软件的开发、质量管理及可靠性

Development、Quality Management and Reliability of Soft Ware

何国伟

（中国航天工业总公司　研究员　　北京　100830）

提要　我国的软件开发还处于初期阶段。其特点是自编、自导、自演，不仅质量与可靠性低，将来的维护与扩展费用也将不堪承受。为此，迫切需要把我国的软件开发转向设计、生产、检验分工的工业化生产，象对硬件一样对软件实行严格的质量管理，按科学规律顺序地按阶段进行开发，并狠抓软件的可靠性。

我国的计算机软件（以下简称软件）开发还处于计算机时代的早期阶段。其特点是自编、自导、自演，这是前工业革命时代（即个体生产及作坊时代）的生产组织方式。绝大多数软件鉴定会上的所谓"测试"，只是对几个预先指定的用例进行一下表演就算通过，几乎根本起不到软件测试应有的作用。

人在工作中不可能不出差错，依靠自我检查一般也不可能100%地纠正差错。科学管理的创始人泰勒提出了"定额"、"奖金"，以及建立专职检验机构、配备专职检验人员的"检验体系"。独立的检验体系是保证硬件质量的必要体制。而软件开发还没有建立起独立的检验体系，因此软件的质量缺乏基本的组织体制保证。交付使用的软件质量、可靠性不高，存在严重的隐患，有的损失以百万元计。

在研制生产硬件时，设计人员出设计图纸，工人按图纸生产出产品，检验人员检查产品是否满足要求，如果满足，就交付使用，设计、生产、检验三权分立。这里有设计人员与工人的脑力劳动与体力劳动的分工（这种分工可以提高劳动生产率），同时，设计人员的成果——设计图纸，既是当前生产的依据，亦是今后修改设计、改进设计的依据。可以设想，如果一台机床没有设计图纸，若要修正错误，或改进设计几乎是不可能的。国内软件开发的情形正是如此。很多软件没有"详细设计报告"等全套软件技术资料，有的甚至只有一本源代码，要从源代码读懂软件确实像读天书。这样，只有源代码的原编者才能对软件进行维护、修改、扩展。事实上，时间一久，就连原编者自己也记不清详细设计了，结果只好重编。软件的寿命不长，可靠性不高，维护费用高，这是"自编、自导、自演"带来的严重后果。这样的软件不能算是工业品，只能算是工艺品。印度一年有几亿美元的软件出口，但我国的软件却进不了国际市场，根本原因是我国的软件还够不上工业品。

历史的教训必须重视。据国外统计，1955年，在计算机成本中，硬件占80%，软件开发占10%，软件维护占10%；到了1985年，硬件只占10%，软件开发占30%，软件维护及扩展占60%。以美国国防部1991～1992年度预算为例，在一年约3 000亿美元的国防费中，软件费用约为300亿美元。其中软件开发约为90亿美元，而软件的维护扩展达约210亿美元。美国的软件水平远比我国高，目前已感到软件质量可靠性不高带来的维护扩展费用包袱愈来愈沉重。我国如果再不认识这个问题的严重性，软件的质量可靠性将会拖住我们发展的后腿，背的包袱将比现在的美国要重得多。

在我国的计算机研制开发中有一个相当普遍的错误认识："在计算机的可靠性设计中，设计的是硬件可靠性"。其实质是把软件可靠性看成是100%。这是不符合实际的。美国国家航空和航天局（NASA）失效分析实验室的基恩（S. J. Keene）在1992年1月发表的文章中指出，当前NASA系统的软件故障率是硬件故障率的10倍。美国在海湾战争中使用了新式飞机F18，它的飞行控制计算机在2万飞行小时中出现的500多次故障，软件故障远超过50%。正因为如此，美国对软件质量可靠性相当重视。1991～1992年度美国防部关键技术研究共21项，总投资45.34亿美元，其中集成电路为4.79亿美元，而软件研究（主要是质量及可靠性研究）为1.49亿美元。注意两者并无数量级的差别。相比之

