第5章 软 件 维 护

软件维护（Software Maintenance）就是指在软件产品交付之后对其进行修改，以排除故障，或改进性能和其他属性，或使产品适应改变了的环境。

软件维护阶段是软件生命周期中持续时间最长的一个阶段，也是需要花费的精力和费用最多的一个阶段。

软件的可维护性指软件被理解、改正、调整和改进的难易程度。可维护性是指导软件工程各阶段工作的一条基本原则，提高可维护性、减少维护的工作量、降低软件的总成本是软件工程的一个重要任务。

本章重点：

如何提高软件的可维护性。

5.1 软件维护的种类、过程和副作用

5.1.1 软件维护的种类

软件维护是软件生命期的最后一个阶段。说来也许令人难以相信，软件的维护可以占到软件开发全部工作量的一半以上。在软件运行过程中，由于种种原因，计算机程序经常需要改变。除了要纠正程序中的错误外，还要增加功能及进行优化。而在修改程序解决现有问题的时候，程序变动本身又会不断产生新的问题，还需要对软件进行修改。

软件维护分为以下 4 种。

1．改正性维护

软件测试不大可能找出一个大型软件系统的全部隐含错误。也就是说，几乎每一个大型程序在运行过程中都会不可避免地出现各种错误。为克服现有软件故障而进行的维护叫做改正性维护（Corrective Maintenance）。

2．适应性维护

计算机技术的发展十分迅速，计算机的软件、硬件环境在不断发生变化，而应用软件的使用寿命往往比原先开发时的系统环境更为长久，因此，常常需要对软件加以修改，使之适应改变了的环境。为使软件产品适应环境的变化而进行的软件维护称为适应性维护（Adaptive Maintenance）。

3．完善性维护

软件交给用户使用后，用户往往会因为工作流程、应用环境的变化，要求增加新的功能和完善性能等。这些为增加软件功能、增强软件性能、提高软件运行效率而进行的维护

是完善性维护（Perfective Maintenance）。

4．预防性维护

为了进一步提高软件的可维护性和可靠性，为改进创造条件，需要对软件进行的其他维护称为预防性维护。

综上所述，所谓软件维护就是指在软件交付使用之后，为了改正错误或满足新的需要而修改软件的过程。

据有关资料统计，各类维护的工作量占总的维护工作量的百分比大致如图5.1所示。

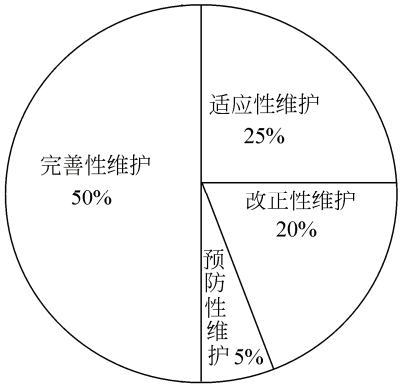

图 5.1　各类维护工作量占总的维护工作量的百分比

5.1.2　软件维护的特点

进行软件维护时常见的问题有：根据软件设计时是否有文档，软件的维护分为结构化维护和非结构化维护两类；软件的维护有可以量化的费用和不明显的代价；维护会产生其他一些问题。

