第11章 软件维护

- 软件维护的概念
- 软件维护活动
- 程序修改的步骤及副作用
- 软件的维护性
- 提高软件维护性的方法

11.1 软件维护的概念

- **软件维护的定义**

软件维护是指在软件运行/维护阶段对软件产品所进行的修改就是所谓的维护。根据维护工作的性质，软件维护的活动可以分为以下4种类型。

- **改正性维护**
- **适应性维护**
- **完善性维护**
- **预防性维护**

11.1 软件维护的概念

1．改正性维护

改正性维护（corrective maintenance）为了识别和纠正软件错误、改正软件性能上的缺陷、排除实施中的误使用，应进行的诊断和改正错误的过程。例如，改正性维护可以是改正原来程序中开关使用的错误；解决开发时未能测试各种可能情况带来的问题等。

11.1 软件维护的概念

2．适应性维护

随着信息技术的飞速发展，软件运行的外部环境（新的硬、软件配置）或数据环境（数据库、数据格式、数据输入/输出方式、数据存储介质）可能发生变化，为了使软件适应这种变化，而修改软件的过程叫做适应性维护（adaptive maintenance）。例如，需要对已运行的软件进行改造，以适应网络环境或已升级改版的操作系统要求。

11.1 软件维护的概念

3．完善性维护

为了满足新的功能与性能要求，需要修改或再开发软件，以扩充软件功能、增强软件性能、改进加工效率、提高软件的可维护性。这种情况下进行的维护活动叫做**完善性维护**（perfective maintenance）。例如，完善性维护可能是修改一个计算工资的程序，使其增加新的扣除项目；缩短系统的应答时间，使其达到特定的要求等。

11.1 软件维护的概念

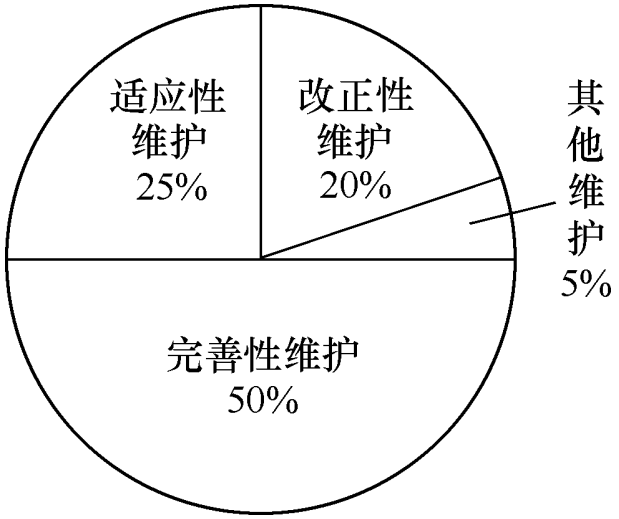
4．预防性维护

预防性维护（preventive maintenance）是指把今天的方法学用于昨天的系统以满足明天的需要。也就是说，采用先进的软件工程方法对需要维护的软件或软件中的某一部分（重新）进行设计、编码和测试。

11.1 软件维护的概念

- **各类维护占总维护工作量的比例**

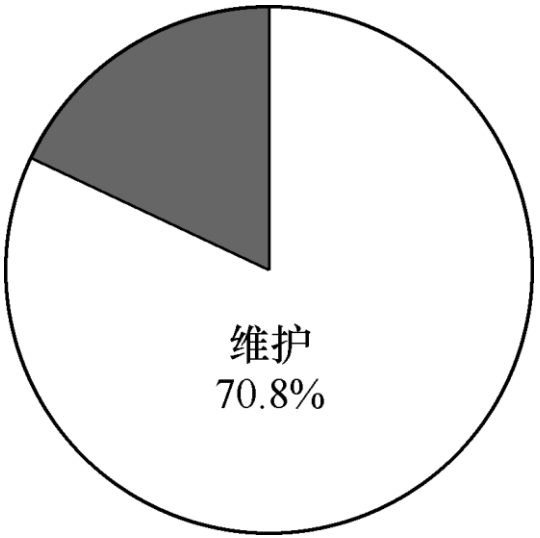
  - **在整个软件维护阶段花费的全部工作量中，预防性维护只占很小的比例，而完善性维护占了几乎一半的工作量。**



