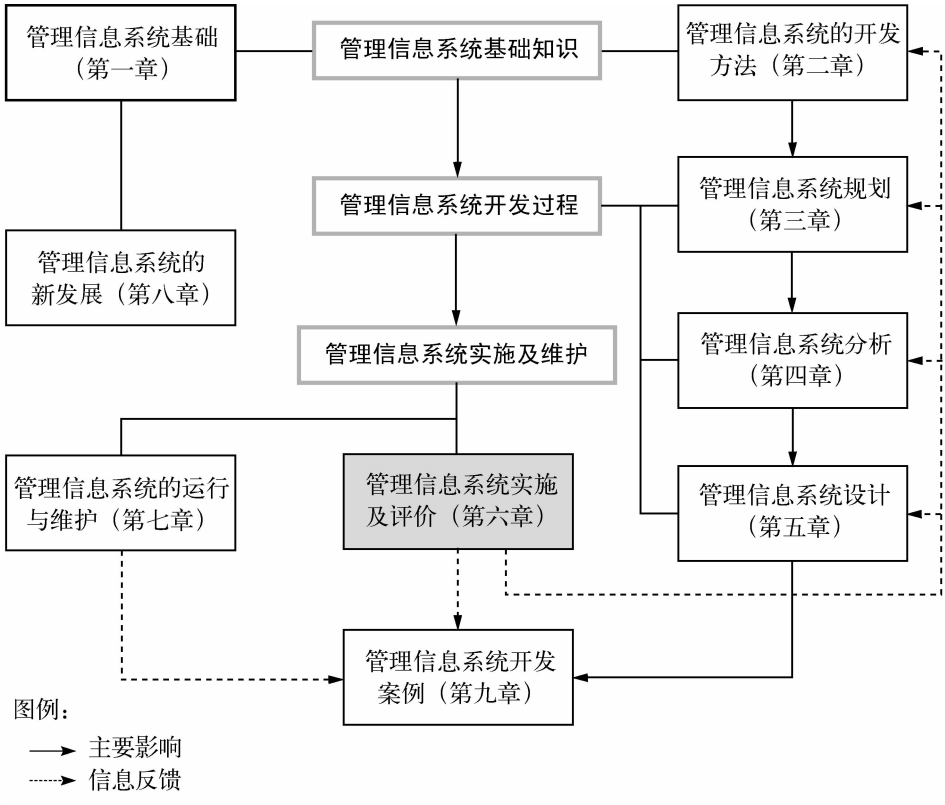
第六章　管理信息系统实施及评价



管理信息系统结构模型

学习目标

（一）知识目标

- 了解系统实施在系统开发中的地位和作用；
- 了解程序设计的基本要求；
- 了解系统测试的目的和意义、基本测试方法和测试流程；
- 了解系统转换的各种方式和特点；
- 了解系统评价的主要内容。

（二）技能目标

- 掌握程序设计的基本方法及常用编程工具的选择依据；
- 能够根据程序设计的步骤，选择恰当的编程工具进行简单程序的编写；
- 掌握系统测试的方法。

当系统分析与系统设计的工作完成以后，开发人员的工作重点就从分析、设计和创造性思考的阶段转入实践阶段。在此期间，将投入大量的人力、物力，并会使用较长的时间进行物理系统的实施、程序设计、系统测试、系统转换、系统评价等一系列工作，这个过程称为系统实施。MIS 的规模越大，实施阶段的任务就越复杂，为此，在系统正式实施之前，就要制定出周密的计划。系统实施完成之后，还要定期对其运行状况进行检查与评价，目的是通过对系统运行过程和绩效的审查来检验系统是否达到了预期目标、是否充分利用了系统内各种资源、系统管理工作是否完善，并提出今后的改进和扩展方向。

第一节　系统实施概述

系统实施是新系统付诸实现的实践阶段，也是取得用户对系统信任的关键阶段。因此，必须严格按照新系统设计报告的要求，进行组织工作。

一、系统实施的地位和作用

系统实施是继系统规划、系统分析、系统设计之后的又一重要阶段，它将按照系统设计选定的方案具体实现系统。系统开发工作主要集中在逻辑、功能和技术设计上，工作成果是以各种系统分析与设计文档来体现的。系统实施阶段要继承此前各阶段的成果，将技术设计转化为物理实施，因此，系统实施的成果是系统分析和设计阶段的结晶。由于它是以系统分析和设计工作为基础的，所以必须按照系统设计的文档进行。只有在系统分析和设计工作完成以后，才能开始系统实施工作，切忌在系统开发工作中提前开展这部分工作。因为某些开发者，特别是程序编制人员，往往把开发重点放在编程上，在没有完全了解系统的需求和总体设计的条件下就匆匆开始编程，这必然会使系统开发工作受阻。另外，系统实施是把系统设计转化为可实际运行的物理系统的必然步骤，再好的系统设计，如果实施工作不到位，同样很难起到令人满意的效果。系统实施作为整个系统的最后物理实现阶段，对于系统的质量、可靠性和可维护性等都有着十分重要的意义。

二、系统实施的主要任务

（一）系统实施的前期准备

系统实施是开发信息系统的最后一个阶段，是实现系统设计阶段提出的物理模型的阶段。要使系统实施工作得以顺利进行，需做好以下几个方面的工作：

1. 成立系统实施领导小组

系统实施阶段所占用的时间最长、耗费资源最多，因此，必须加强系统实施的组织与领导工作。应根据系统实施的目标，将不同部门的人员组织起来，开展有条不紊的工作，安排各项任务并按其不同特点进行协调与配合。系统的实施领导工作应由系统开发领导小组承担，也可由专门成立的实施领导小组承担，但组长必须由用户单位的最高层领导担任。

系统实施领导小组必须做好新系统实施计划的编制工作，布置和协调各方面的关系，检查工作的进度和质量，根据情况进行必要的调整和修改，处理和解决在实施过程中发生和发现的一切重大问题。此外，在系统实施计划的基础上，必须制定各专业组计划以保证实施计划的顺利完成。领导小组还要验收各部分工作，组织新系统的调试，负责现行系统向新系统转换的一切组织工作和管理工作。

2. 人员的培训

信息系统是一个人机系统，需要很多人参加工作，这些人将承担信息系统中人工过程的处理和计算机操作工作。通常，这部分人精通原来的业务但缺乏计算机知识，为了保证计算机系统的调试和新系统的运行能够顺利进行，必须提前培训他们，使他们熟练掌握计算机操作，了解系统分析和设计的基本概念、系统的概貌以及为什么要进行改革，明确他们今后将承担哪些工作等。

3. 数据准备

数据的收集、整理、录入是一项繁琐、劳动量大的工作，如果没有一定基础数据的准备，系统测试就不能很好地进行。一般来说，在确定数据库物理模型之后，就应进行数据的整理和录入工作。

（二）系统实施的主要任务

系统实施的主要任务有：购置和安装计算机网络系统、进行程序设计、进行系统的调试与测试、负责新旧系统的转换等。

1. 购置和安装计算机网络系统

随着信息产业及计算机技术的发展，市场上不同厂家生产的不同型号的计算机产品层出不穷，这一方面给一般用户及信息系统提供了更大的选择空间，另一方面也给系统的实施带来了一定的复杂性。人们必须从这些种类繁多的计算机产品当中挑选出最适合应用需要的产品。一般来说，购置计算机产品可从如下几个方面考虑：第一，计算机系统是否具有合理的性能价格比；第二，系统是不是具有良好的可扩充性；第三，能否得到来自供应商的售后服务和技术支持。

网络系统的实施主要是通信设备的安装、电缆线的铺设及网络性能的调试。常用的组网设备主要有双绞线、同轴电缆、光纤电缆以及微波和卫星通信。计算机作为精密电子设备，对

周围环境、机房温度、空气湿度等都非常敏感，因此，要选择那些灰尘少、温度适中、空气较为流通的场所放置。通常，服务器要专门放置在一个房间，并且安装双层玻璃门窗，进入内部的工作人员要求必须穿戴鞋套。另外，连接各硬件设备的电缆线要安放在防止静电感应的耐压的活动地板下面。为了防止由于突然停电造成系统出现故障，还应配置足够功率的 UPS 电源。

2. 程序设计

程序设计也称软件开发，进行程序设计的目的是实现系统分析和设计中提出的管理模式和业务应用。在进行程序设计之前，设计人员要学习所需的系统软件，包括操作系统、数据库系统和开发工具。必要时，需要对程序设计人员进行专门的系统软件培训。

3. 系统的调试与测试

在进行计算机程序设计之后，需要进行系统的调试与测试。实际上，在编写计算机程序时，一直在进行调试和修改程序中的错误。在完成这种形式的调试之后，还必须进行专门的系统测试。系统测试是保证系统质量可靠性的关键，也是对需求分析、系统设计和编码的最终评审。要运用一定的测试技术和方法，通过单元测试、集成测试、确认测试和系统测试几个步骤，发现系统可能存在的问题。

4. 新旧系统的转换

新旧系统的转换也称系统转换与运行，是指以新开发的系统取代老系统，并使之投入使用的过程。新旧系统转换是管理信息系统实施的最后一项任务，它包括进行基本数据的准备、数据的编码、系统的参数设置、初始数据的录入等多项工作。在系统正式交付使用之前，必须进行一段时间的试运行，以进一步发现及更正系统存在的问题。在系统转换和交付使用的过程中，每项工作都有很多人员参加，而且会涉及多个业务部门，因此，该阶段的组织管理工作非常重要，要做好系统转换计划，控制工作的进度，检查工作的质量，及时地做好各方面的协调，保证系统的成功转换和交付使用。

第二节　程序设计

一、程序设计概述

计算机进行处理工作完全依赖程序，而程序是用计算机语言编写的，它是解决某类问题的一系列语句或指令。目前，信息系统的部分处理工作可由现成的软件完成，但大部分程序还是需要人工编写的。程序的编制应符合软件工程的思想，软件工程是 20 世纪 60 年代末逐步发展起来的一门学科，由于计算机应用的迅速发展，软件系统的开发和能力远远跟不上社会需要，产生了所谓的“软件危机”，软件工程正是为了克服“软件危机”逐步发展起来的。应用软件的编程工作量极大，而且要经常维护、修改，如果编写程序不遵守正确的规律，就会给系统的开发、维护带来不可逾越的障碍。软件工程的思想即利用工程化的方法进行软件开发，通过建立软件工程环境来提高软件开发效率。按照软件工程原理开发程序能使整个程序设计过程任务明确、具体，而且可以达到规范化、系统化和工程化，方便今后程序的测

试、维护、管理等工作。

程序编制的依据是系统分析与设计阶段产生的业务流程图、程序代码、判断树、判断表、数据流程图等，程序员根据以上资料统一选择恰当的程序设计语言进行程序设计。程序设计的任务是为新系统编写程序，即把详细设计的结果转换成某种计算机编程语言写成的程序。该阶段相当于机械工程中图纸设计完成的“制造”阶段，程序设计的好坏直接关系到能否有效地利用电子计算机来圆满地达到预期目的。

高质量的程序必须符合以下基本要求：

(1) 程序的功能必须按照规定的要求，能够满足预期的需要。

(2) 程序清晰、明了、便于阅读和理解。

(3) 程序的结构严谨、简捷，算法和语句选用合理，执行速度快，节省时间。

(4) 程序和数据的存储、调用安排得当，节省存储空间。

(5) 程序的适应性强。程序交付使用后，若应用范围或外界环境有了变化时，调整和修改程序应简便易行。

二、程序设计步骤

一个信息系统包含多种不同的处理，采用结构化设计时，则将不同的处理细分为多个大小不等的程序。通常，一个中等规模的信息系统就要编制几百个程序，合理分配和安排这些程序的编制工作，使其有条不紊地按步骤进行，这是极为重要的。