1．结构化维护与非结构化维护

图 5.2 描绘了因维护要求而引起的可能的事件流程。

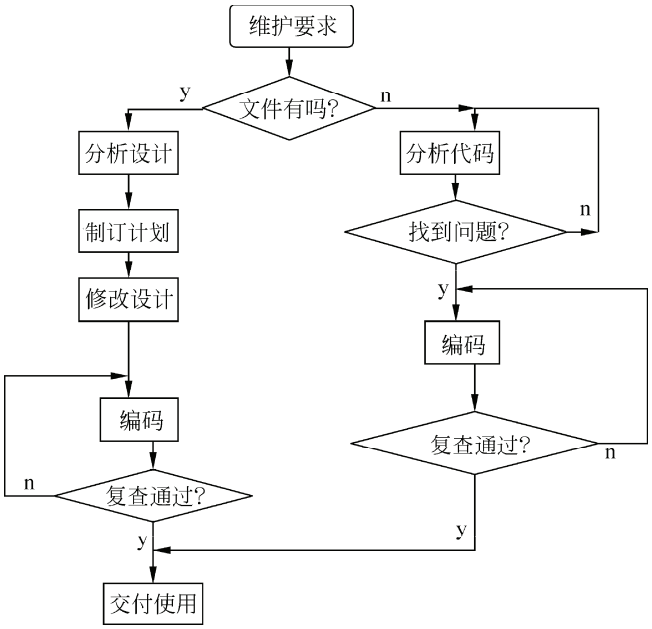

图 5.2　结构化维护与非结构化维护流程图

图 5.2 中右边的流程表示的是“非结构化维护”。在这种情况下，由于所掌握的软件文件只有源程序，维护工作只能从分析源程序开始。源程序内部注解和说明一般不会很详尽，而软件结构、全程数据结构、系统接口、性能和设计约束等细微的特征往往很难完全搞清。令人遗憾的是，维护工作往往正是在进行这种非结构化维护，并为此而付出代价。这种代

价是因没有使用良好定义的方法论来开发软件而造成的。

图 5.2 中左边的流程表示的是“结构化维护”。在这种情况下，由于掌握完整的软件文档，维护任务就可从分析设计文件开始，进而确定软件的结构特性、功能特性和接口特性，确定要求的修改将会带来的影响并计划实施方法。然后修改设计、编写相应的源程序代码、对所做的修改进行复查，并利用在测试说明书中包含的信息重复过去的测试，以确保没有因修改而把错误引入到先前运行的软件中。最后把修改后的软件再次交付使用。

与非结构化维护相比，结构化维护能减少工作量并提高维护的总体质量。这是在软件开发的早期就运用软件工程方法论的结果。

2．维护的费用和代价

有关资料表明，软件维护的费用在不断上升。1970 年用于维护的费用占软件总预算的 35%～40%，1980 年上升为 40%～60%，而 1990 年则已增至 70%～80%。

维护费用仅仅是软件维护的明显代价，还有其他许多不明显的代价：

（1）由于可用的资源必须供维护任务使用，导致耽误甚至丧失开发良机。

（2）有关改错或修改的要求不能及时满足，引起用户不满。

（3）在所维护的软件中，由于修改软件而引入潜伏的新的错误，导致软件质量下降。

（4）当必须把从事开发的软件人员调去从事维护工作时，可能打乱开发过程。

（5）由于维护每条指令的成本数十倍于开发每条指令的成本，造成生产率的大幅度下降。

3．维护的困难性

在软件生命周期的最初两个阶段如果不进行严格而又科学的管理和规划，必然会造成其生命周期的最后阶段——维护阶段产生困难。

下面列举一些软件维护的困难：

（1）理解他人编写的程序往往是非常困难的。软件文档越少，困难越大。如果只有程序代码而没有说明文档，更将出现严重困难。

（2）软件开发人员经常流动，因而当需要进行维护时，往往无法依赖开发者本人来对软件做解释说明。

（3）需要维护的软件往往没有足够的、合格的文档。请注意，光有文档是不够的，容易理解的，并且和程序代码完全一致的文档才对维护真正有价值。

（4）由于维护工作十分困难，又容易受挫，因而难以成为一项吸引人的工作。

在用没有采用软件工程的思想方法开发出来的软件时，总是会出现上述维护阶段的问题，而采用软件工程的思想方法，则可避免或减少上述问题。

5.1.3 软件维护的过程

软件维护过程实际上也是软件问题定义和开发的过程。维护活动和软件开发一样，要有严格的规范，才能保证质量。软件开发机构应建立维护组织，当用户需要软件维护时，应填写维护申请表，维护组织对此进行评价，安排维护活动。软件人员的维护过程要详细记录，并填写维护报告，最后要进行复审。

