TECHNOLOGY AND METHOD OF SOFTWARE ENGINEERING FOR PRACTICE

软件工程实用技术与方法

张湘金　陈石雄　赵滇明　主编

云南科技出版社

（滇）新登字 04 号

本书受云南省学术著作出版基金管理委员会筹备组资助

责任编辑：刘　川　单沛尧
封面设计：薛靖民

软件工程实用技术与方法

云南科技出版社出版发行　（昆明市书林街 100 号）
云南省计算中心科技彩印厂印制

开本：787×1092 1／16　印张：12.75　字数：27 万
1994 年 5 月第 1 版　1994 年 5 月第 1 次印刷
印数：3000

ISNB 7-5416-0425-9／TN・2　定价：20.00元

前　言

多年来，软件工作者一直都关注着软件开发工程化的进程。从60年代末期以来，为了摆脱由于软件开发中出现的高成本、低品质和工期延误而产生的所谓“软件危机”(Software crisis)困扰，软件界作出了许多的努力，取得了瞩目的成效。但是软件开发工程化的进展却不尽如人意，软件工程还没有形成一门完整的“工程科学”。在硬件技术突飞猛进和应用需求与日俱增的强大压力下，软件工程面临严峻的挑战，如何才能大幅度地提高软件生产效率、改善软件产品质量，仍然是一个无法回避的问题。

由于我们从事软件开发工作的需要，迫使我们不能不面对这个问题。我们曾作过许多尝试，也曾采用过其它的一些开发方式，但成效并不显著。几年来，我们同日本筑波大学信息工程研究室的宇都宫公训先生等合作进行了一些研究工作，收到了令人比较满意的效果。我们的主要工作是：按照我们的实际情况，综合地利用软件开发方法和技术成果，并相应地研究和采用一组支持工具，建立一套适应软件开发过程的基本工艺规范，如果用目前比较时髦的说法叫做进行技术、方法和工具的集成。一边研究，一边把它应用到我们的具体软件开发项目中去进行实践检验。这里，要特别说明的是，我们注重的主要是实用性，即能适合我们自己在软件开发过程中的实际需求，从这点出发去构筑我们的规范，并通过我们的具体实践去逐步补充和完善。

在这一过程中，我们不断地征求各方面的意见，也不断地吸收一些软件开发的新技术。1991年4月，我们在日本筑波大学同宇都宫公训先生进行了再次深入讨论，并与日本的一些同行进行了交流。回国后又再次进行了修改。终于，在各方面的鼓励和支持下，我们才下定决心，不怕浅陋，交稿付印，以祈各位同行赐教。

本书共分为四章．第一章基本思想和概要：阐述了采用基于结构化的技术与方法的基本观点；分析了目前流行的五种软件开发模式的特点，以及基于“瀑布”模式并适当地使用“原型”模式的思想；概述了本书介绍的各种结构化技术与方法，及其如何在软件工程各阶段中适宜地选用。第二章结构化系统开发过程：采用数据流图方法描述结构化系统开发，详细地展现了全过程中各个作业层次的数据流图。从而以它为示例，以说明如何采用数据流图类似地描述对象系统的方法；以结构化系统开发作业的数据流图中出现的数据项目为实例，介绍了编写数据字典的方法；并对结构化系统开发各阶段的数据流图中，每个过程的具体作业(最下层作业)按步骤进行了详细地说明。第三章系统分析与设计技术：叙述了DeMarco方法的基本概念，用于结构化系统分析时描述文档的符号与规则，使用注意事项以及实例说明等；介绍了用实体关系图(ERD)的方法进行数据库概念结构设计的基本方法。在“复合／结构化设计”与“Jackson结构化设计”两节中，分别介绍了基于数据流和基于数据结构的分析，从而导出系统的总体结构与程序结构的不同方法。并通过实例介绍Jackson方法，在输入与输出数据结构一致的情况下，可直接根据设计产生的“Jackson结构图解”和“Jackson概要逻辑”编写出源程序代码。第四章系统开发测试技术：介绍了系统开发的各个阶段中如何进行人工测试(Walkthrough)和审查(Inspection)，同时介绍了自顶向下的测试等方法。

由于是共同编写，同时是一边研究，一边实践，一边不断完善，而一些支持工具也无法在这样的篇幅中作一一介绍，因此在阅读本书时或许会出现一些不甚连贯之处，恭请各位原谅。

我们衷心地感谢日本筑波大学宇都宫公训先生的悉心指导，感谢姜世杰博士和在日本的朋友们的真诚帮助。云南省科委刘诗嵩、林文蓝副主任、国际合作处林兆驹处长、孙蓝蓝副处长、计划处刘昌荣处长、云南省引进国外智力办公室魏民主任、辛志晖等领导和同志对这一中外合作研究项目曾给以很大支持和关心，对本书的出版提供了许多帮助；曾在云南省计算中心工作过的丁志强、周寿章、刘建宾以及我单位的张兵、李辉等同志曾参与了部分工作，张瑾、张衍辉、羊红兵、孙虹、文珊等同志为本书的出版做了许多具体工作，在此谨表由衷谢忱。

