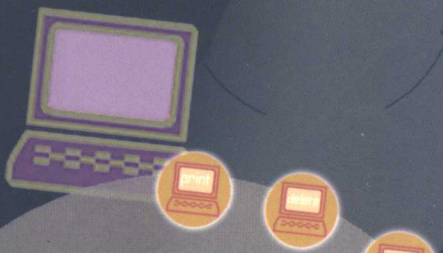
侯清富　编著



RUANJIAN GONGCHENGSHI

PEIXUN 10 JIANG

软件工程师

培训10讲

北京邮电大学出版社
www.buptpress.com

内 容 简 介

本书用于指导刚刚入职的软件工程师运用已掌握的软件工程知识，快速适应自己的岗位需要，成为一名称职的软件工程师。本书的内容包括：理解软件开发过程的要求、理解软件需求、制订设计方案、编写高质量的代码、复查和调试程序缺陷、软件质量保证、软件文档质量保证、必备的非技术技能等。

本书共分为10讲，每1讲均针对软件工程师的实际需要来解决1项基本技能。读者对象是大专院校计算机软件专业的高年级学生、软件工程师和软件项目管理者。

图书在版编目(CIP)数据

软件工程师培训10讲/侯清富编著．—北京：北京邮电大学出版社，2004

ISBN 7-5635-0926-7

Ⅰ.软… Ⅱ.侯… Ⅲ.软件开发—工程技术人员—技术培训—自学参考资料 Ⅳ.TP311.52

中国版本图书馆CIP数据核字(2004)第078745号

书　　名：软件工程师培训10讲
出版发行：北京邮电大学出版社
社　　址：北京市海淀区西土城路10号(100876)
经　　销：各地新华书店
印　　刷：北京源海印刷有限责任公司
开　　本：850 mm×1 168 mm　1/32
印　　张：5
字　　数：125千字
版　　次：2004年9月第1版　2004年9月第1次印刷
印　　数：1—5 000册
书　　号：ISBN 7-5635-0926-7/TP·118
定　　价：10.00元

如有质量问题请与北京邮电大学出版社联系　电话：(010)62282185

E-mail：publish@bupt.edu.cn　　Http://www.buptpress.com

前　言

刚刚加入软件企业的新员工，一般都掌握了程序设计语言的基本知识，也多多少少写过一些程序。如果是从计算机相关专业毕业的话，基本上都选修过软件工程的课程。但是，很多的项目经理宁愿缺人手，也不愿要这些刚从大学校园出来的新员工。根据他们的经验，刚来的新员工要适应自己的岗位，融入到项目的开发工作中，一般需要半年的时间，甚至更长。本书就是要帮助这样的新员工，花三个月或更短的时间，快速地成长为一名称职的软件工程师。

本书中的全部内容来自于编者的培训工作笔记。在近5年的时间里，编者一直领导一支40人左右的核心队伍，从事大型软件系统的设计与开发工作。在这五年期间，我们的系统与知名公司的产品相抗衡，结果取得了极大成功。在与同行竞争的过程中，也带出了一支能征善战的软件开发队伍。记得1999年刚组建软件项目团队时，编者很快发现几乎所有的新员工，尽管有学士或硕士学位，但对如何按软件工程的要求写软件却知之甚少。如果没有人告诉他们要干什么，他们就不知道该干什么。当时曾请过国内知名的软件工程专家进行培训，讲授面向对象的设计方法，软件配置管理等专业知识，但收效甚微。后来发现，是培训的内容不切实际，新员工并不缺乏书本上的知识，而是不知道如何将书本知识联系工作实际。迫于工作的实际需

要，编者和同事们坚持在战斗中学习战斗，通过自身的实践摸索实用的、可操作的书本之外的软件工程经验。

2002 年，编者主持了一个以个体软件过程为主题的内部培训，并编写了培训教材。这次培训满足了软件工程师的实际需要，深受大家的欢迎。2003 年，在原讲稿的基础上，编者对内部教材进行了结构性的调整。相关专家直接参与了新版培训提纲的修改和审订，组织了专门的评审会，重新编写了教材，增补了一些代表性的工程案例。同时，按修订后的内容，安排和组织了新一轮的内部培训。读者所看到的这本书，是在 2003 年培训的基础上，吸取了同行专家的意见后编撰而成的。编者认为这些内容已经经受了实践的检验，可以传播到更广阔的范围中去，供软件业的同行参考。

