# 软件工程的可靠性和可维护性

## 可靠性

随着社会经济和计算机技术的迅猛发展,软件作为提高工作效率的有效手段被广泛应用在各个行业。随之而来的是对软件质量及服务的要求越来越高,软件系统规模越做越大,越做越复杂,其可靠性越来越难保证。由于对软件的依赖程度越来越大,在一些关键应用(如民航订票系统、银行结算系统、证券交易系统、自动飞行控制软件、军事防御和核电站安全控制系统等)中使用质量有问题的软件,还可能造成灾难性的后果。

在20世纪80年代后,国究的投入越来越大,特别是进入20世纪90年代后,软件可靠性已经成为科学界关注的焦点,但软件可靠性的问题丝毫没有缓解。1983年美国IEEE计算机学会对“软件可靠性”做出了明确定义,此后该定义被美国标准化研究所接受为国家标准,1989年我国也接受该定义为国家标准。该定义包括两方面的含义:

a.在规定的条件下,在规定的时间内,软件不引起系统失效的概率。

b.在规定的时间周期内,在所述条件下程序执行所要求的功能的能力。

其中的概率是系统输入和系统使用的函数,也是软件中存在的故障的函数,系统输入将确定是否会遇到已存在的故障(如果故障存在的话)。软件可靠性是在一段时间内,软件正常运行的概率,它与操作有很大关系,是动态的,而不是静态的。软件可靠性R是指软件在规定条件下、规定的时间段内完成的预定功能的概率,或发生的概率。软件可靠性是关于软件能够满足需求功能的性质信息技术的飞速发展,使软件产品应用到社会的各个领域,软件产品的质量自然成为人们共同关注的焦点。质量不佳的软件产品不仅会使开发商的维护费用和用户的使用成本大幅增加,还可能产生其他的责任风险。软件质量不高首先是软件可靠性低,假如程序在执行时总是频繁地、重复地失败,那么其他软件质量因素是否可以接受也就无从谈起。计算机程序的可靠性无疑是软件整体质量的一个重要因素。

2软件可靠性的重要意义

20世纪70年代中后期以来,以软件工程的大力发展为契机,软件可靠性工程得以产生并取得了长足的进展,各种软件可靠性模型相继推出并得到不断改进和优化,模型验证和使用一度成为软件可靠性工程的热点,直到今天也依然是热门话题。软件可靠性设计与测试技术得以开发并逐步应用于工程实践;软件可靠性分析、评估方法不断完善,并在一些特殊的或重点工程项目中得到应用;软件可靠性工程管理技术的开发备受推崇,相应的管理方法被实践所验证,软件业界已充分认识到,绝大多数软件问题是由管理不善所引起的,所以,以过程改进、组织性能改进、管理模式改进、软件开发人员管理为重点的管理体系和管理机制得以产生并日臻成熟;软件可靠性标准化工作得到前所未有的重视,国际电工委员会的TC56技术委员会成立了软件可靠性工作组, 一些迫切需要的软件可靠性、维护性标准相继发布,为基础。目前,通过软件业界和可靠性工程界的不懈努力,软件可靠性工程得到了广泛的研究并不断实践取得了显著的成绩,但遗憾的是直到今天,开发足够可靠的软件并测试和验证其可靠性,仍然是非常困难的问题。复杂软件不管是对大工程系统还是小工程项目都越来越显示出它是一个薄弱环节,即使是通过完备测试与合格验证的软件也常常受到错误的困扰。与此同时,一个前所未有日益增长的需求是,软件应具有检定合格的可靠性,例如,武器装备系统、载人航天系统、核安全控制系统等无不对软件可靠性提出了前所未有的高要求。即使是在工业和日常生活中一般应用程序的开发与销售,市场对其可靠性要求也越来越高。如此，我们还不能保证软件可靠性水平,四十多年前就已波及到全世界范围的软件危机,直到今天依然是难以逾越的障碍。  
3影响软件可靠性的主要因素  
软件可靠性表明了一个程序按照用户的需求和设计的目标,执行其功能的正确程度。这要求一个可靠的程序应是正确的、完整的、一致的和健壮的。软件可靠性的决定因素是与输入数据有关的软件差错,正是因为软件中的差错引起了软件故障,使软件不能满足需求。  
影响软件可靠性的因素主要包括:  
3.1软件开发的支持环境。  
3.2软件的开发方法。  
3.3软件对实际需求表述上的符合度。  
3.4软件可靠性设计技术,软件可靠性设计技术是指软件设计阶段中采用的,用以保证和提高软件可靠性的为主要目的软件技术。  
3.5软件的测试与投放方式等。  
3.6软件的规模和内部结构(即软件复杂度),随着软件规模结构复杂度的增加,软件可靠性的问题越来越突出。  
3.7软件开发人员的能力和经验。  
4提高软件可靠性的技术手段和方法  
4.1加强测试。软件测试是保证软件可靠性的重要手段,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。