作　者

1993 年

目　　录

第一章　基本思想和概要

1.1　引　　言

软件开发工程化的基本目标，按照我们的理解，就是借助于“工程化”的开发手段以提高软件产品的生产效率、产品质量、降低开发成本，从而较为有效地克服所谓“软件危机”，以满足日益增长的社会需求。

近 30 年来，为达到这个目标，人们进行了多方面的研究和探索，在理论和实践方面都取得了可喜的进展。概括起来，主要沿着两个方向进行。一是以形式化技术和人工智能技术为基础，实现软件产品开发的自动化、智能化。形式化技术就是利用数学和逻辑系统的严密科学方法，以实现从软件产品的规格说明直至代码生成的所谓“完全形式化演变”。而这里的智能化则是借助于人工智能和认知科学的成就，为软件产品的开发提供智能化的环境。二是基于传统的软件开发方法上的工具和开发环境的变革，以提高软件开发自动化水平。从某种意义上说，后者比较接近广大软件工作者熟悉的工作方式，比较容易见到效果。因而相对地从事这方面工作的人较多，比如近十多年来计算机辅助软件工程，即 case 技术，就吸引了许多软件工作者的关注。

我们认为，在 60 年代末期提出的结构化分析与设计的基本思想，经历了二十多年的发展与实践已日趋成熟，形成了一个比较完善的方法论。正如本书将向读者提供的，在软件生命周期各阶段中可采用的各种技术与方法那样，不仅具有完整的方法学基础，并在研究与实用中逐渐完善相应的系统开发与运行环境及其支持工具，成为国内外软件研究与生产部门采用的主要方法，其技术方法体系也正在不断地得到充实与发展，是相对成熟和比较有把握的软件开发手段。

由于计算机应用领域与范围的扩大，以及多媒体、分布式、并行处理等技术的发展，使得系统处理的数据规模越来越大，处理程序越来越复杂，适应业务上的动态变化要求也越来越高。同时，计算机系统硬件性能及处理速度的不断提高，要求从软件上改善用户界面以提高应用效率。这对软件开发提出了两个比较突出的问题：一是如何提高人机接口的效率；二是如何适应不断增长的系统复杂性。于是，80 年代末再度提出了“面向对象程序设计”(Object Oriented Programming，简称 OOP)技术。近年来，国内外一些研究部门、高等学校、软件厂商在这一领域投入了不少研究开发力量，从程序设计语言直到操作系统、数据库、用户接口、专家系统以及工作站等软件领域，均开展了“面向对象技术”的应用研究与开发，目前已推出了一些新的产品，但是据专家认为使用中尚有一些问题有待解决。众所周知，面向对象程序设计方法是着眼于现实世界中的事物(即对象)解决问题的一种方法，即是把许多事物(对象)组成的现实世界模型化的程序设计方法。常用的面向数据、面向功能的设计方法是着重于解决算法的实现过程，即先考虑好处理程序后再往里填数据(以程序为主)。而“面向对象”则是把每个对象的数据和对数据进行的操作(即功能)结合起来考虑，即把数据和算法合为一体，并以数据为主体，使得数据本身带有方法，只要对数据给予指示，即可通过对象

之间交换消息而进行自动处理的操作。因此，面向对象程序设计不是以控制为中心，而是以事物(对象)的行为为中心来考虑计算机上的处理体系。它不仅体现了结构化程序设计的特点，而且为程序员提供一种分析与解决程序设计的新方法。于是，试图以这种方法解决大规模程序控制的复杂性问题，以及将“字符用户接口”(CUI)改变为“图形用户接口”(GUI)，从而解决人机接口的效率问题。然而，要在整个软件工程中采用面向对象技术，必须有一整套适应于软件开发各阶段应用的“面向对象技术”方法。目前，较为流行的方法是面向对象程序设计(OOP)，而诸如面向对象的系统分析(OOA)、设计(OOD)、测试(OOT)、数据库(OODB)以及开发工具与环境(OOCASE)等等的技术方法，尚有待于进一步研究并经实际验证的成熟过程。就目前的状况来说，必须解决四个方面的问题：完善方法论；研制开发工具和类库；扩充运行环境；充实“面向对象”化的语言。否则，面向对象技术用于软件工程的基础太薄弱。

我们认为，在我们所进行的工作中，使用多数开发人员相对比较熟悉的结构化方式比较有把握，用这种方法对中、小规模的软件系统的开发，已有过大量的实践并取得了效果。因此，在我们软件产品工程化实践与研究中，首先注意形成一套建立在这种传统的、应用最广也相对较为成熟的软件开发方式基础上的开发规范，并辅以相应的工作，结合本部门的具体开发与维护经验，遵循实施，是非常必要的。这就是我们整个工作的基本思想。在这个思想的指导下，我们工作的主要依据和做法的特点是：

1. 瀑布式的开发模式
2. 向形式化的渐近演变
3. 技术与方法的集合体
4. 结构化技术与方法
5. DeMarco 方法
6. 自动化开发工具
7. 向部件化和重用作逐渐演变

众所周知，对任何类型的软件都很有效的开发与维护方法是不存在的。一般地说，技术与方法的通用性越高，则应用于每一个具体领域时效果越差。如果不对开发和维护的软件类型作某种程度的限制，进行规范化的效果就不大。因此，本书介绍的技术与方法主要针对 MIS、DSS 及 OA 的开发。

