td colspan="2">00: Sold 01: Must be returned</td></tr><tr><td>02</td><td>Customer code</td><td>CUST-CD</td><td>9(5)</td><td colspan="2"></td></tr><tr><td>02</td><td>Commodity code</td><td>PROD-CD</td><td>9(5)</td><td colspan="2"></td></tr><tr><td>02</td><td>Quantity</td><td>QTY</td><td>9(3)</td><td colspan="2">A positive value is entered in this field for returned goods.</td></tr><tr><td>02</td><td>Unit price</td><td>UPRICE</td><td>9(6)</td><td colspan="2">This field is blank for returned goods.</td></tr><tr><td>02</td><td>Total amount</td><td>AMT</td><td>S9(9)</td><td colspan="2">A negative value is entered in this field for returned goods.</td></tr><tr><td>02</td><td>Reserved</td><td>FILLER</td><td>X(36)</td><td colspan="2"></td></tr><tr><td></td><td></td><td></td><td></td><td colspan="2"></td></tr><tr><td>Version</td><td>Date</td><td>Reviser</td><td colspan="3">Description of revision</td></tr><tr><td></td><td></td><td></td><td colspan="3"></td></tr><tr><td></td><td></td><td></td><td colspan="3"></td></tr><tr><td></td><td></td><td></td><td colspan="3"></td></tr></table>						Level	Item name	Coding name	Picture	Description		01	Sales record	SALESAMT				02	Form code	FORM-CD	9(5)	Serial number of a form that is always used for accounting		02	Status code	STATUS	9	00: Sold 01: Must be returned		02	Customer code	CUST-CD	9(5)			02	Commodity code	PROD-CD	9(5)			02	Quantity	QTY	9(3)	A positive value is entered in this field for returned goods.		02	Unit price	UPRICE	9(6)	This field is blank for returned goods.		02	Total amount	AMT	S9(9)	A negative value is entered in this field for returned goods.		02	Reserved	FILLER	X(36)									Version	Date	Reviser	Description of revision																				
Level	Item name	Coding name	Picture	Description																																																																																											
01	Sales record	SALESAMT																																																																																													
02	Form code	FORM-CD	9(5)	Serial number of a form that is always used for accounting																																																																																											
02	Status code	STATUS	9	00: Sold 01: Must be returned																																																																																											
02	Customer code	CUST-CD	9(5)																																																																																												
02	Commodity code	PROD-CD	9(5)																																																																																												
02	Quantity	QTY	9(3)	A positive value is entered in this field for returned goods.																																																																																											
02	Unit price	UPRICE	9(6)	This field is blank for returned goods.																																																																																											
02	Total amount	AMT	S9(9)	A negative value is entered in this field for returned goods.																																																																																											
02	Reserved	FILLER	X(36)																																																																																												
Version	Date	Reviser	Description of revision																																																																																												

6.2 Design Review

Design reviews are performed mostly to check whether functions are properly implemented and whether the partitioning into separate functions is appropriate.

6.2.1 Participants

(1) User review

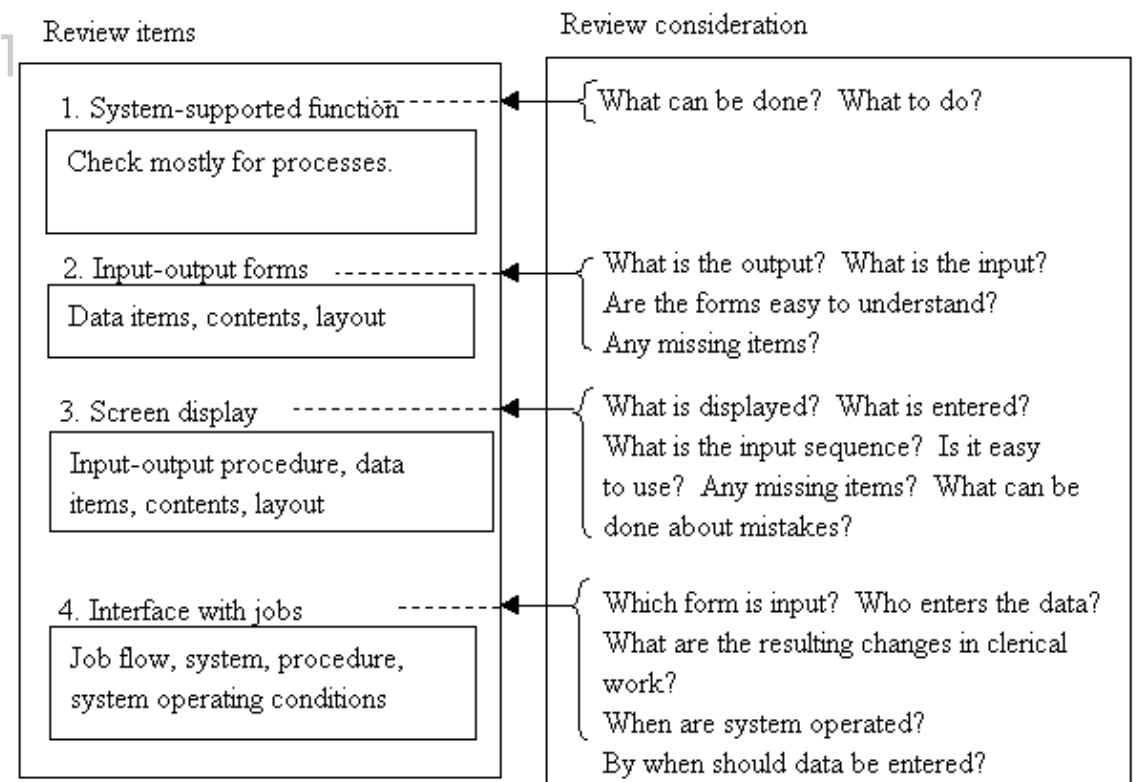
In the user review, all persons with skills and expertise related to systematization gather together to evaluate the contents of design specifications. User reviews are conducted for the following purposes:

- 1) Confirm that all user requirements are included.
- 2) Check whether the system can be used easily.

For input and output forms and screen displays, check for missing items, ease of understanding, and ease of use. Check for missing functions in components, too. Clarify how the job changes by the system.

- 3) Finalize the specifications.

After users' approval is obtained, finalize the design specifications that are appropriate for the time being. If the specifications are not finalized for a long time, development cannot begin and confusion will occur afterwards. Although specification changes are not unusual in system development, specifications must be determined to a certain extent to allow a smooth design process.



In the review by users, the system is checked in units of components. The technical environment and files used for system development are of no direct concerns to users. They therefore need not be explained to users, nor do they have to be checked unless users raise questions. It is better to spend more time to check whether regulatory and organizational measures for system operation are adequate, whether the system is easy to use, whether the system is easy to understand, and whether there are missing items in the listed input-output requirements.

The scope of checks to be performed by all of the members together should be limited to checks of the outline for the entire system. To carry out detailed checks of individual subsystem components, divide members into groups of related departments. If the checks indicate that changes are required, create a plan for the changes after the check, and check the system again.

After the checks are completed, obtain approval from the person in charge of the user department to finalize the design documents. For the later changes, create a rule for change management with the customer, and manage changes to the system design documents accordingly. Make sure to follow this procedure, because any changes in design specifications in subsequent processes are handled based on the current specifications. Note that failure to follow this procedure results in unclear system design.

(2) Peer review by developers

A peer review by developers is conducted for the following purposes:

- 1) Verification that all user requirements are included.
- 2) Check for any problems in the methods of implementing user requirements.

These problems include not only problems concerning hardware and software performance but also inappropriate partitioning into subsystems and components, and interfaces. Users will especially complain about missing functions and performance problems. Therefore, carry out complete checks at this stage. A failure to do so may later result in serious and critical problems. Large systems are often developed in smaller units of subsystems, and therefore, a complete preliminary examination that includes interfaces is extremely important to minimize the later re-work in the design stage.

<http://www.vitec.org.vn>

The purpose of design reviews is to find problems in specifications. Making personal attacks regarding errors and omissions in design is an abuse of the review process, even if such attacks occur because of earnest attempts to discover the cause of design problems.

Instead, it is more important that every developer checks whether the issues pointed out for other developers are also applicable to them.

For peer reviews by developers, it is recommended to include members who are participating in the next process, if possible. This can save time that would otherwise be spent repeating explanations, and allows to transfer exact internal design specifications. For these reasons, include these members wherever possible.

6.2.2 Review types

This section explains the types of review that can be conducted. Select the type of review according to the frequency of review, the participants, how the review is conducted, and the mode of review.

(1) Inspection

The primary purpose of an inspection is to find defects in specifications. An inspection is a design review held as a conference style hosted by a moderator, who is the person

responsible for the review. This type of review does not include a detailed examination of the contents of design documents, because its purpose is to find defects and problems. As this is a general-purpose review, it can be applied at any stage of software development.

(2) Walkthrough

A “walkthrough” is a method of finding defects in specifications while its participants increase their understanding of the specifications. The method involves simulations of system operations as the specifications are read through. This type of review is suitable for checking documents that include flows and/or processes, including control and data flows.

(3) Round robin

“Round robin” is a method of examination where the persons responsible for designs give explanations in turn. This is effective for exchanges of mutually applicable information and improving the technical knowledge and skills of everyone involved. “Round robin” is often used for training rather than as a review.

6.2.3 Format of specifications

After a review, read the minutes to examine the details, and reflect the results in the applicable design documents. Correct portions that have to be corrected to improve the quality of the specifications that have gone through the review. Anticipating the detection of defects, it should be considered to include beforehand a check column so that the progress can be traced.



<http://www.vitec.org.vn>

[Reference: Review items]

Review items	What should be checked
Completeness	<ul style="list-style-type: none"> - Is there an item for which a decision is pending? - Is there a reference to a nonexistent function, input process, or output process? - Is any function missing? - Is there anything left to design?
Consistency	<ul style="list-style-type: none"> - Are notation, terminology, and the use of symbols within specifications inconsistent? - Are items in the specifications consistent with those in the design documents and update specifications?
Feasibility	<ul style="list-style-type: none"> - Is the system designed with ease of use in mind? - Is the system satisfactory from the standpoints of performance, reliability, expandability, maintainability, and portability?
Testability	<ul style="list-style-type: none"> - Are test cases explained using a decision table, etc., so that they can be analyzed easily? - Is there an ambiguous description? - Is everything handled quantitatively?
Reliability	<ul style="list-style-type: none"> - Have all the input error check function been included? - Have all the input combination check function been included? - Has the input value range check function been included? - Has the operation procedure error check function been included? - Has the media setting error check function been included? - Has an examination been carried out for hardware fault preventive measures (such as duplicating)?
Security	<ul style="list-style-type: none"> - Has a function been included to check the authority for using confidential data? - Has a function been included to check the authority for using security functions? - Has a function been included to check for unauthorized access and report it to the system administrator?

<http://www.vitec.org.vn>

Review items	What should be checked
Operability	<ul style="list-style-type: none"> - Are all operational procedures defined, and are the corresponding commands and messages defined? - Is the message format standardized? - Is the system designed to output a message in the event of an error? - Is the method of responding for a response request standardized? - Is the input format standardized? - Are the keywords and parameters for input operands easy to understand? - Is the output format standardized? - Is necessary information such as a title, page number, and output date included in header information? - Are output types clearly defined, so that users can select only those that are necessary? - Are the output units and the calculated values of decimal fractions adequate for users? - Are messages such as error messages from the system clearly distinguished from output data? - Are the response times for an interactive system and online system adequate?
Performance	<ul style="list-style-type: none"> - Are the target performance values clearly defined? - Has the customer confirmed the target performance values? - Are the target performance values higher than those in the existing system? - Has the system life cycle been considered for the target performance values? - Has the validity of performance target setting conditions been examined?
Expandability	<ul style="list-style-type: none"> - Have consideration been made for hardware expansion? - Are the rates of increase in the workload on hardware defined, and do hardware expansion plans correspond to those rates?
Portability	<ul style="list-style-type: none"> - Is software written in a standardized high-level programming language, so that the effect of differences in computer type and OS are minimized? - Is the input/output interface a logical interface, so that differences in hardware have no effect? - Are standard formats used for file formats? - Is the human interface a logical interface, so that differences in hardware and OSs have no effect?

Exercises

1. Select the most suitable description related to design review types.
 - (1) Round robin equalizes the workloads among participants to prevent any effect resulting from differences in participants' experience, skills, and level of knowledge.
 - (2) Inspections enable analysis of problems that have been narrowed down from different angles, and therefore it is easier to find the solutions.
 - (3) In a walkthrough, a leader explains design documents and results, prepared by members to be reviewed by for participants.
 - (4) In an inspection, every participant has to receive explanations about all items during the review meeting, because there is no review in advance.
 - (5) For development with a prototype, reviews can be omitted because users can directly examine the prototype.

2. Select the sentence that does not describe the purpose and results of walkthroughs carried out during the course of a project.
 - (1) By finding errors in the design process rather than in a subsequent process, time and money can be saved.
 - (2) The purpose of walkthroughs is to find errors and does not involve program correction.
 - (3) One of the purposes of walkthroughs is to give managers an opportunity to participate in meetings to evaluate developers.
 - (4) By mutually examining programs by peers, system quality improvements can be expected.

3. Select the correct statement related to design review by walkthrough.
 - (1) A moderator who is neither a developer nor an administrator is selected to be the person responsible for review planning and implementation.
 - (2) Although the focus is placed only on detecting problems, results are reviewed in detail under the initiative of developers.
 - (3) The progress of a project is mainly reviewed from the viewpoint of management.
 - (4) Persons responsible for development and user departments review the appropriateness of a development period and costs.
 - (5) Solutions to problems are examined by discussions held in a free and open environment without restrictions on participants' making comments.

4. Select the two correct statements related to software design review.
- (1) An advantage of design reviews is that errors that may occur in the software life cycle have a high probability of being found early in the upstream process of development.
 - (2) It is preferable to not allow system users to participate in design reviews, because they tend to point out problems from different viewpoints than those of designers, and this makes it hard to make decisions about designs.
 - (3) A design review is useful to set an end point to design work, but the design review may be abbreviated, because it is more effective to conduct a detailed program review in the next phase.
 - (4) Because the purpose of the design review is to evaluate the design itself, designers should not participate in the review, so that designs are evaluated more objectively.
 - (5) People in positions related to the system should participate in meetings to review designs from their separate viewpoints.
5. For the purpose of discovering design errors at an early stage in order to improve software quality and productivity, a review is conducted by designers and persons concerned at the end of each design stage. What is the name of this type of review?
- (1) Top-down test
 - (2) System test
 - (3) Walkthrough
 - (4) Code inspection
6. Select the wrong statement related to document review techniques used in software development.
- (1) A walkthrough requires a period of half a day to one day to find and correct all problems.
 - (2) The purpose of inspection is a review with a focus on finding errors.
 - (3) A project development schedule must include the review period as part of the development work.
 - (4) Conducting a review is expected to not only have the effect of heightening project members' sense of participation but also provide training for new members.
 - (5) Walkthroughs and inspections are quality improvement measures that can be applied throughout the software development processes.

7. Select the correct statement related to a design review by walkthrough.
- (1) The manager prepares a review results report and obtains approval from the quality assurance department.
 - (2) Examination materials outlining the entire system are prepared.
 - (3) Problems pointed out in a meeting are fully discussed immediately in order to determine solutions.
 - (4) The manager plans meetings and selects participants.
 - (5) Examination materials are distributed in advance, so that participants can acquire a general understanding of problems before the meeting.
8. The following explanation is about a certain review method. What is the name of that review technique?
- The examination is performed while the persons responsible for design give explanations in turn. This is effective for exchanges of mutually applicable information and improving the technical knowledge and skills of everyone involved. This method is often used for training rather than as a review.
9. Which of the following documents is not a design document (specifications) prepared during internal design?
- (1) Component decomposition diagram
 - (2) Code design document
 - (3) Component processing specifications
 - (4) Screen design document
 - (5) Form design document
 - (6) File design document
 - (7) Database design document
10. Select the sentence that is not correct as statement related to the contents of form design.
- (1) The display locations and formats of the title, date of creation, and page count should be standardized wherever possible.
 - (2) The positioning of header item names with corresponding data contents should be defined in the way that they become as easily to read as possible.
 - (3) If there are multiple totals such as subtotals, intermediate-totals, and grand totals, they must be arranged on a single line wherever possible.
 - (4) Page feed and line feed conditions are to be noted individually.

Answers for No.3 Part 3 Exercises

Answers for Part 3 Chapter 1 (Internal Design Procedures)

(No exercises for this chapter)

Answers for Part 3 Chapter 2 (Software Component Design)

1. b
2. (1) c (2) g (3) a (4) b (5) h (6) e (7) d (8) f
3. (1) c (2) b (3) a (4) d
4. (1) b (2) c (3) a (4) d
5. (1) d (2) c (3) b (4) a
6. (1) c (2) a (3) d (4) b
7. (1) f (2) b (3) a (4) c (5) b (6) f (7) e (8) d
8. (1) O (2) O (3) O (4) X
9. (1) d (2) a (3) e (4) c (5) b
10. (1) c (2) a (3) f (4) e (5) b (6) d

Answers for Part 3 Chapter 3 (Input-output Design)

1. (1) d, (2) e, (3) b, (4) a, (5) c
2. (1) d, (2) b, (3) e, (4) a
3. (1) x (2) o (3) o (4) o
4. c
5. b
6. (1) b, (2) g, (3) d, (4) e, (5) f, (6) a
7. d
8. (1) b, (2) a, (3) d, (4) c
9. (1) c, (2) f, (3) a, (4) g, (5) h
10. (1) b, (2) e, (3) g, (4) a
11. b
12. (1) d, (2) c, (3) g, (4) e, (5) b

- 13. d
- 14. b
- 15. (1) b (2) c (3) e (4) a, (5) d
- 16. (1) c, (2) b, (3) e, (4) a
- 17. c
- 18. c

Answers for Part 3 Chapter 4 (Physical Data Design)

1. (1) e (2) a (3) c (4) b (5) d

2. (1) b (2) c (3) a

3. d, f

4. c

5. a

6. b

7. b

8. d

9. c

10. f

11. b

12. a

13. d

14. d

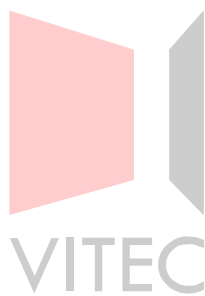
15. b,f

16. (1) i (2) e (3) d (4) b (5) f

17. (1) d (2) b (3) c (4) a

18. (1) c (2) a (3) b (4) e (5) d

19. c



<http://www.vitec.org.vn>

Answers for Part 3 Chapter 5 (Creation and Reuse of Parts)

1. (3)
2. (4)
3. A type part has a logical structure that does not depend on the format and appearance of information, while form part is a part for which the physical format and appearance of information is pre-defined.
4. (3)
5. Quality and productivity during software development can be improved by using commercially available software packages.
6. (1)

Answers for Part 3 Chapter 6 (Creation of Internal Design Documents)

1. (2)

2. (3)

3. (2)

4. (1), (5)

5. (3)

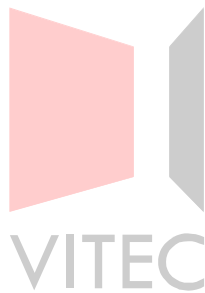
6. (1)

7. (5)

8. Round robin

9. (3)

10. (3)



<http://www.vitec.org.vn>

Part 4

PROGRAM DESIGN

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

1 Procedures for Program Design

Chapter Objectives

This chapter explains the necessary preparations and procedures that must be carried out before program design.

1.1 Program Design Procedures

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

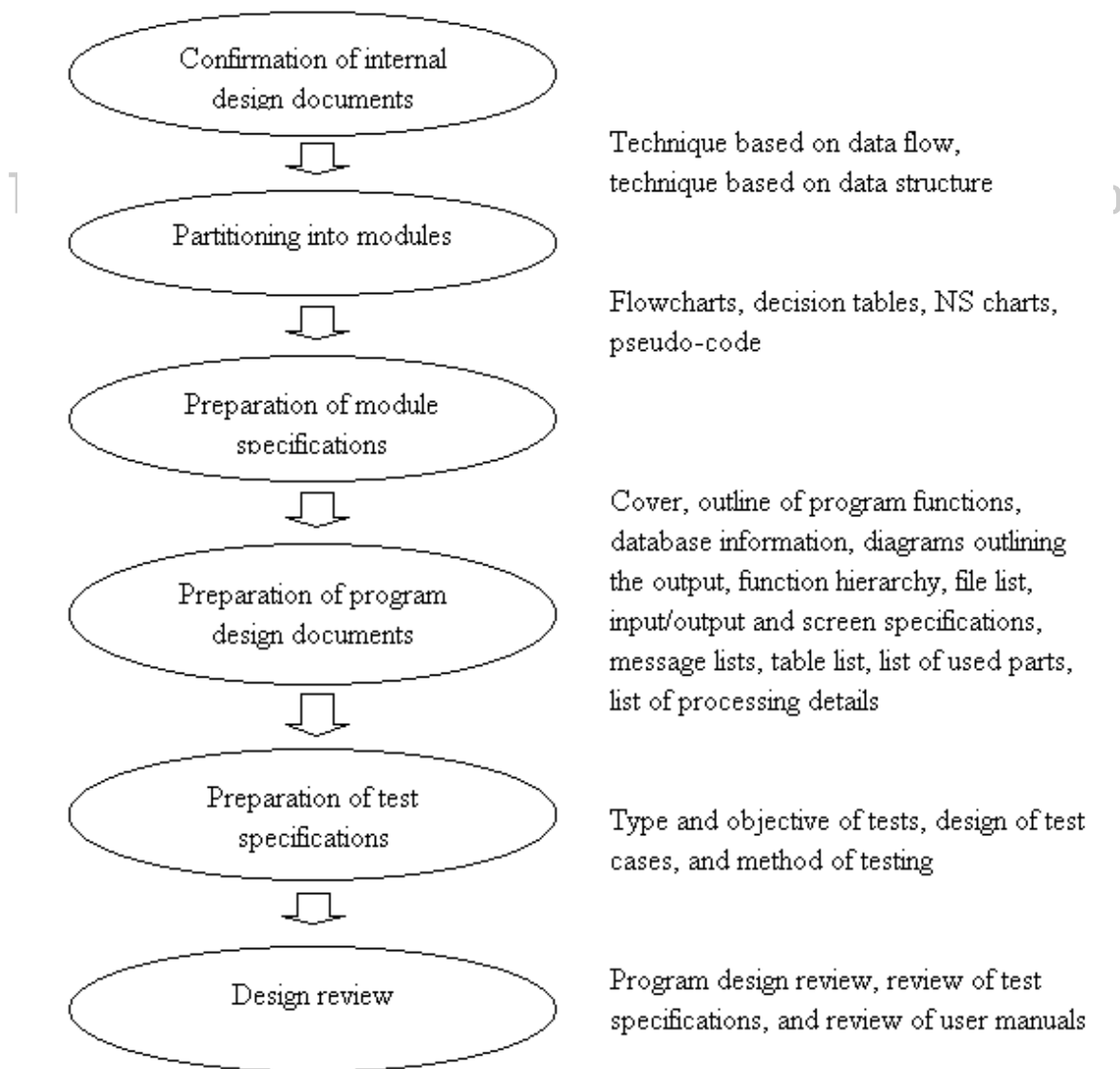
1.1 Program Design Procedures

1.1.1 Confirmation of internal design documents

Internal design documents describe how the subsystems defined during external design are partitioned into functions and provide overall specifications for individual programs. During program design, the contents of processing are partitioned into modules on the basis of the internal design documents and design what kind of processing each module will perform.

First, let's look at the procedures for program design work.

During internal design, such documents as program partitioning diagram, the file/record specifications, and form layout specifications are prepared. The program designer will build the processing logic for a program based on these documents. Program design consists of the following procedures:



1.1.2 Partitioning into modules

During program design, the program is partitioned into modules that combine functions. Modules can come in various sizes. It must also be considered that a program may be called from another program, or that a function module within a program may be called

from within the program itself.

There are generally two methods of partitioning a program into modules: partitioning based on data flow and partitioning based on data structures.

The designer must understand these two methods, in addition to considering productivity during program development and maintenance, as well as product quality. In addition, it is also necessary to consider characteristics of the system and of the developer and select the appropriate method.

1.1.3 Preparing module specifications

The module specifications describe the processing details for each module.

The description methods for module specifications are as follows:

- 1) Flowcharts
- 2) Decision tables
- 3) NS charts
- 4) Pseudo-code

1.1.4 Program design documents

Program design documents can become highly important for providing the quality required by customers and for increasing work efficiency.

Unnecessary re-work can be prevented by having the appropriate documents. Moreover, this makes the work caused by changes in the specifications in the future more efficient.

The structure of program design documents is shown in the following table:

Item	Outline
Setting program design document preparation criteria	Contents of documentation Criteria for changes to do maintenance/improvement Contents of user's manuals
Description of processing outline	Outline of program functions How to use the program
Detailed design of database	Clarification of database input-output conditions Deadlock and rollback
Standards for interfaces between modules	Requirements for interfaces within a program Requirements for interfaces between programs Requirements for interfaces between reusable programs

1.1.5 Preparation of test specifications

The test specifications are for preparing testcases to test whether the program works as specified in the program specifications and also for describing the details of the testing.

This contents include the following:

- 1) Type and objective of tests
- 2) Testcase design
- 3) Testing method

1.1.6 Design review

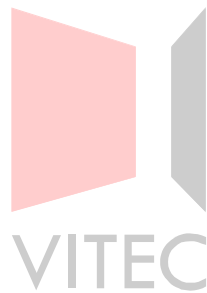
When the detailed processing procedures and the used data areas have been determined, conduct a design review.

The design review allows early detection of errors in the details of processing and is important for preventing problems from being carried over to subsequent work phases, thereby improving the quality of the software.

The design review consists of the following:

- Program design documents review
- Test specifications review
- User's manual review

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

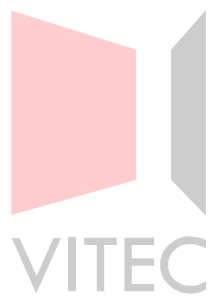
2 Program Design Criteria

Chapter Objectives

This chapter explains the techniques and criteria for module partitioning during program design. It also explains how to combine modules in parts and reuse modules to improve program development productivity and program quality.

- 2.1 Choosing Module Partitioning Technique
- 2.2 Module Partitioning Criteria
- 2.3 Reuse of Patterns and Parts

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

2.1 Choosing Module Partitioning Technique

The techniques for partitioning modules can be roughly divided into two ways.

This classification of techniques is shown in the following table.

Partitioning by data flow	STS partitioning method Transaction partitioning method Common function partitioning method
Partitioning by data structures	Jackson method Warnier method

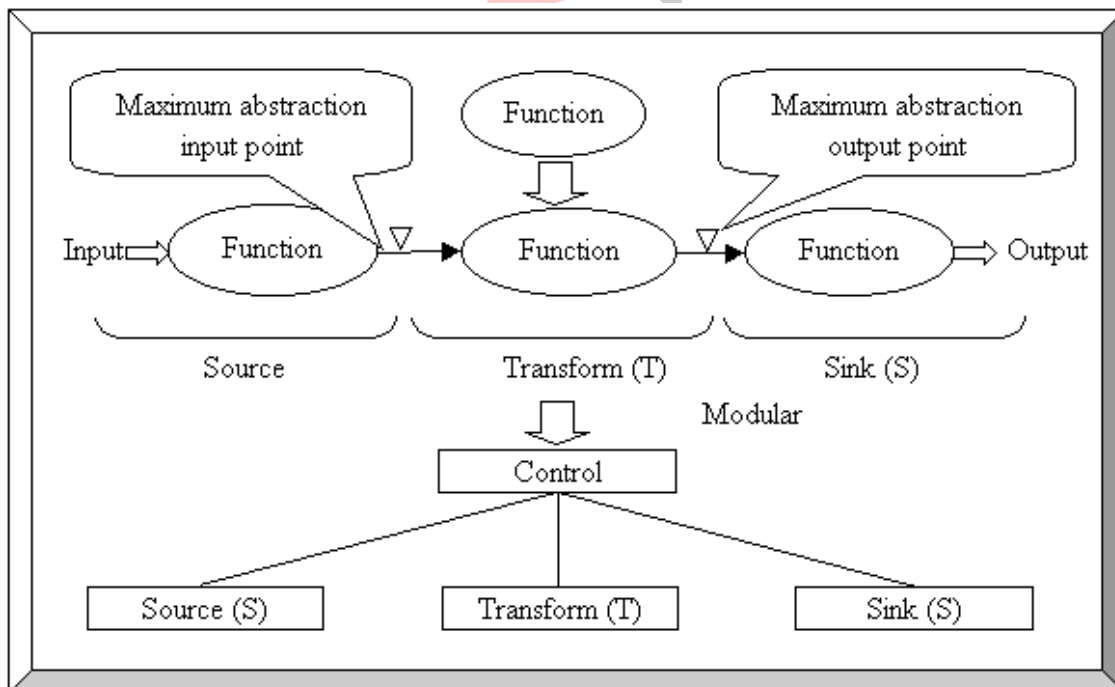
2.1.1 Partitioning by data flow

(STS partitioning method, transaction partitioning method, common function partitioning method)

(1) STS partitioning method

The acronym STS stand for Source, Transform, and Sink. In the STS partitioning method, program processing is divided into three functions: Source (input section), Transform (processing section), and Sink (output section). STS partitioning has partitioning criterion boundary points called the maximum abstraction input point and the maximum abstraction output point, and modules are partitioned using these points. The section from data input to the maximum abstraction input point is defined as S (Source), the section from the maximum abstraction input point to the maximum abstraction output point is defined as T (Transform), and the section from the maximum abstraction output point to output is defined as S (Sink).

The method is illustrated below:



STS partitioning

Maximum abstraction input point	A point immediately before the input transformed losing its original form as it is.
Maximum abstraction output point	The point where the output data begins to be created, as process steps are traced in the reverse order from the final output.

[Procedures of STS partitioning]

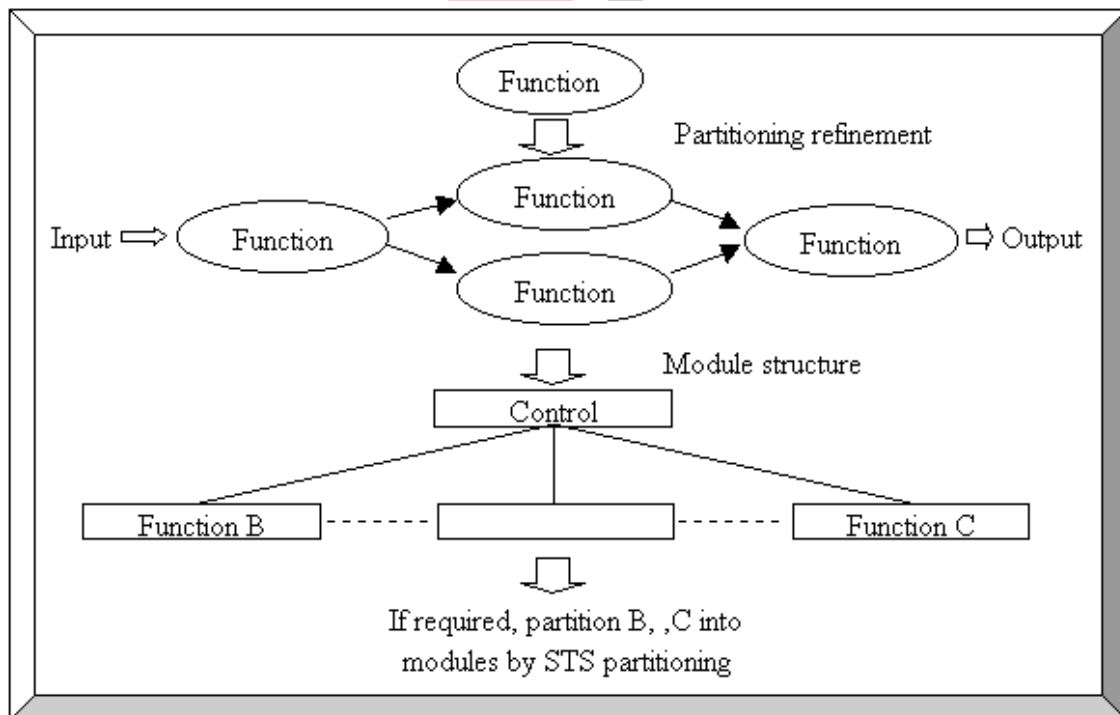
- 1) Clarify and extract the problem structure.
- 2) Determine the main flow of input data and output data in the problem structure.
- 3) Find the logical final point of the input data flow and the logical first point of the output data flow.
- 4) Use these points as partitioning points, and describe each section as a single function.

(2) Transaction partitioning method

The term transaction means a unit of processing for which a request is issued to the computer to perform a specific process.

In the transaction partitioning method, when the data flow has a conditional branch, modules are partitioned with the focus on transactions. In other words, this partitioning method is used when input data can be classified into different types of processing.

If it is difficult to implement STS partitioning, the transaction partitioning method is a more suitable method. The method is illustrated here.

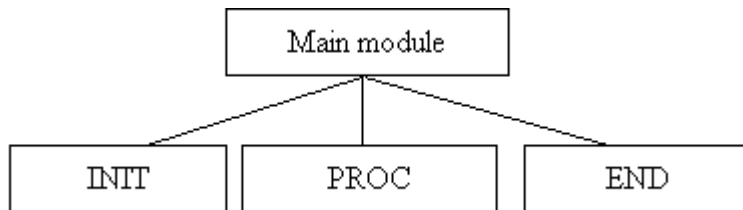


STS partitioning is suited for partitioning based on data flow without branches, while transaction partitioning is suited for branched data flow. These two partitioning methods are classified as structured design.

Data flow without branches	Basically, the same processing is performed for all input data.
Data flow with branches	Two or more types of input data exist, and processing suited to each type is performed.

(3) Common function partitioning method

The common function partitioning method is to separate a module having a commonly used function from those which have been identified as a result of either the STS partitioning on transaction partitioning. Generally, the main module handles three types of processing, INIT (initialization), PROC (procedure), and END (end). The main module thus consists mainly of instructions calling these three processing functions.

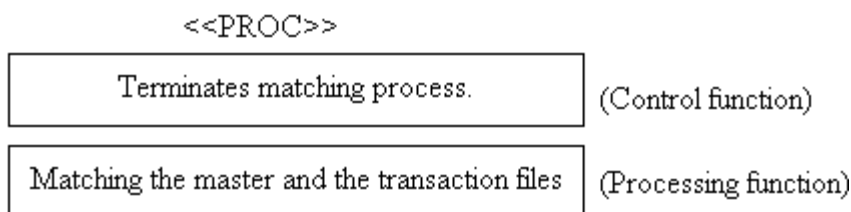


Partitioning the PROC function as follows.

The first step is to find the maximum control function in PROC processing. Generally, the maximum control function for PROC processing is control for processing termination. In this example, the maximum control function is control of the matching process termination of the master and the transaction files.

The next step is determining the process functions, which is dependent of the control function.

The processing functions of the PROC process are the control functions are separated from those which matches the transaction file master.



After that, control functions and processing functions in this processing function are examined.

The maximum control function included in the process function that match the master and transaction files is the comparison of the key items in the master and transaction files. Three processing functions that belong to this function are: processing performed when the key item is found only in the master file; processing performed when the key item in the master file matches with the corresponding key item in the transaction files; and processing performed when no key item is found in the master file. Thus, the hierarchical structure is as shown here.

<<Matching the master file with transaction files>>

Compare key items			(Control function)
Processing when the key item is found only in the master file	Processing when the key item in the master is matched with the corresponding key item in the transaction files.	Processing when the key item is not found in the master file	(Process function)

Find the control function in each processing function, and its subordinate processing functions. Repeat this procedure and partition until no control function can be found.

- Processing when the key item is found only in the master file

This function does not include a control function. Accordingly, this means no further partitioning. At this time, the processing function name can be changed to the master file copying, and this name represents the function more accurately.

- Processing when a key item in the master file matches with the corresponding the corresponding key item in the transaction file.

The control function within this function determines the following correction category in the transaction record.

Correction category	Subordinate processing function
1(Indicate new setting)	Error processing (duplicate)
2(Indicate change)	Change processing
3(Indicate cancellation)	Cancellation processing

The processing functions that are partitioned into three components do not contain a control function. Thus stops the partitioning at this point. Change the name of the processing that is carried out when the master file matches with the transaction files the “update of master file” that concretely represents the function.

- Processing when the key item is not found in the master file

In this case, the control function is to determine the correction category in transaction record.

Correction category	Subordinate processing function
1 (Indicate new setting)	New setting processing
2 (Indicate change)	Error processing (unmatched)
3 (Indicate cancellation)	Error processing (unmatched)

In this case, partitioning ends here. When the key item is not found in the master file, change the name of the processing function to the “creation of a new master file” which concretely represents the function.

When no more control function is found, focus on the processing function to partition major processing functions into single functions.

Single function	Write a new master file	Read a master file	Read a transaction	Write a maintenance listing
Major master file				
Copy a master file	○	○		
Update a master file			○	
Create a new master file			○	
Change processing				○
Cancellation processing				○
Error processing (duplicate)				○
New setting				○
Error processing (change)				○
Error processing (cancellation)				○

The objective of this partitioning work is to obtain information about common internal modules in a program.

If the same function is found in several places, it can be handled as a common internal module in the program.

In this example, the following four functions, which appeared as single functions separately, are considered as a module:

- Write a new master file
- Read a master file
- Read a transaction
- Write a maintenance listing

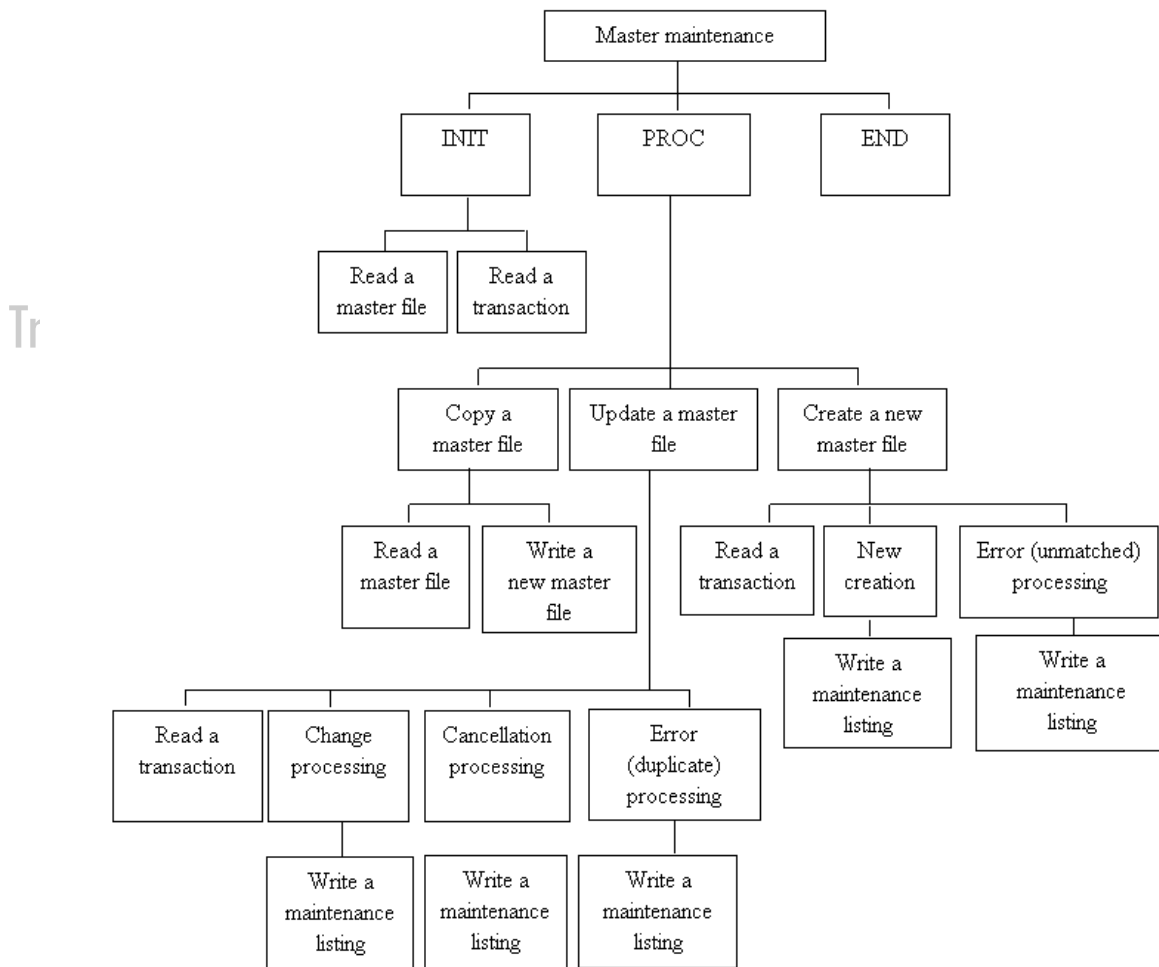
- Combine the functions into unit of module so that it is easier to exist as such.

While consolidate modules that can be shared, name each module in such a manner that represents the functions and its functions can be easily understood from the name.

In this example, error processing for “unmatched state of change indication” and error processing for “unmatched state of cancellation indication” can be combined into one module.

- Use the same method for INIT processing and END processing to refine function and evaluate the whole module structure.

Summing up the description is the above descriptions, the hierarchical structure of program functions is shown here.



2.1.2 Partitioning methods focusing on data structure (Jackson method, Warnier method)

(1) Jackson method

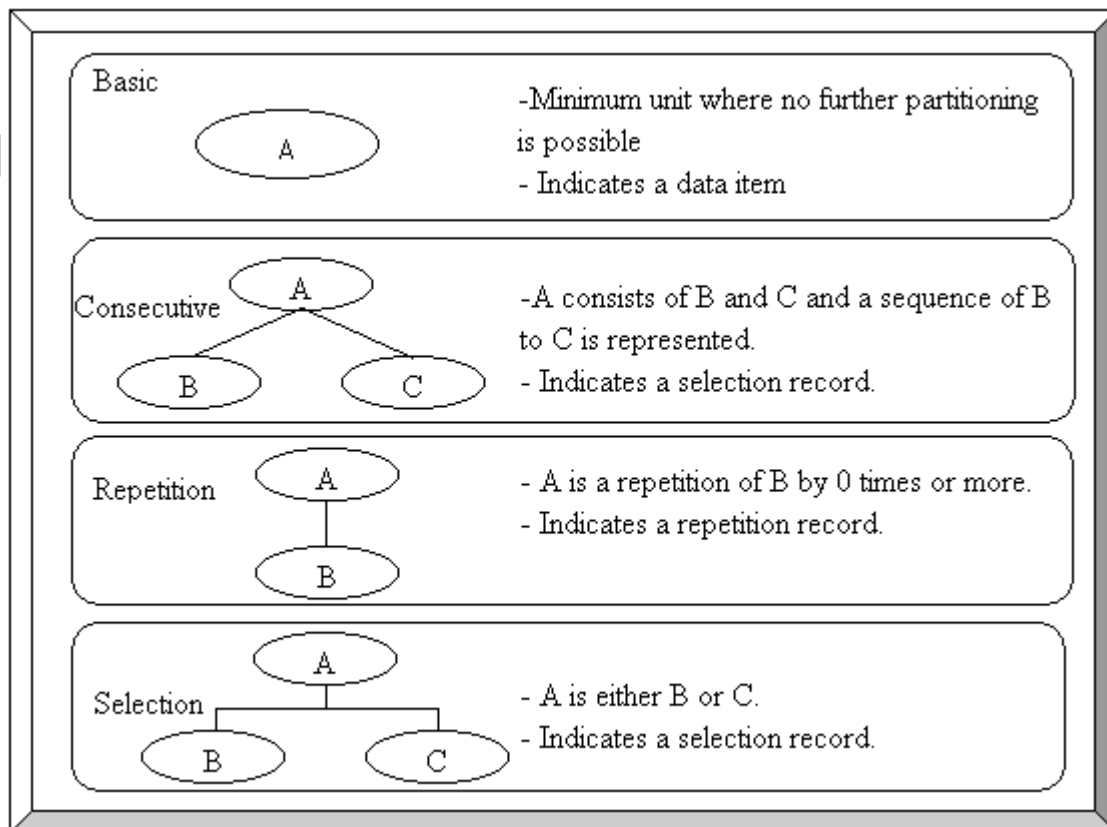
In the Jackson method, the program structure is designed according to the corresponding relationship between the input data structure and output data.

With this method, the program is considered as a system-internal function to convert input data into output data. The method is based on the concept that when the data correspondence relationship is revealed, the structure of the program can be clarified as well. The Jackson method is therefore suitable for handling input-output of a clear data structure, such as in job processing systems.

The procedure for partitioning a program is as follows:

- 1) Define the structure of input data and output data.
- 2) Check for a one-to-one relationship between components of input data and output data. If no relationship is found, define an intermediate data structure.
- 3) Create the program structure based on the corresponding relationship of the components.
- 4) Assign basic instructions according to the program structure.

In the Jackson method, it is important to clarify the structure of the input data and output data. The structure of data can be represented by three basic structures: consecutive (sequential), selective, and repetitive. In these structures, the smallest component is handled as a basic item. The structure of data is then represented with a chart called the JSP tree structure. In the Jackson method, the same representation method is used for both the structure of data and for the program structure.

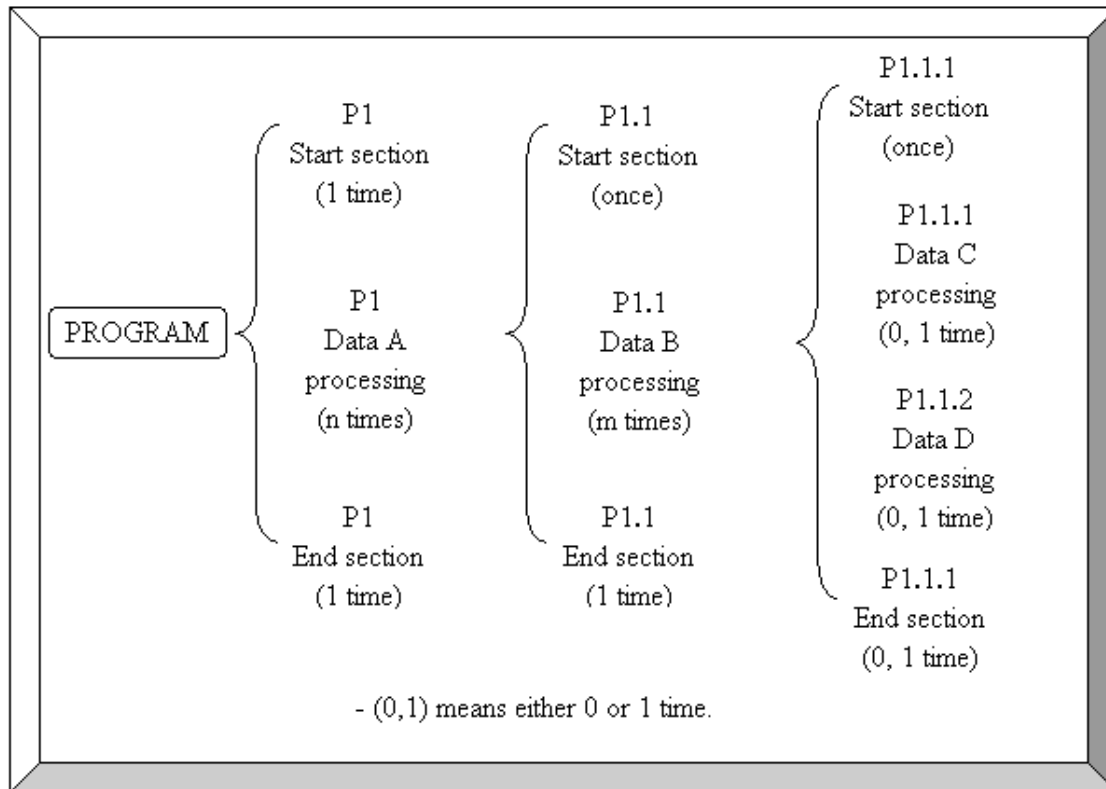


JSP tree structure

(2) Warnier method

In the Warnier method, the program structure is designed based mainly on the structure of the input data. The Warnier methods to analyzes the input data structure with regard to Sequence, Selection, and Repetition, as in the same as Jackson method. Compared with the Jackson method, however, this method places less importance on the basic structure than on Jackson method and places more importance on how many times the data appears during processing.

As the Warnier method performs processing based on the contents of input data, the processing structure is determined according to how often input data appears during processing. The characteristics of this method is to determine the program structure based only on the structure of input data.



Program structure diagram of Warnier method

(3) Combined use of partitioning methods

The modules partitioned by a various ways can be combined. For example, if the contents of processing differ according to the type of data, first use the transaction method to partition a module in units of transactions. Partitioning by the transaction method eliminates branching of the data flow in a module. If further partitioning of a partitioned module is required, partition it based on the STS partitioning method. If branching is included in the data flow inside a module partitioned with this approach, use the transaction partitioning method to allow further partitioning

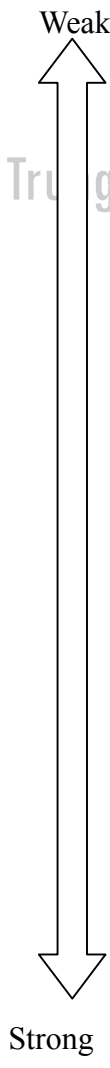
2.2 Module Partitioning Criteria

2.2.1 Module size

The size of the module need not to be determined in particular, but if a module is too large or too small, check whether or not the module can be partitioned or join with other modules. This is because the module independency lowers if it is too large, and management becomes complicated if it is too small.

2.2.2 Module strength

Module strength is one of the major module partitioning criteria, and it represents the degree of mutual dependency in a module. The greater the module strength, the higher the module independence within the module, and this means a better module. There are seven basic levels of module strength, and they are summarized in the following.



Module strength	Features
(1)Encryption strength	<ul style="list-style-type: none">- There is no mutual dependency of elements within the module.- The function of the module cannot be defined.- Change of another module is likely to affect the module.- No advantage can be found in module independence.
(2)Logical strength	<ul style="list-style-type: none">- Execute required processing as indicated by parameters.- Even if two or more functions exist, only one function is executed.
(3) Time strength	<ul style="list-style-type: none">- Two or more functions can be found, and the functions are executed sequentially.- Initialization and end processing modules are typical examples.- Each function may have strong relationships with other modules, but independency is weak.
(4)Procedural strength	<ul style="list-style-type: none">- Execute two or more functions in the module sequentially.-The relationships between functions can be linked by problem specifications and job specifications.
(5)Linkage strength	<ul style="list-style-type: none">- Procedural strength conditions are satisfied, and all functions are linked with the use of data.- Each function refers to the same data, like the result of function A being used by function B.
(6)Information strength	<ul style="list-style-type: none">- Multiple entries are available for each function.- At each entry point, execute a single function.- Changing the data structure of the module does not affect other modules.- The control flow between functions are not related.
(7)Functional strength	<ul style="list-style-type: none">- The inside processes are related to execute one function.- This degree of strength is the highest, and each function is a basic function.

(1) Encryption strength

Encryption strength is also called “incidental strength” and is the weakest degree of strength. Such modules have no special relationship among their module elements. A module incorporating only encryption strength is thus a bad module.

- Definition

When program elements with no special mutual relationship are combined and handled as one module, the binding strength of the program elements is weak. In this case, the program elements are said to be bound with encryption strength.

(2) Logical strength

A module featuring logical strength is one in which a relationship of logical abstraction between module elements can be found and some related functions are executed when the module is called. Logical strength is the second weakest, and a module incorporating it is not a good module.

- Definition

When several functions (program elements) that are in a logical relationship are combined into one module, a parameter reference by the module determines which function is executed. In this state, each function is bound by logical strength. In other words, it is a module whose processing contents and routes are changed with parameters.

(3) Time strength

Time strength includes a time element in logical strength. In other words, the elements within the module have a time relationship that allows their consecutive execution a specific period of time.

- Definition

When several functions that are executed during a specific period of time are combined into one module, these functions (program elements) are bound with time strength.

- Reason why time strength is higher than logical strength

Functions that are bound with time strength share the problems of functions bound with logical strength in a sense that separate functions are collected and combined into one module. However, because the individual functions are not bound by a logical relationship, the importance of logic is reduced, and these functions do not become as complicated as functions that are bound with logical strength. For this reason, the time strength is higher than logical strength.

(4) Procedural strength

A module having procedural strength is a module containing multiple functions that are executed sequentially.

- Definition

When functions whose execution sequences are closely related are combined in one module, the functions are bound with procedural strength.

- Characteristics

A module having procedural strength is comparably high in the strength. However, these modules have the problem that the logic (procedure) is mixed with the functions.

The reason for this is that it is not clear that the processing sequence can be planned after the functions are defined.

(5) Linkage strength

Linkage strength means that elements are linked by their mutual relationship in addition to procedural strength. In other words, all module elements refer to a collection of the same data or data is passed and received among elements in the module.

- Definition

In addition to procedural strength, which means that the functions whose execution sequences are closely related are collected as one module, these functions are bound by a linkage relationship, if each of the functions refers to the same data or are closely related while passing and receiving data.

- Characteristics

Linkage strength is higher than procedural strength because, in addition to the procedural binding strength, the mutual elements are bound and clarified through the data.