11.1 软件维护的概念

- **维护工作量在软件生存期中所占比例**

  - **软件维护活动花费的工作量占整个生存期工作量的70%以上（工作量的比例直接反映了成本的比例）**



11.1 软件维护的概念

- **影响维护工作量的因素**

在软件维护中，影响维护工作量的因素主要有以下6种:

（1）系统规模。

（2）程序设计语言。

（3）系统年龄大小。

（4）数据库技术的应用水平。

（5）所采用的软件开发技术及软件开发工程化的程度。

（6）其他：如应用的类型、数学模型、任务的难度、IF嵌套深度、索引或下标数等，对维护工作量都有影响。

11.1 软件维护的概念

- **软件维护的策略**

根据影响软件维护工作量的各种因素，针对3种典型维护，James Martin等提出了一些策略，以控制维护成本。

1．改正性维护

应用一些诸如数据库管理系统、软件开发环境、程序自动生成系统和高级（第四代）语言等新技术可大大提高可靠性，并减少进行改正性维护的需要。此外，还可考虑利用应用软件包、防错性程序设计、通过周期性维护审查等策略。

11.1 软件维护的概念

2．适应性维护

这一类的维护不可避免，但可以采用以下策略加以控制。

（1）在配置管理时，把硬件、操作系统和其他相关环境因素的可能变化考虑在内，可以减少某些适应性维护的工作量。

（2）把与硬件、操作系统，以及其他外围设备有关的程序归到特定的程序模块中。可把因环境变化而必须修改的程序局部于某些程序模块之中。

11.1 软件维护的概念

2．适应性维护

（3）使用内部程序列表、外部文件，以及处理的例行程序包，可为维护时修改程序提供方便。

（4）使用面向对象技术，增强软件系统的稳定性，易于修改和移植。

11.1 软件维护的概念

3．完善性维护

利用前两类维护中列举的方法，也可以减少这一类维护。特别是数据库管理系统、程序生成器、应用软件包，可减少系统或程序员的维护工作量。

此外，建立软件系统的原型，把它在实际系统开发之前提供给用户。用户通过研究原型，进一步完善他们的功能要求，可以减少以后完善性维护的需要。

11.2 软件维护活动

- **软件维护申请报告**

所有软件维护申请应按规定的方式提出。软件维护组织通常提供维护申请报告（maintenance request form，MRF），或称软件问题报告，由申请维护的用户填写。

- **如果遇到一个错误，用户必须完整地说明产生错误的情况，包括输入数据、错误清单以及其他有关材料。**
- **如果申请的是适应性维护或完善性维护，用户必须提出一份修改说明书，列出所有希望的修改。维护申请报告将由维护管理员和系统监督员来研究处理。**

11.2 软件维护活动

维护申请报告是由软件组织外部提交的文档，它是计划维护工作的基础。软件组织内部应相应地做出软件修改报告（software change report，SCR），指明：