下，为提高我国的计算机水平，在集成电路上有一定的投资，但对软件质量及可靠性研究的投入却少得极不相称。这是决策部门对软件质量可靠性的意义及重要性认识不足造成的。如不及时转变，会在不太久的将来吃大亏。

一、尽快在我国实现软件的工业化生产

将我国软件生产的个体方式转变为工业化方式，是我国软件开发面临的首要任务。软件设计人员应提供软件的详细设计报告等全套设计资料。有了这套完整的设计资料，其他设计人员在需要时就可以进行修改或扩展。合格的软件工人（即编程员，高中毕业生经一两年培训即可胜任）按详细设计报告编程。软件检测人员检测软件的设计、编程是否符合使用方的要求。实现设计、生产、检测的分工，既可提高软件生产效率，更重要的是提高了软件的质量。

软件的工业化生产不是一蹴即就的，还须创造一些条件。首先要转变软件人员的传统观念，认识到放弃传统的个体或作坊式生产方式，学习并掌握不熟悉的工业化生产方式的必要性，从而自觉地完成这个转变。硬件必须有全套完整的技术资料，软件也应如此。按照国军标 GJB438－88《军用软件文挡编制规范》（国标相同）的要求，一项软件产品至少应提供十三类文挡（即技术资料）。这样做当然增加了工作量，但却可以大大减少将来软件维护、扩展的工作量，提高软件的质量。

软件是人头脑的产物，它有没有问题，从开始对软件构思就要进行检验。因此，软件检测人员要从软件用户对软件提出需求起就参与软件的设计、编程工作，及时检测，及时发现错误与缺陷。一般来说，软件检测人员的设计水平不应比设计人员低，而且要有长期检测纠错的实践经验。我国还没有专业的软件检测队伍，应及早抽调能干的设计人员转为软件检测人员，这是一项十分紧迫的任务。

硬件的设计需要很多手段，例如 CAD、CAM 等工具。硬件检测需要很多设备、仪器和工具，有时需要大量经费。对此，领导一般都是理解的。但是软件的设计、检测也需要各种手段，这一点却是很多领导所不理解的。例如，上述十三类文挡的工作量就很大，但是与硬件 CAD 类似的《编文挡的软件》就可以大大节省工作量。又如《软件的检测用软件》可以大大节省检测工作量。这些工具品种、规格、类型繁多，国外已有成千上万，国内最近公布的“青鸟”就是这样的一种工具包（当然还需在使用中不断完善化）。应该根据需要引入或开发这类辅助软件设计及检测的软件。但很多部门领导还未理解其重要性，八五计划中它并没有占应有的重要位置。

正如硬件的标准件（包括标准电源、标准机架、标准系列电机等标准设备）在硬件设计生产中占有重要地位一样，软件标准件在软件设计中同样占有重要地位。大家知道，产品的标准化程度（以标准化系数来描述）愈高，一般说来，它的质量与可靠性就愈高。软件产品更是如此。如很多科学计算都要解常微分方程。如果有一个解某种常微分方程的标准软件，就可以减少大量重复编程的工作量。但是编出这种经充分检测，不存在错误、缺陷的标准软件是要化很大投入的，且只能由国家或国防部门来统一规划组织。美国国防部已提出，军用软件标准化系数应达到 50～60%。如果我国的软件标准化系数也能达到 50%以上，则相当于软件工作量可以翻一翻，其意义极为重大。为此，国家或国防部门即使投入相当人力、物力，也还是值得的。

在软件设计、检测、标准件、工具开发等工作中，我们不能忽视面临的语言问题。总不能够 Fortran，Algol，Pascal，C，Basic，…每种语言都开发一套，这样的投入将是不堪重负的。因此，有必要选定一种相对来说质量与可靠性较高的语言。以计算机的创始人 Babage 的助手 Ada 命名的 Ada 语言，在美国国防部的大力开发下，质量与可靠性确实达到了很高的水平，并已成为美国及北大西洋公约国军用标准语言。笔者建议，我国的软件界转用 Ada 为唯一的语言。几千年前，秦始皇下决心搞标准化，“车同轨，书同文”，对中华民族统一及发展起了极大的促进作用。在计算机语言统一上我国也到该下决心的时候了。当然，这又是对旧习惯的冲击。我国大学中极少教 Ada 语言，各部门目前也几乎不用 Ada 语言的。但愈早统一，震动愈小，过渡也愈容易。

