CHAPTER

第 3 章 结构化设计方法

在编程期间编码、测试、调试发生的问题往往是由于先行工程中一些不能确定的数据造成的，加上编程期间一般不会有充足时间来处理这些问题，所以希望预先在设计阶段就对各种可能出现的问题进行考虑。因此，设计工作是否能做得很完备，左右着软件的综合质量。迄今已有很多实用的技术提案。设计法为从客户需求到编程的过程中，如何实现设计工作的正确且高效性的方法论。本章将简介结构化设计法。

3.1 结构化设计

3.1.1 概要

首先，要明白本设计法的要求。

(1) 以模块分割为主体。

(2) 作为设计的图示说明，需要使用功能表和模块结构图两个种类。

(3) 需要按如下设计步骤进行。

[步骤 1]分析系统(程序)的全体(处理中心的定义)。

[步骤 2]数据处理的解析(处理动作的定义)。

[步骤 3]数据的变换分析(功能表的作成)。

[步骤 4]模块的选择(按照模块评价基准来完成模块结构图)。

(4) 为了判定模块的好坏，需要理解和使用模块的评价基准和结构准则等。

本设计法的目标是将模块进行有效分割，但更困难的是模块的定义。模块的定义如表 3-1 所示。在这里，需理解各程序所具有的特性，并根据其特性来编程。模块数应在一定基准值以下来设定物理的条件。根据经验，按各模块的目的来分成适于开发的各个模块单位。

表 3-1 模块的定义

模块：是具有以下特征的程序语句的集合体。 ① 多个程序，作为具有某种意义的群集而统一在一起； ② 程序语句群由边界区分符(例如 Begin 语句和 End 语句)来分隔开； ③ 程序语句群根据其名称(模块名)，能统一供其他程序参照引用。

3.1.2 设计图的说明

下面对结构设计法常用的功能图和层次结构图加以说明。

1. 功能图

功能图用来表现功能，一边用功能处理数据，一边描述变化的过程，这被称为追踪数据流程。

图 3-1 为功能图的画法。

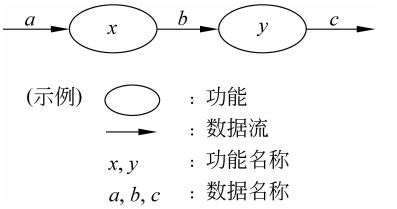


图 3-1 功能图的画法

箭头(→)方向表明了顺序关系。在功能图中简明地记录了功能。图 3-1 数据流程的读法，例如“数据 a 通过功能 x 转换成数据 b；继续通过功能 y 转换成数据 c”。功能图 3-1 中，只有一个输入和输出，图 3-2 中则有多个输入和输出。另外，要注意功能的简化，如不简化，一个功能就可以占满一页，所以必须注意小型化。

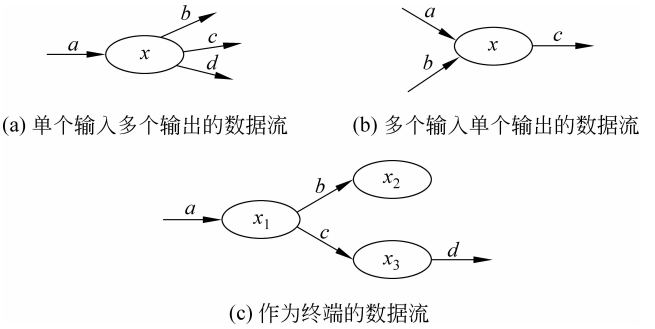
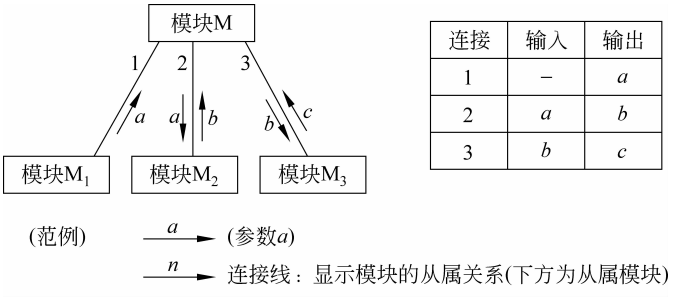


图 3-2 功能图的描绘方法

2. 层次结构图

图 3-3 为各模块之间的从属关系以及参数移动方向的例子，层次结构图由表示模块的长方形和连接模块，以及表示模块之间从属关系的连接线组成，在下方的模块称为从属模块。



