计算机科学1993Vol.20№.4

九十年代软件工程展望

桑 杰 编译

信息技术的发展正促进着软件成为一门独立的工程学科。九十年代，整个社会同信息处理密切难分。随着用户市场的不断扩大，家庭对应用程序的需求量与日俱增，软件质量问题突出了出来。可以说，九十年代是软件质量年代，是软件工程向标准化、定量化发展的关键时期。

一、软件质量

首先，我们来讨论一下什么是软件质量，什么是达到这种质量要求的软件工程，提高软件质量必须有哪些过程，以及如何实现这些过程和技术。

质量，不是一个单一的概念，它是受多种因素制约的复杂概念。这些因素包括人们所关心的事物、对该事物的看法和该事物的质量属性。这里所指的事物又包括最终可以提供的中间产物（如诸条件的说明文件）以及过程的组成部分（如设计阶段）。对特定事物的看法，一般来自最终用户、发展中的机构和项目管理部门。例如，从设计人员的观点看，可读性就是一个需求文件的重要质量属性，从项目管理部门的角度看，逝去的时间就是设计阶段的重要质量属性。为使这些质量属性有客观的量度，对它们进行量化是非常必要的。为此，我们把软件质量问题又划分为软件产品本身的质量和研制过程的质量来讨论。

产品质量。用户满意，是任何一种产品的最终质量目标，软件产品当然也不例外。我们定量描述的对象，就是满足用户各种要求的产品的最终属性。人们用来评价一种产品的最常用的质量属性有：产品的功能、可靠性、价格及其研制周期。其中最重要的是可靠性。功能，是任何一种产品在其设计初期就已确定的较为稳定的一种属性。用户在选择产品时就已经根据自己的需要，选定有相应功能的产品。所以，上述四种质量属性又可以归结为后面三个属性。质量的等级，就是这些属性接近用户要求的程度。

软件产品的这些质量属性不是相互孤立，而是互相影响和制约的。我们用指定时间内软件产品无故障运行的概率来表示可靠性，称之为故障率（记作F）。这样，我们用下列公式来表示质量因数Q：

$$Q=1/(F\times C\times D),$$

式中F表示故障率，C表示软件产品价格，D表示软件研制周期。虽然，这三个量之间的实际关系也许比取其乘积要复杂些，但这种质量因数却能反映软件产品的质量指标。譬如说，故障率低（即可靠性提高），就要求提高价格或延长研制周期，或二者兼有。

过程质量。与软件产品的质量目标相适应，应当有一种面向质量的研制过程。我们可以把这种过程看成是由若干阶段组成的，每个阶段都有反馈途径。每个阶段，都有一个中间供应者研制出一种中间产品，提供给中间用户，再转入下一个阶段。每个阶段也收到前一个阶段提供的一种中间产品。每一

MIDI(A)，A′处于A和非A之间，难以区分使用哪种翻译规则更优越。

由以上分析，可以得出这样的结论：相比之下，R,是比较优越的；并且使用表1中所列的所有翻译规则，都可得出符合古典假言推理的结论，这些翻译规则都是基本合理的。

鸣谢 作者在撰文过程中得到黄叔武教授的悉心指导，在此深表感谢（参考文献略）

种中间产品都具有某种中间质量属性，这些属性影响最终产品的质量属性，但不一定和它们相同。举例来说，在设计阶段，设计人员就是需求说明的用户，他们研制系统体系结构和单元的技术说明，并在一种设计文件中对这些结构和技术说明下定义。这种文件就是设计人员的中间产品。可读性和完整性，则是设计文件的重要质量属性。

软件产品的研制过程，不仅具有中间产品的若干个阶段，而且应当被看做是某个社会团体对事物的半结构化的认识活动。人类的认识过程和社会动态反过来影响软件产品的质量。

描述研制过程，需要有研制过程的模型、衡量这些模型的各种特性的尺度以及获取这些尺度的切实可行的办法。还需要在这些尺度和最终软件产品的质量属性之间建立起某种关系。这样，我们才能够把握研制过程，使其满足质量属性目标。例如，要设计出一种可靠性高的软件产品，应当采取什么方法？而价格和研制周期又应当留有多大的范围？等等。

