分析一个工程的可靠度和可维护性

1. 软件工程中软件的可靠性

软件的[安全](http://www.ltesting.net/ceshi/ceshijishu/aqcs/" \t "https://blog.csdn.net/juandaisy/article/details/_blank)可靠性是衡量软件好坏的一个重要标准，[安全](http://www.ltesting.net/ceshi/ceshijishu/aqcs/" \t "https://blog.csdn.net/juandaisy/article/details/_blank)性指与防止对程序及数据的非授权的故意或意外访问的能力有关的软件属性，可靠性指与在规定的一段时间和条件下，软件能维持其[性能](http://www.ltesting.net/ceshi/ceshijishu/xncs/" \t "https://blog.csdn.net/juandaisy/article/details/_blank)水平能力有关的一组属性。

一、软件可靠性定义

    长期运行的稳定性(鲁棒性)

    输出数据的正确性

    异常情况的可记录性

二、影响软件可靠性的因素

1)  需求分析定义错误

    由于分析失误，从开始就走上了错误的路线，向着错误的目标前进，以后实现中的错误在所难免。

2) 设计质量

    设计水平的高低与设计者的水平有着直接的联系，但可以通过人文方法提高设计水平，但不仅限于此。影响设计质量因素主要有： 对需求的理解程度、对软件环境的理解程度、设计人员的设计水平等。

3) 编码质量

    编码的过程实际上是影响软件可靠性的一个关键因素，影响这个过程的因素有很多，如：程序语言的选择、程序员对语言特性的掌握以及编码水平、编码质量检查与评审制度及执行情况、代码的复用率

4) 无效的测试

    如果在进行单元测试或者集成测试时，如果测试用例设计不合理，测试不完整，容易使测试失效，使得软件在某些未经测试的情况下故障。

5) 文档错误

如果文档(包括正式的设计文档以及程序的注释)不完整、不一致，会导致阅读者对设计或代码理解产生偏差，从而有可能导致下一步的软件错误。

三、如何提高软件可靠性

1.建立以可靠性为核心的质量标准   
　　在软件项目规划和需求分析阶段就要建立以可靠性为核心的质量标准。这个质量标准包括实现的功能、可靠性、可维护性、可移植性、[安全](http://security.chinaitlab.com/" \t "https://blog.csdn.net/gerryzhu/article/details/_blank)性、吞吐率等等，虽然还没有一个衡量软件质量的完整体系，但还是可以通过一定的指标来指定标准基线。   
　　软件质量从构成因素上可分为产品质量和过程质量。   
  
　　产品质量是软件成品的质量，包括各类文档、编码的可读性、可靠性、正确性，用户需求的满足程度等。   
　　过程质量是开发过程环境的质量，与所采用的技术、开发人员的素质、开发的组织交流、开发设备的利用率等因素有关。   
　　还可把质量分为动态质量和静态质量。静态质量是通过审查各开发过程的成果来确认的质量，包括模块化程度、简易程度、完整程度等内容。动态质量是考察运行状况来确认的质量，包括平均故障间隔时间(MTBF)、软件故障修复时间(MTRF)、可用资源的利用率。在许多实际工程中，人们一般比较重视动态质量而忽视静态质量。   
  
　　所定的质量标准度量，至少应达到以下两个目的:   
　　(1).明划分各开发过程(需求分析过程，设计过程，测试过程，验收过程)，通过质量检验的反馈作用确保差错及早排除并保证一定的质量。   
　　(2).在各开发过程中实施进度管理，产生阶段质量评价报告，对不合要求的产品及早采取对策。   
  
　　确定划分的各开发过程的质量度量：   
　　(1).需求分析质量度量  
　　需求分析定义是否完整、准确(有无二义性)，开发者和用户间有没有理解不同的情况，文档完成情况等，要有明确的可靠性需求目标、分析设计及可靠性管理措施等。   
　　(2).设计结果质量度量   
　　设计工时，程序容量和可读性、可理解性，测试情况数，评价结果，文档完成情况等。   
　　(3).测试结果质量度量   
　　测试工时，差错状况，差错数量，差错检出率及残存差错数，差错影响评价，文档等，以及有关非法输入的处理度量。   
　　(4).验收结果质量度量   
  