1．维护组织

软件开发机构应当建立正式的维护组织，或设立专门负责维护工作的非正式组织。维护组织由维护管理员、系统管理员和维护人员组成。系统管理员必须对软件产品程序或将被修改的那类程序相当熟悉。每当软件开发机构收到用户的维护申请后，交给负责此事的维护管理员，由他把维护申请交给系统管理员评价，再由维护人员决定如何进行修改。

2．维护文件

维护要有书面文件：用户填写维护申请表，维护管理员填写维护报告，维护人员填写维护记录。

（1）维护申请表

软件开发机构的维护组织向用户提供空白的维护申请表，由要求维护的用户填写，该表应能完整地描述软件产生错误的情况（包括输入数据、输出数据及其他有关信息）。对于适应性或完善性的维护要求，则应提出简单明了的维护要求规格说明。

维护申请表由维护管理员和系统管理员负责研究处理。

（2）软件维护报告

软件开发组织在收到用户的维护要求表后，维护管理员应写一份软件维护报告。该报告应包含下述内容：

- 按照维护要求表进行维护所需要的工作量；
- 维护要求的性质；
- 该项要求与其他维护要求相比的优先程度；
- 预计软件维护后的状况。

（3）维护记录

维护记录可以有如下内容：

- 程序名称；
- 维护类型；
- 所用的编程语言；
- 程序行数或机器指令条数；
- 程序开始使用的日期；
- 已运行次数、故障处理次数；
- 程序改变的级别及名称；
- 修改程序所增加的源语行数、所删除的源语行数；
- 各次修改耗费的人时数、累计用于维护的人时数；
- 软件工程师的姓名；
- 维护要求表的标识；
- 维护开始和结束的日期；
- 维护工作的净收益。