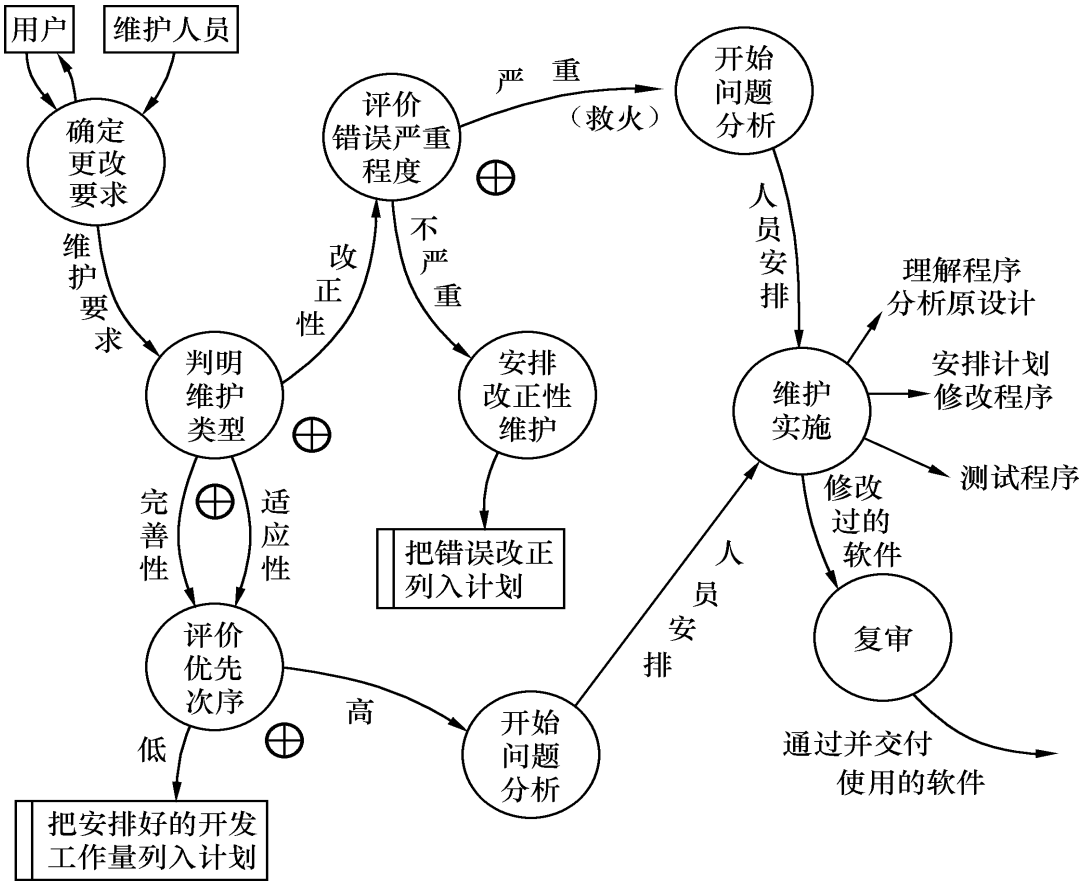
- 所需修改变动的性质；
- 申请修改的优先级；
- 为满足某个维护申请报告，所需的工作量；
- 预计修改后的状况。

软件修改报告应提交修改负责人，经批准后才能开始进一步安排维护工作。

11.2 软件维护活动

- **软件维护工作流程**

软件维护工作流程如下图所示。



11.2 软件维护活动

在每次软件维护任务完成后，最好进行一次情况评审，对以下问题做一总结：

- **在目前情况下，设计、编码、测试中的哪一方面可以改进？**
- **哪些维护资源应该有，但没有？**
- **工作中主要的或次要的障碍是什么？**
- **从维护申请的类型来看是否应当有预防性维护？**

情况评审对将来的维护工作如何进行会产生重要的影响，并可为软件机构的有效管理提供重要的反馈信息。

11.2 软件维护活动

- **维护档案记录**

  内容包括程序名称、源程序语句条数、机器代码指令条数、所用的程序设计语言、程序安装的日期、程序安装后的运行次数、与程序安装后运行次数有关的处理故障次数、程序改变的层次及名称、修改程序所增加的源程序语句条数、修改程序所减少的源程序语句条数、每次修改所付出的“人时”数、修改程序的日期、软件维护人员的姓名、维护申请报告的名称、维护类型、维护开始时间和维护结束时间、花费在维护上的累计“人时”数、维护工作的净收益等。对每项维护任务都应该收集上述数据。

11.2 软件维护活动

- **维护评价**

评价维护活动可参考的度量值有：

- **每次程序运行时的平均出错次数；**
- **花费在每类维护上的总“人时”数；**
- **每个程序、每种语言、每种维护类型的程序平均修改次数；**
- **因为维护，增加或删除每个源程序语句所花费的平均“人时”数；**
- **用于每种语言的平均“人时”数；**
- **维护申请报告的平均处理时间；**
- **各类维护申请的百分比。**