连接	输入	输出
1	–	a
2	a	b
3	b	c

图 3-3 模块的层次结构图

说明：

(1) 模块 M 被分为 M1、M2、M3 的功能，模块 M1、M2、M3 作为 M 的从属模块。

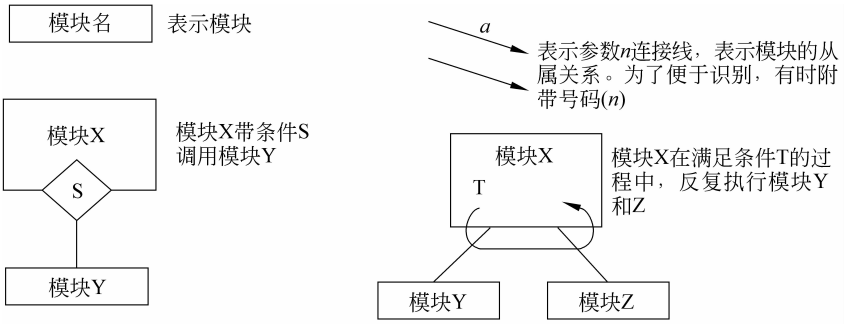
(2) M 与 M1、M2、M3 之间，参数 a、b、c 被传递。

(3) 参数的输入输出，采用从属模块的立场来体现。

为了识别连接线可以对其加上编号，编号如果对解释说明无用也可以省略。

一般如果传递参数越多越难写入图中，最好使用参数表。如果使用了参数表，上述的连接线识别编号就不能省略。如果能在图上直接标注，当然容易理解，可是修正和维护的难度较大，因而选择什么样的方式应采取谨慎态度。通常，在设计的初期阶段使用参数表，如果参数不多，并且设计变更带来的修正的次数也不多(比如，到了设计审查阶段)，作为最终核对方式，可将参数画在图中。

如图 3-4 所示是使用较高的具有代表性的层次结构图的记法。



参数表

连接号码	入(IN)	出(OUT)
1		O_1
2	I_1	O_2
⋮	⋮	⋮

图 3-4　层次结构图的记法

条件调用模块的条件(S)可能会有多个存在。另外，反复和带条件调用，会出现在同一个模块中。

参数表的输入、输出，采用从属模块的立场来表示。也就是说，面向从属模块的箭头为入(IN)(图 3-3 中的从属模块 M_2 的 a 为 IN)。

3.1.3　设计步骤

关于结构化设计法的设计步骤，以下做一简要说明。

1. 处理中心的定义

需调查进入系统的所有输入。因此，首先要明确分析系统全体的处理中心、作用，即需要定义以下内容。

① 接受处理。

② 决定处理形式。

③ 向不同类型的处理地点送出处理。

④ 完成各种数据处理。

2. 数据处理

对数据进行分析，规定处理这些数据的模块。

① 识别数据来自何处。

② 组建以处理为中心的全体系统。

③ 识别处理的形式，并定义处理的动向。

④ 定义处理各类数据所对应的模块。

⑤ 对④的数据处理模块进行分析，可以细分出下方模块。

3. 数据的变换分析

变换分析是基于数据流程的分析，其具体的实施方法描述如下。

① 整理功能。整理、识别系统(程序)的功能，简洁地表现和说明。

② 制作功能图。追踪在系统中被变换数据，按变换过程中的顺序箭头来连接关联的功能。

③ 把功能分成组。将第一层的从属模块分为输入处理部分、变换部分、输出处理部分 3 部分，各部分再顺次细分，随着逐步细化，要注意总结各个模块功能之间的关联性。

④ 把功能组分成模块的层次结构。从系统供给数据源的方向来考虑，称输入处理部分为源模块；变换部分具有将输入变换成输出的功能，所以称为变换模块；最后的输出处理部分，接受变换后的数据，因具有吸收的功能而被称为吸收模块。整个操作进展过程，应参考功能中所写的功能说明，给模块标注上适当的名字，以上所述如图 3-5 所示。