3．维护工作流程

维护工作流程如下：

（1）用户提出维护申请。

（2）维护组织审查申请报告并安排维护工作。

（3）进行维护并做详细的维护记录。

（4）复审。

用户的维护申请提出后，引起的工作流程如图 5.3 所示。

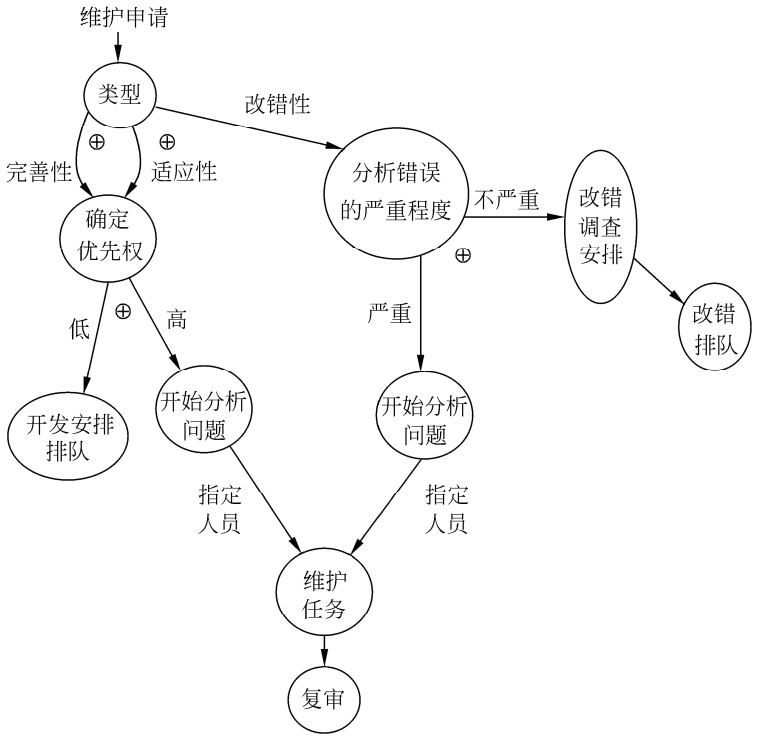

图 5.3　维护工作流程

首先要确定维护属于哪种类型。

如果属于改错性维护，则需要评价其出错的严重性。如果错误严重，就进一步指定人员，在系统管理员的指导配合下，分析错误的原因，进行维护。

对不太严重的错误，则该项改错性的维护和其他软件开发的任务一起统筹安排。

如果属于完善性或适应性维护，则先确定各个维护要求的优先次序，并且安排所需的工作时间。从其意图和目标来看，此种维护属于开发工作，因此可将其视同开发任务。

如果某项维护要求的优先次序特别高，可立即开始维护工作。

不管是改错性、完善性还是适应性维护，都需要进行同样的技术工作，包括修改软件设计、对源程序进行修改、单元测试、组装、进行有效性测试及复审等。

参加软件维护工作的人员并不是越多越好。一般对需要维护的软件比较熟悉的人员，其维护工作的效率往往比较高。

维护人员在维护过程中要做好详细的记录。对于不同类型的维护，其工作的侧重点会有所不同，但总的处理方法基本上是相同的。

当然，有时维护申请的处理过程并不完全符合上述事件流，例如软件出现紧急问题时，

就出现所谓的“救火”维护要求。在这种情况下，就需要立即投入人力进行抢救。

维护工作流程中最后一个事件是复审。维护的复审可以由同事进行，也可以由同行进行。即再次检验软件文档的各个成分的有效性，并保证实际上满足了维护申请表中的所有要求。

软件维护的复审要明确下列问题：

（1）在目前的状况下，设计、编程和测试等方面有什么可改进的。

（2）哪些维护资源应该有，而实际上却没有。

（3）什么是这项维护工作中最主要的障碍。

（4）是否需要预防性维护。

维护情况复审对将来的维护工作有重要意义，可为提高软件组织的管理效能提供重要意见。

4．对维护的评价

在维护过程中，如果缺乏详尽可靠的数据，要评价软件维护工作就很困难。如果有良好的维护记录，就可对维护工作做一些定量的评价，可计算如下一些指标：

（1）每次程序运行的平均出错次数。

（2）用在各类维护上的总的人时数。

（3）平均每个程序、每种语言、每种类型的维护所做的程序变动数。

（4）维护过程中每增加或减少一个源语句平均花费的人时数。

（5）维护每种语言平均花费的人时数。

（6）处理一张维护要求表平均所需的时间。

（7）各类维护申请的百分比。

在上述 7 种指标的基础上，可以做出有关开发技术、语言选择、维护工作计划、资源分配等方面的决定。

5.1.4 软件维护的副作用

维护是为了延长软件的寿命，让软件创造更多的价值，但是维护会产生潜在的错误或其他不希望出现的情况，这称为维护的副作用。维护的副作用有编码副作用、数据副作用和文档副作用三种。

1．编码副作用

使用程序设计语言修改源程序时可能会引入错误。在修改程序的标号、标识符、运算符、边界条件和程序的时序关系等时要特别仔细，避免引入新的错误。

2．数据副作用

修改数据结构时可能会造成软件设计与数据结构不匹配，因而导致软件错误。如在修改局部量、全局量、记录或文件的格式、初始化控制或指针、输入输出或子程序的参数等时，容易导致设计与数据不一致。

3．文档副作用

对数据流、软件结构、模块逻辑或其他任何特性进行修改时，必须对相关的文档进行相应的修改，否则会导致文档与程序功能不匹配，文档不能反映软件当前的状态。因此，必须在软件交付之前对软件配置进行评审，以减少文档的副作用。

5.2 软件的可维护性

软件可维护性是指软件功能被理解、改正、适应和增强的难易程度，可维护性是维护人员对该软件进行维护的难易程度。可维护性是指导软件工程各阶段的一条基本原则，提高可维护性是软件工程追求的目标之一。