最后，我们需要两种模型：一种是用户使用该软件系统情况的模型，另一种是各种应用程序使用的相对临界状态(Criticality)模型。

二、软件工程的模型与衡量尺度

纯硬件系统设计与实现的定量方法早已问世，其设计规程、价格估算和可靠性技术，早在六十年代就相当完备了。软件在这方面的发展就滞后20至30年。造成这种局面的原因是，人们对软件本身的开发以及对硬件与软件工程之间的本质差异（例如生产和开发的差异）理解不深。

要想在各种质量属性之间求得最佳平衡，以满足用户需要，就必须建立软件研制过程和产品的模型、衡量尺度和管理方式。弄清楚时间和精力花在什么地方，弄清楚一种可靠性好的产品的质量属性会有什么样的研制过程，我们才可以使质量属性的模型更加精确，才可以使过程和产品之间的关系更加客观。

为此，须对软件工程的研究范围分门别类，确定各个成分的表示方法，规定这些成分在使用中的相互关系。软件工程的研究范围，包括各种过程和这些过程的组成部分、软件产品以及其它形式的经验。

首先，需要建立各组成部分的描述性模型，以便更好地理解各过程和产品的本质和特征、这些过程和产品之中的变化及其优缺点，以及预测和控制这些过程和产品的途径。

目前，有些部分已经有了模型。例如，有些程序和模块具备若干数学模型，诸如谓词演算、函数和状态机器。又譬如软件价格、软件可靠性、软件生命周期等方面也有一些模型、有的已从研究和开发阶段进入应用阶段，它们正在构成软件工程定量化的基础，推动软件工程从技巧性向科学化迈进。

其次，应当在分析这些描述性模型的基础上建立规定性 (Prescriptive) 模型，改进软件产品及产生这些产品的过程。这种规定性模型必须把各种质量属性联系起来。建立这种模型时，必须考虑到为控制设计过程提供反馈信息，学会吸收成功的经验。

未来十年，对于深入理解和达到软件质量要求起重要作用的几个技术领域将是：

1.形式方法

形式方法，是计算机科学工作者为了更好地理解软件产品本身并使软件功能抽象化，在数学形式方法基础上研制出来的软件产品模型。这些形式方法包括谓词演算、函数和状态机器，这些方法取得理论价值已有多年，只是还未有效地付诸实用，主要是由于不能把它们扩大到适当规模的系统上。目前，形式方法在软件研制中开始见到若干实际应用。这些方法以软件研制的正确性为目标。

2.设计方法

八十年代，引入面向对象的设计方法、

技术和语言，软件设计出现重大突破。九十年代，这种方法会继续影响软件设计。预计，面向对象的软件设计技术，在确定集成化支持环境的过程中起重要作用。用对象来管理系统和设计系统的这种概念，比较好定义，而且将改变人们对系统的思考方式。象功能分解法、面向对象的设计方法，将成为软件工程师智能工具的一部分。在可重用软件设计中，智能工具早已占有重要位置。

3.程序设计语言

九十年代，凡是支持面向对象的设计与面向对象的程序设计的那些语言，例如Ada、、目标C、C++、Smalltalk语言，整个地或局部地续继得到发展。表示方法也将发展，使诸如需求和规格说明这样的较高级抽象概念得以形式化。这些程序设计语言的级别愈高，愈有可能变成面向应用的语言和专用语言，例如第四代专用语言。还将出现效率更高的翻译程序，将这些高级语言翻译成可执行形式。这些表示方法将成为软件设计过程中的基础工具。

4.测量方法

测量方法是和建立模型分不开的，各种测量结果都应当以相应的模型为依据，反映这些模型是否符合要求。以往的测量是着重度量标准，而不注意模型，也就是说，这样收集到的数据没有明确的目标，也没有明确的模型，来龙去脉也不清楚。