1. 熟悉计算机性能

在程序设计前，所有参与开发的程序员都要熟悉系统的开发环境，包括计算机的性能、操作系统、程序设计语言与数据库管理系统。

2. 合理分配编程任务

在把整体的编程任务分解之后，应根据任务的轻重缓急及程序员的人数和能力进行合理的分工。对编程的安排应考虑程序的特点，一般来说，文件或数据库建立的工作量极大，应该尽量安排在前面；文件的建立程序和更新程序、查询程序关系密切，则不宜将它们分给几个程序员编写。有时，一组程序的工作量太大时，可安排几人合编一组或一个程序。在进行任务分配时要估算其工作量，要考虑程序的复杂程度、所使用的语言、输入输出界面的设计、程序员的经验和水平等，这是一个很复杂的问题，很多人进行过这方面的研究，提出过不同的方法，但至今仍没有一个统一的标准。

3. 编写程序

在这个阶段，程序员要充分理解系统的设计要求，仔细阅读系统设计说明书，明确系统设计所提出的任务、功能和目标，了解自己所编程序在系统中所处的位置及与之相关的环境条件，然后完成编程并在计算机上实现。

4. 程序测试

程序编制完成以后，要对程序的正确性做出评价，这就需要对程序进行测试。测试的目的是发现错误并加以改正。程序中常见的错误有语法错误、逻辑错误和输入/输出格式错误等。有关统计表明，程序测试所占用的时间和经费与开发系统的规模成正比，因此，组织测试数据、选择测试方法应引起系统开发者足够的重视。程序测试时应根据程序错误的特点

选择有代表性的测试方法进行测试。

三、程序设计方法

目前，程序设计的方法大多是按照结构化方法、原型方法、面向对象的方法进行。编程的目的是实现开发者在系统分析和系统设计中提出的管理方法和处理构想，因此，在编程过程中，应尽量借用已有的程序模块和各种开发工具，尽快更好地实现系统，而不要在具体的编程和调试工作中花费过多的精力和时间。

（一）结构化程序设计方法

结构化程序设计方法是由 E. Dijkstra 等人于 1972 年提出的，用于在详细设计和程序设计阶段指导人们用良好的思想方法开发出正确又易于理解的程序。结构化程序设计至今还没有一个统一的定义，一般认为，结构化程序设计是一种设计程序的技术，它采用自顶向下逐步求精的设计方法和单入口单出口的控制技术，把系统划分为大小适当、功能明确、具有一定独立性并容易实现的模块，从而把一个复杂的系统的设计转变为多个简单模块的设计。结构化程序中一般没有 GOTO 语句，易于阅读，而且提高了系统的可修改性和可维护性。对于一个分析和设计都非常规范、功能单一、规模又小的模块来说，结构和程序设计方法的作用很有限，但若遇到某些开发过程不规范，模块划分不细，或者是因特殊业务处理的需要模块程序量较大时，结构化程序设计方法则绝对是一种比较有效的方法。一般来说，结构化程序设计方法采用以下三种基本逻辑结构来编写程序：顺序结构、选择结构和循环结构。这三种基本结构如图 6-1 所示。

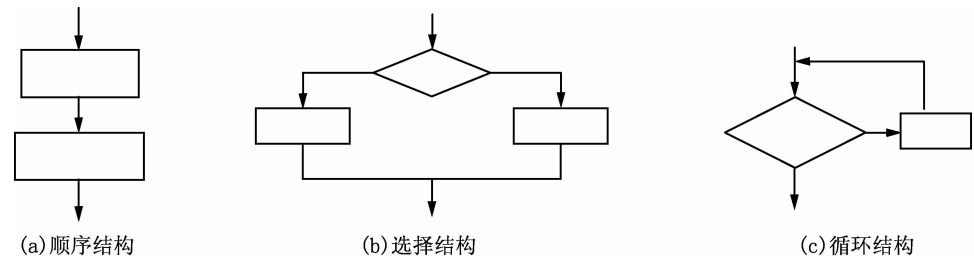


图 6-1　结构化程序设计方法的三种基本逻辑结构

1. 顺序结构

顺序结构是一种线性有序的结构，由一系列依次执行的语句或模块构成。顺序结构的程序设计是最简单的，只要按照解决问题的顺序写出相应的语句就行，它的执行顺序是自上而下，依次执行，一般不能颠倒语句顺序。

例如，“a=3，b=5”，现要交换 a 和 b 的值，这个问题就好像交换两个杯子里的水，要用到第三个杯子，假如第三个杯子是 c，那么正确的程序为“c=a；a=b；b=c；”执行结果是“a=5；b=c=3”。如果改变其顺序，写成“a=b；c=a；b=c；”，则执行结果就变成“a=b=c=5”，不能达到预期的目的。

顺序结构可以独立使用，构成一个简单的完整程序，常见的输入、计算、输出三部曲的程序就是顺序结构。例如，计算圆的面积，其程序的语句顺序就是输入圆的半径 r，计算 $s=3.14159*r*r$，输出圆的面积 s。不过大多数情况下顺序结构都是作为程序的一部分，

与其他结构一起构成一个复杂的程序。

2. 选择结构

选择结构是根据条件成立与否选择程序执行路径的结构，一般有以下两种情况（以 Visual Basic 编程语言为例）：

（1）条件语句。语句结构如下：

```
IF＜条件＞
    ＜语句序列 1＞
[ELSE]
    ＜语句序列 2＞
ENDIF
```

执行该语句时，如果＜条件＞的值为真（T），则执行＜语句序列 1＞，然后跳过＜语句序列 2＞，执行 ENDIF 后面的语句；否则，如果有 ELSE 子句时，则跳过＜语句序列 1＞，直接执行 ELSE 后面的＜语句序列 2＞，然后执行 ENDIF 后面的语句，如果没有 ELSE 子句，则执行 ENDIF 后面的语句。

（2）分支语句。语句结构如下：

```
DO CASE
CASE＜条件 1＞
    ＜语句序列 1＞
CASE＜条件 2＞
    ＜语句序列 2＞
……
CASE＜条件 n＞
    ＜语句序列 n＞
[ORTHERWISE ]
    ＜语句序列 n＋1＞
ENDCASE
```

分支语句是一种扩展的分支结构，DO CASE 语句执行时，依次判断 CASE 子句中的条件（条件表达式），当发现某个 CASE 后面的条件成立时，就执行该 CASE 和下一个 CASE 之间的命令序列，然后转到 ENDCASE 语句后面的第一条语句。如果所有的 CASE＜条件＞都不成立，且有 ORTHERWISE 子句，则执行 ORTHERWISE 与 ENDCASE 之间的命令序列，然后转到 ENDCASE 语句后面的第一条语句；若没有 ORTHERWISE 子句就直接转到 ENDCASE 语句后面的第一条语句。DO CASE…ENDCASE 语句必须成对出现。DO CASE 中的 CASE 子句个数不限。

3. 循环结构

循环结构可以减少源程序重复书写的工作量，用来描述重复执行某段算法的问题，这是程序设计中最能发挥计算机特长的程序结构。其一般格式如下：

```
DO WHILE ＜条件＞
    ＜命令 1＞
```

```
[Loop]
    <命令 2>
[EXIT]
    <命令 3>
ENDDO
```

综上所述，用结构化程序设计方法的结构清晰明了，有利于编写出结构良好的程序，因此，开发人员必须用结构化程序设计的思想来指导程序设计的工作。结构化程序设计的基本思想是按自顶向下逐步求精的方式，由三种标准控制结构反复嵌套来构造一个程序。按照这种思想，可以对一个执行过程模糊不清的模块，以顺序、选择、循环的形式加以分解，最后使整个模块都清晰起来，从而确定全部细节。由于大多高级语言都支持结构化程序设计方法，其语法上都含有表示三种基本结构的语句，所以用结构化程序设计方法设计的模块结构到程序的实现是直接转换的，只需用相应的语句结构代替标准的控制结构即可。因此结构化程序设计方法减轻了程序设计的工作量。

（二）面向对象程序设计方法

面向对象程序设计方法既吸取了结构化程序设计方法的优点，又考虑了现实世界与面向对象之间的映射关系，它所追求的目标是将现实世界的问题尽可能简单化。面向对象程序设计方法将数据及对数据的操作放在一起，作为一个相互依存、不可分割的整体来处理，它采用了数据抽象和信息隐藏技术。它将对象及对对象的操作抽象成一种新的数据类型——类，并且考虑不同对象之间的联系和对象所在类的重要性。面向对象程序设计方法优于传统的结构化程序设计方法，其优越性表现在，它有希望解决软件工程的两个主要的问题——软件复杂性控制和软件生产率的提高，此外，它还符合人类的思维习惯，能够自然地表现现实世界的实体和问题，它对软件开发过程具有重要的意义。

四、衡量程序设计工作质量的指标

管理信息系统的所有功能及设计意图都要通过编程工作来实现，因此，程序的质量会影响到整个管理信息系统的质量、运行与维护。为了高质量地完成编程工作，就需要对编程工作的质量进行衡量。从目前的技术发展来看，衡量编程工作质量的指标大致可分为以下几个方面：

1. 可靠性

管理信息系统的可靠性(reliability)十分重要，在任何时候都是程序设计的基本要求，而且可靠性也是衡量系统质量的重要指标。可靠性指标可以分为安全可靠性和运行可靠性两个方面的内容。程序或系统的安全可靠性反映在多个方面，如数据存取的安全可靠性、通信的安全可靠性、操作权限的安全可靠性等。系统运行的可靠性只能通过高质量的程序设计、周密的程序测试、严格的系统测试来把关。

2. 规范性

规范性(standardability)即系统的划分、书写格式、变量的命名等都有统一的规范要求，这便于程序今后的阅读、修改和维护。

3. 可读性

可读性(readability)即程序结构清晰，没有太多繁杂的技巧，能够使他人容易读懂。对

于大型程序来说，不仅要求它逻辑正确，能执行，而且应当层次清楚，简洁明了，便于阅读。一个逻辑上完全正确但杂乱无章，无法供人阅读、分析、测试、排错、修改与使用的程序是没什么价值的。程序维护的工作量很大，程序维护人员常要维护他人编写的程序，如果一段程序不易阅读，那么就会给程序检查与维护带来极大的困难。要使所写的程序易于阅读，就必须有一个结构清晰的程序框架。实际上，结构清晰是保证程序正确、提高程序可读性的基础。在国外，常常在程序中插入大量的解释性语句，以对程序中的变量、功能、特殊处理细节等进行解释，为今后他人读这段程序提供方便。

4. 可维护性

可维护性(maintainability)即程序各部分相互独立，没有调用子程序以外的其他数据关联，在维护过程中，将牵一发而动全身的连锁反应基本消除或降到最低限度。MIS 的寿命一般是 3～10 年，因此，程序维护的工作量相当大，一个不易维护的程序，用不了多久就会因为不能满足应用需要而被淘汰，所以说可维护性是对程序设计的一项重要要求。

五、程序设计的要求

既然程序的可读性和可维护性对软件的质量有如此重要的影响，那么程序员在程序设计过程中就应当对此充分重视。为了提高程序的可读性和可维护性等性能，在程序设计方面应注意以下几点要求：

1. 程序书写格式要有规律

恰当的书写格式将有助于阅读，在结构化程序设计中一般采用所谓“缩排法”来写程序，即把同一层次的语句向左端对齐，而下一层的语句则向右边缩进若干格书写，它能体现程序逻辑结构的深度。此外，在程序段与段之间安排空白行，也有助于阅读。