(6) Information strength

A module having information strength is one in case two or more functions handling the same data structure are executed. These functions can be used independently. In other words, a module having information strength is the one in case one module is a physically collection of two or more modules that have functional strength (the highest degree of strength). This represents the concept of information localization, and a series of processing for one data item is handled by making it independent in a single module. This means that information strength stands for more independence than functional strength. However, based on the resulting complexity and relationship of the processing logic, information strength is defined as lower than functional strength.

- Definition

When two or more functions that handle a single data structure are executed, these elements are bound with information strength.

(7) Functional strength

In a module with functional strength, all elements are linked for the execution of the same function, and this is the highest degree of linkage. The resulting modules are therefore the best modules.

- Definition

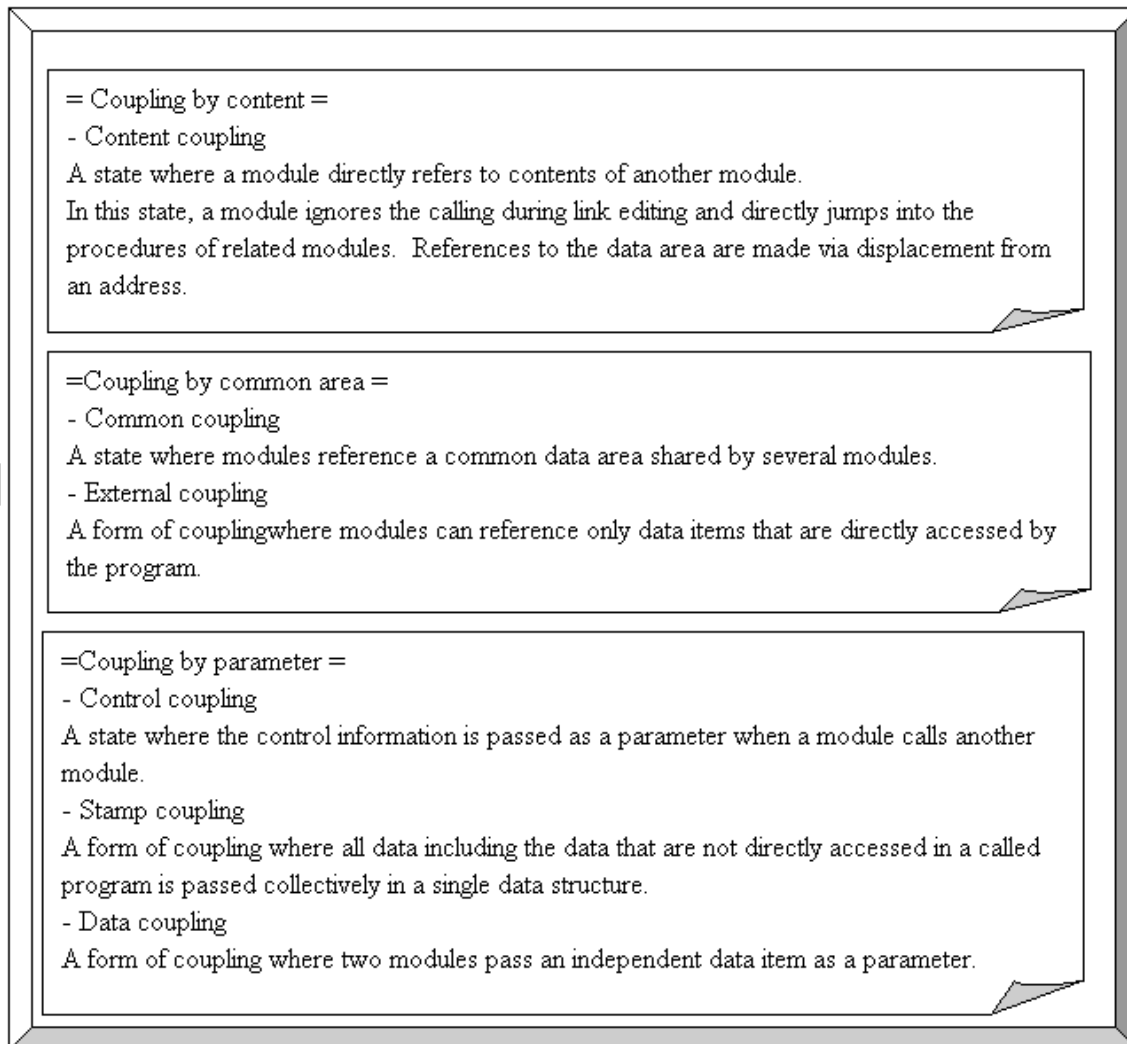
When all elements in a module execute a single function, these elements are bound with a functional strength.

2.2.3 Module coupling

Module coupling is one of the scales used to measure module independence.

In the same way as module strength, module coupling is evaluated from the view point of maintainability. Less coupling means higher module independence.

The six degrees of coupling are as follows:



The following describes coupling in the order of strength from strong one to weak one.

(1) Content coupling

Content coupling is defined as a state where a module refers directly to contents of another module. It is the worst coupling relationship because modules are coupled most strongly.

- Definition

When module content is found to have already been referenced, and another module refers to the same module contents, these two modules are coupled by content.

(2) Common coupling

Common coupling is a state which a common data area shared by several modules is referenced, such as by module coupling via a control block allocated in main memory.

- Definition

When several modules refer to a common data area, these modules are subject to common coupling.

In common coupling, each module is strongly coupled via a common data area. Thus, partial changes of a data area affects all modules referencing that area.

(3) External coupling

External coupling is defined as a state in which other modules refer to externally declared data.

- Definition

When another module refers to an item declared to be external by a module, the two modules are externally coupled.

- Characteristics

External coupling and common coupling are the same with regard to module coupling via data. However, there are the following differences: While external coupling indicates a coupling relationship established via a small-sized data item, common coupling indicates a coupling relationship through a large volume of data. External coupling is unrelated to the structure of data, because inter-module referencing is rather related to individual data items. Thus, external coupling is weaker than common coupling among modules.

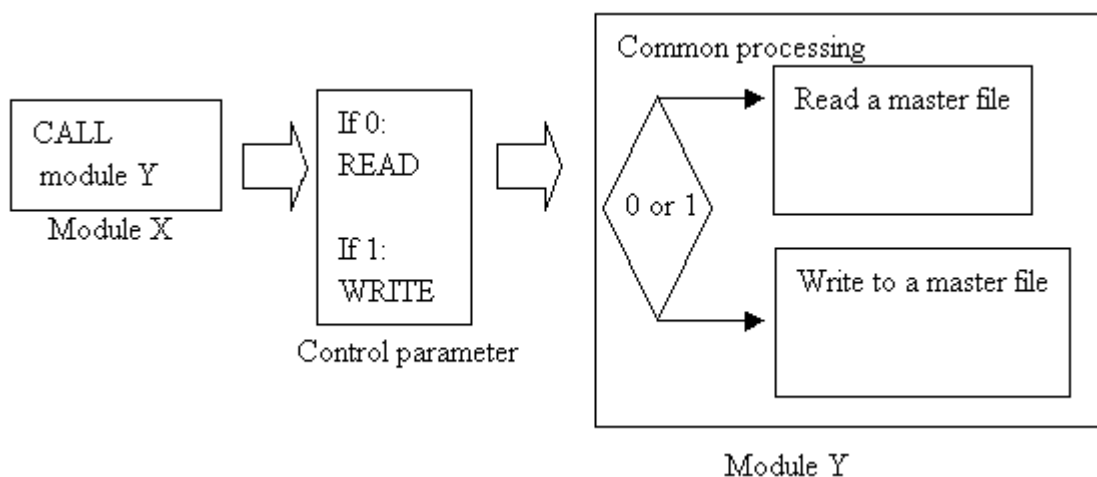
(4) Control coupling

Control coupling is defined as a state in which a module passes control information via a parameter when the module calls another module. The control information has a direct effect on execution of the directly called module.

- Definition

When a module calls another module and passes control information via a parameter, the two modules are subject to control coupling.

- Example of control coupling



- Characteristics of control coupling

Control coupling is a weaker type of coupling among modules. Control coupling is not desirable, because it requires that the contents of processing in the called module is known. In the example, when module X calls module Y and the control parameter is 0,

it is necessary to know in advance that module Y reads the master file. That is, if the processing of module Y is a black box, module X cannot call module Y. In other words, a module that is handled as a black box is a good module in that it can be called without the knowledge of the internal processing.

(5) Stamp coupling

Stamp coupling indicates a state in which two modules refer to the same data structure outside the common area. This occurs when a module is called, while handling the data structure as a parameter.

- Definition

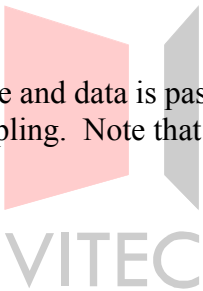
When a module calls another module and a data structure not located in the common area is passed as a parameter or argument, the two modules are subject to stamp coupling.

(6) Data coupling

Data coupling is defined as a state in which two modules pass an independent data item via a parameter. This is the weakest degree of coupling. In this case, a data element parameter is passed and received. The parameter is neither a control parameter nor a parameter for a data structure. Data coupling represents the best type of module coupling. In this type of coupling, the modules have the highest degree of independence, because the degree of coupling is the weakest.

- Definition

When a module calls another module and data is passed as a parameter or argument, the two modules are subject to data coupling. Note that the parameter can only be used for data, not for control information.



<http://www.vitec.org.vn>

2.2.4 Control scope and impact scope

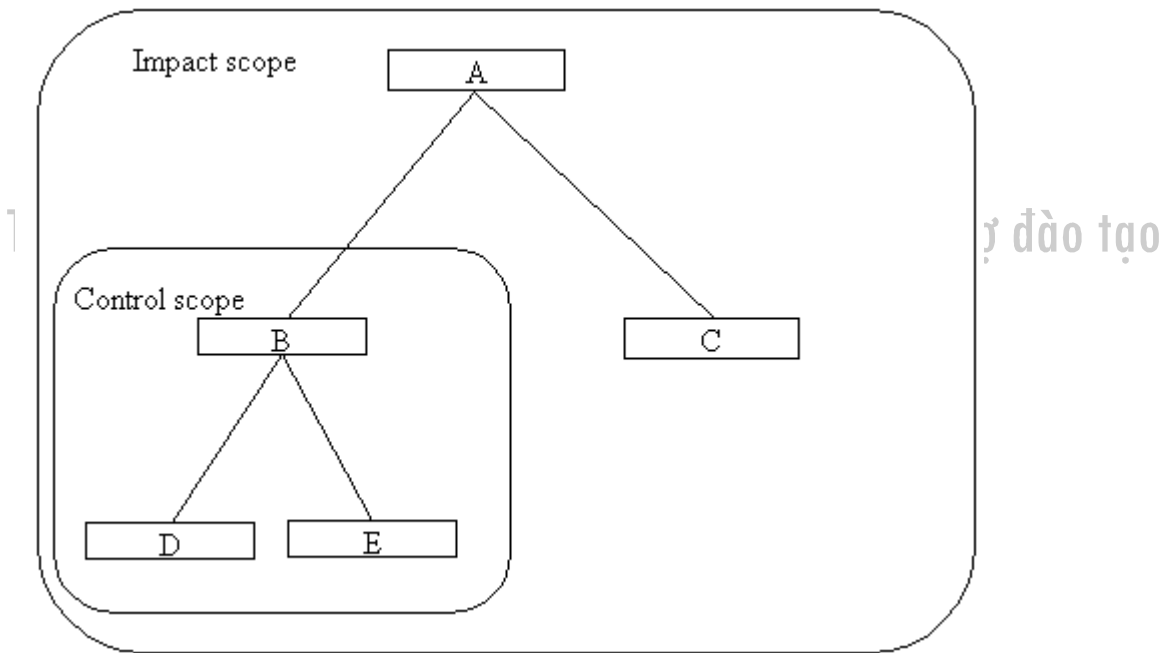
The control scope and impact scope are scales representing the module-related ranges used to make sure that partitioning into modules was properly done.

- Control scope

The control scope means the scope over which a module controls all subordinate modules.

- Impact scope

The impact scope of effect means the range over which modules are affected by the contents specified in a module.



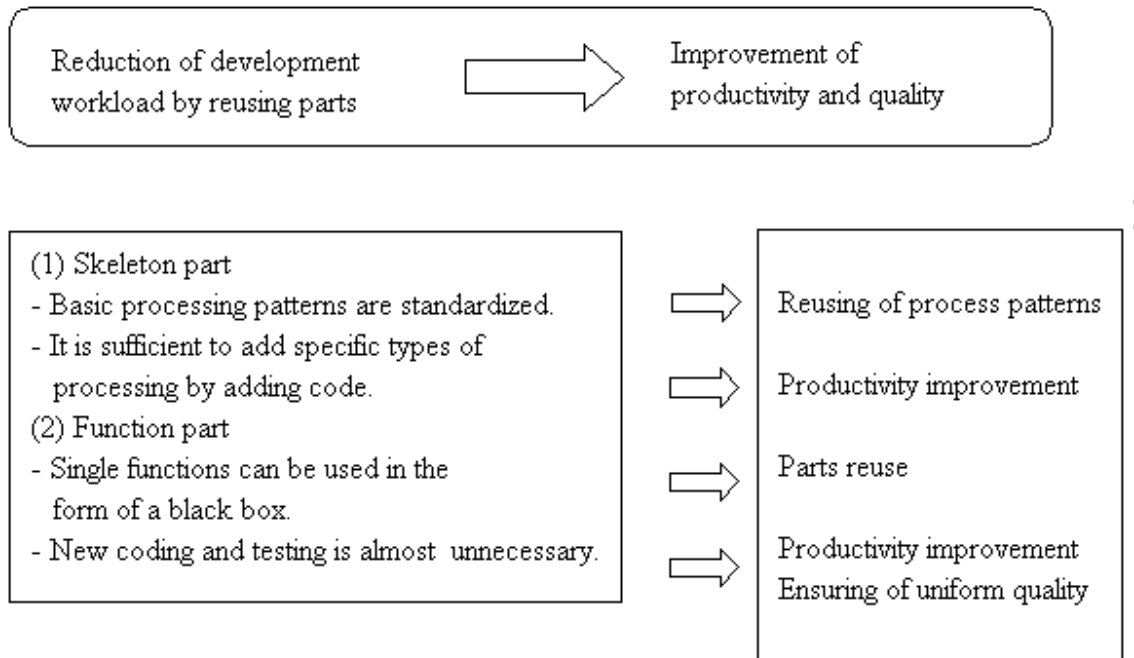
Control scope and impact scope

<http://www.vitec.org.vn>

2.3 Reusing Patterns and Parts

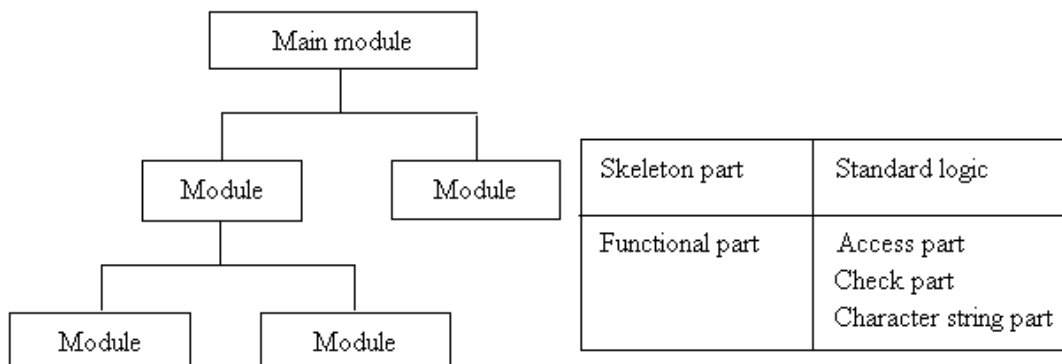
2.3.1 Extracting process patterns and parts

The concept of reusing software parts has its background in a diverse society where rapid changes and developments take place along with new developments in computer systems. In traditional software development, every new project began again right at the start of development, and the similar work was repeated for different systems. With the radical changes and developments in society, however the requests for software development and the demand for softwares increased as well. Software developers using conventional concepts could not cope with the demand. This resulted in an approach in which software parts were not developed newly every time, but reused. Software parts are categorized as skeleton parts and function parts. This concept is illustrated in the following.



<http://www.vitec.org.vn>

In this context, function part coding and testing means coding and testing of an intermediate processing logic that passes information to and receives information from the function part interface.

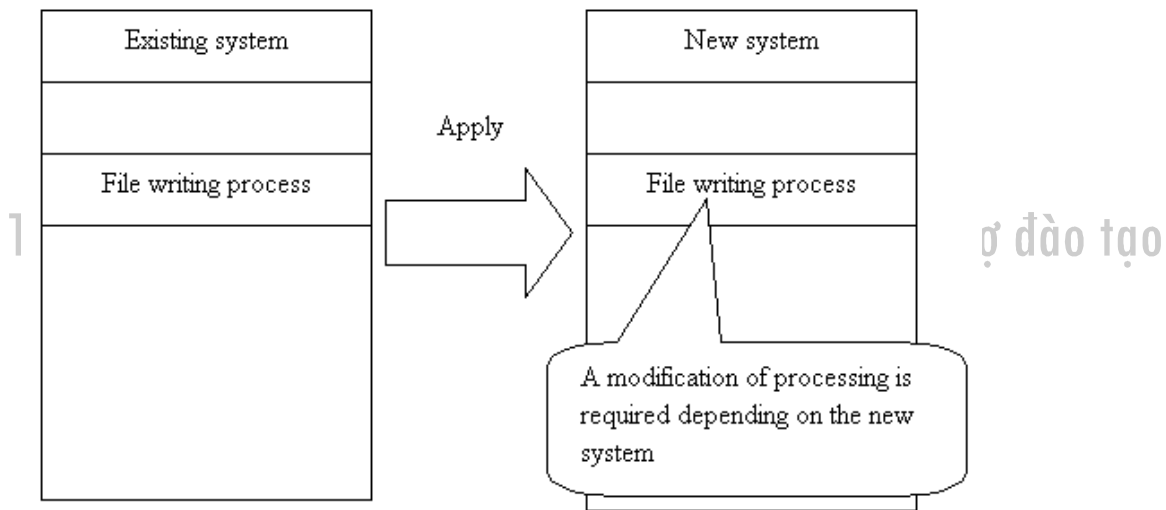


Today, reuse of parts is divided into two types of concepts: intrinsic reuse and schematic reuse. The following explains these concepts:

(1) Heuristic reuse

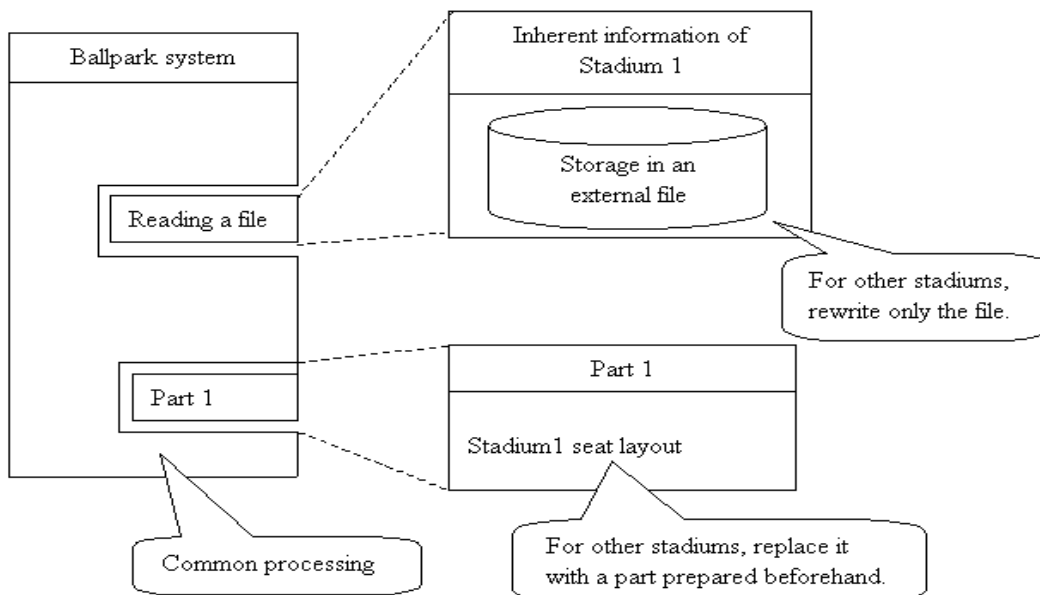
This concept means to try to find a reusable part from a system that is currently operating and then create a part from it. In other words, a function that is required in a system to be developed is identified, and a similar function is then extracted from another system and used in the form of a part. In this concept, a lot of time is required to determine and analyze reusable parts. In addition, such problems as differences in program quality occur depending on the technique used by the person modify a program.

Heuristic reuse is illustrated in the following.



(2) Planned reuse

In system development, this concept means to identify sections that will be commonly usable and sections that will have to be changed, and prepare them as reusable parts. This results in a reduction of the time required for analyzing an existing system and finding reusable sections. With this concept, however, programs must be modified to match with the new systems. Planned reuse is illustrated in the following:



2.3.2 Reviewing the processing pattern and parts list

Check the parts extracted and assumed to be reusable in order to verify their validity and necessity. Items to check are as explained in the following.

Necessity	-Examine whether the function is necessary as a part
Validity	-Examine whether the parties partitioned into the smaller unit -Examine whether a partitioned part has a similar one. If a similar part is found, examine whether those two should be combined.

2.3.3 Reuse rate of processing pattern and parts

The rate of parts reuse can be determined with two approaches. One is the ratio of the number of programs using a part to the total number of programs, and the other is the number of parts used in one program.

Program name	Part 1	Part 2	Part 3	Part 4	Part 5
Program 1	<input type="radio"/>			<input type="radio"/>	
Program 2	<input type="radio"/>	<input type="radio"/>			<input type="radio"/>
Program 3	<input type="radio"/>		<input type="radio"/>		
Program 4		<input type="radio"/>			<input type="radio"/>
Program 5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		

According to the above table, the rate of reuse of part 1 is 80% with the first approach. With the second approach, the total parts utilization rate is 40% for program 1 and 60% for program 2.

2.3.4 Standardizing the processing patterns and parts

Extracted parts are standardized in this phase. Standardization means determining the function of a target part, how it works, and how to call it. These are called the parts external specifications.

The method for external specification is explained in the following.

Candidates for classified parts are listed below together with the classification methods.

(1) Classification of candidate parts

Classify the candidate parts. There are following classifications..

Classification item	Classification method
Calling method	- CALL calling - PERFORM calling
Method of use	- Essential - Replace - Select - External attachment
Program used	- Commonly used - Used in an online program - Used in a batch program
Functions	- Search - Receive - Inquire - Calculate (and other functions)

(2) Determining the parts interface

Interface	Description
Argument	What information is passed to the part?
Return value	What information is received from the part?

(3) Determining function outline and usage restrictions

In this phase, a document is created that defines the internal processing of a target part in detail. It is important to code return information and return values for the case when processing ends normally and abnormally. As an example of a usage restriction, a part for date conversion is explained. In this example, the date range supported for the part is clarified.

[Example of use restriction part for date conversion]

Function	After checking the date, convert the internal system date based on the Gregorian calendar into an external date based on the Gregorian calendar or the lunar calendar.
Usage restriction	The supported range of dates is 1980 to 2079.

(4) Determining internal information in the parts

Information	Description
File	What file is used?

Follow the procedures explained to determine external specifications of the parts. At this time, create a document in each phase to ensure that everybody can easily understand how to use the part.

Exercises

1. Choose the two wrong explanations from the following explanations related to program partitioning that focuses on data flow.
 - (1) Similar functions in a partitioned module are extracted as a common module.
 - (2) Input data is analyzed from the standpoints of Sequence, Selection, and Repetition.
 - (3) The program structure is analyzed based on the relationship between the maximum abstraction input point and the maximum abstraction output point.
 - (4) The program structure is created based on the relationships between components of input data and output data.
 - (5) It is used when input data can be classified according to different types of processing.
 - (6) Program processing is partitioned into three functions: Source (S), Transform (T), and Sink (S).
2. When designing the program structure, which of the following partitioning methods focuses on an analysis of the structure of input data and the number of times the same data reappears?
 - a. Transaction partitioning method
 - b. Warnier method
 - c. STS partitioning method
 - d. Jackson method
3. Select the correct sequence of the statements below for designing a program using the Jackson method.
 - a. Finding a one-to-one relationship between components of input data and output data.
 - b. Basic instructions are assigned according to the program configuration.
 - c. The program structure is created based on the relationship of components.
 - d. Input data and output data are defined in the program control structure according to the relationship of components.

Answers:

- (1) a-b-d-c
- (2) a-d-b-c
- (3) c-a-b-d
- (4) d-a-c-b

4. Which module strength represents the following module strength description?

Choose the correct answer from the list below.

This strength is possessed by a module in which two or more functions handling a single data structure are executed. These functions can be used independently. In other words, the module is a collection of two or more subordinate modules with a higher strength that are physically combined into one module.

Answers

- (1) Procedural strength
- (2) Information strength
- (3) Logical strength
- (4) Linkage strength

5. The following items a to d are descriptions of module strength. Which description represents the highest strength?
 - a. All elements in a module are linked in the sense that they execute a single function.
 - b. The elements in a module are in a time-based relationship for consecutive execution during a specific period of time.
 - c. The elements in a module are closely related in the sense that they reference the same data collection or are passing and receiving data.
 - d. Each element in the module has two or more functions and the functions are executed sequentially.

6. To create a more independent module, module coupling must be weakened. Which is the weakest in regard to the degree of coupling?
 - a. A form of coupling in case another module refers to externally declared data.
 - b. A form of coupling in case two modules pass an independent data item as a parameter.
 - c. A form of coupling in case a module passes control information as a parameter when it calls another module.
 - d. A form of coupling in case a common data area shared by several modules is referenced.

7. Fill a correct phrase(word) in the blank spaces in the following description on reusing parts.
 Parts are categorized as (1) parts and (2) parts. The former means (3) and the latter means (4), (5), and (6).

8. Fill a correct phrase(word) in the blank spaces in the following description on techniques for reusing parts.
 The methods of reusing parts are (1) and (2). The former is a technique by which an existing system is analyzed and system information is acquired during a search for similar processing to create a part. The latter is a technique by which a common section and a section assumed to be changed are identified before a part is created.

9. Describe the four components of the external specifications of parts.

10. Describe the problems arising from parts reuse in heuristic reuse.

11. Put the correct words in the blank spaces in the following description on the STS partitioning method.
 STS represents (1), (2), and (3). The STS partitioning method is a technique by which program processing is divided into three functions: (1) (input section), (2) (processing section), and (3) (output section).

12. Describe the three basic control structures used in the Jackson method.

13. Choose the two checks below that represent the validity check during the parts list review.
 - (1) Examine that a part has been segmented into the smallest possible units.
 - (2) Examine that processing in the part is appropriate.
 - (3) Examine that duplicated sections are unified into one section in a segmented part.
 - (4) Examine that the presence of a part is appropriate.

3 Creating of Program Design Document

Chapter Objectives

This chapter explains how to standardize and create the program design document.

- 3.1 Standardizing the Program Design Document
- 3.2 Description of Processing Outline
- 3.3 Detailed Database Design
- 3.4 Standard Interface between Modules

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

3.1 Standardizing the Program Design Document

3.1.1 Contents

Creating a reliable document is important to achieve the required product quality and proceed efficiently with program development.

The program design document consist of the following ten items:

(1) Front cover

Shows the system name, program name, and creation date, etc., which indicates that the document is a program design document.

(2) Outline of program functions

Contains an outline diagram showing an overall image of the program, and describes the program functions so that the entire program can be clearly understood.

The figure shows how the “Program Functions Outline” is organized.

Program functions Outline	System name	Program name	Created	Created	Page
	Payroll system	Attendance data	2002.06.	Ishii	1/1

<p>General</p> <pre> graph LR Entr{{Entr}} --> Attenda[Attenda] Exit{{Exit}} --> Attenda Attenda --> Attendanc[(Attendanc)] </pre>	File name	Physical file	Organi	I/O	Record
	Entry screen	KG0001	YYYY	I/O	
Remarks					

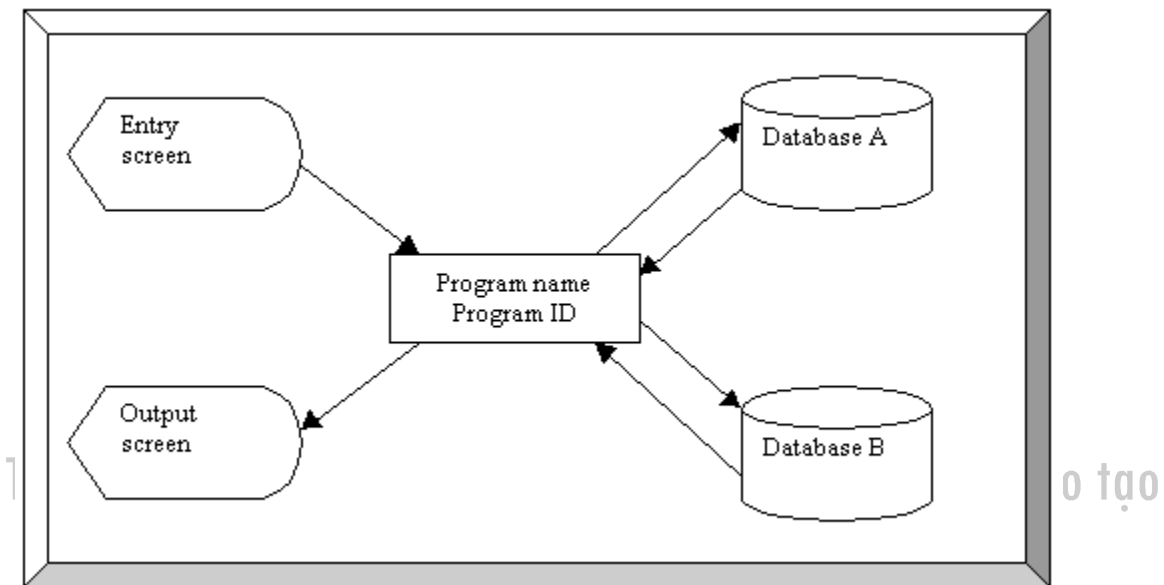
Outline of functions
1. Read data from the entry screen.

5. Exit processing

(3) Input-output outline diagram

The input-output outline diagram shows in an easy-to-understand manner the data flow for the input and output data being processed.

The figure shows how the input-output outline diagram is organized.



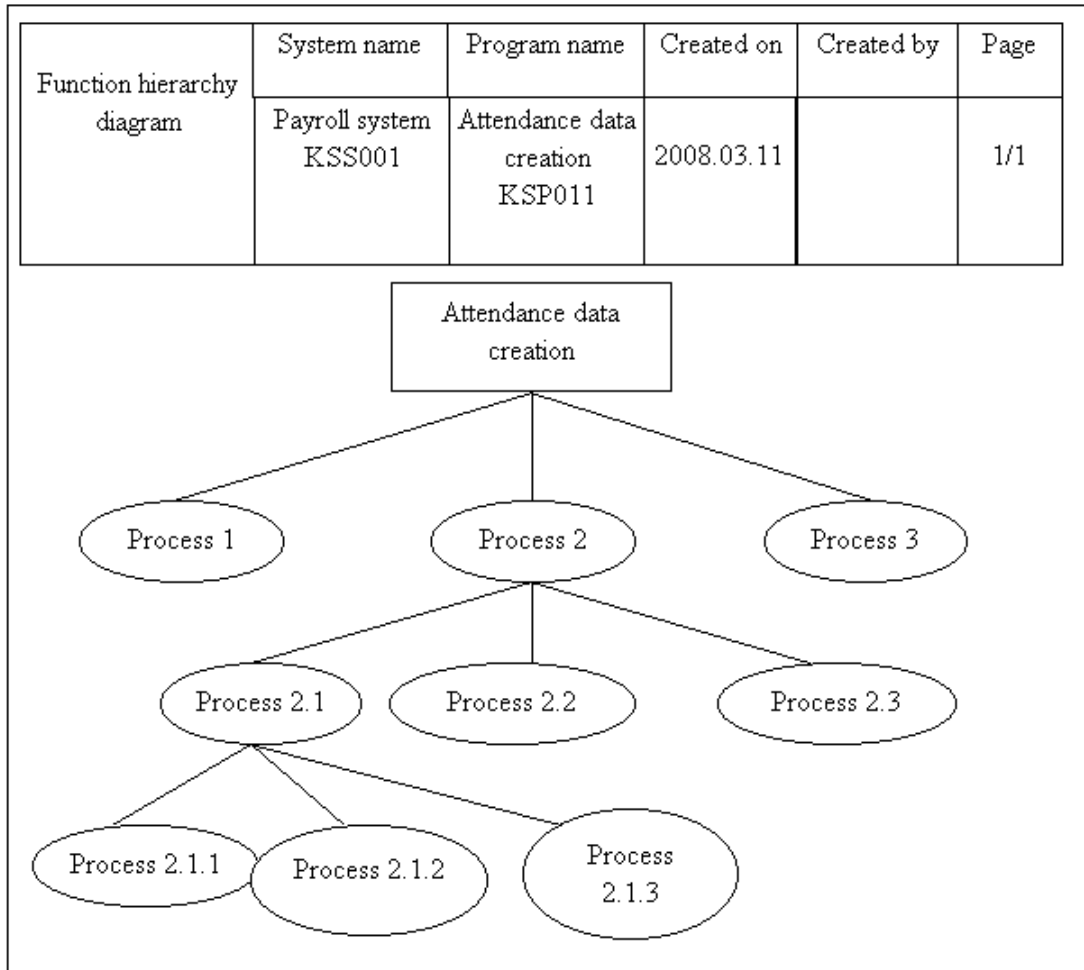
<http://www.vitec.org.vn>

(4) Function hierarchy diagram

The function hierarchy diagram is a hierarchically structured diagram of the related modules into which one program is divided.

The figure shows how the function hierarchy diagram is organized.

Function hierarchy diagram



(5) File list

The file list shows the names of various program-related items, such as the file names and file IDs. Items must be arranged in a way that the user can easily understand the file contents.

The figure shows how the file list is organized.

File list

File list	System name		Program name		Created on	Created by	Page
	Wage system KSS010		Work data creation KKPO11		9999.99.99		1/1

No	File name	File ID	Copy phrase ID Format	Organizat ion Device name	Record lengthBlock length	Size Number of items	Comment
1	Employee master	DBSN	DBSINIF	PF	350	90 KB	

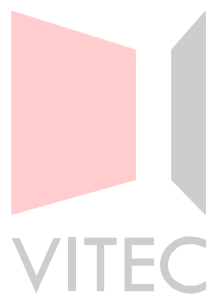
Ver	Data	Reviser	Revision content	Remarks
1	99.99.99	A	New creation	

(6) Message list

The message list is a list of messages displayed by the program. It also explains the meaning of the messages and actions to be taken in response.

The figure shows how the message list is organized.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

(7) Table list

The table list lists the name, code, and contents for each table used in the program.

The figure is an example of how the table list is organized.

Table list	System name	Program name	Created on	Created by	Page
	Payroll system KSS010	Money transfer data creation KFP002	9999.99.99		1/1

No	Table name	Table code	Explanation
1	Deposit account table	KKST010	

(8) Parts usage list

The parts usage list contains the ID, name, and creation date (to be readily identified whether a part is new or old) of the parts used by the program.

The figure shows how the parts list is organized.

Parts list	System name	Program name	Created on	Created by	Page
	Payroll system KSS010	Management file creation KKF011	9999.99.99		1/1

No	Parts ID	Parts name	Created on	Remarks
1	KBP 001	Time calculation	2005.03.11	

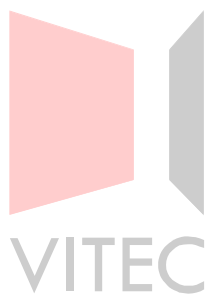
Remarks

(9) Database definition

The database definition lists the items, item attributes, and sizes for each table. In some cases, it can also contain a database table list and a database diagram that indicates the relationship between tables.

The figure shows how the database definition is organized.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

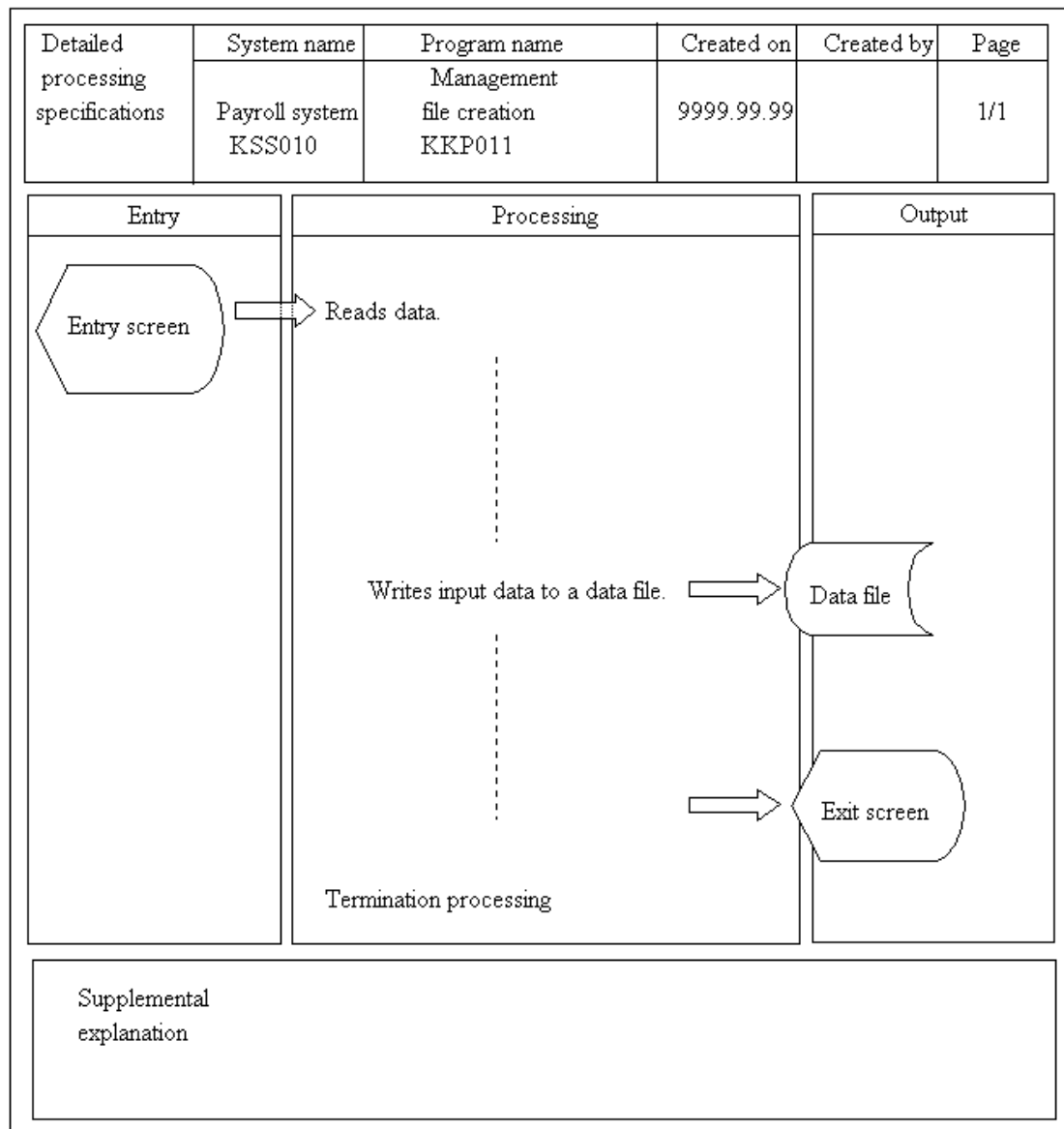


<http://www.vitec.org.vn>

(10) Detailed processing specifications

The detailed processing specifications describe the contents of processing for each program in detail.

The figure shows how the detailed processing specifications are described.



Items 1) to 4) below explain the items in the detailed processing specifications.

1) Entry section

In the entry section, describe data entry source by drawing figures, so that the user can easily visualize it. The data entry source can be a file, a screen, a table in main memory, or the system save area.

2) Processing section

In the processing, it is important to itemize the input and processing, or the relationship between the processing and output) in a concise manner so that one can exactly understand them. Enter the detailed contents in the supplemental explanation section, or attach additional materials, if necessary.

To express data logically, use pseudo-language. When the relationship between conditions and the contents of execution are complicated, use a decision table, which is explained later, to clearly indicate the relationship between conditions and actions.

- Pseudo-code

In order to make the program easily understandable, the program should be described in a way that is close to ordinary sentences (in a natural language). To focus only on the solution for a specific problem or the logical structure, remove restrictive (grammatical) elements of the programming language. This facilitates problem solving.

Pseudo-code, or program description language, refers to writing a program in a form that is close to a natural language, so that programmers can use the result as a “thinking tool” to help them with programming.

- Necessity of pseudo-code

To have a computer execute a process, the process must be described in a programming language. However, programming languages have many syntactical (grammatical) restrictions. Therefore, it can be difficult to use the code written in a programming language as a tool for thinking in the programming process. Therefore, flowcharts and comparable elements are usually used in program design documents. However, while flowcharts are appropriate for expressing “how” processing is performed, they are not necessarily appropriate for expressing “what” is to be performed. In other words, when flowcharts are used as a thinking tool, the programmer will know the object of processing “what” while the programming stage is still in progress; however, the programmer is likely to forget about the “what” eventually, after programming has been completed, and flowcharts which indicate “how” and program code will only remain later on.

A programmer who will be in charge of maintenance later on does not know what is performed in the program, and program corrections will take a lot of time.

To avoid this problem, use pseudo-language as a thinking tool in the programming phase. One advantage of the pseudo-code is that it simultaneously expresses “what” and “how” as by-products for advancing the thinking required during programming, and save statements written in a pseudo-code as a document.

- Using pseudo-code

Pseudo-code is a programmer’s thinking tool and has the purpose of representing

designed information. In view of the purpose, a program may first be written with sentences in the programmer's tongue, disregarding formal syntactical rules.

To execute the program, however, the programmer's language must later be converted to a specific programming language. pseudo-code is not a compiler language, and cannot be processed by a computer without being converted to a programming language. Consequently, considering whether pseudo-code can easily be converted to the various programming languages, one finds that certain rules are required.

3) Output section

In the output section, enter the data destination in drawing.

After a module is partitioned into modules, input-output processing may be performed by subordinate modules. The input and output sections for the main module in the detailed processing specifications may therefore be blank.

4) Supplemental explanation

The supplemental explanation section is available to supplement the input and output sections.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

3.1.2 Modification criteria for maintenance and improvement

(1) Specifications change report

The specifications change report is used to document the modifications when the specifications are revised. In this report, write the contents of modifications, reason and cause for the modification, when the revised was requested, how the change affected, and the required time and costs for modifications in detail.

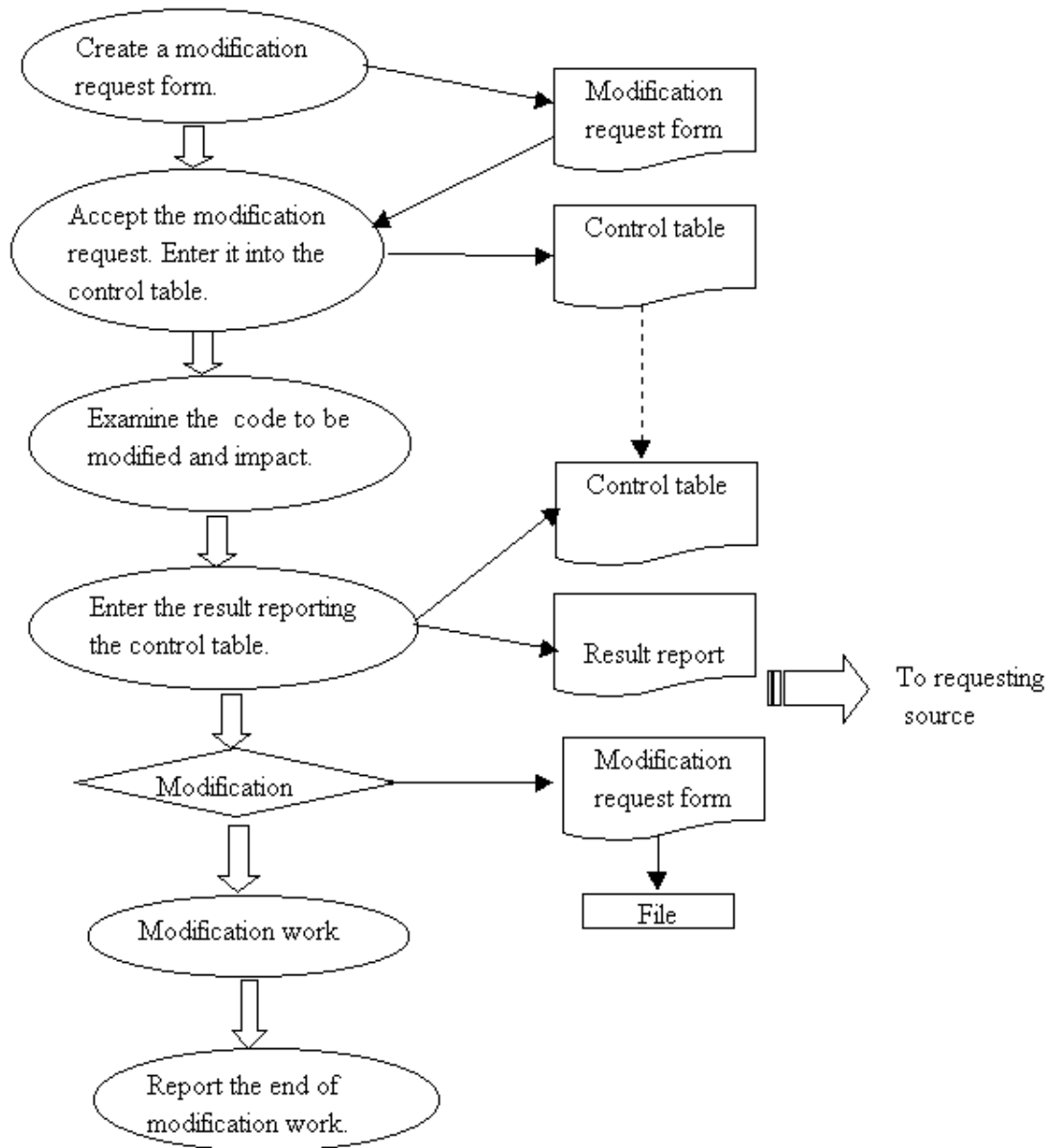
The figure below shows an example of how the specifications report is organized.

Request No.	System name	Requesting department	Person in charge	Requested on
MH-0024-1	MHID 002			9999.99.99
Requested item Insertion of XXX...				
Modified contents Add item xxxxxxxx newly				
Reason or cause				
Effect by modification	Effect level :	Number of required days:	Estimated cost :	
Modification of program		Modification of database		
Check result	Reason	Approved by	Person in charge	

(2) Change approval procedure

The change approval procedure must be documented to prevent inconsistencies between the modification requesting and requested parties.

The figure shows the procedure to be followed when the specifications are modified.



(3) Repository modification procedure

The repository, a software database for registering all information related to system development, is created in the software development process.

The repository will be used for different tasks and by multiple users. If incorrect information is entered into the repository, all subsequent users referencing the information later on will receive the incorrect information.

To allow modifications of a repository, predefine modification criteria and determine the modification policies.

3.1.3 Contents of the user's manual

The user's manual describes how to specify a common program to be called from another module, and explains the contents of the program in an easy-to-understand form. It also explains parameters in detail.

The figure below shows an example of how the user's manual is organized.

Usage	System name	Module name	Created on	Created by	Page
	Payroll system KSS010	Management file creation MID020	9999.99.99		1/1
<u>Format</u> CALL "MID020" USING parameter 1, parameter 2					
<u>Usage</u> 1. MID020: Module name 2. parameter 1: Variable name of data input area 3. parameter 2: Variable name of data output area 4. Return code: Variable of return code 5. Use example: CALL "MTD020" USING PMT1, PMT2					

3.2 Description of Processing Outline

The following methods for providing a processing outline are available.

(1) Matrix representation

a. Decision table

Decision tables are used to combine complicated conditions and processing. They can be classified into four types of matrices. The first one in the upper left (1 to 4 in the figure) is for describing the contents of all conditions that can occur. In the right section, Y is entered when the condition is satisfied and N when the condition is not satisfied. In the lower left (A1 and A2 in the figure below), the contents of processing are described, and in the right, an N or a Y is entered to indicate whether the processing is to be executed or not. With this approach, the relationships between all conditions basically appear as permutation.

Decision table

Conditions	Case					
	a	b	c	d	e	f
1	N	N	Y	Y	N	Y
2	Y	N	N	Y	N	Y
3	N	N	N	Y	Y	Y
4	N	Y	Y	N	N	Y
Processing	a	b	c	d	e	f
A1	N	N	-	Y	N	Y
A2	Y	Y	-	N	Y	Y

b. Relation table

The relation tables are used to represent the specific data to be managed by attribute types. The table is arranged in such a way that a couple of data items to be managed is represented as a row and that the same kinds of data items are represented as a column. The order of rows is not restricted, but the same value must not be specified on multiple rows.

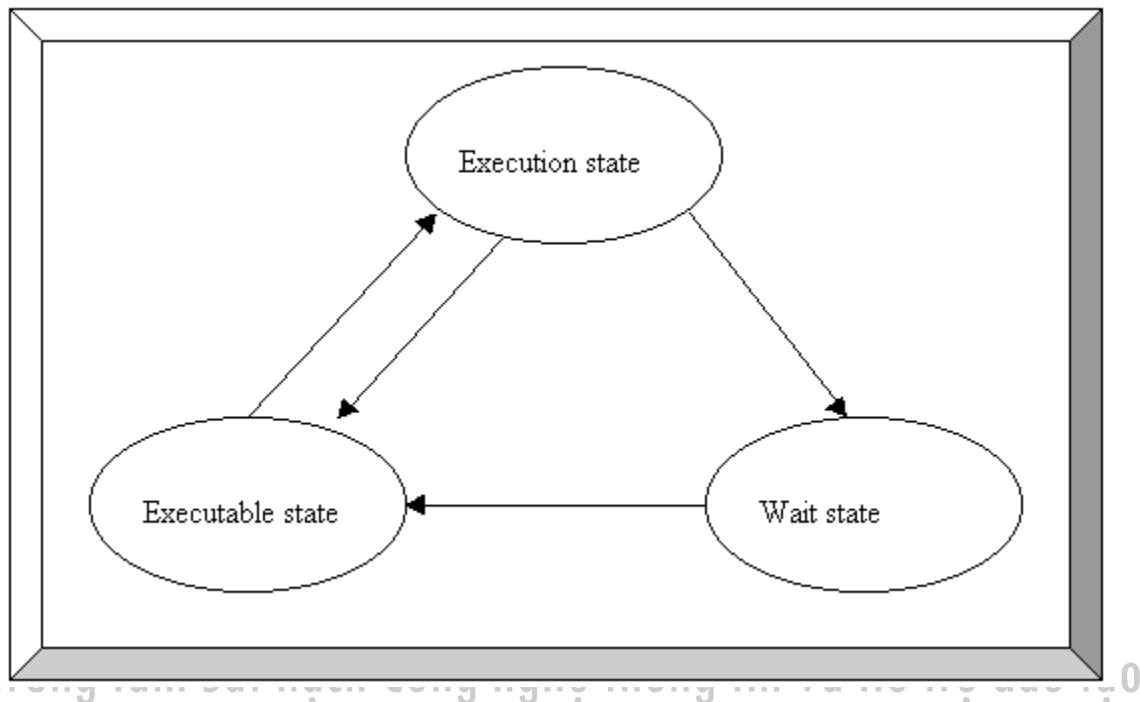
Employee code	Name	Position code	Bonus
1	Full name a	B	300,000
2	Full name b	C	200,000
3	Full name c	E	150,000
4	Full name d	D	100,000
5	Full name e	E	50,000

Relation table

(2) Graphic representation

a. State transition diagram

The state transition diagram is used to indicate input information, transitions between states, and the relationship between contents when the state changes as a time elapses or the contents of input data change.




State transition diagram

b. Data flow diagram (DFD)

The data flow diagram consists of figures and tables used to analyze functions for input, processing, storage, and output of data in a data flow model.

The diagram details processing on a step by step basis, as in the case of a hierarchical function model, instead of giving the full detail of a job, starting from the first step. In other words, it repeats the process from a first partial analysis through a rough data flow diagram, and gradually proceeds to a more detailed data flow diagram. In this case, it is important to draw a rough flow of job processing or the processing system in a form that is easy to understand for the customer.

The figure below shows an example of notation in the data flow diagram.