　　完成的功能数量，各项性能指标，可靠性等。   
　　最后选择一种可靠度增长曲线预测模型，如时间测量、个体测量、可用性，在后期开发过程中，用来计算可靠度增长曲线的差错收敛度。   
　　在建立质量标准之后，设计质量报告及评价表，在整个开发过程中就要严格实施并及时作出质量评价，填写报告表。   
  
　　2. 选择开发方法   
　　软件开发方法对软件的可靠性也有重要影响。   
　　目前的软件开发方法主要有Parnas方法、Yourdon方法、面向数据结构的Jackson方法和Warnier方法、PSL/PSA方法、原型化方法、面向对象方法、可视化方法、ICASE方法、瑞理开发方法等，其他还有BSP方法、CSF方法等。这里特别要提一下的是Parnas方法。  
　　Parnas方法是最早的软件开发方法，是Parnas 在1972年提出来的，基本思想是在概要设计时预先估计未来可能发生变化，提出了信息隐藏的原则以提高软件的可靠性和可维护性。   
　　在设计中要求先列出将来可能要变化的因素，在划分模块时将一些可能发生变化的因素隐含在某个模块的内部，使其他模块与此无关，这样就提高了软件的可维护性，避免了错误的蔓延，也就提高了软件的可靠性。还提出了提高可靠性的措施：   
  
　　(1)考虑到硬件有可能出故障，接近硬件的模块要对硬件行为进行检查，及时发现错误。   
　　(2)考虑到操作人员有可能失误，输入模块对输入数据进行合法性检查，是否合法、越权，及时纠错。   
　　(3)考虑到软件本身有可能失误，加强模块间检查，防止错误蔓延。   
  
　　对瑞理方法可能许多人还不熟悉，这里简要介绍一下。   
　　瑞理（Rational)模式是美国瑞理软件工程公司发展出来的，其模式是：   
　　面向对象；   
　　螺旋式上升；   
　　管理与控制；   
　　高度自动化；   
　　 管理观点和技术观点把软件生命周期划分为起始、规划、建构、转移、进化五个阶段，也可把这五个阶段归并为研究时期（起始和规划）和生产时期（建构和转移），最后是维护时期（进化），特别适合对高风险部分及变动需求的处理。   
　　在以上的众多方法中，可视化方法主要用于与图形有关的应用，目前的可视化开发工具只能提供用户界面的可视化开发，对一些不需要复杂图形界面的应用不必使用这种方法；ICASE 技术还没有完全成熟，所以可视在方法和ICASE方法最多只能用作辅助方法。面向数据结构的方法、PSL/PSA方法及原型化方法只适合于中小型系统的开发。   
　　面向对象的方法便于软件复杂性控制，有利于生产率的提高，符合人类的思维习惯，能自然地表达现实世界的实体和问题，具有一种自然的模型化能力，达到从问题空间到解空间的较为直接自然的映射。   
　　在面向对象的方法中，由于大量使用具有高可靠性的库，其可靠性也就有了保证，用面向对象的方法也利于实现软件重用。   
　　所以建议采用面向对象的方法，借鉴Parnas和瑞理模式的思想，在开发过程中再结合使用其他方法，吸取其它方法的优点。   
  
　　3.软件重用   
　　最大限度地重用现有的成熟软件，不仅能缩短开发周期，提高开发效率，也能提高软件的可维护性和可靠性。因为现有的成熟软件，已经过严格的运行检测，大量的错误已在开发、运行和维护过程中排除，应该是比较可靠的。在项目规划开始阶段就要把软件重用列入工作中不可缺少的一部分，作为提高可靠性的一种必要手段。

　软件重用不仅仅是指软件本身，也可以是软件的开发思想方法、文档，甚至环境、数据等，包括三个方面内容的重用：   
　　(1)开发过程重用，指开发规范、各种开发方法、工具和标准等。   
　　(2)软件构件重用，指文档、程序和数据等。   
　　(3)知识重用，如相关领域专业知识的重用。   
  
　　一般用的比较多的是软件构件重用。   
　　软件重用的过程如下：候选，选择，资格，分类和[存储](http://www.storworld.com/" \t "https://blog.csdn.net/gerryzhu/article/details/_blank)，查找和检索。在选择可重用构件时，一定要有严格的选择标准，可重用的构件必须是经过严格测试的、甚至是经过可靠性和正确性证明的构件，应模块化（实现单一、的完整的功能）、结构清晰（可读、可理解、规模适当），且有高度可适应性。   
  