2. 合理命名变量

理解程序中每个变量的含义是理解程序的关键，因此，变量的名字应该适当选取，使其直观，易于理解和记忆。例如，采用有实际意义的变量名，不用过于相似的变量名，同一变量名不要具有多种意义，变量的名字应该包括类型信息等。此外，在编程前最好能对变量名的选取约定统一标准，以便于以后的阅读理解。

3. 增加程序注释

加在程序中的注释是程序员与日后的程序读者之间交互的重要手段，正确的注释能够帮助读者理解程序，可为后续阶段进行测试和维护提供明确的指导，因此，注释绝不是可有可无的。大多数程序设计语言允许使用自然语言写注释，这给程序阅读带来很大的方便。一些正规的程序文本中，注释行的数量可能占到整个源程序的 1/3 到 1/2。在程序中适当地加上注释后，可以使程序成为一篇“自我解释”的文章，读程序时就不必翻阅其他说明材料了。注释原则上可以出现在程序中的任何位置，但是如果使注释和程序的结构配合起来则效果更好。

注释分为序言性注释和功能性注释。序言性注释通常置于每个程序模块的开头部分，它应当给出程序的整体说明，对于理解程序本身具有引导作用。有些软件开发部门对序言性注释作了明确而严格的规定，要求程序编制者逐项列出，主要包括以下几方面内容：

(1) 程序标题。

(2) 有关本模块功能和目的的说明。

(3) 主要算法。

(4) 接口说明,包括调用形式,参数描述和子程序清单。

(5) 有关数据描述:重要的变量及其用途,约束或限制条件以及其他有关信息。

(6) 模块位置:在哪一个源文件中,或隶属于哪一个软件包。

功能性注释嵌在源程序体中,是用以描述其后的语句或程序段在做什么工作的,或是执行了下面的语句会怎样。

书写注释时应注意以下几点:

(1) 注释应和程序一致,修改程序时应同时修改注释,否则会起反作用,使人更难明白。

(2) 注释应提供一些程序本身难以表达的信息。

(3) 为了方便用户今后维护,注释应尽量多用汉字。

4. 输入输出要让人一目了然

在编写输入和输出程序时,要做到以下几点:

(1) 步骤和格式尽量简单,提示信息要明确,易于理解。

(2) 输入一批数据时尽量少用计数器来控制数据的输入进度,尽量使用文件结束标志。

(3) 应对输入数据的合法性、有效性进行检查,报告必要的输入信息及错误信息。

(4) 交互式输入时,应提供明确可用的输入信息。

(5) 当程序设计语言有严格的格式要求时,应保持输入格式的一致性。

5. 效率

程序的效率是指程序能否有效地利用计算机资源,对效率的追求应注意以下几个方面的问题:

(1) 效率是一个性能要求,在需求分析阶段就要对效率目标有一个明确的要求。

(2) 追求效率应该建立在不损害程序可读性或可靠性基础之上,在程序可靠和正确的基础上追求效率。

(3) 选择良好的设计方法才是提高程序效率的根本途径。设计良好的数据结构与算法,都是提高程序效率的重要方法。编程时对程序语句作调整是不能从根本上提高程序效率的。

此外,程序的效率与可维护性通常是矛盾的,在实际编程工作中,人们往往宁可牺牲一定的时间和空间,也要尽量提高系统的可维护性,片面地追求程序的运行效率反而不利于程序设计质量的全面提高。

六、编程工具

(一) 常用编程工具介绍

随着计算机技术的不断发展,各种编程工具也不断发展,这一领域是整个计算机或信息产业中发展最快的领域之一。目前,编程工具不仅在数量和功能上突飞猛进,而且在其内涵的拓展上也日新月异,为用户开发系统提供了越来越多、越来越方便的实用手段。在当今的信息系统开发中,了解和选用恰当的编程工具是提高系统开发质量和效率的保证。在这里,将针对流行的语言介绍几款较为成熟完善的编程工具。

1. Visual Basic

Visual Basic(VB)是以 Basic 语言作为其基本语言的一种可视化编程工具,它曾一度是中国最为流行的编程工具,到现在还占据着非常重要的地位。VB 作为一种较早出现的开发程序,有易于学习、开发效率较高、具有完善的帮助系统等优点。但由于 VB 不具备跨平台这个特性,从而也决定了 VB 在未来的软件开发中将会逐渐地退出历史舞台。它对组件技术的支持是基于 COM 和 ActiveX 的,在组件技术不断完善发展的今天,显示出了它的落后性。同时,VB 在进行系统底层的开发时也相对复杂,调用 API 函数需声明,调用不方便,不能进行 DDK 编程,不能嵌套汇编,而且面向对象的特性差,网络功能和数据库功能也没有非常出色的表现。综上所述,VB 作为一款可视化的编程工具由于其本身的局限性,导致了它在未来软件开发中将会逐步被其他软件开发工具所替代。

建议:对于编程入门人员,可以先借助 VB 这个可视化环境大致了解可视化编程的特点,并且可开发与系统无关的综合应用程序。

2. PowerBuilder

PowerBuilder(PB)是开发 MIS 系统和各类跨平台数据库的首选工具,它使用简单,容易掌握,在代码执行效率上也有相当出色的表现。PB 是一种真正的 4GL 语言(第四代语言),可随意直接嵌套 SQL 语句,支持语句级游标、存储过程和数据库函数,类似于 SQLJ 的规范,在数据访问中具有无可比拟的灵活性。但是它在系统底层开发中具有同 VB 一样的缺陷,调用 API 函数需声明,调用不方便,不能进行 DDK 编程,不能嵌套汇编。它在网络开发中提供了较多动态生成 Web 页面的用户对象和服务以及系统对象,非常适合编写服务端动态 Web 应用,有利于商业逻辑的封装,但是对于网络通信的支持不足,并且静态页面定制支持有限,因此使得 PB 在网络方面的应用也不够广泛。同时,它的面向对象特性也不是太好。

建议:从事信息管理系统的开发或各类数据库的跨平台开发都可以选用此工具。

3. C++ Builder/Delphi

它们都是基于 VCL 库的可视化开发工具,它们在组件技术的支持、数据库支持、系统底层开发支持、网络开发支持、面向对象特性等各方面都是相当不错的,并且学习使用较为容易,充分体现了所见即所得的可视化开发方法,开发效率高。由于两者都是 Borland 公司的产品,自然继承了该公司一贯的优良传统,即代码执行效率高。但是,它们并不是毫无缺陷的,它们的最大不足之处就是其帮助系统在众多的编程工具中是属于比较差的。C++ Builder的 VCL 库基于 Object Pascal(面向对象 Pascal),因此使得 C++ Builder 在程序的调试执行上都明显落后于其他编程工具;而 Delphi 则有语言不够广泛、开发系统软件功能不足两个比较大的缺点。

建议:C++ Builder/Delphi 在功能上具有非常相似的特点,都可以用来开发数据库和网络多媒体,但是 C++ Builder 的语法较为灵活,使用也较为广泛,而 Delphi 在灵活性和功能性上都不如 C++ Builder,使用人数也少于 C++ Builder。

4. Visual C++

Visual C++(VC)是基于 MFC 库的可视化的开发工具,从总体上说,它是一种功能强大但使用不便的工具。它在网络开发和多媒体开发方面都具有不俗的表现,帮助系统也做得

非常不错(Microsoft 在细节方面的处理往往都让人觉得亲切)。缺点是虽然是使用C++作为基本语言,但是它在面向对象特性上却不够好,主要是为了兼容 C 的程序,结果顾此失彼;在组件支持上也不太好,虽然说除了支持 COM 和 ActiveX 外还支持 CORBA,但是没有任何 IDE 支持,所有 C 编译器的功能都需要 CORBA 中间件支持。此外,其最大的问题是开发效率不高。

建议:如果要使用 Visual C++,一定要对它的 MFC 库非常熟悉,不然是写不好程序的。虽然 Visual C++ 的入门比较难,但是掌握了它之后就可以在网络、系统底层、多媒体开发等领域自由驰骋了。

5. Java 编程工具

目前比较常用的是 Borland 推出的 JBuilder 和 IBM 推出的 Visual Age for Java,两种工具都有一定数量的使用人群。JBuilder 继承了 C++ Builder/Delphi 的特点,在可视化上做得非常不错。Java 语言本身的特点使得其在网络开发中具有突出的优势,而且面向对象特性高,支持的组件技术也非常多,具备跨平台的特性也使得它在现在和未来的开发中占据越来越重要的地位。

建议:除了在开发系统软件、大规模的图像处理时外,都可以使用 Java。

(二) 编程工具的选择

在程序设计之前,从系统开发的角度考虑选用哪种语言来编程是很重要的,一种合适的程序设计语言能减少根据设计完成编程时的困难,可以减少所需要的程序调试量,并且可以得出更容易阅读和维护的程序。选择合适的开发工具首先应该考虑所选择的开发工具所适用的领域,除此之外还要从以下几个方面考虑:

1. 系统用户的要求

如果所开发的系统由用户负责维护,用户通常要求用他们熟悉的语言书写程序。

2. 开发人员的熟练程度

虽然对于有经验的程序员来说,学习一种新语言并不困难,但要完全掌握一种新语言并用它编出高质量的程序,却需要经过一段时间的实践。因此,如果可能的话,应该尽量选择一种已经为程序员所熟悉的语言。

3. 语言可提供的交互功能

选用的语言必须能够提供开放、美观的人机交互程序的功能,如色彩、音响、窗口等,这对用户来说是非常重要的。

4. 软件可移植性要求

如果开发出的系统软件将在不同的计算机上运行,或打算在某个部门推广使用,那么应该选择一种通用性强的语言。

5. 软件的数据库管理能力

大部分信息系统都需要进行大量的数据库操作,所以选择的开发工具应该具有强大的数据库操作能力,通常可选择的数据库开发工具有 Foxpro、SQL、Oracle、PowerBuilder 等。

第三节 系统测试

虽然软件的质量不是测试得来的，但规范化的测试却能帮助人们发现编程中的缺陷并且予以纠正。其实，测试不只限于实施阶段，在交付使用后仍会有测试。另外，修改和测试是不可分的，有程序修改，就必然要进行测试，软件修改和版本升级将贯穿系统的整个生命周期。

一、系统测试概述

（一）系统测试的意义

系统测试是管理信息系统开发周期中一个十分重要而漫长的阶段，其重要性体现在它是保证系统质量与可靠性的最后关口，是对整个系统开发过程，包括系统分析、系统设计和系统实现的最终审查。在管理信息系统的开发过程中，面对着错综复杂的开发流程，尽管在系统开发周期的各个阶段均采取了严格的技术审查，希望尽早发现问题，但人的主观认识不可能完全符合客观现实，开发人员之间的思想交流也不可能十分充分，因此，系统开发周期的各个阶段还是不可避免地会出现差错。开发人员应尽可能早地发现并纠正这些错误，若等到系统投入运行后再回头来改正错误，则将在人力、物力上造成很大的浪费，有时甚至导致整个系统的瘫痪。例如，1963 年在美国用于控制火箭飞行的 FORTRAN 程序中出现了一个错误，把一个循环语句“DO 5 I=1,3”误写为“DO 5 I=1.3”，这个错误在系统测试中心又没被发现，就这一点之差，致使飞往火星的火箭爆炸，造成了 1 000 万美圆的损失。由此可见，对系统进行测试是必不可少的，它是保证系统质量的关键步骤。统计资料表明，对于一些较大规模的系统来说，系统测试的工作量往往占整个系统开发总工作量的 40％以上。

