第一篇　软件工程

在学习了“高级程序设计语言”和“数据结构”课程以后，编写小型程序不会有太大的问题，但要研制开发一个大型软件系统还会有许多困难。例如，在接到项目后，应该从哪儿入手、用什么方法、按照哪些步骤进行开发，如何评价一个软件的好坏等，这些都是初次参加大型软件系统开发人员要遇到的问题。因此，必须学习“软件工程”。

第一章　软件工程概述

“软件工程”是 20 世纪 60 年代末期发展起来的一门新兴学科，主要讲解如何用规范化的方法和技术进行大型软件的研制和开发。

本章先介绍软件的一些基本概念，然后在讨论“软件危机”的基础上，引出“软件工程”的概念，简要介绍目前常用的几种软件开发方法和软件开发工具。

1.1　软件技术概述

1.1.1　软件的概念与特点

计算机程序是一系列指令序列的集合，它能被计算机理解和执行。文档是指用自然语言或者形式化语言所编写的文字资料和图表，用来描述程序的内容、组成、设计、功能规格、开发情况、测试结果及使用方法，如程序设计说明书、流程图、用户手册等。

软件是计算机系统中与硬件子系统相互依存的另一个子系统，是一个包含程序及其文档资料的完整集合，提供了用户与硬件子系统之间的接口。

随着计算机科学技术的发展，人们对软件的认识也在不断深化，这从下面式子的变化就可以看出：软件＝程序（20 世纪 70 年代以前）软件=程序＋文档（20 世纪 70 年代）；软件＝文档＋程序（20 世纪 80 年代以后）。在软件的可维护性变得越来越重要的今天，文档的地位也提高到了前所未有的高度，并且能够自动化地生成。

与小型软件不同，大型软件具有以下特点：

(1) 规模大。现在的软件大多上百兆，需要处理的数据量大，占用的内存也大。对于实时软件，除了规模大以外，还要求可靠性高。

需要全本请访问 www.ertongbook.com

(2) 复杂性高。大型软件由大量的模块集成，模块间的关系、调用方式以及数据和文件的关系都相当复杂。

(3) 开发周期长。大型软件从立项到交付使用，需几十人、几百人经过几个月甚至几年的时间。

(4) 开发、维护和使用人员不同。

(5) 多学科综合。软件开发人员除了具有必备的软件知识外，还应该具有多方面的专业知识和经验。

1.1.2 计算机软件技术

计算机软件技术是指开发计算机软件所需的所有技术的总称。按照软件分支学科的内容划分，计算机软件技术主要有以下几个领域：

(1) 软件工程技术。其中包括软件开发的原则与策略、软件开发方法与软件过程模型、软件标准与软件质量衡量、软件开发的组织与项目管理和软件工程工具及环境等。

(2) 程序设计技术。其中包括程序的结构与算法设计、程序设计的风格、程序设计语言、程序设计方法和程序设计自动化、程序的正确性证明和程序的变换。

(3) 软件工具环境技术。其中包括人机接口技术、软件自动生成、软件工具的集成、软件开发环境和软件的复用等。

(4) 系统软件技术。其中包括操作系统、编译方法、分布式系统的分布处理与并行计算、并行处理技术和多媒体软件技术。

(5) 数据库技术。其中包括数据模型、数据库与数据库管理系统、分布式数据库、面向对象的数据库技术、工程数据库、多媒体数据库，以及数据仓库和数据挖掘等。

(6) 网络软件技术。其中包括网络软件技术、网络管理、局域网技术、网络互连技术和智能网络等。

1.1.3 软件复用

软件复用是指在构造新的软件系统的过程中，对已存在的软件产品（设计结构、源代码、文档等）重复使用的技术。软件复用有三个层次：知识的复用、方法的复用和软件成分的复用，前两个属于知识工程的范畴，这里只讨论软件成分的复用。软件成分的复用包括以下三个级别：

(1) 代码的复用，可以采用源代码剪贴、源代码包含和继承来实现。

(2) 设计结果的复用，是指复用某个软件系统的设计模型，适用于软件系统的移植。

(3) 分析结果的复用，是指复用某个软件系统的分析模型，适用于用户需求未改变，而系统体系结构变化的场合。

Microsoft 等公司提出了 OLE/COM(Object Linking Embeding/Component Object Model，对象链接和嵌入／组件对象模型）概念，并开发出各种独立的标准组件，用户使用这些组件集成自己的软件，提高了软件的质量，软件维护更加容易，同时降低了软件开发成本。

但像程序的重复运行、执行期间的重复调用等都不是软件复用。由于软件复用是利用

已有的软件成分来构造新的软件，因此大大缩减了软件开发所需的人力、物力、财力和开发时间，并且能提高软件的可靠性和可维护性。

1.2 软 件 危 机

1.2.1 软件危机

“软件工程”起因于“软件危机”。什么是软件危机？简单地说，软件危机是指在软件开发过程中遇到的一系列严重问题，如：开发周期延长，成本增加，可靠性降低等。下面是软件危机的两个例子。

其一，IBM OS/360 系统有 346 万条汇编语句，1968～1978 年共投入 5000 人年工作量，共修改了 21 版，结果不能使用。

其二，1962 年美国飞往金星的探测卫星发射失败，原因是控制系统中的一个 FORTRAN 循环语句 DO 5 I=1,3 被误写成 DO 5 I=1.3，由于空格对 FORTRAN 编译程序没有意义，误写的语句被当成了赋值语句 DO5I=1.3，一点之差，使卫星偏离轨道，只好下令引爆，导致 1850 万美元的损失。