下面就目前流行的几种软件开发模式作一个概括的分析。

1.2 软件开发模式

目前国内外使用的软件开发模式，可归纳为：“瀑布”方式；分解开发方式；“清洁房间”方式；原型方式；部件化／再利用方式等等，现分别作简略说明。

1. 瀑布式模式（Water Fall Paradigm）

这种方式又称传统的软件生存期模型，它是将软件的生存周期划分为几个阶段，如象瀑布式地，逐个阶段依次进行实施，如图 1.2-1 所示。

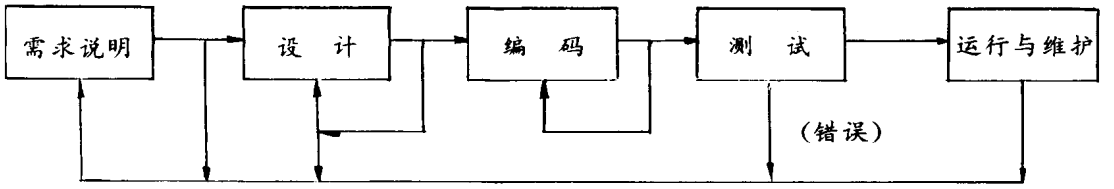

图 1.2-1 瀑布式模型

这种模型的特点是：各个开发阶段的目标明确，要求严格；各阶段均有具体要求的文档格式，便于工程化组织和管理；每个阶段发现的问题，在本阶段不能解决的，必须返回到需求说明阶段重头做起。因此，这种模式的最大缺点是：返工工作量大，开发周期长；其次是所用的文档较复杂，且缺乏相应的支持工具；不适合实时软件的开发。因此，促使人们探求了新的开发模式。

按瀑布模式建立的典型标准，如日立公司的 HIPACE, 富士通公司的 SDEM 等。

2. 分解开发模式

这种方式在需求定义之后，先确定软件的基本结构，然后按软件的任务性质划分为基本处理、主要处理、特殊处理以及错误处理等四个基本的开发内容。

如图 1.2-2 所示，每个基本开发内容可分步进行，其每一步骤均按瀑布式的方法去做。这样，范围划小了，使其工作量相应减少；当用户需求有变化时，可区别加入其中的某一个基本任务去处理；简化了问题，其返工量相应得到减少；在一定程度上，提高了开发效率。诚然，每个基本内容的划分及工效的提高，很大程度上还取决于开发者的经验。显然，此种模式基本上仍属于瀑布式的范畴，不可能有效地克服瀑布式的弱点。

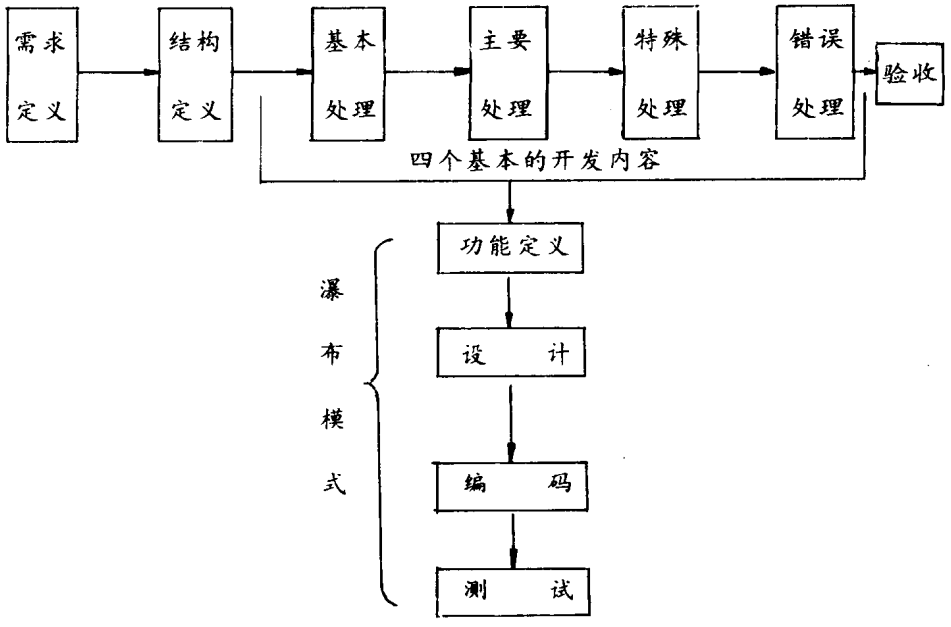

图 1.2-2 分解开发模式

3. 清洁房间模式（Clean Room Paradigm）

这是 H.Mills 提出的方法（如图 1.2-3 所示），顾名思意，所谓“清洁房间”即是不允许每个房间留下任何拉圾，对软件开发来讲，要求每一步都不能存在错误。为此，必须采用适当的技术与方法做保证，才能达到高可靠性。而且要求每一步都做得细，必然会降低软件生产效率。

这种模式的关键技术与方法是：结构化程序设计+抽象数据型+验证（Structured Programming+Abstract Data Type+Verification）。