<div style="border: 1px solid black; padding: 2px; display: inline-block;">Name</div>	Data source/sink
	Data flow
<div style="border-bottom: 1px solid black; padding: 2px; display: inline-block;">Name</div>	Data file
<div style="border: 1px solid black; border-radius: 50%; padding: 5px; display: inline-block;">Name</div>	Data process

Data flow diagram

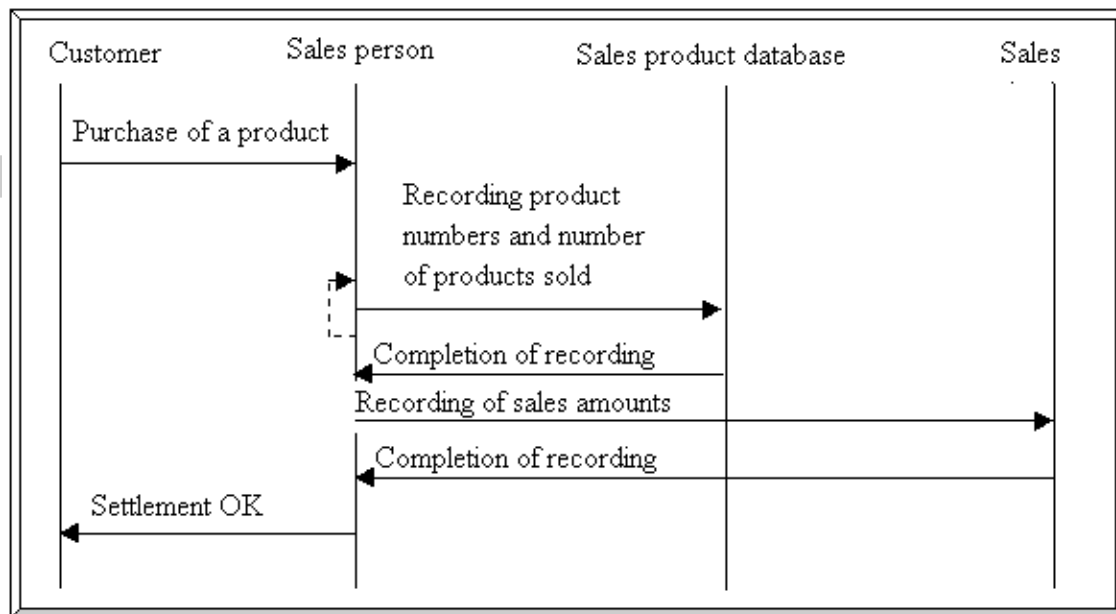
c. Event trace diagram

The event trace diagram shows when an event occurs on the basis of a data flow diagram and system components. This diagram indicates process timings by arrows in the direction in which events are caused. This diagram also uses arrows to show repetitive processing steps.

This diagram is used as an auxiliary drawing that indicates the timing of state transitions.

The figure shows an example of how the event trace diagram is organized.

Event trace diagram



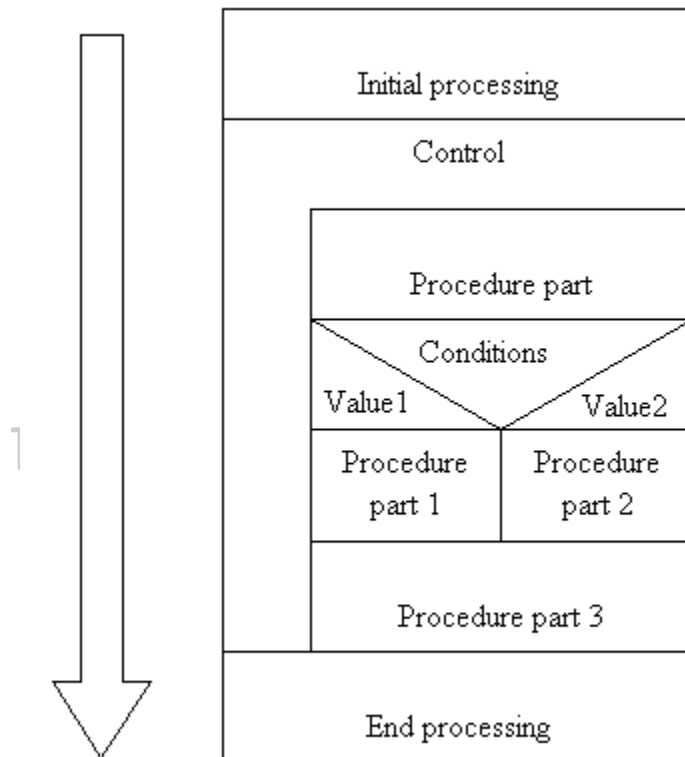
<http://www.vitec.org.vn>

(3) Area representation

a. NS chart

The NS chart is used to represent a processing procedure for a structured program module. In this chart, all processes are indicated with rectangles.

Process flow



Example of NS chart

VITEC

<http://www.vitec.org.vn>

g tin và Hỗ trợ đào tạo

3.3 Detailed Database Design

3.3.1 Clarifying database input-output conditions (Deadlock and rollback)

(1) Deadlock

a. Deadlock occurrence conditions

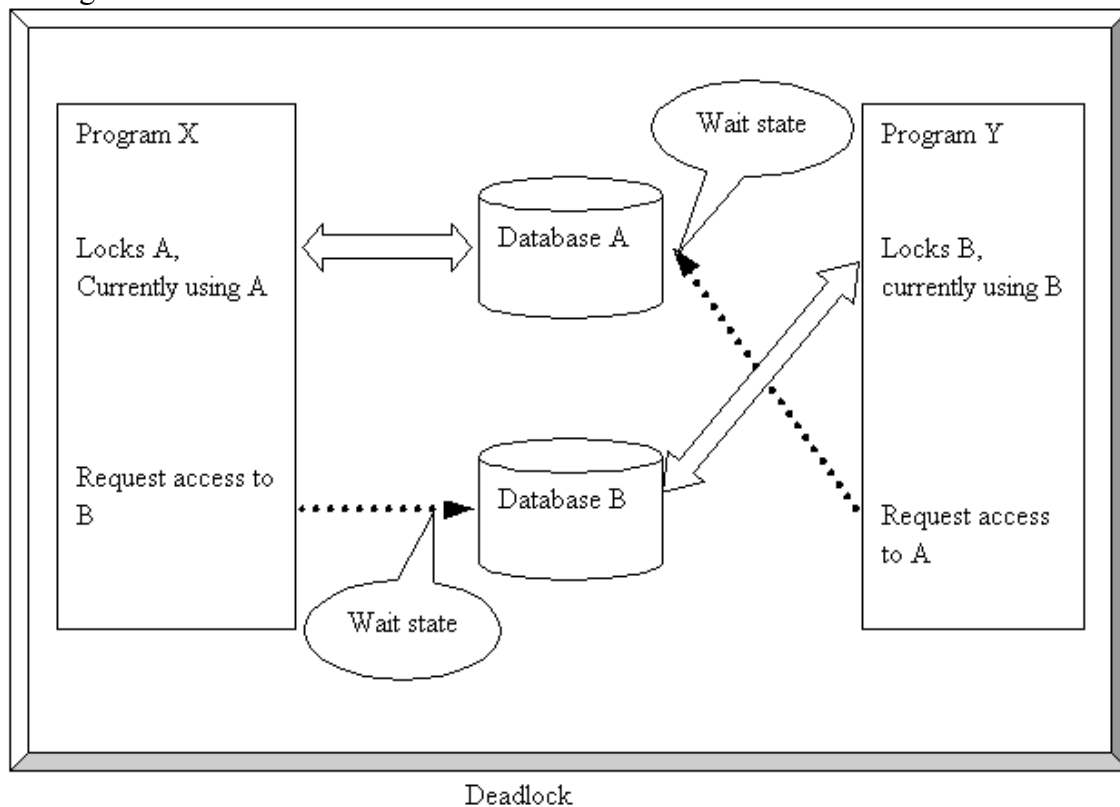
-Exclusive processing

When multiple users share a database, there are cases where it would be inconvenient if other users were using the currently processed data at the same time.

Many systems provide a function that can be used to lock a database, so that other users cannot use the currently processed data until the on going process ends. This function is called “exclusive control”.

Such an exclusive control function is installed in most current database systems. However, use of this function can result in cases where multiple users wait for the release of locked data, and the users is unable to proceed with processing until the data is released. This state is called deadlock.

The figure illustrates the deadlock state.



The meaning of this figure is as follows:

- 1) Program X first locks database A before using it. Program Y locks database B before using it.
- 2) After processing database A, program X attempts to access database B, but because database B is locked by program Y, program X must wait until the lock is released.
- 3) After processing of database B terminates, program Y also attempts to access database A, but because database A is locked by program X, program Y must wait until the lock is released.

In this manner, programs X and Y keep waiting for each other. A state in which processing is suspended as shown is called deadlock.

b. Avoiding deadlock

To prevent deadlock, the program must be designed to ensure that multiple data items are not locked at the same time, when exclusive processing is performed for a database.

Three methods for avoiding deadlock are available as follows:

1) Copying a database

The system provides a backup file to which the database is copied in preparation for failures and a journal file that contains transaction records (which cause database updates during job processing) as well as the contents of the database before and after the update. These two files can be used to return the database to its state just before a failure occurs.

The processing for recovering a database to the latest state as explained is called roll forward processing.

2) Splitting storage media

If a failure occurs during database updates, the update of records fails, which results in loss of data consistency.

When a failure occurs, one must determine in which record of which database a loss of consistency occurred.

When splitting storage media, the contents before the update, contents of transactions, and the contents after the update are stored on external storage media before updating a database.

3) Establishing rules and keeping rules

-Defining the access order in the first place

With this method, each program performs exclusive processing for a shared database in the same sequence. For example, if there are four sharable databases: A, B, C, and D, each program always hold those databases in the order of A-B-C-D.

-Rule for using character strings

The queuing must be performed using the same character string. Character strings must be used in the same order.

-Two-phase lock method

When there are multiple programs and multiple sharable databases, exclusive processing is performed in two phases: reservation and release of a sharable database. In this method, each program reserves required databases; however, after even one database is released, it does not reserve a new one.

c. Detecting deadlock

Deadlock is detected by monitoring whether each program is in a wait state in which each program is waiting for the release of a database.

Process control determines that a deadlock has occurred when the execution time exceeds the predetermined time.

(2) Rollback

In a system that includes many hardware and software products, failures can occur for unexpected reasons. If any failure occurs, its cause must be resolved quickly, programs and data must be recovered, and the process must be executed again.

a. Journal

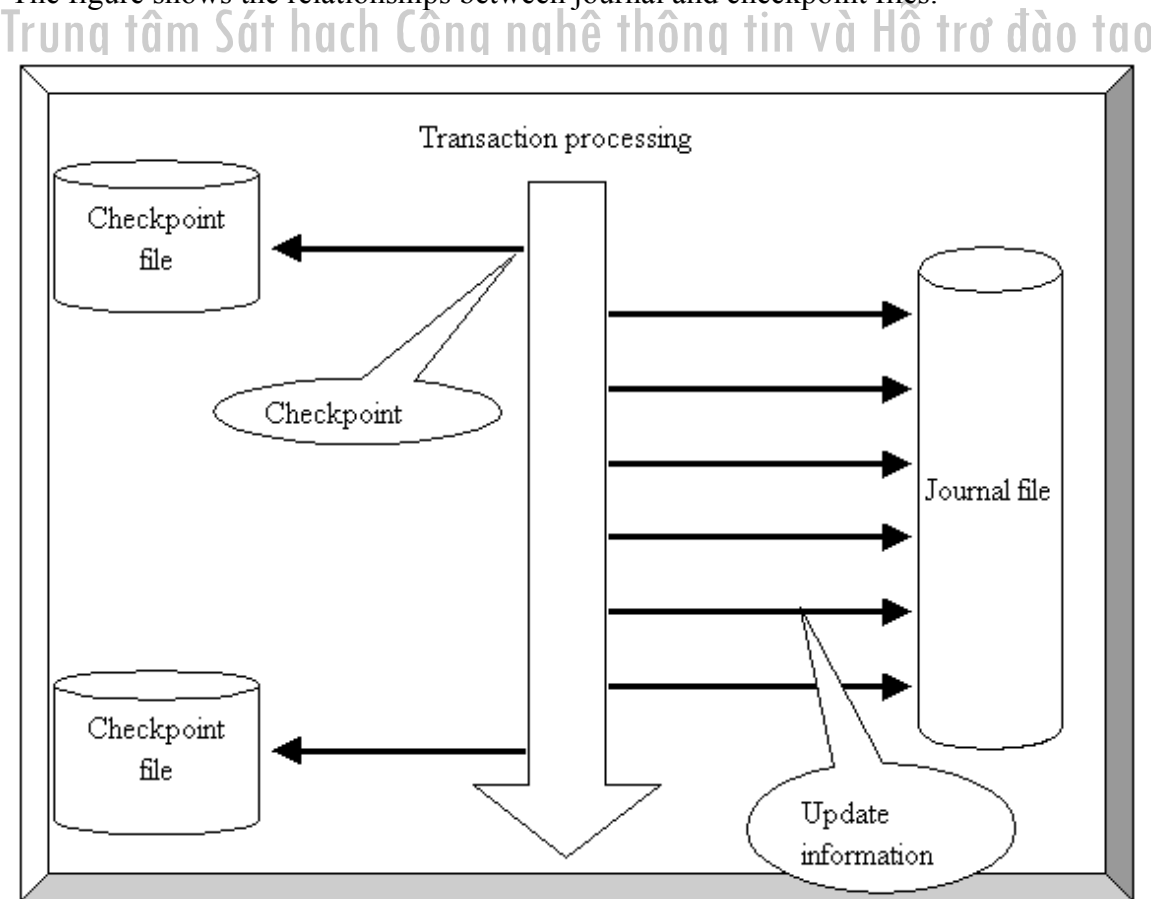
Information file to update a database is called a transaction. A file that stores only update information, that is, what update processing is performed in the transaction, is called a journal file.

The journal file contains the contents of a record before and after the update as is; therefore, this file can be used to recover a database.

b. Checkpoint

A checkpoint is used in a case where there is a large amount of transaction data. The state of data processing in a database up to a checkpoint is recorded in a checkpoint file. Unlike the journal file, the checkpoint file is used for backups up to the last checkpoint.

The figure shows the relationships between journal and checkpoint files.



c. Commit instruction

Commit processing means that processing terminates as soon as all databases have been updated by transactions. When a program terminates abnormally, the results of operations performed until the commit instruction was executed are not discarded by rollback processing.

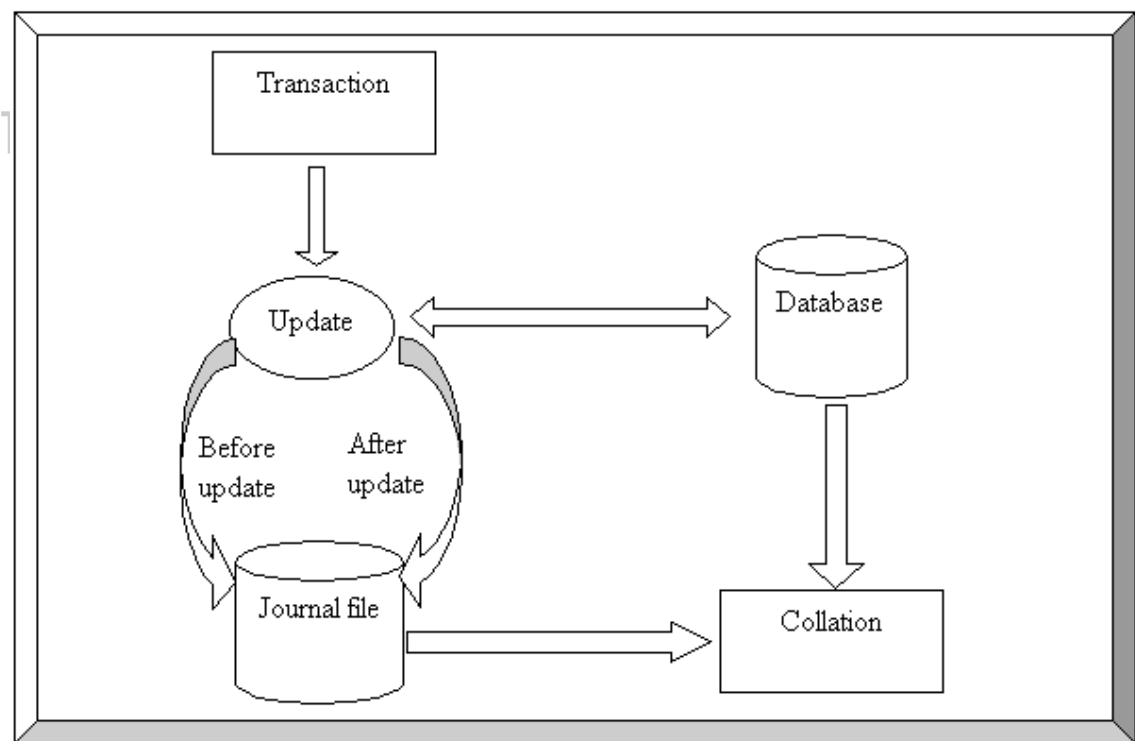
d. Rollback processing

If a failure occurs during database updates, records are not updated, or data consistency may be lost.

To recover data consistency, rollback processing may be used to restore the state of database before the update.

- 1) As previously described, the contents of a database before and after the update are stored in a journal file.
- 2) The system reads the journal file when a failure occurs, and collates the record contents of the database.
- 3) The system determines whether the failure occurred before or after the database update, and performs recovery processing.

The processing for returning a database to the state before a failure occurred is called rollback processing. The figure illustrates rollback processing.



Rollback processing

3.3.2 Physical database design

(1) Designing the storage structure

1) Record organization

- Use the result of data normalization to determine record types and data items. For each data item, specify a data type (numeric value or alphabetic character) and the data length for storage.
- Determine the record layout for data items in a record, and allocate sufficient space to allow the addition of a new item without problems.
- Specify whether the record shall be of fixed length or of variable length.

2) Determining a physical database

- Examine the relationships between records from the result of data normalization those how the data characteristics are related.
- Examine the record access order, and determine the record storage order.

3) Organizing a database

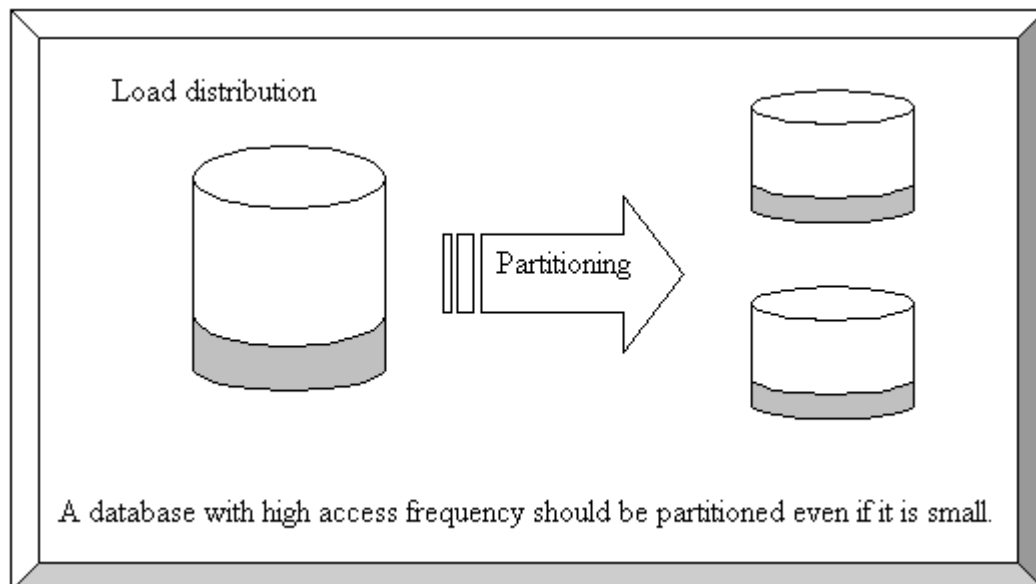
- Determine a selectable database organization, considering various characteristics such as data characteristics, use characteristics, job characteristics, required resource characteristics, and DBMS characteristics.

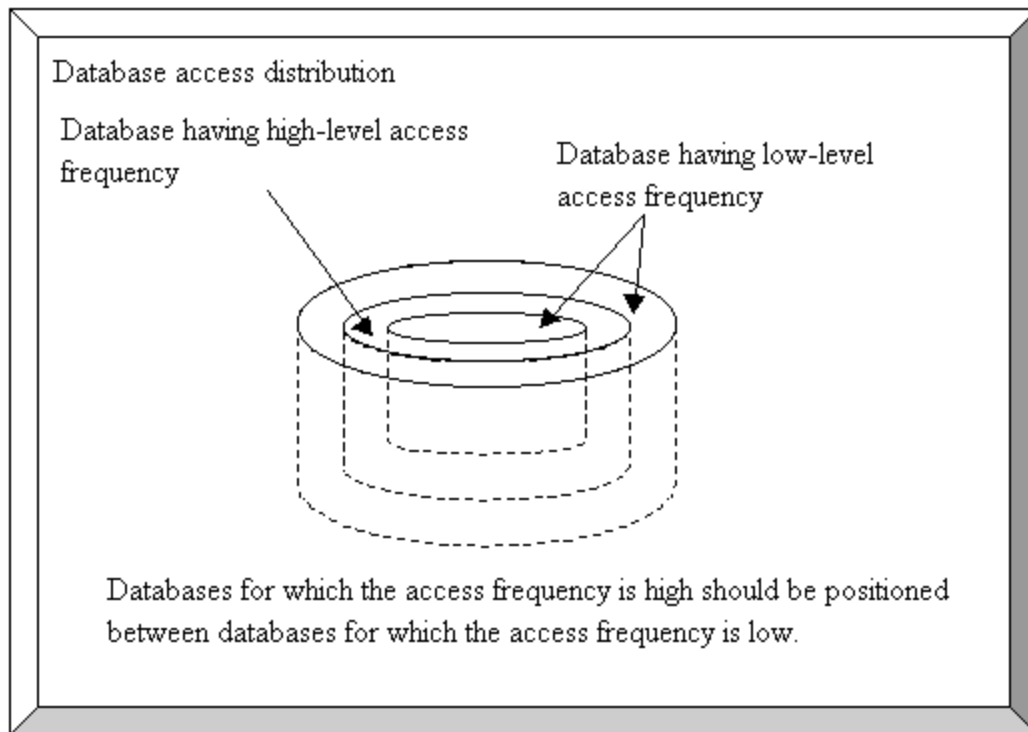
4) Allocation in databases

- Estimate the number of records for each database, considering the amount of data to be stored from database installation up to n years in the future.
- Examine the record length, block length, organization method, unit type, and so on, and estimate the required file size.
- Estimate a suitable overflow area for each unit, considering how often the database must be reorganized and the rate of data increase.

In other words, allocation in databases only in accordance with the physical requirements is insufficient. The points in the following also need to be considered:

- For the physical allocation of the database, consider the load distribution based on the amount of data and the access frequency, as well as the physical accesses of the database in accordance with the job contents and processing mode.
- Where the database is to be allocated in storage depends on how many times the database is accessed. Allocate databases for which the access frequency is high between databases for which the access frequency is low.





(2) Index design

1) Index

The index consists of data values used to search for records and addresses used to indicate record positions in a file. These items of information are stored in a certain order. An index is used to efficiently access a specific record. The ratio between the number of stored records and the number of indexes is called index density. When the number of records is equal to the number of indexes, the density level is 1. As the number of records becomes greater than the number of indexes, the index density becomes higher.

<http://www.vitec.org.vn>

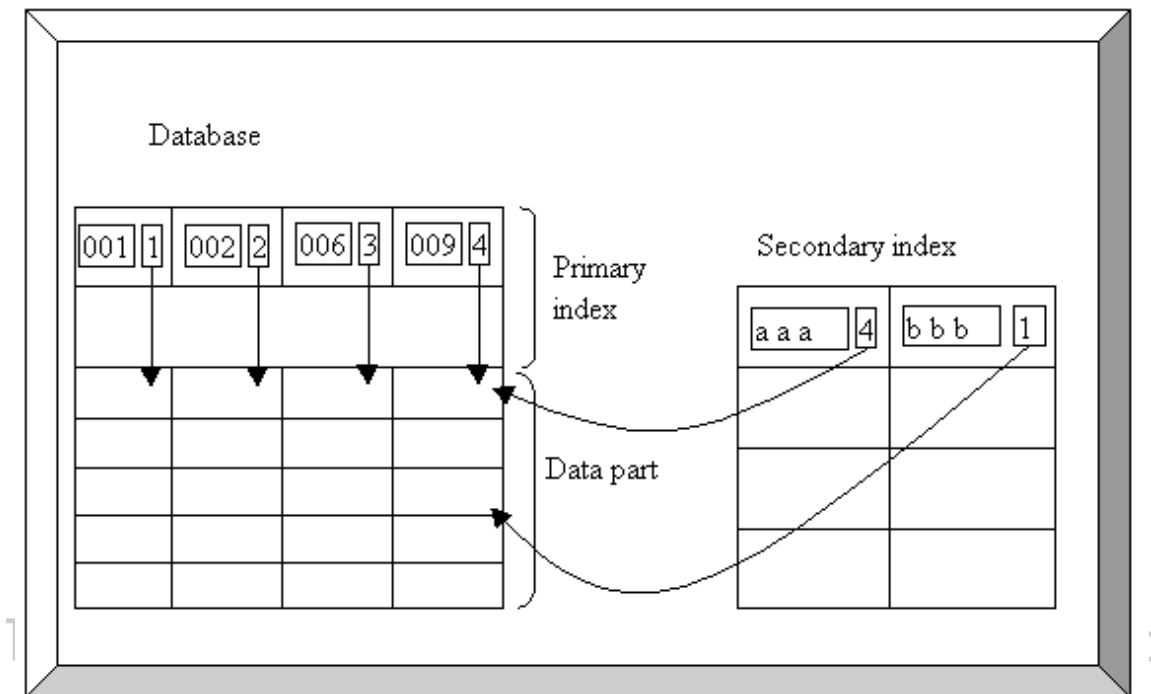
Although index structure varies depending on the DBMS used, with higher index density, more indexes can be stored.

2) Primary index and Secondary

Indexes can be divided into two types: 1) primary indexes, which are used to search for a record position with the value of the primary key and 2) secondary indexes, which are used to search for a record position with the value of a data item other than the primary key.

The structure of the primary index is determined by a DBMS type, and the value of the primary key must be unique in the file. For the secondary key, multiple records may have the same value; consequently, any data item in a file can be used as a secondary key. However, if a new record is added, index records must be added in the secondary in accordance with the number of data items, which increases processing time significantly. Therefore, the index must be designed considering the requirements for the individual job.

The figure shows the relationships between the primary index and secondary.



3) Indexing

Databases provide a reverse-list method that contain all the record addresses corresponding to the values of a specific data item as secondary indexes.

Direct addressing is an organization method for shortening the access time by using the address of a magnetic disc unit. However, when the direct addressing method is used, any change in a storage location requires that the index address related to that location also be changed. For this reason, maintenance will take more time, and the time for database reorganization increases considerably.

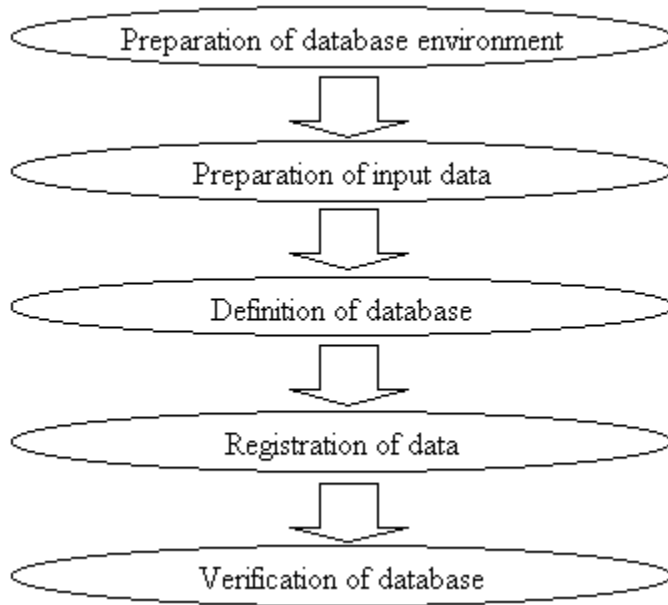
Indirect addressing is a method of using all the primary keys of records corresponding to the values of a specific data item as secondary indexes. When using this method, secondary indexes need not be changed even if a storage location changes. On the other hand, accesses take more time than with direct addressing.

In addition, some databases support the chained address method, where the address of a record is stored as a data item of the previous record, and accesses can be made using only the address of the first record in a record group. Using this method is advantageous in that the required index space is smaller than for other methods. On the other hand, fetching a record requires long time.

The indexing method must be selected depending on which criteria for processing is the most important.

3.3.3 Database creation

The procedure for creating a database is as follows.



(1) Preparation of database environment

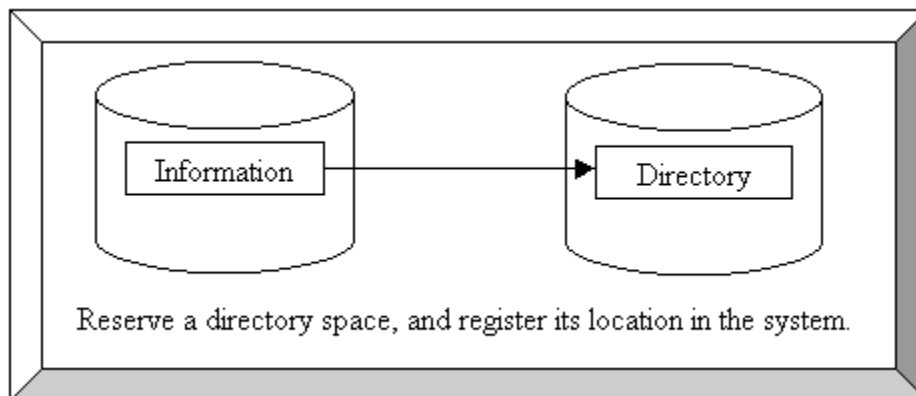
1) Preparation for operating the database management system

To operate the database management system, first install the database management system (DBMS), and register the DBMS program in the system file of the computer. In some cases, it may be necessary to create a database in accordance with the configuration and operating environment of the computer system. (This depends on the DBMS.) Whether databases can be used easily depends on how they are created. When creating a database, sufficiently prepare the database environment in advance.

A DBMS uses different information items for processing. Therefore, sufficient file space for storing the required information must be reserved and the information must be registered in the system.

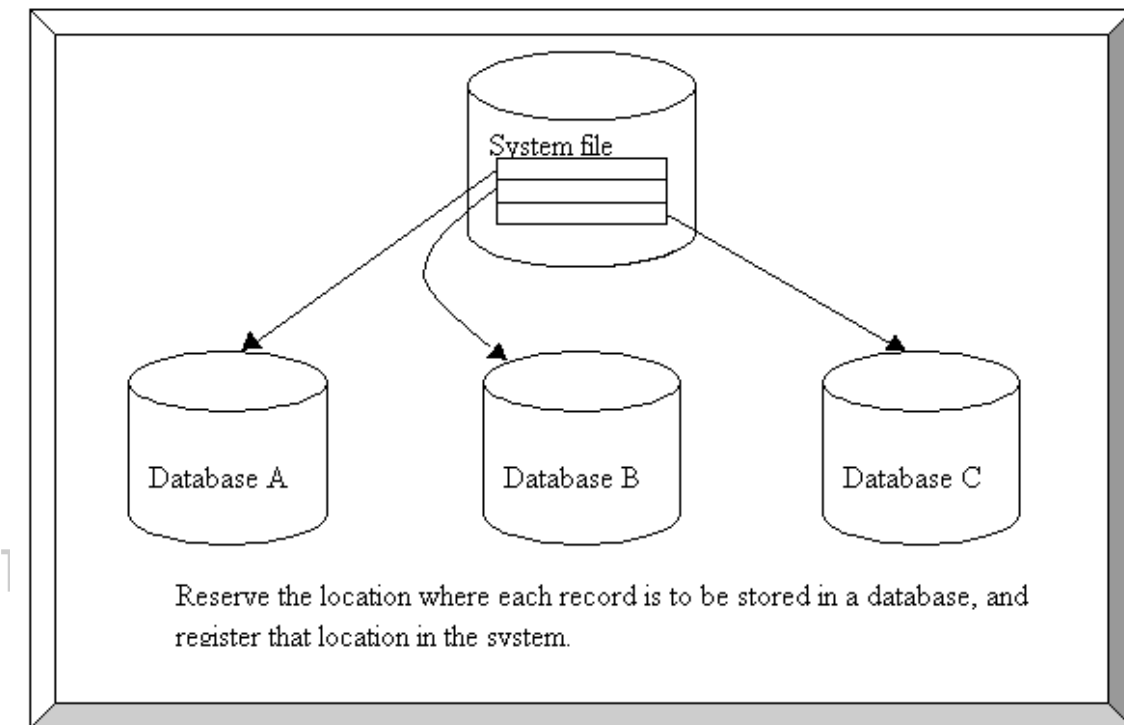
2) Preparation for database definition

To make the database definitions, reserve a directory for storing the required information, and register the storage location in the system.



3) Reserving the database area

Reserving the database area means reserving the location for storing each actual record in a database and registering that location in the system.



(2) Preparations for input data

1) Converting input data

Depending on the form in which input data is stored, conversion of input data in accordance with the record format of the database may be necessary.

a. Creating new input data

There are a number of data entry methods; for example, 1) direct data entry in online real-time processing, 2) entry via an OCR, or 3) entering data via floppy disk. Irrespective of the method, one must prepare measures to address problems, such as by determining when data is to be collected and how data must be handled until it is incorporated into the database.

b. Checking input data

Several methods of checking input data are in use: for example, 1) checking at data entry, 2) checking based on the relationships with other data items, and 3) checking based on the relationships with records in a database. Irrespective of the selected method, it is very important to execute a check of input data.

c. Converting an old file to a sequential file

Convert an old file into a sequential file by using a utility program. If the original data for the database is not computerized, newly create the data.

d. Creating a load file

Create a file for loading as suitable for the record structure and for record format of the new database. In this case, check the relationships with new data and records in the old master file.

e. Correcting the load file

Correct the load file created in the preceding step on the basis of errors detected by using the relationship check function. After this, create a checklist, and verify the data.

f. Creating corrected data

Enter the corrected data in the checklist, and create a file with corrected data.

2) Classifying input data

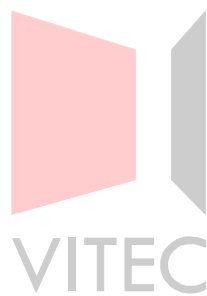
At file creation, classify input data in the order of the primary keys included in the database to be created newly.

(3) Making database definitions

1) Defining a relational database

Store database definition information in a directory, using the data definition language to describe that information.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

< Employee>

Data item name	Data type	Length	Index
Employee ID	X	5	1*
Full name	N	20	
Zip code	X	8	
Address	N	40	
Telephone No.	X	12	
Sex	N	1	
Education	N	15	
Graduated on	X	6	
Hired on	X	8	
Department	N	10	
Position	N	5	
Years of employment	X	2	
Monthly salary	X	6	
Qualification	N	15	
Number of absence days	X	3	
Number of late days	X	3	
Number of remaining paid holidays	X	2	
Registered on	X	8	
Updated on	X	8	

* The number in the index column indicates the priority level of the primary key.

2) Defining a table

To define a table, use the CREATE TABLE statement.

The CREATE TABLE statement is based on a table to be defined. For definition statements to be created, the specified data type and specifiable contents vary with the DBMS used.

2) Table existence constraints

The table existence constraints mean restrictive conditions related to data addition and deletion. For example, a parent record must exist in order that a child record can exist.

4) Table reference constraints

The table reference constraints mean compatibility restrictions depending on the mutual relationships between databases.

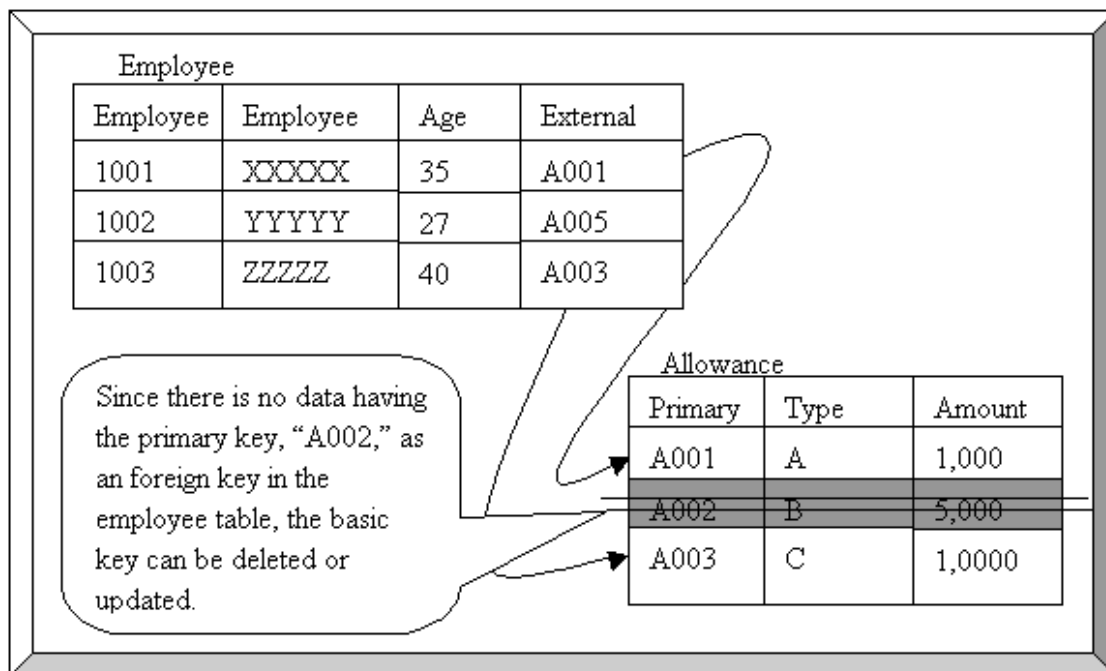
The primary key, which is also called a basic key, is used to uniquely identify a row in a specific database. The foreign key is used to link a specific row in a database to the one in another database. In this case, the value of the foreign key is the same as that of the primary key in the corresponding row of another database.

In this manner, data items have primary and foreign keys. The following rules apply to those keys:

- The foreign key is a basic key for records in the table.
- To change or delete a basic key, table reference restrictions must be defined in such a way that one of the operations listed in the following can be performed automatically.

-RESTRICT operation

The basic key can be deleted or updated only when the specified record is not used as an foreign key. The figure shows an example of the RESTRICT operation.

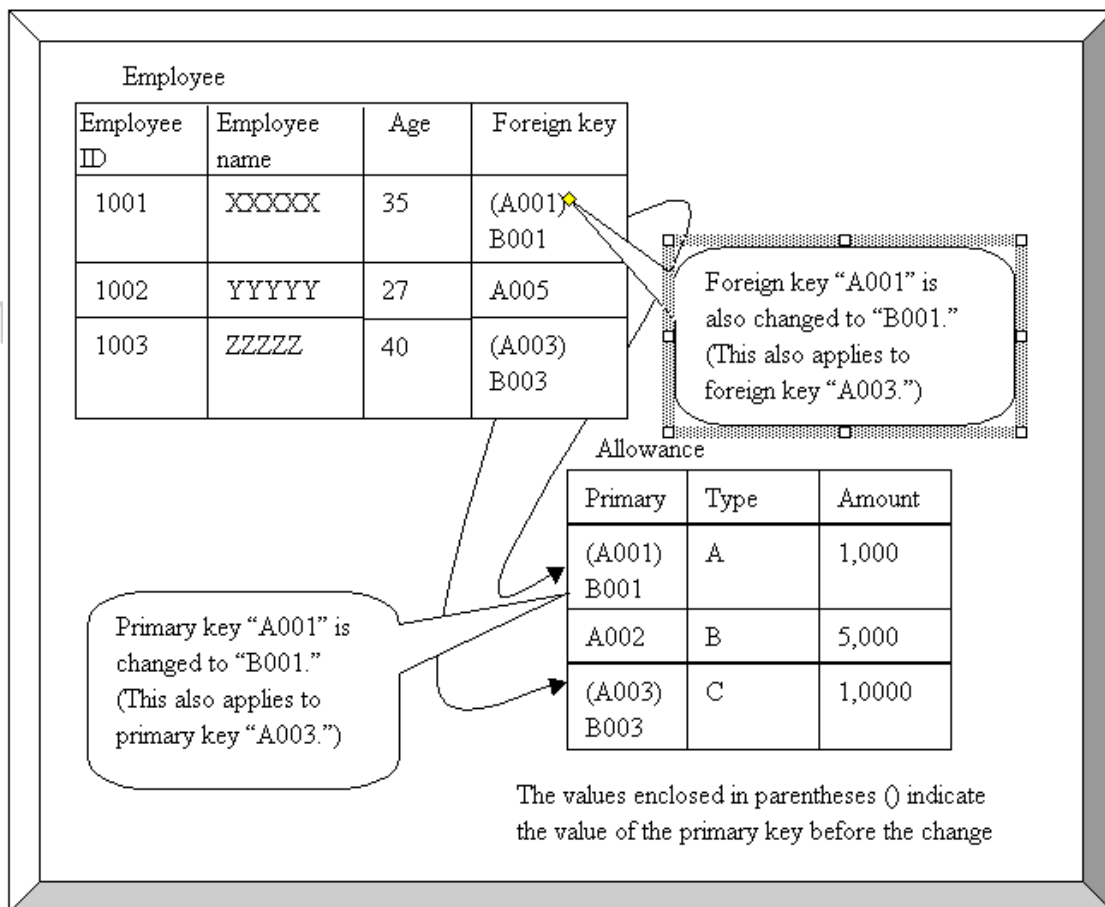


RESTRICT operation

In the previous figure, there is no record with primary key “A002” in the salary table as an foreign key in the employee table. Therefore, it is possible to delete or update the record having primary key “A002” in the salary table.

-CASCADE operation

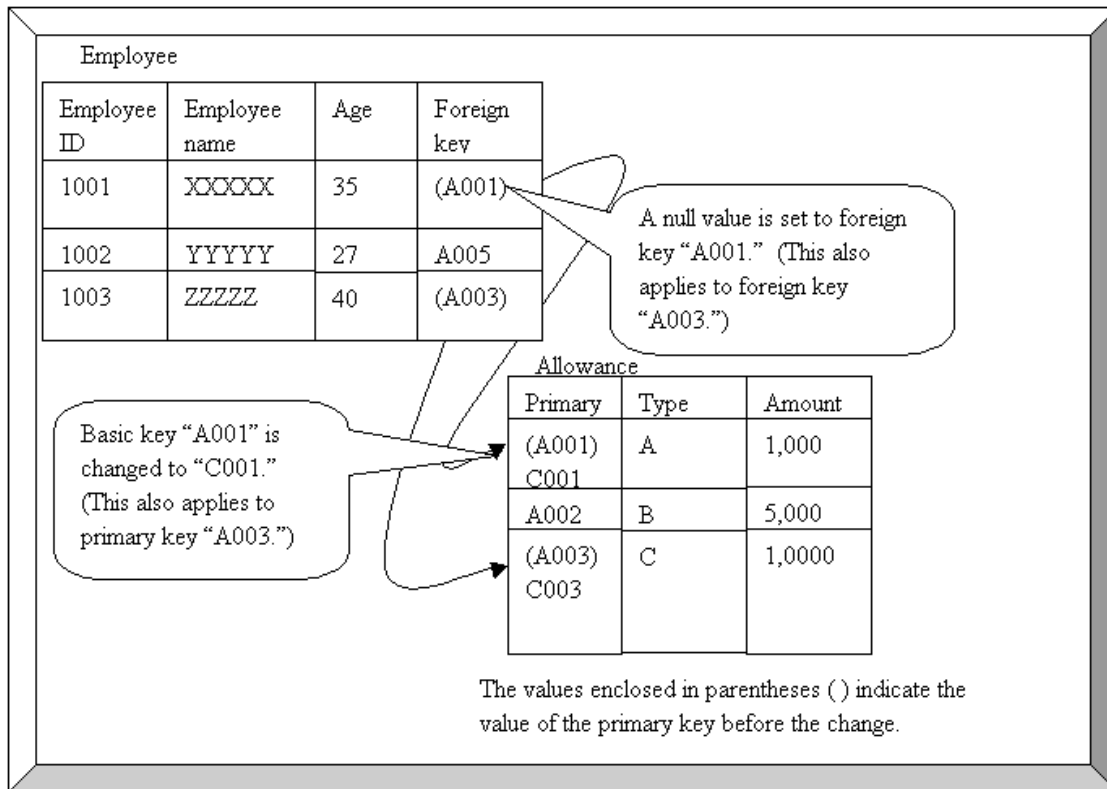
Performing the CASCADE operation deletes or updates the range from the basic key to the related external key. The figure is an example of the CASCADE operation.



CASCADE operation

-SET NULL operation

Only the basic key is deleted or updated, and the related external key is set to null. The figure is an example of the SET NULL operation.



SET NULL operation

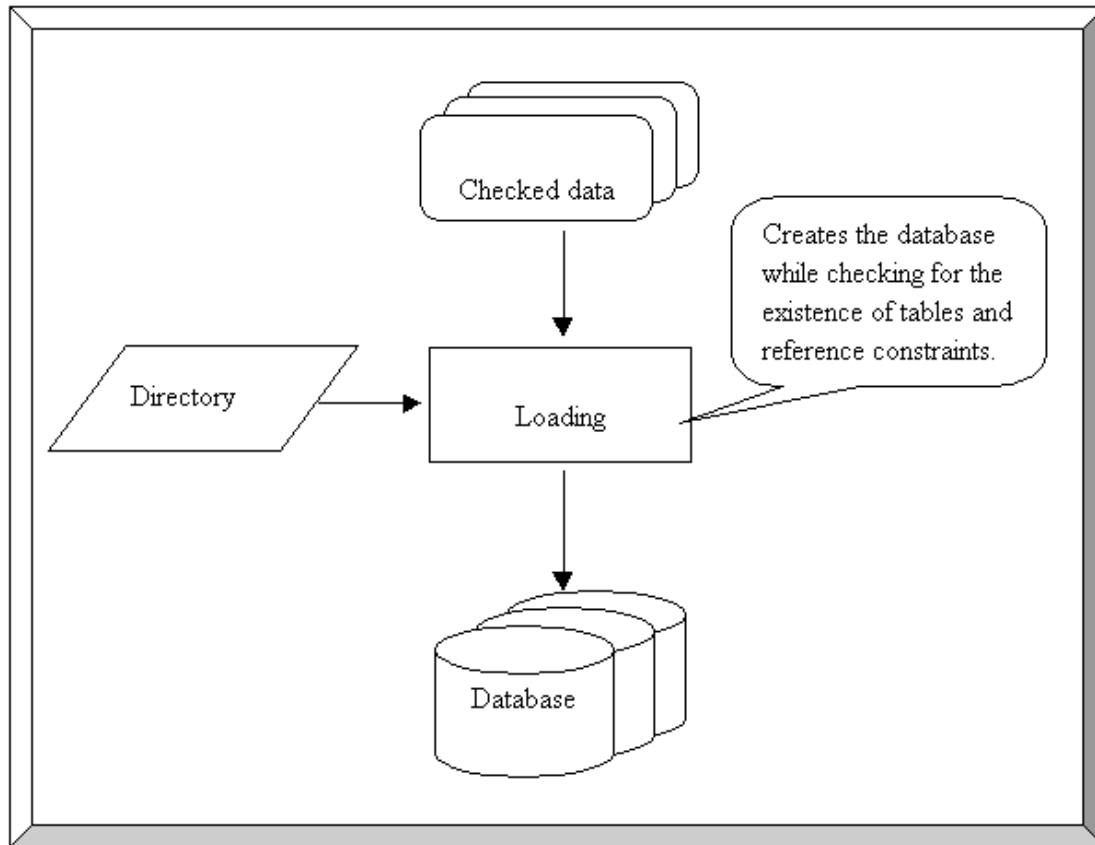
(4) Registering data

Defining data with SQL (structured query language) statements enables storing data while checking for data consistency, such as with respect to table existence constraints and reference constraints. This assures consistency of the data in the database. When inconsistencies are detected in the data loading process, the processing time becomes complicated to predict. Therefore, confirm the consistency of data with respect to table existence constraints and reference constraints while creating the input data.

If one database is created over multiple external storage devices, it becomes difficult to process large volumes of data in a short time. In this case, create an independent database for each external storage device. When processing can be performed separately for each storage device, it becomes easy to handle large data volumes over a short time.

When using a database management system with the data consistency check function, specify in which physical database to be accessed in the database definition.

-SQL is a typical database language for relational database management systems.



Data registration

(5) Verifying the databases

If the verification of a database is neglected, it will become difficult to determine whether any failure occurred at database creation or later on. Although it is not difficult to verify all data when the volume of data is small, such verification becomes difficult for large volumes of data. In actual operation for large data volumes, checks are performed only for a specific data range, and if no error is found, all data is assumed to be correct.

3.4 Standard Interface between Modules

-Interface

This section discusses the interface, which is one of the basic elements of a module. Module partitioning into functional units refers to partitioning modules into independent sub-modules for each function.

Functional independence here means that the interface between modules is simple. Conversely, as the interface becomes simpler, module independence becomes higher. The interface can therefore be understood as a scale for independence between modules.

Another method of enhancing the module independence is strengthening the relationship between the elements in a module. As the relationship between elements becomes stronger, the independence between modules increases.

In other words, the following two methods can be used to increase the independence between modules:

Guideline for enhancing independence between modules

Module-inside guideline	→	Strong Binding
Module coupling level	→	Weak Coupling

The degree of modularization, which is related to the module design method, indicates how strong each element in a module is bound. For example, if a program having multiple functions is partitioned into separate functions, the degree of modularization becomes stronger.

The module coupling level, which is related to the individual handling of variable, indicates how complicated the interface between modules is or how interfaces affect each other.

For example, if modules reference common areas, the coupling level between the modules is strong, and the independence between modules is not maintained.

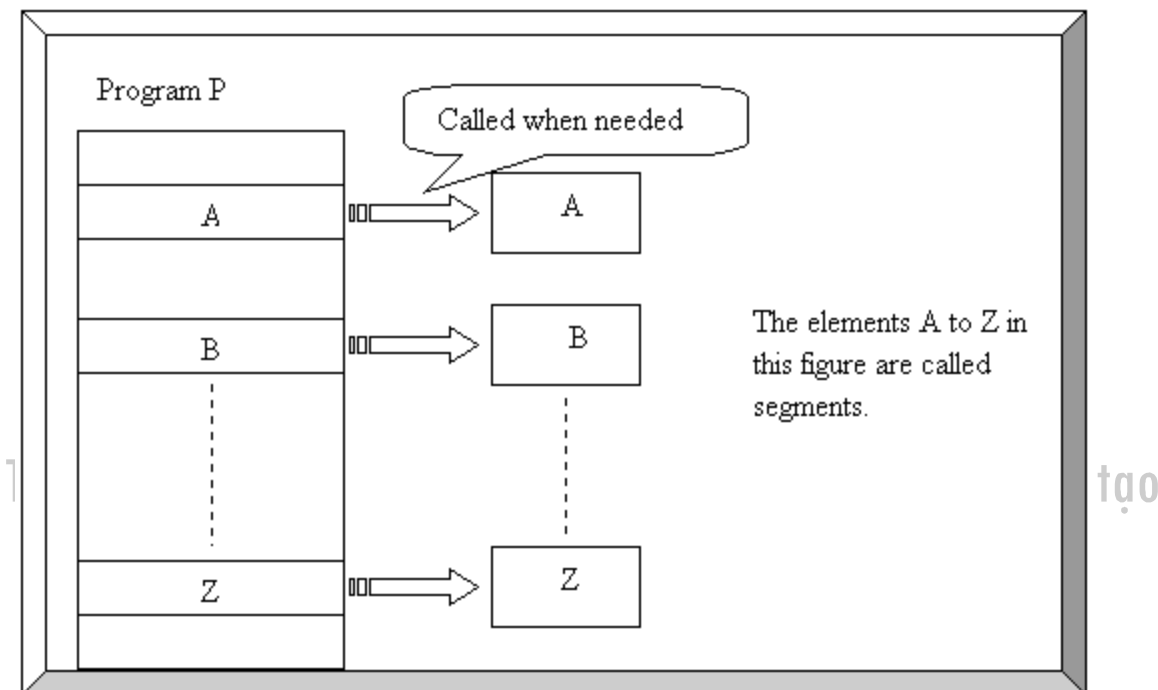
In a software product that consists of multiple programs, programs execute processes while linked to each other. Within a single program, multiple modules are linked with each other. Both can cause processing inconsistencies or design inconsistencies unless reliable interface conditions are established.