国内很多开发人员，甚至项目经理都只注重编程和编程人员，而忽视测试和测试人员，测试过程往往被省略掉，使项目带着许多缺陷就被推出，这是影响软件质量的一个重要原因。做好测试除了需要标准和工程的方法以外，更重要的是转变观念，走出误区。不少项目经理认为，只要雇用软件工程师就行，其他的人都不必要，或让软件工程师占团队很高的比例。他们也许认为开发人员越多，写出来的程序也越多，这实际上是错误的观点。项目的目的是为完成软件，而不是完成更多代码，有些工作是不宜交给软件工程师做的。要想真正保证软件项目如期完成，不仅取决于开发人员，还取决于测试人员。

（二）系统测试的目的

人们往往会认为测试的目的是说明软件没有问题，因此，编程完毕后，只找几个数据使程序能够顺利完成就结束了测试工作。从软件工程的角度看，这不仅不正确，而且是十分有害的，它会使人们自觉不自觉地寻找容易使程序通过的测试数据，回避那些易于暴露软件错误的测试数据，从而致使隐藏的错误不被发现。

实际上，系统测试是以找错误为目的，而不是要证明程序无错，因此，需要精心选取那些易于发生错误的数据进行测试，以十分挑剔的态度，证明程序有错，在测试时应想方设法使程序的各个部分都投入运行，力图找出所有错误。即使这样，测试通过也不能证明系统绝对无误，只不过说明各模块、各子系统的功能和运行情况正常，相互之间连接无误。因此，不要

认为程序设计完成后就万事大吉了，其实后面大量的系统测试工作才刚刚开始。

在系统测试中发现的错误可能是各种各样的，按其范围和性质可划分为如下几类：

（1）系统错误，指与外部接口的连接错误、参数调用错误、子程序调用错误、输入输出地址错误以及资源管理错误等。

（2）过程错误，主要指算术运算错误、初始过程错误、逻辑错误等。

（3）功能错误，是指由于功能规格说明书不够完整或叙述不够确切，致使在编码时对功能有误解而产生的错误。

（4）编码错误，语法错误、变量名错误、局部变量与全局变量混淆、程序逻辑错误和编码书写错误等都属于编码错误。

（5）数据错误，指数据结构、内容、属性错误，动态数据与静态数据混淆，参数与控制数据混淆等。

（三）系统测试的原则

基于以上对系统测试的理解，在进行系统测试时应该遵循以下基本原则：

（1）测试工作应避免由原开发软件的个人和小组来承担。测试的目的就是挑剔地找错误，从心理学上讲，软件开发人员对自己的成果有所偏爱，总会认为自己的程序没有错误或错误不大，因而有一种不愿否定自己成果的心理；另外，如果开发人员对软件的功能有理解错误，由本人去测试，也很难找出错误。

（2）在设计测试方案时，测试用例不仅要包括输入数据，而且要包括预期的输出结果。也就是说，在执行程序之前应该对期望的输出有很明确的描述，测试后可将程序的输出同预期结果仔细对照检查。若不事先确定预期的输出，则可能把似乎是正确而实际是错误的结果当成是正确结果。

（3）测试数据既要能够检查在合法操作下程序的运行情况，又要能够检查在非法操作下程序的运行情况。在编写程序的时候，由于思维定势的影响，往往都会只考虑程序的输入按照自己的期望进行，都会无意识地假设用户的操作是合法的，从而忽略了用户的非法操作，忽略了当程序的输入不是按照自己的期望进行，发生无效输入的时候程序的运行情况。在程序的实际运行中，许多错误恰恰是由用户的非正常操作或接口数据异常造成的，如用户按错键、输错数、键入非法命令等，如果软件不能做出适当的反应而失控，就不能说明软件是可靠的。

（4）除了检查程序是否做了它应该做的工作外，还应检查程序是否做了它不该做的事情。

（5）保留所有的测试用例，作为软件文档的组成部分。应该长期保留所有的测试用例，并且将其作为管理信息系统软件的组成部分之一。在管理信息系统的测试中，无论测试用例是否发现了软件中的错误，毕竟它是花费了大量精力设计出来的，如果轻易地就将用过的例子丢弃了，以后一旦程序纠错、改进或扩充后，需要再测试有关的部分时就需要重复很多工作，而且人们往往不愿重新认真地设计测试用例，因而下次测试时很少会像初次测试时那样全面、严格。如果将所有测试用例作为系统的一部分保存下来，就可以避免这种情况的发生。

（6）应意识到软件中仍存在错误的概率和已发现的错误个数是成正比的。有时软件经测试发现了许多错误后，测试者认为错误已找得差不多了，因而不再继续测试了。但经验和统计结果表明，发现的错误越多，程序中潜在的错误可能也越多。

（四）系统测试流程图

系统测试流程图如图 6-2 所示。

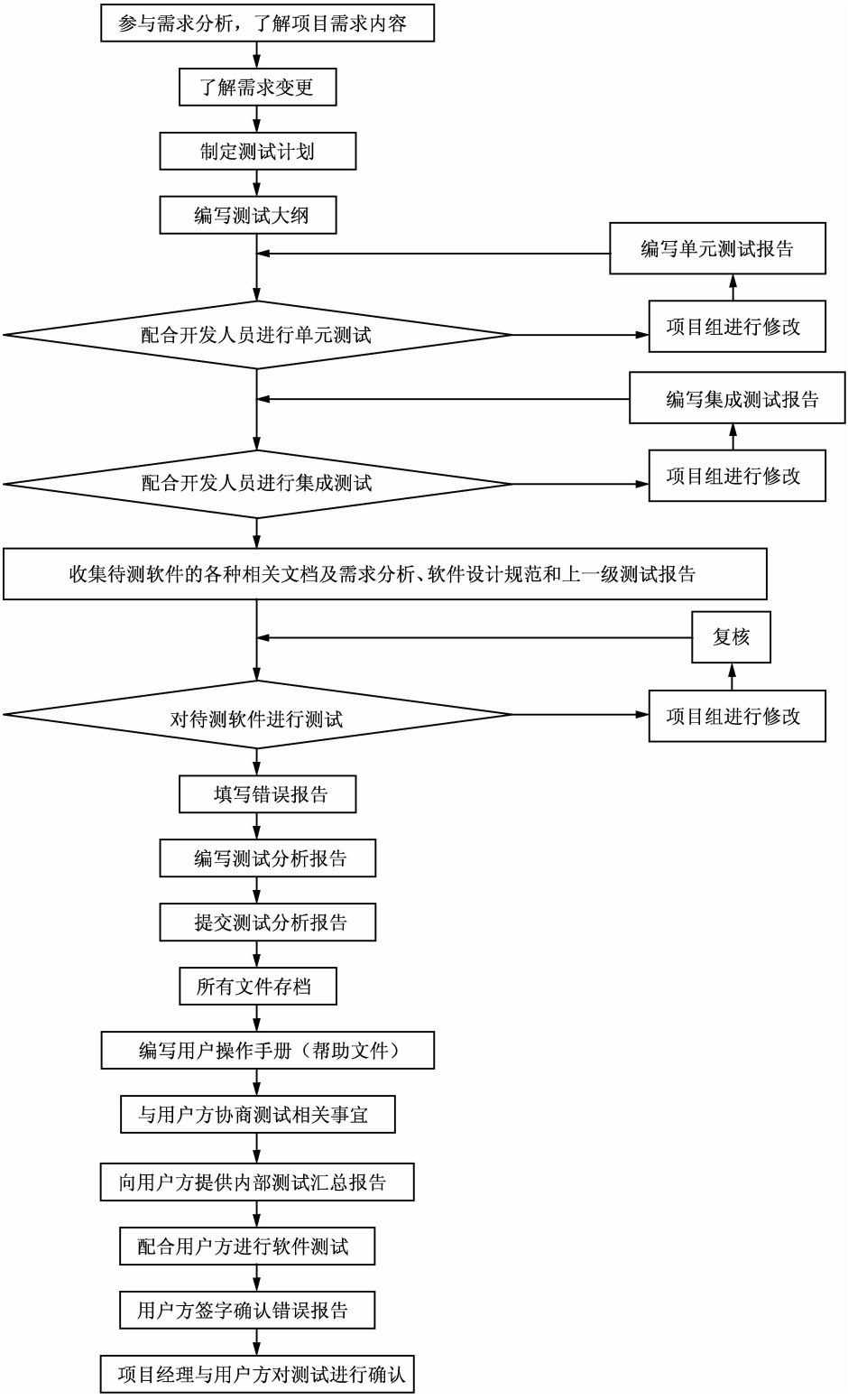

参与需求分析，了解项目需求内容
了解需求变更
制定测试计划
编写测试大纲
编写单元测试报告
配合开发人员进行单元测试
项目组进行修改
编写集成测试报告
配合开发人员进行集成测试
项目组进行修改
收集待测软件的各种相关文档及需求分析、软件设计规范和上一级测试报告
复核
对待测软件进行测试
项目组进行修改
填写错误报告
编写测试分析报告
提交测试分析报告
所有文件存档
编写用户操作手册（帮助文件）
与用户方协商测试相关事宜
向用户方提供内部测试汇总报告
配合用户方进行软件测试
用户方签字确认错误报告
项目经理与用户方对测试进行确认

图 6-2　系统测试流程图

二、系统测试方法

迄今为止，人们还无法证明一个大型复杂程序的正确性，只能依靠一定的测试手段说明该程序在某些条件下没有错误。虽然测试的目的在于寻找错误，并且找出的错误越多，测试就越成功，但是要通过测试把所有隐藏的错误全部找出来是不现实的。

为了节省时间和资源，提高测试效率，就必须精心设计测试用例，即从数量极大的测试用例中挑选一部分具有代表性的进行选择测试，从而获得以较少的测试数据发现最多的错误的最佳测试效果。

所谓测试用例，就是以发现程序错误为目的而精心设计的一组测试数据，包括预定要测试的功能，应该输入的测试数据和预期的结果。设计测试用例是开始程序测试的第一步，也是有效地完成测试工作的关键。设计测试用例最困难的问题是设计测试的输入数据，不同的测试数据发现程序错误的能力差别很大，为了改善测试效果、降低测试成本，应该选用高效的测试数据。因为不可能对所有情况都进行测试，所以选用少量最有效的测试数据做到尽可能完备的测试就很重要了。设计测试用例的基本目标就是确定一组最可能发现多个错误或多类错误的测试数据。

一般来说，对系统进行测试的有效方法有人工测试、黑盒测试和白盒测试三种，其中黑盒测试和白盒测试又合称为机器测试。

（一）人工测试

人工测试又称代码复审，是指不依靠计算机运行程序，而靠人工审查程序或评审软件。人工审查程序重点是对编码质量进行检查，而软件审查除了审查编码还要对各阶段的软件产品进行复查。人工测试可以发现计算机不易发现的错误，特别是软件总体设计和详细设计阶段的错误。据统计，人工测试能有效地发现30%～70%的逻辑设计和编码错误，可以减少系统测试的总工作量。人工测试在发现错误的同时，就能确定错误的位置、类型和性质，所以人工测试在系统测试中有着不可忽视的重要作用。

人工测试主要有个人复查、走查和会审三种方法。

1. 个人复查

个人复查是指源程序编码完成以后，直接由程序员自己进行检查。由于心理上对自己程序的偏爱，所以有些习惯性的错误自己不易发现，如果对功能理解有误，自己也不易纠正。因此，这是针对小规模程序常用的方法，效率不是很高。