除了不能正常运行的软件，软件危机还反映在以下几个方面：

(1 对软件成本、开发成本和开发进度的估计不准确；软件成本在计算机系统总成本中所占的比例逐年上升；

(2 用户对“已完成的”软件系统不满意的现象时常发生；

(3 软件产品的质量往往靠不住；

(4 软件通常没有适当的文档资料，维护困难。

(5 软件开发生产率的提高速度跟不上计算机应用的普及和深入的趋势。

1.2.2 软件危机产生的原因

在 1946 年第一台计算机“诞生”以后的很长一段时间里，人们都是用计算机来编制一些小程序，解决一些“小问题”。随着计算机软硬件的发展，人们用计算机来解决的问题越来越多，程序规模也越来越大，而开发大型软件与编制小程序有一定的区别，具体体现在：

(1 人员。小程序从确定要求、设计、编制、使用，直到维护，通常由一个人完成；而大型软件则必须由用户、项目负责人、分析员、初级程序员、资料员、操作员等组成一支开发队伍来协同完成。

(2 文档。小程序是编制者脑中的“产品”，很少有书面文档；而大型软件是集体劳动的“产物”，必须有规范化的文档，便于开发和维护。

(3 产品。小程序通常是一次性的，如果需做大的修改，则宁可舍弃旧程序而重新编写；但大型软件的开发耗费了大量的人力与物力，所以不可能轻易抛弃，而总是在旧软件的基础上一再改动，以延长它的使用期，因此“版本”在不断升级。

大型软件的开发提出了许多新的问题，而开发方法却还停留在编制小程序的方法上，经验和技巧已不能满足开发大型软件的需要，从而导致软件开发过程混乱；使用的开发方

法和技术不当，没有适当的文档，不易交流，维护困难，开发成本高，软件质量低等等，这些问题是造成软件危机的主要原因。

1.2.3 软件危机的解决方法

软件危机使人们认识到，软件的研制和开发不能像以前那样——开发过程混乱、无规范化的文档、个体作坊式的开发，而必须立足于科学理论的基础上，像生产产品、研制一台机器或建造一座楼房那样，以“工程化”的思想来指导软件开发，解决软件研制中面临的困难和混乱，从根本上解决软件危机。从技术上，以软件工程技术、程序设计方法和技术为基础，力求将软件工程与知识工程、人工智能技术结合起来，以构造基于知识的软件开发环境；从管理上，以管理学为依托，对开发人员、成本、项目、文档等加强管理，对软件开发全过程进行控制。

1.3 软 件 工 程

1.3.1 软件工程的概念

软件工程是指用工程的概念、原理、技术和方法来开发和维护软件，把经过时间考验证明正确的管理技术和当前能够得到的最好的技术方法结合起来，指导计算机软件的开发和维护的工程学科。

软件工程采用的生命周期方法学是指从时间的角度对软件开发和维护的复杂问题进行分解，把软件生存的漫长周期依次划分为若干阶段，每个阶段都有相对独立的任务，然后逐步完成每个阶段的任务。

把从软件开发项目的提出到软件产品完成使命而报废的整个时期称为软件的生命周期。软件生命周期划分为三个大的阶段：

软件定义阶段：包括问题定义、可行性研究和需求分析三个子阶段；

软件设计阶段：包括总体设计、详细设计、编码和测试四个子阶段；

软件维护阶段：使软件在运行期间满足用户的需要。

上述软件生命周期可以用瀑布模型来表示（见图 1-1）。从图 1-1 可以看出，按照软件生命周期方法学来进行软件的开发，各阶段的工作按自上而下，从抽象到具体的顺序进行，就好像是奔流不息的瀑布，从高向低流淌，因此称为瀑布模型。

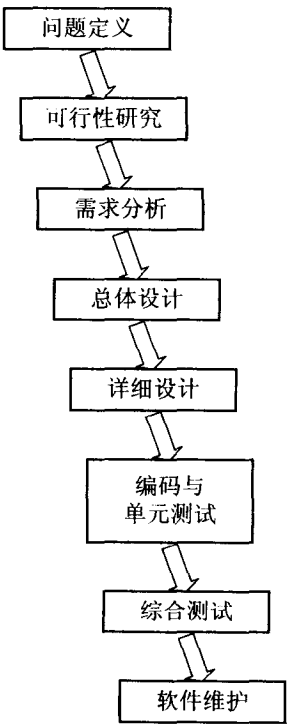


图 1-1 瀑布模型

传统的瀑布模型有如下特点：

(1) 阶段间有顺序性和阶段性。这有两重含义：第一，

必须在前一阶段的工作完成之后，才能开始下一阶段的工作；第二，前一阶段产生的文档，是下一阶段工作的依据。只有前一阶段的工作是正确的，下一阶段工作的正确性才能得到保证。反之，如果前一阶段的工作有错误，等到后面的阶段才发现，则要花费较大的代价。就像水流到下面后，再把水提高需要花费较大的代价一样。