3.4.1 Interface conditions within a program

(1) Program structure

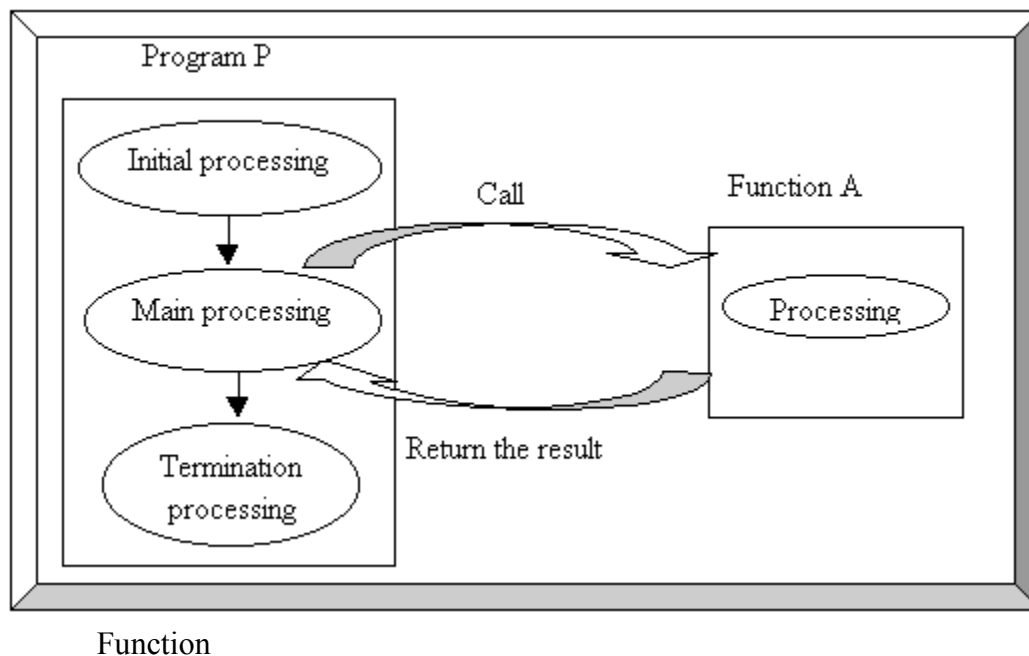
a. Segmentation

Partitioning a function within a program into smaller units is called segmentation. Segmentation enhances independence between modules.



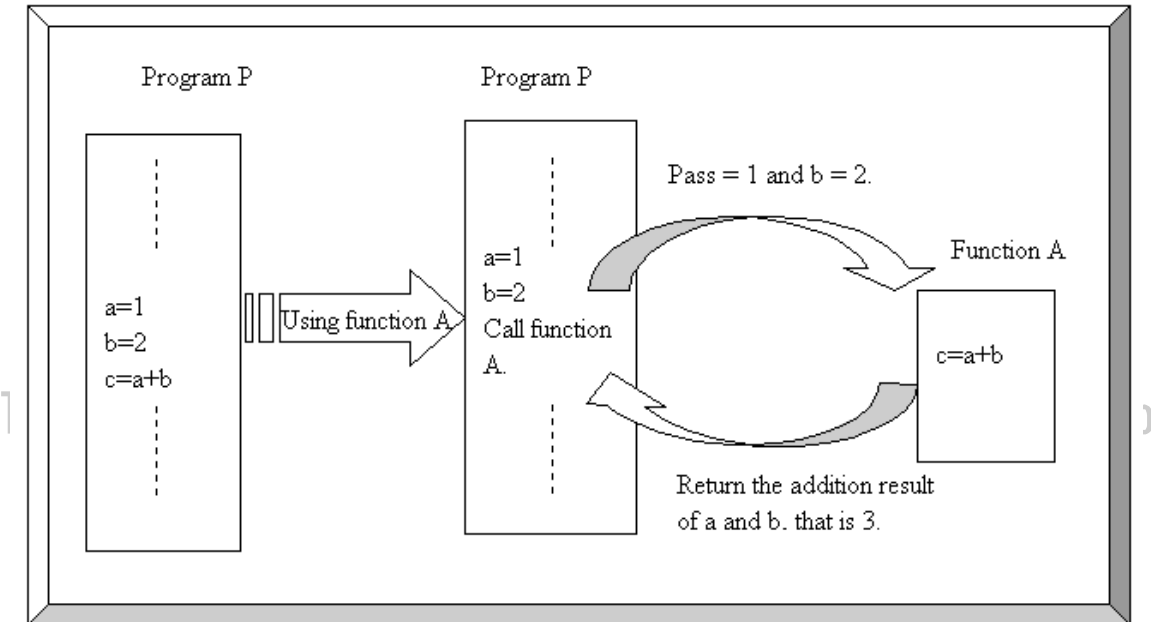
b. Creation of functions

-A function returns the result of the specified processing when it is called.



-Creation of functions means collecting similar processing items in a program into one as a function to provide shared functionality.

By function creation, similar processing items are collected in a single processing element. Therefore, the same processing can be called multiple times, and repeat the same process by simply calling a function.



Creation of functions

(2) Common and local areas

To determine the interface conditions between modules, it is important to understand which areas are common and which are local.

Use of a common area may be convenient for localizing information that must be consistent at the same location within the system. However, a common area makes it possible to reference and change the information from any location at any time. For this reason, a value in the common area cannot be assured, which can result in program errors.

Programs using a common area must perform processing by applying multiple check routines. This makes the program complicated. One important principle of system structuring is to remove the common area, and localize to local areas within modules. This weakens the coupling level between modules and enhances independence between modules, as described before.

Control and processing functions

A program processes input data and then outputs it. The code in a program includes groups of data, function statements for processing that data, and control instructions that define how these commands are to be combined.

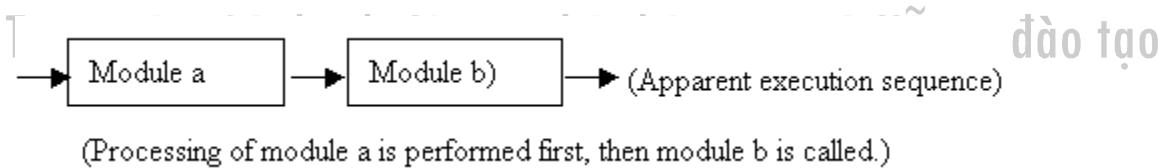
- Data
- Processing
- Control

For program design, be sure to understand this concept and the group of data processing instructions from the group of control instructions for examination. The data processing instructions mainly perform data transfer, calculation, and edit processing. The control instructions define processing conditions and the sequence of data processing instructions.

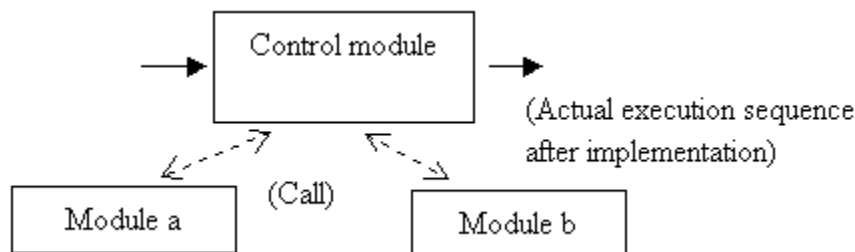
For example, a program would be written so that the processes for input data check and output data transfer are created as individual modules and called from a separate control module. By using this approach, the processing to be performed in each module is distributed over multiple modules, and the processing complexity within the system is reduced.

However, note that the apparent execution sequence is different from the actual execution sequence after implementation.

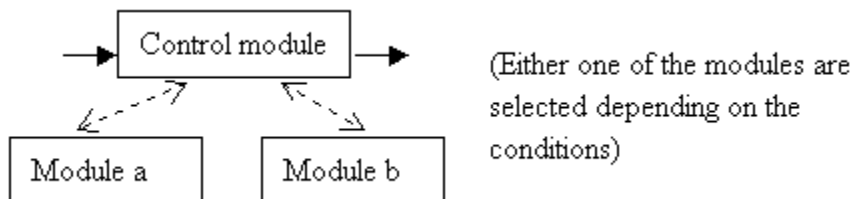
-Example of sequential processing
Module b is executed after module a.



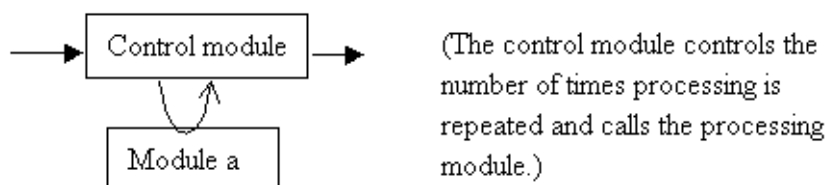
This process includes the original processing functions to be implemented for each module, as well as a control function for determining which process is to be performed after the original process terminates.



- Example of selection processing



- Example of repetitive processing



Partitioning a program into control and processing modules enhances the independence of the program, and it simplifies the interfaces.

(1) Encapsulation of data and processing

Encapsulation means collecting data and processing in one unit. A data type obtained by encapsulating the data structure of a specific data item and the processing for that data item is called an abstract data type. The actual processing is then executed as a procedure or function that operates on the defined data structure.

A defined data type can only be referenced or used in an operation from outside, and data types are designed so that only minimum information is supplied to outside.

3.4.2 Interface conditions between programs

The following are the interface conditions between programs:

1) File interface

The linkage between programs is performed via an intermediate file.

2) Arguments

The linkage between programs is performed by transferring arguments. Using arguments, the calling program uses only the specified arguments and the arguments for return values, and it is not necessary to understand the detailed processing contents of the called module.

In other words, arguments allow information hiding.

3) System common area

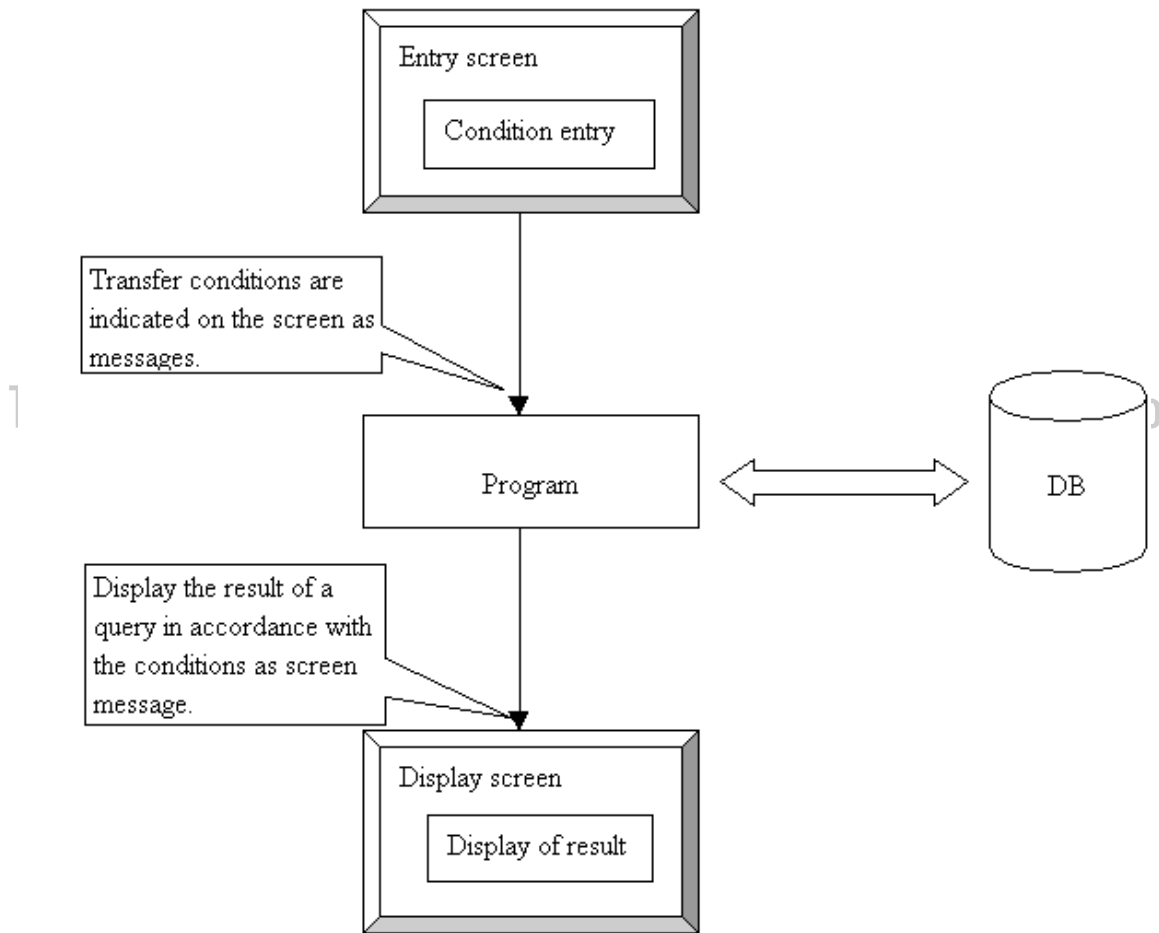
Linkage between programs is implemented via a system common area (an area used as a common interface among all programs).

<http://www.vitec.org.vn>

4) Message communication

A message requests execution of specific tasks.

Message communication performs program linkage via messages. Processing proceeds while information is exchanged between programs via messages.

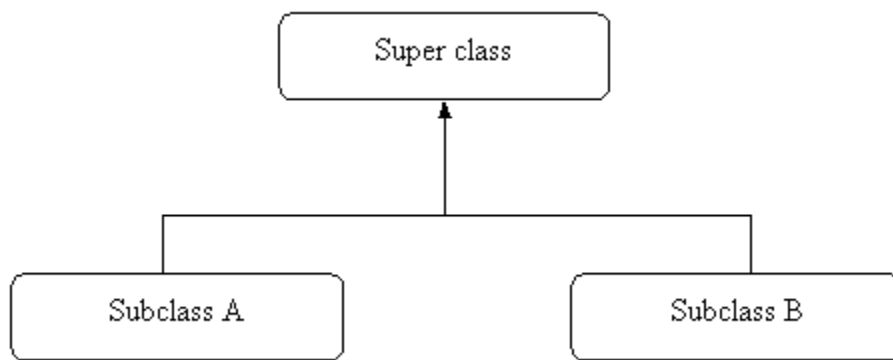


[Message communication]

5) Class inheritance

Inheritance refers to inheriting a specific property. This relationship holds between classes, for example, when the property defined in a super class (a class that functions as the inheriting source in an object-oriented language program) is inherited to a subclass (a class that inherits the super class function in an object-oriented language program).

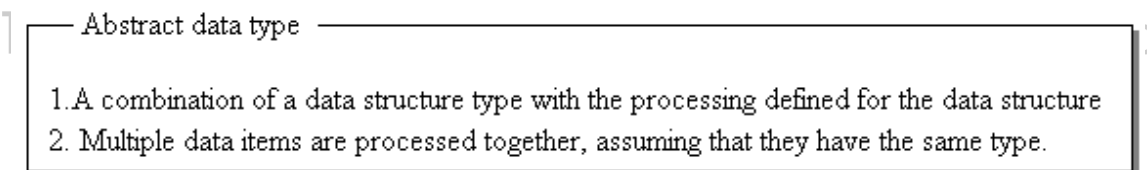
When defining a subclass, add only the required properties.



Class inheritance

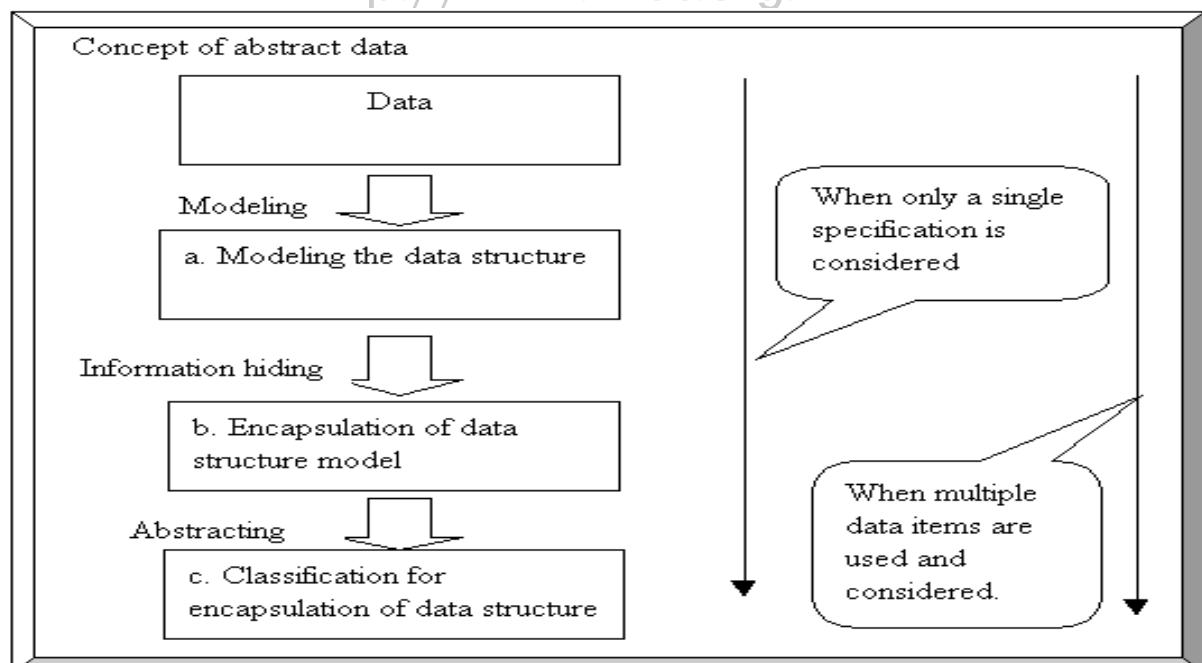
3.4.3 Conditions for reusing program interfaces

(1) Design methodology for reuse



1) Concept of abstract data types

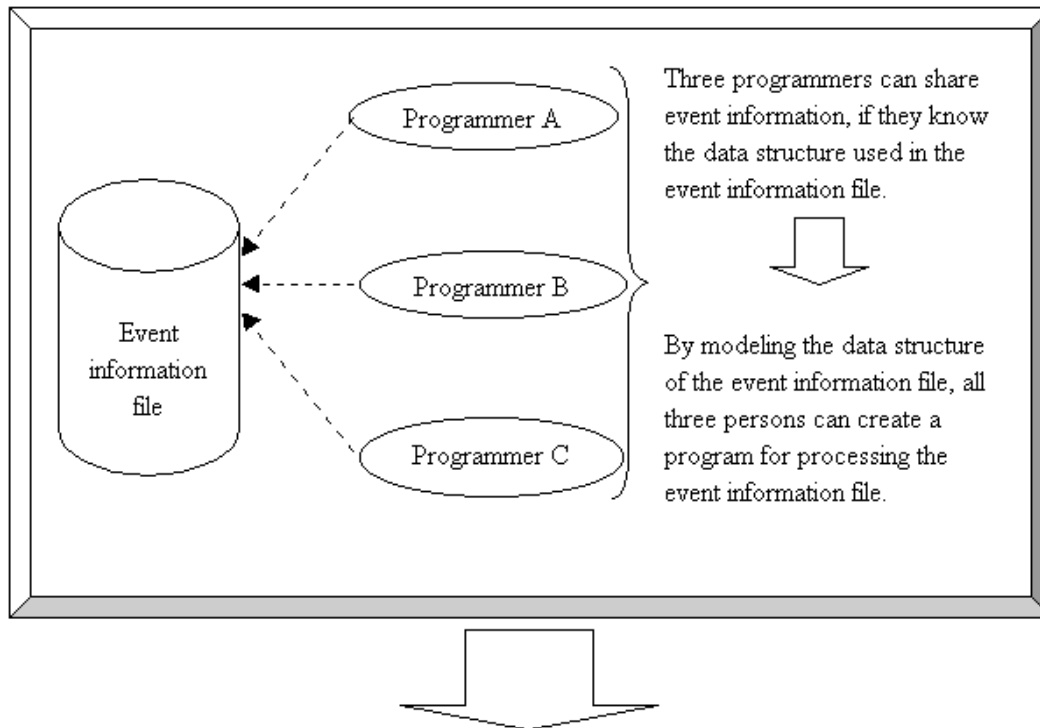
If only a single specification is considered, it is sufficient to execute the sequence in the diagram up to step “b, Encapsulation of data structure model” to enhance independence between modules. In practical use, however, user programs (UP) and packages are diverse and subject to change. One cannot design a program while ignoring these software characteristics. To enable reuse, one must analyze the software with respect to diversity and changeability. During analysis, the concept behind item c in the diagram, “Classification for encapsulation of data structure model,” must be considered.



a. Modeling the data structure

Contents of modeling the data structure

- Modeling the data structure means expressing the configuration of the data structure area on the basis of the specified conditions. This function is necessary to allow multiple programmers to share information.



It is insufficient only to create a data structure model.

- a.. Can each programmer adhere to the data structure model of the event information file?
- b.. Can new rules be informed surely to all programmers when the event information file undergoes a model change?

-Even if the data structure model is defined by this process, each programmer does not necessarily write a program that follows that data structure model, because programmers may choose the design they like. This means that, if data changes later on, it is useless to set new rules for the data structure model; each programmer must correct program contents independently. These kinds of corrections would cost extra money unless the authors perform the correction by themselves.

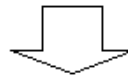
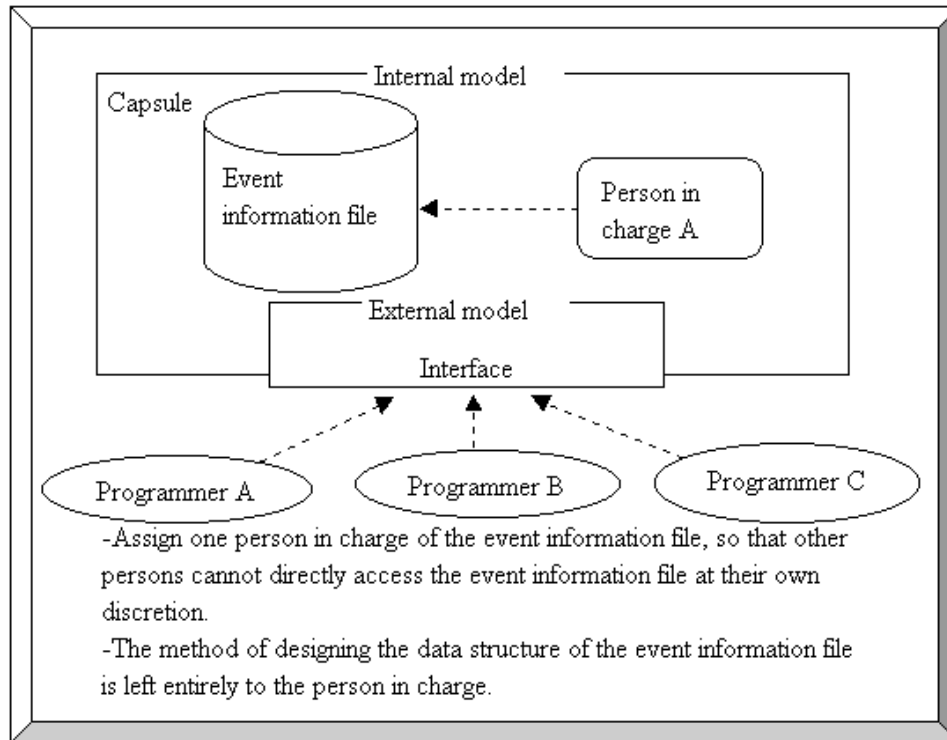
The problem arising from question a is that program design varies depending on programmers; this may decline program reliability and the debugging efficiency by review.

The problem arising from question b is that the correction costs increase, and the reliability of the corrected program becomes declines.

b. Encapsulating the data structure model

What does encapsulating the data structure model mean?

Encapsulating the data structure model refers to the inclusion of the data structure model into a tight shell ("capsule") so as to conceal components other than an interface.

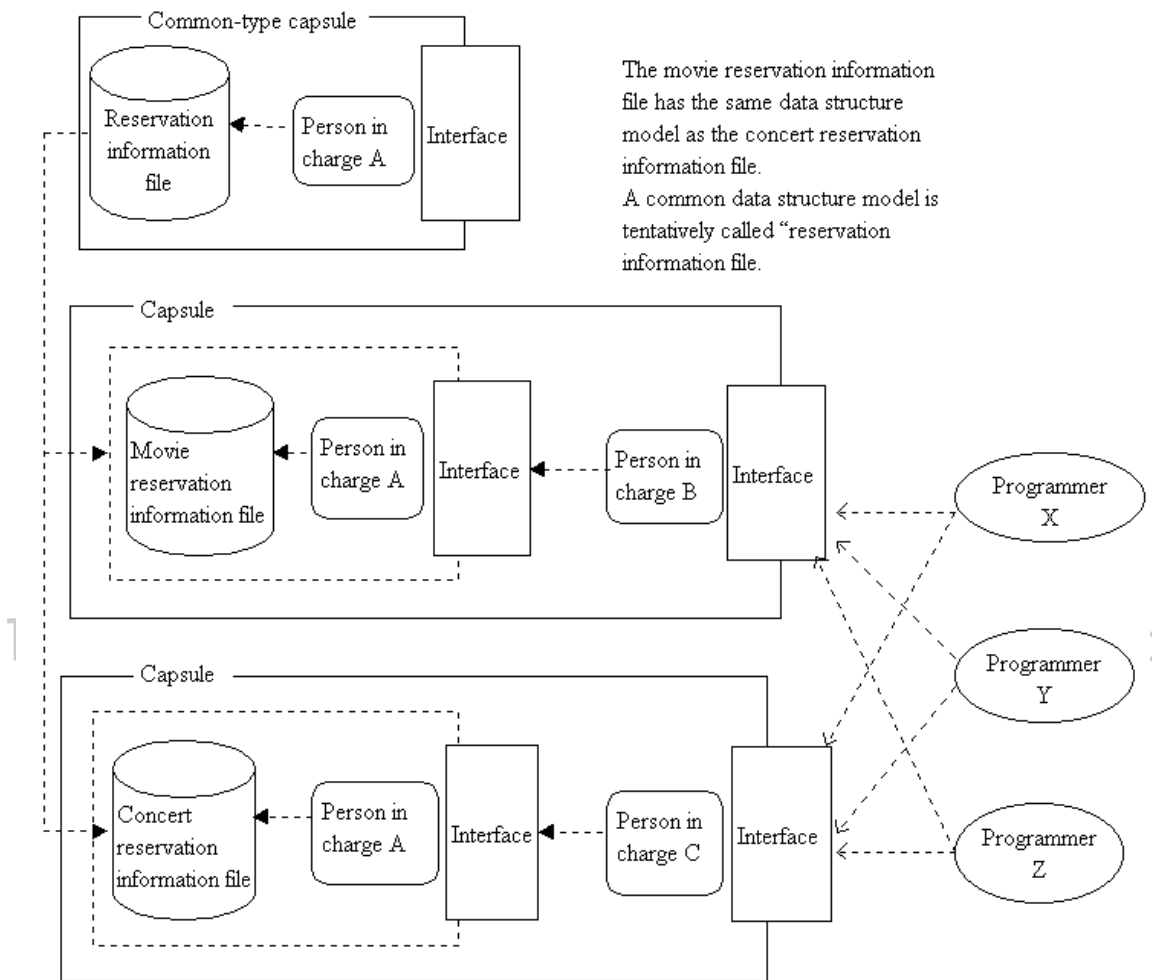


-Persons other than the person in charge can use the event information file only with the knowledge of the interface.
-The person in charge can freely change the data structure model as long as a common interface is defined.

-In the abstract data type, a data structure model supplied to outside as an interface is called an external model, and the actual data structure model only the person in charge can manage is called an internal model.

One characteristic of abstract data types is that one can easily change the internal model without modifying the external model.

c. Classification for encapsulating the data structure model



When the movie reservation information file has the same data structure model as the concert reservation information file, handling of these files is left entirely to the person in charge of the common data structure model. In other words, when persons in charge B and C handle the respective files, they ask person in charge A to perform the actual tasks to be performed.

The reservation information file therefore functions as a high-level “capsule” of common type that includes both the movie and concert reservation information files.

To process an abstract data type in COBOL, reserve a suitable data area in the user program. When defining the concept of the abstract data type in a program language other than an object-oriented language such as COBOL, one must create a class for encapsulating the data structure model.

(2) Kinds of reusable programs

1) Reusable program

A reusable program is a program that can be repeatedly executed once it is loaded into the main storage. The program is usable even after the program control has ended and the first execution of the program is finished.

To make a program reusable, the initial value of the area used is included in the system

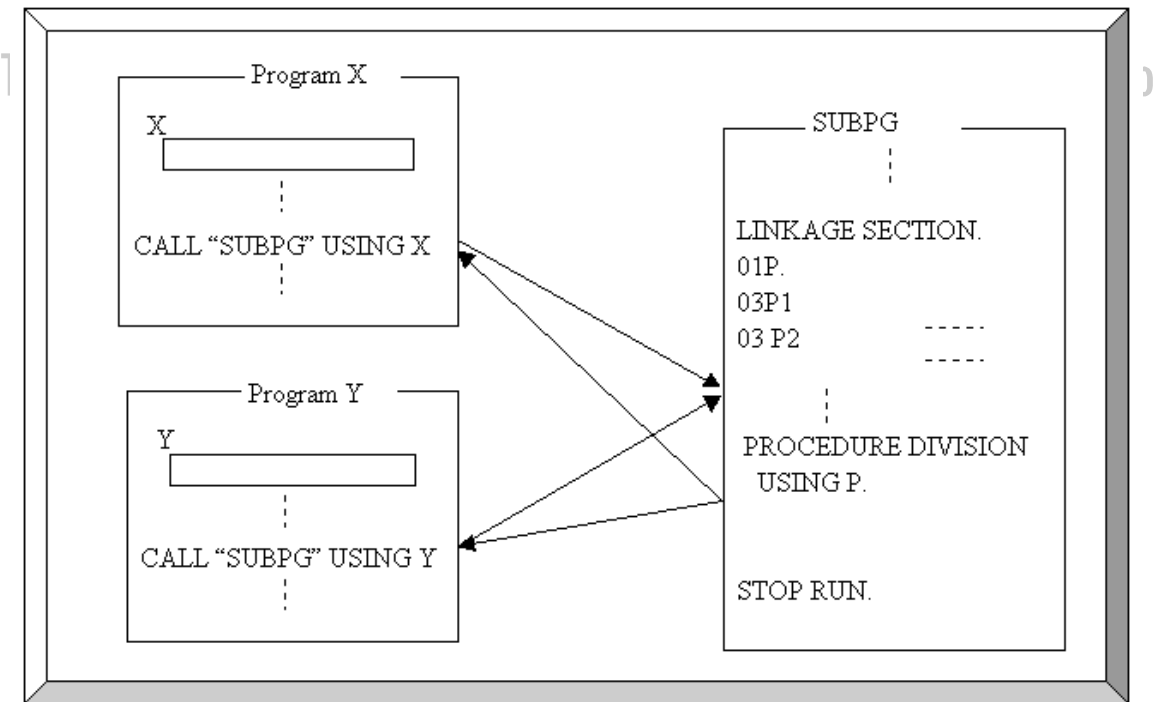
in a way that the processing contents are not changed during execution. Moreover, it must be ensured that the program returns to the initial values (if the contents are changed during execution) that were effective when the program was originally included in the system.

Reusable programs are mainly used for offline batch processing; they cannot be used for online real-time processing.

2) Reentrant program

A reentrant program is a program that is called from another transaction before it terminates. If such a program is called from multiple transactions in real-time processing, a new data area is reserved for each transaction so that other transactions are not affected.

When using reentrant programs, the calling program reserves the required area and transfers this area in parameters.



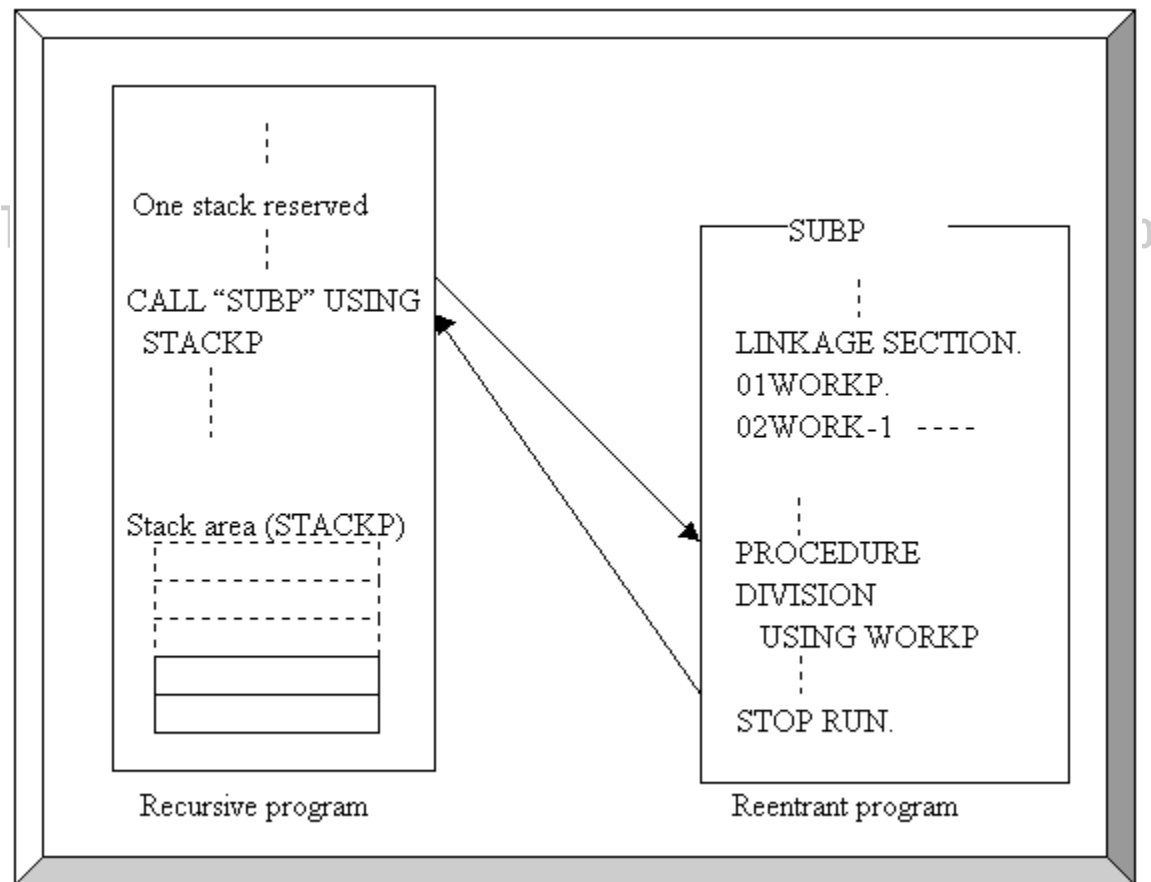
Reentrant program

3) Recursive program

The recursive program is a program that can call itself and define new program contents. For this program type, multiple areas of the size required for one execution must be reserved beforehand, and the program structure enable the same program to be called multiple times.

The group of areas to be reserved newly each time the recursive program is called is called a stack.

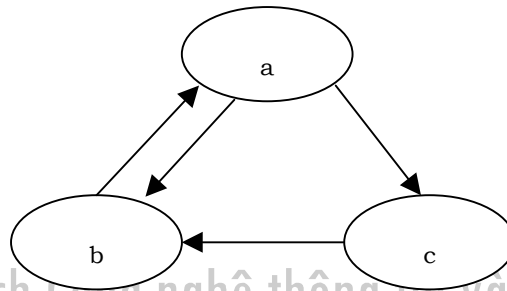
The area used is released when processing is terminated and the system returns to the original program.



Recursive program

Exercise

1. Select a means for expressing a processing procedure for a structured program module where all processes are indicated with rectangles in an outline of processing.
 - a. Data flow diagram
 - b. Decision table
 - c. NS chart
 - d. Hierarchical function diagram
2. Select the suitable terms for items a to c in the figure.



- (1) Wait state (2) Execution state (3) Executable state
3. Select the correct statements from the following explanations of exclusive control processing in a database management system.
 - (1) To prevent deadlock, enlarge the range of exclusive control processing.
 - (2) When an exclusively controlled program terminates abnormally, immediately release the exclusive control state before confirming the state of the failure, so that other programs are not affected by the failure.
 - (3) For preventing deadlock, it is important to release the previously locked resource before the next one and unify the resource-usage sequence.
 - (4) Exclusive control processing is suitable for online updates, but is not necessary for batch updates.
 4. Select the item(s) that appropriately explain(s) the recovery processing to be performed when the transaction program terminates abnormally in an online system.
 - (1) Output database records before and after update, and a journal file containing the processing program.
 - (2) When a database is updated, by an error in job processing, use the updated information in the journal file.
 - (3) When an access error occurs, use the latest backup file and information in the journal file before the update to recover data.
 - (4) In preparation for database corruption, periodically copy the contents of the database to a backup file, and record update information in a journal file during online operation.
 5. What term refers to the act of storing transaction information in the auxiliary storage at a specific timing before restarting processing with the fault recovery function of the database management system? Select the appropriate item.
 - a. Roll-back
 - b. Backup copy
 - c. Checkpoint

d. Commit

6. Select the item(s) that appropriately explain(s) the purpose of the secondary key in an indexed sequential file.

(1) It is possible to register two or more records for the same secondary key.

(2) The secondary index manages which track is used to store a record for a key like the primary index.

(3) The secondary key, first used to search for the basic data area, can later be used to check the overflow area if the target is not found.

(4) When using the secondary key to fetch all records having value P or greater, specify the secondary index to first search for records having value P, and then extract the corresponding records in the sequence in which they are physically stored.

7. Select the characteristic necessary to normally execute the currently processed program when the program is called again from another program.

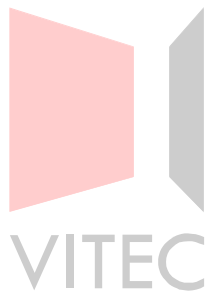
a. Reusable

b. Recursive

c. Reentrant

d. Reloatable

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

4 Creating Module Specifications and Test Specifications

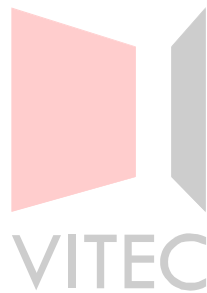
Chapter Objectives

This chapter explains the procedures for creating specifications for modules, which are the smallest units of a program, and test specifications.

4.1 Creating module specifications

4.2 Creating test specifications

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

4.1 Creating module specifications

The functions of modules and their interfaces identify module specifications, and the specifications include designs and documentation on the logical construction and area for modules to use.

This section explains the procedures and methods for creating module specifications.

4.1.1 Procedures

This first part focuses on the basic concept and procedures of module design. If its processing is complicated, writing a program to cover the entire logic all at once is difficult. To facilitate the creation of the program and its maintainability, divide the program into several parts based on independent functions before start writing the program. This unit of a basic function is a module.

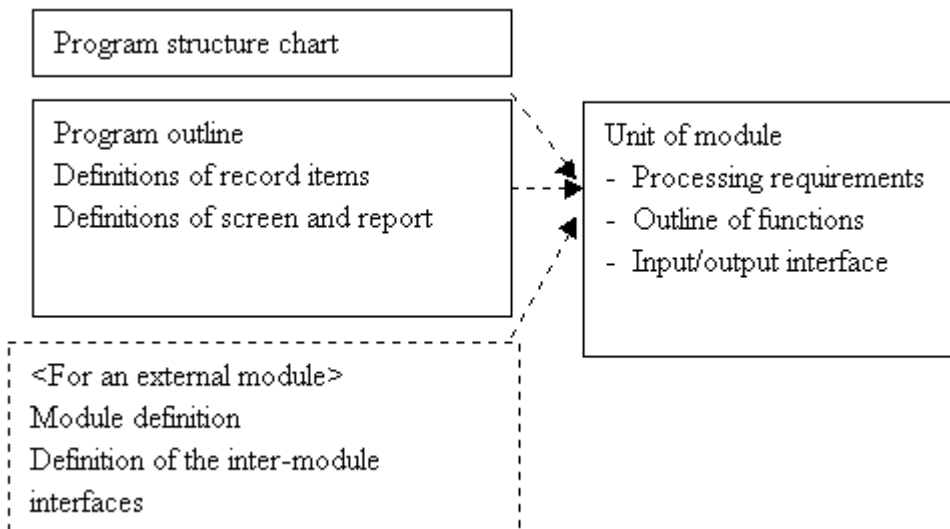
In module design, each defined module is broken down to an appropriate level where its coding can be done.

Module designs with detailed processing procedures can be created in the following steps:

- (1) Confirm an outline of module functions
- (2) Design detailed processing procedures for modules
- (3) Define data areas

(1) Confirm an outline of module functions

Confirm an outline of module functions by using documents created in the processes prior to module design.



(2) Design detailed processing procedures for modules

Design detailed processing procedures that are consistent with the design principles of structured programming. At this point, focus attention on the following points:

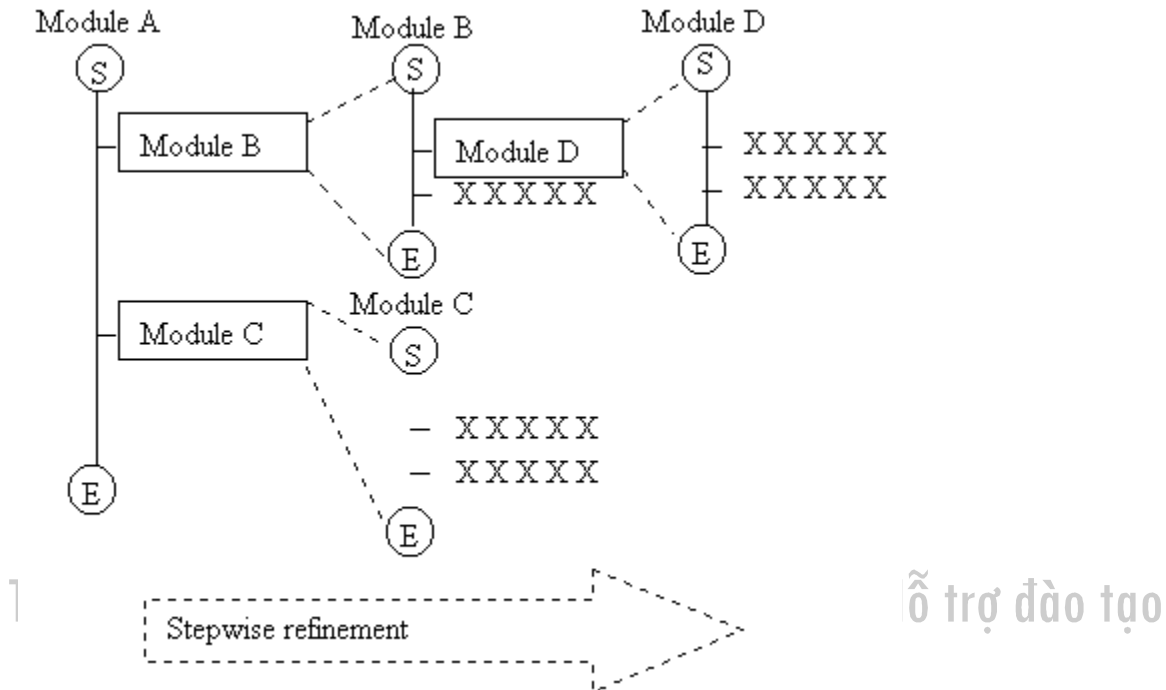
- Develop the program logic so that it can be read from top to bottom sequentially.
- Create only one entry point and one exit point.
- Do not use unnecessary GO TO statements.
- Organize the instructions into meaningful groups representing functions.

The basic principle of module design is to standardize the control structure so that the program logic has only one entry point and one exit point. Standardizing program structures involves three structures: sequence, selection, and repetition. If all parts of a module design are expressed by combining these structures, the those points are almost satisfied.

If this basic structure is not observed, and many unconditional branch instructions, such as the GO TO statement, are used carelessly in a design, the control structure becomes disorganized and the control logic is complicated. If the control logic is developed in the basic control structure, the processing flow can be easily understood just by following the logic from top to bottom, thereby making it easy to develop a program that provides clear functions in its individual units of processing. These rules make the structure and logic of programs easy to understand for not only developers but also maintenance personnel. One of the basic principles of program design is to create programs whose designs are easy for the others to understand.

To design detailed processing procedures, refine stepwise the processing from rough processing into detailed processing. Do not think about detailed processing at first. For example, to design detailed processing procedures for module A, in the example, focus attention on the functions assigned to module B and module C, the times at which these modules are called, and requirements for calling these modules. However, module D does not have to be considered.

If a refined processing procedure corresponds to one instruction in a programming language, stop the refinement.



(3) Define data areas

Define the data areas for the processing procedures. These data areas are identified in module designs, and they include data areas for counters, control flags, and intermediate results. Some areas may be allocated for data that is saved temporarily during processing. Other than the designers of the software using such data, people may have difficulty figuring out the purpose of such data, and this may increase the complexity of the programs. Therefore, these data areas must be defined clearly in documents.

Focus attention on the following points when defining data areas:

- Use a data item for a single purpose
- Use a data area defined in a module only in the module

4.1.2 Methods

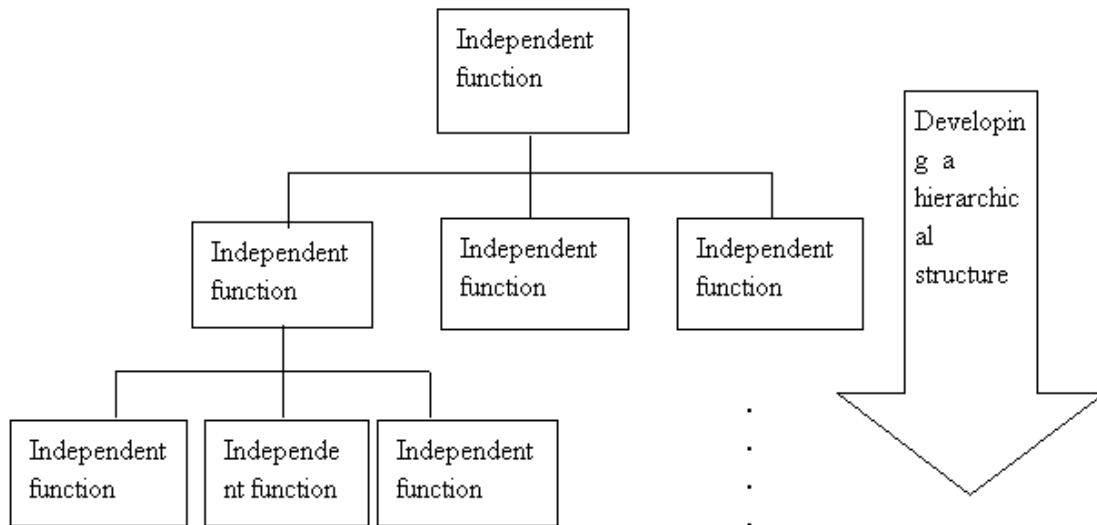
Multiple functions are integrated organically into a system. If these functions have a complex interrelationship, a variety of aspects have to be considered at the same time. The corresponding complexity tends to exceed an author's capability, leading to the incorporation of many errors, and this eventually results in a less reliable program being produced. For this reason, it is preferable to use a program design method that does not require too many aspects to be considered.

Methods for logical design are as follows:

- (1) Structured design method
- (2) Data structure-oriented method

(1) Structured design method

In the structured design method, the internal structure of a program is developed hierarchically from the highest to lowest levels. In other words, the large functional units at higher levels are broken down into "independent small functional units" at lower levels in a top-down approach.



Top-down development of a program internal structure

The basic concept of partitioning programs by using a hierarchical (top-down) structure of modules is described in the following:

Abstract problems in the same way as the human thought process. In other words, capture the core concept of a problem, and break it down into smaller concepts.

- To break down a concept, limit the range of thought only to those related concept, and focus attention within the concept.
- Before attention can be focused on only the range defined for a concept, the range of concept must be defined during design at a higher level.

The most basic procedure for structured design is to break down an abstract concept into smaller concepts in successive stages, that is, to break down the concept into steps in the same way as the human thought process.

Based on this, structured design has the following three basic guidelines:

- Partition abstract concepts into modules, beginning from the upper level to the lower

level in a hierarchical structure, according to function.

- Partition functions into independent modules.
- Partition functions into small modules to make them easy to understand.

Next, examine the approach for developing a structured design. The scale of software to be developed differs depending on the processing. Generally, the procedure for designing the internal structure of a program is as follows:

1) Partition the program into independent programs

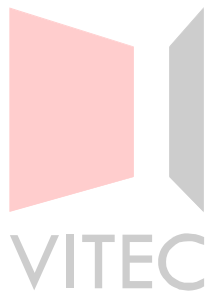
Since a large software application usually consists of multiple independent programs, divide it into individual programs.

2) Design the module structure

Break down the internal structures of the programs in a top-down approach to create the module structure.

3) Evaluate the module structure

Validate the module structure for the individual programs by verifying that the independence of each module is maintained, and modify the structure if necessary.



<http://www.vitec.org.vn>

(2) Data structure-oriented method

In the data structure-oriented method, a program is partitioned into modules by focusing the analysis on groups of data (such as units of repetition) in the program file. The basic concept of breaking down a program in the data structure-oriented method is described as follows:

- Analyze the data structure of the main input file.
- Analyze the data structure of the main output file.
- Consider the program to convert the input data structure into the output data structure.
- Partition the program into modules, with consideration of its functional independence, ease of understanding, and commonality.

To partition a program into modules, it is better to use a method where partitioning can be executed independently, without affecting other modules, by considering maintenance of modules. Hence, select a partitioning method with weak binding. Module partitioning methods are explained in the following:

- 1) Source Transform Sink (STS) partitioning method
- 2) Common function partitioning method
- 3) Transaction partitioning method
- 4) Jackson method
- 5) Warnier method

1) STS partitioning method

In STS partitioning, a program is divided into three functions: input, processing, and output. Each of these functions is defined as a module.

2) Common function partitioning method

In common function partitioning, functions found to be the same are extracted as one common module.

3) Transaction partitioning method

In transaction partitioning, modules are defined by transaction. Different types of processing are selected according to the transaction.

4) Jackson method

In the Jackson method, data structures are analyzed, program structures are created from the input data and output data, and individual modules are defined with the combination of three structures: sequence, selection, and repetition.

5) Warnier method

In the Warnier method, where one file is regarded as one group, each group is broken down into parts that are defined as modules. If the same processing is repeated more

than once, it is organized in a loop structure.

4.1.3 Main considerations

To enhance the reliability of a program, it is crucial that the "criteria (paradigm) for the module independence " be completely understood, and that the program be created based on the criteria. This is a major guideline concerning whether structured design is considered as a "technique for partitioning into modules." Since they are basic points about methods following the structured design method, understand them clearly.

(1) Basic characteristics of modules

Before considering the method of enhancing the independence of modules, examine the characteristics of the modules.

A module can said to have four basic characteristics or attributes. These characteristics are quite different. So unless the differences are thoroughly identified, programs cannot be organized into good structures consisting of independent modules.

<Basic characteristics of modules>

- Functions: What does it do?
- Logic (procedures): How are the functions carried out?
- Interface: How do modules relate to other modules?
- Size: What is the size of each module?

Take the these characteristics into consideration from three aspects with regard to modules: module structure, module specifications, and the interface between modules. Module specifications are measured by module strength, and the interface between modules is measured by module binding strength.

(2) Functions (What) and Logic (How)

Examine the relationship between the program functions and logic. The functions must be partitioned into modules by first identifying the functions (What) and logic (How). There are clear differences between the functions and logic, as explained in the following.

After defining the functions (What), determine the logic (How), which serves as the procedures for providing the functions.

When partitioning the functions into modules, it is important to implement partitioning processing by considering the differences between the functions and logic, as explained in the following.

<Guideline for module partitioning>

Implement module partitioning according to the functions (What)

If a program is partitioned into modules in a top-down approach following this guideline, it is possible to design a hierarchical structure of modules that are independent by function. If the program cannot be partitioned into modules by function, try to use appropriate words to describe each function. For example, if "Do something" are the words used to describe a function, the function defined as "Do something" can be considered to be a unit of a module.

(3) Interface

The next step is to think about the "interface," which is the one of the fundamental characteristics of a module. "Module partitioning by function" means partitioning a function into independent modules by function.

There are two criteria used to enhance the independence of modules.

<Guidelines for enhancing the independence of modules>

- Guidelines inside modules → Strong binding (increased binding strength)
- Module coupling → Weak coupling (weaker binding strength)

- Guidelines inside modules

One of the methods for enhancing the independence of modules is to strengthen the relationship between the internal elements composing the module. The stronger the relationship, the more independent the module.

- Module coupling

If modules are independent by function, the interface between modules is simple. Conversely, the simpler the interface, the more independent the module. Therefore, this interface is one of the criteria used to measure independence between modules.

The strength of a module indicates how strongly elements are coupled within the module, and it pertains to the design method of modules. For example, if a program incorporating multiple functions is partitioned by function, the strength of the module is enhanced.

In addition, module coupling indicates the complexity of the interface between modules and the extent to which modules affect one another, and it mainly pertains to a method of handling variables.

For example, if multiple modules refer to a single common area, coupling between these modules is strong, and it would be difficult to maintain the independence of the modules.

In software made up of multiple programs, the programs work together to execute individual processing tasks. Multiple modules in a program work together in a similar way. Use of either multiple programs or multiple modules is likely to cause

inconsistency in processing and design unless concrete interface requirements are established as a standard.

1) Interface requirements between programs

The interface requirements between programs are as follows:

a) File interface

In a file interface, programs are correlated using intermediary files.

b) Argument interface

In an argument interface, programs are correlated using arguments.

c) System common area interface

In a system common area interface, programs are correlated through the system common area (area used as a common interface by all programs).

2) Common area and local areas

It is important to distinguish the common (global) area and the local areas, with consideration of the interface requirements between modules. Although it is sometimes convenient to centralize information to ensure consistency in a system, the data in the common area is not protected because it is possible to refer to and revise the data from anywhere at any time. This may lead to program errors.

Many check routines should be performed for programs using the common area, which can make such programs more complicated.

Hence, in the structured design method, the basic principle is to eliminate the common area and place information in local areas in a single module. This leads to weakening module coupling and increasing module independence.