本书的重点不是写软件工程，而是指导新员工运用已掌握的软件工程知识，快速适应自己的岗位需要，成为一名称职的软件工程师。它强调通俗易懂、可实际操作，能学以致用。它的目的在于帮助新员工了解项目开发中所面对的任务和困难，并学会运用已掌握的知识，完成这些任务，克服面临的困难。编者过去五年多的实践，已经使上百人从中受益，为本书的实用价值提供了一个很好的佐证。

编者

2004 年 8 月

目　录

1 软件过程与质量控制

从事软件设计的工程师，多多少少都选修过软件工程的课程。本书的10讲培训内容，并不是重复软件工程书本的内容，而是认为读者已经具备了软件工程的基础知识。经过本书的培训，读者能够快速适应软件项目开发的实际需要，具备从事软件产品开发的基本技能，包括适应软件开发过程中的要求，编写、讨论软件需求和设计方案文档，编写有质量保证的代码，复查和调试代码缺陷以及软件工程师必备的非技术技能，等等。本书的每1讲都是针对软件工程师实际需要的1项基本技能。

本讲是概论，论述完成一个软件项目，要做哪些重要的事情。做这些事情的时候，有一些什么样的问题要解决。在解决这些问题时，有一些什么样的办法。

本讲分为6个主题。第1个主题讲解做软件与写程序之间的关系。写得出好程序不一定能做出好软件，因为软件开发的基础是软件过程，而不是个人技巧。第2个主题讲解软件过程的作用。接着的两个主题讲解两个非常实用的软件开发过程，一个是“瀑布式软件过程”；另一个是“增量式软件过程”。对于经验不多的软件工程师，主要是承担软件模块的设计、编码、调试与测试等任务。那么，在完成这些任务时，软件过程是如何体现的呢？第5个主题主要讲解软件过程中的“作业指导书”。第

6个主题是“软件过程中的质量控制”，运用一些简单实用的手段，有效地保证所做的工作符合质量要求。

1.1 会编码不等于能做软件

从软件过程看，程序编码是做软件的一个环节，不是做软件的全部。做软件除了编码之外，还有需求分析、方案设计、系统联调、系统测试、版本发行、质量控制、计划和进度安排等重要事务。软件设计新手在学校基本上都编过代码，但会编码并不意味着会做软件。

从工程技术的角度来理解，我们可以做一个简单的比较。

从目的来看，编码是为了干什么呢？毫无疑问，编码是在有一个设计方案之后，让计算机能理解设计者的方案，满足设计者的要求，得到所预期的结果。换一句话说，编码的目的就是让计算机能理解和执行设计者的意图。做软件和编码不一样，做软件的目的是满足软件用户的需求。例如，用户要求将自己的局域网与因特网连起来，软件需求就是做一个通信网关将两个网络连起来。用户需求可能很多，例如这个软件不能停机，能24小时工作；要有流量控制功能，能控制流量拥塞；还要有计费的功能，等等。做软件除了满足用户的需求外，还要满足开发团队利益的需要。没有开发团队会无限制地、不计代价地开发一套软件，保证开发团队有利可图也是做软件的目的之一。

做软件和编码所要解决的问题不一样。编码是以编程语言（如C、C++、JAVA等）和配套的编译器为工具，解决计算机理解设计者意图的问题。编码所针对的问题，是人与计算机对话的问题，编码语言是人与机器对话用的语言，所编写的程序是对话的内容。做软件所针对的问题，是满足人的需求，软件系统是

满足需求的中介。例如，电话网络中的通信软件，就是要满足人与人之间远距离沟通的需求，它所要解决的问题是话音的接收、转换和传送等。

做软件和编码所追求的目标不同。做软件在满足用户需求的同时，要通过实施软件过程，保证按时交付软件、控制质量和降低成本。项目延期意味着财力和物力的投入有超出预算的风险，质量没能达到要求必然导致软件维护困难和增加投入成本，有效地控制成本才能获得更多的收益。编码追求的目标是程序能完整地、正确地理解和执行设计者的意图。在实际的工作中，让程序能正确地工作比较容易，困难的是做到完整地符合设计者的意图。