(2) 推迟实现的观点。实践表明，软件规模越大，编码实现的时间越晚越好。因为前一阶段的工作做的不充分，过早地进行编程，往往导致大量的返工，甚至带来灾难性的后果。

(3) 质量保证的观点。为保证开发的软件质量，在瀑布模型的每一个阶段，都应该做到以下两点：

①每个阶段都必须完成规定的、合格的文档；

②每个阶段结束前必须对文档进行严格的复审、验证和确认。

经验表明，越早潜伏的错误，越晚发现，纠正错误所花费的代价也越高。因此，及时审查和纠正错误，是保证软件质量、降低软件成本的重要措施。

1.3.2 软件工程原理

自 20 世纪 60 年代末期出现“软件工程”术语以来，许多学者和软件工程师从系统工程和工程管理学观点出发，根据各国的实践经验，提出了 100 多条软件工程的准则。B.W.Boehm 综合众多学者的经验，总结出七条软件工程基本原理，并认为这些原理是确保软件产品质量和提高开发效率的最小集合。这七条基本原理是：

(1) 严格按照计划进行管理；

(2) 坚持进行阶段评审；

(3) 实行严格的产品控制；

(4) 采用现代化的程序设计技术；

(5) 结果应能清楚地审计；

(6) 开发小组的人员应该少而精；

(7) 承认不断进行软件工程实践的必要性。

1.3.3 软件开发方法简介

目前有很多方法可以用于软件开发，本节简单介绍三种常用的方法：结构化方法、快速原型法和面向对象法，第二章和第三章将详细讨论结构化方法和面向对象法。

1. 结构化方法

结构化方法是目前软件开发的主要方法之一，它以“结构化”的思想、方法和工具进行软件开发。“结构化”的含义是指用一组标准的准则和工具从事某项工作，它最早出自结构化程序设计。在结构化程序设计之前，程序员编写程序没有统一的标准，按各自的习惯编写程序，程序结构往往难以理解。“GOTO 语句是有害的”一文发表后，人们总结以往的经验教训，提出了结构化程序设计的基本思想：只使用顺序、选择、循环三种基本结构来编写程序，它们都是单入口单出口的；使用自顶向下逐步求精的程序设计方法，即利用三种基本结构实现程序结构的连续分解，产生较低层次的结构，直到设计下降到能使用低层

需要全本请访问 www.ertongbook.com

伪代码或高级语言中的三种基本语句表达为止。结构化程序设计是一种成功的程序设计方法，但不能解决系统的结构问题，更不能解决系统总体模型表达方面的问题，但启发人们提出了结构化软件设计的思想。按照结构化软件设计原则，一个系统由层次化的程序模块构成，每个模块只有一个入口和一个出口，每个模块归其上级模块调用。应当构造内部联系紧密的模块，降低模块间的联系。使用系统结构图等图形工具表达系统结构。结构化设计还采用自顶向下的模块化设计方法，但是结构化设计并不能对系统分析有帮助。当问题比较复杂，软件规模较大时，系统分析是必不可少的阶段。不论从用户角度还是从系统结构设计角度，都需要有一个逻辑模型定义系统的逻辑功能。结构化分析是定义系统逻辑模型的一种方法学。

结构化方法把软件开发过程分成调查、分析、设计、编码、测试和维护六个阶段，其中在分析、设计和编码阶段均采用结构化的思想和工具进行软件的开发，如分析阶段的数据流图、数据字典、实体联系图和状态转移图等；设计阶段的软件结构图、层次图等，编码阶段的结构化编程等。

结构化方法具有简单易学、易交流等特点，因而得到了广泛应用，但它也存在着明显的缺陷。第一，它试图在系统建立之前对用户需求进行严格定义或预先加以明确说明，这通常是不切实际的。软件开发的实践表明，用户对系统的需求常常是在学习和使用一个模型系统中体验和明确的。在许多情况下，尽管进行了详细的系统分析和设计，但当系统建成之后，随着软件的使用和用户对问题的深入理解，用户仍然感到所开发的软件不完备、不令人满意，因此要求改变原来的需求，进而提出新需求的事情经常发生。而以后添加、删除和修改系统的代价是昂贵和困难的。第二，用户只参加软件开发的软件定义阶段的工作，在软件投入使用之前不再参与开发，这样不易发现开发中的问题，导致维护代价增加。第三，静态的建模工具（文字和图形）缺乏直观的感性认识，用户希望见到一个能“运行”的模型，如同建筑工程中的大楼模型。一个具体的系统模型，其直观性和动态性比静态模型更容易理解和沟通。

为克服上述缺陷，人们提出了快速原型法。

2. 快速原型法

原型是系统的早期版本，是系统的物理模型，只实现了系统的一些最基本的功能，反映了系统的行为特性，但不一定满足全部需求。

快速原型法是在结构化生命周期的编码阶段之前插入一个建立系统原型的阶段，原型的建立一般分为四步：第一步是确定用户的基本需求，而不是全部需求；第二步是建立一个工作原型；第三步是试用原型；第四步是修改和补充原型。