3) Control function and processing function

Programs are designed to process the input data and to produce output by specific kinds of operations. Programs are written to include the data for processing, functional instructions for the data, and instructions controlling the behavior of a set of other instructions as a group.

- Data
- Processing
- Control

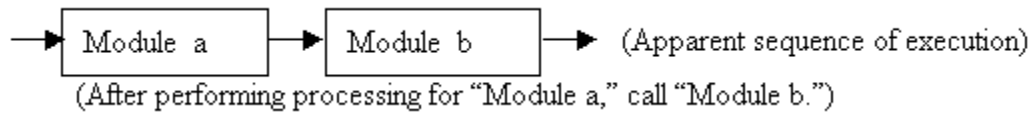
For program design, it is essential to understand these characteristics. Consider classifying the groups of instructions into the following two categories for program design: groups of instructions for processing data, and groups of control instructions. The data processing instructions focus on data transfers, processing operations, and editing, and the control instructions define the conditions for processing the data processing instructions and the sequence of processing.

For example, create a module for checking the input data and the module for the transfer processing of individual output data items, and execute them by calling them from the control module. If a program is designed in this way, the processing of individual modules is shared, and the structure is less complicated.

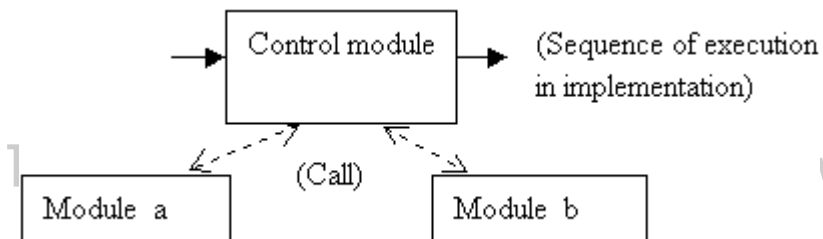
Also note that if a program is designed in this way, the apparent sequence of execution differs from that of implementation.

<Example of sequential processing>

Process “Module a” followed by “Module b.”

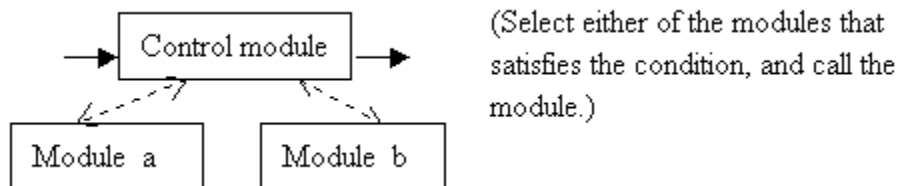


This example of processing includes functions intended to be processed in individual modules, and a control function for defining the sequence of the next processing task after the current processing task is completed.

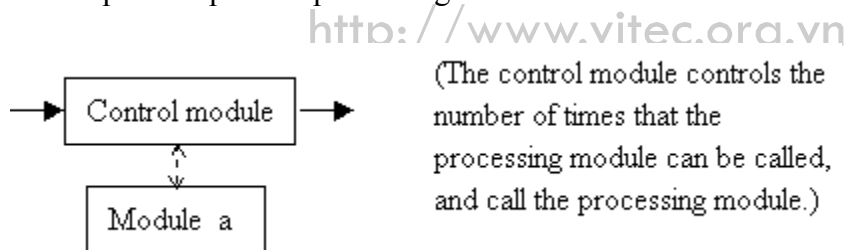


và Hỗ trợ đào tạo

<Example of selective processing>



<Example of repetitive processing>



If the program is partitioned into modules by dividing the control function and processing function, the independence of the program is enhanced, and the interface becomes independent.

(4) Size of modules and logic

Finally, regarding the sizes of modules, if modules are partitioned by function, a problem arises with regard to the extent to which they should be partitioned. For programmers, the smaller are the modules, the easier is the programming. In other words, if a module is smaller, programmers do not have to think about numerous aspects in a similar way at the same time. This makes it easier to consider the

remaining work related to "logic." However, as a prerequisite the independence of modules has to be enhanced beforehand.

After all, module size is an important metric that significantly affects the ease in developing the program logic and ease in which the relevant programs are understood by maintenance personnel. Then, what is the appropriate module size? People experienced in structured programming have given the following comments:

- With regard to the size of a module, the smaller, the better. It would be good if a program code is so short that it could be written on a single sheet of paper. (Harlan Mills)
- It should be 30 statements or fewer. About 30 statements are the maximum that human beings can understand at one time. (Gerald Weinberg)

Module size is ultimately determined by physical restrictions, such as the size of printing paper, and limitations in the capabilities of the human thought process, such as people's abilities to understand coding easily. The following guidelines are for determining the sizes of modules:

<Guidelines for determining module size>

- (1) Physical restrictions
Coding can be written on a single sheet of printing paper.
- (2) Limitations in the capabilities of the human thought process
The algorithm of each module can be formulated in one's mind.
- (3) Restrictions imposed by the status inside the module
The functions in modules cannot be partitioned any further.
Nothing can be gained by further partitioning into modules

The following is criteria concerning the size of a general module, and it is preferable to create a module within the following ranges:

- External module: 500 to 1000 steps
 - Internal module: 20 to 100 steps
- (The above has been calculated in the number of lines of code for a source program in a high-level programming language.)

Finally, module partitioning includes the following caution:

If a mistake has to be made in module partitioning, it would be better to partition a function into modules that are too small. This is because it is easy to put them together to make a larger module after they have been partitioned into very small modules. However, the opposite task is difficult.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

4.2 Creating test specifications

Test specifications are the document that specify types of tests and their objectives, test plans, and test methods. This section explains different types of tests and their objectives, the procedure for creating the test plans, and test methods.

4.2.1 Types of tests and their objectives

Tests can be roughly divided into four categories, depending on the programs to be tested, the scale of processing, and their contents.

- (1) Unit test
- (2) Integration test
- (3) System test
- (4) Operation test

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

(1) Unit test

The unit test is conducted at the level of a module, which is the minimum unit of a program, or by a program. The purpose of the test is to make sure that modules and programs operate according to the module or program specifications, as expected in the initial objectives.

The unit test is not intended to check if programs are logically correct. The programs being tested usually have errors. The objective of the unit test is to find errors and correct them.

The more errors that are found and corrected at an early stage (unit test) of development, the less work is required to be done. This is because the scale of testing becomes greater as work progresses further along, from the unit test to the integration test to the system test, and more work would be necessary as the test scale becomes larger to correct any errors that are found.

If as many errors as possible are found and corrected at the unit test stage, the quality and reliability of the entire system can be improved efficiently.

(2) Integration test

After the unit test is completed for the program, the integration test checks the results of integration to determine if it is successful. All kinds of integration are checked, including between-modules, between-programs, between-subsystems, and inter-system integration as well as integration from the terminal and from line to program. It also checks if the desired functions are consistent as a result of integration. Generally, the staff that conduct the unit test conduct this test too, following the integration test specification and the job flow.

(3) System test

The system test checks if the “product objectives” for the entire system can be met, and this involves checking for inconsistencies with the objectives defined in the planning process. Such objectives are related to system functions, reliability, and operations.

(4) Operation test

The operation test is conducted in an environment similar to the operating environment. Generally, while the test is conducted on a new system, the existing system is also operated in parallel, so that users and operators become get familiar with the new system, and that the existing system can be replaced with the new system.

In the process from analysis to design of individual programs, large (whole) functions are gradually broken down into small functions (sections). In contrast, in the testing process, the tests are conducted by building a system, beginning from sections to whole functions as individual processes and programs are integrated.

Test items and testcases are extracted for each testing target and compiled into test specifications. Finally, the staffs who conduct testing are organized, the criteria for starting and ending the test are determined, and the test environment, test tools, and resources are chosen. This kind of work is referred to as “test design.”

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

4.2.2 Important considerations in testcase design

There are two types of testing methods: the function test for verifying functions extracted from program specifications, and the logic test for verifying whether the logic of detailed processing procedures is correct.

(1) Extracting test data for the function test

Serious consequences would result if there were omissions and errors resulting from misunderstandings in program design documents (such as the module structure diagram, module definitions, and module internal processing procedures).

Therefore, extract testcases from program specifications (program outline, file and record specifications, report layout). If the necessity for testing is unclear because there is no description in program specifications, be sure to confirm with the system engineers or team leaders. The following points should be a reference for creating a function test.

(Considerations for designing a functionality test)

Input	Range check	What is the range of values for the item? What value within the range causes a change in the processing flow? What will happen if an out-of-range value is inputted? Are all testcases supported? Examples of range tests include the date check and numerals range check.
	Data attribute check	What kind of characters can be inputted for each item? Can characters, only numerals, or a space character be inputted? What are possible entry points for inputting such data items? Are all possibilities checked?
	Check for correlation between items	Does an inputted item have a counterpart? If the item has a counterpart input item, it must also be included in the list of check items. Conversely, make sure to exclude items that cannot be used as an input item. Are all such checks conducted?
	0 input items	If the number of data items to be inputted for a file is “zero,” has logic been developed to handle this event as an error?
	Check for the maximum and minimum values	Has good consideration been given to the processing results, when the maximum or minimum value is specified as the input item?
	Matching check	Are all matching results (>, =, <) considered for every combination? Are combinations of 0 to N, 1 to N, N to 0, and N to 1 considered for making comparisons? What about “N” to “M”?
Output	Logical check	Is there any contradiction in calculation results? (Is the following equation valid?: quantity x unit price = amount) (The check is conducted manually.)
	Totaling check	When a quantity or amount is totaled, is the total correct? (The check is conducted manually.)
	Count check	Does the number of the inputted records match the number of the outputted records? (The test is conducted manually.)
	Line feed check	Have all possibilities of a line feed occurring at the time of listing output been considered?
	Page change check	Have all possibilities of changing to new pages when a listing being outputted over multiple pages been considered?
Internal	Table	Have all of the boundary values for the internal table been checked?
	Working area	Have all of the boundary values for areas and subscripts been checked?
	Arithmetic operation	Is there any possibility of a zero division or overflow?

First, classify the possible values of input data into groups based on the processing of the program, and design tests so that at least one case from each group is tested. Second, for the possible values of output data, design tests in the same way. It is important to test the line feed operation, page change operation, and output locations, especially for list output. Finally, define appropriate settings to check if the tables and areas for internal use are set up correctly, check if subscripts have the correct values, and check results if a value exceeds the limit.

(2) Extracting test data in the logic test

After finishing extraction of the function testcase, the next step is to extract testcases for the logic test to check the program logic. In this test design, focus on the conditions indicating control of the detailed processing procedures, read the symbols to find out the associated conditions, and create cases to meet (and not to meet) the conditions. If all conditions are identified and combined to make the cases, the testcases for testing all of the logic are created. However, if many conditions coincide, the number of the testcases becomes astronomical and less realistic.

Hence, the number of testcases must be reduced without reducing their effectiveness. There are two levels for measuring test coverage.

<Test coverage at the module level>

- C0: Execute all instructions at least once.
- C1: Execute all execution routes at least once.
- C2: Execute all of the items for predicates in all execution routes at least once.
- C3r: Execute the loop up to “r” times (the maximum number of times).

<Test coverage at the system level>

- S0: Call all modules at least once.
- S1: Execute all calls for each of all modules at least once.
- S2: Execute every type of response method from each of all modules at least once.
- S3: Execute the program at least once for all the predicates of all parameters on the interface between modules.
- S4d: Execute via the interfaces between modules all modules up to depth “d” (corresponding to C0-C3r at the module level).

It is advisable not to create too many testcases, because this leads to delays in work. But if there are too few testcases, quality assurance is difficult to achieve.

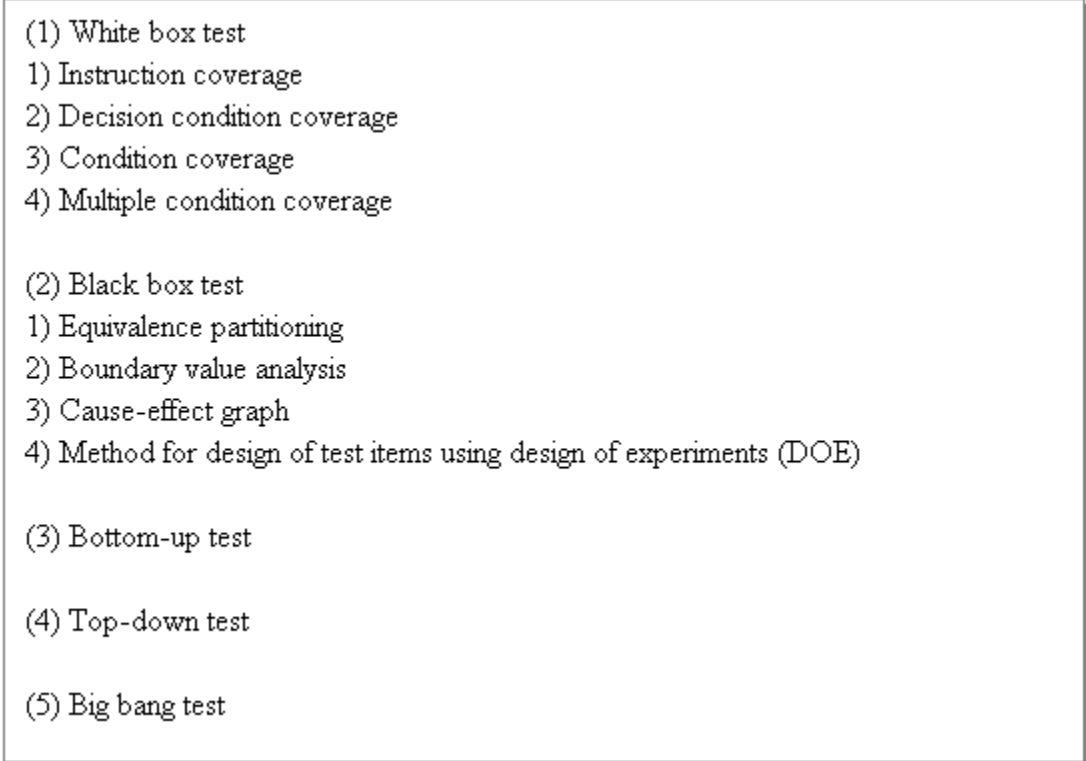
Generally, tests are conducted at a level ranging from C0 to S0 and from C1 to S1. This textbook adopts the use of levels ranging from C1 to S1.

If the flow of a testcase is traced with a red pencil on a flowchart, the processing flow is clarified. This processing flow is called the “logic path.” If all logic paths are traced on

a flowchart, all flows on the chart are traced in red during testing at levels C1 to S1. If a route is not marked in red, this means the test is not sufficient.

4.2.3 Testing methods

The following five testing methods are commonly used:

- 
- (1) White box test
 - 1) Instruction coverage
 - 2) Decision condition coverage
 - 3) Condition coverage
 - 4) Multiple condition coverage
 - (2) Black box test
 - 1) Equivalence partitioning
 - 2) Boundary value analysis
 - 3) Cause-effect graph
 - 4) Method for design of test items using design of experiments (DOE)
 - (3) Bottom-up test
 - (4) Top-down test
 - (5) Big bang test

(1) White box test

In the white box test, focus is placed on the internal program structure to make sure that the program structure is correct.

1) Instruction coverage

In instruction coverage, all program instructions are tested at least once. However, the truth and falsity of decision conditions are not considered.

2) Decision condition coverage

In decision condition coverage, the test is conducted for the both cases where the decision condition is True and where it is false.

3) Condition coverage

In condition coverage, the test is conducted at least once, for the combination of true and false decision conditions.

4) Multiple condition coverage

In multiple condition coverage, the test is conducted for all the combination of the true and false decision conditions

(2) Black box test

In the black box test, focus is placed on the user interface to check whether the defined functions can be performed or not.

1) Equivalence partitioning

In equivalence partitioning, test data is classified into a valid equivalent class and an invalid equivalent class, and one of the pieces of data is taken out from each class for testing.

2) Boundary value analysis

In boundary value analysis, the boundary values of a valid equivalent class are defined as test data to conduct the test.

3) Cause-effect graphing

On a cause-effect graphing, if true-or-false evaluation results differ depending on the combination of multiple conditions, the relationships of inputs and outputs are summarized in a decision table, and the test is conducted according to the conditions in the table.

4) Method for the design of test items using DOE (design of experiment)

A method for the design of test items using DOE is effective in cases where all testcases cannot be designed, because there are too many combinations of data and conditions. In such cases, the analyzed test data is extracted to conduct the test.

(3) Bottom-up test

In the bottom-up test, high-level modules are combined with low-level modules. If a high-level module is incomplete, a driver that is a module for calling lower-level modules is used to compensate for them.

(4) Top-down test

In the top-down test, low-level modules are combined with high-level modules. If a low-level module is incomplete, temporary lower-level modules called “stubs” are used to compensate for them.

(5) Big bang test

In the big bang test, all modules are integrated at once regardless of the hierarchical structure after all tests for modules are completed.

Exercises

1. Which one of the following module partitioning methods focuses on the data structure?
 - (1) Common function partitioning method
 - (2) STS partitioning method
 - (3) Jackson method
 - (4) Transaction partitioning method

2. From the following sentences, select the one that does not describe an important point on implementing module partitioning in the program design process.
 - (1) Restrict the number of dependent modules calling from one module.
 - (2) Create each module with an appropriate number of lines of code.
 - (3) For calling from module to module, avoid creating a hierarchical structure that is too deep.
 - (4) Simplify the interface.

3. Select the best description about the Warnier method in structured design of programs.
 - (1) In the Warnier method, an input-output data structure diagram is created with a focus on the data structure, and then an input data structure diagram is created.
 - (2) In the Warnier method, the functions in a data flow are divided into three groups: input, processing, and output.
 - (3) In the Warnier method, the independence of each module is enhanced by treating software as a group of data and process
 - (4) In the Warnier method, the logic design is done based on the control flow indicating the calling relationships, focusing on the program control structure.

4. Select the best description about the Jackson method.
 - (1) This is a data modeling method based on the normalization method.
 - (2) A data flow diagram and E-R diagram are created.
 - (3) Input-output data structures are represented as sequence, selection, and repetition.
 - (4) Every program is regarded as a capsule of data and operations.

5. What is the combination of factors that is most closely related to the three basic structures in the structured design method?

- a. Data
- b. Internal memory
- c. Processing
- d. Parallel processing
- e. Decision
- f. Defined processing (subroutines)
- g. Loops (repetition)
- h. Connector
- i. Terminal

Answers:

- (1) a, b, c
- (2) a, b, d
- (3) c, e, g
- (4) d, e, g
- (5) f, h, i

6. When a program is partitioned into modules in module design, it is important to focus attention on the independence of the modules. Select the correct term for the concept of measuring the independence of modules.

- (1) Module binding strength
- (2) Module accuracy
- (3) Module depth
- (4) Module difficulty

7. Which one of the following sentences is the best description about the unit test?

- (1) Prepare the correct results for the input data prior to the test.
- (2) Start with the module integration test when the program is partitioned into multiple modules.
- (3) Choose several modules from all branches of the program, and conduct the test.
- (4) Only the person who writes the program conducts the test.

8. Which one of the following sentences is the best description about testcase design in the black box method?

- (1) Extract test data from actual data at random.
- (2) Design testcases based on the external specifications of the program.
- (3) Design testcases so that all instructions of the program are executed at least once.
- (4) Design testcases based on the internal logic of the program.

9. Suppose that testcases are designed for a program that accepts integers ranging from 100 to 200 as input data, using boundary value analysis. From the group of choices listed below, which is the best combination for the testcases when values in ranges of 99 and below, 100 to 200, and 201 and above are defined for the equivalent classes?

Answers:

- (1) 100, 200
- (2) 50, 150, 250
- (3) 99, 100, 200, 201
- (4) 50, 100, 150, 200, 250

10. Select one of the sequences listed below as the sequence of test coverage from low to high for program testing with the white box test method.

- a. Test all execution routes.
- b. Test all execution statements.
- c. Execute all branches at all the branch points.

Answers:

- (1) a, b, c
- (2) a, c, b
- (3) b, c, a
- (4) c, a, b

11. Select the sentence that correctly describes a module test.

- (1) The method for testing modules individually and testing an entire program by integrating them all at once is called the “big bang” test.
- (2) Non-incremental testing is more economically efficient than incremental testing, because errors can be found much faster.
- (3) The top-down test and the bottom-up test are non-incremental tests.
- (4) Drivers are required for the top-down test, and stubs are required for the bottom-up test.

12. If module D is modified at the integration test stage for modules that have the following calling relationships, which module does not have to be tested again?

Main module A calls module E after calling module B.

Module B calls module D after calling module C.

Module E does not call other modules.

- (1) Module A
- (2) Module B
- (3) Module C
- (4) Module D

13. Which one of the following sentences is a correct description about the system test?

- (1) To conduct the system test, avoid creating testcases that place a larger load on the system than that specified in requirement definitions.
- (2) Avoid testing whether the system can recover easily from damaged files and hardware failure, because such testing may damage the entire system.
- (3) There is no reason to test the user interface and ease of operation, because they vary depending on the operating environment of the system.
- (4) The “black box” test is effective for checking the extent to which the detailed requirement definitions have been realized.

14. Which one of the following flows shows the correct sequence of the testing process in system development?

- (1) Operation test → System test → Unit test → Integration test
- (2) Operation test → Unit test → Integration test → System test
- (3) Integration test → System test → Unit test → Operation test
- (4) Unit test → Integration test → Operation test → System test
- (5) Unit test → Integration test → System test → Operation test

<http://www.vitec.org.vn>

15. Which one of the following sentences is the best description of using stubs in the testing process?

- (1) Check whether processing is performed correctly by displaying part of the register and main memory whenever an instruction is executed.
- (2) To test program modules in a top-down approach, prepare temporary lower-level modules to check the behavior.
- (3) While a program is being executed, its variables and detailed registers are tested if necessary. After corrections are made, follow-up testing is conducted as necessary.
- (4) For testing with the unit test for modules incorporated in a program, prepare temporary upper-level modules for calling the modules to check the performance.

16. Which one of the following sentences is the best description about the bottom-up test?

- (1) Test a module by integrating it with modules that have already been tested.
- (2) A "driver" module is required for calling lower-level modules.
- (3) "Stub" modules are required for those modules which have been completed yet.
- (4) Couple all modules at once.

17. Which one of the following sentences is the best description about the integration test conducted immediately after the unit test in the testing process of system development?

- (1) The test checks whether the functions have been implemented as specified in the external specifications.
- (2) The test checks whether the target processing time and target response time are met.
- (3) The test checks for any problem concerning the type and number of input/output devices and the communication devices to be connected.
- (4) The test checks the interface between the modules that are parts of the program.
- (5) The purpose of the test is to check whether multiple jobs can be executed and terminals can be connected concurrently, as described in its objectives.

<http://www.vitec.org.vn>

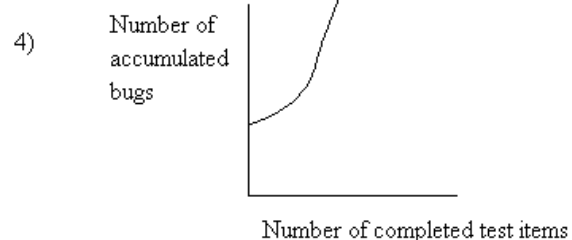
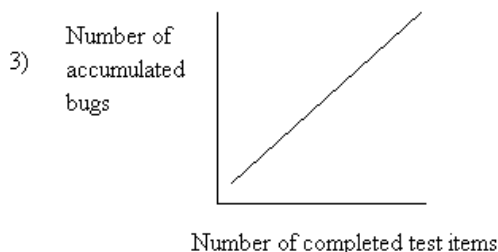
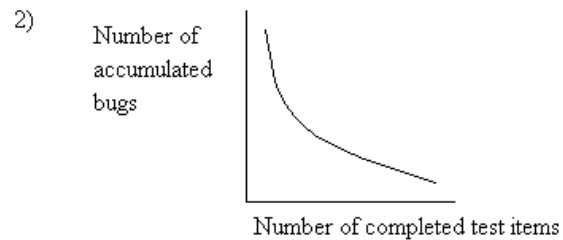
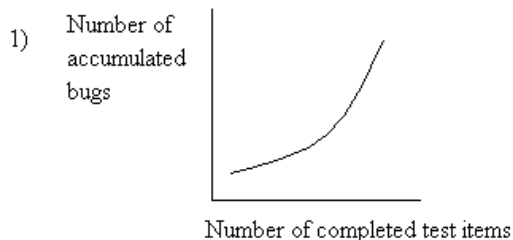
18. Suppose that an error is corrected during the unit test, but the same error occurs at the integration test stage following the unit test. The sentences listed below describe different situations of the integration test. Which situation should be reviewed in order to analyze the error?

- (1) The unit test is completed, and the specifications have not been changed in the period from completion of the unit test to the integration test. The test equipment has not been changed either.
- (2) The module test and the test for interfaces between modules have been completed.
- (3) Test data for the unit test is added to test data for the integration test.
- (4) A parallel integration test for other sub-systems is conducted, and it progresses smoothly.
- (5) A library is created separately after the unit test, and it includes the correction of any problems detected during the unit test.

19. In boundary value analysis, one of the testing methods, which one of the following data items is the most suitable for test data?

- (1) Minimum value and maximum value
- (2) Minimum value, maximum value, minimum value minus 1, and maximum value plus 1
- (3) Minimum value and minimum value minus 1,
- (4) Maximum value and maximum value plus 1

20. One of the control items for determining the status of quality control in the testing process is the correlation between the number of completed test items and the number of accumulated detected bugs. Which graph indicates that quality is maintained at a consistent level?



5 Design Review

Chapter Objectives

This chapter explains the purpose and necessity of reviews in the program design process, as well as the objectives and key points of such reviews.

5.1 Program Design Documents Review

5.2 Test Specifications Review

5.3 User Manuals Review

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

5.1 Program Design Documents Review

The purpose of reviewing program design documents is to check that individual modules and groups of modules corresponding to the functions defined in detailed design documents (program design documents) are designed in such a way that they are surely implementable.

(1) Necessity for design reviews

Unlike other types of products, software is created as logical products. This means that errors made in the design phase have a greater effect on the end result than for other industrial products. For this reason, design reviews are conducted to detect errors caused by misunderstandings and by mistakes of designers. Such errors should be detected as early as possible in the design process to prevent problems in subsequent processes.

Suppose that a particular requirement definition for a system is divided into five external specifications, each of which is subdivided into five internal specifications. A single requirement definition thus consists of a total of 25 internal specifications. The number of such specifications increases in the program specification phase. Assuming that each phase involves equivalent workloads for correcting problems found in that phase, the factor of the resulting workload in the requirement definition phase can be considered to be 1, but it is 5 in the external specification phase and 25 in the internal specification phase. This value increases further in the program specification phase. This increase in workload shows the importance of finding and correcting problems as

If a single failure or error related to the requirements definition phase is found in a design at the end of the internal design process, the corrective factor in workload can be calculated as follows:

(Total correction work) = (Correction in requirements definitions) + (Correction in external specifications) + (Correction in internal specifications)

That is, $(31) = (1) + (5) + (25)$

early as possible.

(2) Purpose of reviewing program design documents

Program design documents are reviewed for five purposes, which are listed in the following:

- To find errors in program design documents
- To find any program specifications that do not follow the program design documents
- To find inappropriate interfaces
- To check the appropriateness of maintenance measures
- To check the extent to which the criteria for creating programs are followed

(3) Checks in reviewing program design documents

Program designs and module designs undergo reviews with checks of the following items:

- 1) Degree of observance regarding methods of dividing modules and criteria for program creation
- 2) Validity of module division
 - Independence and strength
 - Interface uniformity
 - Data structure and integrity
 - Validity of assured areas for reusable modules
 - Ease of creation
 - Ease of understanding
 - Module size
- 3) Degree of reflection of items written in upper-level design documents
- 4) Validity of standard patterns and parts in actual use
- 5) Validity of making common blocks as parts
- 6) Validity of blocks in the detailed processing specifications

(4) Design review method

Design reviews involve different methods of implementation, depending on what is to be reviewed and what has priority. This section explains the concepts of current typical review methods and their application.

- 1) Walkthrough
- 2) Inspection
- 3) Prototyping method
- 4) Round robin method
- 5) Simulation method

1) Walkthrough

In this method, several testcases are provided for reviewing target procedures, and these procedures are traced and simulated in individual tests in a simulation environment in order to check the validity of a design. The review targets also include the operating conditions for the procedures, the operating environment, and the functions and roles of the procedures themselves.

Basically, the scope of a walkthrough covers the work flow of a specific sequence or direction (such as control and data flows), as well as source codes and flowcharts. This review method is inappropriate for requirements definitions and external specifications.

It is important to check the validity of input to and output from procedures, and to evaluate simulation results when a walkthrough is carried out.

2) Inspection

As the name suggests, this is a method for verifying that review targets are correct. In this method, the purpose of the test is defined and materials are prepared in advance, a review moderator is assigned, and then all related persons gather together in a meeting to carry out a review.

The scope of an inspection covers the output from all processes in the software life cycle. Such output includes planning documents, requirements definitions, external design specifications, internal design specifications, program specifications, and operation specifications. Unlike walkthroughs, inspections make it possible to review not only procedures and flows, but also partitions, configurations, and structures. Note that "code inspections" are limited in their scope of application because their targets are source codes.

It is important to define and be clearly familiar with the purpose and contents of reviews when an inspection is carried out.

3) Prototyping method

In this method, a subset of screens and sample programs are provided for the target software, and the program can be run for reviews.

Use of this method is advantageous, because specific images can be obtained, but it involves a high workload that includes advance preparations and setting up the operating environment. The scope of application covers programs that are important for user interfaces, and the method is suitable for operation reviews.

It is important to remember that reviews always target prototype programs, not actual programs, when the prototyping method is implemented. Differences between prototype programs and actual programs must be compared in terms of data volumes and the operating environment.

4) Round robin method

In this method, all review participants serve as the review moderator by turn until the entire system has undergone reviews.

This method is intended to determine the best way to implement a review, rather than carrying out the review. Its advantage is that it helps improve the morale of review participants, because everyone serves as the review moderator.

The scope of application covers all development processes. From the viewpoint of review operations, however, the method is suitable for reviewing large-scale systems.

It is important to remember that all participants act as their own review moderator when the round-robin method is implemented. Thus everyone is required to have certain skills. Also everyone is responsible for carrying out their own parts of the design process.

5) Simulation method

In this method, a simulation model is set up for a particular system, mathematical approximations are made in order to evaluate processing for an actual system. Unlike the prototyping method, which uses miniature versions of the actual system, the simulation method provides a simulation model, not an actual operating model.

The range of application targets is rather limited at present, but it is expected to expand in the future together with technological progress. Note that implementing the simulation method requires a high workload for advance preparations and setting up the environment, similar to the workload for the prototyping method.

The scope of application basically covers programs that are to be evaluated from a variety of standpoints, programs that involve strict performance requirements, and programs that require a lot of resources.

It is important to check the validity of simulation conditions and evaluate simulation results accordingly when the simulation method is implemented. Careful consideration should particularly be given to the method of implementing simulations.

(5) Implementing design reviews

Before implementing a review, the following conditions must be clarified: review organization, review standards, review procedures, and review management.

- 1) Review organization
- 2) Establishing review standards
- 3) Establishing review procedures
- 4) Review management

1) Review organization

The review organization generally consists of a moderator, review members, a review presenter, and a secretariat, but that depends on the purpose of the review as well as the scale and degree of complexity of the items under review.

a) moderator

As the moderator of the design review organization, he or she is responsible for coordinating work and making decisions throughout the review process. The moderator also gives instructions to solve problems and decides on whether or not to give approval to the completion of a review.

The moderator must have a wide range of experience and achievements, be excellent in making decisions and judgments, and be capable of taking the initiative. Ordinarily, the leader of the design group serves in this post.

b) Reviewers

As those who are engaged in actual review tasks, review committee members are responsible for reading review materials and compiling questions, problems, and proposals before the review.

Reviewers should be selected from among those who can check the items under review from a variety of standpoints (quality, productivity, and reliability). The following people are examples: from the technological aspect, person engaged in design and development; from the reliability aspect, person engaged in quality control; and from the standpoint of users, salespeople and end users.

c) Review presenter

As the person creating an item under review, a review presenter is responsible for creating review materials in advance, briefing others about the item under review, and giving answers to any questions and suggestions brought up during a review. The presenter is also responsible for having review results reflected in designs and correcting problems after the review is completed.

d) Secretariat

The secretariat is responsible for making preparations before a review and handling administrative work, so that management progresses smoothly.

The secretariat is specifically responsible for distributing a review meeting notice and review materials, selecting a meeting location, conducting a review, and creating the review report.

2) Establishing review standards

Before implementing a review, it is necessary to determine the review date and the standards to be employed for a variety of documents in the review.

a) Review timing

The review date is set at the end of the development process. This must be conducted for a final check of the current phase and for a decision on whether to proceed to the next process.

b) Review Planning document

The review planning document must include the purpose, target, timing, and location of the review, as well as the amount of workload, number of review hours, participants, and review method.

c) Review report

The review report must include the review plan, review target, review results, review points, review hours, tools used, suggestions, problems, evaluation, and actions to be taken.

d) Review checksheet

A standardized checksheet should be created for reviews.

3) Establishing review procedures

The procedures and key points for implementing reviews are explained below.

a) Preparations

The moderator and the secretariat are responsible for doing the following:

- Creating a review plan
- Creating a review schedule based on the schedule for the entire project
- Clarifying review points and establishing completion criteria
- Selecting reviewers and setting up the review organization
- Creating an meeting notice and a checklist

b) Implementation

All participants of a review implement it as follows under the direction of the moderator:

- Explanations, questions, answers, and discussions based on the checklist

- Detect problems and give directions for their countermeasures
- Compare their work against the review completion criteria, and make a decision whether the review is complete.

c) Post-review tasks

The secretariat is responsible for carrying out the following post-review tasks:

- Creating a report on review
- Managing the situation of review

4) Review management

In addition to the review, it is important to manage and follow up on the results of review. The items to be managed are listed in the following.

a) Problems brought up

Problems that were brought up during the review, and a history of the actions taken against these problems.

b) Average delay in days

Differences between the date of a review as scheduled in a planning document and the actual date of implementation in average.

c) Ratio of review implementation

Ratio of the actual number of reviews to the scheduled number of reviews

d) Ratio of review coverage

Number of lines and pages per review document

e) Ratio of achievement of bug detection

Ratio of the actual number of bugs handled as compared to the target number of bugs specified for the review

(6) Key points in design reviews

- 1) Distribute review materials in advance.
- 2) Carry out a review in short time.
- 3) Record any errors.
- 4) Do not argue about corrections.
- 5) Clarify the check items of the review.
- 6) Have only the qualified persons engage in reviews.
- 7) Exclude managers from participating
- 8) Follow up after completion of the review.

1) Distribute review materials in advance

The review presenter should distribute review materials at least a few days before the review, so that reviewers have enough time to read them carefully. These reviewers must be required to check the materials in advance to find any problems.

In reality, however, materials are often distributed to participants on the day of the

review. Good results cannot be expected from such reviews.

2) Carry out a review in short time

Because they have already examined the review materials in details distributed in advance, reviewers should point out any important errors in a short period of time. It is possible to save time by handing out design documents where minor errors, such as wrong or missing characters, have already been indicated in red. Since all development personnel are extremely busy, the time spent in each review should be up to two hours, and anything left unresolved in one review should be discussed in another review on another occasion. Another consideration is that no person should attend more than three reviews per day.

3) Record any errors

Any errors that have been pointed out during review must be written down. Review presenters should ask questions to make sure that they have a good understanding of such errors. And take care the remaining ambiguity. They must avoid pretending to understand, because misunderstandings lead to other errors in the related design documents.

4) Do not argue about corrections

Arguments about the best method to correct a problem should not be made during the review. The discussion for correction should be set in another occasion. Every effort in reviews should be focused on finding errors.

5) Clarify the check items of the review

Simply browsing through materials is not an efficient means to detect errors. It is important for reviewers to have clarified check items depending on the types of material.

6) Have only qualified persons engage in reviews

The purpose of a review is to detect errors. Avoid assigning people who assume the posts of reviewers only because of a sense of duty or obligation. Otherwise, time is wasted because pointless questions are likely to be asked.

7) Exclude managers from participating

Avoid having the managers participate in reviews. Otherwise, other participants may refrain from pointing out any errors, because they fear of a resulting negative influence on their promotion, future salary increase, and/or personal rating. Such reservations themselves are important even if there is no real reason for actual apprehension.

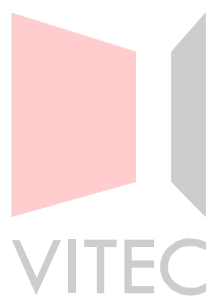
Two or more reviews are usually taken place at the same time. Managers who find it difficult to attend all of these reviews should not attend any of them.

8) Follow up after completion of the review

Errors that have been recorded undergo a follow-up review to determine the success of measures to correct them. The follow-up management continues until these errors are corrected.

After analysis, the causes and patterns of such recorded errors may be disclosed, so that other people do not repeat similar errors. However, disclosing errors requires careful consideration not to indicate problems with any individuals.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

5.2 Test Specifications Review

The purpose of reviewing test specifications is to check whether test items include all the modules corresponding to functions defined in a detailed design.

(1) Purpose of reviewing test specifications

Test specifications are reviewed for the following purposes:

- 1) To check whether test items have been defined for all modules
- 2) To check whether all test items follow the corresponding module specifications

(2) Checks in reviewing test specifications

The following checks are for reviewing test specifications:

- 1) Check of the input and output conditions for modules
- 2) Check of the validity of module functions
- 3) Check of the degree of inclusion of items in upper-level design documents



<http://www.vitec.org.vn>

5.3 User Manuals Review

The purpose of reviewing user manuals is to review user manuals conforming to the GUIs and functions defined in a detailed design (program design) and created in the system design phase. If there are any changes, correct the user manuals.

(1) Purpose of reviewing user manuals

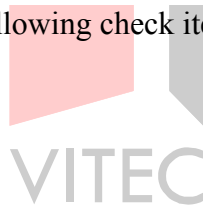
User manuals are reviewed for the following purposes:

- 1) To find any functions that have been changed in the program design
- 2) To find any screen specifications, reports, databases, or other component that have been changed in the program design
- 3) To check whether the operating procedures are appropriate

(2) Check items in reviewing user manuals

- 1) Consistency of the screen transitions, specifications, etc.
- 2) Consistency of reports, database items, etc.
- 3) Confirming implementation of functions defined in upper-level design documents

User manuals review include the following check items:



<http://www.vitec.org.vn>

Exercises

1. At the end of each design process, the designer and several person involved in the related work are engaged in a design review for the purpose of finding design errors at the earlier stage increase software quality and productivity. From the following choices, select the appropriate method for such a review.

- (1) Walkthrough
- (2) Debugging in a simulated environment
- (3) System test
- (4) Top-down test

2. From the following sentences, select the one that describes a design review carried out in the program design phase.

- (1) It is not suitable for spiral models, but effective for waterfall models.
- (2) Design reviews should include not only third parties, but also the program designer, so that specific problems can be understood, thereby enabling redesigning.
- (3) A development schedule does not generally include the schedule of reviews, because it is dependent on the progress of program design.
- (4) Review reports should be compiled with a focus on the specifics of discussions, rather than on the results of the reviews.

3. From the following sentences, select the one that correctly describes the walkthrough in system development.

- (1) Walkthrough is part of design reviews, and it is also an ISO 9000 requirement.
- (2) A development manger must attend the final review to determine whether or not the particular solution is reasonable.
- (3) Nothing is delivered to any of the participants in advance, because they must be free from preconception during reviews.
- (4) The purpose is to detect problems as early as possible in the design process. It is possible to discuss solutions at a later date.

4. Fill in the blanks with the appropriate terms chosen from the following group of choices.

System development is advanced through _____(1). That is, rough designs are first completed, and they are then divided to obtain clear definitions of components. Components are further divided into parts indicating detailed definitions. In this design method, the number of errors _____(2), because an error taking place in the first phase propagates to several locations. In contrast, design reviews are intended to minimize the number of errors in the _____(3) by preventing increases in the number of errors.

It is important to reduce _____(4) through _____(5) in the design process. Errors must be detected at the requirement definition phase, so that they do not slip into external designs, and this is important in design reviews.

Answers:

- a. multiplication of errors
- b. design phase
- c. increases
- d. stepwise refinement
- e. early detection of errors
- f. decreases

5. From the following statements, select the correct one.

- (1) The original meaning of the "walkthrough" is rehearsal. Worded differently, the purpose of walkthrough is to carefully check designs in a simulated environment. The contents of a specification or design document are checked to determine how well software runs during a production run, and to look for problems and omissions. Another purpose is early detection of errors in the design process. This therefore means that the moderator plays a major role.
- (2) Because walkthroughs require that everything is determined on the spot, managers must attend.
- (3) For inspections, document designers give brief explanations regarding the contents of the material that they deliver. The explanation should be specific enough to serve as advance knowledge when the receiver of the material reads it.
- (4) Inspections require a moderator to be trained to serve as staff member of the project manager. The moderator must be familiar with everything involved in the project, including the significance, roles, and method of carrying out inspections, as well as methods for analyzing problems and standardization. Except for the inclusion of a moderator, inspections are equivalent to walkthroughs.

6. From the following sentences, select the one that best describes walkthrough carried out in the program design phase.

- (1) If a design error is found in walkthrough, it is best to determine how to correct it.
- (2) For walkthroughs, problems regarding the best way to write the target design document and the best way for program designers to work should be determined before personnel undergo training.
- (3) For walkthroughs, it is preferable to assign specific roles, such as the maintenance person and the standardization person, to review participants, so that they can give opinions from their individual viewpoints in such roles.
- (4) System users should attend walkthroughs make it possible to check whether the system meets user requirements.

<http://www.vitec.org.vn>

7. From the following list of methods, select the best method of carrying out a walkthrough-based review.

- (1) We carried out a one-day review to make it possible to accurately point out even detailed problems.
- (2) We carried out a walkthrough to review the source code, but we did not do so for the function specifications, because its effect was considered to be minor.
- (3) We asked top management to join our team and solved all problems one by one in the review stage.
- (4) We asked end users to join the review in the requirement and design phases, which is early in the design process.

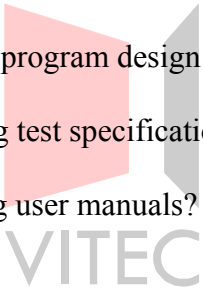
8. From the following sentences, select the one that best describes inspection as a method of design review for assuring the quality of software.

- (1) The program designer gathers all development-related people, and they follow all the steps of the program to be reviewed in accordance with the operating flow of the running program.
- (2) Materials to be reviewed are prepared in advance and a review moderator is assigned, and all development-related people gather together in a meeting to conduct a systematic review.
- (3) Part of the target software is prototyped for actual operation.
- (4) All review participants serve as the moderator by turn until the entire system has undergone reviews.

9. What is the purpose of reviewing program design documents? Name five objectives.

10. What is the purpose of reviewing test specifications? Name two objectives.

11. What is the purpose of reviewing user manuals? Name three objectives.



<http://www.vitec.org.vn>

Answers for No.3 Part 4 Exercises

Answers for Part 4 Chapter 1 (Procedures for Program Design)

(No exercises for this chapter)

Answers for Part 4 Chapter 2 (Program Design Criteria)

1. (2) (4)
2. b. Warnier method
3. (4)
4. (2) Information strength
5. a.
 - a. functional strength
 - b. time strength
 - c. linkage strength
 - d. procedural strength
6.
 - a. External coupling
 - b. Data coupling
 - c. Control coupling
 - d. Common coupling
7. (1) skeleton
(2) function
(3) standard logic
(4) ☐ access part ☐ check part ☐ text part
8. (1) Heuristic reuse
(2) Planned reuse
9. Classification of candidate parts
Determining the parts interface
Determining function outline and usage restrictions
Determining information in the parts
10. A lot of time is required to determine and analyze the existing systems.
Differences in program quality occur depending on the technique used by the person correcting a program
11. (1) Source (2) Transform (3) Sink
12. consecutive (sequential)) selection ☐ repetitive
13. (1),(3)

Answers for Part 4 Chapter 3 (Creating of Program Design Document)

1 C

2 a. 2, b. 3, c. 1

3 (3)

4 (4)

5 C

6 (1)

7 C

Answers for Part 4 Chapter 4 (Creating Module Specifications and Test Specifications)

1. (3) Jackson method

2. (1) Restrict the number of subordinate modules calling from one module.

3. (1) In the Warnier method, an input-output data structure diagram is created with a focus on the data structure, and then an input data structure diagram is created.

4. (3) Input/output data structures are represented as sequence, selection, and repetition.

5. (3) c: Processing, e: Decision, g: Loops (repetitive)

6. (1) Module binding strength

7. (1) Prepare the correct results for the input data prior to the test.

8. (2) Design testcases based on the external specifications of the program.

9. (3) 99, 100, 200, 201

10. (3) b, c, a

11. (1) The method of testing the module individually and testing the whole program by integration them all at once, is called the Big Bang test.

12. (3) Module C

13. (4) The “black box” test is effective for checking the extent to which the detailed requirement definitions have been realized.

14. (5) Unit test → Integration test → System test → Operation test

15. (2) To test program modules in a top-down approach, prepare temporary lower-level modules to check the performance.

16. (2) A "driver" module is required for calling lower-level modules.

17. (4) The test checks the interface between the modules that are parts of the program.
18. (5) A separate library is created separately after the unit test, its information is used for correcting errors during the subsequent unit tests.
19. (2) Minimum value, maximum value, and values in single increments
20. (4)

Answers for Part 4 Chapter 5 (Design Review)

1. (1) Walkthrough
2. (2) A limit is defined on the number of dependent modules that can be called from a single module.
3. (4) The purpose is to detect problems as early as possible in the design process. It is possible to discuss solutions at a later date.
4.
 - (1) d. stepwise refinement
 - (2) c. increase
 - (3) b. design phase
 - (4) e. early detection of errors
 - (5) a. multiplication of errors
5. (4) Inspections require a moderator to be trained to serve as staff member of the project manager. The moderator must be familiar with everything involved in the project, including the significance, roles, and method of carrying out inspections, as well as methods for analyzing problems and standardization. Except for the inclusion of a moderator, inspections are equivalent to walkthroughs.
6. (3) For walkthroughs, it is preferable to assign specific roles such as the maintenance person and the standardization person to review participants, so that they can give opinions from their individual viewpoints in such roles.
7. (4) We asked end users to join the review in the requirement and design phases, which is early in the design process.
8. (2) Materials to be reviewed are prepared in advance and a review moderator is assigned, and all development-related people gather together in a meeting to conduct a systematic review.
9.
 - (1) To find errors in program design documents
 - (2) To find any program specifications that do not follow the program design documents
 - (3) To find inappropriate interfaces
 - (4) To check the appropriateness of maintenance measures
 - (5) To check the extent to which the criteria for creating programs are followed
10.
 - (1) To check whether test items have been defined for all modules

(2) To check whether all test items follow the corresponding module specifications

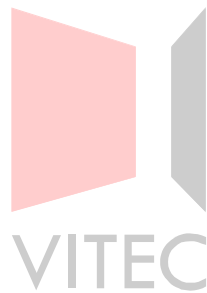
11.

(1) To find any functions that have been changed in the program design

(2) To find any screen specifications, reports, databases, or other component that have been changed in the program design

(3) To check whether the operating procedures are appropriate

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Part 5

SYSTEM OPERATIONS AND MAINTENANCE

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

1 System Operations

Chapter Objectives

Understanding an outline of system operations is given so that

- 1 Understanding the outline of system operations
- 2 Understanding planning for system operations
- 3 Understanding what are managed during system operations

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

1.1 What are managed in System Operations?

System Operations refers to the phase where a system is in actual operation. System operations should be performed based on operation service standards and carried out with the following management items

- Resource management: Effective use of system resources requires correct knowledge of system's resources. Therefore it is important to manage resources.
- Problem management: Since there are no systems with no problems, it is necessary to manage problems. Standard procedures to be taken in the event of a problem should be defined and followed.
- Facility management: This is important to keep the system safe and healthy condition not affected by any power failure or disasters etc.
- Security management: This is required to make sure that use of the system and access to system's data is strictly controlled to avoid security problems.
- Performance management: Collecting and analyzing performance data makes it possible to ensure that system's service meets the requirement and to prevent possible failure.
- Cost management: Consideration should be given to cost management so as to minimize the total cost of ownership of the system.

<http://www.vitec.org.vn>

Summary of the Management items in System Operation

Type of management	What are managed	Purpose
Resource Management	<ul style="list-style-type: none"> • Hardware • Software • Data • Network 	To use system resource effectively
Problem Management	<ul style="list-style-type: none"> • Identification and reporting of a problem • Analysis of a problem • Recovery from a problem 	To find a problem as early as possible, and solve it as speedy as possible
Facility management	<ul style="list-style-type: none"> • Power Supply • Air Conditioning • Disaster Prevention • Crime Prevention • Storage 	To operate a system safely and securely
Security Management	<ul style="list-style-type: none"> • Accounts • Permissions • Records of system usage 	To prevent unauthorized use of a system and information leakage
Performance Management	<ul style="list-style-type: none"> • Response times, turn-around times • Throughput • CPU usage rate etc. 	To check whether services offered to users meet the operational requirement
Cost Management	<ul style="list-style-type: none"> • Initial Cost • Running Cost 	To ensure that system operational costs meet the organization's requirement

1.1.1 Resource Management

Resource management occupies an important place among the operation management items. To use system resources effectively, it is necessary to have correct knowledge about the resources required for operations.

(1) Hardware Resource Management

Hardware resource management indicates the management of computers and their peripheral devices. Hardware resources must be maintained properly.

Actually, operation managers should confirm that resources are used effectively by checking how hardware equipment is being used. In addition, if unevenness in use is found, the rearrangement of hardware resources must be considered to better distribute the load. The basic concept is to use resources effectively to increase the rate of operation of each hardware device equally.

The life of hardware resources should also be taken into account. Generally, equipment in use for more than a certain period becomes likely to cause problems more frequently. Considering the replacement of devices by checking the problem-generating rate is an important item in managing hardware resources as well.

In hardware resource management, following data are collected and their results should be evaluated and analyzed regularly:

- Response performances
- Processing capabilities (the number of processed items per hour)

<http://www.vitec.org.vn>

(2) Software Resources

Software resource management indicates the management of programs running on a system. Contrary to hardware resources, many parts of software resources are invisible. Therefore, keeping to pre-determined standards is important for management.

① Library management

The items to be conducted in library management include:

- Where libraries (including backup libraries) are physically stored should be clarified.
- The version data in a library should be managed (The co-existence of new and older versions of the same software should be avoided).
- Libraries should be protected (for security and from computer viruses).

② Prevention of illicit use

The following measures should be taken to prevent illicit use of software resources:

- Whether illegitimate copying is made or not should be managed.
- How software resources are used should be managed.

(3) Data Resource Management

Data resource management indicates the management and adjustment of data used in a system from overall organizational viewpoints. Users themselves manage much of the data. However, the objective of data resource management is to manage this data systematically, and to select important data for special management aiming at securing perfection.

In many of the systems today, databases are used. Therefore, database resource management on an operational basis should be covered.

The following should be performed in data resource management:

- Securing perfection
- Ensuring security (prevention of illegitimate use)
- Systematic management of data resources

In addition, data audit, conducted to investigate and analyze methods to carry out data resource management and to perform the management at a higher perfection level, is an important item as well.

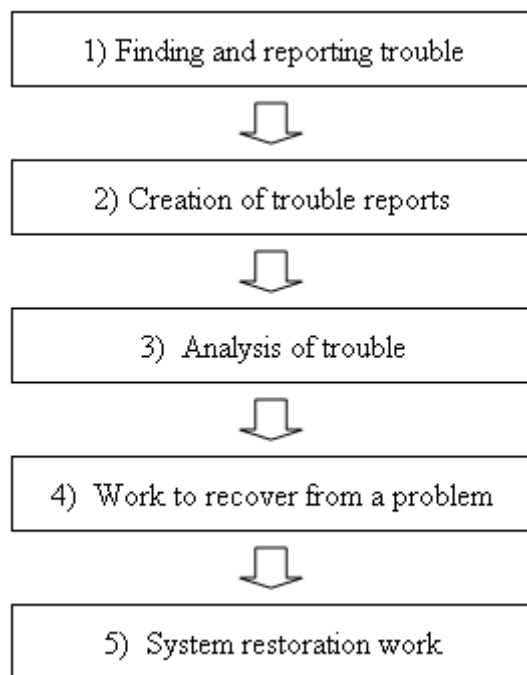
(4) Network Resource Management

It is no exaggeration to say that computer systems today are always connected to networks. In network resource management, equipment making up networks, such as CCU (Communication Control Unit, DCE (Data Circuit Terminating Equipment), etc. is managed. Network resource management is basically performed under hardware management. However, as for communications circuits, the handling of re-routing circuits, in addition to backbone circuits, is involved. Therefore, a management organization including telecommunications carriers must be established.

1.1.2 Problem Management

It is desirable that no problem occurs in system operations. However, in actuality there is no system where no problem occurs. Therefore, an important aspect is how speedily a system can be restored after a problem has occurred. Problem management is for measures to be taken when a system problem occurs.

Standard procedures to be taken when a problem occurs are as follows:



hệ thống tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

After the restoration work is completed, it is necessary to evaluate whether the measures taken were appropriate and to reflect the results to measures to be taken thereafter.

(1) Finding and Reporting Trouble

The earlier a problem is found, the smaller the effect on the system as a whole and the easier the measures to be taken will be. Therefore, to find a problem as early as possible, it is important to take always care about data collected in resource management and to grasp ordinary operating situations.

