Unit 3 Discrete Mathematic

Text A About Discrete Mathematic

1. Introduction to discrete mathematics

Discrete❶ Mathematics is **the general term**❷ for several branches of mathematics, which is based on the study of mathematical structures that are fundamentally discrete rather than continuous. **In contrast to**❸ real numbers that have the **property**❹ of varying "smoothly", the objects studied in discrete mathematics-such as integers, graphs, and statements in logic-do not vary smoothly in this way, but have **distinct**❺, separated values.[注1] Discrete mathematics therefore **excludes**❻ topics in "**continuous mathematics**❼" such as **calculus**❽ and analysis. Discrete objects can often be **enumerated**❾ by integers. More formally, discrete mathematics has been characterized as the branch of mathematics dealing with **countable sets**❿ (sets that have the same **cardinality**⓫ as **subsets**⓬ of the integers, including **rational**⓭ numbers but not real numbers). However, there is no exact, universally agreed, definition of the term "discrete mathematics." Indeed, discrete mathematics is described **less** by what is included **than**⓮ by what is excluded: continuously varying quantities and related **notions**⓯.

Research in discrete mathematics increased in the latter half of the twentieth century partly due to the development of digital computers which operate in discrete steps and store

---

❶ 离散的
❷ 一般术语
❸ 相反地
❹ 性质,性能
❺ 清楚的,有区别的
❻ 排除,排斥
❼ 连续数学
❽ 微积分学
❾ 枚举
❿ 可数集
⓫ 基数
⓬ 子集
⓭ 有理数
⓮ 少于
⓯ 概念

data in discrete **bits**⑯. Concepts and **notations**⑰ from discrete mathematics are useful in studying and describing **objects**⑱ and problems **in branches of**⑲ computer science, such as computer **algorithms**⑳, programming languages, **cryptography**㉑, automated theorem proving, and software development. **Conversely**㉒, computer **implementations**㉓ are significant.

2. Topics in discrete mathematics

Theoretical computer science includes areas of discrete mathematics relevant to computing. It draws heavily on graph theory and logic. Included within theoretical computer science is the study of algorithms for computing mathematical results. **Computability**㉔ studies what can be computed in principle, and has close ties to logic, while **complexity studies**㉕ the time taken by computations. Automata theory and formal language theory are closely related to computability. Computational geometry applies algorithms to **geometrical**㉖ problems, while computer image analysis applies them to representations of images. Theoretical computer science also includes the study of various continuous computational topics.

Logic is the study of the principles of valid reasoning and **inference**㉗, **as well as**㉘ of **consistency soundness**㉙, and **completeness**㉚. For example, in most systems of logic (but not in **intuitionistic logic**㉛) Peirce's law $(((P \to Q) \to P) \to P)$ is a **theorem**㉜. For classical logic, it can be easily **verified**㉝ with a truth table. The study of mathematical proof is particularly important in logic, and has applications to **automated**㉞ theorem proving and

---

⑯ 位数
⑰ 符号
⑱ 对象,物体
⑲ 分支
⑳ 算法
㉑ 密码学
㉒ 相反地
㉓ 实施
㉔ 可计算性
㉕ 复杂研究
㉖ 几何的
㉗ 推断,推理
㉘ 也
㉙ 一致性,可靠性
㉚ 完整性
㉛ 直觉主义逻辑
㉜ 定理,原理
㉝ 验证
㉞ 使自动化

formal **verification**㉟ of software.

Logical formulas are discrete structures, as are proofs, which form finite trees or, more generally, **directed acyclic graphstructures**㊱(with each inference step combining one or more **premise**㊲ branches to give a single conclusion). The truth values of logical formulas usually form a finite set, generally restricted to two values: true and false, but logic can also be continuous-valued, e. g. , fuzzy logic. Concepts such as infinite proof trees or infinite **derivation**㊳ trees have also been studied, e. g. infinitary logic.

Set theory is the branch of mathematics that studies sets, which are collections of objects, such as {blue, white, red} or the (infinite) set of all prime numbers. **Partially**㊴ ordered sets and sets with other relations have applications in several areas.