11.3 程序修改的步骤及修改的副作用

为了正确、有效地进行程序修改，需要经历3个步骤：分析和理解程序、实施修改以及重新验证程序。

- **分析和理解程序**

经过分析，全面、准确、迅速地理解程序是决定维护成败和质量好坏的关键。在这方面，软件的可理解性和文档的质量非常重要。为此必须：

（1）研究程序的使用环境及有关资料，尽可能得到更多的背景信息；

（2）理解程序的功能和目标；

11.3 程序修改的步骤及修改的副作用

（3）掌握程序的结构信息，即从程序中细分出若干结构成分，如程序系统结构、控制结构、数据结构和输入/输出结构等；

（4）了解数据流信息，即所涉及的数据来自何处，在哪里被使用；

（5）了解控制流信息，即执行每条路径的结果；

（6）如果设计存在，则可利用它们来帮助画出结构图和高层流程图；

（7）理解程序的操作（使用）要求。

11.3 程序修改的步骤及修改的副作用

为了容易地理解程序，要求自顶向下地理解现有源程序的程序结构和数据结构，为此可采用如下几种方法。

（1）分析程序结构图。

（2）数据跟踪。

（3）控制跟踪。可采用符号执行或实际动态跟踪的方法，了解数据是如何从一个输入源到达输出点的。

（4）在分析的过程中，应充分阅读和使用源程序清单和文档，分析现有文档的合理性。

（5）充分使用由编译程序或汇编程序提供的交叉引用表、符号表，以及其他有用的信息。

（6）如有可能，争取参加开发工作。

11.3 程序修改的步骤及修改的副作用

- **修改程序**

对程序的修改，必须事先做出计划，有准备地、周密有效地实施修改。

1．设计程序的修改计划

程序的修改计划要考虑人员和资源的安排。修改计划的内容主要包括以下几项：

（1）规格说明信息：数据修改、处理修改、作业控制语言修改、系统之间接口的修改等。

（2）维护资源：新程序版本、测试数据、所需的软件系统、计算机时间等。

11.3 程序修改的步骤及修改的副作用

（3）人员：程序员、用户相关人员、技术支持人员、厂家联系人、数据录入员等。

（4）提供：纸质、计算机媒体等。

针对以上每一项，要说明必要性、从何处着手、是否接受、日期等。通常，可采用自顶向下的方法，在理解程序的基础上做如下工作：

（1）研究程序的各个模块、模块的接口及数据库，从全局的观点提出修改计划。

（2）依次把要修改的、以及那些受修改影响的模块和数据结构分离出来。

11.3 程序修改的步骤及修改的副作用

（3）详细地分析要修改的，以及那些受变更影响的模块和数据结构的内部细节，设计修改计划，标明新逻辑及要改动的现有逻辑。

（4）向用户提供回避措施。用户的某些业务因软件中发生问题而中断，为不让系统长时间停止运行，需把问题局部化，在可能的范围内继续开展业务。

11.3 程序修改的步骤及修改的副作用

2．修改代码，以适应变化

（1）正确、有效地编写修改代码；

（2）要谨慎地修改程序，尽量保持程序的风格及格式，要在程序清单上注明改动的指令；

（3）不要匆忙删除程序语句，除非完全肯定它是无用的；

（4）不要试图共用程序中已有的临时变量或工作区，为了避免冲突或混淆用途，应自行设置自己的变量；

（5）插入错误检测语句；

（6）保持详细的维护活动和维护结果记录；

（7）如果程序结构混乱，修改受到干扰，可抛弃程序重新编写。

11.3 程序修改的步骤及修改的副作用

- **修改程序的副作用及其控制**

所谓程序修改的副作用是指因修改软件而造成的错误或其他不希望发生的情况，有以下3种副作用：

1．修改代码的副作用

在使用程序设计语言修改源代码时，都可能引入新的错误。例如，删除或修改一个子程序、删除或修改一个标号、删除或修改一个标识符、改变程序代码的时序关系、改变占用存储的大小、改变逻辑运算符、修改文件的打开或关闭、改进程序的执行效率，以及把设计上的改变翻译成代码的改变、为边界条件的逻辑测试做出改变时，都容易引入错误。

11.3 程序修改的步骤及修改的副作用

2．修改数据的副作用

在修改数据结构时，有可能造成软件设计与数据结构不匹配，因而导致软件出错。**修改数据的副作用**是修改软件信息结构导致的结果。例如，在重新定义局部的或全局的常量、重新定义记录或文件的格式、增大或减小一个数组或高层数据结构的大小、修改全局或公共数据、重新初始化控制标志或指针、重新排列输入/输出或子程序的参数时，容易导致设计与数据不相容的错误。数据副作用可以通过详细的设计文档加以控制。