现在有不少测量方法，以模型为依据，围绕目标来实施。这些方法把软件过程模型、产品模型和质量模型同目标融为一体，使这些目标和模型适应于特定的要求。举例来说，假如我们的目标是评价一个系统检测方法查出故障的能力，那么你选择的检测过程的模型和检测故障的模型是好用的。还要注意收集和综合有助于说明问题的信息，例如，所采用的测量方法的效率如何？进行测量的人对需求条件理解的程度如何？系统测量结果与相似的项目相比较，失误率是多少？等等。

一些确定可测量目标的方法也已问世，它们包括目标-问题-测度范例、质量函数使用法以及软件质量测度法。预计这些方法将来会得到更多利用。

5.软件的使用及简化操作软件

软件的使用将左右软件的发展。操作简表（Profile）、预期的用户操作集以及这些操作出现的概率，一般同时被确认为软件系统的需求。有些操作其功能相类似，只是各自的环境不同。有些操作按其适合场合的临界状态加以分类。这样分类以后得到的操作简表，就成为确定整个软件研制过程的优先级和工作量分配的依据。

不久的将来，极有可能出现简化操作软件，它类似于现在的简化指令集计算（RISC）。这种预测的根据是：大多数软件，它的一些操作是常用的，而许多操作却非常用。不常用的操作占去大量的研制时间、文件篇幅和维护费用，而且还使系统复杂化，用户培训起来也有很多困难。简化操作软件就可以尽量避免执行许多不常用的操作，常常用一系列更基本更常用的操作来替代那些不常用的操作。这些基本操作可以由系统和软件设计人员来设计，也可以由用户自行决定，视各自的需要而定。

6.重用

以往，重用只限于代码级，而且主要靠个人的经验。近来人们对重用的兴趣和重用技术的发展都十分活跃。人们能够而且必须重用全部软件经验，但问题是，重用一个对象就要求重用与此有关联的其它对象。例如，分面（faceted）模式、模板和搜寻策略，就是随着可重用的软件成分一起发展起来的。有些对象是为特殊项目而设计的，因此，要想重用它，必须对它的重用可能性加以评估，而且必须建立能够重用的过程。

未来十年，随着人们更深入地理解重用的内涵，随着后援技术的发展，重用技术会有更大的进展。面向对象的设计方法会使重用技术更容易实现。

7.认知心理学

软件工程说到底是一种解决问题的活动。认知心理学，则是研究解决问题方法的一门学问。人们可以利用认知心理学来研究软件研制过程中人的各种智能活动。目前，具有这两方面训练的研究人员不多，因而很少有人把认知心理学应用于软件工程。这两种领域之间存在文化上的差异，相互渗透有些困难。一般软件研究人员目光盯在控制规范和逻辑上；认知心理学工作者着眼于发现人类智能过程中的缺陷和弱点。

软件工程在应用认知心理学方面，普遍把注意力放在人-机接口上，体现在计算机辅助软件工程工具之中。这种面向工具的研究工作肯定会继续进行下去，但是人们的注意力很有可能更多地放在独自解决各自的内部问题的努力上，更着重研究各种方法论和环境要素，以提高这种努力的质量和效益。

软件研制人员对应用部门的认识程度，是他们的工作质量千差万别的主要原因。可是，人们常常认为软件研制工作是个独立的门类，它既可以抽象化又可以无师自通，这是不符实际的。事实上，程序设计人员应当在某种程度上专门化。应当注重组织、宣传和提高专门领域知识的教育，不管是正规的还是在职的。以往那种"闭门造车"是行不通的。一位优秀的软件设计人员，应当具备精良的社会交往、谈判、组织和其它技能。他应当对其所从事的系统的用途和结构有清楚的了解。他应当结交一批专家并形成一个专家网。在大学里就应当设置这方面的教程，到了工作岗位应当继续下去。

8.软件社会学

大多数软件研制工作都是一种集体行为，涉及集体动态特点、通讯网络和组织策略等各种复杂事物，也就是说，软件工程具有社会性。软件研制过程中的集体行为，将对软件质量和软件生产率产生极大影响。研究这种集体行为，既有助于深入了解软件研制过程，也有助于提高软件质量和软件生产率。目前这方面的研究尚处于初级阶段。