测试的方法多种多样,举例如下:

a.静态测试。代码审查主要依靠有经验的程序设计人员根据软件设计文档,通过阅读程序来发现软件错误和缺陷。代码审查虽然在发现程序错误上有一定的局限性,但它不需要专门的测试工具和设备,且具有一旦发现错误就能定位错误和一次发现一批错误等优点。

b.动态测试。白盒测试是一种按照程序内部的逻辑结构和编码结构设计并执行测试用例的测试方法。采用这种测试方法,测试者需要把握被测程序的内部结构。白盒测试通常根据覆盖准则设计测试用例,使程序中的每个语句、每个条件分支、每个控制路径都在程序测试中受到检验。白盒测试需要运行程序,并能在运行过程中跟踪程序的执行路径。黑盒测试是一种从软件需求出发,根据软件需求规格说明设计测试用例,并按照测试用例的要求运行被测程序的测试方法。它较少关心程序内部的实现过程,侧重于程序的执行结果。单元测试尽量采用白盒法,集成测试、系统测试则采用黑盒法。

4.2软件复用。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:

a.开发过程重用,指开发规范、各种开发方法、工具和标准等。

b.软件构件重用,指文档、程序和数据等。

c.知识重用,如相关领域专业知识的重用。

4.3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、制定质量标准度量。这个质量标准包括实现的功能、可靠性、可维护性、可移植性、安全性、吞吐率等等。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。  
质量标准度量包括:  
a.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、分析设计及可靠性治理措施等。  
b.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,文完成情况等。

c.测试结果质量度量。测试过程中差错状况,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。

d.验收结果质量度量。包括完成的功能数量、各项性能指标、可靠性等。通过以上的质量保证措施可以进一步明确划分各开发过程,在各开发过程中实施进度治理,产生阶段质量评价报告,对不合要求的产品及早采取对策。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。

a.重视人的因素,配备适当人员,明确职责,制定奖优罚劣政策。人的治理比较困难,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,使工作连续性难以保证,直接影响工作进程。  
b.采用先进技术和工具,并及时组织相应的培训,充分发挥先进技术和工具的作用。  
开发一个大的软件系统,离不开开发治理工具,仅仅靠人来治理是不够的,需要有开发治理工具来辅助解决开发过程中碰到的各种各样的问题,以提高开发效率和产品质量。使用先进的治理工具可以带来不少好处: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;治理软件多重版本;治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。

4.4采用适宜的软件开发方法。软件开发方法对软件的可靠性有重要影响。选用最适合的先进的软件开发方法(如结构化方法、面向对象方法、形式化方法等)和相应的软件生存周期模型,作为软件开发和组织治理的统一依据。目前的软件开发方法主要有Parnas方法、SASA 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。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,所以可视化方法和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。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。此方法在设计中要求先列出将来可要变化的因素,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。采取的一些具体措施如下:

1. 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。
2. 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。
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。所以在开发过程中可以采用面向对象的方法,Parnas方法,再结合使用其他方法,吸取其它方法的优点,从而提高软件的可靠性。

软件可靠性问题对于安全关键软件而言具有非常重要的意义,目前软件工程主要通过软件的可靠性。没有软件工程,便没有软件可靠性。软件可靠性必须贯穿于软件的整个生存周期,必须在认真实施软件工程的基础上,专门开展一些软件可靠性的技术和治理活动,系统地考虑软件生存周期全过程。通过采用正确的方法、技术和工具,以得到一个错误少、可靠性高的软件。