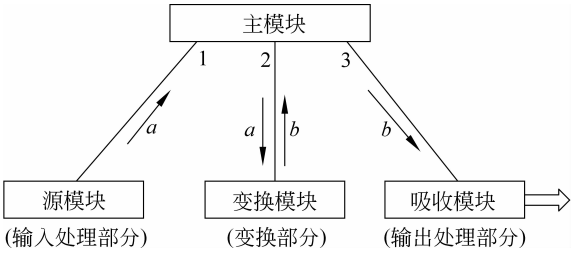


图 3-5　第 1 层的从属模块结构

⑤ 继续细分模块直到满足停止条件后就结束设计。停止条件是指再不可细分的模块。如果不满足停止条件则返回①，随着模块被分得越来越细，同时还需给各模块定义合适的名称。

4. 挑选模块

在模块层次结构的制作过程中，为了制作出更好的模块，在进行了整体结构布局后，

还要再精练挑选模块。

评价模块的好坏需要使用评价标准，模块的独立程度可以通过耦合和内聚来判定，如表 3-2 所示。

表 3-2　模块的评价标准

	耦　合	特　　征
强 弱	外部通用耦合	变量不用通过参数，就可在多个模块中使用，而对不应参照的模块则不能使用这些变量
	控制耦合	指把开关变量等作为参数传递的模块。由开关变量决定能否实行模块时，接收端的模块就变得复杂，独立性变差
	数据耦合	把数据作为参数来传递(不是控制数据)。因此，独立性高，能把调用的模块作为黑盒使用
	内　聚	特　　征
弱 强	逻辑内聚	使 1 个模块具有多个功能，多参数，多控制成分(开关等)
	时间内聚	从逻辑上的内聚上，附带了时间所决定的模块。将初始设定时的 open 语句，结束时的 close 语句等，统一写在一处
	功能内聚	1 个模块只执行 1 个功能

内聚是用来表示模块内的各个部分的关联程度，耦合是表示各模块间关联的强弱的尺度。一般说来，高内聚，低耦合标志着对模块的评价度高，但定量评价还是比较困难的。

以下将简单介绍几种典型的结构化设计方法。

3.2　数据结构主导设计法

数据结构的主导设计法有 Warnier 法和 Jackson 法。

Warnier 法作为方法论发表后，由于开发者具有长年实践经验，所以这个方法具有很高实用价值，受到很高的评价，数年后被列入法国软件行业招人、求职广告中所必需的资格条件。

1. Warnier 法的三个步骤

(1) 逻辑设计：分析数据的层次结构，处理模块化，把握数据的流程。具体操作如下。

① 作成输出数据结构图。

② 作成输入数据结构图。

③ 必要时使用真值表来明确处理条件。

(2) 详细设计：将处理该业务必要的处理命令和这些命令所使用的场合(模块)相互关联起来。具体操作如下。

① 将处理命令的种类列成一览表。

② 将一览表的命令分配到各程序内的相应模块中。

③ 作成各模块分开的处理命令的一览表(编程表)。

(3) 编程：根据编程表及程序语言来编码。

从机械性考虑，编程就成为实际处理命令的顺序集合，在这个阶段需规定各自的处理内容。

Warnier 法与其他设计法有所不同，Warnier 法从应用直到编码阶段，而其他方法不会涉及步骤(3)。

以上步骤(1)～步骤(3)的设计顺序如图 3-6 所示。

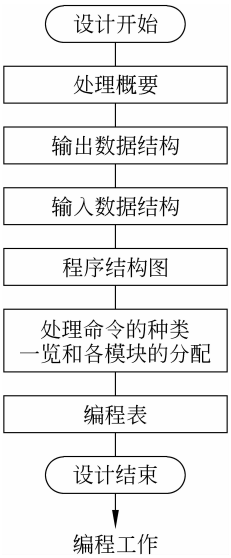


图 3-6　Warnier 法的设计顺序

2. Warnier 法的特征

Warnier 法有以下特征。

(1) 有层次性数据结构，就是将程序结构设计当成一个平台，将输入数据和输出数据按一定的细分法则作成层次型的数据结构。

(2) 引入制作结构图的考虑方法，进行数据和业务的分析。

3.3　系统的层次分割法

针对特别复杂系统的设计，将其分成程序模块。为了解决这个问题，采用阶段分割系统的方法，其中之一就是层次分割法，对于这个分割方法，各个功能的包含关系是关注的焦点。

首先，整体把握系统。接着，考察是否能将系统全体功能分成若干个子系统。

子系统的分割有多种方法。从中找出最简单相互连接的分割方法。接着就是整理各子系统的输入输出、数据结构、数据的输入输出媒体等外部关系。确认是否确保了其作为子系统的独立性。

在明确各子系统作为独立单元的功能的同时，还要考察各子系统还有没有继续分割的可能性。

关于再分割的步骤，只是将最初子系统的分割方法重复使用，再分割的功能单位再一次在子系统中被定位。

反复进行上述操作，最终将每个功能单元的编码量控制在 200～300 行以下，这就是把程序模块具体化。

这里可以用图示简明地描述包含关系，图 3-7 为 Sot Tech 提案的 SADT(Structured Analysis and Design Technique)方法。