In discrete mathematics, countable sets (including finite sets) are the main focus. The beginning of set theory as a branch of mathematics is usually marked by Georg Cantor's work distinguishing between different kinds of infinite set, motivated by the study of **trigonometric series**㊵, and further development of the theory of infinite sets is outside **the scope of**㊶ discrete mathematics. [注2] Indeed, **contemporary work**㊷ in descriptive set theory makes extensive use of traditional continuous mathematics.

Graph theory, the study of graphs and networks, is often considered part of combinatory, but has grown large enough and **distinct**㊸ enough, with its own kind of problems, to **be regarded as**㊹ a subject in its own right Which **in all areas of**㊺ math and science have extensive application.

Graphs are one of the prime objects of study in Discrete Mathematics. They are among the most **ubiquitous**㊻ models of both natural and human-made structures. They can model many types of relations and process dynamics in physical, biological and social systems. In computer science, they represent networks of communication, data organization, computational devices, the flow of computation, etc. In Mathematics, they are useful in Geometry and certain parts of **topology**㊼, e. g. **Knot Theory**㊽. Algebraic graph theory

---

㉟ 验证
㊱ 有向无环图结构
㊲ 前提
㊳ 引出,来历
㊴ 部分地
㊵ 三角级数
㊶ 范围
㊷ 当代作品
㊸ 区分
㊹ 看作
㊺ 在所有的领域
㊻ 普遍存在的
㊼ 拓扑学
㊽ 结点理论

has close links with group theory. There are also continuous graphs, however for the most part research in graph theory falls within the **domain**[49] of discrete mathematics.

Operations research provides techniques for solving practical problems in business and other fields—problems such as allocating resources to maximize profit, or scheduling project activities to minimize risk. Operations research techniques include linear programming and other areas of **optimization**[50], queuing theory, scheduling theory, network theory. Operations research also includes continuous topics such as **continuous-time Markov process**[51], **continuous-time martingales**[52], **process optimization**, **and continuous and hybrid**[53] control theory.

Although topology is the field of mathematics that **formalize**[54] and **generalizes**[55] the **intuitive**[56] notion of "**continuous deformation**[57]" of objects, it gives rise to many discrete topics; this can be attributed in part to the focus on topological invariants, which themselves usually take discrete values. See **combinatorial topology**[58], **topological graph theory**[59], **topological combinatorics**[60], **computational topology**[61], **discrete topological space**[62], **finite topological space**[63].

Words

algorithm	*n*.（名词）	算法;算法式(algorithm 的复数)
automate	*adj*.（形容词）	自动化的;机械化的
bit	*n*.（名词）	比特,位
calculu	*n*.（名词）	结石;微积分学
cardinality	*n*.（名词）	基数;集的势
computability	*n*.（名词）	可计算性
conversely	*adv*.（副词）	相反地

---

[49] 领域
[50] 优化
[51] 连续时间马尔可夫过程
[52] 连续时间鞅
[53] 过程优化及连续混合控制理论
[54] 形式化
[55] 一般化
[56] 直觉
[57] 连续变形
[58] 组合拓扑
[59] 拓扑图论
[60] 拓扑组合
[61] 计算拓扑
[62] 离散空间
[63] 有限拓扑空间

cryptography	*n*.（名词）	密码学;密码使用法
derivation	*n*.（名词）	引出;来历;词源
discrete	*adj*.（形容词）	离散的,不连续的
distinct	*adj*.（形容词）	明显的;独特的;清楚的;有区别的
domain	*n*.（名词）	领域;域名;产业;地产
enumerate	*n*.（名词）	枚举类型
exclude	*vt*. & *vi*.（动词）	排除;排斥;拒绝接纳;逐出
formalize	*vt*. & *vi*.（动词）	使形式化;使正式;拘泥礼仪
generalize	*vt*. & *vi*.（动词）	概括;推广;使……一般化
geometrical	*adj*.（形容词）	几何的,几何学的
hybrid	*n*.（名词）	混合
implementation	*n*.（名词）	实施
inference	*n*.（名词）	推理;推论;推断
intuitive	*adj*.（形容词）	直觉的;凭直觉获知的
notation	*n*.（名词）	符号;记法
notion	*n*.（名词）	观念;小商品
object	*n*.（名词）	物体;对象
optimization	*n*.（名词）	最佳化,最优化
partially	*adv*.（副词）	部分地;偏袒地
premise	*vt*. & *vi*.（动词）	引出,预先提出;作为……的前提
property	*n*.（名词）	性质,性能;财产;所有权
rational	*adj*.（形容词）	合理的;理性的
subsets	*n*.（名词）	子集合(subset 的复数)
theorem	*n*.（名词）	定理原理
topology	*n*.（名词）	拓扑学;地质学;局部解剖学
ubiquitous	*adj*.（形容词）	普遍存在的;无所不在的
verification	*n*.（名词）	确认,查证;核实
verified	*adj*.（形容词）	已查清的,已证实的