## 可维护性

软件可维护性即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。软件的易维护性差是软件维护工作量和费用激增的直接原因，因此在软件工程的各个阶段都要保证软件具有较高可维护性，从而降低软件维护成本，这是软件工程的重要目标之一。本文简要地介绍了软件可维护性的概念， 软件可维护性的方法。  
　　软件维护性是软件的内在固有属性，它是软件质量的重要组成部分。软件质量是软件产品中能够满足给定需求的各种特性的总和，这些特性包括：功能度、可靠性、维护性、可重用性、易用性、安全性等。软件质量，对开发者来说就是优良的设计或代码，对使用者来说就是响应迅速、功能强大，对维护人员来说就是易于升级、修改，对管理者来说就是较低的开发成本以及合理的开发时间。在软件的质量特性中，维护性与可靠性、可重用性的关系尤为密切。软件是武器装备系统不可分离的重要组成部分，软件维护不仅与维护性有关，而且涉及到软件的快速分发、软件的鉴别、软件与固件的供应等问题，但是较高的维护性是实现软件维护的关键。  
　　软件维护的类型有四种：改正性维护、适应性维护、完善性维护、预防性维护，下面我们一一介绍：  
　　1 改正性维护  
　　在软件交付使用后，因开发时测试的不彻底、不完全，必然会有部分隐藏的错误遗留到运行阶段。这些隐藏下来的错误在某些特定的使用环境下就会暴露出来。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，应当进行的诊断和改正错误的过程就叫做改正性维护。  
　　软件测试不可能找出一个软件系统中所有潜伏的错误，一次，改正性维护是在软件运行中发生异常或故障时进行的。然而，对所发现的程序错误进行修改，一般都应该十分谨慎，以防造成不良后果。改正性维护的工作可能是：  
　　改正原来程序中并未发现的错误；  
　　解决开发是未能测试各种可能条件带来的问题；  
　　解决原来程序中遗漏处理文件里的最后一个记录的问题等等。  
　　改正性维护主要是针对开发人员和测试人员，因此开发人员和测试人员能进行很好的沟通和交流的话，可以提升改正性维护的效率，减少错误的发生，这也需要测试人员中提供测试数据的时候尽可能的想到如何提供更好、更恰当、更多的数据来进行测试。但是人思考的东西毕竟有限，因此有些奇怪的我们不能想到的数据也是很正常的，这也是改正性维护存在的必要性。  
　　2 适应性维护  
　　随着新的计算机硬件系统的不断发展，新的操作系统或操作系统的新版本的不断推出。外部设备和其他部件也经常修改和改进。在使用过程中，外部环境（新的硬、软件配置），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。为使软件适应这种变化，而去修改软件的过程就叫做适应性维护。  
　　适应性维护就是要使运行的软件能适应外部环境的变动。适应性维护工作可能是：  
　　为现有的没偶个应用问题实现一个数据库管理系统；  
　　对某个指定编码进行修改，例如从3个字符改为4个字符；  
　　缩短系统的应答时间，使其达到特定的要求；  
　　修改两个程序，时他们可以使用相同的记录结构等等；  
　　修改程序，使其适用于另外的终端。  
　　适应性维护的可能性非常的多，在这瞬息万变的环境中，软件需要不断的来适应整个环境来实现这个软件的应用价值。  
　　3 完善性维护  
　　在软件的使用过程中，用户往往会对软件提出新的功能与性能要求。为了满足这些要求，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。  
　　当一个软件系统投入使用和成功运行时，用户会根据业务发展的实际情况，提出增加新功能、修改已有功能以及一般的改进要求等等。虽然这些内容在需求说明书中并未规定，但是，为了扩充原有系统的功能、提高原有系统的性能，满足用户的实际需求，这项工作是必不可少的。  
　　4 预防性维护  
　　预防性维护是为了提高软件的可维护性、可靠性等，为以后进一步改进软件打下良好基础。预防性维护定义为：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、编制和测试。  
　　维护人员不要单纯等待用户提出维护请求，应该选择那些还能使用若干年、不浅虽能运行但不久就须作重大修改或者加强的软件，进行预先的维护。  
　　在整个软件维护阶段所花费的全部工作量中，完善性维护占了几乎一半的工  
　　作量。软件维护活动所花费的工作占整个生存期工作量的70%以上，这是由于在漫长的软件运行过程中需要不断对软件进行修改，以改正新发现的错误、适应新的环境和用户新的要求，这些修改需要花费很多精力和时间，而且有时会引入新的错误。  
　　5 软件维护的策略  
　　5.1 改正性维护：通常要生成100%可靠的软件并不一定合算，成本太高。但通过使用新技术，可大大减少进行改正性维护的需要。这些技术包括：数据库管理系统、软件开发环境、程序自动生成系统、较高级（第四代）的语言。以及新的开发方法、软件复用、防错程序设计及周期性维护审查等。  
　　5.2 适应性维护：这一类维护不可避免，但可以控制。  
　　5.2.1 在配置管理时，把硬件、操作系统和其它相关环境因素的可能变化考虑在内。  
　　5.2.2 把与硬件、操作系统，以及其它外围设备有关的程序归到特定的程序模块中。  
　　5.2.3 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。  
　　5.3 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，就可以减少以后完善性维护的需要。  
　　5.4 合理控制维护成本。有形的软件维护成本是花费了多少钱，无形的维护成本有更大的影响：一些合理的修复或修改请求不能及时安排，使得客户不满意；变更的结果引入新的故障，使得软件整体质量下降；把软件人员抽调到维护工作中，干扰了软件开发工作。  
　　软件维护的代价是降低了生产率，在做老程序的维护时非常明显。例如，开发每一行源代码耗资25美元，维护每一行源代码需要耗资1000美元。维护工作量包括生产性活动（如分析和评价、设计修改和实现）和“轮转”活动（如力图理解代码在做什么、试图判明数据结构、接口特性、性能界限等）。  
　　维护工作量的模型：  
　　M=p+Kec-d  
　　其中M是维护中消耗的总工作量，p是上面描述的生产性工作量，K是一个经验常数，c是因缺乏好的设计和文档而导致复杂性的度量，d是对软件熟悉程度的度量。  
　　模型指明，如果使用了不好的软件开发方法（未按软件工程要求做），原来参加开发的人员或小组不能参加维护，则工作量（及成本）将按指数级增加