从以上说明可知，使用系统层次分割法将系统分割成编程模块实体，最终成了功能单位。

需要注意的是，在导出最终功能单位过程中的中间功能单位(子系统、子子系统)。更重要的是要平滑而不是勉强地分出最终的功能单位(模块)。

由于程序员非常关心作为编程实体的最终功能单位(模块)，所以试图从系统一下子分成最终功能模块。根据经验这样易诱发各种各样的设计错误，所以提倡一段段地细分

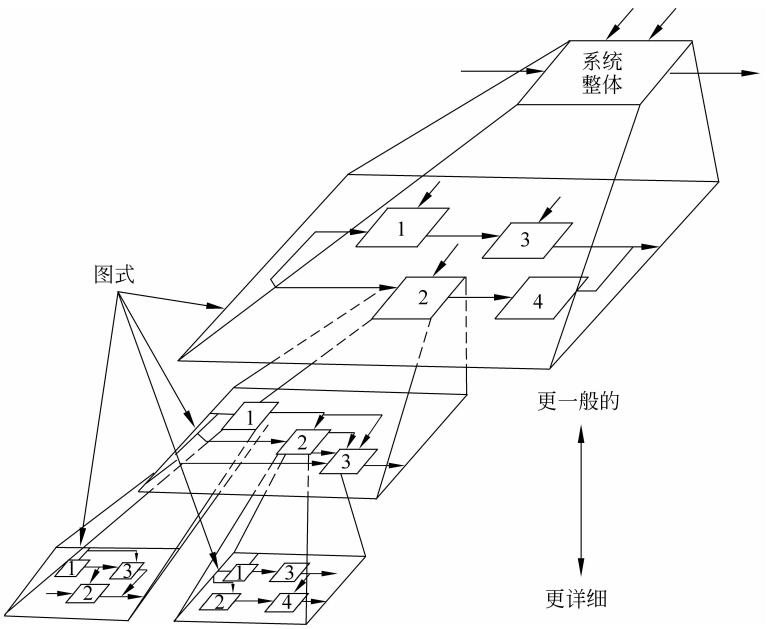


图 3-7　系统的分层划分法的例子 SADT

化的方法论。SADT 设计方法的特征是重视设计过程。

以前往往从系统全体一下子分解到程序模块。以往的设计和功能分层化的设计对比如图 3-8 所示。

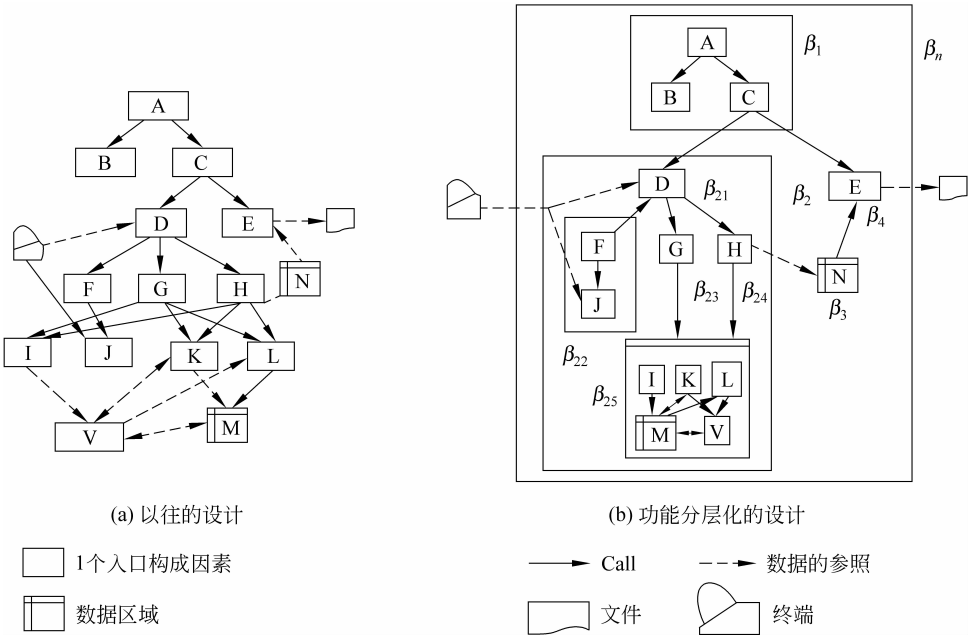


图 3-8　以往的设计和功能分层化的设计

图 3-8(a)为过去的设计法，设计人员基于其所具有的高技术水平，从系统全体功能向模块构成图直线分解。图 3-8(b)则用功能层次化设计法将系统分成如图 3-9 所示的最终模块构成图，其中至少有 6 个中间模块。