2. 走查

走查一般是由3～5人组成测试小组来进行，测试小组的成员应是从未介入过该软件的设计工作的有经验的程序设计人员。测试在预先阅读过该软件资料和源程序的前提下，由测试人员扮演计算机的角色，用人工方法将测试数据输入被测程序，并在纸上跟踪监视程序的执行情况，让人代替机器沿着程序的逻辑走一遍，以发现程序中的错误。由于人工运行很慢，因此走查只能使用少量简单的测试用例，实际上走查只是个手段，是随着“走”的进程不断从中发现错误。

3. 会审

会审测试小组的成员与走查相似，要求测试成员在会审前仔细阅读软件有关资料，根据

错误类型清单(从以往经验看一般容易发生的错误)填写检测表,列出根据错误类型要提问的问题。会审时,由程序作者直接逐个阅读和讲解程序,测试人员逐个审查、提问,讨论可能产生的错误。会审时,对程序的功能、结构及风格等都要进行审定。

(二) 黑盒测试

黑盒测试是指把被测试对象看成一个黑盒子,测试人员完全不考虑程序的内部结构和处理过程,只在软件的界面上进行测试,只看输入和程序结果,以此来证实软件功能的可操作性,检查程序是否满足功能需求及是否能很好地接收数据并产生正确的输出。因此,黑盒测试又称为功能测试或数据驱动测试,一般用来检查系统的基本特征。例如,对于自动售货机,输入钱币就可以得到指定的商品,使用者对其内部如何验钱、找零、查出商品、弹出商品一概不知,也不必知道。

黑盒测试的任务是发现以下错误:

(1) 是否有不正确或遗漏的功能。

(2) 在界面上能否正确地处理合理和不合理的输入数据,并产生正确的输出信息。

(3) 访问外部信息是否有错。

(4) 性能上是否满足要求。

(5) 初始化和终止错误。

黑盒测试又可细分为等价类划分测试法、边界值分析测试法和错误推测测试法等方法。

1. 等价类划分测试法

从前面的叙述可以知道,不可能用所有可能的输入数据来测试程序,而只能从输入数据中选择一部分进行测试。等价类划分测试法就是把所有可能的输入数据划分成若干部分(子集),然后从每一个子集中选取少数具有代表性的数据作为测试用例的一种测试方法。其中,等价类是指某个输入域的子集合,在该子集合中,各个输入数据对于揭露程序中的错误都是等效的,并合理地假定测试某等价类的代表值就等于对这一类值都进行了测试。因此,可以把全部输入数据合理地划分为若干等价类,然后在每一个等价类中取一个数据作为测试的输入条件,这样就把无限的随机测试变成有限的有针对性的等价类测试,从而有效地提高测试效率。利用等价类测试的步骤如下:

(1) 划分等价类。划分等价类时,需要研究程序的功能说明,以确定输入数据的有效等价类和无效等价类。有效等价类是指对于程序的规格说明来说是合理的、有意义的输入数据构成的集合。利用有效等价类可检验程序是否实现了规格说明中所规定的功能和性能。在具体问题中,有效等价类可以是一个,也可以是多个。无效等价类是指对程序的规格说明来说是不合理的或无意义的输入数据所构成的集合。对于具体的问题,无效等价类至少应有一个。在确定输入数据的等价类时常常还需要分析输出数据的等价类,以便根据输出数据的等价类导出对应的输入数据等价类。具体来说,就是从程序的功能说明(如需求说明书)中找出每个输入条件(通常是一句话或一句短语),然后将每一个输入条件划分为两个或多个等价类。可将输入条件、有效等价类、无效等价类以列表形式表现出来,以便查看。

正确分析被测程序的功能和输入条件是正确划分等价类的基础。划分等价类没有一个完整的法则,在具体划分等价类时,可以遵照如下几条经验:

1）如果规定了输入值的范围，则可划分出一个有效的等价类（输入值在此范围内）和两个无效的等价类（输入值小于最小值和大于最大值）。

例如，要求输入学生成绩，范围是 0～100，则可以将有效的等价类划分为“0≤成绩≤100”，无效等价类为“成绩＜0”和“成绩＞100”两个。

2）如果规定了输入数据的一组值，而且程序对不同输入值做不同处理，则每个允许的输入值是一个有效的等价类，此外还有一个无效的等价类（任一个不允许的输入值）。

例如，输入条件上说明教师的职称可为助教、讲师、副教授、教授四种职称之一，则分别取这四个值作为四个有效等价类，把这四个职称之外的任何职称作为无效等价类。

3）如果规定了输入数据必须遵循的规则，则可以划分出一个有效的等价类（符合规则）和若干无效的等价类（从各种不同角度违反规则）。

4）如果已划分的等价类中各元素在程序中的处理方式不同，则应将此等价类进一步划分为更小的等价类。

以上列出的几条经验只是测试时可能遇到的情况中的很小的一部分，实际情况千变万化，根本无法一一列出。为了正确划分等价类，一是要注意积累经验，二是要正确分析被测程序的功能。此外，在划分无效等价类时，还必须考虑编译程序的检错功能，一般说来，不需要设计测试数据用来暴露编译程序肯定能发现的错误。需说明的是，上面列出的经验虽然都是针对输入数据说的，但是其中绝大部分也同样适用于输出数据。

（2）确定测试用例。完成等价类的划分后，就可以开始设计测试用例了。首先，为每一个等价类编号，然后设计一个有效等价类的测试用例。对于各个输入条件，使其尽可能多地覆盖尚未被覆盖过的有效等价类。重复这一步，直到覆盖了所有的有效等价类。然后设计一个无效等价类的测试用例，由于在输入中有一个错误存在时，往往会屏蔽掉其他错误显示，因此，设计无效等价类的测试用例时，只覆盖一个无效等价类，重复这一步，直到覆盖了所有的无效等价类。

例如，某程序规定，对用户输入的分数进行评级，其中 90～100 为 A，80～89 为 B，70～79为 C，60～69 为 D，60 以下为 E，输入分数要求必须是正整数或 0。根据分析可得出如表 6-1所示的等价类划分。

表 6-1　等价类划分

输入条件	有效等价类	无效等价类	编　　号
分数	0～59	空	1
	60～69	负数	2
	70～79	大于 100 的数	3
	80～89	小数	4
	90～100	含字母的字符串	5

然后在有效等价类和无效等价类的各个编号里取一个数据进行测试即可，如在有效等价类中取 48、65、72、88、99 等进行测试；在无效等价类中取空格、－8、125、5.6、abc 等进行测试。

2. 边界值分析测试法

边界值是指输入等价类或输出等价类边界上的值。实践表明，程序员在处理边界情况

时,很容易因疏忽或考虑不周发生编码错误。例如,在数组容量、循环次数以及输入数据与输出数据的边界值附近程序出错的概率往往较大。采用边界值分析法,就是要这样来选择测试用例,使得被测程序能在边界值及其附近运行,从而更有效地暴露程序中潜藏的错误。

边界值分析测试法与等价类划分测试法有两方面的不同:第一,与之前的等价类划分测试法中的抽取一个任意在范围里的值的方法不同,边界值分析测试法中需要一个或多个元素,以使边界类的每一个边界都经过一次测试;第二,与仅仅关注输入条件不同,边界值分析测试法还需要考虑到输出空间的边界范围。

使用边界值分析测试法时,应遵循以下原则:

(1) 如果输入条件规定了值的范围,可以选择正好等于边界值的数据作为有效的测试用例,同时还要选择刚好越过边界值的数据作为无效的测试用例。

例如,输入条件说明学生成绩的取值只能是整数,范围是 0～100,则可以选择正好等于边界值的数据 0 和 100 作为合理的测试用例,同时还要选择刚好越过边界值的数据－1 和 101 作为不合理的测试用例。

(2) 如果输入条件指出了输入数据的个数,则按最大个数、最小个数、比最小个数少 1、比最大个数多 1 等情况分别设计测试用例。例如,一个输入文件可包括 1～255 个记录,则分别设计有 255 个记录、1 个记录、0 个记录以及 256 个记录的输入文件的测试用例。

(3) 对每个输出条件分别按照以上原则(1)或(2)确定输出值的边界情况。例如,一个学生成绩管理系统规定,只能查询 2004—2007 级大学生的各科成绩,可以设计测试用例,保证可以查询范围内的学生的成绩,还需设计查询 2003 级、2008 级学生成绩的测试用例(无效等价类)。

(4) 如果程序的规格说明给出的输入或输出域是个有序集合(如顺序文件、线性表、链表等),则应选取集合的第一个元素和最后一个元素作为测试用例。

通常设计测试用例时总是联合使用等价类划分和边界值分析两种技术。例如,税法规定个人的收入所得税从超过 2 000 元开始征收。如果用一个程序来计算税款,则“收入≤2 000”就是一个判定条件,满足条件的人免税,否则对超出 2 000 元的部分征税。在选择测试用例时,可以用 1 800、2 200 两个测试数据分别代表免税和征税两个等价类,还可以以 2 000 这个边界值作为测试数据。

3. 错误推测测试法

使用边界值分析测试法和等价类划分测试法可以帮助开发人员设计具有代表性的、容易暴露程序错误的测试用例,但不同类型不同特点的程序通常又有一些特殊的容易出错的情况。此外,有时分别使用每组测试数据时程序都能正常工作,这些输入数据的组合却可能检测出程序的错误。一般来说,即使是一个比较小的程序,可能的输入组合数也往往十分巨大,因此必须依靠测试人员的经验和直觉来推测程序中可能存在的各种错误,从而有针对性地设计检查这些错误的测试用例,这就是错误推测测试法。

错误推测测试法的特点是没有确定的步骤,在很大程度上是凭经验进行的。对一个排序程序,可以列出以下几种特别需要检查的情况:

(1) 输入表为空。

(2) 输入表中只有一行。

(3) 输入表中所有的行都具有相同的值。

(4) 输入表已经是排序的。

对一个采用两分法的检索程序,可以列出以下这些需要检查的情况:

(1) 被检索的表格只有一行。

(2) 表格的行数恰好是 2 的幂(如 64)。

(3) 表格的行数比 2 的幂多 1 或少 1(如 65、31)。

(三) 白盒测试

由于黑盒测试的工作量大,于是人们想到可以把程序打开,让程序的每一条语句都执行一次本测试用例。例如,程序中有判断语句"if X then A else B",如果测试用例选得不当,测了 100 次都是 A,从没有测到过 B,那么即使这样的测试通过了,在用户的实际操作中,当某个条件执行到 B 时,还是有可能出错的。白盒测试正是为了避免这种情况才提出来的。

白盒测试也叫路径测试或结构测试。它将软件看成一个透明的白盒子,按照程序的内部结构和处理逻辑来选定测试用例,主要是对软件的逻辑路径及过程进行测试,检查它与设计是否相符。逻辑覆盖法是一种典型的白盒测试方法,它从程序内部的逻辑结构出发设计测试用例。由于不可能测试程序中的所有路径,那么有选择地执行程序中某些最有代表性的路径是唯一可行的方案。根据测试数据覆盖程序逻辑的程度,可以划分为如下五种不同等级的覆盖:

1. *语句覆盖*

语句覆盖是指设计的测试用例能使被测试程序中的每个语句至少执行一次。图 6-3 是一个被测程序的流程图。

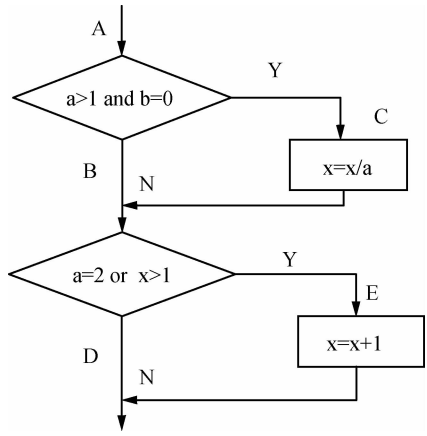


图 6-3　被测程序的流程图

它的源程序(用 C 语言书写)如下:

```
void example(a,b,x)
{
    float a,b,x;
    if ((a>1) && (b==0))
      x=x/a;
    if ((a==2) || (x>1))
```

```
    x=x+1;
}
```