Phrases

as well as	也;和……一样;不但……而且
be regarded as	被认为是;被当作是
combinatorial topology	组合拓扑学;组合拓扑
complexity studies	复杂性研究
computational topology	计算拓扑
consistency soundness	一致性,可靠性
contemporary work	当代作品

continuous deformation	连续形变
continuous mathematics	连续数学
continuous-time Markov process	连续时间的马尔可夫过程
continuous-time martingales	连续时间鞅
countable sets	可数集
directed acyclic graphstructures	有向无环图结构
discrete topological space	离散拓扑空间
distinct enough	截然不同
finite topological space	有限拓扑空间
in all areas of	在各方面的
in branches of	分支的
in contrast to	相比之下
intuitionistic logic	直觉主义逻辑
knot theory	结点理论;[数] 纽结理论
the general term for	一般的术语
the scope of	范围
topological combinatorics	拓扑组合
topological graph theory	图论拓扑
trigonometric series	三角级数

Exercises

【Ex1】 Answer the questions according to the text:

(1) What is Discrete Mathematic?

(2) Why did Discrete Mathematic develop so fast in the twentieth century?

(3) How many topics are there in this chapter, and what are they?

(4) What are the logic formulas?

(5) What does Operations research involve?

【Ex2】 Translate into Chinese:

(1) Graph theory is an old subject with modern applications.

(2) Relation between elements of sets is represented using the structure called a relation.

(3) Much of discrete mathematic is developed of discrete structures, which are used to represent discrete objects.

(4) Discrete mathematic is the gateway to more advanced courses in all parts of the mathematical sciences.

(5) The computer chip is primarily responsible for executing instructions.

(6) Tape must be read or written sequentially, not randomly.

(7) Deselect the text by clicking anywhere outside of the selection on the page or pressing an arrow key on the keyboard.

(8) Faster than many types of parallel port, a single USB port is capable of chaining many devices without the need of a terminator.

【Ex3】 Choose the best answer:

(1) Very long, complex expressions in program are difficult to write correctly and difficult to _____.

A. defend B. detect C. default D. debug

(2) _____ is the study of the principles of valid reasoning and inference, as well as of consistency, soundness, and completeness.

A. Graph theory B. Logic

C. Topology D. Operation research

(3) The _____ storage area that you can use to copy or move selected text or object among applications.

A. exponent B. order C. temporary D. superior

(4) Software design is a _____ process. It requires a certain _____ of flair on the part of the designer.

A. create, amount B. created, amounted

C. creating, mount D. creative, mounted

批 注

注1 In contrast to 短语引导的独立结构; that have the property of varying "smoothly" 的先行词为 real numbers, 属定语从句; the objects 为句子的主语; studied 为谓语; do not vary smoothly in this way, but have distinct, separated values. 补充说明。

注2 The beginning of set theory 为主语; as a branch of mathematics 为方式状语; is marked, motivated, and further development of the theory of infinite sets 为句子谓语。

Text B Tree

A **connected graph**❶ shown in Figure 3-1, that contains no simple **circuits**❷ is called a tree. Trees were used as long ago as 1857, when the English mathematician Arthur Cayley

❶ 连通图

❷ 电路, 回路

used them to count certain types of chemical **compounds** ❸. Since that time, trees have been employed to solve problems **in a wide variety of**❹ **disciplines**❺.