5.2.1 决定可维护性的因素

使软件的维护工作变得困难的原因是多方面的，有维护人员素质的因素，也有技术条件和管理方面的因素等。可维护性与开发环境有关的因素如下：

- 是否拥有一组训练有素的软件人员；
- 系统结构是否可理解，是否合理；
- 文档结构是否标准化；
- 测试用例是否合适；
- 是否已有嵌入系统的调试工具；
- 是否使用合适的程序设计语言；
- 是否使用标准的操作系统。

以上影响软件可维护性的因素中，结构合理性是软件设计时应当考虑的。系统结构若不合理，对其维护当然困难较大。所谓结构的合理性主要是以下列几点为基础的：模块化、层次组织、系统文档的结构、命令的格式和约定、程序的复杂性等。

其他影响维护难易程度的因素还有应用的类型、使用的数据库技术、开关与标号的数量、IF 语句的嵌套层次、索引或下标变量的数量等。

此外，软件开发人员是否能参加维护也是值得考虑的因素。

5.2.2 可维护性的度量

软件的可维护性是难以量化的，然而借助维护活动中可以定量估算的属性，能间接地度量可维护性。例如进行软件维护所用的时间是可以记录、统计的，可以从下列维护工作所需的时间来度量软件的可维护性。

- 识别问题的时间；
- 修改规格说明书的时间；
- 分析、诊断问题的时间；
- 选择维护工具的时间；
- 纠错或修改软件的时间；
- 测试软件的时间；
- 维护评审的时间；
- 软件恢复运行的时间。

软件维护过程所需的时间越短，说明维护越容易。

软件的可维护性主要表现在它的可理解性、可测试性、可修改性和可移植性等方面。

因而，对可维护性的度量问题，也可分解成对可理解性、可测试性、可修改性和可移植性的度量问题。

1．可理解性

软件的可理解性表现为维护人员理解软件的结构、接口、功能和内部过程的难易程度。模块化、结构化设计或面向对象设计，与源程序一致的、完整正确详尽的设计文档，源代码内部的文档和良好的高级程序设计语言等，都能提高软件的可理解性。

也可以通过对软件复杂性的度量来评价软件的可理解性。软件越复杂，理解起来就越困难。可参考 4.5.5 节程序环形复杂度的度量。

2．可测试性

可测试性代表软件容易被测试的程度。它与源代码有关，要求程序易于理解；还要求有齐全的测试文档，要求保留开发时期使用的测试用例。好的文档资料对诊断和测试至关重要。

可测试性要求软件需求的定义便于对需求进行分析并易于建立测试准则；还要便于就这些准则对软件进行评价。可测试性是指证实程序正确性的难易程度。

此外，有无可用的测试、调试工具及测试过程的确定也非常重要。在软件设计阶段就应该注意使差错容易定位，以便维护时容易找到纠错的办法。

3．可修改性

可修改性是指程序容易被修改的程度。一个可修改的程序往往是可理解的、通用的、灵活的和简明的。所谓通用是指不需要修改程序就可使程序改变功能；所谓灵活是指程序容易被分解和组合。

要度量一个程序的可修改性，可以通过对该程序做少量简单的改变来估算对这个程序改变的困难程度，例如对程序增加新类型的作业、改变输入输出设备、取消输出报告等。如果对于一个简单的改变，程序中必须修改的模块超过 30%，则该程序属难于修改之列。

模块设计的内聚、耦合和局部化等因素都会影响软件的可修改性。模块抽象和信息隐蔽越好，模块的独立性越高，则修改时出错的机会也就越少。

4．可移植性

可移植性就是指软件不加改动地从一种运行环境转移到另一种运行环境下运行的能力，即程序在不同计算机环境下能够有效地运行的程度。可移植性好的软件容易维护。

5.2.3 如何提高软件的可维护性

要提高程序的可维护性，应从下列诸方面入手。

1．明确软件的质量目标

在软件开发的整个过程中，始终应该考虑并努力提高软件的可维护性，尽力将软件设计成容易理解、容易测试和容易修改的软件。

2．利用先进的软件技术和工具

软件工程在不断发展，新的技术和工具在不断出现，对软件工程的新技术和新工具应及时学习并应用。

3．选择便于维护的程序设计语言

机器语言、汇编语言不易理解，难以维护。一般只有在对软件的运行时间和使用空间

有严格限制或系统硬件有特殊要求时才使用机器语言或汇编语言。

高级语言容易理解，可维护性较好。查询语言、报表生成语言、图像语言更容易理解、使用和维护。选择适当的程序设计语言非常重要，要慎重、综合地考虑各种因素，并征求用户的意见。