为了使程序中每个语句都被执行一次，程序的执行路径应该是 ACE，为此只需要输入测试数据“a=2，b=0，x=5”就能满足要求。

但语句覆盖测试不充分，上述测试用例不能测试路径 ABD，也不能检查出第一个 if 语句中的逻辑运算符“&&”错写成“||”的错误，可见语句覆盖发现错误的能力比较弱。

2. 判断覆盖

判断覆盖是指设计的测试用例不仅使每个语句必须至少执行一次，而且每个判断的每个分支都至少执行一次。

对于上述例子来说，若设计两组测试数据，在一次测试中能分别覆盖路径 ACE 和 ABD 或分别覆盖路径 ACD 和 ABE，就可达到判断覆盖的标准。例如，可选择数据如下：

a=3，b=0，x=1　　（覆盖 ACD）；

a=2，b=1，x=3　　（覆盖 ABE）。

判断覆盖比语句覆盖的判断能力强，但是对程序逻辑的覆盖程度仍然不高，例如，若第二个判断语句中的 x>1 写成了 x<1，上面的测试数据就检查不出这个错误。

3. 条件覆盖

条件覆盖的含义是：不仅每个语句至少执行一次，而且使判断表达式中的每个条件能取到各种可能的结果。

上述例子中共有两个判断表达式，每个表达式中有两个条件，为了做到条件覆盖，应该选取测试数据，使得在第一个判断表达式处有下述各种结果出现：a>1，a≤1，b=0，b≠0；在第二个判断表达式处有下述各种结果出现：a=2，a≠2，x>1，x≤1，只需要使用下面两组测试数据就可以达到上述覆盖标准：

a=2，b=0，x=4　　（满足 a>1，b=0，a=2 和 x>1 的条件，执行路径 ACE）；

a=1，b=1，x=1　　（满足 a≤1，b≠0，a≠2 和 x≤1 的条件，执行路径 ABD）。

条件覆盖通常比判断覆盖测试充分，但是也可能有例外的情况，虽然每个条件都取到了两个不同的结果，判断表达式却始终只取一个值。例如，如果使用下面两组测试数据，则只满足条件并不满足判断覆盖标准：

a=2，b=1，x=1　　（满足 a>1，b≠0，a=2 和 x≤1 的条件，执行路径 ABE）；

a=1，b=0，x=5　　（满足 a≤1，b=0，a≠2 和 x>1 的条件，执行路径 ABE）。

4. 判断/条件覆盖

判断/条件覆盖的含义是：选取足够多的测试数据，使得判断表达式中的每个条件都取到各种可能的值，而且每个判断表达式也都取到各种可能的结果。

对于上述例子而言，下面两组测试数据满足判定/条件覆盖标准：

a=2，b=0，x=4；

a=1，b=2，x=1。

但是有时判定/条件覆盖也并不比条件覆盖更强。

5. 条件组合覆盖

条件组合覆盖是更强的逻辑覆盖标准，它要求选取足够多的测试数据，使得每个判断表达式

中条件的各种可能组合都至少出现一次。对于上述例子，共有八种可能的条件组合，它们是：

a＞1，b＝0；

a＞1，b≠0；

a≤1，b＝0；

a≤1，b≠0；

a＝2，x＞1；

a＝2，x≤1；

a≠2，x＞1；

a≠2，x≤1。

下面的四组测试数据可以使上面列出的八种组合每种至少出现一次：

a＝2，b＝0，x＝4；

a＝2，b＝1，x＝1；

a＝1，b＝0，x＝2；

a＝1，b＝1，x＝1。

上述测试用例的设计技术各有优缺点，没有哪一种是最好的，更没有哪一种可以代替其余所有技术。同一种技术在不同场合效果可能相差很大，因此需要联合使用。通常，设计测试用例的做法是用黑盒法设计基本的测试用例，再用白盒法补充一些方案。应该强调指出，即使使用上述综合策略设计测试用例，仍然不能保证通过测试可以发现一切程序错误。

三、系统测试的类型

按照测试的对象划分，系统测试可以分为四种类型，即单元测试、集成测试、确认测试和系统测试。

（一）单元测试

单元测试就是对一个模块或几个模块组成的小功能单元模块进行的测试。这种测试一般使用白盒测试技术，可在多个模块间并行进行。模块编码完成且没有语法错误之后，可以开始单元测试。模块不是孤立的，需要设计驱动程序才可以测试。驱动程序是一种简单的调用程序，它便于各种测试用例的输入，可记录被测模块用到的全局数据，并且给出明显的输出。

在这种测试中，测试用例应该覆盖接口、模块的数据结构、边界条件、独立路径和错误处理路径等内容。

（二）集成测试

集成测试就是把本信息系统的所有子模块组合在一起进行测试。一般来说，单元测试比较容易进行，但是集成测试的问题比较多，且运行一次的时间也很长。因此，选择集成测试的策略非常重要。集成测试有两种测试策略，即自底向上集成和自顶向下集成。

（1）自底向上集成。最底层的模块称为原子模块，首先测试这些原子模块，然后逐步上升，测试相关的模块组合，最后测试整个程序。这种策略的优点是易于设计测试用例；缺点是不到最后，总没有一个完整的概念。

（2）自顶向下集成。这种测试一开始就抓住主控模块，一块块地进行测试。根据被测

试系统的特点，可以采用深度优先方式，测完一块再进入下一层模块，直到进入最底层模块，然后再测其他未测模块。

（三）确认测试

集成测试完成之后，所开发的软件已经是一个完整的软件包，这时开始进行确认测试。确认测试就是验证该软件是否达到了用户的要求。信息系统设计规格说明书是确认测试的准则。

确认测试一般是通过一系列的黑盒测试来验证系统是否满足用户的要求的。需要确认的内容主要包括以下三个方面：

（1）所有功能需求都可以满足。

（2）所有的性能要求都可以满足。

（3）所有的文档都已经修改结束。

（四）系统测试

程序软件只是信息系统的一部分，最后要形成一个整体，还需要包括硬件、软件和与此相关的设备，对这个整体进行测试即为系统测试。典型的系统测试包括恢复测试、安全测试、强度测试和性能测试。

（1）恢复测试，是人为地制造一种故障，查看系统能否正确恢复。如果系统能够正确恢复，那么需要评价它的重新初始化、数据恢复、重新启动是否正确。如果需要由人工恢复，则要评价它操作的复杂程度、耗费时间、效果等能否被接受。

（2）安全测试，就是测试系统对抗外来无意或恶意攻击的能力。测试时，测试人员充当黑客，从本软件的最薄弱之处进攻，什么方法都可以，如以正常手段获取口令、解体新开发的软件、将海量数据压向系统、破坏恢复能力等。

（3）强度测试，是测试软件在不正常情况下失衡的极点，如数据流通量达到多大就会失衡，是否可以全天候工作等。

（4）性能测试，是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。负载测试和压力测试都属于性能测试，两者可以结合进行。负载测试主要是确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。压力测试是通过确定一个系统的瓶颈或者不能接收的性能点，来获得系统能提供的最大服务级别的测试。

四、调试

调试与测试的意义不同，仅就测试而言，其目的是发现系统中的错误，但发现错误并不是开发者的最终目的，系统开发的最终目的是开发出高质量的完全符合用户需求的信息系统。因此，测试发现问题后，还必须诊断错误，改正错误。进行测试时，通过比较测试结果与预期结果的差异来确认错误的存在，而错误如何改正，这就是调试的内容，所以调试又称排错或纠错。准确判定出错位置并不是一件容易的事，它要占去测试工作的大部分工作量，而在找到错误后，改正错误往往相对容易得多。

（一）调试步骤

一般来说，系统调试可遵从以下步骤：

（1）从错误的外部表现形式入手，确定程序中出错的位置。

(2) 研究有关部分的程序,找出错误的内在原因。

(3) 修改设计代码以排除这个错误。

(4) 重复进行能够暴露这个错误的某些相关测试,判断错误是否被排除。

(5) 如果所作修改无效,则撤销此次行动,重复上述过程,直到找到一个有效的解决办法为止。

(二) 调试方法

目前,关于调试还缺乏系统的理论方法,因此调试方法多是实践中的经验和积累。以下几种方法在实际调试过程中可以借鉴:

1. 在程序中插入打印语句

此方法的优点是可以显示程序的动态过程,比较容易检查源程序的有关信息;缺点是效率低,可能输出大量无关的数据,发现错误带有偶然性。同时,使用此方法还要修改程序,可能会掩盖错误,改变关键的时间关系或把新的错误引入程序。因此,一般是在可能出错的地方插入打印语句,调试完毕要再将打印语句删除或注释掉。

2. 运行部分程序

有时为了测试某些被怀疑有错的程序段,需要将整个程序反复执行多次,这样就使很多时间浪费在执行已经确定正确的程序段上。在这种情况下,应设法使被测程序只执行需要检查的程序段,以提高效率。

3. 借助于调试工具

目前,大多数程序设计语言都有专门的调试工具,可以利用这些工具分析程序的动态行为。例如,借助"追踪"功能可以追踪子程序调用、循环与分支执行路径、特定变量的变化情况等;利用"设断点"可以执行特定语句或改变特定变量值引起的程序中断,以便检查程序的当前状态。另外,还可以借助调试工具观察或输出内存变量的值,这样能大大提高调试程序的效率,但缺点是会产生大量的无关信息。

4. 归纳法

归纳法是从个别到整体的推理过程。它从收集个别的故障症状开始,分析各种症状的相互关系后,就有可能将它们归纳为某一些假想的错误,如果这些假想能被证实,就找到了真实的病根。归纳法排错的步骤如下:

(1) 设置数据,具体做法是列出所有已知信息,即程序能正确完成什么、存在什么类型的错误。

(2) 组织数据,是从特殊到一般的处理过程,也就是使所设置的数据结构化,便于从中发现矛盾。

(3) 假设错误可能原因,研究错误迹象与数据间的关系,提出一个或多个假设原因。若提不出假设,则要设置更多的测试数据和执行附加的测试用例;若存在几种假设,则首先选择可能性最大的一个。

(4) 证明错误原因假设,通过比较假设的错误原因与原来的错误数据,确定该假设是否完全解释了原有的错误迹象。如果比较结果不满意,则说明该假设不完全适用,要重新假设可能的错误原因或重新设置数据。

5. 演绎法

演绎法是从一般到特殊的推理过程。根据测试获得的错误症状，可以先列出一批可能的原因，接着在这一大范围的设想中，逐一排除根据不足或与其他测试结果有明显矛盾的病因，然后对余下的一种或数种病因进行详细的鉴别，确定真正的原因。演绎法的步骤如下：

(1) 列出可能的错误原因。

(2) 仔细分析现有数据，寻找矛盾，排除所有无关因素，找出主要原因。若全部原因都被排除，则要设计附加的测试用例来发现新的错误原因。

(3) 利用可靠的错误迹象来完善错误原因假设，使之更加具体化。

(4) 证明错误原因假设的正确性。

第四节 系统转换

完成系统的测试工作之后，系统的实施将进入到新旧系统的转换阶段，这是前面工作的延续。这一阶段的工作常常被人们所忽视，但实际上这一工作的好坏对系统的安全性、可靠性、准确性都是非常重要的。用计算机辅助的管理信息系统一般是在现行的手工管理系统基础之上建立起来的，因此必须协调新旧系统之间的关系，否则，将造成企业正常工作的紊乱与中断。