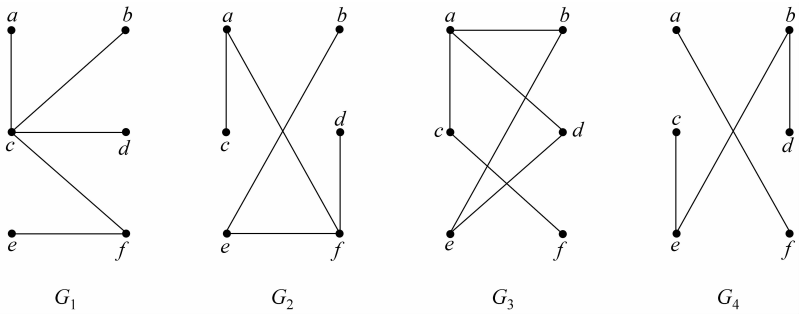

Figure 3-1 Graphs

Tree are particularly useful in computer science, for instance, trees **are employed to**❻ construct efficient algorithms for locating **items**❼ in a list. They are used to construct network with the least expensive set of telephone lines storing and **transmitting**❽ data.[注1] Trees can model procedures that **are carried out**❾ using a sequence of decisions. This makes trees valuable **in the study**❿ of sorting algorithms.

1. Definitions

A tree is an undirected simple graph G that satisfies any of the following **equivalent**⓫ conditions[注2]:

- G is connected and has no cycles.
- G has no cycles, and a simple cycle is formed if any edge is added to G.
- G is connected, and it is not connected anymore if any edge is removed from G.
- G is connected and the 3-vertex complete graph K_3 is not a minor of G.[注3]
- Any two **vertices**⓬ in G can be connected by a unique simple path.

If G has finitely many vertices, say n of them, then the above statements are also equivalent to any of the following conditions:

- G is connected and has $(n-1)$ edge.
- G has no simple cycles and has $(n-1)$ edge.

---

❸ 混合物
❹ 以更广泛的
❺ 学科,分支
❻ 用于
❼ 条款,项目
❽ 传播;发射
❾ 执行
❿ 研究
⓫ 等价的
⓬ 结点

Which of the graphs shown in Figure 3-1 are trees?

G_1 and G_2 are trees, since both are connected graphs with no simple circuits; G_3 is not a tree because e . b . a . d . e is a simple circuit. [注4]

In this graph. Finally, G_4 is not a tree since it is not connected.

2. Facts

(1) A tree with n vertices has $(n-1)$ edges.

(2) A full m-ary tree with i internal vertices contains $n = m * i + 1$ vertices.

(3) A full m-ary tree with.

(i) n vertices has $i = (n-1)/m$ **internal**⑬ vertices and $L = [(m-1) * n + 1]/m$ leaves.

(ii) i internal vertices has $n = m * i + 1$ vertices and $L = (m-1) * i + 1$ leaves.

(iii) L leaves has $n = (m * L - 1)/(m-1)$ vertices and $i = (L-1)/(m-1)$ internal vertices.

3. Tree spices

(1) M-ary tree is a tree with the property that every internal **vertex**⑭ has no more than m children.

(2) Binary tree is an m-ary tree with m = 2 (each child may be designated as a left or a right child of its parent).

(3) Ordered tree is a tree in which the children of each internal vertex are **linearly**⑮ ordered. [注5]

(4) Balance tree is a tree in which every vertex is at level h or h-1. where h is the height of the tree. [注6]

(5) Binary search tree is a binary tree in which the vertices are labeled with items **so that**⑯ a label of a vertex is greater than the labels of all vertices in the left subtree of this vertex is an less than the label of all vertices in the right sub tree of this vertex.

(6) Decision tree is a rooted tree where each vertex represents a possible outcome of a decision and the leaves represent the possible solutions.

(7) **Spanning tree**⑰: a tree containing all vertices of a graph.

(8) **Minimum spanning tree**⑱: a spanning tree with smallest possible sum of weights of its edges. [注7]

---

⑬ 内部的
⑭ 结点
⑮ 呈直线地
⑯ 致使
⑰ 生成树
⑱ 最小生成树

4. Tree traversal algorithms

Procedures for systematically visiting every vertex of an ordered rooted tree are called **traversal algorithms**⑲. We will describe three of the most commonly used such algorithms preorder traversal, inorder traversal, and postorder traversal.

(1) Preorder traversal

Preorder traversal⑳ is listing of the vertices of an ordered rooted tree defined recursively by specifying that the root is listed. Followed by the first subtree, followed by the other sub trees in the order they occur from left to right.