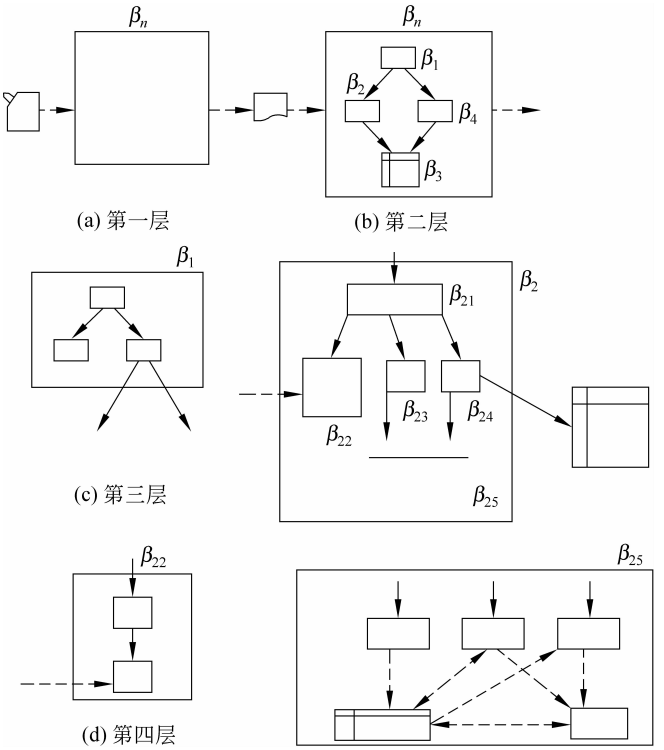


图 3-9　分层化的过程

虽然手续比较复杂，加之中间模块会不会对下阶层产生影响都需明确，但如将该部分修正为如图 3-8(b)所示的最终功能单位，由于工作量较少，从进行分割的角度来讲，还是有利的。

相比之下，图 3-8(a)所示的情况，如果要更改包括追加功能，执行者必须从最初开始重新设计，花的时间不少于最初的设计人员。图 3-9 说明了分层化的过程。

3.4　Top-Down 设计法

在软件工程领域里，Top-Down 的测试方法和 Bottom-Up 的测试方法是非常普遍的两种方法。在设计法中，虽然存在着 Top-Down 和 Bottom-Up 设计法，但比较起来，Bottom-Up 设计法不太为人所知。

一般认为用 Top-Down 来表示分解，而 Bottom-Up 来表示统合。也就是说，Top-Down 设计方法“分解程序成为部分程序”，而 Bottom-Up 设计法是将已经定义的程序文件进行统合，从部分程序上升到全体程序的设计法。

Top-Down 设计法首先选择程序主要素(主程序)来定位最上层的模块;然后,根据调用包含在主程序中的子程序,来将主程序分解成多个子程序,即主程序一个个调用子程序,完成各种处理,以下再对子程序,子子程序……进行分解,直到没有可分解的子程序为止。

图 3-10 为这样完成的模块结构概念图。从制作过程推测,比起定位最上位的模块结构来说,还不如控制各下位模块的实行顺序。以这种意义,在大规模系统中,最上层的模块称为主控制模块,子程序是属于下层的程序,是具有控制内部处理和下层子程序功能的模块。

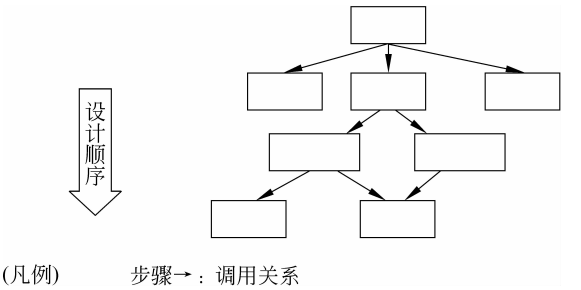


图 3-10 根据 Top-Down 设计方法的模块结构概念图

虽然从上层(全体)到下层(部分)细分化的工作流程,与前节的系统层次分割法相类似,但有一点不同就是层次分割法从上层分割到中层,中层并不作为程序模块的实体,而基于从上到下设计法中间水平的构成要素,连接着上层和下层,作为分担处理的一部分程序模块实体存在。

Top-Down 设计法可以看到设计对象的全局,特别是可以推定主线的处理流程,能够有效地表现序列功能程序。

Top-Down 的缺点是有点偏重控制程序,到子程序不能分解,分解得太细,子程序为分担内部处理和其他模块相连接变得太复杂。

习题

1. 试说明结构化设计的基本思路与步骤。
2. 试比较 Top-Down 和 Bottom-Up 的设计方法。
3. 系统的层次分割方法适合于简单系统吗?
4. 试简单描述 Warnier 法的特征。