原型要求快速建立，通常只有几周时间，所以称这种方法为快速原型法。

由于可以“运行”的物理模型比静态的逻辑模型更容易理解和沟通，因此有了一个原型以后，用户可以通过与原型交互及早发现需求中的问题，并向开发人员提供真实的反馈意见；开发人员也可以检查设计的可行性。这样可以在目标系统的详细设计之前较容易地改正原型设计的问题。当然这也需要有快速建立系统原型的工具，其中包括超高级语言。这种通过早期的原型开发活动，由于有用户反馈，开发者修改原型的重复迭代过程，缩短了软件开发周期，降低了开发和维护费用，也提高了用户的满意度。因此，快速原型法成

为目前比较流行的实用的软件开发方法。

3. 面向对象方法

上述软件开发方法仍然存在缺陷，它们使用的核心技术是结构分析与设计技术。这些技术存在着比较明显的缺点：即软件的稳定性、可修改性和可重用性都比较差。

结构分析与设计技术的本质是功能分解，即把一个复杂的目标系统自顶向下不断地分解为子处理，直到这些子处理完成单一功能、容易实现时为止，并写出各个最低层处理的描述。处理描述可以用静态的描述工具如 IPO 图、PDL 语言、判定树等来表示。因此，结构分析与设计技术是围绕实现处理功能的“过程”来构造系统的。而用户需求的改动大部分是针对功能的，因此，这种变化对“面向过程”的设计来说是灾难性的，设计出的系统结构常常是不稳定的，即用户需求的变化往往会引起软件结构的变化，而改变软件结构将花费很大的代价。

结构分析与设计技术清楚地定义了目标系统的边界，软件通过界面与客观世界通信。由于用结构分析与设计技术开发出的软件结构依赖于对系统边界的定义，因此，很难把这样的系统扩展到新的边界。也就是说，这样的系统较难修改和扩充。

结构分析与设计技术，把处理分解成子处理的过程多少有些任意性，也就是说，不同的开发人员开发相同的系统时，可能经分解而得出不同的软件结构。此外，该技术在本质上具有冯·诺依曼计算机的结构特点，把数据和操作作为分离的实体，以致在实现阶段，一些具有潜在可重用价值的软件部件（也称为软构件）已和具体应用环境密不可分，因此开发出的软件复用性较差，或不能实现真正意义上的软件复用。

基于上述种种因素，一种新的软件开发方法——面向对象方法应运而生。面向对象方法（Object-Oriented，OO）尽可能模拟人类习惯的思维方式，使软件开发方法与过程尽可能地接近人类认识世界解决问题的方法与过程。它最主要的特征之一是整个生命周期使用相同的概念、表示法和策略，即每一件事都围绕对象进行。

面向对象方法主要由面向对象分析（Object-Oriented Analysis，OOA）、面向对象设计（Object-Oriented Design，OOD）和面向对象程序设计（Object-Oriented Programming，OOP）组成。

OOA 是软件开发过程中的问题定义阶段。它从对问题的初始陈述开始，运用应用领域知识来识别该领域中的物理实体和概念，提取出对象，分析对象之间存在的相互关系，最后从三个不同的方面建立系统模型，即对象模型、动态模型和功能模型。系统模型描述了系统的对象结构，是对问题领域精确、清晰的定义。对象模型描述了系统的对象结构，动态模型表示系统中的对象与时间和运算次序有关的行为。动态模型对交互系统十分重要。功能模型表示数据如何被处理，可用传统的数据流图等表示系统的功能。

OOD 决定如何将系统组织成子系统，每个子系统分成更小的子系统。较低层的子系统称模块。一个遵循对话独立性原则的交互软件系统常分成用户界面和应用功能核心两个子系统。面向对象系统的控制结构方式可以是过程驱动、事件驱动或共行方式的。

OOP 是将 OOD 的结果用一种程序设计语言实现。通常总是选择一种面向对象的程序设计语言。

1.4 软件工程环境

软件开发手段经历了从手工编码到使用支撑软件产品的自动化软件工具的变迁。现在，从软件的开发、运行到维护各阶段都有软件工具，这些工具形成了现代化软件工程环境的基础。

软件工具是指可以用来帮助开发、测试、分析、维护其它计算机程序的程序以及文档资料的集合，它可以实现软件生产过程自动化，提高软件的生产率、可靠性，降低软件生产成本。软件工具在各种状况下都能被简单、方便地使用，能给软件的开发带来极大的方便大型软件生产所使用的软件工具是一种自动化系统，包括需求分析工具、设计工具、编码工具、确认工具、维护工具等。比如，需求分析工具能够辅助系统分析员把用户所提出的含糊的用户说明，经过分析及一致性、完备性检查后，快速生成指导系统设计用的“需求规格说明书”及其相应的文档资料。设计工具能够依据输入的需求规格说明，自动设计出一系列软件设计文档，如软件结构说明、模块接口说明等。编码工具的主要功能是输入设计阶段产生的文档，自动生成特定语言编制的程序，如各种应用程序生成器等。

尽管软件工具种类繁多，形式多样，但都只是用于软件生存周期中的某一个阶段或某一个环节，而不能对整个生命周期有效。为了能够对软件整个生命周期提供支持，于是出现了软件工程环境的新课题。软件工程环境(Software Engineering Environment,SEE) 是指用以支持需求定义、程序生成，以及软件维护等整个软件生命周期全部活动的，并把方法、规模和计算机程序集成在一起的整个体系。软件工程环境又称为软件开发环境，软件支撑环境，自动开发环境等。