Let T be an ordered rooted tree with root r. if T consists only of r, then r is the preorder traversal of T. Otherwise, suppose that $T_1, T_2, \cdots, T_n$ are the **sub trees**㉑ at r from left to right in T. The preorder traversal begins by visiting r. It continues by traversing T_1 in preorder, then T_2 in preorder, and so on, until T_n is traversed in preorder.

(2) Inorder traversal

Inorder traversal㉒ is **a listing of**㉓ the vertices of an ordered rooted tree **defined** recursively **by**㉔ specifying that the first sub tree is listed **followed by**㉕ the root, followed by the other sub trees in the order they occur from left to right.

Let T be an ordered rooted tree with root r. if T consists only of r, then r is the inorder traversal of T. Otherwise, suppose that $T_1, T_2, \cdots, T_n$. are the subtrees at r from left to right in T. The inorder traversal begins by traversing T_1 in inorder. Then visiting r, It continues by traversing T_2 in inorder, then T_3 in inorder, $\cdots$, And finally T_n in inorder.

(3) Postorder traversal

Postorder traversal㉖ is a listing of the vertices of an ordered rooted tree defined recursively by specifying that the sub trees are listed in the order they occur from left to right, followed by the root.[注8]

Let T be an ordered rooted tree with root r if T consists only of r, then r is the post order traversal of T. Otherwise, suppose that $T_1, T_2, \cdots, T_n$. are the subtrees at r from left to right. The postorder traversal begins by traversing T_1 in postorder, then T_2 in postorder, $\cdots$, Then T_n. in postorder, and ends by visiting r.

---

⑲ 遍历算法
⑳ 前序遍历
㉑ 子树
㉒ 中序遍历
㉓ 列表
㉔ 被定义为
㉕ 紧跟着
㉖ 后序遍历

Words

circuit	*n*.（名词）	电路;环路;巡回
compound	*vt*. & *vi*.（动词）	合成;混合;和解妥协;掺和
connected graph	*n*.（名词）	连接图,[数]连通图
discipline	*n*.（名词）	纪律
equivalent	*adj*.（形容词）	等价的,相等的;同意义的
inorder traversal	*n*.（名词）	中序遍历
internal	*n*.（名词）	内部的;内在的;国内的
item	*n*.（名词）	项目;名目;所有物品
preorder traversal	*n*.（名词）	先序遍历
transmitting	*vt*. & *vi*.（动词）	传递,发射
vertex	*n*.（名词）	顶点;[昆]头顶;[天]天顶
vertice	*n*.（名词）	制高点;天顶;头顶

Phrases

be carried out	进行
be employed to	用来
defined by	定义为
followed by	紧随其后
in a wide variety of	各种各样的
in the study	在这项研究中

批　注

注 1　storing and transmitting data 用于修饰 network,作定语。此句话的含义是：他们用花销代价最少的电话线构建网络,用它来存储,传输数据。

注 2　此句话的含义是：一个树是一个无向简单图 G,它满足以下任何等价条件。

注 3　G 是连通的,3 个顶点构成了图 K_3,它不是一个 G 图的镜像图。

注 4　图 G_1 和图 G_2 是树,因为它们都是有向图,没有回路的有向图,而图 G_3 不是树,因为 *ebade* 是一个简单的回路。

注 5　主语：Ordered tree;系动词：is;in which 引导一个定语从句;the children 为定从的主语;are ordered 是定从的谓语。这句话的含义是：有序树是它的内部子顶点都是线性地排序的树。

注 6　主语：Balance tree;系动词：is;in which 引导一个定语从句;every vertex 为从句的主语。这句话的含义是：平衡树是它的每个顶点都在 h 或 $h-1$ 层,而 h 是该树的高度。

注 7　最小生成树：它的边权总数为最小的树。

注 8　主语：Postorder traversal;谓语：is;表语：a listing of the vertices of an ordered rooted tree;by 引导状语主语。

Associated Reading

Topics in Discrete Mathematics

Discrete mathematics is the study of mathematical structures that are fundamentally discrete rather than continuous. The main topics in discrete mathematics have as the following:

(a) Theoretical computer science
(b) Information theory
(c) Logic
(d) Set theory
(e) Combinatorics
(f) Graph theory
(g) Probability
(h) Number theory
(i) Algebra
(j) Calculus of finite differences, discrete calculus or discrete analysis
(k) Geometry
(l) Topology
(m) Operations research
(n) Game theory, decision theory, utility theory, social choice theory
(o) Discretization
(p) Discrete analogues of continuous mathematics
(q) Hybrid discrete and continuous mathematics

相关离散数学主题:

(a) 理论计算机科学
(b) 信息论
(c) 逻辑学
(d) 集合论
(e) 组合数学
(f) 图论
(g) 概率论
(h) 数论
(i) 代数
(j) 差分演算,离散模型符号验证或者离散分析
(k) 几何学
(l) 拓扑学

（m）运筹学
（n）博弈论、决策论、效用理论、社会选择理论
（o）离散化
（p）连续数学的离散近似
（q）离散和连续混合数学

Unit 4 Software Engineering

Text A Software Processes

A software process is a set of activities that leads to the production of a software product. These activities may involve the development of software from **scratch**❶ in a standard programming language like Java or C. Increasingly, however, new software is developed by extending and modifying existing systems and by **configuring**❷ and **integrating**❸ off-the-shelf software or system components. 注1

Software processes are complex and, like all **intellectual**❹ and creative processes, rely on people making decisions and judgments. **Because of**❺ the need for judgment and creativity, attempts to **automate**❻ software processes **have met with**❼ limited success. Computer-aided software engineering (CASE) tools can support some process activities. However, there is no possibility, **at least**❽ in the next few years, of more extensive automation where software **takes over**❾ creative design from the engineers involved in the software process. 注2

One reason the effectiveness of CASE tools is limited is because of the **immense**❿ diversity of software processes. There is no ideal process, and many organizations have developed their own approach to software development. Processes have evolved to exploit the capabilities of the people in an organization and the specific characteristics of the systems that are being developed. For some systems, such as critical systems, a very structured development process is required. For business systems, with rapidly changing requirements, a flexible, agile process is likely to be more effective.

Although there are many software processes, some **fundamental**⓫ activities are common to all software processes:

---

❶ 草稿
❷ 配置
❸ 合并;整合
❹ 智力的
❺ 由于
❻ 使自动化
❼ 实现;遇见
❽ 至少
❾ 接管,负责
❿ 极大的
⓫ 基本的

(1) **Software specification** the functionality of the software and **constraints**⑫ on its operation must be defined.

(2) **Software design and implementation** the software to meet the specification must be produced.

(3) **Software validation**⑬ the software must be validated to ensure that it does what the customer wants.

(4) **Software evolution** the software must evolve to meet changing customer needs.

Although there is no 'ideal' software process, there is **scope**⑭ for improving the software process in many organizations. Processes may include outdated techniques or may not take advantage of the best practice in industrial software engineering. Indeed, many organizations still do not **take advantage of**⑮ software engineering methods in their software development.

Software processes can be improved by process standardization where the **diversity**⑯ in software processes across an organization is reduced. This leads to improved communication and a reduction in training time, and makes automated process support more economical. Standardization is also an important first step in introducing new software engineering methods and techniques and good software engineering practice.

Software process models

A software process model is an abstract representation of a software process. Each process model represents a process from a particular **perspective**⑰, and thus provides only **partial**⑱ information about that process.[注3] In this section, I introduce a number of very general process models (sometimes called process **paradigms**⑲) and present these from an **architectural perspective**⑳. That is, we see the framework of the process but not the details of specific activities.

These **generic**㉑ models are not **definitive**㉒ descriptions of software processes. Rather, they are abstractions of the process that can be used to explain different approaches to

---

⑫ 约束
⑬ 有效性
⑭ 范围
⑮ 充分利用
⑯ 差异
⑰ 观点;客观判断力
⑱ 部分的
⑲ 范例
⑳ 体系结构角度
㉑ 一般的;普通的
㉒ 最佳的;最完整可靠的

software development. You can **think of** them **as**㉓ process frameworks that may be extended and adapted to create more specific software engineering processes.[注4]

(1) **The waterfall model**㉔ this takes the fundamental process activities of specification, development, validation and evolution and represents them as separate process **phases**㉕ such as requirements **specification**㉖, software design, implementation, testing and so on.[注5]

(2) **Evolutionary development**㉗ this approach **interleaves**㉘ the activities of specification, development and validation. An initial system is rapidly developed from abstract specifications. This is then refined with customer input to produce a system that satisfies the customer's needs.