11.3 程序修改的步骤及修改的副作用

3．修改文档的副作用

对数据流、软件结构、模块逻辑或任何其他有关特性进行修改时，必须对相关技术文档进行相应修改。如果对可执行软件的修改不反映在文档里，会产生**文档的副作用**。例如，对交互输入的顺序或格式进行修改，如果没有正确地记入文档中，可能引起重大的问题。过时的文档内容、索引和文本可能造成冲突，引起用户业务的失败和不满。因此，必须在软件交付之前对整个软件配置进行评审，以减少文档的副作用。

11.3 程序修改的步骤及修改的副作用

为了控制因修改而引起的副作用，要做到：

（1）按模块把修改分组；

（2）自顶向下地安排被修改模块的顺序；

（3）每次修改一个模块；

（4）对于每个修改了的模块，在安排修改下一个模块之前，要确定这个修改的副作用，可以使用交叉引用表、存储映象表、执行流程跟踪等。

11.3 程序修改的步骤及修改的副作用

- **重新验证程序**

1．静态确认

修改的软件，通常伴随着引起新的错误的危险。为了能够做出正确的判定，验证修改后的程序至少需要两个人参加。要检查：

（1）修改是否涉及规格说明？修改结果是否符合规格说明？有没有歪曲规格说明？

（2）程序的修改是否足以修正软件中的问题？源程序代码有无逻辑错误？修改时有无修补失误？

（3）修改部分对其他部分有无不良影响（副作用）？

对软件进行修改，常常会引发别的问题，因此，有必要检查修改的影响范围。

11.3 程序修改的步骤及修改的副作用

2．确认测试

在充分进行了以上确认的基础上，要用计算机对修改程序进行确认测试。

（1）确认测试顺序：先对修改部分进行测试，然后隔离修改部分，测试程序的未修改部分，最后再把它们集成起来进行测试。这种测试称为**回归测试**。

（2）准备标准的测试用例。

（3）充分利用软件工具帮助重新验证过程。

（4）在重新确认过程中，需邀请用户参加。

11.3 程序修改的步骤及修改的副作用

3．维护后的验收

在交付新软件之前，维护主管部门要检验：

（1）全部文档是否完备，并已更新；

（2）所有测试用例和测试结果已经正确记载；

（3）记录软件配置所有副本的工作已经完成；

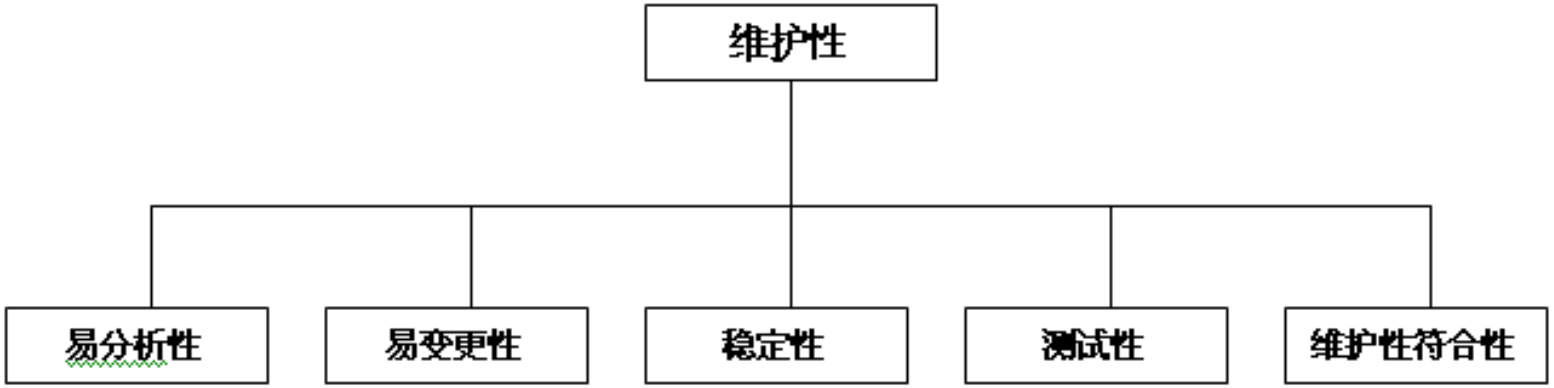
（4）维护工序和责任是明确的。

11.4 软件的维护性

- **软件维护性的定义**

➢ **软件维护性是指当对软件实施各种类型的维护而进行修改时，软件产品可被修改的能力。**

➢ **软件维护的子特性：**



11.4 软件的维护性

- **软件维护性度量的任务是对软件产品的维护性给出量化的评价。**
- **软件维护的度量也分为内部维护性度量和外部维护性度量，两者的差别如下表。**

内部维护性度量与外部维护性度量

	内部维护性度量	外部维护性度量
度量的目的	预测修改软件所需的工作量	度量修改软件产品的工作量
度量的时机	软件产品设计和编程阶段	代码完成后测试和运行时
度量的对象	对软件中间产品实施静态测量	执行代码收集数据

11.5 提高软件维护性的方法

- **使用提高软件质量的技术和工具**

1．模块化

模块化技术的优点是如果需要改变某个模块的功能，则只要改变这个模块，对其他模块影响很小；如果需要增加程序的某些功能，则仅需增加完成这些功能的新的模块或模块层；程序的测试与重复测试比较容易；程序错误易于定位和纠正；容易提高程序效率。