二、软件的质量管理

在软件产品转为工业化生产后，对软件工厂必须像硬件生产一样进行严格的质量管理。包括组织落实及规范化管理。

“组织落实”就是要“建立并保持一个通过文件加以规定的质量体系”（国际标准 ISO9000－3:91）。我国很多单位已建立了质量体系。产品质量体系的国际标准 ISO9000 对产品的含义原来就很明确，包括：硬件、软件、流程型材料及服务。把软件的质量管起来本来就是质量体系的责任。但我国的质量体系基本上都是针对硬件建立的，因此对现有的质量体系人员应进行培训，使他们掌握 ISO9000 关于软件质量管理的要求。

国际标准指出，软件的“设计和实施活动是将需方要求规范转换为软件产品的过程。由于软件产品的复杂性，必须以严格的方法进行这些活动，以

便生产出质量合格的产品”。这些“严格的方法”由质量体系制定成规范，并监督其在软件开发过程中切实得到实施，这就是“规范化管理”。应当指出，获得高质量软件的设计方法不一定没有争议，例如对用不用 GOTO 语句就是如此。但多年实践证明，GOTO 语句很少出现在质量好的程序中；有了 GOTO 语句的程序增加了检测的难度，检测结果也证明它易出差错；又因 GOTO 语句可用标准的结构化构件代替（只是有时稍复杂一些），因此规定不用 GOTO 语句就可以作为规范的一项。

保证软件产品开发质量的规范有很多内容，主要有下述几个方面：

1. 要按系统工程的要求逐级结合硬件开发软件。

在现代的复杂系统中，已大量使用计算机。目前很多系统采取的开发方法是：各分系统、整机需要用计算机实现其功能时，就各自配置自己的计算机及软件。于是从整个系统来看，各部分的那么多的计算机是各自为政的，整个系统的计算机硬件与软件没有一个总体设计，并且软、硬件之间的协调往往配合不好，还易出疏漏。对于一个复杂系统来说，信息的获取、传输及硬、软件加工组成的信息分系统是一个关键性的分系统，从系统工程的功能分析的角度看，要求从分配开始，直到综合、权衡、评价、决策都需要把硬、软件结合起来进行。

2. 要按科学的规律顺序地按阶段开发软件。

美国国防部 DOD－STD－2167A 规定按瀑布模型（Waterfall model）的顺序按阶段开发软件。即：系统需求分析→软件需求分析→概要设计→详细设计→编程及单元（CSU）测试→模块（CSC）组装及测试→软件配置项目（CSCI）测试→系统综合和试验。国外绝大多数软件工厂都采用了这种开发模式。实践已经充分证明，如果超越科学规律的阶段顺序去开发软件往往带来混乱，事倍功半。

3. 质量体系要制定软件质量保证大纲，并根据软件具体情况制订质量保证计划。

此项计划中应规定为保证软件质量必须进行的工作项目，它们的要求、工作内容、监督及评审点、进度、完成的方式、负责人及资源，等等。有的软件是由分承制方开发的。分承制方软件的质量管理要求与承制单位应当相互一致。

4. 质量体系制订的软件开发过程中应贯彻执行保证软件质量的标准、条例、规范、规定及监督措施。

软件总体要规定软件的详细设计准则。有的可以引用军标或国标，例如参照美军标制定的国军标 GJB437－88 规定：“必须用系统的自顶向下的方法将软件的需求转换成软件设计”，并规定了实行结构化编程“必须使用五种结构进行程序设计和编码”。这五种结构是顺序（sequence）结构、条件转移（if then else）结构、当循环（do while）结构、直到循环（do until）结构、分情况（case）结构）。