一、系统转换前的准备工作

根据信息系统实际开发和应用的情况，确定了系统转换的方式以后，要做大量的准备工作，主要包括数据准备、文档准备、人员培训和系统初始化四个方面。

(一) 数据准备

数据准备是从老系统中整理出新系统运行所需的基础数据和资料，即把老系统的文件、数据加工成符合新系统要求的数据，其中包括历史数据的整理、数据口径的调整、数据资料的格式化、统计口径的变化以及个别数据及项目的增、删、改等。特别是对于那些采用手工方式进行信息处理的老系统，需要用户和系统开发人员共同参与数据的准备过程。因为这些数据要被装入到新系统中，而计算机信息系统对数据的加工处理有自己的要求，所以数据的准备不仅是要归类整理，还要进行编码等工作。由于系统执行需要的可能是一年、几年甚至更长的时间段内的数据，所以，数据的输入流程所耗费的人力、时间是相当多的，相应地也必然耗费一定的财力。如果新系统是在已有的信息系统上开发的，应尽可能使数据转换工作自动化。这种转换工作也是十分复杂且耗时的，有时要涉及数据库的改组或重建。

(二) 文档准备

系统调试完以后应有详细的说明文档供人阅读。该文档应使用通用的语言说明系统各部分如何工作、维护和修改。系统说明文档大致可分为以下四类：

1. 系统一般性说明文件

系统一般性说明文件包括以下几个方面的内容：

(1) 用户手册：主要向用户介绍系统的全面情况，包括目标和有关人员情况。

(2) 系统规程:是为系统的操作和编程等人员提供的总的规程,包括计算机操作规程、监理规程、编程规程和技术标准。

(3) 特殊说明:是系统随着外部环境的变化而作出相应调整的说明,它是可以不断进行补充和发表的。

2. 系统开发报告

系统开发报告包括以下几个方面的内容:

(1) 系统分析说明书:包括系统分析建议和系统分析执行报告。

(2) 系统设计说明书:涉及输入、输出、数据库组织、处理程序、系统监控等方面。

(3) 系统实施说明书:主要涉及系统分调、总调过程中某些重要问题的回顾和说明以及人员培训、系统转换的计划及执行情况。

(4) 系统利益分析报告:主要涉及系统的管理工作和对职工所产生的影响,系统的费用、效益分析等方面。

3. 系统说明书

系统说明书主要包括整个系统程序包的说明、业务流程图和数据流程图、程序清单、程序实验过程说明、输入输出样本、程序所有检测点设置说明、各个操作指令、控制台指令、操作人员指示书等。

4. 操作说明

操作说明包括系统的操作顺序、各种参数输入条件、数据的备份和恢复操作方法以及系统维护的有关注意事项。

(三) 人员培训

整个管理信息系统开发的成功与否取决于人们是否理解它,是否知道如何有效地使用它。因此,为了使新系统能够按预期目标正常运行,要对相关人员进行必要的培训。

实际上,在系统开发的早期阶段,就应该开始考虑制定一份培训计划。在这份计划中,首先要确定培训人员,然后要针对不同的人员确定培训内容。

需要进行培训的人员主要有以下三类:

1. 事务管理人员

新系统能否顺利运行并获得预期效果,与事务管理人员(或主管人员)的理解和支持有密切的关系。因此,可以通过讲座、报告会的形式对事务管理人员进行培训。培训的主要内容应该包括新系统的目标、功能;新系统的结构及运行过程;采用新系统对企业组织机构、工作方式等产生的影响;采用新系统后,对职工必须学会新技术的要求;今后如何衡量任务完成情况等。

2. 系统操作员

对系统操作员的培训是与编程和测试工作同时进行的,应该给系统操作员提供比较充分的培训时间。其培训的主要内容应该包括必要的计算机硬、软件知识;键盘指法、汉字输入等训练;新系统的工作原理;新系统的输入方式和操作方式;简单出错的处置知识;运行注意事项等。系统操作员是管理信息系统的直接使用者,统计资料表明,管理信息系统在运行期间发生的故障大多数是由于使用方法错误而造成的。因此,对用户系统操作员的培训应该是人员培训工作的重点。

3. 系统维护人员

对于系统维护人员来说，除了要求具有较好的计算机软、硬件知识外，还必须对新系统的原理和维护知识有较深刻的理解。在较大的企业或部门中，系统维护工作一般由计算机中心或计算机室的计算机专业技术人员担任。培训系统维护人员的最好途径就是让他们直接参与系统的开发工作，这样有助于他们了解整个系统的全貌，为今后的工作打下良好的基础。对于大、中型企业或部门用户，系统维护人员培训工作应列入该企业或部门的教育计划中，在系统开发单位的配合下共同实施。

（四）系统初始化

信息系统从开发到投入应用必须经过一个初始化的过程。系统初始化包括对系统的运行环境和资源进行设置、设定系统运行和控制参数、加载数据以及调整系统与业务工作同步等内容。其中，加载数据是工作量最大且时间最紧迫的一个重要环节，因为大量的原始数据需要一次性输入系统，而组织生产经营管理业务活动又会不断产生新的信息，如果不能在有限的时间内将数据输入完毕并启动系统，则新的数据变化会造成系统中的数据失效。系统初始化中大量的数据加载工作是系统启动的先决条件，并且大多都是由手工输入完成的，因此，输入中最重要的是正确性。数据加载中出现的数据错误大体有四种类型：原始数据中就存在错误；数据整理工作中产生的错误；输入错误；新系统可能的程序错误。在系统初始化过程中要注意采取一定的手段来查错和纠错，以防止错误的数据进入系统。例如，为了保证输入的正确性，有时采用数据重复输入法，把同一批数据分两次重复输入，由系统自动核对输入的差异，以检查数据输入的错误。这样，尽管输入工作量增加了一倍，但有效地避免了数据的输入错误。如果数据内部有计算或平衡关系，可用程序对输入的数据进行检查，以发现其可能存在的错误。如果老系统是计算机系统，则数据加载的主要工作将是进行数据和文件的转换，使数据转入新系统中。总之，数据加载工作量大、要求高，应予以高度重视。

二、系统转换的方式

系统转换是指用新的信息系统代替原有系统的一系列过程，其最终目的是将信息系统完全移交给用户使用。为了保证原有系统有条不紊地顺利转移到新系统中，在系统转换前应仔细拟订方案和措施，确定具体的步骤。

系统的转换方式通常有三种，如图 6-4 所示。

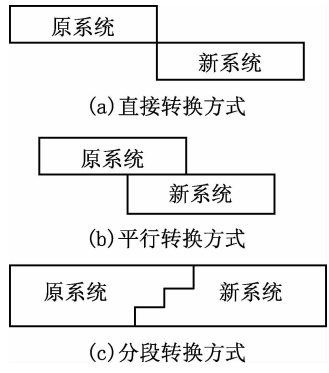


图 6-4 系统的转换方式

（一）直接转换

直接转换就是在原有系统停止运行的某一时刻，新系统立即投入运行，中间没有过渡阶段。用这种方式时，人力和费用最省，适用于新系统不太复杂或原有系统完全不能使用的场合，但新系统在转换之前必须经过详细调试和严格测试。同时，转换时应做好准备，万一新系统不能达到预期目的，须采取相应措施。此外，采用这种转换方式，管理层还要仔细慎重选择转换的时间。直接转换的示意图如图 6-4(a)所示。

（二）平行转换

平行转换就是新、旧系统同时并行工作一段时间（一般为 3～5 个月），先以旧系统为作业系统，以新系统的处理做校核；过一定阶段后，再以新系统作为作业系统，而以旧系统的处理做校核；最后，用新系统取代旧系统。平行转换的示意图如图 6-4(b)所示。平行转换方式的主要问题是费用太高，这是因为并存期间，新、老系统的工作人员也要并存，需要双倍的费用。这种转换方式有利于减轻管理人员心理压力，安全性较好，两个系统的数据一般不具备可比性，适合于处理过程复杂、数据重要的系统。

（三）分段转换

分段转换就是分阶段，一部分、一部分地以新系统取代旧系统。分段转换的示意图如图 6-4(c)所示。分段转换方式的特点是先把系统的部分工作交给新系统处理，经过一段运行确认系统稳定后，再把另外一部分工作换下来，这样逐步把整个系统换成新系统。它既避免了直接转换方式的风险性，又避免了平行转换方式发生的双倍费用。采用分段转换时，各自系统的转换次序及转换的具体步骤均应根据具体情况灵活考虑。该方式适用于较大的重要系统，既稳妥可靠而工作量又不是太大，适合于会计系统应用。

第五节　系统评价

信息系统建成并运行一段时间后，就要对其进行技术性能及经济效益等方面的评价。评价的目的是检查系统是否达到了预期的目标，技术性能是否达到设计要求，系统的各种资源是否得到充分利用，经济效益是否理想等，指出系统的优势与不足，为以后的改进与扩展提出意见。

一、系统评价的意义

不论是在发达国家还是在发展中国家，信息系统项目建设的成功率都远不及其他建设项目，曾有 IT 黑洞之说。由此，信息系统项目建设中存在的问题日益为人们所重视。通过分析案例，人们发现造成信息系统建设失败的原因纵然有很多，但是，几乎每个失败的案例都和缺乏及时必要的评价有很大关系。因此，信息系统评价被提上议程，并成为信息系统项目管理中的一个热点。

所谓系统评价，就是指对信息系统的性能进行全面估计、检查、测试、分析和评审，包括对实际指标和计划指标进行对比，以提高系统目标的实现程度，并对系统建成后产生的经济

效益和社会效益进行全面的评价。系统评价对企业来说具有十分重要的作用,因为评价不仅可以帮助企业对信息系统的投资价值进行正确的评估,还可以作为一种重要的反馈和学习工具,帮助企业发现信息系统项目中成功或不成功的潜在要素,改善信息系统管理的方法和过程,从而促进系统项目向期望的方向发展。

对信息系统进行评价必须有明确的目的,就是说,为什么要对信息系统进行评价,对信息系统进行哪些方面的评价,在评价之前都要确定下来。主要的评价内容有信息系统的使用效果、开发质量和稳定性、各项指标是否达到总体规划的要求、经济效益等,对于一些比较大的系统,还有可能要对其进行全方位的综合评价。

要对系统做出客观公正的评价,就要按照一定的步骤来进行。系统评价的一般步骤是:首先要确定评价的对象,也即确定对哪一个信息系统进行评价;其次是明确评价的目标,就是明确信息系统评价的主要目的;再次是建立指标体系,一定要保证建立的评价指标体系能反映被评价问题的主要方面;然后选择适当的评价方法,适当指的是选择的方法并不一定要复杂,只要能达到评价的目标,就是一个好的方法;最后根据系统的实际情况,对系统进行客观公正的评价,写出评价报告。

对信息系统做出一个客观公正的评价有着重要的意义。在一个信息系统的生命周期中会涉及不同的主体,这些主体都希望对信息系统有一个客观公正的认识。对于企业内部的领导者来说,信息系统评价可以帮助他们进行判断。例如,在项目投资前期,判断做或不做以及应该采用何种方式进行项目的建设;在建设过程中判断资源的分配是否合理和是否需要变更建设方式甚至放弃当前的项目;在 MIS 项目完成时,判断项目成功与否,建设效率和效果如何。对信息系统的供应商来说,通过评价可以让他们了解现在的信息系统的优点和不足,从而为他们不断地改进信息系统的功能提供思路,也可以通过评价提高信息系统的知名度,以利于进一步在行业领域内推广信息系统。