2．结构化程序设计

结构化程序设计不仅使得模块结构标准化，而且将模块间的相互作用也标准化了，因而把模块化又向前推进了一步。采用结构化程序设计可以获得良好的程序结构。

11.5 提高软件维护性的方法

3．使用结构化程序设计技术，提高现有系统的可维护性

（1）采用备用件的方法——当要修改某一个模块时，用一个新的结构良好的模块替换掉整个模块。

（2）采用自动重建结构和重新格式化的工具（结构更新技术）。

（3）改进现有程序的不完善的文档。

（4）使用结构化程序设计方法实现新的子系统。

（5）采用结构化小组。

11.5 提高软件维护性的方法

- **实施开发阶段产品的维护性审查**

质量保证审查除了保证软件得到适当的质量外，还可以用来检测在开发和维护阶段内发生的质量变化。一旦检测出问题来，就可以采取措施纠正，以控制不断增长的软件维护成本。

为了保证软件的可维护性，有4种类型的软件审查。

- **检查点审查**
- **验收检查**
- **周期性的维护审查**
- **对软件包进行检查**

11.5 提高软件维护性的方法

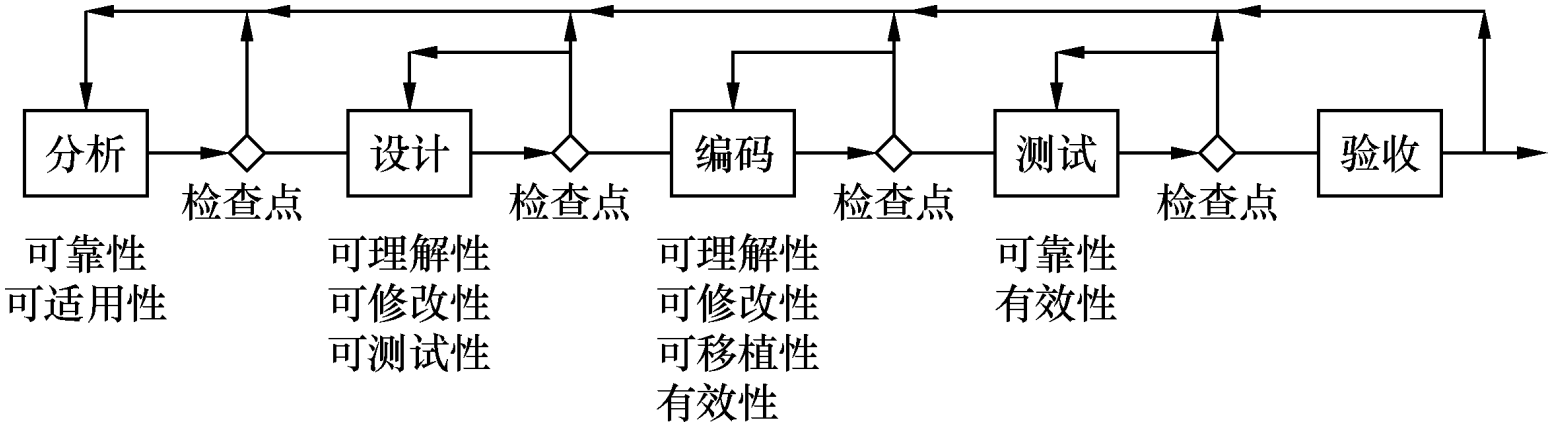
1．检查点审查

- 保证软件质量的最佳方法是在软件开发的最初阶段就把质量要求考虑进去，并在开发过程每一个阶段的终点，设置检查点进行检查。
- 检查的目的是要证实，已开发的软件是否符合标准，是否满足规定的质量需求。

11.5 提高软件维护性的方法

1. 检查点审查

- **在不同的检查点，检查的重点不完全相同，例如，在设计阶段，检查重点是可理解性、可修改性、可测试性。可理解性检查的重点是程序的复杂性。如下图所示。**



11.5 提高软件维护性的方法

2．验收检查

验收检查是一个特殊的检查点的检查，是交付使用前的最后一次检查，是软件投入运行之前保证可维护性的最后机会。以下是验收检查必须遵循的最小验收标准。

（1）需求和规范标准

① 需求应当以可测试的术语进行书写，按优先次序排列和定义。

② 区分必须的、任选的、将来的需求。

③ 包括对系统运行时的计算机设备的需求；对维护、测试、操作，以及维护人员的需求；对测试工具等的需求。

11.5 提高软件维护性的方法

（2）设计标准

① 程序应设计成分层的模块结构。每个模块应完成唯一的功能，并达到高内聚、低耦合。