为保证每个步骤的准确，必须用结构化程序设计的方法（目前认为最好的结构化语言是 Ada，或是与 Ada 接近的 CDL—Common Design Languge），使数据高度地抽象化，以适应数学抽象的形式化验证，且每一个步骤都进行验证。验证的方法有两种：一种是形式化验证（即严密的数学验证），它要求有较高的理论和技术基础，目前还不能完全实现（据称现在 IBM 公司，在编码部分实现了 30%的形式化验证）；另一种是非形式化验证（即说明输入与输出关系的验证），一般采用逻辑上严密的自然语言描述，以求实现验证的准确。

为保证系统开发的高可靠性，IBM 公司在设计方法上采用了过程操作型的设计，其基本特点是：

（1）结构化程序设计（即阶段的详细化）

（2）不用 GOTO 语句

（3）状态机模型

（4）信息隐蔽

（5）验证

（6）测试

同时采用相应的开发工具，如 IBM 在 SNA 里用的 FAPL 工具，能自动变换生成 PL／I 语言程序。

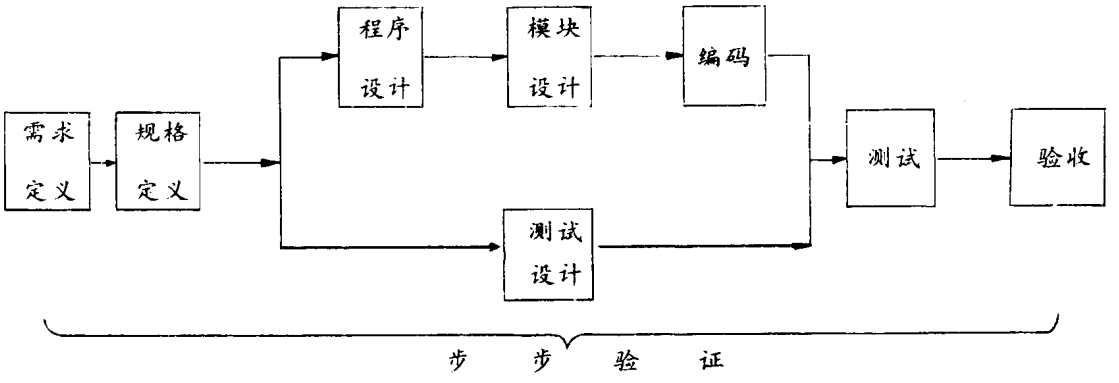

图 1.2-3 “Clean Room”模式的基本过程

4. 原型开发模式（Prototyping Paradigm）

这种方式又称样机开发方式，如图 1.2-4。其特点是首先研制样机（原型），原型系统做好后，请用户试用和评价，若用户不满意可及时进行修改，直到用户满意后才进行正式的

系统开发。因此，该模式最大的优点是：适合于用户需求不明确的情况，原型可促使用户从中考虑自己的要求是否满足业务需要，开发者可在开发工具的支持下作快速的修改，用户可以得到满意的软件。这是一种比较理想的方式。但是，只有在相应的开发工具支持下，方能显示原型制作的效率和成功性。