由于用户要求变化无常，软件研制人员普遍面临着工作效率低下，软件质量不高的局面。有些变化是不可避免的，因为用户的要求是随时提出来的。但是，软件人员和用户之间交流不畅是造成这种局面的主要原因。一个成功的软件研制过程是个协同作战的过程，在这个过程中，软件人员应当向应用领域学习，掌握用户的运行方式；反过来，用户则应当了解软件设计的现实，学会提出有效的选择。

这个过程中必然会有谈判和争辩。学习和谈判过程处理得当，是成功的关键。在软件生命期内，没有解决的设计问题占有多大百分比，是衡量软件的先进程度和预测未来故障率的良好指数。衡量标准在谈判过程的具体化和谈判协议的明确方面起重要作用。

人们往往把许多设计中的问题归咎于提供的文件不合适，事实上是主客双方交流不畅所致。提供的文件不可能完全满足设计人员的要求；他们可以通过非正式的联系网络获取信息。所以，应当做出更多的努力来促进、培育这种非正式网络。

三、软件质量的改进途径

软件设计首先需要有特定范围内各种研究对象的模型。这些模型应当尽量接近所描述的实体，而且应当具备可以控制的反馈环路，以便随时了解模型与实体之间的差异。在软件设计中，人们缺乏的就是足够的模型；而有了模型，又不十分了解这些模型与另一领域里实体之间的关系。

软件设计问题必然是多种多样的。要研究这些问题的多样性，也得有模型。这样一个过程，应当包括反馈、学习，以及环境模型的精确化等内容。

制造业部门早已学会利用产品和制造过程的模型和测量数据来控制生产的技术。它们的反馈过程，例如计划—执行—检查—修改执行这样的循环，为生产部门提供了面向质量的过程。在计划阶段，提出质量属性的指

标并确立达到这些指标的方法。在执行阶段，按照研制标准和质量政策生产出产品。在检查阶级，检查生产出来的产品质量是否达到指标。在修改执行阶段，对存在的问题提出报告，作为采取正确行动的依据，达到质量指标才能转入下一个阶段。

当然，软件研制和制造业有很大区别。但是在某一阶段还是可以采用相同的原则。软件设计需要的是能够向项目和机构本身反馈的闭环过程，这种过程要考虑到软件研制的特点。有人提出一种改造软件质量的范例方法，它有助于人们应用、开发、改编和改进软件研制过程中的各种模型和各种测量范围。

这种范例方法，也有类似于制造业部门所采用的那种计划—执行—检查—修改执行的循环。不同的是，在计划阶段要求有各种软件产品和产品质量属性的模型、过程和过程质量属性的模型，以及环境要素的模型。这些模型应当是可以量化的，而且应当提出模型的测量标准。研制人员则应当了解具体项目的要求，诸如功能、进度、造价和可靠性。另一个和制造业部门不一样的是，软件研制过程没有单一的模型，所以研制人员必须选择满足用户提出的综合质量属性的过程。

执行和检查阶段，要求跟踪已选定的过程，并采取一些测量方法判断那些模型是否合适。许多模型都比较粗糙，必须跟踪它们，看它们是否和预测的一致，如果不一致，就加以修改，使其更接近实际情况。

在修改执行阶段，要求闭环设计循环，通过反馈来修改模型和过程。这个阶段包括分析和收集某一项目上获取的经验，然后把它用于其它项目。分析工作包括事后检查反馈回来的数据，用以评价现用的模型，接着确定出存在的问题，记录下得到的结论，最后提出将来改进模型的建议。收集工作包括改进模型并把获取的知识存入经验数据库，以利以后的项目应用。这就相当于项目与项目之间传递知识的闭环机构循环系统。

强调软件的设计过程，会有助于达到软件研制的质量目标，而且有利于机构内外的技术交流。

四、促进软件设计技术的交流

促进软件技术的交流，是软件工程进一步发展和提高的必由之路。技术交流需要循序渐进的实验过程。在这个过程中，不断地积累成功的经验，改进软件研制过程和软件产品。这些经验，应当是容易掌握的，而且是可以修改的，就如同经验模型。经验模型应当归入经验数据库，可以按照新用户的要求进行访问和修改（也就是重用）。这样形成的过程模型和产品模型就有助于进行技术交流，是人们在软件研制过程中掌握的并可以应用的模型。