从取得成效的基础来说，做软件取得成功，关键靠有适合团队特点的软件过程。因为一个人的能力毕竟有限，团队的动作要靠软件过程来协调和统一。相对来说，编码主要靠个人的经验和技巧，个人的因素起主导性的作用。

1.2 软件过程的作用

软件开发需要正规的软件过程，因为软件开发不是写程序，而是一个凝结智慧和创造力的过程。

对于一个人能完成的小项目，个人才华有可能使软件开发获得成功，但实际成功的例子也不多。

对于要一个团队（不少于4人）完成的项目，是整个团队的工作方式对软件开发起着决定性的作用。对一个项目来说，团队力量不是个人力量的简单相加，而是在一起工作的方式所构成的整体的力量。既可能是“三个臭皮匠，赛过诸葛亮”，也可能是“三个和尚没水喝”。开发团队的规模越大，软件过程的影响

力就越大。对团队中的个人来说，他所起的作用主要取决于自身适应软件过程的程度。如果适应得好，他的个人能力就会对软件开发起到帮助作用；如果适应得不好，反而会起到阻碍作用。很多软件项目经理，宁愿其领导的项目缺人手，也不愿接纳新来的软件工程师，这就是一个例证。

软件过程对软件开发有着非常重要的作用。如果不知道自己究竟要开发些什么，就不可能得到一个较好的设计。如果在开发过程中，被迫对需求分析和设计方案进行修改，则代码也必须作修改，这就势必引起返工重做，使软件产品无法按时交付给用户。可能有人会反驳说："在现实中，没有真正一成不变的需求，所以软件过程只是分散人的注意力而已"。可实际中所采用的软件过程决定着需求的稳定性。如果需要保持软件需求有更大的灵活性，可以按计划分阶段交付软件，而不是一次性交付完毕，那么这就是软件过程所关注的问题了。无数的事实清楚地表明，改正需求和分析错误，要比改正设计或代码错误费事得多。软件过程最终决定了项目的成功或失败。

以上原则对方案设计同样适用。如果在完成方案设计之前，就急着编码的话，以后再对结构进行重大的修改就很困难了。所有的软件工程师都钟爱自己编写的代码，除投入了大量的时间和精力，还倾注了很多的感情。很少有软件工程师乐意为新的设计去修改代码，而是坚持自己编写的代码不动，要求别人去修改设计来适应已完成的代码。一旦已经开始添砖加瓦了，要想再修改工程图纸就会困难重重，软件开发也是如此。

靠个人的英雄行为来做软件，常出的问题可概括为四个方面：一是软件的质量得不到保证；二是开发进度得不到保证；三是团队成员难以进行交流和协作，四是软件开发工作交接和后续维护无据可依。以上的任何一个方面，都能轻易地从实际项目中找到很多的例子来说明。

作者曾经领导一个开发项目，团队中有一个软件高手，称之为高手云吧。大家都知道他最聪明，得过很多的程序设计竞赛大奖。大家都认为拥有他是上帝的恩宠。没有人怀疑他的天才与判断力，尽管他没有项目开发经验，却被奉为团队中的精英人物。

当高手云开始最有挑战性的工作时，他回房关上门，此时陪伴他的只有重金属音乐、《黄河》的旋律、……、以及6箱果汁和可乐，一切就绪后，他开始工作了。

其他组员都对他充满信心，非常庆幸拥有他这样的天才，可以承担这么艰难的任务。在门外，我们听到他快速敲击键盘的声音日夜不停，大家心中都在微笑，尽管没有人知道在关上的门内是多么伟大的程序，但我们都觉得有高手云的参与，真是大家的福气。

他像所有的天才人物一样，特别是在项目最繁忙的时候，总感觉白天的干扰使他过于分心无法工作，而要等到晚上夜深人静时才能写程序。

白天他去睡觉，去选唱片，去健身……。当别人白天下班时，他开始回到办公室。

……

慢慢地，我们开始觉得他遇到了困难。他渐渐地不回家了，音乐也停止了，斗大的汗珠出现在他的额头，他显然是遇到了麻烦。

整个团队只有他的工作不接受检查，他只要在完成时交代码就行了。当其他人都已经如期完成份内的工作时，我们不得不小心地问他："云，做得怎么样了？"里面一阵迟疑，之后听到他说："还不错。""什么时候可以完成呢？"经过更长时间的停顿，我们几乎听到他哽咽着回答："就快了。"