软件工程环境的全部需求可以概括为：

(1) 它应该是集成化的系统；

(2) 它应该是通用的系统；

(3) 它应该是既可剪裁又可扩充的系统；

(4) 它应该是实用的、经济合算的系统。

近几年来，软件工程领域中出现了一种新趋势，即将软件工程方法、工具与环境方面的新技术同形式化语义理论有机地结合起来，形成高水平的计算机辅助软件工程(Computer Aided Software Engineering，CASE）系统，这标志着软件开发技术的发展进入到一个新阶段。

CASE系统可以帮助开发人员执行许多和软件开发有关的艰苦工作，包括对各种计划、合同、规格说明、设计、源代码和管理信息之类的文档进行组织。可以说， CASE系统可以对软件生产过程的每一步提供辅助手段。

习　　题

1. 试说明软件与程序的概念有何不同？
2. 什么是软件复用？软件复用的范围有哪些？
3. 什么是软件危机？软件危机有哪些表现？
4. 什么是软件工程？

5. 试说明软件生命周期的概念。
6. 试比较结构化方法、快速原型法和面向对象方法。
7. 你认为理想的软件工程环境是怎样的？

第二章　结构化分析和设计方法

本章介绍经典的软件开发方法——结构化方法，并以此为基础，介绍软件的开发过程及任务、方法和技术。

2.1　问题的定义

问题定义、可行性研究和需求分析是软件生命周期中的软件定义阶段，而问题定义又是整个软件生命周期的第一个步骤。

问题定义的主要任务是确定“软件要解决的问题是什么？”如果不知道问题是什么，或者只是了解一点皮毛，就急于去开发软件，显然是盲目的，只能白白浪费时间和费用，最终开发出的软件肯定是毫无实际意义。尽管确切定义问题的必要性和重要性是十分明显的，但在实践中又是最容易被忽视的一个步骤。

通过问题定义，系统分析员应该提出关于问题性质、工程目标和规模的书面报告。通过对系统的实际用户和使用部门的访问调查，分析员应该扼要地写出对问题的理解，并在用户和使用部门负责人参加的会议上认真讨论这份报告，澄清含糊的地方，改正分析员理解得不正确的地方，最后得出一份令双方都满意的问题定义文档——《问题目标和规模报告书》，接着就可进入可行性研究阶段。

2.2　可行性研究

可行性研究的主要任务是“确定问题定义阶段定义的问题是否有可行的解？”因为并不是所有问题都有简单明显的解决方法。实际上，有许多问题不能在预定的系统规模之内解决。如果问题没有可行的解，那么花在这项开发上的任何时间、资源、人力和经费都是无谓的浪费，因而，可行性研究十分必要。

可行性研究的目的就是用最小的代价，在最短的时间内确定问题是否能够解决，是否值得去解决。为此，需要从以下四个方面加以分析：

(1) 技术可行性。分析系统采用的技术是否先进，能否实现系统目标，开发人员的素质是否具备等。

(2) 经济可行性。分析目标系统能否用最小的代价获得最大的经济效益、社会效益和技术进步。

(3) 操作可行性。分析系统的操作方式在这个用户范围内是否可行。

(4) 法律可行性。分析系统开发可能造成的责任问题，有无违法行为？如合同的责任问

题、专利版权问题、用于软件开发的软件平台的版权问题等。

问题定义阶段提出的对工程目标和规模的报告通常比较含糊。可行性研究阶段应导出系统的高层逻辑模型（通常用数据流图表示），并且在此基础上更准确、更具体地确定工程规模和目标，然后分析员更准确地估计系统的成本和效益。对建议的系统进行仔细的成本／效益分析是这个阶段的主要任务之一。

可行性研究的结果是使部门负责人作出是否继续进行这项工程决定的重要依据，一般说来，只有投资可能取得较大效益的那些工程项目才值得继续进行下去。对不值得投资的工程项目，系统分析员则应建议终止，可以避免更大的浪费。该阶段产生的文档是《可行性研究报告》。

2.3 结构化分析

可行性研究后，软件系统就可以立项开发，进入软件需求分析阶段。该阶段的任务仍然不是具体解决问题，而是准确地确定“为了解决这个问题，目标系统必须做什么”的问题，主要是确定目标系统必须具备哪些功能。可以从以下五个方面分析系统的综合要求：

(1) 功能要求。指系统必须完成的所有功能。

(2) 性能要求。如联机系统的响应时间，系统需要的存储容量以及系统的健壮性和安全性等方面的要求。

(3) 运行要求。指系统运行所需要的软硬件环境。

(4) 未来要求。指系统将来可能的扩充要求。

(5) 数据要求。指系统所要处理的数据以及它们之间的联系。

系统分析员在需求分析阶段必须和用户密切配合，充分交流信息，分析系统的综合要求，导出经过用户确认的系统逻辑模型。系统逻辑模型完整准确地反映用户的要求，是以后设计和实现目标系统的基础。

需求分析有多种方法和工具，其中结构化分析方法是目前常用的方法之一。

2.3.1 结构化分析方法概述

结构化分析（Structured Analysis,SA）采用软件工程中控制复杂性的两个基本手段：“分解”和“抽象”，将系统自顶向下逐层分解，直到找到满足所有功能要求的可实现软件为止。由于人的理解能力、记忆力有限，对于一个复杂问题不可能触及到问题的所有方面和全部细节。为将复杂性降低到人们可以掌握的程度，常将其拆成若干小问题再分别解决，这就是分解。分解是分层进行的，先考虑问题最本质的属性，暂时略去细节，再逐层添加细节，直至达到必要的详细程度，这就是抽象。逐层分解充分体现了分解和抽象的原则。