In addition, the establishment of an organization allowing trouble found to be reported swiftly to the manager in charge of problems is important.

(2) Creation of Trouble Reports

Trouble reports must be produced immediately when a report that trouble has occurred is received. The trouble report has two uses. One is for analyzing the problem and taking measures swiftly as well as correctly; the other is as one of the statistical data utilized to prevent problems in advance.

In addition, at this time, the area to be affected by the problem is identified and notification is made to the sections concerned. In particular, problems that are considered to greatly affect system operations need support for the work of restoration from the problem. Therefore, immediate notification is essential.

(3) Analysis of Trouble

To investigate the causes of a problem, analysis is conducted into the situations where the problem occurred. To investigate the causes, logged data at the time when the problem occurred and dump listings are generally used. Many of the causes of hardware problems are found with these means. However, for software-related problems, finding the causes sometimes takes much time. In such a case, temporary measures may be taken, with a search for the real causes left until later.

The following methods are used as measures, if log and dump listing data are insufficient to find the causes:

- The situation when the problem occurred is re-generated artificially.
- A measure is taken that, if a similar trouble occurs again, allows detailed data to be obtained.

Making the causes of problems clear enables preventing the occurrence of similar problems again.

(4) Work to Recover from a Problem

Based on the causes of problems, the methods to restore the system are determined and restoring operations are performed. The methods are various depending on the problem causes.

① Hardware trouble

- Back-up devices are put in use.
- Problem devices are separated. Then, they are repaired (for example, by the manufactures of these devices).

② Software trouble

- The software is re-activated.
- The older version software is restored in place of the present version software.

- Correction is made for the present software.

③Data trouble

- Data that caused the problem is removed or modified.
- Roll-back or roll-forward operations are performed.

In addition, keeping trouble reports on how the restoration work was performed enables them to be used as documents to consider measures for similar problems occurring later.

(5)System Restoration Work

A system whose operations are stopped is restored. It is checked to see whether the work to recover from a problem enables the system to function normally. Then, the system is restored with ordinary services offered again.

Depending on the ways used to recover from a problem, the following cases should also be taken into consideration:

①Hardware trouble

The following should be considered, if recovery is made by use of backup hardware:

- Performances in comparison with the main hardware
- Restoration work when the repair of the main hardware is completed

②Software trouble

The following should be considered, if recovery is made by use of an older version of the software:

- Drop in the level of functions available (such as services available)
- The limitation of use in consideration of response capabilities

③Data trouble

The following item should be considered, if data is corrected:

- Corrected data is consistent with that not corrected.

System restoration work continues until all the functions of a system are fully restored.

1.1.3 Facility Management

To operate a computer system, the facility and equipment of the computer center must be maintained above a certain quality level. As for the installation standards for facilities and equipment, the Economy, Trade and Industry Ministry (formally the International Trade and Industry Ministry) has put out a guideline entitled "Standards for Information System Safety Measures."

In investigating operations (designs) of equipment, the three aspects of reliability, expandability and cost must be taken into account. Concerning expandability, designs must include margins in consideration of recent advancements in technology and variations in external factors.

Facilities to be considered in system operations include the following:

- Power supply-related facilities
- Air conditioning facilities
- Disaster prevention facilities
- Crime prevention facilities
- Storage facilities

(1) Power Supply-Related Facilities

Computer systems cannot be operated without power supply. Therefore, facilities continuously supplying stable power must be provided.

① Primary power supply

Ordinarily, a commercial power supply system is used for the primary power supply. However, a mechanism to secure stable power supply is required. Actually, to handle quality degradation in supply voltage from a commercial power supply system, measures to maintain supply voltage at a constant level, including the use of an AVR (Automatic Voltage Regulator), are taken.

② Non-utility electric facilities

A non-utility electric facility is used as a backup when the primary power supply has a problem (for example, due to a power outage). The facility may be used as the primary power supply if a commercial power supply system is unavailable. However, ordinarily it is used only when a problem occurs. Therefore, inspections to check the storage of power generation fuel and failure of the facility must routinely be conducted.

③ UPS (Uninterrupted Power Supply)

UPS (uninterrupted power supply) is temporarily used from the time when power supply from the primary power supply is stopped to the time when power supply from a non-utility electric facility is started. UPS also plays the role of compensating short breaks in the primary power supply.

④ Others

Battery

Ordinarily, batteries are charged for each device while power is being supplied. In some cases, batteries charged with battery chargers or other means are provided separately from the devices.

Power distribution facilities

It is also important to regularly inspect distribution boards, breakers, surge protectors and other related equipment.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

(2) Air Conditioning Facilities

Most equipment used in a system generates heat in operation. Even though the heat from each device is small, the total amount of heat generated in a computer room, where many devices operate, becomes significant. Ordinarily, temperatures and humidity enabling stable operation are specified for each piece of system equipment. Therefore, providing air conditioning facilities is important to maintain the proper conditions.

There are two types of air conditioning facilities: the centralized type and the distributed type. With the centralized type, a large air conditioner is installed in a computer center, controlling the temperature of the room as a whole. This type of air conditioning facility requires a small installation space. However, once the air conditioning facility fails, the operations of the center as a whole may be stopped. With the distributed type however, an air conditioner is equipped with each device. Therefore, this type of facility offers high expandability and also the merit that danger due to power failure is distributed. However, this type is slightly inferior in the efficiency of facility investment to the centralized type.

Water-cooled cooling equipment may be used for a facility generating lots of heat, such as a large host computer. Today, much equipment generates less heat and can be used in ordinary office environments. Consequently, the use of air conditioning facilities will likely be limited to center facilities.

(3) Disaster Prevention Facilities

Stable operations of a system require the use of disaster prevention facilities provided in case of disasters, such as fire, earthquake, etc. Typical disaster prevention facilities include the following:

① Fire prevention facilities

Fire prevention facilities include automatic fire alarms (the heat-sensing type and the smoke-sensing type) and fire-extinguishing equipment. Sprinkler systems cannot be used as fire-extinguishing equipment in center facilities. Therefore, fire-extinguishing equipment using halide or carbon dioxide is used instead. The type of fire extinguishers for electricity-caused fires is different from that for ordinary fires. Therefore, fire-extinguishing exercises should be conducted routinely.

② Anti-earthquake facilities

As a direct measure for earthquakes, fixing equipment with anchor bolts, or topple-preventing equipment, using floor vibration-absorbing equipment, is used. However, what is feared in earthquakes is secondary fire. To prevent secondary fire, equipment to shut down power supply by sensing vibration is also used.

③ Emergency announcement equipment

When a disaster occurs, the role of emergency announcement equipment announcing correct information is important.

Disaster prevention facilities are not used in ordinary times. Therefore, conducting inspections to check that the facilities work in emergencies without failure is important. At the same time, it is desirable to provide a mechanism (such as a disaster drill) to check measures to be taken in emergencies as well as communications measures and contact routes.

(4) Crime Prevention Facilities

Crime prevention facilities are for protecting systems from man-made menaces such as information stealing and destructive activities. Since man-made menaces are brought in by outside persons, the facilities are, in many cases, for preventing the entrance of such persons.

① Entry/exit control equipment

Entry/exit control equipment allows managing persons to enter and exit from a center facility. To identify whether a person is entitled to enter or depart from a center facility, the authentication of a person is performed by means of a password or passwords, a card for enter/exit, such as a magnetic card or IC card, a fingerprint, a voiceprint, or a retina pattern.

It is also important to manage entry/exit logs (names of persons entering or exiting from a center and their entry/exit times. This measure is effective to prevent man-made menaces by persons who have a genuine right to enter a center).

② Monitor

Monitors are equipped in the vicinity of a center facility to find suspicious persons and to prevent those persons from entering the facility. Monitors are also installed inside a facility to check whether there is a person behaving suspiciously. Formerly, more than one monitor had to be installed, because stationary monitors were used. However, many monitors using a wide lens and having a variety of functions, including direction-moving and zooming functions, have recently become available. Therefore, a sufficient effect can be expected with a small number of monitors installed.

(5) Storage Facilities

Even though the highest level security functions are used to prevent data from being stolen, they are meaningless if backup or some other data are taken out easily. Sufficient care must be taken about facilities to store backup data and other data outputted from a system.

A storage facility is for storing important data. Therefore, the facility must be equipped with disaster prevention functions, such as fireproof and waterproof functions, in addition to security functions provided to prevent stealing.

Basically, a storage facility must be built at a place far apart from its center facility. In addition, entry/exit control must be conducted in the same way as for center facilities.

<http://www.vitec.org.vn>

1.1.4 Security Management

The objectives of security management are to prevent the illicit use of a system and information leakage in system operations.

Security management includes the following:

- User management
- Access management
- Use management

The typical technology used includes:

- Encryption

(1) User Management

The management of users of a system constitutes the base for all of security management. To use a system, it is necessary to have a user ID issued by the system administrator. The user ID is information given to each user with the objective of enabling checking the authentication of a user's right to use a system and grasping the user's use status.

To manage user ID, the following should be considered:

- Only a necessary number of user IDs should be issued. User IDs having become unnecessary should be deleted as soon as possible to prevent the illicit use of those IDs.
- Sharing user ID must be prohibited. Different ID must be issued, even to each member of a development team.
- The amount of authority set to user ID should be as small as possible.

Passwords to authenticate users must also be managed. The password is most widely used as a means to authenticate a user.

To manage passwords, the following should be considered:

- Use of a password which cannot be easily guessed should be encouraged. The manager must check passwords regularly, and ask for a change if a password allowing an easy guess is found.
- A mechanism to change passwords regularly should be provided, for example, by limiting the term of validity for them.
- The security level of files recording passwords must be increased by use of encryption or other means. In addition, reference of the files by general users must be prohibited.

(2) Access Management

In access management, different access rights can be set for each user. By so doing, even for users of the same system, information and/or services available can be set differently, based on a user's position.

(3) Use Management

In use management, collecting the use data of a system increases the security level of the system as a whole.

Data to be collected include the following:

- User name (user ID)
- Use date
- Use time (log-in/log-out times)
- Terminals used

- Systems used
- Resources used

(4) Encryption

Corporate secrets handled by systems should be encrypted to prevent them from being altered. It is desirable for data resident in the systems to be encrypted as well.

1.1.5 Performance Management

One objective in managing the performance of system operations is to check whether services offered to users meet the required standards or not. At the same time, finding local performance degradation leads to prevention of a systems failure.

Items to be managed include:

- Response times and turn-around times
- Throughput
- Available time (starting time and ending time)
- The maximum number of terminals operating
- Quality of output data
- SLA (Service Level Agreement) of networks

Collecting and analyzing these data enables one to determine whether performance expected for a system is maintained.

Attention should also be paid to user complaints, because it may potentially involve a performance degradation, which cannot be identified with simple measurement.

In addition, changes in external factors (such as an increase in the number of transactions) may make it impossible to maintain system performance unless some measures are taken. Performance management should also be used routinely to enable the prediction of such a situation occurring, and to allow the making of proposals for new equipment and for changing up software versions.

1.1.6 Cost Management

Systems operation naturally means cost. Cost management is very important for enterprises seeking to increase profit, and due consideration must be given to it.

Therefore, collecting cost data correctly, and cutting unnecessary costs as much as possible are important in running the systems.

TCO (Total Cost of Ownership) used for cost management is classified into the following two items.

(1) Initial Cost

The initial cost is one-time cost in the system installation phase, and not generated after the system operation is started.

The types of initial costs include:

- Purchasing cost of equipment (system equipment, network equipment, terminals, etc.)
- Purchasing cost of software (basic software, software packages, etc.)
- Software development cost

These costs are not always generated. For example, equipment-purchasing cost is not required for rental equipment. Therefore, careful consideration should be given to these costs for installing a system.

(2) Running Cost

The running cost becomes necessary once a system is in operation. It is a persistent cost which is generated periodically and fixed.

The types of running cost include:

- Rental cost of equipment (system equipment, network equipment, terminals, etc.)
- License fees of software (basic software, software packages, etc.)
- Maintenance cost (for hardware and software maintenance)
- Maintenance cost of equipment (communication live cost, heating and lighting expenses)
- Expendables cost
- Personnel expenses

The running cost is classified into the fixed cost that is generated constantly, and the variable cost and expenses which depends on conditions. The variable costs and expenses should be managed on a monthly basis, to find the differences between the actual and budget data. Then, proper measures should be taken, if necessary.

A charging method may be introduced in which users (user organizations) pay the running cost. The charging method includes the proportional allocation method, in which cost is allocated depending on the amount of use, and the base allocation method in which the amount to be allocated is determined depending on the ratios of profits gained by the use of a system.

1.1.7 Other Operation Management

(1) System Operation

In view of the use, systems must be operated with consideration to the following:

- Operation manuals, describing system operation methods and operation procedures, are provided.
- Job control statements (job scheduling) are provided to enable the automatic processing of jobs.
- Integrity measures for inputting data are provided in addition to the management of inputting and outputting data.

(2) System Operation Tools

Various system operation tools are used to make smooth and easy system operations possible.

Typical system operation tools include:

- Tools for automatic operations
- Monitoring tools
- Diagnostic tools.

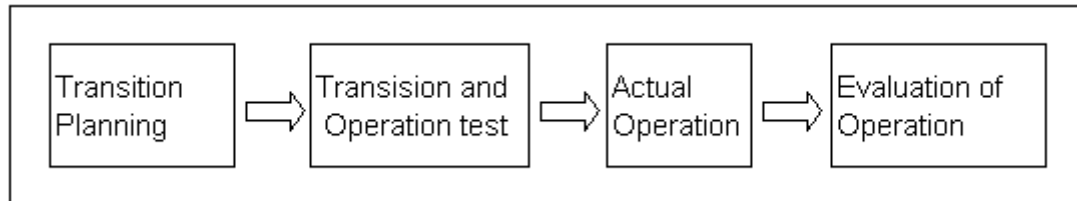


<http://www.vitec.org.vn>

1.2 System Transition Steps

System Transition includes a series of activities needed for a newly-developed system or an updated system to start actual daily operation.

Preparation for system transition should start in the early phase of a system development so that the transition to a new or an updated system goes smoothly.



System Transition Steps

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

(1) Transition Planning

Create such plans as hardware installation plan, software installation plan, data migration plan etc. needed for the system transition. What to be done in which order by which organization (or staff) etc. should be clarified and summarized in the transition master plan, and details should be defined in transition manuals. What requires transition depends on the differences between the current system and the new (or updated) system.

(2) Transition and Operation Test

Migrate the system from the development and testing environment to the actual operational environment. Confirm that the system operates without problems in the actual operational environment by performing operation tests.

(3) Actual Operation

Bring the system into actual operation.

(4) Evaluation of Operation

Evaluate the system operation in terms of CPU usage, memory usage, network traffic, response time, throughput etc, in comparison with corresponding data in the existing system.

1.3 Operational tests

Operation tests are performed using actual data and based on actual operation schedules. The following table shows operation test items. Operational test results are examined to make sure that the system runs in an actual operational environment, with expected accuracy, operability, reliability, efficiency etc. Operation tests are completed when this becomes sure, then the system is to be brought into actual operation.

Regular operations	<ul style="list-style-type: none">• daily startup and shutdown of the system• everyday business workflow• day-end processing, weekly processing, monthly processing, quarterly processing etc.
Ad-hoc operations	<ul style="list-style-type: none">• non-regular processing
Operation hours	<ul style="list-style-type: none">• at early morning• daytime• nighttime
Others	<ul style="list-style-type: none">• network connection• data exchange with another system• backup procedure

<http://www.vitec.org.vn>

1.4 Evaluation of System Operations

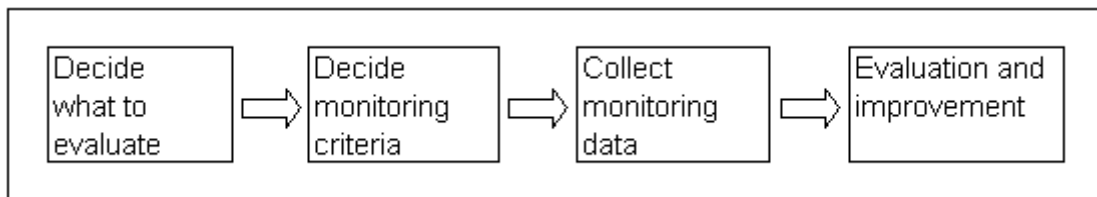
It is important to collect and analyze data on hardware resource activity ratios for the purpose of system evaluation.

System operation and management engineers are responsible for evaluation of system operation. Collecting and analyzing information in order to evaluate user satisfaction is necessary. Answers to the following questions are to be found by doing so.

- To what extent does the system satisfies the requirement specifications?
- Requirement specifications are appropriate?

Some problems resulting from increase system load and functional enhancement may occur. The evaluation of these problems also requires the periodical collection and analysis of data.

Following tasks are included in the evaluation of system operation.



(1) Decide what to evaluate	e.g. Hardware, Software, Network etc.
(2) Decide monitoring criteria	e.g. Performance by response time, throughput, etc. Reliability by number of errors, MTTR(Mean Time to Recover), money amount etc.
(3) Collect monitoring data	e.g. Quantitative data to be collected automatically by some tools, qualitative data to be collected by questionnaire (feedback form)
(4) Evaluation and Improvement	e.g. Analysis of performance data, identify problems, take some measures

RASIS (Reliability Availability Serviceability Integrity Security)

Reliability (To what degree the system is reliable)	MTBF (Mean Time Between Failure)
Availability	$MTBF/(MTBF+MTTR)$
Serviceability (How it is easy to repair the system in case a failure occurs)	MTTR (Mean Time To Repair)
Integrity	e.g. data integrity in database
Security	Ability to ensure that the system is secure

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

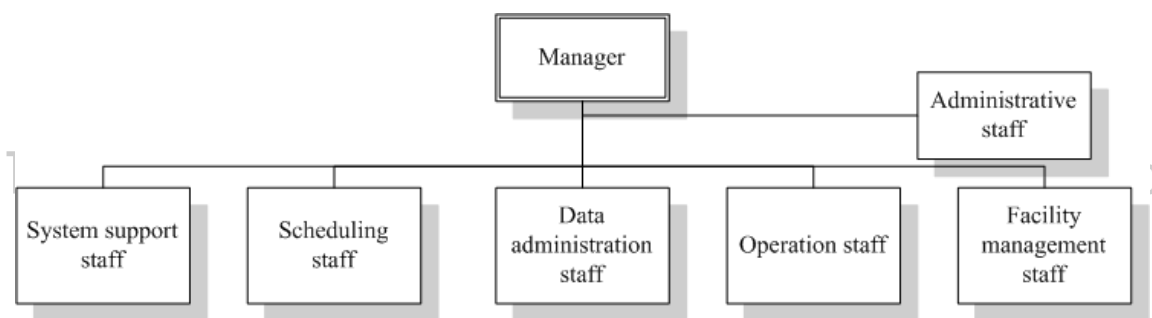


<http://www.vitec.org.vn>

1.5 Operation Organization

The operation organization means the organization responsible for performing operation work when system. This organization is prepared so as to achieve effective operations. This organization may be a part of an IT division or may be an external entity as a result of outsourcing operation work.

An example of operation organization is shown below.



System support staff	monitor system operation status, install hardware, software etc.
Scheduling staff	develop system operation schedules
Data administration staff	in charge of data administration
Operation staff	perform daily operations such as keeping track of system operation, doing failure recovery
Facility management staff	in charge of managing facilities

2 System Maintenance

Chapter Objectives

Understanding an outline of system maintenance

- 1 Understanding the outline of system maintenance
- 2 Understanding maintenance tasks and procedures
- 3 Understanding types of maintenance

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

2.1 What is Maintenance

2.1.1 Importance of Maintenance work

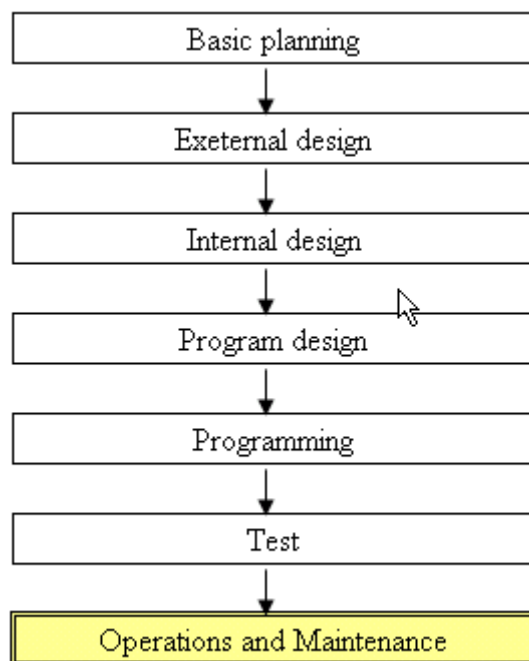
Actual operations of a new (or an updated) system are started after operation tests.

However it is common and inevitable to make changes on the system so that errors are to be fixed or modification requests from the system's user can be satisfied etc.

Maintenance activities must continue for many years. The importance of software maintenance cannot be overestimated.

Maintenance is often thought of as the final activity in the software lifecycle. It is the final discrete activity and also the activity that lasts the longest. Many things have to be considered to prepare for and enter maintenance.

The following figure shows “operations and maintenance” activity as the final activity in the waterfall model.

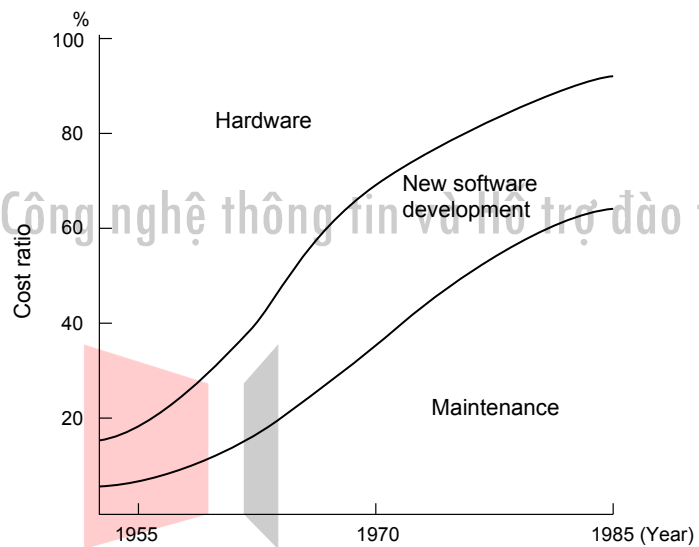


2.1.2 Maintenance Cost

It is widely recognized as the highest cost phase of the software life cycle with estimated costs of between 60% and 80% of the total software budget. Good maintenance requires configuration management and good documentation.

The following figure shows Boehm's transition curves. This curve indicates the yearly ratios of maintenance cost occupies in the total development cost.

Boehm's transition curves



<http://www.vitec.org.vn>

2.2 Planning for Future Maintenance during development

An often-overlooked activity is planning for future maintenance. Since maintenance activities must continue for many years, it is necessary and important to plan for maintenance in advance. This is related to the following activities: design, coding, and documentation.

Maintainability cannot be added after the development is completed; instead, maintainability comes from good design and coding standards and documentation.

Modules	should be designed with high modularity, high cohesion, and low coupling, as well as good use of abstractions and information hiding
Program codes	should be written conforming to certain coding standards.
Project documentation	The accurately reflects the current state of all parts of the project (requirements, design, coding, and test cases and results) should be ready, making it possible to do maintenance tasks by maintenance personnel.

It is necessary to define documentation standards in the development project to make sure that the project documentation is meaningful. It is also important to reflect any changes to the documentation. It is not a matter of whether documentation exists or not, but the matter of whether meaningful and latest documentation is available or not for maintenance.

CASE tools with round-trip engineering capabilities help developers ensure that the project documentation reflects the latest state of all parts of the project.

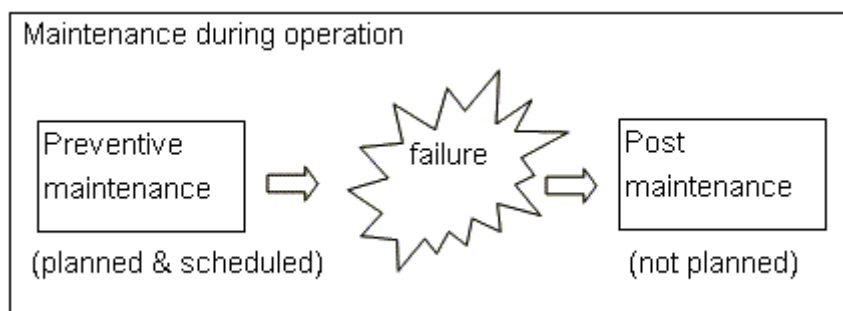
<p>Planning for future maintenance during development</p> <ul style="list-style-type: none">- Carefully designed modules- Codes written conforming to the coding standards for better readability- Updated (meaningful) documents

2.3 Types of Maintenance during system operation

2.3.1 Preventive Maintenance and Post Maintenance

Types of maintenance during system operation include the preventive maintenance and post maintenance. Those are summarized in the following table.

Preventive Maintenance	<p>Performed to prevent failures</p> <ul style="list-style-type: none"> • Daily maintenance, without stopping system operations • Scheduled maintenance, conducted at pre-determined schedules. During this maintenance, the system operation is temporarily stopped.
Post Maintenance	<p>Performed when a failure has occurred or seems to occur.</p> <ul style="list-style-type: none"> • When an unusual situation is found in system operations, or when recovery from a failure is needed.



(1) Preventive Maintenance

Preventive maintenance is performed to prevent failures, and the maintenance plans are devised in advance. Preventive maintenance, allows for a planned acquisition of maintenance personnel, and enables efficient maintenance.

Preventive maintenance includes:

- Daily maintenance
- Scheduled maintenance

① Daily maintenance

The daily maintenance is conducted every day to check the states and performances of equipment making up the systems. Ordinarily, measures allowing checks to be

conducted without stopping system operations are provided. Remote maintenance is widely in use in systems that involve locations far apart.

② Scheduled maintenance

The maintenance is conducted at pre-determined intervals.

This maintenance is a comparatively large-scale work in which system operations are either stopped or performed with alternative equipment. Since operations are stopped temporarily the work must be carried out quickly.

(2) Post Maintenance

Post maintenance is conducted when a failure has occurred, or may occur.

The maintenance is not a work which is scheduled in advance. Therefore, the acquisition of maintenance personnel for the work may be difficult. In case such a situation occurs, having extra emergency maintenance personnel is desirable.

Post maintenance includes the following:

- Tentative maintenance
- Emergency maintenance

① Tentative maintenance

Tentative maintenance is conducted only when an unusual situation is found in system operations, such as a sudden sharp deterioration in performance or an unusual sound is detected coming from the operating equipment.

The objective of this maintenance is to take measures before a failure actually occurs.

② Emergency maintenance

Emergency maintenance is conducted to make a recovery from the failure when it occurs. The work results in stopping systems temporarily without any notification in advance.

The necessity of such emergency maintenance work should be avoided as much as possible. Therefore, if the number of emergency maintenance works tends to increase, maintenance plans, including the reconsideration of the system as a whole, must be devised and executed.

Conducting maintenance naturally requires maintenance personnel. However, retaining the personnel may pose a problem with the cost. Therefore, maintenance work may be entrusted to external companies.

Benefits gained by entrusting the work to external companies with a maintenance contract include:

- The cost can be reduced.
- Personnel problems can be solved, because consideration for the personnel becomes unnecessary.
- It eliminates the need for the 24-hour duty service from employees.
- Detailed maintenance by specialists is expected.

However, the demerits include:

- Problems may be created from security aspects.
- Maintenance personnel may be changed at the discretion of the company contracted with.
- Not all the maintenance personnel assigned are talented.
- Raising issues on the operation is not done.

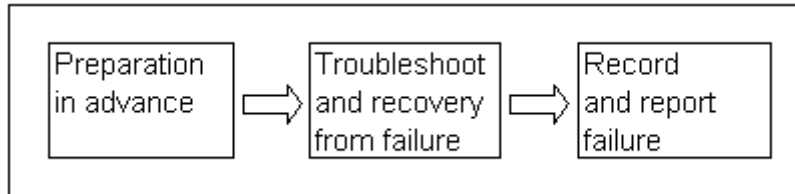
In concluding a maintenance contract, various aspects, including companies to be entrusted with the work, and maintenance mode (resident or on call), must be considered.



<http://www.vitec.org.vn>

2.3.2 Recovery from Failure

Following steps should be taken in preparation for and in the event of a failure.



(Reference: Concerning the second step, 1.1.2 Problem Management has further information.)

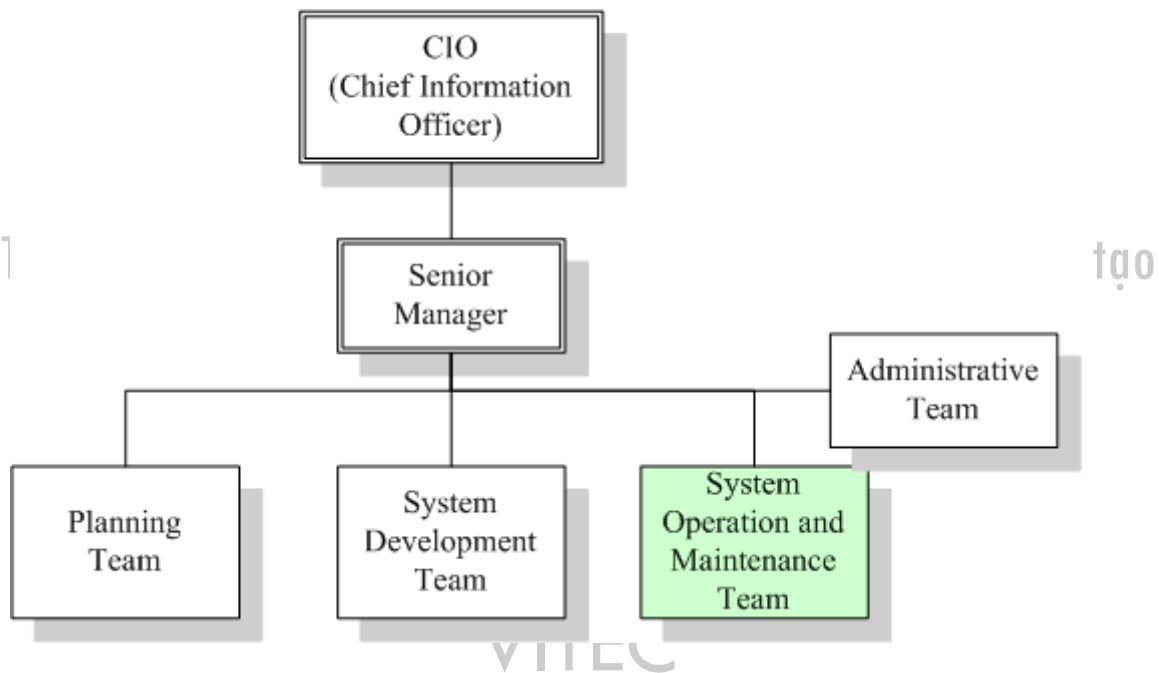
The summary of above steps is shown in the following table.

Preparation in advance	<ul style="list-style-type: none">• Monitoring system operation so as to detect any failure as quickly as possible• Decide maintenance organization e.g. who is the maintenance administrator, the first person to contact in the event of failure, and other contact persons• Prepare documentation (troubleshooting manuals, trouble report form etc.)
Troubleshoot and recovery from failure	<ul style="list-style-type: none">• Collect information about the failure, identify the failure location, and do recovery from the failure (e.g. system restart, application restart, database recovery etc.)
Record and report failure	<ul style="list-style-type: none">• Fill in and submit the trouble report that includes: what failure occurred when and where, how long the system stopped operation, reason, recovery measures taken etc.

2.3.3 Maintenance Organization

The maintenance organization means the organization responsible for performing maintenance work when maintenance is needed for a system. This organization is prepared to make sure that maintenance tasks go smoothly.

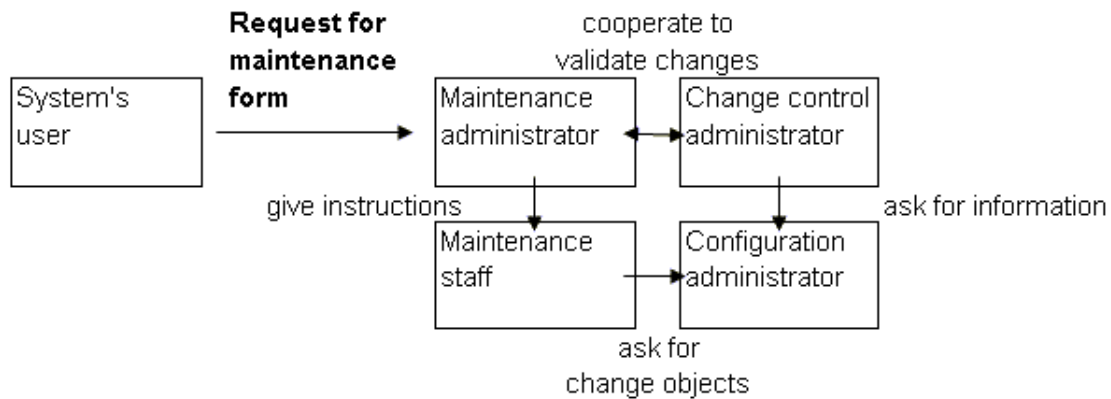
The maintenance organization may be a part of an IT division as shown in the following example. It may be an external entity as a result of outsourcing maintenance work.



To establish a maintenance organization, particular considerations must be given to the following:

- Conducting the maintenance work as well as making development efforts at the same time is impossible. The development organization and the maintenance organization must be established separately.
 - It is desirable that the person in charge of maintenance be involved in the system development.
 - Outputs from a system development include documents such as design documents, in addition to the system itself. Therefore, a person or persons in charge of documentation management must be assigned to create some unified management of these documents.
 - A person (maintenance administrator) liaison between users and the maintenance organization must be assigned. The number of such liaisons must be limited to one.
- If a developer performs maintenance work at his own discretion, a serious problem may ensue. Therefore, a group must be organized to perform maintenance work.

A maintenance organization example is shown below.



The roles played by each member are described below.

Maintenance administrator	The first person to contact when maintenance work is needed. He or she receives requests from users.
Maintenance staff	They perform actual maintenance tasks by given instructions from the maintenance administrator.
Change control administrator	This administrator examines the validity of changes for maintenance, keeps track of the changes made on software and documentation.
Configuration administrator	This administrator manages versions of systems.

Any request for maintenance should be submitted in a written format and added to the list of requests. Requests are carefully examined for validation. Then, once decided to make changes based on the request, actual changes are made under the control of configuration management.

Actual works are as follows:

- ① Users request maintenance to the maintenance administrator.
- ② After receiving the request, the maintenance administrator checks, discusses the validity, and makes final decision together with the change control administrator and person responsible for the system.
- ③ Once the execution of the maintenance work is decided on, the maintenance administrator gives instructions to the maintainers to carry out the work. In addition, each system modification is documented and managed.

- ④ Maintenance workers report their work progress status to the maintenance administrator at appropriate times and their work completion to the maintenance administrator and configuration administrator.
- ⑤ The configuration administrator updates the version of the targeted system.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

2.3.4 Hardware Maintenance and Software Maintenance

Hardware Maintenance

Special attention to the following has to be paid in the hardware maintenance.

- Reliability of hardware
- Balance of hardware components
- Availability of expendables (such as printer toners) or spare parts

① Reliability

Maintenance items in view of reliability include the following:

- Important phenomena monitoring

Important events in systems are monitored.

- Failure tendency monitoring

In this monitoring, data are sampled for a long period to analyze failure tendencies.

- Specified monitoring

This monitoring is for important issues in consideration of the reliability characteristics of equipment making up a system.

② Performance balance

If a system is in use for a long time, the replacement of old devices with new ones may cause an imbalance between them. If the balance is made worse, devices with insufficient performances may affect the performance of a system as a whole. Therefore, performance balance between devices must be checked.

③ Inventory status

Expendables necessary for the systems and the inventory of parts to be replaced or repaired must be checked. It is important that these items are checked and inspected routinely to prevent being forced to stop operating until parts arrive.

Software Maintenance

According to the “IEEE Standard for Software Maintenance,” software maintenance is "Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment".

Software maintenance is not easy. Some maintenance organizations report that between 20 percent and 50 percent of changes to a program causing additional errors. These statistics make configuration management and change control extremely important.

Different types of software maintenance are summarized below.

Adding or improving functionalities	make changes so that the system can cater for functional requirement changes
Adaptive maintenance	make changes so that the system adapts to a changing environment
Perfective maintenance	make changes so that the system works better
Corrective maintenance	make changes so that errors are to be fixed

Adaptive maintenance

Adaptive maintenance refers to changing software to adapt to a changing environment. The hardware environment that software is developed for can change dramatically over a long-time lifecycle. In adaptive maintenance, it is necessary to go back over the requirements, identify those requirements that are affected by the environment, identify the design changes necessary, and affect those changes in the code. After the changes are made, retesting is required. For adaptive maintenance, retesting is often extensive and frequently more time consuming than the original system. This occurs because the new system has many of the parts of the original system but also has many new and modified parts. Testing must ensure that the original parts still work, that the new and modified work as required, and that functionality of the original system is not inhibited by the new or modified parts. This is extremely difficult. Regression testing is required and for large adaptive maintenance efforts, complete retesting, to a new series of tests, is required.

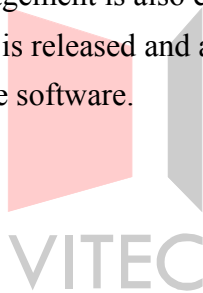
Perfective maintenance

Perfective maintenance refers to taking software that works and modifying it to be better. The system's users may not be aware of any changes. In perfective maintenance, no problem exists, so great care must be taken not to introduce any. Careful testing of perfective changes is required. In addition, the results from static and dynamic test tools can be examined to ensure that perfective changes are not being performed on seldom-used pieces of code.

As noted above, maintenance causes more errors than development. No matter how small the change, some testing should be performed.

Corrective maintenance

Corrective maintenance refers to the process of fixing errors in the system. These errors can be trouble reports (from the system's users) or internal (discovered by developer side). The most important part of corrective maintenance is updating and maintaining the documentation so that future modifications to the system will have accurate documentation. Configuration management is also extremely important, because it determines when the fixed program is released and also ensures that all users in the field are running the proper version of the software.



<http://www.vitec.org.vn>

2.3.5 Discarding the system

At the end of its lifecycle, a system is to be discarded. For example, after the transition from an existing system to a new system, the existing system may be discarded. To be more specific, the system itself, or some software, or data may be discarded. What are to be discarded and become unavailable should be announced in advance to the system's users. Careful consideration should be taken so that unauthorized usage of discarded things will never happen. Especially when discarding data, utmost care should be taken so that any confidential or important data will not be disclosed.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Part 6

PROJECT MANAGEMENT

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

1 Project Management

Chapter Objectives

Understanding an outline of project management

- 1 Understanding the outline of project management
- 2 Understanding project management standards
- 3 Understanding project management activities

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

1.1 Overview of Project Management

1.1.1 What is Project Management?

Planning for a software development project in advance and managing the project based on the plans is a must for the project's success.

A Project

- is focused on the completion of a unique end product
- has a finite duration
- has defined goals and strategic plans that should optimally reflect and complement the goals and plans of the organization.
- is performed based on an organizational structure that is often similar to the functional structure of the organization.
- can be of any size and involve any number of resources.

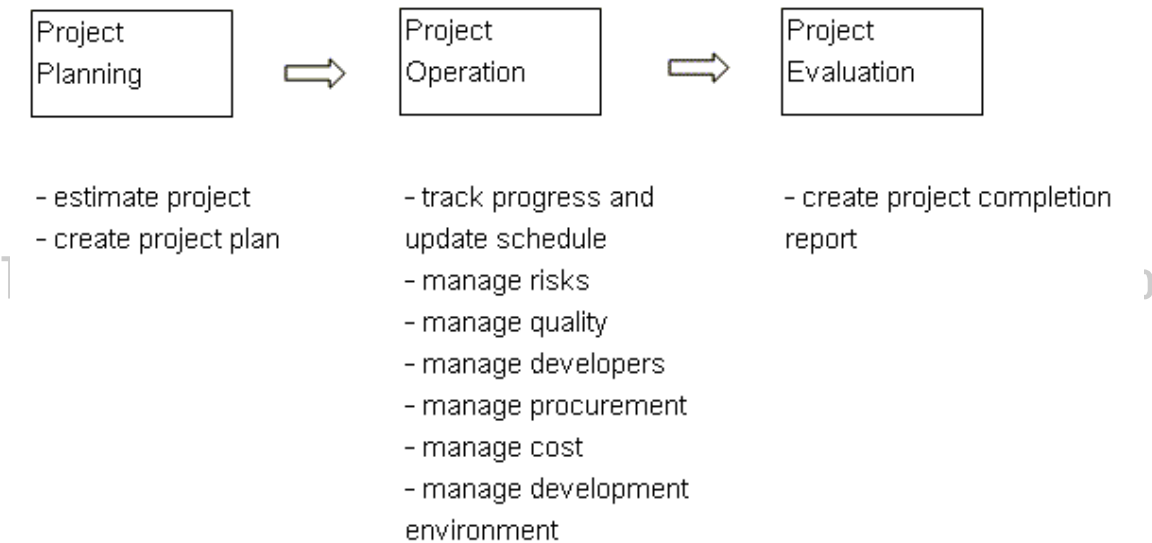
Project management can be defined as the collection of procedures, practices, technologies, skill, and experience necessary to successfully manage a project.

The individual responsible for managing a project is called the project manager. He or she is typically responsible for the planning, implementing, controlling, and reporting of status on a project.

A software lifecycle model defines the stages or phases through which a software development project moves and is a primary element of an organization's software process. The lifecycle model is a view of the activities that occur during software development. It guides a project from the inception of an idea, through development, to final deployment and eventual retirement. It provides a view of how software is developed. Mature software organizations have defined software processes that guide the utilization of resources and the implementation of the project.

The major project management functions of planning, organizing, staffing, directing, and controlling are completed in varying forms and degrees for every project. These functions are satisfied by specific activities, including developing the project plan, defining the project work breakdown structure, staffing the project, developing the project schedule, and tracking and controlling the project.

The following image illustrates the project management steps.



<http://www.vitec.org.vn>

1.1.2 Project management items

Project management items can be classified into the following. Those are to be planned, clearly described in project plans, and managed throughout the process.

- Project itself
- Developers
- Procurement
- Development Costs
- Development environments

Items	What to be managed
Project itself	Scope, Schedule, Quality, Risks
Developers	Organizations, Human Resources, Communications
Procurement	What/Where to procure, Delivery Quality, Cost
Development costs	Personnel costs, Equipment costs, Other expenses
Development environments	Hardware, Software and others

Managing Project itself

As for the project itself, the following have to be clarified and managed.

Scope	Project scope, Products
Schedule	Tasks, Schedules
Quality	Specifications Satisfying the user's requirement, Programs Satisfying Specifications
Risks	Identification, prioritization and management of risks, Measures taken against risks

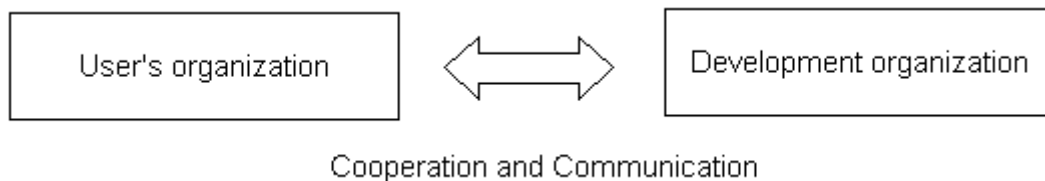
Managing Developers

Organizing and managing a development team based on the development requirement is important in a project.

Development organizations	Project Manager, Project Leaders, Staff members
Human resource development	Develop skills needed as developers
Communication plans	Ways of communications between project members

Development organizations

To bring a software development project to success, solid organizations (the development organization and the user's organization) should be established. Two organizations should cooperate and have close communications with each other.



The user's organization is the organization who will actually use the system developed. Without the user organization's active involvement in the development project, successful completion of the project cannot be expected.

The development organization is responsible for doing actual development. Number of persons within a development organization varies from time to time within the development process. It is common to perform development tasks up to design phase within the internal development organization, and ask contractors (third parties or external software development companies) to perform programming and testing. Then after testing by contractors, the internal development organization conducts attesting work.

① Types of the development organization

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

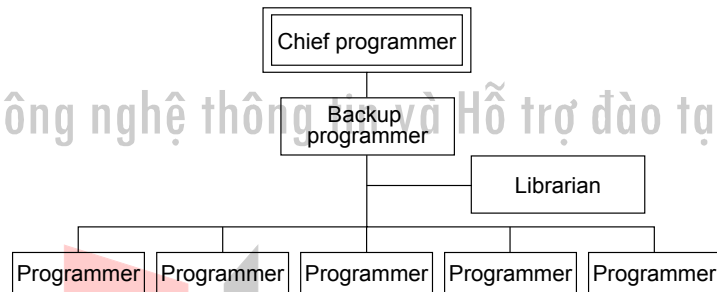
There are three typical types of project teams.

- Chief programmer team
- Specialist team
- Hierarchical team

a. Chief programmer team

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

A chief programmer team example



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

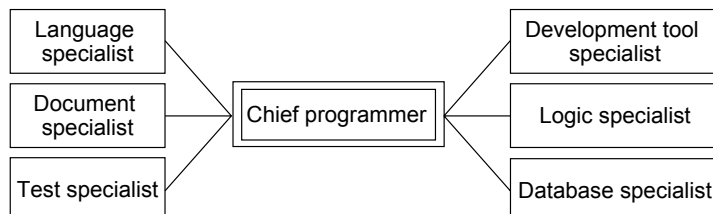
< Characteristics >

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

b. Specialist team

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

A specialist team example



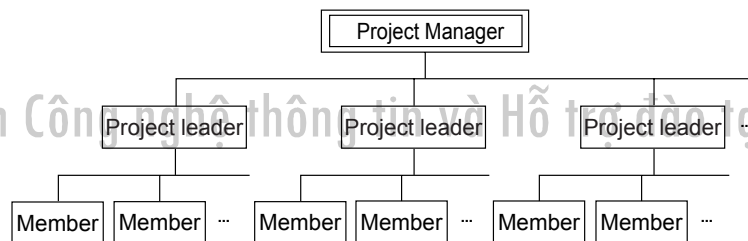
<Characteristics>

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

c. Hierarchical team

The hierarchical team is composed of a project manager, several project leaders and workforce members.

A hierarchical team
example



<Characteristics>

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

② Team members' roles

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

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

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

<Roles>

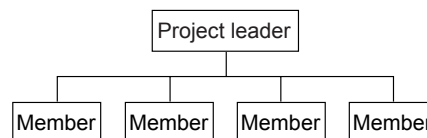
- Planning, drafting plans, execution and evaluation of projects.
- Communication with user and related organization (including parties outside the company)
- Vitalization of project work (including personnel deployment and transfer of power)
- Other management work

b Project leader

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

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

A sub-project team example



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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo
c.Member

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

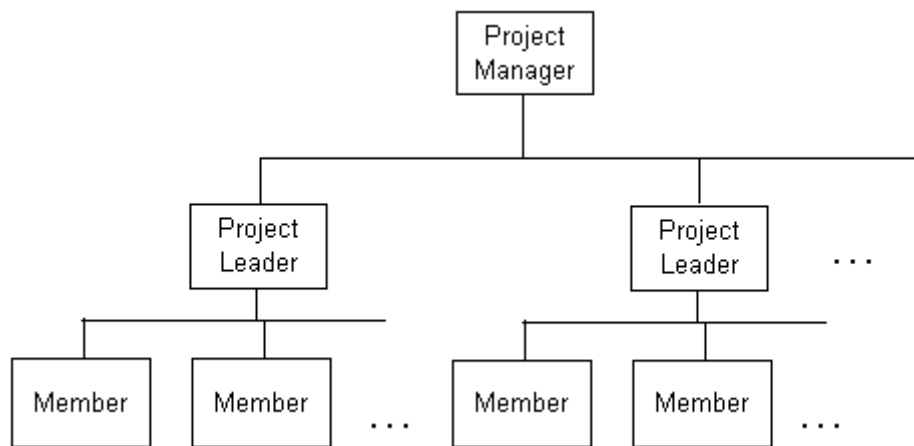


<http://www.vitec.org.vn>

Types of development organizations

Chief programmer team	<ul style="list-style-type: none"> • a team with a relatively small number of members • composed of the chief programmer, the backup programmer, the librarian and programmers
Specialist team	<ul style="list-style-type: none"> • a variation of the chief programmer team • composed of the chief programmer, technical specialists (e.g. database specialist) and programmers
Hierarchical team	<ul style="list-style-type: none"> • suitable for relatively large-scale system developments • composed of a project manager, several project leaders and workforce members

Figure An example of a development organization (a hierarchical team)



- The project manager is the person fully responsible for the development project
- The project leader assists the project manager and leads a sub-project team
- Members perform actual development work

Managing Procurement

In software development, it is common to work with some third parties (or contractors from external organizations) instead of doing everything by in-house staff members. In addition, procurement of hardware and software may be also needed.

What to procure	Decision of what to procure
Where to procure	Decision of where to procure considering “delivery, quality and cost”
Contract types	Selection of contract types

Procurement strategy (the manner or approach to securing the necessary resources for carrying out the work of the project) should be established early on in the project planning phase, since procurement strategy decisions will affect the design of the work breakdown structure. Procurement strategy may range from completing all of the work in-house, or a single complete contract to a number of separate trade contracts under the direction of a coordinating (e.g. construction) manager.

Managing Development Costs

Software project managers are responsible for controlling project budgets so, they must be able to make estimates of how much a software development is going to cost.

Development costs include the following. Among these, personnel (effort) cost is the most difficult to estimate and control, and has the most significant effect on overall costs.

Personnel costs or Effort costs	Costs of paying software engineers
Equipment costs	Costs of hardware needed for development
Other expenses	Traveling cost, training cost etc.

Software cost estimation is a continuing activity which starts at the proposal stage and continues throughout the lifetime of a project. Projects normally have a budget, and continual cost estimation is necessary to ensure that spending is in line with the budget.

Effort can be measured in staff-hours or staff-months (Used to be known as man-hours or man-months).

Managing Development Environments

Which development environments (e.g. within the customer's site) will be used under what conditions should be also clarified and managed.

Hardware	Computers, network equipment etc.
Software	OS, development tools, middleware etc.
Others	others

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

1.2 International Standards

There are a number of international standards established in the field of software project management. The ISO 9000 series of standards is one example. ISO 9000 is a set of documents dealing with quality systems that can be used for quality assurance purposes. These documents specify quality system requirements for use where a contract between two parties requires the demonstration of a supplier's capability to design and supply a product.

1.2.1 PMBOK (Project Management Body of Knowledge)

The Project Management Institute (PMI), founded in 1969, is dedicated to the promotion of excellence in project management and its establishment as a unique discipline and independent profession. PMI establishes project management standards, provides seminars, educational programs and professional certification. Its efforts in defining project management resulted in the creation and subsequent refinement of the Project Management Body of Knowledge (PMBOK).

PMI publishes "A Guide to the Project Management Body of Knowledge PMBOK Guide 2000 Edition".

The currently available PMBOX Guide is its 2000 edition that supersedes the previous 1996 edition. Its first edition was published in 1987.

The PMBOK describes the sum of knowledge within the profession of project management. It includes knowledge of proven practices that are widely incorporated as well as knowledge of innovative and advanced practices that have seen limited use. The PMBOK provides both theoretical and operational definitions of project management. These definitions are based on a basic structure or framework for project management activities.

The following table shows brief descriptions of the nine knowledge areas.

Knowledge area	Description
Project Integration Management	Project integration management includes those aspects of project management that promote and ensure proper coordination between various project elements. It includes project plan development, execution, and change control
Project Scope Management	Project scope management includes processes for completely determining all required work for a project. It also provides mechanisms for determining work that is out of scope for the project. Scope management includes project initiation and scope planning, definition, verification, and change control
Project Time Management	Project time management includes processes for consistently completing projects on time. It includes activity definition, sequencing, duration estimating, and schedule development and control
Project Cost Management	Project cost management includes processes for consistently completing projects within approved budgets. It includes resource planning, cost estimating, cost budgeting, and cost control
Project Quality Management	Project quality management involves establishing processes for ensuring that a project meets the requirements for which it was performed. It includes quality planning, quality assurance, and quality control
Project Human Resource Management	Project human resources management includes processes for effectively using the people resources available to a project. It includes organizational planning, project staffing, and project team development
Project Communications Management	Project communications management includes processes for efficiently generating, collecting, distributing, storing, and controlling project information. It includes communications planning, information distribution, and performance reporting
Project Risk Management	Project risk management includes processes for identifying, analyzing, and controlling project risk. It includes risk identification, quantification, response development, and response control

Project Procurement Management	Project procurement management includes processes for efficiently acquiring services and products from organizations external to the project. It includes procurement planning, solicitation, source selection, and contract administration and closeout
--------------------------------	--

One of PMI's primary services is accreditation and certification in the field of project management. This certification applies to professionals and not organizations. PMI's approach is that people, namely project managers, develop and implement the processes that drive projects. Therefore, educating project managers on correct principles will result in improved processes and successful projects.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

1.2.2 Capability Maturity Model (CMM)

In 1986, the Software Engineering Institute (SEI), a federally funded research and development center that is under contract to Carnegie Mellon University, began development of the Capability Maturity Model (CMM).