我们知道他说很快，其实还早得很。现在全开发团队的人

都只能呆呆地站着干等，一点也帮不上忙。作为项目经理的我，面临着困难的选择：可以开除他，但考虑到他是惟一能帮助我们的人，是惟一真正了解当前困难所在的人。所以大家不假思索地决定，让他继续完成他的工作，他毕竟是团队中最优秀的人才，也是团队中惟一能够掌握这个问题的人。

我们惟一能做的，只有替他买更多的可乐。

当时，我想：是否给他加点压力，让他清楚自己身系团队的危急存亡，所有组员都指望他不凡的才华和飞快的手指，得催逼他做出个像样的东西来。但稍作三思，还是觉得这也不是个好办法，对生产力没有帮助。我们没有办法摆脱这个困境，我们只能往好的方面想：他会完成他的工作的，只要他没有死或离职。经过这次教训，他大概再也不会为我们工作了。现在我们惟一能做的事就是帮他买更多的可乐，其他什么办法都没有。

在这个项目中，笔者真实地体验了一位前辈所讲的一段话：允许一位开发人员关在房间里而没有人知道他的工作进度，这常常是开发团队的致命伤。将团队的命运维系在某位所谓的英雄身上，就像拿项目的前景去摸彩票中奖一样，99.99%的时候难逃失败。

1.3 瀑布式软件过程

在介绍软件工程的书中，都会讲到瀑布式软件过程。尽管瀑布式软件过程已不是很实用，但它有助于理解和掌握做软件的规律。

在过去的十多年中，专家们指出了做软件所要涉及的主要工作，大致上分为七个环节。

第一个环节是需求分析。主要是了解用户的需求，最终要

落实到一份或一套需求文件中。在文件中描述用户所要的软件系统是什么样子。

第二个环节是结构设计。通过结构设计，提出一个解决方案，来满足用户的需求。

第三个环节是详细设计。对于需要组建一个团队来开发的软件系统，其可划分成子系统，子系统又可划分成模块。详细设计就是定义模块中的接口、数据结构和算法。

第四个环节是编码和调试。

第五个环节是单元测试。确保所做的模块是可以用的，基本上没有错误，或没有很容易发现的错误。

第六个环节是系统测试。两个模块都没有错误，但它们合起来并不能肯定没有问题。通过系统测试，软件就可以交付给用户使用了。

第七个环节是软件维护。用户在使用软件后，仍会发现软件有各种各样的问题。一种可能是软件中的错误在测试中没被发现，但在用户使用过程中出现了。另一种可能是需求发生了变化，用户又有了新的需求。软件维护要求及时、有效地处理和排除错误，满足新的需求。

瀑布式软件过程理解起来比较简单，但却不便于在实际工作中操作和执行。原因有多个方面，一个主要原因是瀑布式软件过程有一个假设，就是前一个环节的工作全部完成之后，才开始第二个环节的工作。但这样的假设与实际情况并不完全相符。以需求分析为例，做需求分析是很困难的，很难说得清楚需求分析的结果是否已经没有问题了。如果等到需求分析没有问题了，才开始结构设计，那就不知何时才能开始。现实的情况是，需求分析中总会存在各种各样的问题，甚至是很严重的问题，如果按瀑布式过程去操作的话，是根本不可能的。

在做实际的商业软件时，没有哪个开发团队会完全按照瀑

布式软件过程去做。基本上可以说，瀑布式软件过程是不可管理的，无法有效地控制项目开发的质量和进度。

针对瀑布式软件过程的局限性，人们提出了增量式软件过程。

1.4 增量式软件过程

在增量式软件过程中，将瀑布式软件过程的前六个环节连结成一个封闭的环形。这个封闭环的意思是，在开始需求分析时，尽管不能把握好全部的需求，但总是可以把握好其中的一部分。例如，要开发一个通信网关软件系统，尽管开始时不太可能把计费的需求搞得很清楚，但把两个网络连通起来的需求还是比较容易把握好的。针对已经把握好的这一部分需求，去做结构设计、详细设计、编写代码、做好调试，最终将系统交付给用户使用。用户在使用过程中，发现网络连通没有问题了，但发现有时网络的速度不快，忙时流量拥塞比较厉害。这时即可进入第二次循环，分析需要加进一些什么样的功能，保证网络 24 小时通畅，不会出现流量拥塞。循环一次，就是增量一次。通过一次循环，解决一部分问题，满足一部分用户需求，再循环一次，再解决一部分新的问题，满足用户新的需求。增量式软件开发的用户需求是逐步增加的。