SA 方法具有以下特点：

(1) 用户共同参与系统开发，面向用户；

(2) 建立系统的逻辑模型，强调逻辑而不是物理；

(3) 使用图表工具明确表达系统逻辑模型，作为与用户和系统开发人员的通信媒介；

(4) 采用自顶向下的方法进行系统分析；

需要全本请访问 www.ertongbook.com

(5）使用同一份系统分析资料，避免了重复性，增强了一致性。

目前大多数计算机系统都是用来替代一个已存在的系统，它可以是一个计算机系统，也可以是一个人工系统。对这类系统， SA 可按以下步骤进行：

(1) 分析当前系统，导出当前系统的物理模型；

(2）从当前系统的物理模型抽象出当前系统的逻辑模型；

(3）分析目标系统与当前系统逻辑上的差别，建立目标系统的逻辑模型；

(4）补充目标系统的逻辑模型。

SA 方法采用介于形式语言和自然语言之间的描述方式书写系统需求分析说明书。系统需求分析说明书通常包括以下几部分：

(1) 一套分层的数据流图；

(2）一本数据字典；

(3）一组小说明；

(4）补充材料，如表达数据分析结果的实体联系图等。

2.3.2 数据流图

数据流图是描述系统逻辑功能的图形工具。图中没有具体的物理元素如显示终端、磁盘文件、打印输出等。它仅用来表达系统的逻辑功能，即数据在系统内的逻辑流向和数据的逻辑处理。

数据流图有四种基本成分：

(1）外部项（源点或汇点）。外部项是指系统以外的事物或人，它表达了该系统数据的外部来源或去处，用方框表示。方框内是外部项的名字。名字通常是名词，如人或事物。为避免在数据流图上出现数据流的线条交叉，同一外部项可以在一张图上出现若干次（如 n 次），这时须在该外部项的右上角画上 n－1 条斜线。确定了外部项，实际上也就确定了系统和外界的分界线。

(2）处理。处理表达了对数据的逻辑加工或变换功能。对数据的加工处理的结果，或是变换了的数据的结构，或是在原有数据的基础上产生新的数据。处理用圆表示，圆中是处理的名字。名字应恰当地反映处理的含义，使之容易理解，通常是动宾结构。可以用数字对数据流图中的处理编号。一个处理可以对应于一个模块，一个程序，也可以是“穿孔”、“打印输出”或者甚至是“目视检查数据正确性”的人工处理过程。

(3）数据流。数据流指示数据的流动方向，用带箭头的直线或弧线表示。直线或弧线上带有数据流的名称，名称通常是名词。数据流可以由一个外部项产生，也可由某一处理产生，或者来自某一数据存储。

(4）数据存储。数据存储指明了保存数据的地方，它并不代表具体的存储介质。数据可以存储在磁盘、磁带、内存及任何物理介质。数据存储使用右端开口的矩形框表示，框内标有存储的数据的名称，通常是名词。同外部项一样，为避免图中线条交叉，可在一张图中多次出现相同的数据存储，这时只须在矩形左侧加竖线，并标上数据存储的名字。

可以采用自顶向下的方法分层画数据流图，步骤如下：

第一步，提取数据流图中的四个基本成分；

第二步，画出高层数据流图；

第三步，逐层分解较高层数据流图中的处理，得到一套分层数据流图。

数据流图的分解应遵循下列原则：

(1) 分解要自然，概念要合理；

(2) 以分层方式对处理进行编号；

(3) 注意父图与子图的平衡，即子图中所有的输入和输出数据流应当和父图中相应处理的输入和输出数据流一致；

(4) 一个处理一般可分解成 7 ± 2 个子处理，不宜过多；

(5) 当进一步分解可能涉及具体的物理实现手段时，分解应终止。

下面以一个简单的例子说明数据流图的画法。

某工厂采购部门每天要开出定货清单，清单中包括定购部件的部件号、部件名、规格、说明、定购量、当前价格、主要供应商和辅助供应商。部件入库或出库称为业务，通过仓库中的终端把业务报告给定货系统。当某种部件的库存量少于标准线以下时，仓库管理员就应该及时通知定货系统开出定货清单，交由采购员采购。

为画出上述定货系统的数据流图，首先从系统的简述中提取数据流图的四个成分。

(1) 源点和汇点。仓库管理员可视为源点，采购员视为汇点。

(2) 处理。处理通常是系统简述中的动词短语，如产生定货清单，处理库存业务等。

(3) 数据流。从系统的源点流出和流入汇点的数据流即是系统的输入数据流和输出数据流。

(4) 数据存储。确定哪些数据应保存在数据存储中，这需要从时间上考虑。库存业务一旦产生就立即被处理，所以不必保存。定货清单一天只产生一次，故需要保存产生定货清单的数据。有关库存零部件的信息包括定货标准线也应作为数据存储，统称为库存数据。