这就意味着，项目开发研制工作可以由项目设计机构来完成；而把系统地掌握和积累可重用的经验的工作交由所谓的经验工厂来实施。设计机构的任务是，利用一切可以利用的经验，提供用户所需要的系统。经验工厂的任务是：监视和分析项目研制工作；以知识、过程、工具和产品的形式收集可重用的经验；然后把这些经验根据需要提供给设计机构。

在这种意义上，经验工厂是逻辑结构，是物理结构，或兼而有之。经验工厂如同经验仓库，它分析和综合各种经验，需要时把这些经验提供给各种各样的项目使用。经验工厂不仅要对收集到的经验作出评估，而且还要建立软件过程、软件产品以及其它形式知识的模型和测量标准。这些工作由经验工厂通过人、文件和自动支持系统来完成。

五、推动理论与实践相结合 建立软件人员群体组织

近年来，有的产业部门中软件工作有一定规模，也受到重视，所以通过改进软件研制过程收到一定的效益。但是，总起来说，在交流和传播新的软件设计方法和软件设计工具方面成效不大。造成这种局面的部分原因是，软件设计是个过程，而不是一般产

品；同时它又是一种抽象的脑力劳动结果，不象一般产品那样看得见摸得着，在知识产权尚未完全被承认的地方，尤其难以交流。

就软件工程本身而言，它的理论与实践无论在组织机构上还是在价值观念上，都是相脱离的，从而也阻碍它的良好交流。

再从软件人员的素质看，大多数从事实际应用的，并不了解各种各样的改进方法；大多数搞理论研究的，不熟悉新技术在付诸实用之前还有许多问题有待解决。结果，有些工具和方法往往不是难学就是难用，或者既难学又难用。对于改进就是实施变革这一点，大多数人都认识到了，但是让他们去应付阻碍变革的价值观、个人动机以及其它方面的因素时，却没有多少人了。要想改变这种局面，从事理论和从事实际应用的人就得正视上述因素，而不是看看而已。

从事软件应用的，应当熟悉改进软件的需要和可能，要普遍接受新方法的训练，着重软件工具方面的全面训练。这些方法和工具应当易学易用，并且随着实际应用不断发展和完善。

要想弄清楚项目设计要求与新技术提供的机遇之间的联系，就得调查市场。而技术上的探讨可以找到解决这些设计要求的办法，还可以提供把握这些机遇的有效技术。可以说，技术研究与市场调查相互影响。

无论是从事市场研究的，还是从事技术研究的，都需要培训。不过，培训只是技术交流过程的一部分。专家们还应当举办新技术实施实验班，以此来促进理论与实践的尽快结合。

软件工具的开发，是新技术应用所必须的，同时又是技术交流的组成部分。

市场可以开通同用户联系的渠道，不断地吸收用户反馈信息，而技术部门利用这些信息来改进软件工具、培训方式、理论研究以及市场过程本身。

所有这些活动，都需要一整套熟练的技能，而这一点难于在一个人身上具备。所以，软件技术的发展和软件质量的提高，非常有必要建立知识互补的一个群体，它包括软件方面的理论研究人员、教育人员、研制人员、顾问人员和经营人员。这个群体中的成员之间应当有相互信任和良好的交往。这个群体还应当在用户中建立广泛的关系网。技术的进步不再仅仅取决于技术本身。

参考文献

[1] V.R.Basili, "The future engineering of software, A management perspective", Computer, Vol.24, No.9, 1991

（接第48页）

wledge Acquisition for Expert systems, a Practical Handbook, NewYork, Plenum Press

[9] Eshelman, L.(1988), MOLE, a knowledge acquisition tool for cover-and-differentiate system, in Automating knowledge Acquisition in Expert Systems

[10] Muson, M.A.(1989), Automated generation of model-based knowledge acquisition tools, Research Notes in AI

[11] Chandrasekaran,B.(1988),Generic tasks as building blocks for knowledge-based systems, Knowledge Engineering Review, 3

[12] Chandrasekaran, B., et al., Explaining Control Strategies in Problem Solving, IEEE Expert