(3) **Component-based software engineering**㉙ this approach is based on the existence of a significant number of reusable components. The system development process focuses on integrating these components into a system rather than developing them from scratch.

The waterfall model

The first published model of the software development process was **derived**㉚ from more general system engineering processes (Royce, 1970). This is illustrated in Figure 4-1. Because of the **cascade**㉛ from one phase to another, this model is known as the waterfall model or software life cycle. The principal stages of the model map onto fundamental development activities:

i. Requirements analysis and definition the system's services, constraints and goals are, established by **consultation**㉜ with system users. They are then defined **in detail**㉝ and **serve as**㉞ a system specification.

ii. System and software design the systems design process **partitions**㉟ the requirements to **either** hardware **or**㊱ software systems. It establishes overall system architecture. Software design involves identifying and describing the fundamental software

---

㉓ 把……认为是
㉔ 瀑布模型
㉕ 阶段
㉖ 规范;说明书
㉗ 演化开发模型
㉘ 交替
㉙ 基于组件的软件工程
㉚ 派生
㉛ 嵌套
㉜ 咨询
㉝ 详尽的
㉞ 作为
㉟ 部分;分割
㊱ 或……或

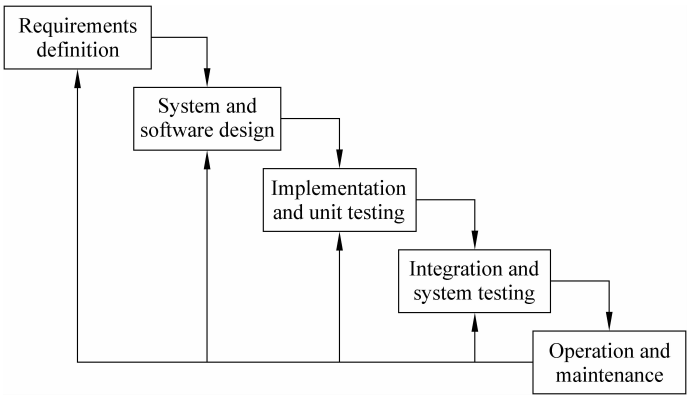

Figure 4-1 The software life cycle

system abstractions and their relationships.

iii. Implementation and unit testing during this stage, the software design is realized as a set of programs or program units. Unit testing involves **verifying**[37] that each unit meets its specification.

iv. Integration and system testing the individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer. [注6]

v. Operation and maintenance[38] normally (although not necessarily) this is **the longest life-cycle phase**[39]. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered. [注7]

In principle, the result of each phase is one or more documents that are approved ('Signed off'). The following phase: should not start until the previous phase has finished. In practice, these stages **overlap**[40] and **fleed** information **to**[41] each other. [注8] During design, problems with requirements are identified; During coding design problems are found and **so on**[42]. The software process is not a simple linear model but involves a sequence of **iterations**[43] of the development activities.

Because of the costs of producing and approving documents, iterations are costly and

---

[37] 核实;核准
[38] 操作与维护
[39] 最长的生存阶段
[40] 部分重叠
[41] 反馈,传递
[42] 等等
[43] 迭代

involve significant **rework**㊹. Therefore, after a small number of iterations, it is normal to freeze parts of the development, such as the specification, and to continue with the later development stages.[注9] Problems are left for later resolution, ignored or programmed around. This premature freezing of requirements may mean that the system won't do what the user wants. It may also lead to badly structured systems as design problems are **circumvented by**㊺ implementation **tricks**㊻.

During the final life-cycle phase (operation and maintenance), the software is put into use. Errors and **omissions**㊼ in the original software requirements are discovered. Program and design errors emerge and the need for new functionality is identified. The system must therefore evolve to remain useful.[注10] Making these changes (software maintenance) may involve repeating previous process stages.

The advantages of the waterfall model are that documentation is produced at each phase and that it fits with other engineering process models. Its major problem is its **inflexible**㊽ partitioning of the project into distinct stages. **Commitments**㊾ must be made at an early stage in the process, which makes it difficult to respond to changing customer requirements.