由上述分析，可得定货系统中数据流图的四个基本成分，见表 2-1。这些成分有的直接从系统问题简述中提取，有的则是隐含在问题简述中。

表 2-1　定货系统数据流图的基本成分

源点/汇点	处理	数据流	数据存储
管理员	产生定货清单	定货清单	定货数据
采购员	处理库存业务	库存业务	库存数据

然后画出系统的高层数据流图(图 2-1)。高层数据流图强调源点、汇点和输入输出数据流。高层数据流图毕竟太抽象了，需要分解其中的处理，得到功能级的数据流图（图 2-2)。注意分解过程中要遵循数据流图的分解原则。

最后通过分解关键处理，对数据流图进行细化，得到细化的数据流图（图 2-3)。

得到系统的数据流图后，必须根据数据流图的分解原则对其进行检查，尤其注意数据是否守恒，父图和子图是否平衡。注意，在图 2-3 中对处理编号的方法。处理 1.1、1.2、1.3 是由上一层数据流图中处理 1 分解得到的，是处理 1 的组成元素。



图 2-1　定货系统的高层数据流图

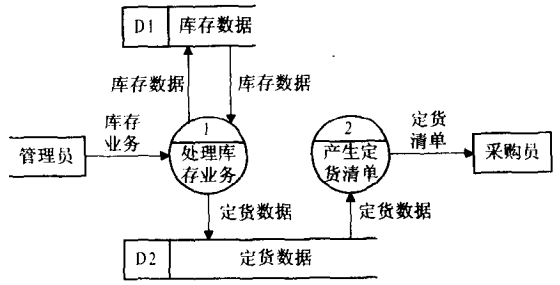


图 2-2　定货系统的功能级数据流图

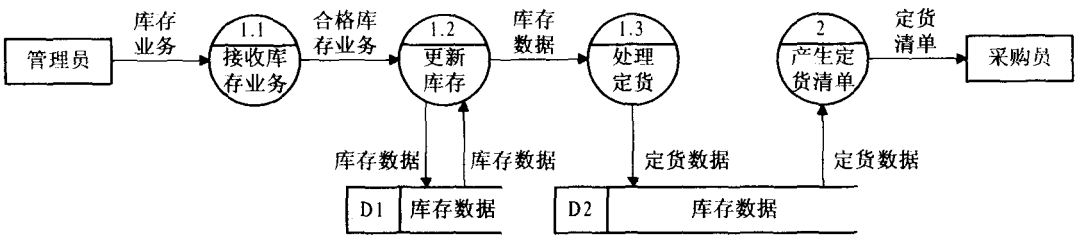


图 2-3　定货系统细化的数据流图

数据流图简洁、清楚地描述了系统的逻辑模型，易于理解和评价。作为信息交流的工具，数据流图易于系统分析员与用户交流。数据流图也是结构化软件设计的基础，由它出发可以映射出软件的结构。

数据流图反映了数据在系统中的流向和数据被加工处理的情况，但它无法详细描述数据流、数据存储、处理逻辑和外部项的内容，这样数据流图就不严格，也难以发挥作用，因此还必须辅以其它工具。这些工具包括数据字典、结构化自然语言、判定表和判定树等。

2.3.3　数据字典

数据字典是关于数据的信息集合，也就是对数据流图中四个基本成分详细定义或说明的集合。

数据字典中数据的最小单位是基本数据元素，基本数据元素是不可再分的数据单位，简称数据元素。若干数据元素，或若干数据元素加上数据的结构构成数据结构。数据流和数据存储中的信息正是这样的数据结构，它们是层次型的。数据字典中包括数据元素和数据结构一览表如表 2-2 和表 2-3 所示。数据字典中有关数据流的定义和说明包含数据流编号、名称、数据流的来源和去处，数据流的组成等内容；数据存储的定义和说明包含数据存储编号、名称、组成等内容；外部项应包括外部项编号、名称、相关数据流等；处理逻辑的表达是一个较为复杂的问题，将在下一节介绍，数据字典中只能作简要记载，以备查询。通常对于处理逻辑的说明包括处理编号、名称，在数据流图中的层号，处理逻辑的简单描述，相关的输入和输出数据流等。处理的功能描述可另外由结构化语言、判定树和判定表等加以表达。表 2-2～表 2-7 是上例的数据字典，其中只列出了一些最基本的内容。

表 2-2　定货系统的数据元素表

编号	数据元素名称	程序内部名	类 型	长 度	小数点后位数
DE01	部件号	part_no	字符	10	
DE02	部件名称	descript	字符	20	
DE03	规格	spec	字符	40	
DE04	单价	price	数值	8	2
DE05	供应商编号	sup_no	数值	3	
DE06	供应商名称	sup_name	字符	30	
DE07	供应商地址	address	字符	50	
⋮					

表 2-3　定货系统的数据结构表

编号	数据结构名称	程序内部名	数据结构组成
DS01	库存数据	part	DE01,DE02,DE03,DE04,DS02
DS02	供应商	supplier	DE05,DE06,DE07
⋮			

表 2-4　定货系统的外部项表

编　号	外部项名称	输入数据流	输出数据流
E01	管理员	—	DF01
E02	采购员	DF02	—

表 2-5　定货系统的数据流表

编　号	数据流名称	来 源	去 处	组成
DF01	库存业务	E01	P01	DS03
DF02	定货清单	P04	E02	DS04
⋮				

表 2-6　定货系统的数据存储表

编　号	数据存储名称	数据存储组成
DB01	库存数据	DS01
DB02	定货数据	DS04

表 2-7　定货系统的处理表

编　号	处理名称	层次名	数据流名称	输入/输出标志
P01	接收库存数据	1.1	DF01	0
			DF03	1
P02	产生定货清单	2	DF04	0
			DF02	1
⋮				