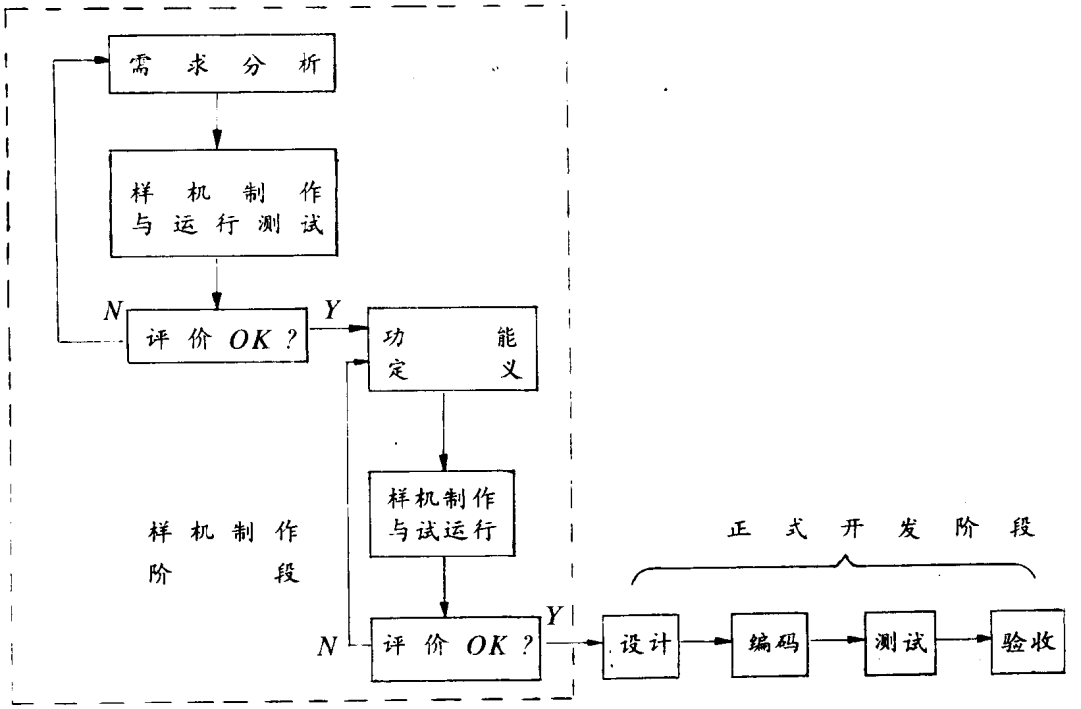

图 1.2-4　Prototyping 开发模式

在原型制作阶段要有专门费用，为了作出更好的产品，通常不采用原型软件作为产品的部分，而是把它扔掉，要重新制作产品又要开支一笔费用。所以，原型模式的缺点是：成本较高。此外，由于要求相应的开发支持工具，需要有一定的技术装备投资，因而限制了它的使用与发展。

5. 部件化／再利用的开发模式

这种方式，旨在开发具有各种一般性功能的软件模块，并考虑其适应各种界面的接口规格，将它们组成软件重用库，进而供软件开发时利用。这种再利用技术（或称重用技术 Reusing）的主要优点是：可减少软件生产中的重复开发，避免耗费大量重复劳动（据专家估计，这种重复工作量在一个程序中高达 80%以上），提高开发效率，缩短开发周期，降低成本。这种模式的组成结构如图 1.2-5 所示，其开发阶段与瀑布模式大体一致，所不同的是在设计至编码阶段中增加一个部件选择阶段，在此过程中，亦可开发新的部件，扩充到重用部件库中。诚然，重用部件库不仅要便于选择使用，而且还应具有允许扩充，积累其成员的性能。虽然，这种开发模式被认为是一种经济和高效的方式，可是重用部件库的建立必然是较长期大量劳动的产物，其中尚有许多技术问题有待研究解决。

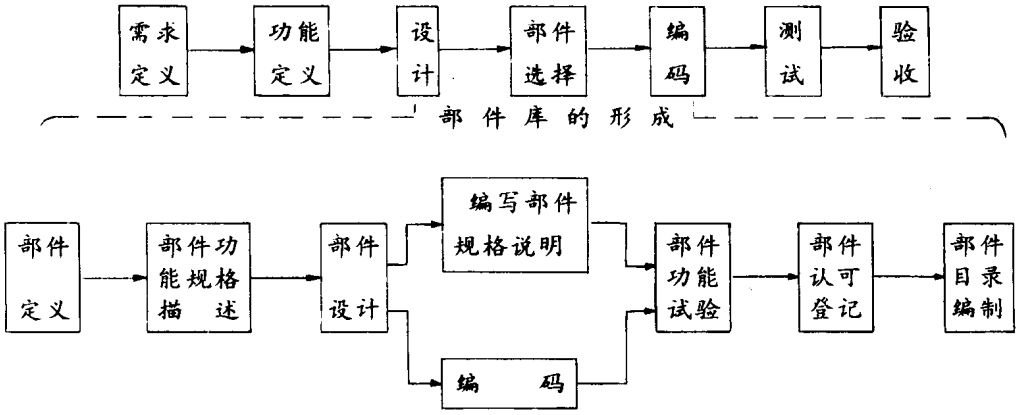

图 1.2-5　部件化／再利用开发模式

通常，软件重用可在两个级别上重用：即在源程序级上重用(指重用程序是以各种源语言形式存库)；在目标程序级上重用(指保存重用程序是经过编译的目标程序)。在源程序级重用时，新编程序与重用程序在变量名、类型名、函数名等方面如何协调才能引用，如何建立抽象数据类型库，以提高软件重用性等问题尚有待研究。

综上所述可知，分解开发模式、清洁房间模式，都是瀑布式模式的扩充和变形，部件化／再利用模式，可看成是瀑布式模式的扩充，而独具特色的是原型开发模式。此外尚有其它一些模式，如增量式(Incremental)、螺旋式(Spiral)等开发模式，其基本出发点都是试图克服瀑布式开发模式的不足，此处不予赘述。

瀑布式模式是自提倡软件工程学以来的开发模式，有一定的应用价值。虽然它存在着返工工作量大、开发周期长等主要缺点。然而，我们认为软件开发和维护作为一个工程，必然以瀑布式模式为其基础，只不过是按具体情况划分的开发阶段不同，或采用的技术与方法各具特色而已。原型开发模式可以被认为是与瀑布式模式并存的。由于原型开发是在制作正式的产品之前，先作一个原型，然后通过这个原型的运行，对其规格化和设计的正确性及其实现可能性进行验证。避免到测试阶段才发现规格化和设计的错误，可节省大量的修改费用。所以，原型模式的出发点是尽可能早地排除这些错误，尤其在用户接口、外部接口比较重要与复杂的场合较为有效。但是，仍存在制作原型的成本高等缺点。如能使用有效的原型开发支持工具，这一缺点是可以克服的。但是，应对对象加以限制，才能获得效果好的支持工具。