增量式软件过程是 20 世纪 70 年代早期由 Mills(软件工程之父)提出的。Mills 提出，任何软件系统都应由增量式开发来增长。也就是说，系统首先能运行，即使它无任何实用功能，除了调用一系列伪子系统。接着，系统一点一点地被充实，子系统轮流被开发，或者是在更低的层次调用程序、模块、伪程序(Stub)等。

与瀑布式软件过程相比，增量式软件过程表现出很多工程方面的优点：

一是有效地控制进度。因为要实施的需求已经是把握好的，因而能比较准确地估计开发进度，而不是像瀑布式过程那样猜测开发进度。

二是能有效地吸纳用户的反馈。如果没有一个系统供用户使用，用户就会比较随便地提出自己的需求，因为他们自己也并不十分清楚需要一个什么样的软件系统。但是当有一个系统给用户使用时，用户提出的需求会更有针对性，描述起来也准确得多。

三是有效地控制质量。采用增量式过程进行项目开发能方便地进行统计质量管理和变更管理。在瀑布式软件过程中，这一点是做不到的或很难做到的。

在开发大型软件系统时，增量式软件过程的优点会体现得更明显。因而，增量式软件过程受到工程界的喜爱和重视。在《人月神话》(布鲁克思，2002)中，作者对增量式软件过程给予了极高的评价。其中特别提到，增量式软件过程对开发士气的推动有非常直接的作用。当有一个可运行的系统，即使是一个非常简单的系统出现时，开发人员的热情就迸发出来了。当一个新图形软件的第一幅图案出现在屏幕上时，即使是一个简单的长方形，工作的动力也会成倍地增长。在开发过程的每一个阶段，总有可运行的系统，即使在短短的4个月内，采用增量式开发的效果也是非常明显的。

1.5 软件过程的具体体现

软件过程最终会落实到一系列具体的技术规定、质量要求和其他规章制度等等上。其中与技术开发工作直接相关的制度

性文件，就是作业指导书了。简单地说，作业指导书是规定软件过程中某些子过程的执行步骤。例如，“软件编码作业指导书”会重点规定软件编码的作业流程，就编码前的准备工作、编码进行中的注意事项以及编码完成后的质量验收等内容作出指导性的要求。

可能大家有一个顾虑，就是按作业指导书来做事情的话，会极大地限制做事情的自由。所以，很多时候员工不太愿意按照作业指导书来做，往往视作业指导书为一个负担，而不是前进道路上的一根拐杖。

可以从两个方面来认识这个问题。一个方面的原因是，作业指导书本身的可操作性不够好，所以操作起来不方便，常常碍手碍脚。另一个方面的原因是，有些做事情的好方法自己还没有体会到。既然没有完全地理解和体会，自然也就不太愿意接受作业指导书所要求的做法。

个人如何对待这些作业指导书呢？给大家的一个建议：尽管感觉作业指导书不太好用，仍要坚持按作业指导书来做手头的工作。为什么呢？因为写一份作业指导书并不容易，它要求归纳以往的工作经验，特别是吸取犯过错误的教训。如果抛开作业指导书，就很容易犯以前别人犯过的错误，这是不值得的。反反复复地犯别人或自己犯过的错误，这样的现象在软件开发中并不罕见。作业指导书能够帮助我们避免这一点。

作业指导书是保证开发质量的措施。事实上，执行作业指导书的一个重要出发点，就是要保证软件开发的质量。质量管理的一个基本思想是，在做任何一件事情时，如果落实了所要求的质量管理要素，质量就能得到保证。作业指导书已将质量管理要素与具体的技术操作结合起来形成了作业步骤。如果不按作业指导书操作，就意味着某些质量要素将得不到落实，也就无法保证质量。