　　4.使用开发管理工具   
　　开发一个大的软件系统，离不开开发管理工具，作为一个项目管理员，仅仅靠人来管理是不够的，需要有开发管理工具来辅助解决开发过程中遇到的各种各样的问题，以提高开发效率和产品质量。   
　　如Intersolv公司的PVCS软件开发管理工具，在美国市场占有率已超过70％，使用PVCS可以带来不少好处：规范开发过程，缩短开发周期，减少开发成本，降低项目投资风险；自动创造完整的文档，便于软件维护；管理软件多重版本；管理和追踪开发过程中危及软件质量和影响开发周期的缺陷和变化，便于软件重用，避免数据丢失，也便于开发人员的交流，对提高软件可靠性，保证质量有很大作用。  
  
　　在我国，开发管理工具并没有得到有效地使用，许多软件公司还停留在人工管理阶段，所开发的软件质量不会很高。   
　　人的管理比较困难，在保证开发人员素质的同时，要保持人员的稳定性，尽可能避免人员的经常流动。人员流动影响了软件的质量，工作连续性难保证，继承者不可能对情况了解很清楚等，也可能影响工作进程等。PVCS也提供了适当的人员管理方法。   
  
　　5.加强测试   
　　软件开发前期各阶段完成之后，为进一步提高可靠性，只有通过加强测试来实现了。为最大限度地除去软件中的差错，改进软件的可靠性，就要对软件进行完备测试。要对一个大的软件系统进行完备测试是不可能的，所以要确定一个最小测试数和最大测试数，前者是技术性的决策，后者管理性的决策，在实际过程中要确定一个测试数量的下界。总的来说，要在可能的情况下，进行尽可能完备的测试。   
　　谁来做测试呢？一般说来，用户不大可能来进行模块测试，模块测试应该由最初编写代码的程序员来进行，要在他们之间交换程序进行模块测试，自己设计的程序自己测试一般都达不到好的效果。   
　　测试前要确定测试标准、规范，测试过程中要建立完整的测试文档，把软件置于配置控制下，用形式化的步骤去改变它，保证任何错误及对错误的动作都能及时归档。   
  
　　测试规范包括以下三类文档：   
　　(1)测试设计规范：详细描述测试方法，规定该设计及其有关测试所包括的特性。还应规定完成测试所需的测试用例和测试规程，规定特性的通过/失败判定准则。   
　　(2)测试用例规范：列出用于输入的具体值及预期输出结果。规定在使用具体测试用例时对测试规程的各种限制。   
　　(3)测试规程规范：规定对于运行该系统和执行指定的测试用例来实现有关测试所要求的所有步骤。