5. 设计评审是把好软件质量关的重要手段。

软件主要是人头脑的产物，软件的质量主要取决于人的设计思想正确与否及实现设计思想过程中有无疏漏。因此，组织非本系统的同行专家在各个阶段末尾对设计进行评审，指出设计缺陷、错误及改进方法极为必要。在系统需求分析（功能分析及功能分配）后进行系统需求评审（SRR），在系统设计后进行系统设计评审（SDR），在软件需求分析末了进行软件规范评审（SSR）（包括软件验证及确认计划评审），在软件概要设计阶段末了进行概要设计评审（PDR），在详细设计阶段末了进行关键设计评审（CDR），在正式验收测试前进行测试准备状态评审（TRR）。未经评审通过不能进行下一阶段工作。此外，对硬、软件的技术状态项目（CI）都必须进行功能技术状态审核（FCA）及物理技术状态审核（PCA），最终是正式合格鉴定评审（FQR）。上述美军 DOD－STD－2167A 的一系列规定已被软件工厂广泛采用。

严格按照详细规定的设计准则进行设计，严格按照设计评审规范规定的审核表（Checklist）进行评审是保证软件质量的根本措施。

6. 对软件的技术状态管理（即配置管理）必须严格。

众所周知，对硬件来说，如果对某一零、组、部件进行修改，而对上下左右的协调配合没有足够注意及分析研究，极易产生产品故障，因此对硬件产品的技术状态管理必须是严格的。软件比硬件好改，但因此而出故障的可能性比硬件还大，从而软件的配置管理应该比硬件还要严格。

三、软件产品的可靠性

我国的软件生产基本上仍处于个体及作坊式生产时代，或者说还处于所谓“混沌阶段”。按国外类似阶段软件的统计数据，这一时代水平的软件在开发阶段每千行代码平均有 50～60 个故障，交付后每千行代码平均还有 15～18 个故障。这种可靠性水平显然是相当低的。如果我国的软件开发走上工业化的轨道，软件工厂在严格的质量管理下开发软件，根据国外类似阶段软件的统计数据，这一水平的软件在开发阶段每千行代码的平均故障为 20～40 个，交付后由于严格的检测降为 2～4 个。软件的可靠性几乎可以提高一个数量级。但这种可靠性水平还不能满足高可靠性的要求。

因此需要在软件开发中运用一系列可靠性管理、可靠性设计、可靠性测试技术，使得软件在开发

阶段的平均故障降到每千行代码0～20个，交付后降到0～1个。大体上，这相当于在交付后大约运行10^4～10^6小时出一次故障。

在软件可靠性方面处于最高水平的是美国IBM公司为航天飞机开发的机载软件，它是IBM公司总结了历来软件可靠性成功及失败的经验教训后，采取了一系列严格的可靠性技术后开发出来的。在开发阶段每千行代码平均有6～12个故障，而交付后则为每千行代码仅0.1个故障。但高可靠性是通过大量工作的高投入得到的。IBM公司为这一机载软件的50万行代码，投入了5亿美元，合每行1 000美元，相当于美国民用软件开发费用每行不到10美元的100倍。大体相当于宇航级(S级)电子元器件与民用电子元器件的价格比。由此我们应该建立起一个概念，高可靠元器件是很花钱的(很多领导对此都能理解)，高可靠软件同样要投入大量资源，也是很花钱的(很多领导对此还很不理解)！

软件可靠性设计已经有了迅速的发展。有不少软件可靠性设计技术是从硬件可靠性设计移植过来的。例如，用来确定软件关键件、重要件的重要分析方法"软件的故障模式后果分析(SFMEA)"及"软件故障树分析(SFTA)"就是如此。在移植或借鉴硬件可靠性设计方法时，应结合软件的特点，不能生搬硬套。例如，硬件可以将几个相同的部件作可靠性并联以提高可靠性，这是硬件中最基本的"冗余设计"技术。但对软件而言，将相同的几个软件并联使用是徒劳无益的。因为相同的软件有相同的潜伏故障，一错俱错，提高不了可靠性。因此，软件的可靠性并联必须用相互独立的所谓"多版本"软件来实现冗余。如何达到多版本有很多技巧。一般来说，找两个人来编，编出的软件不见得是独立的。