二、系统评价的内容

管理信息系统的评价与其他工程系统的评价相比具有自己的特点。管理信息系统中包括了信息资源、技术设备、人和环境等诸多因素,系统的效能是通过信息的作用和方式表现出来的,而信息的作用又是通过人在一定的环境中,借助以计算机技术为主体的工具进行决策和行动表现出来的。管理信息系统的效能既有有形的,也有无形的;既有直接的,也有间接的;既有固定的,也有变动的。管理信息系统的评价具有复杂性和特殊性。

系统评价的主要依据是系统日常运行记录和现场实际监测得到的数据。评价的结果可以作为系统维护、更新以及进一步开发的依据。信息系统的评价工作由系统开发人员、系统管理与维护人员、系统用户、用户单位领导及系统外专家等共同参与,评价方式可以是鉴定或评审等,评价的结论以书面的评价报告或评价意见等提出。评价结论也是系统的重要文档,应予以收存归档,统一保管。具体来说,系统评价主要包括以下三个方面的内容:

(一) 预定的系统开发目标的完成情况

预定的系统开发目标的完成情况是指对照系统目标和组织目标检查系统建成后的实际完成情况,查看是否满足了科学管理的要求,各级管理人员的满意程度如何,有无进一步的改进意见和建议,为完成预定任务用户所付出的成本(人、财、物)是否限制在规定范围之内,开发工作和开发过程是否规范,各阶段文档是否齐备,功能与成本比是否在预定的范围

之内，系统的可维护性、可扩展性、可移植性如何，系统内部各种资源的利用情况等。

（二）系统运行实用性评价

系统运行实用性评价主要评价系统运行是否稳定可靠，系统的安全保密性能如何，用户对系统操作、管理、运行状况的满意程度如何，系统防止错误操作和进行故障恢复的能力如何，系统功能的实用性和有效性如何，系统运行结果对组织各部门的生产、经营、管理、决策和提高工作效率等的支持程度如何，以及对系统的分析、预测和控制的建议的有效性如何，实际被采纳了多少，被采纳的建议的实际效果如何等。

（三）设备运行效率的评价

设备运行效率的评价主要评价设备的运行效率如何，数据传送、输入、输出与其他加工处理的速度是否相匹配，各类设备资源的负荷是否平衡及其利用率如何等。

信息系统在运行与维护过程中不断地发生变化，因此，评价工作不是一项一次性的工作，系统评价应定期地进行或在系统有较大改进后进行。信息系统的第一次评价一般安排在开发完成并投运一段时间进入相对稳定状态后。大型管理信息系统的开发往往分成一期工程、二期工程、三期工程等阶段，前期工程的评价对决定是否继续开发后续工程有参考作用。

三、系统的技术评价

系统技术性能方面的评价主要是评价现有系统硬件和软件资源在技术性能上是否能够满足应用系统的要求。例如，数据传输率是否能够满足数据处理的要求，是否有足够的辅助存储器来保存必要的数据文件，在规定的时间内 CPU 能否对所有的请求做出快速响应等。系统技术评价主要评价以下几个方面：

（一）对管理信息系统的功能评价

在新系统的开发规划中，已经明确地规定新系统要实现的功能目标。对新系统的功能评价，就是按照规划来检查新系统的功能实现情况。例如，预期的功能是否已经全部实现，是否能够满足用户的要求，服务质量如何，人员组织和安全保密措施是否完善等。

（二）系统操作方面的评价

系统操作方面的评价主要是对输入出错率、输出的及时性和利用情况等进行评价。例如，是否能够正确地提供输入数据，输出结果是否可用和适用等。

（三）对现有硬件和软件的评价

对管理信息系统中现有硬件和软件进行评价的目的是检查系统内是否有未被充分利用的资源，或者是否会由于某些资源不足与性能不够完善而影响了系统的功能和效率的提高。对硬件和软件系统的评价主要借助于硬件监控器、软件监控程序、系统运行记录和现场实际观测记录等方法和工具。

(1) 利用硬件监控器和软件监控程序进行评价。硬件监控器既能收集到 CPU 工作的数据，也能收集到外部设备工作情况的数据。软件监控程序可以记录特定程序或程序模块执行情况的数据。利用监控器和监控程序可以对闲置的资源、瓶颈设备以及负荷不均匀的情况进行及时检测，从而帮助用户查明系统工作效率过低的原因。

(2) 根据系统运行记录和现场实际观测记录进行评价。新系统日常运行记录是进行系统评价的主要参考资料。通过对运行记录的分析,可以明确使用得最多、最频繁的软件设计是否合理,目前效果如何以及系统的故障率等问题。另外,通过对计算机运行情况的现场观测,可以有效地评价系统资源安排是否合理。

四、系统经济效益的评价

使用新系统后产生的经济效益是评价新系统的一个决定性因素,但是经济效益的评价是一个非常复杂的问题,因为要收集各种定量的指标值需要较长的时间。同时,有的经济效益是不能单纯通过数字来反映的。目前,一般将系统经济效益分成直接经济效益和间接经济效益两种进行统计。

(一) 直接经济效益

系统的直接经济效益是指可以定量计算的效益,通常可通过以下指标来反映:

(1) 系统投资额。包括系统硬件、软件的购置和安装,应用系统的开发或购置所投入的资金。另外,企业内部投入的人力、材料等也应计入这部分。要较精确地计算直接经济效益还应考虑资金投入的时间等因素。

(2) 系统运行费用。包括通信和消耗性材料费用、系统投资折旧费及硬件日常维护费等。由于信息系统的技术成分较高、更新换代快,一般折旧年限取 5～8 年。另外,管理人员费用等也应计入系统运行费用。

(3) 系统运行新增加的效益。主要反映在成本降低、库存积压减少、流动资金周转加快与占用额减少、销售利润及人力减少等方面。新增效益可采用总括性的、在同等产出水平下使用信息系统所致的年生产经营费用节约额来表示,也可分别计算上述各方面的效益,然后求和表示。由于引起企业效益增减的因素关联错综复杂,新增效益很难进行精确的计算。

(4) 投资回收期。是指通过新增效益,逐步收回投入的资金所需的时间,它也是反应系统经济效益好坏的重要指标。经简化后不考虑贴现率的投资回收期可用下式计算:

$$S = a + b/(c - d)$$

其中,S 为投资回收期,单位为年;a 为资金投入至开始产生效益所需的时间,单位为年;b 为投资额,单位为万元;c 为系统运行后每年系统新增加的效益,单位为万元/年;d 为系统运行费用,单位为万元/年。

(二) 间接经济效益

间接经济效益是指通过改进组织结构及运作方式、提高人员素质等途径,促使成本下降、利润增加而逐渐地间接地获得的效益。由于成因关系复杂,计算困难,只能对其进行定性的分析。尽管间接效益难以估计,但其对企业的生存与发展所起的作用往往要大于直接经济效益。

一般信息系统的成功应用所产生的间接经济效益可以体现在以下几个方面:

(1) 用计算机代替人工处理信息,减轻了管理人员的劳动强度,使他们有更多的时间从事调查研究和决策工作;系统信息的共享与交互使部门之间、管理人员之间的联系更紧密,可以加强他们的协作,提高企业的凝聚力,从而提高了管理效率。

(2) 对组织为适应环境所作的结构、管理制度与管理模式等的变革起到巨大的推动作

用，这种作用一般无法用其他方法实现。

(3) 能显著地提高企业知名度，对外可提高客户对企业的信任程度，对内可提高全体员工的自信心。信息系统的应用也使管理人员获得许多新知识、新技术与新方法，进而提高他们的技能素质，拓展思路，进入学习与掌握新知识的良性循环。

(4) 对企业的规章制度、工作规范、定额与标准、计量与代码的基础管理产生了很大的促进作用，为其他管理工作提供了有利的条件。例如，销售管理系统的建立可提供较强的查询功能，可以提高服务质量并及时提供各项经营决策；财务管理系统的建立可大大提高业务处理能力，减少差错，提高资金周转率；建立生产管理系统可以更合理地安排人力物力、及时掌握生产进度和产品质量，从而提高生产率和生产管理水平；物资管理系统的建立可以明显提高库存记录的准确性和及时性，减小库存量，从而减少物资的积压浪费，同时也能保证生产用料的供应，避免因原料短缺而使生产停顿，最终提高生产力等。以上这些都是间接经济效益的表现形式。

总之，计算机管理系统的建立将对企业或部门的管理工作产生重大影响，对这些直接或间接的效益必须要充分认识，给以肯定。

五、系统评价报告

系统评价的结果应形成正式的书面文件即系统评价报告，它应包括以下几个方面的内容：

(1) 列出系统分析时所提出的新系统的目标、结构与功能，并将它们与实现的新系统逐一比较，说明其满足的程度。

(2) 有关的文件、任务书、参考资料等。

(3) 经济指标的评价，主要包括以下几个方面的内容：

1) 系统开发及运行的总费用与原来估计费用相比。

2) 预期效益与实际效益相比。

3) 费用—效益分析的结果。

(4) 性能指标的评价，主要包括以下几个方面的内容：

1) 工作效率指标。

2) 适应性：新系统对环境变化的适应能力及可扩充性。

3) 维护性：对系统出错的维修能力。

4) 安全、可靠性及保密性。

(5) 管理指标的评价：用户、领导者与管理人员的反映及评价。

(6) 综合性评价：包括对各类指标的综合考虑与分析以及系统的不足与待改进之处。

总之，要有可靠的数据、定量的分析、令人信服的事实，配以各种图表，方可形成系统评价报告。系统评价报告既是对新系统开发工作的最好的评定与总结，也是进一步进行维护工作的依据，通常由此产生对新系统的调整报告与维护申请。系统正是在不断的维护、评价过程中逐步完善起来的。一旦一般的维护工作不能满足要求，一个新的更加先进的系统的开发过程就又要开始了。

本章小结

系统实施是系统开发的最后阶段，也是将前一阶段的设计结果最终在计算机系统上实现的阶段，这一阶段的任务包括：购置和安装计算机网络系统、进行程序设计、进行系统测试与调试、负责新旧系统的转换等。购置和安装计算机网络系统主要是根据系统目标做好设备选型。程序设计中，应采用模块化程序结构和结构化程序设计方法，提高程序的可靠性、可维护性、可理解性和开发效率，同时，还应注意选择合适的开发工具，为系统开发提供一个良好的操作环境。系统测试是保证系统质量与可靠性的最后关口，也是对整个系统开发过程包括系统分析、系统设计和系统实现的最终审查，系统测试的目的就是给系统找错，选择合适的测试用例是测试工作成功的关键。系统转换是系统实施的最后阶段，一般有直接转换、平行转换、分段转换三种方式，在实际应用中要根据具体情况灵活运用。系统投入运行后，要定期对系统的功能、软硬件性能、应用状况和系统的经济效益进行评价，以检查系统是否达到预期目标并提出今后的发展方向。

思考练习

1. 简述系统实施的地位和作用。
2. 在系统实施之前要对各类人员进行哪些方面的培训工作？
3. 一段高质量的程序应当符合哪些基本要求？
4. 程序设计一般应遵循怎样的步骤？
5. 系统测试的意义是什么？测试与调试有什么区别？
6. 简述白盒测试和黑盒测试的异同点。
7. 系统转换有哪几种方式？各有什么特点？
8. 系统评价应包括哪些方面的内容？