2.3.4 处理的逻辑表达方式

数据流图中的每个“处理”应该有一个“小说明”，用来描述处理的逻辑加工。小说明通常采用结构化自然语言、判定树、判定表等工具来表达处理的逻辑。

1. 结构化自然语言

结构化自然语言介于自然语言和程序设计语言之间。它是在自然语言的基础上加上有限的词汇和语句而形成的语言。

结构化自然语言的词汇表包括数据字典中定义的数据元素、数据结构、数据流等名词，加上自然语言中有限的含义明确的执行性动词以及一些常用的运算符，包括算术、关系和逻辑运算符等。使用的语句仅限于简单的祈使语句、判断语句和循环语句以及由这三种语句组成的复合语句。下面是结构化自然语言的语句表达处理的例子。

GET 库存业务；
EDIT 库存业务；
IF 库存业务有错
  THEN 置库存业务的错误标志
  ELSE 去掉库存业务的错误标志。

2. 判定树

使用判定表、判定树表示判断逻辑比用结构化自然语言更直观、清楚，易于理解。判定树是表达嵌套的多层判断的有效工具。

例如，某工厂的职工超产奖励政策如下：对产品 X，实际生产数量超过计划指标 50 件（含 50 件）以下，每超产 1 件奖励 1 元；超产数量在 51～100 件，超过 50 件部分，每超产 1 件奖励 1.2 元；超产数量在 100 件以上，超过 100 件部分，每超产 1 件奖励 1.5 元。对产品 Y，实际生产数量超过计划指标 25 件以下（含 25 件），每超产 1 件奖励 2 元；超产数量在 25 件以上，每超产 1 件奖励 3 元。奖励政策用判定树表达如下：

$$
\text{奖励政策}\begin{cases}\text{产品 X}\begin{cases}1\leqslant N\leqslant 50 \Rightarrow 1\times N\text{元}\\ 50<N\leqslant 100 \Rightarrow 50+1.2\times(N-50)\text{元}\\ N>100 \Rightarrow 110+1.5\times(N-100)\text{元}\end{cases}\\ \text{产品 Y}\begin{cases}1\leqslant N\leqslant 25 \Rightarrow 2\times N\text{元}\\ N>25 \Rightarrow 50+3\times(N-25)\text{元}\end{cases}\end{cases}
$$

3. 判定表

判定表是表达条件和操作之间相互关系的一种规范的方法。当某个处理依赖于多个逻辑条件时，判定表比判定树更有效。

例如一个用结构化自然语言表达的“检查定购单”处理的逻辑描述如下：

IF 金额超过 500 元且未过期
  THEN 发出批准单和提货单：
IF 金额超过 500 元且已过期

THEN 不发批准单；

IF 金额低于 500 元

THEN 不论是否过期都发批准单和提货单，在过期的情况下还发出通知书。

用判定表描述见表 2-8。

表 2-8 检查定货单的判定表

金 额 状 态	>500 未过期	>500 已过期	≤500 未过期	≤500 已过期
发批准单	×		×	×
发提货单	×		×	×
发通知单				×

判定表和判断树只适合表达判断，不适宜表达循环。

2.3.5 数据分析

软件系统本质上是信息处理系统，在软件系统的整个开发过程中，都应该考虑两个方面的问题—— “数据”及对数据的处理。因此，分析系统的数据要求是系统分析的一个重要任务。系统的数据分析应当包括收集系统所使用的数据，适当地将数据元素组织成合理的数据结构。复杂的数据是由许多基本的数据元素组成的，必须通过分析，详细完整地记录和描述系统的输入数据、输出数据以及在系统中被处理的数据。在细化数据流图的过程中，将已收集的数据元素录入数据字典。数据结构表示数据元素间的逻辑关系。数据流和数据存储中的数据是基于数据的结构的，这并不是数据的最恰当的组织方式，通常用 ER（Entity-Relationship）图表示数据的组织及数据之间的关系。为减少数据冗余，简化修改数据的过程，还应该对数据进行规范化。对数据的规范化和 ER 图将在后面相关章节介绍。

2.4 结构化设计

软件分析阶段得到了软件的需求说明书，它明确地描述了用户要求软件系统“做什么”的问题，即定义了系统的主要逻辑功能、数据以及数据间的联系，现在是决定“怎么做”的时候了，即建立一个符合用户需求的软件系统。软件开发进入软件设计阶段。

设计阶段通常分为两步。一是系统的总体设计或一般设计，它的任务是确定软件结构。二是系统的详细设计，即进行各模块内部的具体设计。软件设计方法有多种，本节介绍结构化设计方法。

2.4.1 结构化设计方法概述

结构化设计（Structured Design,SD ）方法用于设计软件结构，它的目标是根据系统分析资料确定软件应由哪些子系统或模块组成，它们应采用什么方式联结，接口如何，才能构成一个好的软件结构，如何用恰当的方法把设计结果表达出来。同时还要考虑数据库的逻辑设计。

需要全本请访问 www.ertongbook.com