The CMM presents a description of recommended practices in several key process areas of software development that have been proven to enhance software process capability. The CMM provides organizations with guidance on how to control their software development and maintenance processes and increase their maturity and capability to produce quality software. It has been used as a guide in selecting process improvement strategies by determining current process maturity and identifying the most critical issues for software quality and process improvement. The intent of the CMM is to help organizations focus on a limited set of activities and work aggressively to achieve them, thereby improving the organization-wide software process capability. The popularity and success of the CMM in improving software processes has made it the preferred model for software development and maintenance processes.

Maturity Levels

According to the CMM, software organizations fall into one of five levels of process maturity: initial, repeatable, defined, managed, and optimized. A maturity level is a well-defined evolutionary plateau toward achieving a mature software process.

- Level 1 has unpredictable results, with ad-hoc approaches. (Organizations at this level typically have a lot of rework, customer's frustration and cost and schedule overruns)
- Level 2 organization achieves repeatable performance.
- Level 3 organization shows improving performance within an organization.
- Level 4 organization is characterized by improving organizational performance.
- Level 5 organization has rapidly reconfigurable organizational performance.

Primary goals for most organizations are to achieve Level 3 maturity. Organizations at Level 3 can consistently produce reliable software on time.

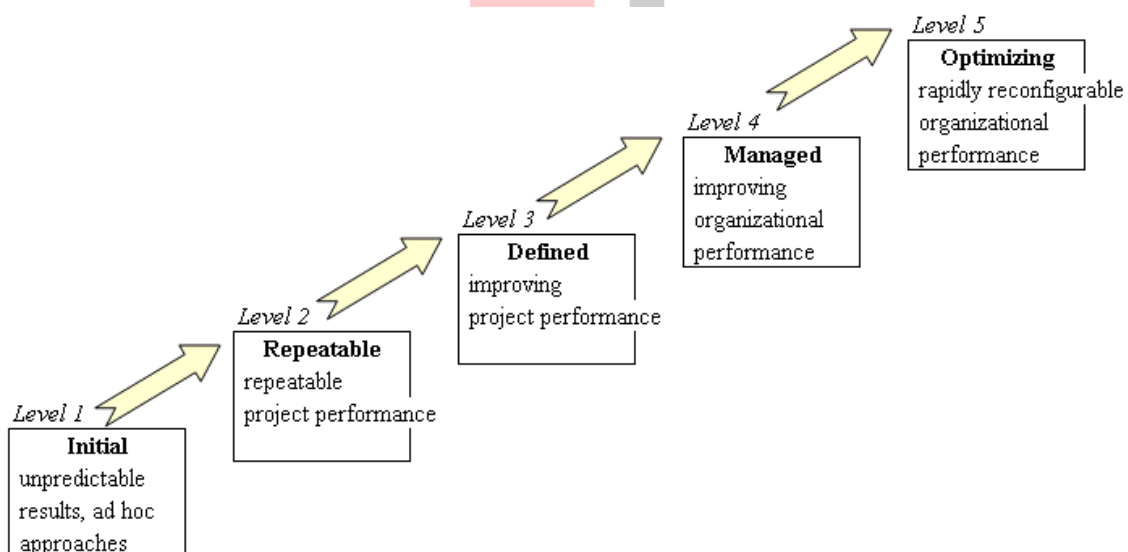
Key Process Areas

Key process area (KPA) identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important.

Focusing on and improving specific key process areas within each level enables organizations to steadily improve their software capability maturity.

The following table shows the associated key process areas for maturity levels.

Maturity Levels		Key Process Areas (KPA)
1	Initial	
2	Repeatable	Requirements management; Software project planning; Software project tracking and oversight; Software subcontract management; Software quality assurance; Software Configuration Management
3	Defined	Organization process focus, Organizational process definition, Training program, Integrated software management, Software product engineering, Intergroup coordination, Peer reviews
4	Managed	Process measurement and analysis; Quality management; Defect prevention
5	Optimizing	Technology innovation, Process change management

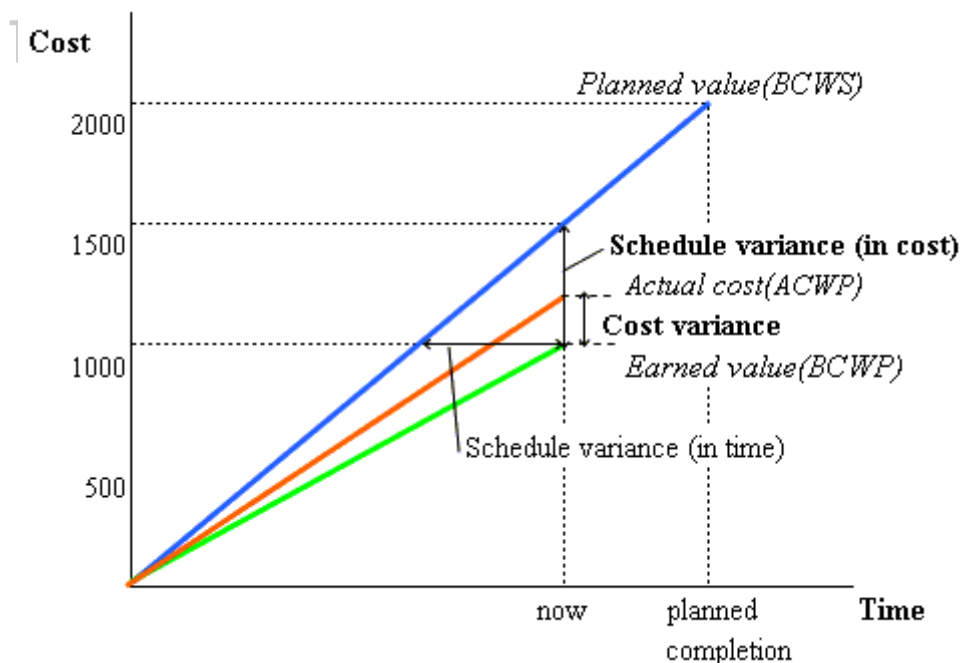


1.2.3 Earned Value Management (EVM)

The Earned Value Management is a valuable tool in the management of all projects, software projects in particular.

It has a focus on its percent complete position against its (100 percent) defined scope. It requires a master project schedule (detailed, bottoms-up performance plan, also called a project master schedule (PMS)), measurement taken against one's own plan, periodic forecast of the final expected results, based on actual performance results.

The following is an example of a Project Performance display graph (show cost against time.)



ợ đào tạo

Planned value (PV)	the physical work <u>scheduled</u> , and the authorized budget to accomplish the scheduled work (also called “BCWS budgeted cost of work scheduled”)
Earned value (EV)	the physical work <u>performed</u> , and the authorized budget for the work (also called “BCWP budgeted cost of work performed”)
Actual cost (AC)	total costs incurred in accomplishing work during a given time period (also called “ACWP actual cost of work performed”)
Schedule variance (SV)	difference between the planned cost and earned value
Cost variance (CV)	difference between the actual cost and earned value

Example

In a certain 4-week project, total budget is 2,000\$.

At the end of the 3rd week, 2/3 of the scheduled work has been completed.

- Planned value is 1,500\$.
- Earned value is 1,000\$.
- Actual cost (from cost ledger) is 1,200\$.
- Schedule variance is $1,500 - 1,000 = 500\$$
- Cost variance is $1,200 - 1,000 = 200\$$

If this project continues at present cost efficiency rate of $1,000/1,200 = \text{approx. } 83\%$, estimated cost at completion is $1,500/0.83 = \text{approx. } 1,800\$$.

Planning for performance measurement

1. Define the project scope using a WBS diagram.
2. Decompose the project scope into measurable tasks, each with an estimated value, and assign responsibility.
3. Prepare a detailed plan and schedule for the entire major critical tasks.
Do a few iterations to complete PMS (project master schedule) fully supported by Critical path method.
4. Estimate required resources (provide time estimates, assign people and equipment, and indicate task dependencies)

1.3 Project Planning

The purpose of the project plan is to provide a foundation for execution of the project. Other uses of the plan, such as communication are important, but are secondary in nature. Planning should be completed prior to any work being performed on project tasks. The project plan contains both general and detailed planning information for the project tasks.

The project plan is an essential tool for the project manager to manage a project. It includes the following.

- Project scope, goal, assumptions etc.
- Plans for managing quality, cost, risks etc.
- Products of the project (when to submit in what format)
- A definition or breakdown of the work comprising the total project
- Tasks and schedules (WBS, milestones, detailed schedules)
- cost estimates for all labor resources and other resource types
- Plans for project organizations
- Procurement plans

Development of the project plan is an iterative process. As information is refined or clarified the individual elements of the plan are updated. Additional plans that address support and related activities are added to the project plan. These include plans for requirements definition and specification, risk management, configuration management, software quality assurance, testing, etc. Once the plan is completed, peers and subject matter experts should review it. It is then submitted to upper management for approval. Approval of the plan allows the project to be baselined and clears the way for execution.

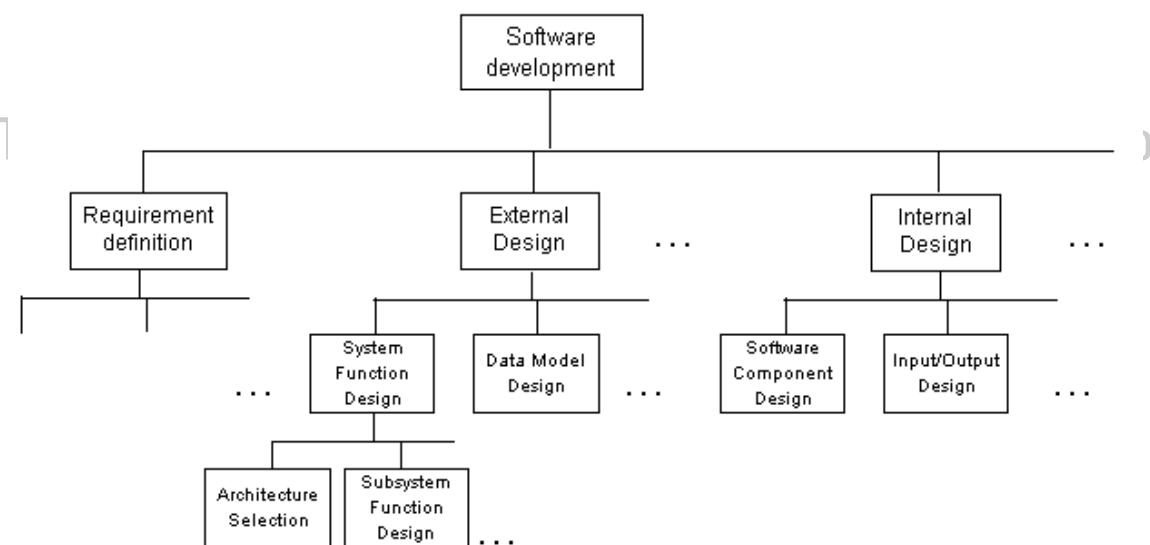
1.3.1 Work Breakdown Structure (WBS)

A work breakdown structure (WBS) is a family tree composed of different levels of Tasks. This can be used to clarify the structure of activities within a project.

- The level deciding the major structure of a project
- The work level that constitutes the framework of each phase
- More detailed work levels

Every task within a work breakdown structure should have an associated cost estimate and schedule.

Figure An Example of WBS



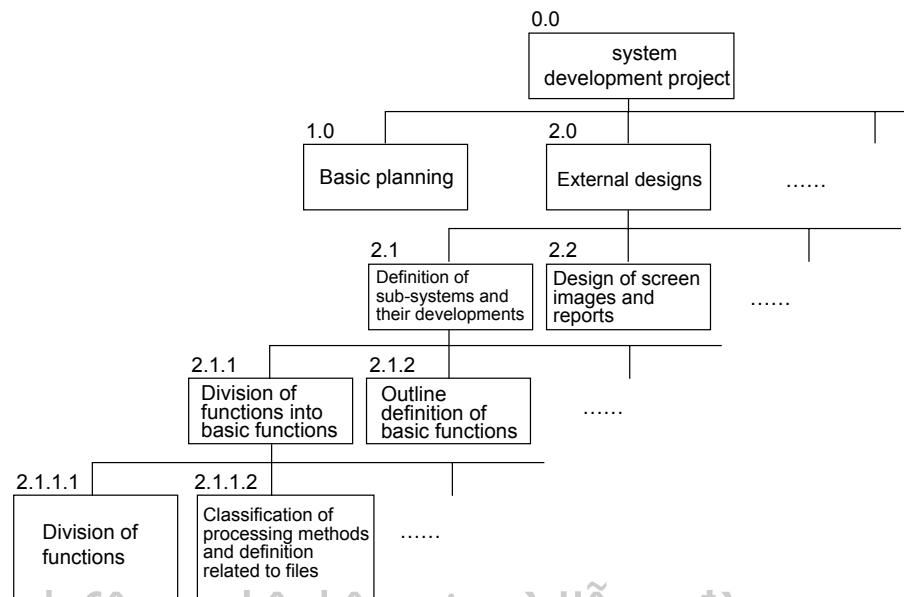
Work Breakdown Structure

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

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

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

A work breakdown structure



Use of WBS provides the following merits.

- Cost estimates and data for the cost analysis are provided.
- Work structure and work coverage of a project and responsibility for the work are clarified.
- Grasping actual progress for each work unit, and the planning of work are made easier.
- Names of broken-down work units and the classification system are a piece of expertise know-how.

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

<http://www.vitec.org.vn>

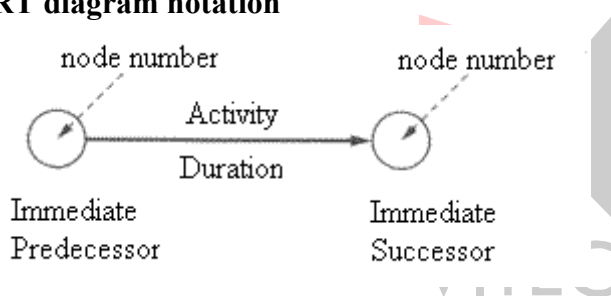
1.3.2 PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method)

The scheduling creation and operational management of a large scale construction plan or a system development plan can be performed by using the PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method). PERT is a method used to make and manage schedules so that the total project duration becomes shortest, and CPM is a method which shortens the whole schedule while minimizing the cost increase. The Gantt chart (bar chart) can be also used but this does not show the order in which activities are to be done.

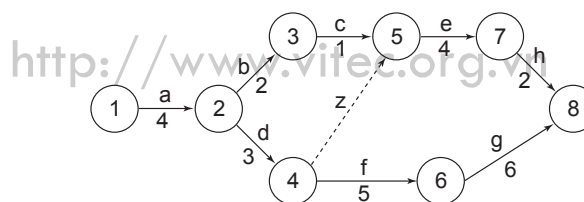
PERT (Program Evaluation and Review Technique)

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

PERT diagram notation



A PERT diagram example



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

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

<Characteristics>

- PERT can handle the development of large-scale and complex systems as well.
- It enables the total number of days required (the minimum period necessary) to be calculated.
- The order in which work is to be done is made clear, enabling important management points to be clarified.

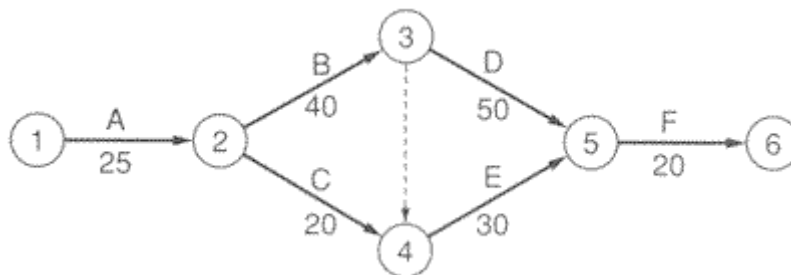
- The number of days included in each work section as a margin is easily calculated.
- It can be applied to calculations to reduce development costs and to decrease the number of days needed for work (based on CPM (Critical Path Method) or other methods).

Example

List of Activities

Activity	Activity Description	Immediate Predecessor	Estimated Duration (days)
A	System Design		25
B	Program Design	A	40
C	Hardware Selection	A	20
D	Programming	B	50
E	Test Design	B,C	30
F	System Test	D,E	20

PERT diagram created from the above activity list



<http://www.vitec.org.vn>

<Procedures>

- (1) The estimated number of days needed for each work section is determined. Then, the results are put together in a table like the above example.
- (2) Based on the work estimate table, the PERT diagram is drawn (take care about the order of works).
- (3) The following numbers of days are decided at each node.
 - Earliest node time: Indicates the earliest possible time, before which work cannot be started.
 - Latest node time: Indicates the latest time by which the work must be completed.
- (4) The path generated by connecting the nodes, for each of which the earliest possible node time and the latest possible node time are the same (indicating that no

margin is provided), is called "critical path." Works on the line are the most important for management.

Critical Path Method

As shown above, a critical path is a path through the network all of whose nodes have no delay allowance, or $[\text{latest node time}] = [\text{earliest node time}]$. Since a critical path is a path with zero slack, a delay in the time of any critical path activity results in a delay in the project completion time. Because of their importance for completing the project, the importance of managing those activities should be emphasized. The method to identify important activities (activities on the critical path) as above is called CPM (Critical Path Method).

In other words, CPM, or critical path method, indicates which tasks are time-critical.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

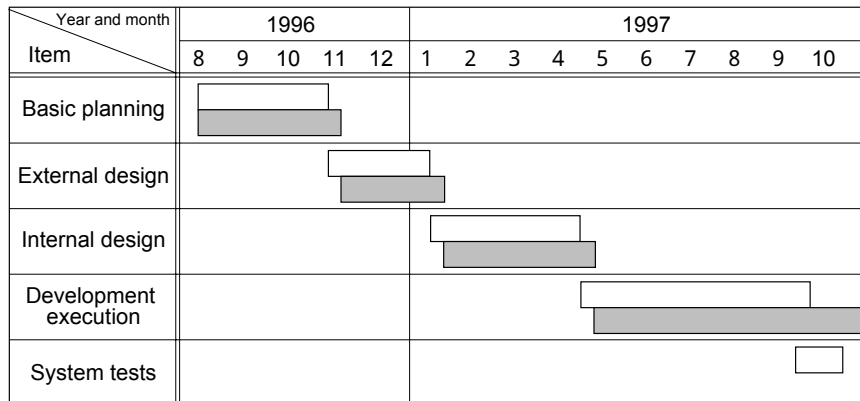


<http://www.vitec.org.vn>

Gantt chart

The Gantt chart is also called "bar chart."

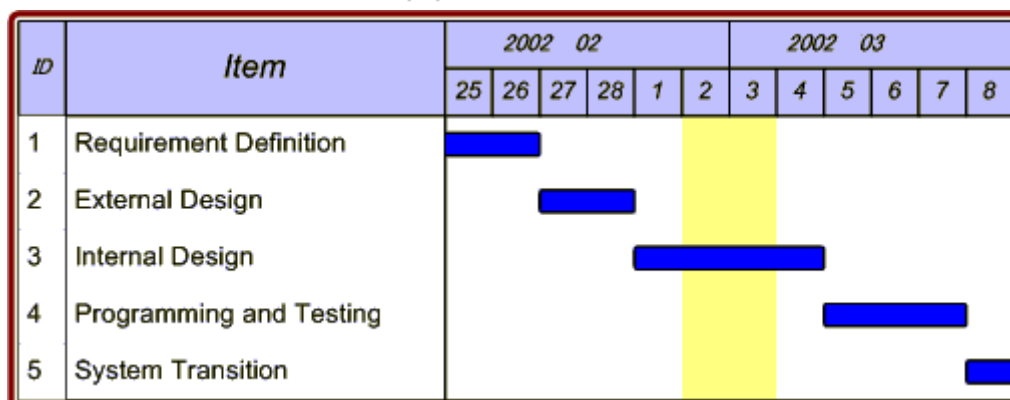
An example Gantt chart is shown below.



<Characteristics>

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

The following is an example Gantt chart created by using a drawing tool, Microsoft Visio 2000.



Task Responsibility Matrix

A matrix that maps WBS tasks to the organizations and individuals assigned as responsible and those supporting those responsible. This clarifies the relationships between planned tasks and persons in charge. In a matrix, tasks are listed vertically (usually the left), involved people or groups are listed horizontally (usually the top), and involvement of a particular individual or group in a particular activity is signified by a tic mark where the row and column intersect.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

1.3.3 Scheduling

The project schedule is a dynamic, detailed record of tasks or activities that must be accomplished to satisfy the project objectives. As described in the preceding section on work breakdown structures, these activities are generally the smallest elements at the bottom of the WBS. The project schedule is used in executing and controlling these activities and is used as a basis for decision making throughout the lifecycle of the project. Scheduling involves estimating the work content of an activity, assigning resources to the activities, and determining when each activity should be performed in order to meet the overall project deadline. Some of the major factors that affect the project schedule include the project objectives, requirements of other projects, unique resource requirements and constraints, and the individual capabilities of team members. According to Cori, there are seven specific steps necessary to the development of any project schedule. These steps must always be undertaken in the proper sequence, while the amount of time devoted to them may vary with each project. The steps include:

Define project objectives

Develop the WBS

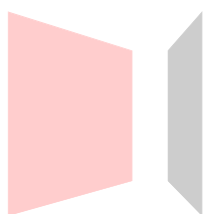
Sequence the project activities

Estimate activity duration and costs

Address schedule and time constraints

Address schedule and resource constraints

Review the schedule



<http://www.vitec.org.vn>

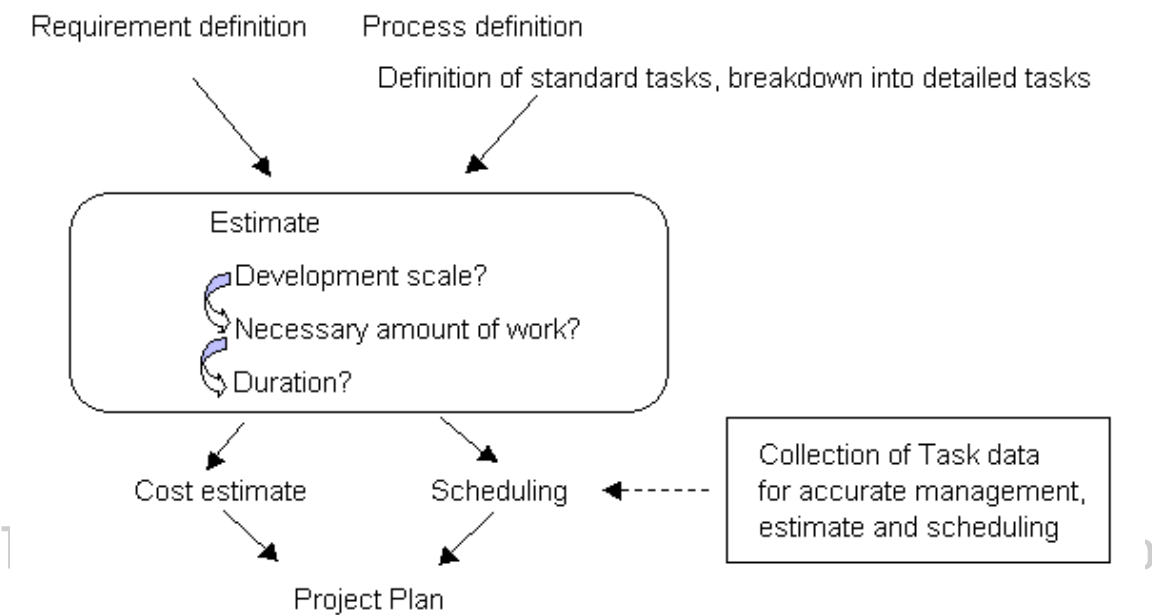
1.3.4 Creation of project plans

Creation of a project plan requires clear understanding of the user's requirements. From there and also from a set of pre-defined standard tasks, the following estimates are made, cost estimates are derived, project schedules are determined so that they are to be included into the project plan after approved.

As for schedules, a master schedule that gives the whole project schedule at a glance as well as detailed schedules that make it possible to manage the progress in detail should be created.

To estimate and manage tasks and schedules with more accuracy, it is necessary to collect actual task data (records of which task took how long). Empirical cost estimate and scheduling became possible based on the collected (earned) values.

Figure Creating estimate and schedules for the project plan



PMI addresses several of these steps in describing five specific processes of time management defined in the PMBOK. According to PMI, each project follows the scheduling processes: definition, sequencing, duration estimating, development, and control. Each process has specific input information, various tools and techniques for schedule manipulation, and specific outputs that ultimately will define the project schedule.

VITEC

<http://www.vitec.org.vn>

1.3.5 Metrics

Project management requires the ability to effectively manage and organize people and to accurately set project schedules. In addition, there must be some type of feedback mechanism (some type of metric) to allow the project manager to gauge progress.

Metrics are extremely important to software managers, because they let them gauge progress and identify problem areas in both the process and the product. Therefore, two types of metrics are needed: process metrics and project metrics.

Process metrics give insight into the quality of the development process. Measurements of such things as defect rates (by modules and total development) and rework time are important. What is not important is tracking individual defect rates—simply because the rate is meaningless. A high individual defect rate could be because a person is working on a harder or more complex task. What is important is to use the metrics to track the causes of the errors, and then work on improving the process.

Project metrics are used more for planning purposes. If the estimated size of a project is known, using data from prior projects, project completion time can be estimated. Once a project is under development, there exist metrics that can determine quality and completeness of design and code.

<http://www.vitec.org.vn>

1.4 Managing Requirements

This refers to the process of defining the user/customer requirements and building the system requirements before going on to develop the specifications in detail.

Requirements should be comprehensive and clear, well structured, traceable and testable. They should give rise to clearly specified project deliverables and should be tested against the original set of requirements. Any changes to the initial requirements should be traceable i.e. documented and explainable.

In a project, what are to be produced, until when, in what format etc. ... are determined based on the user's "requirements." Therefore, requirements should be captured in written format as clearly as possible such that later they can be verified. (or tested or measured.) Functional requirements as well as other requirement should be captured. (e.g. response time within 5 sec.) For better understanding of a requirement, the requirement itself as well as the background of the requirement should be recorded.

The goal of requirements management is to minimize difference in requirements between the system's users and the developers so that the final product will satisfy the system's users.

Requirements are subject to change from time to time, or new requirements may be found throughout the software life cycle. Therefore, requirements should be tracked throughout the life cycle or development process. Numbering each requirement may help identifying and tracking of a requirement with ease. Changes on requirements should be placed under strict control and reviewed for validation.

1.5 Risk Analysis and Management

Many software projects fail because they were not adequately planned. Risk management is frequently thought of as a way to manage risks. This is only partially true. Risk management (and analysis of the risks) is simply spending the time and money to become aware of things that can impact the development effort. Good planning is inadequate if you do not know all the facts-and risk management is nothing but fact gathering.

Many political issues affect risk management. Some organizations merely wish to "keep their heads in the sand." Other organizations do not have the budget to perform proper risk planning. And in many organizations, people are reluctant to come forward and identify risks, for fear of being associated with the risk (or for fear that having identified the risk, it becomes their responsibility). This is bad for business, because opportunities come from risks. If your organization is taking no risks when developing products, you are not developing innovative or even mainstream products-you are using technology and ideas that are out-of-date as soon as your software is released.

At least, possible risks for the organization should be identified then they should be prioritized. The following shows several possible risks noted by Boehm.

- Personnel shortfalls
- Unrealistic schedules
- Not understanding the requirements
- Building a poor user interface
- Trying to gold-plate when the customer does not want to
- Not controlling changing requirements
- Shortfalls of reusable or interfaced components
- Shortfalls in internally performed tasks
- Shortfalls in externally performed tasks
- Poor response time
- Attempting to exceed the capability of current computer technology

1.6 Software Configuration Management (SCM)

Software Configuration Management (SCM) is the discipline for managing the evolution of computer program products during all stages of development and maintenance. The software changes are identified, controlled, and managed throughout the project lifecycle.

The Software Engineering Institute Capability Maturity Model (SEI SW-CMM) defines SCM as stated below.

“ [SCM] involves identifying the configuration of the software (i.e. selected software work products and their descriptions) at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software lifecycle. The work products placed under software configuration management include the software products that are delivered to the customer (e.g., the software requirements document and the code) and the items that are identified with or required to create these software products (e.g., compiler). ”

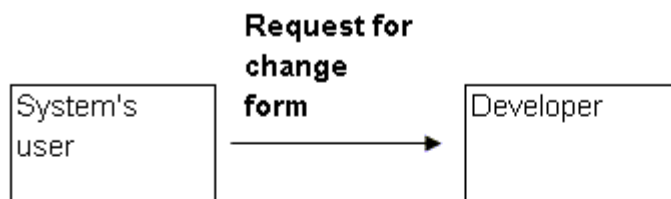
SCM provides the means to manage software processes in a structured, orderly, and productive manner. SCM spans all areas of the software lifecycle and impacts all data and processes. It provides visibility into the status of the evolving software product. Software developers, testers, project managers, Quality Assurance (QA) personnel, and the customer benefit from SCM information. SCM answers who, what, when, and why.

- Who makes the changes?
- What changes were made to the software?
- When were the changes made?
- Why were the changes made?

Common SCM activities include the following

- Accessing and retrieving software
- Retrofitting changes across the development lifecycle
- Migrating changes across the development lifecycle
- Managing the compile and build process
- Managing the distribution of changes
- Obtaining approvals and signoffs
- Managing software change requests
- Coordinating communication between groups
- Obtaining project status
- Tracking bugs and fixes

The following shows steps for managing changes.



- 1) accept change request form and add to the change request lists
- 2) validate the change
- 3) check the affect of the change to avoid unexpected bugs caused by the change
- 4) decide whether to change or not
- 5) make actual changes

1.7 Software Quality Assurance (SQA)

Software Quality Assurance (SQA) is a group of related activities employed "throughout the software life cycle" to positively influence and quantify the quality of the delivered software.

In other words, software quality is not what to be added to software during specific activity (e.g. testing) in the life cycle but what to be woven to software throughout the life cycle.

Activity	Purpose
Testing	This helps ensure quality.
Defined software development process and the development organization's capability to follow the process	This also ensures quality
Frequent and meaningful reviews at "all" stages of the process (reviews starting from the requirements analysis phase)	This helps us detect errors earlier

Object Oriented Testing

"Object Oriented Unit Testing" includes

- Black box testing (functional testing) : This tests based on external specifications
- White box testing (structural testing) : This tests internal code paths

Note: Tools for automating unit testing are available

e.g. vbunit for VB, junit for Java

"Object Oriented Incremental testing" tests interactions between objects of different classes (e.g. video class and copy class, video class and db class)

Note: A dummy class as a stub or as a driver may be needed for incremental testing.

"Object Oriented System Testing" tests the overall application.

- Usecases and sequence diagrams can be traced through object's interactions
- Transitions of states in statecharts are also tested

1.8 Project Management Software

The field of project management has produced a number of tools and support services to aid and assist software organizations. These support options include specific tools to automate project management functions, consulting and expertise services, academic and professional organizations that promote common interests, and conference and other presentation and discussion forums for the proposal and debate of ideas and innovations. This section provides a discussion of the various tools and support areas for software project management.

Today, there are wide ranges of commercial software for project management available in the market. Those come with a wide range of prices. They focus on various aspects of project management.

Some tools from Microsoft Corporation are shown below as examples.

Product name	Purpose
Project 2002	To schedule, organize, and analyze tasks, deadlines, and resources e.g. creating a working schedule with information you provide on tasks, resources, and costs.
Visio 2002 Professional Edition	To draw project management charts such as Gantt chart
Visual Source Safe	The version control system for development teams using Visual Studio <ul style="list-style-type: none">• To increase developer productivity• To improve software quality through effective source code control• To support team software management

Part 7

ADDITIONAL EXERCISES

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Exercises for No.3 Part1 (Software Engineering)

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

- A) In view of profitability, businesses have increased the degree of manufacturing internally without outsourcing systems or operations to external companies.
- B) So-called "end-user computing," in which users by themselves build systems and have access to or process information for their own applications, is gaining support.
- C) To cut costs or to shorten software development time, applications have been developed based on orders received rather than by use of package software.
- D) Widely spreading use of networks reduces the range affected by a system failure, making security management easier.

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

- A) An application is broken down into sub-units. Then, each of them is designed and manufactured sequentially one after another.
- B) System development is made in the order of processes, without work being returned to a higher level process.
- C) An operational experimental product is generated, and checking requirement specifications and evaluation are made in an early phase.
- D) Development time is shortened by participation by users, by development by a fewer number of engineers and by effective use of development tools.

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

- A) Work is performed in the order of basic planning, external design, internal design, program design, programming and test. Therefore, with the technique, a perspective of the work as a whole is gained, making the determination of schedules and allocation of resources easier.
- B) An experimental product is produced in an early phase of system development, enabling the removing of ambiguities and differences in recognition between the user and development organizations.
- C) The software is classified into software whose specifications are fixed and do not require modification, and software whose specifications require modification. Then, for the software whose specifications require modification, the process of development, reconsideration and modification is repeated.
- D) A large-scale application is broken down into sub-units, each of which is highly independent. Then, for each sub-unit, the process of design, programming and test is repeated, gradually expanding the development area.

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

- A) Present problems are investigated and analyzed, then the requirements for a targeted system are defined.
- B) Functions required for building a system are partitioned into programs to make process flow clearer.
- C) Detailed processing procedures are designed, coded, and corrected.
- D) Tests are conducted.
- E) The structured design of each program is made based on internal design documents.
- F) Based on requirements for a system, functions necessary for the system are defined.

- A) a-f-b-c-e-d
- B) a-f-b-e-c-d
- C) a-f-e-b-c-d
- D) a-f-e-c-b-d

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

- A) Design specifications are produced from implemented software. Then, software is developed based on specifications produced thus.
- B) Software is designed in the order of output, processing and input.
- C) Functions having been implemented with software are achieved with hardware.
- D) A development language and development tools are selected depending on the processing characteristics of the software.

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

- A) DFD
- B) ERD
- C) NS chart
- D) State transition diagram
- E) Warnier diagram

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

- A) NS chart
- B) PERT diagram
- C) State transition diagram
- D) Bubble chart

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

- A) Instance
- B) Encapsulation
- C) Clustering
- D) Abstraction

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

- A) Data exchange among objects is executed through instances.
- B) The object indicates descriptions of class characteristics.
- C) Encapsulation indicates putting together classes as a library.
- D) A class can inherit methods from its parent class.

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

- A) In screen transitions, a direct selection method intended for experienced users should be used instead of step-by-step selection by use of menus.
- B) Input items on screens should be enclosed with or [] to make it clear that the spaces are for input fields.
- C) Screen layouts should be designed so that items to be referred to can be arranged from left to right or top to bottom.
- D) To complete a processing operation, designs should be made so that the suspension of data inputting and returning to a previous screen cannot be allowed.
- E) Standardized screen layouts, for example, unified places for displaying titles and messages, should be used.

Q11 Which of the following is the most appropriate as the description of code design and code management?

- A) Codes inevitably vary, so it is important to put code books in order and to manage them.
- B) It is desirable that codes are understandable by themselves. Therefore, the use of longer code is better.
- C) Numerals should mainly be used as codes, and Chinese characters should not be used.
- D) Codes should be assigned to make data classification easier, but the addition or expansion of codes should not be considered.

Q12 Assume that there is a four-numeral code $N_1N_2N_3C$. The right-most C indicates the check digit, which is calculated with the formula below.

$$C = \text{mod} ((N_1 \times 3 + N_2 \times 2 + N_3 \times 1), 10)$$

Here, mod (a, b) indicates the remainder of a/b.

Then, what is the figure for \square in the following four figure code "81□6?"

- A) 0
- B) 2
- C) 4
- D) 6
- E) 8

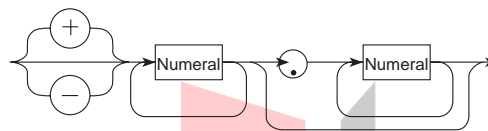
Q13 Which of the following is the most appropriate as the explanation of re-entrant programs?

- A) The program gives correct results even if it is re-executed without being re-loaded after being once executed.
- B) Placed at any addresses on a real memory, the program can be executed.
- C) The program is partitioned into more than one segment, and can be loaded and executed on a segment basis.
- D) Even if more than one task executes the program in parallel, correct results can be obtained.

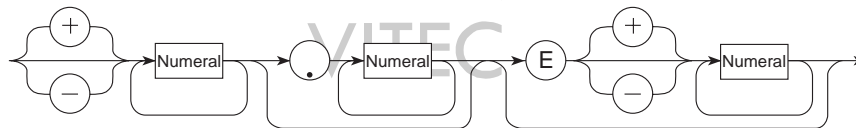
Q14 Which of the following is the most appropriate as the explanation of structured programming, an important factor in the logic design of modules in program generation?

- A) It means that indenting rules for coding are provided to make source lists more easily readable.
- B) It means that notes are effectively used to enable understanding of process methods just by reading them.
- C) It is described with the three basic structures of "sequential," "selection" and "repeating."
- D) A standard size of a module should be 50 to 150 steps.

Q15 There is a syntax which is described with the following syntactic structure. Expressions like -100, 5.3, and +13.07 meets the syntax.



When this description method is used, which of the following numeric expressions meet the syntax specified by the following syntactic diagram?



<http://www.vitec.org.vn>

- A) 5.2E - 07
- B) + 1.E4
- C) - .9
- D) 9.89E

Q16 Which of the following gives the correct answer for the formula described below with reverse Polish notation? Here, $xy-$ indicates that y is subtracted from x , while $xy+$ indicates the quotient from an $xy/$ operation.

Formula: 4 3 5 - +

- A) -2
- B) -0.2
- C) 0.2
- D) 0.5
- E) 5

Q17 Which of the following is correct as the explanation of Java?

- A) It is a communications protocol used on the Internet.
- B) It is a browser for the Internet.
- C) It is an object-oriented programming language.
- D) It is a coding technique for statistic color image data.

Q18 Which of the following is the test case generation method used in the white box test?

- A) The cause-effect graph
- B) The experimental design method
- C) Condition coverage
- D) Equivalence partitioning

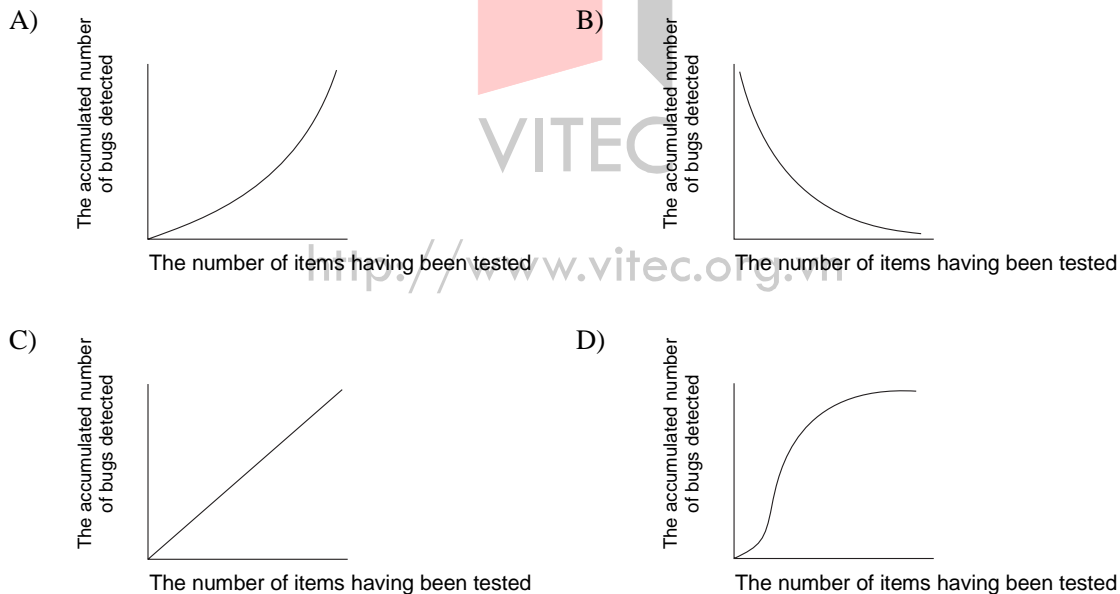
Q19 Which of the following is the most appropriate test to be used in boundary value analysis?

- A) The minimum and maximum values
- B) The minimum and maximum values, and those adding 1 to each of the values
- C) The minimum value and that adding 1 to the value
- D) The maximum value and that adding 1 to the value

Q20 To maintain the quality of design documents, reviews are conducted in each phase of development. Which of the following is the most appropriate as the explanation of the inspection, a review technique?

- A) Reviews as a whole are conducted with each participant playing the responsible role in turn.
- B) For reviewing, part of the target software is experimentally produced and is actually executed.
- C) Items to be reviewed are selected in advance. Then, documents are reviewed quickly by checking an item at a time.
- D) The person who authored the design documents to be reviewed chairs the review meeting.

Q21 The relationship between the number of items having been tested and the accumulated number of bugs is used as a management item to check quality status of the test process. Which of the following graphs indicates that the quality is becoming stabilized?



Q22 Which of the following is classified as a function of upstream CASE tools?

- A) Source program-analyzing function
- B) System analysis and definition function
- C) Test-supporting function
- D) Automatic program generation function
- E) Project management function

Q23 Software quality characteristics include reliability, usability, maintainability and portability. Then, which of the following explains reliability?

- A) It indicates how easily operations can be mastered.
- B) It indicates whether functions required for software can always be maintained normally under designated conditions.
- C) It indicates the degree of modification which becomes necessary when software is to be used in a different computer environment.
- D) It indicates the degree of ease in which modification requests from users or troubles can be handled.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Exercises for No.3 Part2 (External Design)

Q1 Which is the most appropriate work to be done during External design, as part of system development activity?

- a. Physical data design
- b. Structured design of programs
- c. Requirement definition
- d. Logical data design

Q2 In the activities shown below, which is inappropriate as the activities in external design stages?

- a. Screen layouts and screen transitions are to be designed.
- b. Code design is to be done.
- c. Structures and functionalities of programs are to be determined.
- d. Selection of report output media, design of report layout are to be done.
- e. User's business flow is to be organized and confirmed so as to clarify user's requirements.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Exercises for No.3 Part3 (Internal Design)

Q1 Which is the most appropriate work to be done during internal design, as part of system development activity?

- a. Code design
- b. Physical data design
- c. Structured design of programs
- d. Definition of requirements
- e. Logical data design

Q2 Choose two tasks to be performed during physical data design in the internal design stage.

- a. Estimating access time and capacity
- b. Identifying data items
- c. Analyzing data relationships
- d. Creating file specifications
- e. Determining the layout of records

Q3 Which is the technique for representing functions and the flow of data by means of symbols showing data flow, processing (functions), data store and external sources (source of generated data and destination)? (This technique is one of structured analysis methods.)

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

Q4 Which is the appropriate description of the HIPO, one of the structured design methods?

- a. A diagram of contents and a data flow diagram are used.
- b. Control information to be passed between processing blocks is described along with an arrow in a diagram of contents.
- c. A diagram of contents shows the overall function of a program, and numbers entered in processing blocks show the order of processing.
- d. Symbols in a flowchart are used to show what is chosen and what is repeated.
- e. Relationships between input/output and process steps can be represented clearly.

Q5 In the consideration shown below about the screen design in external and internal design stages, which is an inappropriate consideration?

- a. Screen transition should be designed with consideration of not only step-by-step selection using a menu, but also with direct access to a desired screen for users having a high level of proficiency.
- b. Each item to which data is inputted on the screen must be enclosed in a rectangle or square brackets, to emphasize that it is a data input field.
- c. The screen layout must be designed in such a manner that items to be referred to are arranged from left to right, and from top to bottom.
- d. To complete ongoing processing, the screen must be designed to prevent a user from aborting data input, or returning to the previous screen.
- e. The screen layout must be standardized; rules on the positions where titles and messages are shown must be established.

Exercises for No.3 Part 4 (Program Design)

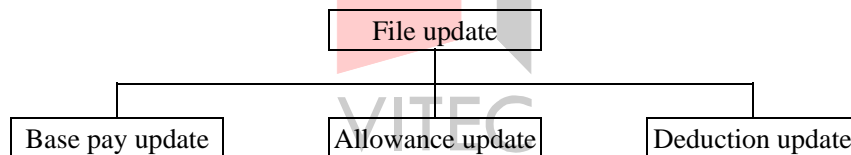
Q1 Which is inappropriate as a remark made concerning the work of module partitioning in the program design stage?

- The number of subordinate modules that one module can invoke must be limited.
- One module must be designed so that it contains a proper number of steps.
- In designing a hierarchical structure in which one module invokes the other, care should be taken to keep the depth within specified limits.
- Interfaces between modules must be simplified.
- Proper comments should be included to make the logic inside a module easy to understand.

Q2 When a program for reading data, choosing numeric data alone and showing the average value is subjected to STS partitioning, functions are sorted into sink, source and transform categories. Choose a right combination from combinations shown below.

	Functions			
	Inputting data	Choosing numbers	Calculating the average value	Showing the result
a	Sink	Sink	Source	Transform
b	Sink	Source	Source	Transform
c	Source	Source	Transform	Sink
d	Source	Transform	Transform	Sink
e	Transform	Sink	Sink	Source

Q3 There is a program for accepting base pay update, allowance update and deduction update slips and updating a payroll calculation file. This program was partitioned into modules, as shown below. Which module partitioning method was used?



- STS partitioning method
- Jackson method
- Transaction partitioning method
- Warnier method

Q4 Which is the method that you should use to convert a data flow diagram created by structured analysis into a structure chart to be used for structured design?

- KJ method
- OMT method
- Jackson method
- Transaction partitioning method

Q5 Which one is a data structure-oriented module partitioning technique?

- Common function partitioning method
- Source/transform/sink partitioning method (STS partitioning method)
- Jackson method
- Transaction partitioning method (TR partitioning method)

Q6 Which of the remarks shown below best describes the Warnier method used to create the structured design of a program?

- a. Structure diagrams of input and output data are drawn with the main attention given to the structure of data. A program structure diagram is then prepared based on an input/output data structure diagram.
- b. Functions in the flow of data are grouped into source, transform and sink categories with the main attention given to the flow of data to deal with.
- c. Software is considered a collection of data and processes. The module independence is increased by encapsulating these data and processes.
- d. With the main attention given to the control structure of a program, program logic is designed based on the control flow that shows invocation relationships.

Q7 Module coupling is a measure of module independence. The weaker the module coupling, the higher the level of module independence becomes. Which of module coupling types shown below has the strongest coupling?

- a. Common coupling
- b. Stamp coupling
- c. Data coupling
- d. Content coupling

Q8 When the following programs are executed, which of the results shown below do you get? x (formal argument) is a call by value and y is a call by reference.

Main program

a=3 ;
b=2 ;
sub (a, b) ;

Subprogram sub (x, y)

x=x+y ;
y=x+y ;
return ;

a. a=3, b=2

b. a=3, b=7

c. a=5, b=2

d. a=5, b=7

VITEC

<http://www.vitec.org.vn>

Exercises for No.3 Part 5 (System Operation and Maintenance)

Q1 Which of the following is not included in resources to be managed in operation management?

- a) Computers
- b) Databases
- c) Programmers
- d) Programs

Q2 Which of the following is correct as the standard procedure to be performed in a system failure?

- a. Temporary measures are taken.
- b. The use of temporary measures is terminated and they are removed.
- c. Failure portions are identified, and failures are localized.
- d. Failure portions are separated.

- a) The occurrence of a failure→finding the failure→a→c→d→permanent measures are taken→b
- b) The occurrence of a failure→finding the failure→b→c→a→permanent measures are taken→d
- c) The occurrence of a failure→finding the failure→c→a→b→permanent measures are taken→d
- d) The occurrence of a failure→finding the failure→c→d→a→permanent measures are taken→b

Q3 Which of the following is the most appropriate for user ID management?

- a) All ID users participating in the same project use the same ID.
- b) A user having more than one ID sets the same password for all the IDs.
- c) If authority is given to a user ID, it should be limited to as minimal as possible.
- d) The deletion of a user ID must be made long after the abolition of the ID has been notified.

Q4 Which of the following is the most inappropriate for the handling of passwords and password files in the system management organization?

- a) Whether passwords can be guessed easily or not is regularly checked, and the use of different passwords is urged for problematic ones.
- b) In order to reduce the extent that passwords are referred to, it is recommended that users record their passwords on pocket books or elsewhere.
- c) If effective terms can be set for passwords, the functions must be used.
- d) Reference by ordinary users to password files must be prohibited, even if the passwords are encrypted.

Q5 Which of the following is unrelated to service standards for online system operations?

- a) Response times
- b) Operation starting time
- c) Failure recovery time
- d) Debugging time

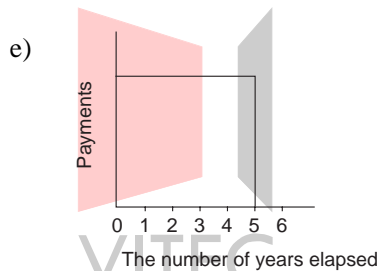
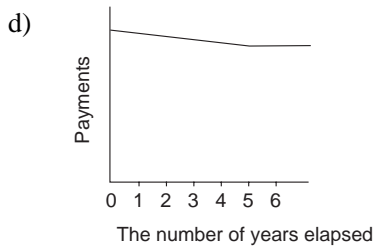
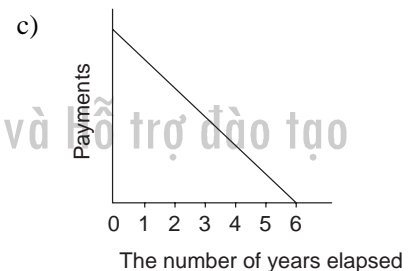
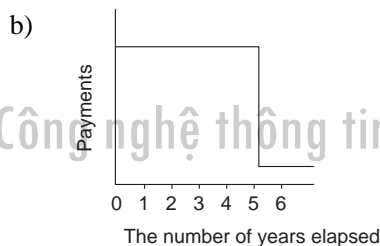
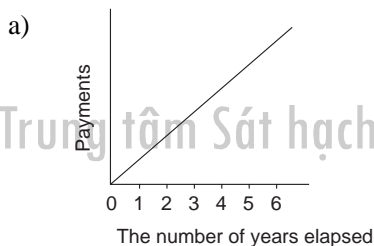
Q6 Which of the following graphs, with the number of years elapsed as the horizontal axis and monthly payments for the vertical axis, is the most appropriate if computers are introduced under the following conditions?

① Computer cost

- For the initial five years, the fixed monthly amount (of lease payment) calculated, based on purchasing prices of the computers and their lease rates, is paid.
- In the sixth and later years, the fixed monthly amount calculated, based on the one-tenth purchasing prices of the computers and their lease rates, is paid.

② Maintenance cost

- The fixed monthly amount is paid as maintenance charges to a maintenance company.



Q7 Which of the following software test methods is used to check whether modifications made for software maintenance have affected other portions or not?

- a) Operation tests
- b) Linkage tests
- c) System tests
- d) Regression tests

Q8 Which of the following is inappropriate for the descriptions of maintenance work for the application software which was developed in house?

- a) The operation manager starts using new software after modifications have been approved, and removes old software based on a pre-determined plan.
- b) Programmers who have developed the original programs perform program modification associated with the modification of specifications made after a development has been completed. Then, new software is quickly put in use in actual environments.
- c) Persons conducting tests must analyze areas affected by a modification, conduct testing of those parts which are related to modified programs, and make evaluations.
- d) In performing maintenance, standards, methodologies and procedures regarding document administration, maintenance methods and program modifying procedures must be provided in advance.

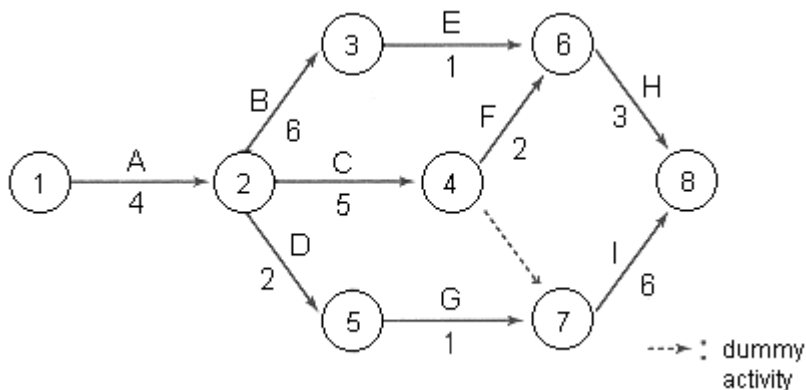
Exercises for No.3 Part 6 (Project Management)

Q1 Suppose that we created an execution plan of a system development project using PERT and calculated a critical path. Which of the followings is the best way of utilizing a critical path?

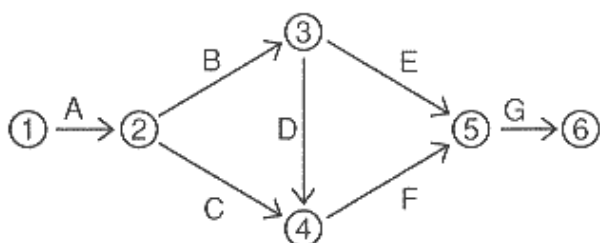
- To identify the activity that requires the most careful attention from the viewpoint of system quality.
- To identify the activity whose execution order is changeable.
- To identify the activity that directly causes the delay of a whole project.
- To identify the most costly activity.