基于上述分析，本书的出发点是：依据瀑布式模式进行产品的开发，尽量使调查、分析、设计阶段所产生的文档对用户易于理解，并通过复审的措施，尽可能地减少规格化和设计中的错误。对于原型方式，在确定用户接口的限定范围内，用较少的费用能制作出原型的情况下，也可以采用。并要尽量提高自动化以及部件化／再利用的程度。自动化与部件化／再利用程度越高，那么，在测试中发现错误的修改费用越低，而且维护费用也会下降。我们认为维护实际上是通过对于现已存在的系统开发文档及编码等的再利用，以开发好的系统。

原型开发模式并不是与瀑布式模式相对立的东西，而不过是在瀑布式模式中的某些部分，对其加以扩充，如果在瀑布式模式里提高了自动化程度和部件化再利用程度，那么导入

了原型方式思想的软件开发会自然地成为现实。

1.3 技术与方法概述

在这里我们所考虑的技法，几乎都是基于结构化的思想。同时要注意到，软件开发周期各阶段所采用的技法是因对象而异的，其选择原则是以提高开发维护效率和产品质量为主。如果能针对研究对象的情况合理选择，把适宜的技法组合起来加以运用，则能发挥出应有的效果。就目前我们所进行的软件开发来说，大部分是以数据库为中心的 MIS、DSS 及 OA 系统，现将其各开发阶段常用的技法作如下简介，以便读者参考。

1. 需求分析、定义及结构设计阶段

为作到准确的需求定义，对用户的现行系统状况进行调查分析时，必须采用用户易于看懂和理解的表达方式进行描述，通常采用 DeMarco 方法较为合适。

(1) DeMarco 方法：是一种结构化分析方法(Structured Analysis，简称 SA)，它以数据流图(Data Flow Diagram，简称 DFD)为主要描述工具。DFD 图应用较为普遍，具有描述符号少简单易画，比较直观易懂等特点，可以说是一种容易被用户接受的语言。

在需求分析与定义的整个阶段中，都用 DFD 作为主要描述工具，从“现行系统物理模型”的描述开始，到抽象出的“现行系统逻辑模型”，直到经需求分析构想出的“新系统逻辑模型”到“新系统的物理模型”均用 DFD 描述。并且按由顶向下、逐层分解细化的原则，所实施的系统分析也用 DFD 实现。用 DFD 描述系统模型各个层次的结构图作为系统说明书中的主要文档。此外，在系统分析中，同时产生的主要文档还有数据字典(DD-Data Dictionary)，用它对各个层次 DFD 中的有关构成要素(如每个数据流名、文件名、加工名等)作确切的定义。DFD 与 DD 二者的密切结合是系统分析(需求定义)阶段的结果产物——系统说明书(或称结构化说明书)。

结构设计(也称总体设计或初步设计)的任务是以 DFD 图为依据，导出整个应用系统的功能模块结构图，即模块的层次划分。在此阶段，为描述底层的基本功能模块，确定其 I/O 数据流之间的变换规则(即加工逻辑)，用结构化语言书写的小说明书(Mini Specification—简称 MS)，以及因逻辑因素和逻辑状态复杂，用结构化语言难于表达清晰时，用“判定表”或“判定树”所作的辅助描述，是此阶段产生的主要文档。

我们认为对于小型应用系统，根据开发者的经验，可将总体设计阶段合并到需求定义阶段一次完成。此外，根据对象的结构复杂程度，分析和设计阶段还可结合采用“实体关系图法”和“状态迁移图法”。

(2) 实体关系图法(Entity Relation Diagram—简称 ERD)

对数据关系较为复杂的系统，为有效地分析与确定数据结构，减少数据的冗余量，采用 ERD 法进行系统分析和设计是较为优越的。尤其是对 DFD 不能描述清楚的问题，比如，企业经营管理系统，内部实体与实体之间复杂关系的确定。战略决策计划，自顶向下决策计划等决策支持系统中，各种实体之间复杂关系的分析确定，采用 ERD 法效果更佳。

ERD 方法的特点是直观地表现了实体、实体集之间的联系。其关键是如何确定实体、实体集的属性，以及实体与实体，属性与属性之间的关系。

(3) 状态迁移图法（State Transition Diagram，简称 STD）

任何一个系统，在某一时刻的活动情况是一组确定的状态，当情况发生变化时，其当前状态就要随之改变，即发生了状态迁移。为清晰地描述出状态迁移时的激发条件，可能发生的操作与处理以及状态之间的联系情形，我们可以采用状态迁移图（STD）来刻画一个系统的行为。对较复杂的系统状态变化可用有限状态机来描述。

STD 同 DD 一样，不论哪一类系统分析与设计都是一种有用的工具。在事务处理系统中，STD 能描述数据实体之间状态的切换过程及其关系。在网络软件、电话交换机复杂的状态迁移逻辑系统中 STD 非常有用。据专家们的进一步研究认为，STD 最有应用价值的场合是对实时软件的分析与设计。用 STD 描述法的优点还在于它能清楚的刻画出，数据实体之间若干状态操作的先后过程，这正是 DFD 描述的局限性。