在硬件可靠性设计中，"简单就是可靠"是一条设计原则。在软件中这条原则更为突出。只有简单，才易于检测、评审，不易出错，出了错也易于发现与排除。软件设计中什么叫"简单"，随不同设计阶段、不同设计对象而定。软件设计的结构化及模块化无疑是两种使软件简单化的途径，在质量管理中已定为规范。已经对软件不同对象提出了几十种简单程度的度量。可以根据具体情况选用若干个作为设计准则。软件简单程度度量最早是霍尔斯特德(Halstead)于1970年提出的所谓"软件科学"。尽管后来不少人提出不同意见，但对编码阶段而言，该度量仍是可参考使用的。又如麦凯布(Mc Cabe)度量，它定义软件复杂性为图的基本路径数，并规定不超过10。诸如此类。这些度量可以控制住设计的复杂性，从而提高可靠性。

软件不仅要"对正确的输入得到正确的输出"，还需要"对不正确的输入能有正确的响应"。这就是说：软件要有"顽健性"(robstness)。例如，"从已知三角形的三边求三角"。如果输入的两边之和不大于第三边，计算机是算不出结果的。此时，计算机不应该停机，而应指出输入有错。当然，"不正确的输入"应有一定的范围。如果漫无边际，那软件也是难于应付的。

对于硬、软件结合的系统，有些硬件的故障可以用软件来弥补。例如，在宇宙空间工作的计算机，在宇宙空间重粒子轰击下，在寄存器及存贮器中有可能出现突发性错码。利用信息论中现成的检错、纠错理论，软件可能发现及纠正这种硬件故障。同理，有些软件故障可以用硬件来弥补。这就是所谓"容错、纠错"技术。

高可靠软件的管理还没有像硬件那样形成标准(IEC有一个草案，是比较简单的)，但已经有一些成功的方法可资借用。IBM为开发航天飞机机载软件发展的洁净室(clean room)方法就是一种有效的可靠性管理技术。它强调对工作质量进行统计质量控制(SQC)。用高工作质量来保证软件质量。它还建立了一系列控制软件质量可靠性的办法，都是值得采用的。

检测是发现及纠正软件错误的重要手段。每一个软件的可能输入是输入空间的一个点。试遍输入空间所有的点当然可以检测出软件的所有错误，但由于工作量太大，是不可能实现的。因此现实的作法是双管齐下，一方面用白盒子及黑盒子测试技术从单元起，测试软件的正确性；一方面把输入空间分为若干典型子空间，在每一个子空间选若干个输入点作为测试用例。另外还用广泛的强化测试用例及广泛的非正常输入测试用例来考验软件。

在可靠性的定量评估方面，目前还不象硬件那么成熟。已经总结出了几十种模型。正因为出了几十种模型，所以没有一种模型是通用的。看来，稳定生产的软件工厂的产品会有一定的可靠性模型，但不同软件工厂的模型不尽相同。究竟适用那一种或几种模型，需要软件工厂自己从实践中去总结。模型一定，分析评估是数理统计学不难应付的。

美国国防部委托卡内基--梅隆学院的软件工程研究所(SEI)提出了考核软件工厂水平的办法。把软件工厂分为五级。据到1990年10月为止的统计，已考核过的美国软件工厂，74%为一级，22%为二级，4%为三级。达到五级的只有一项，即IBM开发的航天飞机机载软件。我国的软件生产单位恐怕连一级还不够。但应该按这种分级标准制定上等级的计划，例如，2～3年上一级应该是可能的。愿我们软件界共同为软件工厂上等级而努力，比如在2000年左右，能出现一批三级以上的软件工厂，承担起关键性的软件生产任务。　(责任编辑　孙立明)