Therefore, the waterfall model should only be used when the requirements are well understood and unlikely to change **radically**㊿ during system development.

However, the waterfall model reflects the type of process model used in other engineering projects. Consequently, software processes based on this approach are still used for software development, particularly when the software project is part of a larger systems engineering project.

Words

automate	*vt.* & *vi.*（动词）	自动化,自动操作
cascade	*vt.* & *vi.*（动词）	嵌套
circumvented	*vt.* & *vi.*（动词）	包围;陷害;绕行
commitment	*n.*（名词）	承诺,保证;委托;承担义务;献身
configuring	*vt.* & *vi.*（动词）	配置;使成形
constraint	*n.*（名词）	约束;限制;约束条件
consultation	*n.*（名词）	咨询;磋商;[临床] 会诊;讨论会

㊹ 重做
㊺ 由……包围
㊻ 技巧;诡计
㊼ 省略
㊽ 死板的;硬的
㊾ 承诺;保证
㊿ 完全的;彻底的

definitive	*adj*.（形容词）	决定性的;最后的;限定的
derived	*vt*. & *vi*.（动词）	得到;推断
fundamental	*adj*.（形容词）	基本的,根本的
generic	*adj*.（形容词）	类的;一般的;属的;非商标的
immense	*adj*.（形容词）	巨大的,广大的;无边无际的
inflexible	*adj*.（形容词）	顽固的;不可弯曲的
integrating	*vt*. & *vi*.（动词）	整合;积分;集成化
intellectual	*adj*.（形容词）	智力的;聪明的;理智的
interleave	*n*.（名词）	交替
iteration	*n*.（名词）	迭代次数;反复
omission	*n*.（名词）	疏忽,遗漏;省略;冗长
overlap	*n*.（名词）	重叠;重复
paradigm	*n*.（名词）	范例,模范
partial	*adj*.（形容词）	局部的;偏爱的;不公平的
partition	*vt*. & *vi*.（动词）	分开
perspective	*adj*.（形容词）	透视的
phase	*n*.（名词）	阶段,时期
radically	*adv*.（副词）	根本上;彻底地;以激进的方式
scope	*n*.（名词）	范围
scratch	*n*.（名词）	草稿
specification	*n*.（名词）	规格;说明书;详述
validation	*n*.（名词）	确认;批准;生效
verifying	*vt*. & *vi*.（动词）	验证;核查

Phrases

architectural perspective	体系结构角度
at least	至少
be delivered to	送到
because of	因为;由于
component-based software engineering	基于组件的软件工程
evolutionary development	演进式开发
fleed…to	反馈,传递
in detail	详细地
in earlier stages of the life cycle	在生命周期的前期
take advantage of	利用
takes over	接管
the longest life-cycle phase	最长的生命周期阶段

the waterfall model	瀑布模型
think of	记起,想起;考虑;想象;关心

Exercises

【Ex1】 Answer the questions according to the text:

(1) What is the software process?

(2) What are the fundamental activities common to all software processes?

(3) How to improve the software process?

(4) Why do we need to freeze parts of the development after a small number of iterations?

(5) Why must the system evolve to remain useful?

【Ex2】 Translate into Chinese:

(1) A software process is a set of activities that leads to the production of a software product.

(2) Because of the need for judgement and creativity, attempts to automate software processes have met with limited success.

(3) One reason the effectiveness of CASE tools is limited is because of the immense diversity of software processes.

(4) For business systems, with rapidly changing requirements, a flexible, agile process is likely to be more effective.

(5) A software process model is an abstract representation of a software process.

(6) The system development process focuses on integrating these components into a system rather than developing them from scratch.

(7) During this stage, the software design is realized as a set of programs or program units.

(8) The advantages of the waterfall model are that documentation is produced at each phase and that it fits with other engineering process models.

【Ex3】 Choose the best answer

(1) _____ means "Any HTML document a HTTP Server".

A. Web server B. Web page

C. Web browser D. Web site

(2) The term "_____ program" means a program written in high-level language.

A. compiler B. executable C. source D. object

(3) Very long complex expressions in program are difficult to write correctly and