从上述可见，ERD 和 STD 强化了 Demarco 方法。

2. 设计阶段的技法

系统设计阶段的任务是，将系统说明书中的用户需求转换成应用系统的具体方案。其目标是要保证系统的可靠性、可变更性、工作质量与效率。首先，要将整个系统按层次结构划分为独立的模块，并以模块结构图的形式，将组成系统的所有模块及其模块层次间的调用关系描述出来。然后写出每个模块的功能说明。模块结构图与模块功能说明形成了设计阶段的结果——模块说明书。模块说明书既是编程的基础，也是测试的依据。

在软件工程领域里，系统设计方法较多，常用于信息处理软件设计的技法有：结构化设计(或称复合设计)法，Jackson 方法，Parnas 方法等等。这些方法都采用了模块化，自顶向下，逐步细化等基本思想，主要差别是构成模块的原则不同。

(1) 结构化设计法(Structured Design，简称 SD)

结构化程序设计的基本思想是，将一个复杂的系统进行有层次的分解，成为一个个相对独立的模块结构，每个模块可以单独地进行设计、编码、测试和修改，以提高系统的质量并简化开发工作。用 SD 方法进行系统分解，并建立模块结构的依据是系统的数据流。它可以同分析阶段的 SA 方法衔接，以 DFD 为基础导出系统的模块结构。它可以适用于任何软件系统的总体设计。

SD 方法的设计原则是，将系统设计成由相对独立、功能单一的模块组成的结构。相对独立意味着每个模块具有可以独立地被理解、编码、测试、排错或修改等性能；功能单一指每个模块最好只完成一个功能。这就要求模块之间数据往来简单、界面清晰，以防止错误在模块间蔓延、增强系统的可靠性。

SD 方法在研究模块分解影响的基础上，提出了评价模块质量的两个标准：即模块之间的联系(亦称耦合度)应尽量的少(即耦合度低)；模块内部各成分之间的联系(称内聚度)应尽量的多(即内聚度高)。为此，在考虑模块分割时，应尽量将密切相关的一些成分组织在同一个模块内，以求实现块内联系紧密，也相对地减少了模块间的联系，提高了模块的相对独立性。这样，便提高了软件的可理解性、可维护性和可靠性。

数据处理系统的 DFD 图通常有两类典型的结构，SD 方法提出了由 DFD 图导出模块结构的一般规则：

(a)“变换型结构”：即可明显地分成输入、主加工、输出三部分的线性结构。可按三步

进行：首先找出主加工及其逻辑输入和逻辑输出；然后设计模块结构的顶层和第一层；最后设计中、下层模块。

(b)"事务型结构"：即是某一个加工将其输入分离成一串并行的数据流输出，供后面的加工选择执行的复杂结构，可按四步进行：首先要找出主模块；然后为每一种类型的事务处理设计一个事务处理模块；再为每个事务处理模块设计下属的操作模块；最后，为操作模块设计细节模块。

显然，模块分割的质量优劣在于设计者的经验。通常，从 DFD 图导出模块结构的初始图以后，还应经过反复分析改进。

(2) Jackson 方法

这是英国的 M. Jackson 提出的方法。在作出了大部分系统的处理是有层次结构的数据及程序结构与问题结构的对应性之后，Jackson 提出了由数据结构导出程序结构的系统设计方法。其实质仍是实现层次分解的模块化。

Jackson 方法规定，分解时只能使用三种控制结构：即顺序结构、循环结构和选择结构。根据处理数据的结构层次，分别用三种基本结构画出 Jackson 结构图解。Jackson 结构图解描述对应于数据结构的程序结构，表示程序结构元件之间的组成关系及执行规则。它与 SD 方法中常用的 Yourdon 结构图虽相似，但两者含义完全不同。Yourdon 结构图的上下层连线有箭头，表示上层模块对下层模块是调用关系；而 Jackson 结构图解的上下层组成元件之间连线不带箭头，表示的是组成关系，即上层元件由与它连接的下层元件组成。为弥补 Jackson 结构图解不能表达出循环条件和选择条件的不足， Jackson 方法引入了另一个工具——概要逻辑，概要逻辑是用类似于伪代码的描述方式，将对应的 Jackson 结构图解书写成程序的粗框架，同时列出所需的条件。因此， Jackson 方法可应用于设计和编码阶段，尤其对小型系统的开发极为流行。

Jackson 设计方法的全过程分三个步骤进行：

第一步　研究问题的环境，确定要处理的数据结构，画出 Jackson 结构图解。

第二步　用概要逻辑描述出基于数据结构的程序结构骨架。

第三步　用初等操作定义概要逻辑中描述的基本任务，并将它们分配到程序结构的合适元件中。在此基础上作进一步的细化，便可转换成用各种高级语言书写的程序。

其它结构化设计方法本书不能一一叙述，比如 Warnier 方法与 Jackson 方法十分相似，Warnier 结构图解的功能与 Jackson 结构图解类似，不同的是在 Warnier 结构图解中能明确表述出循环和选择条件。这种方法也广为国内开发者采用。此外，Parnas 设计方法，是一种以信息隐蔽为原则划分模块的结构化设计方法。它有利于适应将来的模块修改，在设计模块时将可能发生变化的因素隐含在某个模块内，使其它模块与这个因素无关，将来这个因素变化需修改时都不影响到其它模块，不致于使整个程序结构发生变化，这正是 Jackson 方法不足之处。