② 通过一些知道预期变化的实例，说明设计的可扩充性、可缩减性和可适应性。

11.5 提高软件维护性的方法

（3）源代码标准

① 尽可能使用程序设计语言的标准版本。

② 所有的代码都必须具有良好的结构。

③ 所有的代码都必须文档化，在注释中说明它的输入、输出，以及便于测试/再测试的一些特点与风格。

（4）文档标准

文档中应说明程序的输入/输出、使用的方法/算法、错误恢复方法、所有参数的范围、默认条件等。

11.5 提高软件维护性的方法

3．周期性地维护审查

- 检查点复查和验收检查，可用来保证新软件系统的可维护性。对已有的软件系统，则应当进行周期性的维护检查。
- 软件在运行期间，必须对软件做周期性的维护审查，以跟踪软件质量的变化。
- 周期性维护审查实际上是开发阶段检查点复查的继续，并且采用的检查方法、检查内容都是相同的。

11.5 提高软件维护性的方法

4．对软件包进行检查

软件包是一种标准化了的、可为不同单位、不同用户使用的封装软件。

使用单位的维护人员首先要仔细分析、研究开发商提供的用户手册、操作手册、培训教程、新版本说明、计算机环境要求书，以及开发商提供的验收测试报告等，在此基础上，深入了解本单位的希望和要求，编制软件包的检验程序。该检验程序检查软件包程序所执行的功能是否与用户的要求和条件相一致。

11.5 提高软件维护性的方法

- **改进文档**

程序文档是对程序总目标、程序各组成部分之间的关系、程序设计策略、程序实现过程的历史数据等的说明和补充。程序文档对提高程序的可理解性有着十分重要作用。在软件维护阶段，利用历史文档，可以大大简化维护工作。

11.5 提高软件维护性的方法

- 历史文档有如下3种：

（1）系统开发日志：它记录了项目的开发原则、开发目标、优先次序、选择某种设计方案的理由、决策策略、使用的测试技术和工具、每天出现的问题、计划的成功和失败之处等。

（2）错误记载：它把出错的历史情况记录下来，对于预测今后可能发生的错误类型及出错频率有很大帮助。也有助于维护人员查明出现故障的程序或模块，以便去修改或替换它们。

11.5 提高软件维护性的方法

（3）系统维护日志：记录了在维护阶段有关系统修改和修改目的的信息。包括修改的宗旨、修改的策略、存在的问题、问题所在的位置、解决问题的办法、修改要求和说明、注意事项、新版本说明等信息。

That's All!