Q2 Concerning the following project, in order to shorten the necessary duration of a critical path by one day, which of the followings is the suitable action to be taken? In the figure, an alphabetic letters above an arrow represents the name of the activity; and the number represents the necessary duration for the activity.

- To shorten the activity B by one day.
- To shorten the activity B and F by one day respectively.
- To shorten the activity H by one day.
- To shorten the activity I by one day.



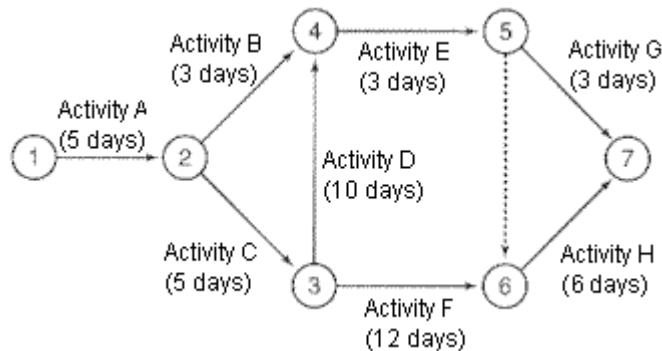
Q3 Which of the following is the latest start time for the activity E in the scheduling diagram shown below?



Activity	Normal time for activity (days)
A	3
B	6
C	5
D	3
E	4
F	5
G	3

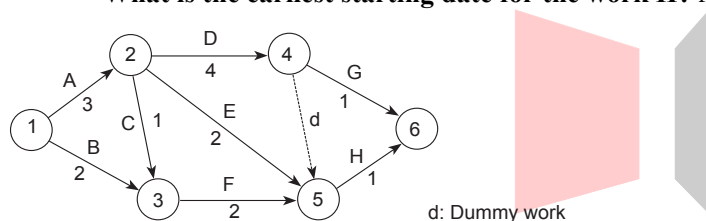
- 7
- 9
- 12
- 13

- Q4** By examining each activity in the following arrow diagram, we found out that only activity D can be shortened by three days. As a result of this reduction in D, how many days can be eliminated from the necessary duration of the entire project? In the following diagram, a dotted-line arrow indicates a dummy activity.



- a. 0 b. 1 c. 2 d. 3

- Q5** There is a project described with the following PERT diagram. The symbol above each arrow indicates a work name, while the number indicates the number of days needed for the work. What is the earliest starting date for the work H? Assume that the project start day is day 0.



- A) 4 B) 5 C) 6 D) 7 E) 8

- Q6** Which of the following is the most appropriate as the description of the characteristics of the chief program team system compared with the hierarchical system?

- A) The amount of burden on the leader is heavier in the hierarchical system than in the chief program team system.
- B) The hierarchical system is less appropriate for large-scale systems than the chief program team system.
- C) Communications within a team is easier in the hierarchical system than in the chief program team system.
- D) The chief program team system does not need a back-up programmer.
- E) The chief program team system needs a librarian.

Answers for No.3 Part1 (Software Engineering)

Answer list

Answers

Q1. b	Q2. b	Q3. b	Q4. b	Q5. a
Q6. a	Q7. a	Q8. b	Q9. d	Q10. d
Q11. a	Q12. a	Q13. d	Q14. c	Q15. a
Q16. a	Q17. c	Q18. c	Q19. b	Q20. c
Q21. d	Q22. b	Q23. b		

Answers and Descriptions

Q1

Answer

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

Description

In this question, the most appropriate characteristic of information technology use in businesses today is to be identified.

- A) In view of profitability, businesses have increased the degree of manufacturing internally without outsourcing systems or operations to external companies.
- C) To cut costs or to shorten software development time, applications have been developed based on orders received rather than by use of package software.
- D) Widely spreading use of networks reduces the range affected by a system failure, making security management easier.

A) Outsourcing is increasing because it is less costly than in-house development. →

Incorrect

B) EUC (End user computing) → Correct, answer

C) For this purpose (cost-cutting and shortening development time) use of package software is better. → Incorrect

D) In network systems, the range of affect by system failure becomes wider, and threats from network are increasing. → Incorrect

Q2

Answer

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

Description

In this question, the most appropriate explanation of the waterfall model is to be identified.

In the waterfall model, each phase cascades into the next phase. One phase had to be completed before the next phase was begun. It does not provide for feedback between phases or for updating/re-definition of earlier phases. → The answer is B.

- A) An application is broken down into sub-units. Then, each of them is designed and manufactured sequentially one after another.
- C) An operational experimental product is generated, and checking requirement specifications and evaluation are made in an early phase.
- D) Development time is shortened by participation by users, by development by a fewer number of engineers and by effective use of development tools.

Q3

Answer

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

Description

In this question, the most appropriate description of prototyping is to be identified.

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.

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

This describes the waterfall model.

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

This describes prototyping correctly. → Answer

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

This describes the spiral model where a series of processes composed of design, programming and test are repeated for each sub-unit of a system, with the development made iteratively and multiplicatively

Q4

Answer

- B) a-f-b-e-c-d

Description

In this question, the option that gives the correct order of the development processes is to be identified.

- A) Present problems are investigated and analyzed, then the requirements for a targeted system are defined.
B) Functions required for building a system are partitioned into programs to make process flow clearer.
C) Detailed processing procedures are designed, coded, and corrected.
D) Tests are conducted.
E) The structured design of each program is made based on internal design documents.
F) Based on requirements for a system, functions necessary for the system are defined.

- A) a-f-b-c-e-d B) a-f-b-e-c-d
C) a-f-e-b-c-d D) a-f-e-c-b-d

A) Requirement Analysis, B) Internal Design, C) Implementation, D) Test, E) Program Design F) External Design

Therefore, the order of those is A), F), B), E), C) and D). → The answer is b)

Q5

Answer

- A) Design specifications are produced from implemented software. Then, software is developed based on specifications produced thus.

Description

In this question, the most appropriate explanation of reverse software engineering is to be identified.

“Reverse engineering” means the process of deriving specifications of existing software.

Existing software is analyzed so that the system's components and their interrelationships are identified and representations of the system created → the answer is A.

- B) Software is designed in the order of output, processing and input.
- C) Functions having been implemented with software are achieved with hardware.
- D) A development language and development tools are selected depending on the processing characteristics of the software.

Q6

Answer

- A) DFD

Description

In this question, the diagram used in the structured analysis where data and functional flows are expressed with symbols respectively indicating data flows, processing (functions), data storage, and externals (data sources and data destinations) is to be identified.

- | | | |
|-----------------------------|--------------------|-------------|
| A) DFD | B) ERD | C) NS chart |
| D) State transition diagram | E) Warnier diagram | |

Among the options, DFD (Data Flow Diagram) describes a system focusing on its data flow. An arrow represents a data flow, a box is a process, and two lines represent a data store. A rectangle represents data origin or destination.

Q7

Answer

- A) NS chart

Description

In this question, the diagram used in structured programming and expresses the entire structure of a program in the form of a hierarchical structure is to be found.

- A) NS chart B) PERT diagram C) State transition diagram D) Bubble chart

NS chart is developed by Nassi and Schneiderman.

In this diagram, the whole structure of a program is represented in hierarchies. → The answer is A,

Unlike the flow chart method, NS charts show logical flow without using connectors.

This diagram is also called a "structured diagram"

Q8

Answer

- B) Encapsulation

Description

In this question, the object-oriented programming terminology that makes details of object implementations invisible by putting together data and methods is to be identified.

- A) Instance B) Encapsulation
C) Clustering D) Abstraction

In object-oriented programming, objects encapsulate data and methods. The user of an object views the object as a black box that provides services. An object's implementation details are not visible to the user. This is called encapsulation. → The answer is B.

Q9

Answer

- D) A class can inherit methods from its parent class.

Description

In this question, the most appropriate description of object-oriented programming is to be identified.

- A) Data exchange among objects is executed through instances.
B) The object indicates descriptions of class characteristics.
C) Encapsulation indicates putting together classes as a library.
D) A class can inherit methods from its parent class.

B) An object is an instance of a class.

C) Encapsulation means putting object's data and methods together.

D) Correct. → The answer is D)

Q10

Answer

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

Description

In this question, inappropriate item(s) to be considered in screen designs of external or internal design is to be identified.

- A) In screen transitions, a direct selection method intended for experienced users should be used instead of step-by-step selection by use of menus.
- B) Input items on screens should be enclosed with or [] to make it clear that the spaces are for input fields.
- C) Screen layouts should be designed so that items to be referred to can be arranged from left to right or top to bottom.
- D) To complete a processing operation, designs should be made so that the suspension of data inputting and returning to a previous screen cannot be allowed.
- E) Standardized screen layouts, for example, unified places for displaying titles and messages, should be used.

In screen design, human interface should be given considerations.

(e.g. Transitions of screens, layout of input and output items in each screen)

C) Not allowing suspension of input and returning to a previous screen makes the user's operation restricted and less flexible → Inappropriate → Answer

Consistency in screen design should be also paid attention.

Q11

Answer

- C) Numerals should mainly be used as codes, and Chinese characters should not be used.

Description

In this question, the most appropriate description of code design and code management is to be identified.

- A) Codes inevitably vary, so it is important to put code books in order and to manage them.
- B) It is desirable that codes are understandable by themselves. Therefore, the use of longer code is better.
- C) Numerals should mainly be used as codes, and Chinese characters should not be used.
- D) Codes should be assigned to make data classification easier, but the addition or expansion of codes

should not be considered.

Code design, the designing of the code, such as the determination of a coding system, is conducted as one of the external design activities.

In a code system, the use of Chinese characters should be avoided. → The answer is C.

D) A code system should make data classification easier. (Up to here is correct). But the addition or expansion of codes should not be considered. (Incorrect. This consideration should be given.)

Q12

Answer

A) 0

Description

In this question, the figure for \square in the following four figure code "81 \square 6" is to be obtained.

The check digit is

$$C = \text{mod}((N1 \times 3 + N2 \times 2 + N3 \times 1), 10)$$

Here, $\text{mod}(a, b)$ means the remaining of the division a / b .

81 \square 6

Assume p is the value in the above blank.

Then the following equation is correct.

$$\text{mod}((8 \times 3 + 1 \times 2 + p \times 1), 10) = 6$$

$$\text{mod}((26 + p \times 1), 10) = 6$$

if $p=0$ then $\text{mod}(26, 10)=6$

if $p=2$ then $\text{mod}(28, 10)=8$

if $p=4$ then $\text{mod}(30, 10)=0$

if $p=6$ then $\text{mod}(32, 10)=2$

if $p=8$ then $\text{mod}(34, 10)=4$

Therefore, the answer is a) 0 out of the following five options.

A) 0

B) 2

C) 4

D) 6

E) 8

Q13

Answer

- D) Even if more than one task executes the program in parallel, correct results can be obtained.

Description

In this question, the most appropriate explanation of re-entrant programs is to be found.

- A) The program gives correct results even if it is re-executed without being re-loaded after being once executed.
B) Placed at any addresses on a real memory, the program can be executed.
C) The program is partitioned into more than one segment, and can be loaded and executed on a segment basis.
D) Even if more than one task executes the program in parallel, correct results can be obtained.

a describes "reusable" programs.

b describes "relocatable" programs.

c describes "overlay" programs.

d describes "re-entrant" programs. → Answer

Q14

Answer

- C) It is described with the three basic structures of "sequential," "selection" and "repeating."

Description

In this question, the most appropriate explanation of structured programming is to be identified.

- C) It is described with the three basic structures of "sequential," "selection" and "repetition."

c describes structured programming correctly. → Answer

- A) It means that indenting rules for coding are provided to make source lists more easily readable.
B) It means that notes are effectively used to enable understanding of process methods just by reading them.
D) A standard size of a module should be 50 to 150 steps.

a, b, d are coding conventions to ensure program's readability, but not descriptions of structured programming.

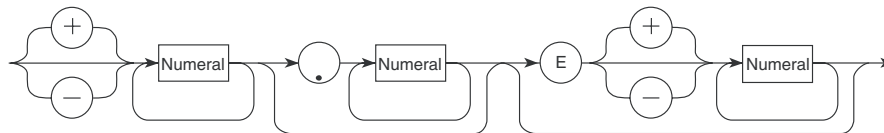
Q15

Answer

A) $5.2E - 07$

Description

In this question, the numeric expression that meets the syntax specified by the following syntactic diagram is to be found.



A) $5.2E - 07$

B) $+ 1.E4$

C) $- .9$

D) $9.89E$

<p>This part represents that a plus or minus sign can be in the leftmost position optionally</p>	<p>Then, one or more numbers, optionally followed by a decimal point and fractional part (also one or more numbers)</p>	<p>Lastly and optionally, the “E” and an optional plus or minus sign and one or more numbers</p>
	<p>5.2</p>	<p>E-07</p>

A) $5.2E-07$ conforms to the above syntax. Therefore, the answer is A.

Q16

Answer

A) -2

Description

In this question, the correct answer for the formula described below with reverse Polish

notation "4 3 5 - ÷"

- A) -2 B) -0.2 C) 0.2 D) 0.5 E) 5

The reverse Polish notation "4 3 5 - ÷" means

$$4 \div (3-5) = 4 \div (-2) = -2$$

Therefore, the answer is A.

Q17

Answer

- C) It is an object-oriented programming language.

Description

In this question, the correct explanation of Java is to be found.

- A) It is a communications protocol used on the Internet.
B) It is a browser for the Internet.
D) It is a coding technique for statistic color image data.

Java is an object oriented programming language developed by Sun Microsystems.

It has wide range of applications, from small applications (called applets) running on a web browser, stand-alone applications to server-side applications. Java applications can easily be delivered over the Internet, or any network, without operating system or hardware platform compatibility issues.

Q18

Answer

- C) Condition coverage

Description

In this question, the test case design method used in the white box test is to be found.

- A) The cause-effect graph B) The experimental design method
C) Condition coverage D) Equivalence partitioning

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.

Therefore, the answer is c (Condition coverage).

a,b,d are test case design methods for black box test.

Q19

Answer

- B) The minimum and maximum values, and those adding 1 to each of the values

Description

In this question, the most appropriate test to be used in boundary value analysis is to be identified.

- A) The minimum and maximum values
- B) The minimum and maximum values, and those adding 1 to each of the values
- C) The minimum value and that adding 1 to the value
- D) The maximum value and that adding 1 to the value

For the black box test, test techniques such as equivalence partitioning, boundary value analysis, cause-effect graphs, and error estimations are used.

In boundary value analysis, test data values are chosen to lie along data extremes. Boundary values include maximum, minimum, just inside/outside boundaries, typical values, and error values. The hope is that, if a systems works correctly for these special values then it will work correctly for all values in between. Therefore, the answer is B.

Q20

Answer

- C) Items to be reviewed are selected in advance. Then, documents are reviewed quickly by checking an item at a time.

Description

In this question, the most appropriate description of inspection is to be identified.

- A) Reviews as a whole are conducted with each participant playing the responsible role in turn.
- B) For reviewing, part of the target software is experimentally produced and is actually executed.
- C) Items to be reviewed are selected in advance. Then, documents are reviewed quickly by checking an item at a time.
- D) The person who authored the design documents to be reviewed chairs the review meeting.

a In an inspection, a "moderator" is in charge, not "participants in turn."

b describes prototyping. In an inspection, documents are delivered and being reviewed, but

not prototypes.

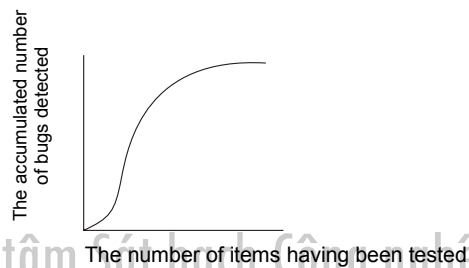
c correct --> answer

d who conducts the inspection is a "moderator".

Q21

Answer

D)



Description

In this question, the graph that indicates the quality is becoming stabilized is to be found.
(software quality observed in graphs)

The answer is d).

This is called "software reliability growth curve"

Q22

Answer

B) System analysis and definition function

Description

In this question, the function that is classified as a function of upstream CASE tools is to be found.

- | | |
|--------------------------------------|--|
| A) Source program-analyzing function | B) System analysis and definition function |
| C) Test-supporting function | D) Automatic program generation function |
| E) Project management function | |

upper CASE functionalities

a this is for reverse engineering and in lower CASE

b correct (answer)

c test support is in lower CASE

d code generation is in lower CASE

e project management is related to the whole process, or both upper and lower phases.

Q23

Answer

- B) It indicates whether functions required for software can always be maintained normally under designated conditions.

Description]

In this question, the explanation of “reliability” is to be identified.

- A) It indicates how easily operations can be mastered.
B) It indicates whether functions required for software can always be maintained normally under designated conditions.
C) It indicates the degree of modification which becomes necessary when software is to be used in a different computer environment.
D) It indicates the degree of ease in which modification requests from users or troubles can be handled.

a describes usability

b describes reliability --> answer

c describes portability

d describes maintainability



<http://www.vitec.org.vn>

Answers for No.3 Part2 (External Design)

Answer list

Answers

Q 1: D Q 2: C

Answers and Descriptions

Q1

Answer

d. Logical data design

Description

In this question, the most appropriate work to be done during External design, as part of system development activity is to be found among the following.

- a. Physical data design b. Structured design of programs
- c. Requirement definition
- d. Logical data design

External design clarifies system's functionalities (or what the system does.) External design is based on the result of requirement definition, and external design result is to be brought over to the next step, internal design.

Internal design activities includes

- Designing subsystem functions and interfaces
- Screen design, Report design
- Logical data design
- External design review

Therefore the answer is d. Logical data design

Q2

Answer

- c. Structures and functionalities of programs are to be determined.

Description

In this question, an inappropriate activity in external design is to be identified.

- a. Screen layouts and screen transitions are to be designed.
- b. Code design is to be done.
- c. Structures and functionalities of programs are to be determined.
- d. Selection of report output media, design of report layout are to be done.
- e. User's business flow is to be organized and confirmed so as to clarify user's requirements.

In the above, c is an inappropriate because it is one of the activities to be done during internal design.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Answers for No.3 Part3 (Internal Design)

Answer list

Answers

Q 1: B

Q 2: A,E

Q 3: A

Q 4: E

Q 5: D

Answers and Descriptions

Q1

Answer

b. Physical data design

Description

In this question, the most appropriate work to be done during internal design, as part of system development activity is to be found among the following.

- a. Code design b. Physical data design
c. Structured design of programs d. Definition of requirements e. Logical data design

Internal design, unlike External design, clarifies how to implement the system from the viewpoint of developers.

System functionalities are to be divided into modules (programs) and interfaces between modules (programs) are clearly defined.

Internal design is based on the result of external design, and internal design result is to be brought over to the next step, program design.

Internal design activities includes

- Functional decomposition
- Structured design
- Physical data design
- Input output design
- Internal design review

Therefore the answer is b. Physical data design

Q2

Answer

- a. Estimating access time and capacity e. Determining the layout of records

Description

In this question, two tasks to be performed during physical data design in the internal design stage are to be identified.

- a. Estimating access time and capacity b. Identifying data items
c. Analyzing data relationships d. Creating file specifications
e. Determining the layout of records

Physical data design includes such activities as file capacity estimate, database storage design etc.

Among the given options, a and e are to be performed in internal design.

Others (i.e. b, c and d) are done in external design.

Q3

Answer

- a DFD

Description

In this question, the diagram used in structured analysis for representing functions and the flow of data by means of symbols showing data flow, processing (functions), data store and external sources (source of generated data and destination) is to be identified.

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

a DFD

DFD stands for "Data Flow Diagram" and this diagram is what is described in the question sentence. → answer

b E-R diagram

This diagram is used in data modeling to show entities and their relationships.

c NS chart

This diagram is used in structured programming.

d state transition diagram

This represents possible states and transitions between states.

e Warnier diagram

This diagram is also used in module design.

Q4

Answer

- e. Relationships between input/output and process steps can be represented clearly.

Description

In this question, the appropriate description of the HIPO, one of the structured design methods is to be identified.

- a. A visual table of contents and a data flow diagram are used.
- b. Control information to be passed between processing blocks is described along with an arrow in a visual table of contents.
- c. A visual table of contents shows the overall function of a program, and numbers entered in processing blocks show the order of processing.
- d. Symbols in a flowchart are used to show what is chosen and what is repeated.
- e. Relationships between input/output and process steps can be represented clearly.

HIPO stands for “Hierarchy plus Input-Process-Output diagram.”

The “Hierarchy” part is a visual table of contents that displays the modules in a hierarchy similar to an organization chart. The “plus Input-Process-Output” part is a diagram that shows all input, all processes and all output.

Therefore a, b, c and d are all incorrect.

The answer is e.

Q5

Answer

- d. To complete ongoing processing, the screen must be designed to prevent a user from aborting data input, or returning to the previous screen.

Description

In this question, an inappropriate consideration concerning screen design in external design and internal design is to be identified.

- a. Screen transition should be designed with consideration of not only step-by-step selection using a menu, but also with direct access to a desired screen for users having a high level of proficiency.
- b. Each item to which data is inputted on the screen must be enclosed in a rectangle or square brackets, to emphasize that it is a data input field.
- c. The screen layout must be designed in such a manner that items to be referred to are arranged from left to right, and from top to bottom.
- d. To complete ongoing processing, the screen must be designed to prevent a user from

aborting data input, or returning to the previous screen.

- e. The screen layout must be standardized; rules on the positions where titles and messages are shown must be established.

In the above, d is an inappropriate because it only pays attention to system's convenience, not the user's convenience or ease of operation. Screen design should be done by giving consideration to human interface element.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Answers for No.3 Part4 (Program Design)

Answer list

Answers

Q 1:	E	Q 2:	C	Q 3:	C	Q 4:	D	Q 5:	C
Q 6:	A	Q 7:	C	Q 8:	B				

Answers and Descriptions

Q1

Answer

- e. Proper comments should be included to make the logic inside a module easy to understand.

Description

In this question, an improper remark on module partitioning is to be identified.

- a. The number of subordinate modules that one module can invoke must be limited.
- b. One module must be designed so that it contains a proper number of steps.
- c. In designing a hierarchical structure in which one module invokes the other, care should be taken to keep the depth within specified limits.
- d. Interfaces between modules must be simplified.
- e. Proper comments should be included to make the logic inside a module easy to understand.

All of a, b, c and d are module design considerations.

e is a good coding practice but not related to module partitioning.

Therefore, the answer is e.

Q2

Answer

Functions				
	Inputting data	Choosing numbers	Calculating the average value	Showing the result
c	Source	Source	Transform	Sink

Description

In this question, the right combination of a program's function classifications based on the STS partitioning. This program reads data, chooses numeric data alone and shows the average value.

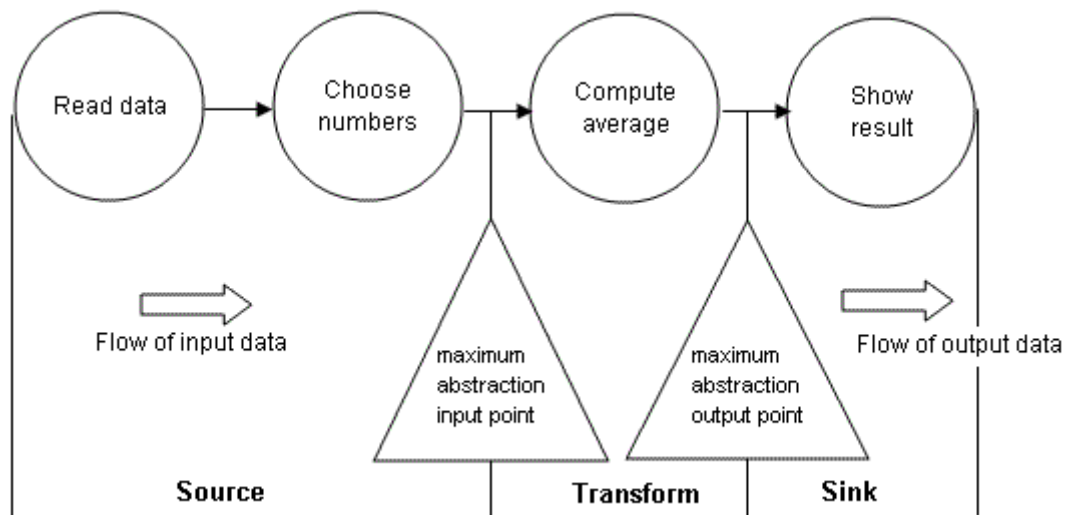
Functions				
	Inputting data	Choosing numbers	Calculating the average value	Showing the result
a	Sink	Sink	Source	Transform
b	Sink	Source	Source	Transform
c	Source	Source	Transform	Sink
d	Source	Transform	Transform	Sink
e	Transform	Sink	Sink	Source

In STS partitioning (or decomposition) method, a program is divided into three parts as shown below

- 1) Source (Input processing function)
- 2) Transform (data processing function)
- 3) Sink (output processing function)

In this method, a maximum abstraction input point, where data can be no longer be considered input data, and a maximum abstraction output point, where data begins to take shape as output data.

The program in question “reads data”, “chooses numeric data alone” and “shows the average value”. In addition “computes data” is also performed. This plus actual computation of the average can be shown as below.



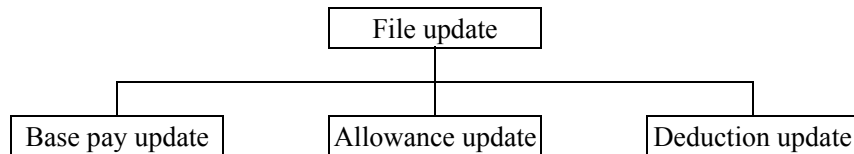
Q3

Answer

c. Transaction partitioning method

Description

In this question, the name of the decomposition method used for dividing the following file update program is to be identified.



a. STS partitioning method

b. Jackson method

c. Transaction partitioning method

d. Warnier method

Decomposition is the process of breaking the function of a module into less complex pieces.

There are three types of “data flow driven” decomposition methods.

- 1) Source/Transform/Sink decomposition
- 2) Transactional decomposition
- 3) Functional decomposition

There are also “data structure driven” decomposition methods. They are

- 1) Jackson method
- 2) Warnier method

In the above case, since each piece after decomposition represents a different transaction, the answer is c. Transactional decomposition.

Q4

Answer

d. Transaction partitioning method

Description

In this question, the method that you should use to convert a data flow diagram created by structured analysis into a structure chart to be used for structured design?

- a. KJ method b. OMT method c. Jackson method d. Transaction partitioning method

The answer is d.

a is a scientific approach to problem-solving, developed by Professor Jiro Kawakita in Japan.

b is one of the object oriented methodologies.

c is data “structure” driven partitioning method.

Q5

Answer

- c. Jackson method

Description

In this question, a module partitioning technique that focuses on the data structure is to be identified.

- a. Common function partitioning method
- b. Source/transform/sink partitioning method (STS partitioning method)
- c. Jackson method
- d. Transaction partitioning method (TR partitioning method)

All of the a, b and d are data “flow” oriented partitioning techniques.

Therefore the answer is c, Jackson method or JSP (Jackson Structured Programming.)

Q6

Answer

- a. Structure diagrams of input and output data are drawn with the main attention given to the structure of data. A program structure diagram is then prepared based on an input/output data structure diagram.

Description

In this question, the best description of the Warnier method used to create the structured design of a program is to be identified.

- a. Structure diagrams of input and output data are drawn with the main attention given to the structure of data. A program structure diagram is then prepared based on an input/output data structure diagram.
- b. Functions in the flow of data are grouped into source, transform and sink categories with the main attention given to the flow of data to deal with.
- c. Software is considered a collection of data and processes. The module independence is increased by encapsulating these data and processes.
- d. With the main attention given to the control structure of a program, program logic is designed based on the control flow that shows invocation relationships.

Since the Warnier method is a data “structure” oriented partitioning method, the answer is a.

Q7

Answer

d. Content coupling

Description

In this question, the strongest module coupling among the given four kinds is to be identified.

- a. Common coupling b. Stamp coupling
c. Data coupling d. Content coupling

Module Coupling

One of the goals for module design is to minimize module coupling with other modules. In other words make it as independent as possible. There are also seven types of module coupling. From least dependence to highest dependence, they are as follows.

No direct coupling	A module does not share data with other module at all.	best or ideal
Data coupling (c)	Two module share information only via "homogenous" data items.	
Stamp coupling (b)	Two modules reference the same non-global data structure.	
Control coupling	One module passes and explicit element of control to the other module	
External coupling	A group of modules reference a limited access global data structure (file global).	
Common coupling (a)	A group of modules reference a global data structure.	
Content coupling (d)	one module directly references the insides of the other	worst

Q8

Answer

b. a=3, b=7

Description

In this question, a pair of subprogram call result is to be identified.

Main program	Subprogram	sub (x, y)
a=3 ;	x=x+y ;	
b=2 ;	y=x+y ;	
sub (a, b) ;	return ;	

x (formal argument) is a call by value and y is a call by reference.

In the main program, 3 is assigned to a, 2 is assigned to b.

Then the subprogram called “sub” is being executed.

Since x is “call by value” and y is “call by reference”, when calling the subprogram, the value of a is passed and the address of b is passed.

Therefore, at the end of the subprogram call, a is NOT changed, b becomes 7.

($x=x+y=3+2=5$, $y=x+y=5+2=7$)

Consequently, a is 3 (no change) and b=7. → The answer is b from the following.

a. a=3, b=2

b. a=3, b=7

c. a=5, b=2

d. a=5, b=7

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Answers for No.3 Part5 (System Operations and Maintenance)

Answer list

Answers

Q1. c

Q2. d

Q3. c

Q4. b

Q5. d

Q6. b

Q7. d

Q8. b

Answers and Descriptions

Q1

Answer

- c) Programmers

Description

In this question, what is not included in resources to be managed in operation management is to be identified among the following.

- | | |
|----------------|--------------|
| a) Computers | b) Databases |
| c) Programmers | d) Programs |

a) computers are resources to be managed in operation and maintenance phase.

b) databases are also the same.

c) programmers are not managed --> answer

d) programs are also managed.

Q2

Answer

- d) The occurrence of a failure→finding the failure→c→d→a→permanent measures are taken→b

Description

In this question, the correct standard procedure to be performed in a system failure is to be identified.

- | |
|---|
| a) The occurrence of a failure→finding the failure→a→c→d→permanent measures are taken→b |
| b) The occurrence of a failure→finding the failure→b→c→a→permanent measures are taken→d |
| c) The occurrence of a failure→finding the failure→c→a→b→permanent measures are taken→d |
| d) The occurrence of a failure→finding the failure→c→d→a→permanent measures are taken→b |

The answer is:

d failure-->discovery

-->identify-->isolate-->temporally measure

-->permanent measure-->remove temporary measure

First of all, failure part must be identified and isolated, and then temporally measures must be taken.

Q3

Answer

- c) If authority is given to a user ID, it should be limited to as minimal as possible.

Description

In this question, the most appropriate thing for user ID management is to be found out of the following.

- a) All ID users participating in the same project use the same ID.
 - b) A user having more than one ID sets the same password for all the IDs.
 - c) If authority is given to a user ID, it should be limited to as minimal as possible.
 - d) The deletion of a user ID must be made long after the abolition of the ID has been notified.
- a) Each user should be given a different user ID.
- b) A user should avoid setting the same passwords for all the IDs he or she is given.
- c) Least possible authority should be given. → correct → Answer
- d) When a user ID becomes not-in-use, the ID should be immediately deleted.

Q4

Answer

- b) In order to reduce the extent that passwords are referred to, it is recommended that users record their passwords on pocket books or elsewhere.

Description

In this question, the most inappropriate thing for the handling of passwords and password files in the system management organization is to be identified.

- a) Whether passwords can be guessed easily or not is regularly checked, and the use of different passwords is urged for problematic ones.
- b) In order to reduce the extent that passwords are referred to, it is recommended that users record their passwords on pocket books or elsewhere.
- c) If effective terms can be set for passwords, the functions must be used.

d) Reference by ordinary users to password files must be prohibited, even if the passwords are encrypted.

a, c and d are appropriate.

b is inappropriate because if passwords are written, the possibility of stealing them become higher.

Q5

Answer

d) Debugging time

Description

In this question, a thing that is unrelated to service standards for online system operations is to be found.

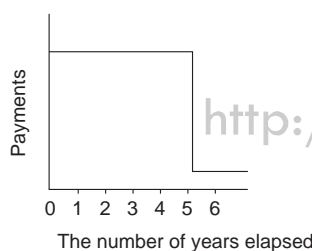
- | | |
|--------------------------|----------------------------|
| a) Response times | b) Operation starting time |
| c) Failure recovery time | d) Debugging time |

a, b and c are related but d is not related (it is a matter during development, not during actual operation)

Q6

Answer

b)



Description

In this question, the most appropriate graph describing the years vs. lease payment if computers are introduced under the following conditions is to be found.

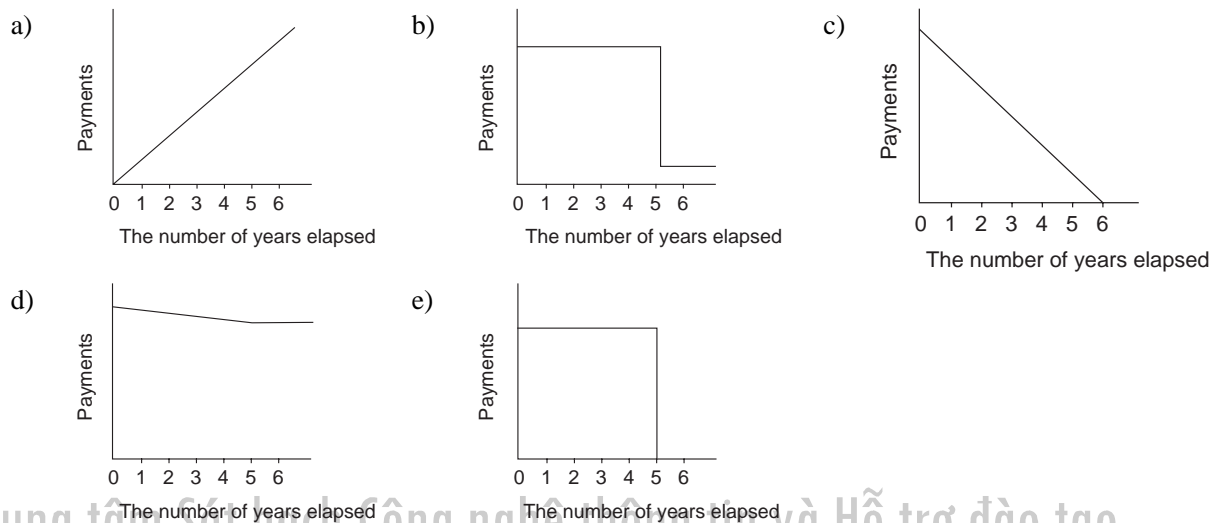
① Computer cost

- For the initial five years, the fixed monthly amount (of lease payment) calculated, based on purchasing prices of the computers and their lease rates, is paid.

- In the sixth and later years, the fixed monthly amount calculated, based on the one-tenth purchasing prices of the computers and their lease rates, is paid.

② Maintenance cost

- The fixed monthly amount is paid as maintenance charges to a maintenance company.



Q7

Answer

- d) Regression tests

Description

In this question, the software test method used to check whether modifications made for software maintenance have affected other portions or not is to be identified.

- | | |
|--------------------|----------------------|
| a) Operation tests | b) Integration tests |
| c) System tests | d) Regression tests |

a operational test

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

b integration test

Integration tests are designed to test integrated software components to determine if they actually run as one program. Operations of programs and interfaces between modules are checked.

c system test

System testing ensures that the entire integrated software system meets requirements.

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

results.

d regression test

Regression tests are a series of tests to ensure that no adverse changes are introduced to the application during maintenance changes, upgrades, or other changes. → Answer

Q8

Answer

- b) Programmers who have developed the original programs perform program modification associated with the modification of specifications made after a development has been completed. Then, new software is quickly put in use in actual environments.

Description

In this question, the inappropriate for the description of maintenance work for the application software which was developed in house is to be identified.

- a) The operation manager starts using new software after modifications have been approved, and removes old software based on a pre-determined plan.
- b) Programmers who have developed the original programs perform program modification associated with the modification of specifications made after a development has been completed. Then, new software is quickly put in use in actual environments.
- c) Persons conducting tests must analyze areas affected by a modification, conduct testing of those parts which are related to modified programs, and make evaluations.
- d) In performing maintenance, standards, methodologies and procedures regarding document administration, maintenance methods and program modifying procedures must be provided in advance.

a correct (modification should be under control. It should be planned and got approval before actually done)

b modifications need not be done by the programmer who coded the program in question)

c correct (regression test should performed)

d correct (operation and maintenance rules should be pre-determined)

Answers for No.3 Part6 (Project Management)

Answer list

Answers

Q 1:	C	Q 2:	D	Q 3:	D	Q 4:	B
Q 5:	D	Q 6:	E				

Answers and Descriptions

Q1

Answer

- c. To identify the activity that directly causes the delay of a whole project.

Description

In this question, the best way of utilizing the critical path is to be found among the following.

- a. To identify the activity that requires the most careful attention from the viewpoint of system quality.
- b. To identify the activity whose execution order is changeable.
- c. To identify the activity that directly causes the delay of a whole project.
- d. To identify the most costly activity.

Since the critical path is the sequence of activities that determines the duration of the whole project, a delay in any of the activities on the critical path makes the whole project delayed.

Therefore the answer is c.

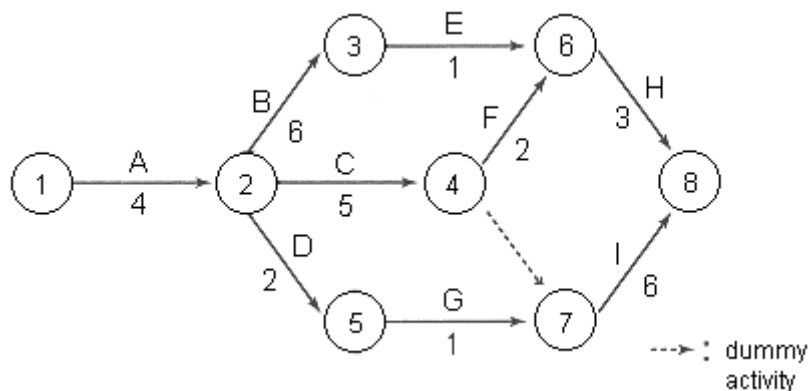
Q2

Answer

- d. To shorten the activity I by one day.

Description

In this question, the suitable action to be taken in order to shorten the necessary duration of a critical path by one day in the given diagram is to be identified.



The earliest and latest node times in the given diagram are as follows

node	1	2	3	4	5	6	7	8
earliest	0	4	10	9	6	11	9	15
latest	0	4	11	9	8	12	9	15

The critical path of this diagram is

(1) → (2) → (4) → (7) → (8)

(or Activity A → Activity C → dummy activity → Activity I)

Therefore, among the given options shown below,

- To shorten the activity B by one day.
- To shorten the activity B and F by one day respectively.
- To shorten the activity H by one day.
- To shorten the activity I by one day.

d is the answer because I is on the critical path.

Q3

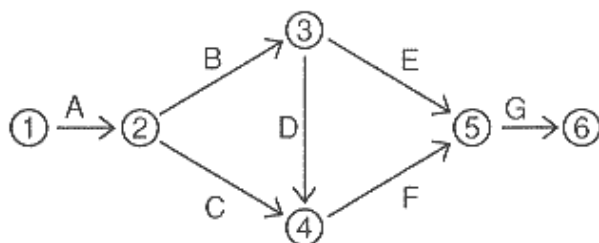
Answer

d. 13

Description

In this question, the latest start time for the activity E in the scheduling diagram shown below is to be identified.

- 7
- 9
- 12
- 13



Activity	Normal time for activity (days)
A	3
B	6
C	5
D	3
E	4
F	5
G	3

In the given diagram, the paths to node (4) are as follows.

A→B→D, 3+6+3=12 days

A→C, 3+5=8 days

Therefore, to node (5), taking the A→B→D→F path makes the node (5) start time 17, taking the A→C→F path makes it 13.

To complete the activity E by the start time of node (5), at the latest, E should be started on the 17-4=13th day → the answer is d.

Q4

Answer

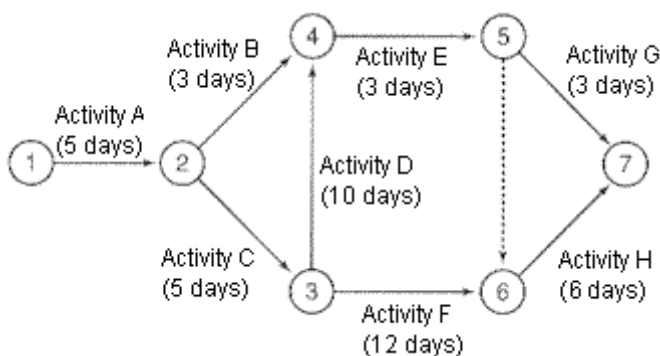
b. 1

Description

<http://www.vitec.org.vn>

In this question, the number of days that can be eliminated in the entire project by reducing three days in the activity D is to be identified.

- a. 0 b. 1 c. 2 d. 3



Original schedule

The original critical path is (1) → (2) → (3) → (5) → (6) → (7)

(or Activity A Activity C Activity D → Activity E Activity H)
 $5+5+10+3+6=29$ (days)

After D is shorted by 3 days

The new critical path will be (1) → (2) → (3) → (6) → (7)
 (or Activity A Activity C Activity F Activity H)
 $5+5+12+6=28$ (days)

Therefore the answer is 1day → b

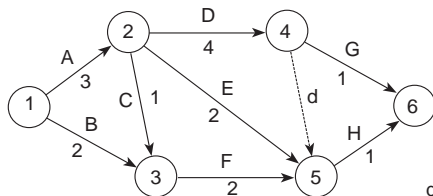
Q5

Answer

D) 7

Description

In this question, the earliest starting date for the activity H is to be calculated in the following PERT diagram.



d: Dummy work

<http://www.vitec.org.vn>

node number	earliest start time at node	predecessor activity	remarks
1	0	-	Project start date=0
2	3	A	A(3days)
3	4	B,C	A(3days)+C(1days)
4	7	D	A(3days)+D(4days)
5	7	D,E,F	A(3days)+D(4days)
6	8	G,H	A(3days)+D(4days)+G(1day)

The earliest start time of the activity H is the earliest start time at node No.5.

Therefore the answer is 7 days. D)

A) 4 B) 5 C) 6 D) 7 E) 8

Q6

Answer

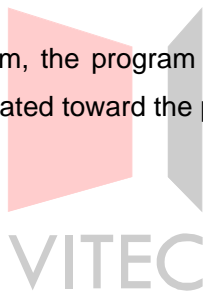
- E) The chief programmer team system needs a librarian.

Description

In this question, the most appropriate description of the characteristics of the chief programmer team system compared with the hierarchical system is to be found.

- A) The amount of burden on the leader is heavier in the hierarchical system than in the chief programmer team system.
- B) The hierarchical system is less appropriate for large-scale systems than the chief programmer team system.
- C) Communications within a team is easier in the hierarchical system than in the chief programmer team system.
- D) The chief programmer team system does not need a back-up programmer.
- E) The chief programmer team system needs a librarian.

In the chief programmer team system, the program librarian is responsible for storing all versions of all programs and data created toward the project → The answer is E,



<http://www.vitec.org.vn>

Index

[A]

abstract data type 1-30
Abstraction 1-46
access rights 5-13
Actual cost (AC) 6-18
Ada 1-144
air conditioning facilities 5-10
ALGOL 1-63, 1-140
AVR 5-9

[B]

backlog 1-2
backup programmer 6-6, 6-9
bar chart 6-25
base allocation 5-15
BASIC 1-63
battery 5-10
black box test 1-100, 1-140, 4-88
BNF 1-32, 1-40
bottom-up test 4-89, 4-92, 4-94
boundary value analysis 1-100

[C]

C language 1-88
C++ 1-8, 1-79, 1-82, 1-88, 1-89
Capability Maturity Model (CMM) 6-15
CASE 1-3, 1-4, 1-9, 1-119, 1-128,
CASE tool 1-128, 1-129, 1-130
cause-effect graph 1-100, 1-101
charging 5-15
chief programmer 6-6, 6-7, 6-9
chief programmer team 6-6, 6-9
class 1-30, 1-31, 1-79, 1-81, 1-82
class library 1-79, 1-82, 3-114
CMM 6-15
COBOL 1-63, 1-75, 1-140, 1-146
code 1-16, 1-17, 1-19, 1-30, 1-81
code review 1-109
coding 1-7, 1-69, 1-108, 1-116
common coupling 4-14
common function
partitioning method 4-2, 4-4
compiler 1-79, 1-81, 1-88, 1-89
conceptual model 2-35
condition coverage 1-104, 1-105
control 6-15
control coupling 4-14
control structure 1-7, 1-47, 1-48
controlling 6-3

cost management 5-14
Cost variance (CV) 6-18
coverage 1-104, 1-105, 1-108
CPM 6-22, 6-23, 6-24
creation of parts 3-108
crime prevention facilities 5-11
critical path 6-24

[D]

daily maintenance 5-26
DASD 3-129, 3-132
data 1-5, 1-6, 1-16, 1-17, 1-19
Data analysis 2-36
data audit 5-5
data coupling 4-15
Data Flow Diagram 1-121
Data Flow Diagram (DFD) 2-29
Data flow refinement 2-28
data resource management 5-5
data specifications 2-57
data trouble 5-8
data type 1-6, 1-32, 1-78
decision table 1-33, 1-34, 1-40
defined 6-2, 6-15
Defining subsystem
functional specifications 2-16
design process 1-44
design review 1-109, 1-117, 3-2
Design Review 2-58
development organization 1-145
DFD 1-6, 1-27, 1-38, 1-40, 1-87
dialog box 3-68, 3-75
directing 6-3
disaster prevention facilities 5-10
division into subsystems 2-15
document 3-47, 3-112, 3-118
duration 6-2

[E]

Earned value (EV) 6-18
Earned Value Management (EVM) 6-17
editor 1-120, 1-121, 1-134
emergency maintenance 5-27
encapsulation 1-84, 1-89, 1-90
encryption 5-14
entity 1-30, 1-138
Entity 2-35
entry/exit log 5-12
ER model 2-35
event-driven 1-37
evolutionary 6-15

experience 6-2
external coupling 4-14
external design 1-110
External Design Documents 2-55
External Design Procedures 2-5

[F]

failure tendency monitoring 5-33
fixed cost 5-15
flowchart 3-78
flowcharts 1-48
FORTRAN 1-63, 1-75, 1-140
forward engineering 1-130, 1-131
functional programming 1-8, 1-78
functional structure 6-2

[G]

Gantt chart 6-22, 6-25, 6-35
goals 6-2
groupware 1-123, 1-135
GUI 1-24, 1-37, 1-122, 1-123

[H]

hardware configuration 2-9
hardware resource
management 5-4
hardware trouble 5-7, 5-8
HCP 1-73, 1-87, 1-121
hierarchical team 6-7, 6-9
high-level language 1-140
HIPO 1-74, 1-87, 1-121
HTML 1-142

[I]

identification key 2-37
important phenomena
monitoring 5-33
indexed sequential file 3-82, 3-106
information hiding 1-49, 1-58
inheritance 1-83, 1-90
initial 6-15
Input-output Specifications 2-57
inspection 1-109, 1-110, 1-116
instruction coverage 1-104
Integration test 2-51
integrity measure 5-16
interfaces between modules 1-108
internal design 1-50, 1-109, 1-110
interpreter 1-78, 1-81, 1-88, 1-89
invalid equivalence class 1-100

inventory status 5-33
ISO/IEC 9126 1-139

[J]

Jackson method 4-2, 4-7, 4-8, 4-21
Java 1-8, 1-79, 1-81, 1-82, 1-89
job flow diagram 2-14, 2-24
job scheduling 5-16

[K]

key process area 6-15
Key process area (KPA) 6-15

[L]

librarian 6-6, 6-9
library management 5-4
logic programming 1-8, 1-75
logical strength 4-11
low-level language 1-140

[M]

maintenance 6-15
maintenance administrator 5-30
maintenance contract 5-27
managed 6-15
management functions 6-3
maturity 6-15
maturity level 6-15
Maturity Levels 6-15
maximum abstraction input
point 4-2, 4-21
maximum abstraction output
point 4-2, 4-21
member 6-6, 6-8
message 1-8, 1-31, 1-84, 1-85
method 1-4, 1-6, 1-15, 1-18, 1-23
Metrics 6-29
moderator 1-23, 1-37
module 1-6, 1-7, 1-8, 1-18, 1-45
module coupling 4-13, 4-14, 4-15
module independence 4-10, 4-13
module strength 4-10, 4-13, 4-21

[N]

natural language 1-33, 1-40
network resource
management 5-5
non-procedural language 1-140
NS chart 1-71, 1-87, 1-121

[O]

object 1-8, 1-15, 1-20, 1-26, 1-30
object-oriented design 1-50, 1-56
object-oriented language 1-79
object-oriented programming 1-8
OCR 3-37
operation manual 5-16
Operation test 2-51
optimized 6-15
organizing 6-3

[P]

PAD 1-72, 1-87, 1-121
Pascal 1-140
password 5-11, 5-13
performance balance 5-33
PERT 6-22, 6-23
physical data design 3-2, 3-4, 3-78
PL/I 1-63, 1-140
Planned value (PV) 6-18
planning 6-3
PMS (project master
schedule) 6-18
pointer 1-78, 1-88
polymorphism 1-84, 1-85, 1-90
post maintenance 5-27
power distribution facilities 5-10
practices 6-2, 6-15
prevention of illicit use 5-4
preventive maintenance 5-26
problem management 5-6
procedural language 1-140
procedural programming 1-46, 1-47
procedural strength 4-11, 4-12
procedures 6-2
process 1-2, 1-4, 1-10, 1-12, 1-13
process improvement 6-15
process management 1-121, 1-129
process model 1-10, 1-12, 1-13
processing mode 3-30
Products 6-4
program design document 4-3
Program test 2-51
programmer 1-17, 1-69, 1-138
programming 1-7, 1-14, 1-15, 1-16
progress management 6-20
project leader 6-8, 6-9
Project Management Body of
Knowledge (PMBOK) 6-12
Project Management Institute
(PMI) 6-12
Project management items 6-4
project manager 6-2, 6-7, 6-8, 6-9
project plan 6-3
Project scope 6-4
Prolog 1-8, 1-75, 1-76, 1-79

proportional allocation 5-15

[Q]

quality 6-15
quality characteristics 1-113

[R]

relationship 1-5, 1-6, 1-23, 1-24
Relationship 2-35
reliability growth curve 1-111
remote maintenance 5-27
repeatable 6-15
repository 1-130, 1-131
requirement definition 1-32, 1-124
requirement specification 1-28
resource management 5-4
resources 6-2
reusable 1-138, 1-142
reuse 3-2, 3-5, 3-108, 3-113
reverse engineering 1-130, 1-131
review 1-109, 1-110, 1-117
running cost 5-15

[S]

schedule 1-18, 6-3
Schedule variance (SV) 6-18
scheduled maintenance 5-26
Schedules 6-4
scheduling 6-22, 6-27, 6-28
scope of systematization 2-10
security policy 2-19
SEI 6-15
size 6-2
skill 6-2
software capability 6-16
software configuration 2-9
Software Configuration
Management (SCM) 6-32
software development 6-2
Software Engineering
Institute 6-15
software life cycle 1-70, 1-120
software lifecycle model 6-2
software resource
management 5-4
software trouble 5-7, 5-8
SPD 1-73, 1-87
specialist team 6-6
specified monitoring 5-33
spiral model 1-12, 1-13, 1-35, 1-41
staffing 6-3
stamp coupling 4-15
standardization 1-141
Standards for Information
System Safety Measures 5-9

storage facilities	5-12	systematization requirements		unit tests	1-106, 1-115
structure theorem	1-65	definition	2-3	UPS	5-9
structured analysis	1-6, 1-20, 1-23			user	1-12, 1-14, 1-17
structured charts	1-71, 1-86, 1-121	[T]		user ID	5-13
structured design	1-43, 1-50, 1-58	Tasks	6-4	user manual	2-49
structured design method	4-72, 4-75	TCO	5-15		
structuring	1-47, 1-65	technical specialist	6-6, 6-9	[V]	
STS partitioning method	4-2, 4-9	tentative maintenance	5-27	valid equivalence class	1-100
subordinate module	4-16, 4-21	test	1-7, 1-95, 1-99, 1-100, 1-103	variable cost and expenses	5-15
sub-project team	6-8, 6-9	test coverage	1-108, 1-116	VSAM file	3-82, 3-106
Subsystem interface		test design	2-51		
definition	2-17	text editor	1-123, 1-134	[W]	
Subsystem Relationship		time strength	4-11	walkthrough	1-110, 4-117
Diagram	2-55	top-down approach	1-6	Warnier method	4-2, 4-7, 4-8, 4-9
System Configuration		top-down test	4-89, 4-92	waterfall model	1-12, 1-19, 1-35
Diagram	2-55	Total Cost of Ownership	5-15	WBS	6-18, 6-19, 6-20, 6-21
System Function of		tracking	6-3	white box test	1-104, 1-140, 1-141
Specifications	2-56			window	3-52, 3-58
System Job Flow	2-56	[U]		work breakdown structure	6-3
system operation tools	5-16	UML	1-125, 1-144, 1-145	Work Breakdown Structure	6-20
System test	2-51	unit test	4-82, 4-91, 4-94, 4-95		
system test items	2-52				



<http://www.vitec.org.vn>