3. 编码阶段的技法

除上述 Jackson 方法可实现编码之外，通常，编码阶段采用的技术是结构化程序设计 (Sructured Programming，简称 SP)。编码阶段的目标是编写出逻辑上正确、易于阅读和理解的源程序代码。编码的依据是设计阶段产生的模块说明书。

SP 方法要求：程序逻辑只能用顺序、选择和循环三种基本结构表示，用这三种基本结构反复地嵌套构成结构化的程序，并限制使用 GOTO 语句。大多数高级语言都含有表示三种基本结构的语句和 Call 一类的调用语句，支持 SP 方法。这样，我们就可以采用自顶向下、逐步细化的方式编写程序。这种结构化的程序易于阅读，也易于验证其正确性。

4. 测试阶段的技法

美国的 J.Mers 对“测试”作了客观的定义：“程序测试是为了发现错误而执行程序的过程”，这说明进行程序测试的目的，是为了从程序中找出尽可能多的错误，而不是为了显示程序是好的。在一般软件开发中，通过数学方法证明程序正确性，不现实；用穷举测试法也不可能实现。这意味着实际的程序测试不可能保证程序中没有错误，而是为尽量减少错误。测试工作的目标应当是，用合理的、有限的测试实例尽可能查出最多的错误，这种观念应当明确。

目前，常用的程序测试技术有“白盒测试”和“黑盒测试”两种途径，事实证明，用这两种方法的理想测试实例：“穷举路径测试”和“穷举输入测试”是不现实和不可能的。通常的办法是利用两种方法的结合，并且只能设计一个可能发现最多错误的，可实现的测试实例，按它编制一个测试例程进行测试。可是在实际工作中，要给一个具体的程序系统设计一个最有效的测试实例，并非轻而易举的事。测试工作是一项较为复杂的创造性的高智力劳动，其工作性质要求测试者应尽心尽力地去做，方可取得成效。

（1）首先要明确测试工作的基本原则

（a）应避免由程序作者、程序设计机构的人员做测试者，方能取得好效果。

（b）在制作测试实例时，要明确两个主要内容：

- 输入数据组：应包含合法的、预期的、非预期的输入数据。
- 预期的输出结果描述。

（c）在测试实例中应包含：检查程序应做的事和检查是否做了不应做的事。

（d）应将测试实例作为程序的一部分，长期保存直到这个程序作废，以利于今后的维护工作。

（2）在测试实例设计中，用“白盒法”与“黑盒法”结合使用的方式是：先用黑盒方法设计出测试实例，然后再用白盒方法设计补充测试实例。在设计测试实例时，可视具体实例相应采用的方法有：

（a）白盒方法的等价划分，边值分析，因果图，猜测错误。

（b）黑盒方法的语句覆盖，判定覆盖，条件覆盖，判定／条件覆盖，多重条件覆盖。

（c）要按测试过程，针对模块级测试、系统级测试分别设计测试实例。

模块级测试的方式，可用自顶向下或者自底向上的增式测试，究竟选用自顶向下或是自底向上的方式，因它们各有优缺点，应视具体被测试的程序来权衡选用。

系统级测试包含功能测试、系统测试、验收测试以及安装测试等，往往可以找出数据转换过程中的遗漏、误解和干扰等大部分软件错误。

测试工作的质量，除取决于测试者的技术素质、经验和智慧之外，还与管理、测试工具等环境条件有关。笔者认为在目前的开发条件下，除尽力按测试技术要求开展测试工作外，建议注重两方面的工作：

（a）对编制好的程序组织人工测试：即进行代码审查（Code Inspections）人工运行(Walkthroughs)，以求事先找出程序的逻辑和编码错误。

（b）程序作者在将程序提交测试前，应认真做好自我测试。

我们认为，开发以数据库为中心的数据处理系统，使用 Demarco 的结构化方法进行调查、分析、设计是适合的。为合理地应用数据和进行数据库设计，也可结合采用 ERD 的分析、设计法，并能从 ERD 导出编码。在使用状态机(State Machine)模型考虑问题时，为使分析设计变得容易，可使用 STD 方法。STD 所描述的内容，用 Prolog 很容易实现，适用于验证用户接口的原型方法及自动装置的模拟等广泛的领域。在进行程序结构设计时，以结构化设计为标准。结构化设计和 Demarco 方法一样，与目前使用的程序设计语言有很高的结合性。可使用 BD(Bubble Diagram)进行物理文件设计，这种方法实现自动化容易，特别是在属性个数很多的场合，自动化会带来很大的效果。在编码和测试阶段，从最上面的模块开始向下进行编码，逐渐地完成并进行每个单体模块的测试。这样，有利于对开发中的系统整体结构进行早期测试。在进行测试实例设计时，为进行黑盒测试的测试实例设计可使用同值分割，边界值分割、因果图、错误推测等。而白盒测试可使用分支网络，白箱测试对黑盒测试是一种补充。复审很重要，它不仅使产品的质量提高，而且对提高生产效率、提高技术人员的水平都有很大的作用。因此，应当采用非正式的复审(Walkthrough)和正式的复审(Inspection)相结合的方式，以尽量提高开发软件的质量。