　　测试的方法多种多样：   
　　(1)走查（Walk-through），即手工执行，由不同的程序员(非该模块设计者)读代码，并进行评论。   
　　(2)机器测试，对给定的输入不会产生不合逻辑的输出。   
　　(3)程序证明或交替程序表示。   
　　(4)模拟测试，模拟硬件、I/O设备等。   
　　(5)设计审查，关于设计的所有各方面的小组讨论会，利用所获得的信息，找出缺陷及违反标准的地方等。   
　　以上可以交替并行循环执行，在实际测试过程中要使用测试工具提高效率。   
　　除正常的测试之外，还要对软件进行可靠性测试，确保软件中没有对可靠性影响较大的故障。制定测试计划方案，按实际使用的概率分布随机选择输入，准确记录运行时间和结果，并对结果进行评价。   
　　没有错误的程序同永动机一样是不可能达到的。一般常用排错方法有试探法、追溯法、归纳法、演绎法。还要使用适当的排错工具，如UNIX提供的sdb和dbx编码排错工具，这些排错工具只有浏览功能，没有修改功能，是实际的找错工具。   
　　6.容错设计   
　　提高可靠性的技术一般可以分为两类，一类是避免故障，在开发过程中，尽可能不让差错和缺陷潜入软件，这类常用的技术有：   
　　算法模型化，把可以保证正确实现需求规格的算法模型化。   
　　模拟模型化，为了保证在确定的资源条件下的预测性能的发挥，使软件运行时间、内存使用量及控制执行模型化。   
　　可靠性模型，使用可靠性模型，从差错发生频度出发，预测可靠性。   
　　正确性证明，使用形式符号及数学归纳法等证明算法的正确性。   
　　软件危险分析与故障树分析：从设计或编码的结构出发，追踪软件开发过程中潜入系统缺陷的原因。   
　　分布接口需求规格说明：在设计的各阶段使用形式的接口需求规格说明，以便验证需求的分布接口实现可能性与完备性。   
　　这些技术一般都需要比较深厚的数学理论知识和模型化技术。   
　　另一类就是采用冗余思想的容错技术。   
　　容错技术的基本思想是使软件内潜在的差错对可靠性的影响缩小控制到最低程度。   
　　软件的容错从原理上可分为错误分析、破坏程度断定、错误恢复、错误处理四个阶段。   
　　常用的软件容错技术有N-版本技术、恢复块技术、多备份技术等。   
　　N-版本程序设计是依据相同规范要求独立设计N个功能相等的程序(即版本)。独立是指使用不同的算法，不同的设计语言，不同的测试技术，甚至不同的指令系统等。   
　　恢复块技术是使用自动前向错误恢复的故障处理技术。   
　　以上这些技术可参考有关文献，这里要说的是防错性程序设计，在程序中进行错误检查。被动的防错性技术是当到达检查点时，检查一个计算机程序的适当点的信息。主动的防错性技术是周期性地搜查整个程序或数据，或在空闲时间寻找不寻常的条件。采用防错性程序设计，是建立在程序员相信自己设计的软件中肯定有错误这一基础上的，有的程序员可能对此不大习惯，因为他可能太相信自己，相信自己的程序只有很少错误，甚至没有错误，作为一个项目管理员应该能说服他或者强制他采用这种技术，虽然在设计时要花费一定的时间，但这对提高可靠性很有用。

1. **软件工程中软件的可维护性（可维护性的7个特性）**
2. 可理解性：

可理解性表明人们通过阅读源代码和相关文档，了解程序功能及其如何运行的容易程度

二.可靠性：

可靠性表明一个程序按照用户的要求和设计目标，在给定的一段时间内正确执行的概率。

三.可测试性

可测试性表明论证程序正确性的容易程度。程序越简单，证明其正确性就越容易。而且设计合用的测试用例，取决于对程序的全面理解。  
一个可测试的程序应当是可理解的，可靠的，简单的。  
用于可测试性度量的检查项目如下：  
程序是否模块化? 结构是否良好?  
程序是否可理解? 程序是否可靠?  
程序是否能显示任意中间结果?  
程序是否能以清楚的方式描述它的输出?  
程序是否能及时地按照要求显示所有的输入?  
程序是否有跟踪及显示逻辑控制流程的能力?  
程序是否能从检查点再启动?  
程序是否能显示带说明的错误信息?

四.可修改性

可修改性表明程序容易修改的程度。

五.可移植性

可移植性表明程序转移到一个新的计算环境的可能性的大小。或者它表明程序可以容易地、有效地在各种各样的计算环境中运行的容易程度。  
一个可移植的程序应具有结构良好、灵活、不依赖于某一具体计算机或操作系统的性能。

**用于可移植性度量的检查项目如下：**

1.是否是用高级的独立于机器的语言来编写程序?  
2.是否使用广泛使用的标准化的程序设计语言来编写程序? 是否仅使用了这种语言的标准版本和特性?  
3.程序中是否使用了标准的普遍使用的库功能和子程序?  
4.程序中是否极少使用或根本不使用操作系统的功能?

六.效率

效率表明一个程序能执行预定功能而又不浪费机器资源的程度。  
这些机器资源包括内存容量、外存容量、通道容量和执行时间。  
用于效率度量的检查项目如下：  
 程序是否模块化? 结构是否良好?  
 是否消除了无用的标号与表达式，以充分发挥编译器优化作用?

七.可使用性

从用户观点出发，可使用性定义为程序方便、实用、及易于使用的程度。一个可使用的程序应是易于使用的、能允许用户出错和改变，并尽可能不使用户陷入混乱状态的程序。