4．采取有效的质量保证措施

在软件开发时需要确定中间及最终交付的成果，以及所有开发阶段各项工作的质量特征和评价标准。在每个阶段结束前的技术审查和管理复审中，也应着重对可维护性进行复审。加强软件测试工作，保证软件的质量。用户要求确保质量，如果做不到这一点，软件开发人员就不称职。本书第9章将进一步介绍软件质量保证问题。

5．完善程序的文档

软件文档的好坏直接影响软件的可维护性，软件文档应满足下列要求：

（1）描述如何使用系统，没有这种描述系统就无法使用。

（2）描述怎样安装和管理系统。

（3）描述系统需求和设计。

（4）描述系统的实现和测试。

在软件工程生命周期的每个阶段的技术复审和管理复审中，应对文档进行检查，对可维护性进行复审。

在维护阶段，利用历史文档可大大简化维护工作。历史文档有三种：系统开发文档、错误记录和系统维护文档。

为了从根本上提高软件的可维护性，在开发时明确质量目标、考虑软件的维护问题是必需的、重要的。在开发阶段提供完整的、一致的文档，采用先进的软件开发方法和软件开发工具是提高软件可维护性的关键。

小　　结

软件维护就是指在软件产品交付之后对其进行修改，以排除故障，或改进性能和其他属性，或使产品适应改变了的环境。

软件维护分为4种：改正性维护、适应性维护、完善性维护和预防性维护。

软件可维护性就是指维护人员对该软件进行维护的难易程度，具体包括理解、改正、改动和改进该软件的难易程度。

可维护性是指导软件工程各阶段的一条基本原则，提高可维护性是软件工程追求的目标之一。

在开发时明确质量目标、考虑软件的维护问题是必需的、重要的。在软件开发阶段提供完整、一致的文档，采用先进的软件开发方法和软件开发工具是提高软件可维护性的关键。

习　题　5

1．什么是软件维护？它有哪几种类型？

2．非结构化维护和结构化维护的主要区别是什么？

3．软件维护有哪些主要的副作用？

4．什么是软件的可维护性？它主要由哪些因素决定？

5．如何度量软件的可维护性？

6．如何提高软件的可维护性？

7．从下面选项中选出关于软件可维护性的正确论述：_____。

（1）在进行需求分析时就应该同时考虑软件可维护性问题。

（2）在完成测试作业之后，为缩短源程序长度，应删去源程序中的注解。

（3）尽可能在软件生产过程中保证各阶段文件的正确性。

（4）编码时应尽可能使用全局量。

（5）选择时间效率和空间效率尽可能高的算法。

（6）尽可能利用硬件的特点。

（7）重视程序的结构设计，使程序具有较好的层次结构。

（8）在进行概要设计时应加强模块间的联系。

（9）提高程序的易读性，尽可能使用高级语言编写程序。

（10）为了加快维护作业的进程，应尽可能增加维护人员的数量。

8．填空题

（1）维护阶段是软件生命周期中，持续时间_____的阶段，花费精力和费用_____的阶段。

（2）软件维护的副作用有三种：_____、_____、_____。

（3）软件维护的工作流程为_____，_____，_____，_____。

（4）在软件交付使用后，由于软件开发过程产生的_____没有完全彻底在_____阶段发现，必然有一部分隐含错误带到_____阶段。

（5）软件的可维护性是指软件功能被_____、_____、_____的难易程度。